Storage Foundation Cluster File System High Availability 7.4.2 管理者ガイド - Linux

VERITAS™

最終更新: 2020-07-15

法的通知

Veritas Technologies LLC
2625 Augustine Drive
Santa Clara, CA 95054
http://www.veritas.com

テクニカルサポート

テクニカルサポートはグローバルにサポートセンターを管理しています。すべてのサポートサービスは、サポート契約と現在のエンタープライズテクニカルサポートポリシーに応じて提供されます。サポート内容およびテクニカルサポートの利用方法に関する情報については、次の Web サイトにアクセスしてください。
https://www.veritas.com/support

Veritas Account 情報は、次の URL で管理できます。
https://my.veritas.com

現在のサポート契約についてご不明な点がある場合は、次に示すお住まいの地域のサポート契約管理チームに電子メールでお問い合わせください。

世界中 (日本以外)	CustomerCare@veritas.com
日本	CustomerCare_Japan@veritas.com

マニュアル

マニュアルの最新バージョンがあることを確認してください。各マニュアルには、2 ページ目に最終更新日が記載されています。最新のマニュアルは、ベリタスの Web サイトで入手できます。
https://sort.veritas.com/documents

マニュアルに関するご意見やご感想

ご意見、ご感想をお待ちしています。改善すべき点や、マニュアル上の誤記、欠落がありましたらお寄せください。お送りいただく際は、マニュアルの題名とバージョン、章のタイトル、セクションのタイトルを明記してください。フィードバックの送信先:
infoscaledocs@veritas.com

ベリタスのコミュニティサイトで、マニュアルに関する情報を確認したり、質問を投稿することもできます。
http://www.veritas.com/community/

Veritas Services and Operations Readiness Tools (SORT)

Veritas SORT (Services and Operations Readiness Tools) は、時間のかかる特定の管理タスクを自動化および単純化するための情報とツールを提供する Web サイトです。製品に応じて、SORT はインストールとアップグレードの準備、データセンターのリスクの識別、効率性の改善に役立ちます。使用している製品に対して SORT が提供しているサービスおよびツールについては、次のデータシートを参照してください。
https://sort.veritas.com/data/support/SORT_Data_Sheet.pdf

目次

1

Storage Foundation Cluster File System High Availability の紹介

1

Storage Foundation Cluster File System High Availability の概要

この章では以下の項目について説明しています。

Storage Foundation Cluster File System High Availability について

SFCFSHA（Storage Foundation Cluster File System High Availability）は、管理と拡張が可能で堅牢なストレージ配備を可能にするストレージ管理ソリューションです。SFCFSHA は、異機種混在のサーバーとストレージのプラットフォーム全体においてストレージの効率性、可用性、アジリティ、パフォーマンスを最大化します。SFCFSHA は、ストレージエリアネットワーク環境で共有データをサポートするために、Storage Foundation を拡張します。SFCFSHA を使うと、複数のサーバーがアプリケーションに対して透過的

に共有ストレージとファイルに同時にアクセスできます。SFCFSHA は可用性とパフォーマンスの高められた自動化とインテリジェントな管理も提供します。

Storage Foundation Cluster File System High Availability は、個別または連携して使うことのできる製品コンポーネントと機能で構成され、パフォーマンスと耐障害性を向上し、ストレージやアプリケーションの管理を容易にします。

表 1-1 では、Storage Foundation Cluster File System High Availability のコンポーネントを示します。

表 1-1 Storage Foundation Cluster File System High Availability コンポーネント

コンポーネント	説明
DMP（Dynamic Multi-Pathing）	システムで設定されている物理ストレージデバイスの I/O パフォーマンスとパスの可用性を管理します。 DMP は、各 LUN へのすべてのパスに対して DMP メタデバイスを作成します。DMP は、この DMP メタデバイスを使って、物理デバイスのパス全体でパスフェールオーバーと I/O 負荷分散を管理します。 DMP メタデバイスは、VxVM（Veritas Volume Manager）と VxFS（Veritas File System）の基盤を提供します。DMP は、DMP デバイスでネーティブオペレーティングシステムのボリュームとファイルシステムもサポートします。
Veritas Volume Manager（VxVM）	オペレーティングシステムのデバイスとアプリケーション間で論理ストレージの抽象化レイヤーまたはストレージを管理します。 VxVM では、物理ディスクや LUN 上にボリュームという論理デバイスを作成できます。ファイルシステムまたはデータベースなどのアプリケーションは、ボリュームにアクセスするときに、それがあたかも物理デバイスであるように扱い、物理的な制限なしでアクセスできます。 VxVM の機能を使えば、データ可用性を妨げることなくオンラインでストレージ I/O パフォーマンスを設定、共有、管理、最適化できます。追加の VxVM 機能は、耐障害性およびディスク障害またはストレージアレイ障害からの迅速なリカバリを拡張します。

コンポーネント	説明
Cluster Volume Manager（CVM）	1 つのクラスタでの複数のシステムでの使用に向けて VxVM 論理ボリューム層を拡張します。 各システムまたはクラスタノードは、同一の論理デバイスまたはボリュームへのアクセスを共有します。各ノードは同一の論理ストレージを同一の状態で認識します。 CVM は、ストライプ化、ミラー化、スナップショットの作成といったパフォーマンスを向上させる VxVM 機能をサポートします。クラスタ内の 1 つのノードから標準の VxVM コマンドを使って共有ストレージを管理できます。他のすべてのノードは、ユーザーの操作がなくてもディスクグループとボリューム設定の変更をすぐに認識します。
Veritas File System（VxFS）	高性能なジャーナルファイルシステムを提供します。 VxFS は、大容量のデータを処理し、優れたパフォーマンスと高い可用性を必要とする操作環境にとって最適なファイルシステムです。 VxFS の機能は、アプリケーションの迅速なリカバリ、拡張性のあるパフォーマンス、連続的な可用性、増加された I/O スループットと構造整合性を提供します。
Cluster File System（CFS）	1 つのクラスタの複数のシステム（またはノード）での使用に向けて VxFS ファイルシステムを拡張します。 CFS では、同一のファイルシステムを複数のノードに同時にマウントできます。 CFS は、管理の簡素化、パフォーマンスの向上、アプリケーションとデータベースの高速フェールオーバーを実現します。
Cluster Server（VCS）	高可用性の機能を提供します。 VCS は、クラスタノードの障害の監視と通知機能を提供します。VCS は、コンポーネント層の起動とシャットダウンを制御し、サービスのフェールオーバーを別のノードに適用します。
Replicator（VR）	ディザスタリカバリのために、1 つ以上のリモートの場所で一貫性のあるアプリケーションデータのコピーを保持できます。 Replicator は、VVR（Volume Replicator）を使ったブロックベースの継続的なレプリケーションと VFR（File Replicator）を使ったファイルベースの定期的なレプリケーションによって生まれる柔軟性を提供します。Replicator オプションは、Storage Foundation Cluster File System High Availability の別途にライセンス可能な機能です。
I/O フェンシング	潜在的なスプリットブレイン状態を示すネットワーククラスタメンバーシップの変更をクラスタ内のノードが検出した場合に、共有ディスクのデータを保護します。

関連製品、Veritas Operations Manager には、Veritas InfoScale 製品で使うことができる集中型管理コンソールが用意されています。

p.40 の「Veritas InfoScale Operations Manager について」を参照してください。

メモ: このマニュアル内で RHEL (Red Hat Enterprise Linux) オペレーティングシステムについて使用するコマンドは、サポート対象の RHEL 互換配布にも適用されます。

DMP (Dynamic Multi-Pathing) について

Dynamic Multi-Pathing (DMP) は、システム上で設定されているオペレーティングシステムのネイティブデバイスに対するマルチパス機能を提供します。DMP は DMP メタデバイス (既知の DMP ノード) を作成して、同じ物理 LUN へのデバイスパスをすべて示します。

DMP は Storage Foundation Cluster File System High Availability のコンポーネントとして使うことができます。DMP は、DMP メタデバイス上の Veritas Volume Manager (VxVM) ボリュームと、これらボリューム上の Veritas File System (VxFS) ファイルシステムをサポートします。

DMP メタデバイスは OS ネイティブの論理ボリュームマネージャ(LVM)をサポートします。DMP メタデバイス上に LVM ボリュームとボリュームグループを作成できます。

Veritas Volume Manager (VxVM) ボリュームとディスクグループは LVM ボリュームとボリュームグループと共存させることができます。ただし、各デバイスでサポートできる形式は 1 つだけです。ディスクに VxVM ラベルが付いている場合、そのディスクは LVM で利用できません。同様に、ディスクが LVM によって使用中の場合、そのディスクは VxVM で利用できません。

Veritas Volume Manager の概要

Veritas 社の Veritas™ Volume Manager (VxVM)は、物理ディスクや論理ユニット番号(LUN)をボリュームと呼ばれる論理デバイスとして管理できるようにする、ストレージ管理サブシステムです。 VxVM ボリュームは、ファイルシステム、データベース、その他の管理対象データオブジェクトの設定ができる物理デバイスとして、アプリケーションとオペレーティングシステムに表示されます。

VxVM は、コンピュータ環境とストレージエリアネットワーク(SAN: Storage Area Network)環境で、使いやすいオンラインディスクストレージ管理を実現します。RAID(Redundant Array of Independent Disks)をサポートすることにより、VxVM はディスクやハードウェアの障害からシステムを保護し、I/O スループットを高めるように設定できます。 さらに VxVM には、耐障害性およびディスク障害またはストレージアレイ障害からの迅速なリカバリを拡張する機能があります。

VxVM では、論理ボリューム管理レイヤーによって、ハードウェアディスクデバイスや LUN からの制約を克服できます。このため、ボリュームを複数のディスクと LUN に分散できます。

VxVM は、パフォーマンスを高め、データの可用性と整合性を確保するツールを提供しています。また、VxVM を使って、システムがアクティブなときにストレージを動的に設定できます。

Veritas File System について

ファイルシステムとは、コンピュータファイルとそれに含まれるデータを保存、整理する単純な手法で、検索とアクセスを容易にします。形式的に言えば、ファイルシステムとは、データのストレージ、階層編成、操作、ナビゲーション、アクセス、取り込みなどが実装された抽象データ型（メタデータなど）の一式です。

Veritas File System（VxFS）は、商用では初めてのジャーナルファイルシステムです。ジャーナルを使うと、メタデータの変更はまずログ（またはジャーナル）に書き込まれ、次にディスクに書き込まれます。変更を複数の場所に書き込む必要がなく、メタデータは非同期で書き込まれるため、スループットが大幅に速くなります。

また、VxFS は、エクステントを管理単位としたインテントログファイルシステムです。VxFS は、優れたパフォーマンスと高い可用性、そして大容量のデータ処理能力を必要とする操作環境にとって最適なファイルシステムです。

作成できるファイルシステムの最大サイズは、ブロックサイズによって異なります。

ブロックサイズ	現在サポートされているファイルシステムの最大サイズ
1024 バイト	68,719,472,624 セクタ（≈32 TB）
2048 バイト	137,438,945,248 セクタ（≈64 TB）
4096 バイト	274,877,890,496 セクタ（≈128 TB）
8192 バイト	549,755,780,992 セクタ（≈256 TB）

VxFS の主要コンポーネントは次のとおりです。

ファイルシステムのログ	「Veritas File System のインテントログについて」
エクステント	「エクステントについて」
ファイルシステムのディスクレイアウト	「ファイルシステムのディスクレイアウトについて」

Veritas File System のインテントログについて

通常のファイルシステムはシステム障害からリカバリする際、そのための唯一の手段である fsck ユーティリティによるファイルシステム構造全体の検証に依存しています。ディスク構成の規模が大きい場合には、このユーティリティによる構造全体の調査、ファイルシステムの整合性の検証および不整合部分の修正の処理には、必然的に相当の時間が必要になります。VxFS は、VxFS インテントログと VxFS インテントログのサイズ変更機能で高速リカバリを実現します。

VxFS では、VxFS インテントログでファイルシステムの動作を追跡することによって、システム障害からのリカバリ時間が短縮されます。この機能は、ファイルシステム構造に加えられた変更のうち保留中になっているものを、循環式のインテントログに記録します。システム障害の場合を除き、ユーザーやシステム管理者がインテントログによるリカバリ機能に気付くことはありません。デフォルト設定時の VxFS ファイルシステムでは、ファイルのトランザクションをディスクにコミットする前にログに記録し、ファイルシステムが予想に反して停止した場合にファイルシステムのリカバリに要する時間を短縮させます。

システム障害からのリカバリ中、VxFS の fsck ユーティリティはインテントログの再生（インテントログのスキャン）を実行し、システム障害時に実行されていたファイルシステム操作を取り消すか、または完了させます。そのため、ファイルシステムの構造全体を検査しなくてもファイルシステムをマウントできる状態になります。ディスクにハードウェア障害があると、インテントログを再生しても損傷を受けたファイルシステム構造のリカバリを完了できない場合があります。そのような場合は、VxFS の fsck ユーティリティを使って、ファイルシステムの構造全体の検査を完了させる必要があります。

メモ: VxFS ファイルシステムイメージと VxFS ソフトウェアリリースとの互換性は、主にディスクレイアウトバージョン (DLV) によって決定します。しかし、ファイルシステムイメージにリカバリが必要な場合 (システムクラッシュが発生した場合、またはファイルシステムイメージがアレイレベルのスナップショットで作成された場合) は、そのソフトウェアリリースには追加の制限事項があります。このような場合、リカバリの実行に使用するソフトウェアリリースは、該当のファイルシステムが直近でマウントされたときのソフトウェアリリースと少なくとも同じかそれよりも新しい必要があります。

---

mount コマンドは、ファイルシステムでダーティログを検出すると、VxFS の fsck コマンドを自動的に実行してインテントログの再生を実行します。この機能は、Veritas Volume Manager（VxVM）ボリュームにマウントされたファイルシステムでのみサポートされ、クラスタファイルシステムでサポートされます。

fsck_vxfs（1M）マニュアルページとmount_vxfs（1M）マニュアルページを参照してください。

VxFS インテントログは、ファイルシステムを最初に作成する際に割り当てられます。インテントログのサイズは、ファイルシステムのサイズによって異なり、ファイルシステムが大きくなるほど、インテントログのサイズも大きくなります。fsadm コマンドを使ってインテントログのサイズを後から調整できます。

fsadm_vxfs (1M) のマニュアルページを参照してください。

ディスクレイアウトバージョン 7 以降のインテントログの最大デフォルトサイズは 256 MB です。

メモ: インテントログを不適切なサイズに設定すると、システムの処理効率に悪影響を及ぼす可能性があります。

p.261 の「インテントログサイズ」を参照してください。

エクステントについて

エクステントは、コンピュータファイルシステム内の連続したストレージ領域で、ファイル用に予約された領域です。ファイルに対する書き込みを開始すると、エクステント全体が割り当てられます。ファイルに再び書き込みを行うと、前回の書き込み場所に続けてデータが書き込まれます。これにより、ファイルの断片化を少なくするかまたは回避できます。エクステントは「アドレスと長さの組み合わせ」で表現され、これによって開始ブロックのアドレスと、(ファイルシステムまたは論理ブロックの)エクステントの長さが決まります。Veritas File System (VxFS) は、エクステントベースのファイルシステムのため、エクステント(複数ブロックで構成可能)を使ってアドレス指定します。単一ブロックセグメントにアドレス指定するのではありません。したがって、エクステントを使うとファイルシステムのスループットを向上できます。

エクステントを使った場合、格納領域に連続するブロックが割り当てられると、複数のブロック単位でディスク I/O が可能になります。順次 I/O の場合、複数ブロック単位での処理の方が 1 ブロック単位のものよりもかなり高速になり、ほぼすべてのディスクドライブで複数ブロックの I/O 操作を実行できます。

エクステント単位の割り当ての場合、i ノード構造体からアドレスブロック情報を解釈する方法が、ブロック単位の割り当ての場合と比較して多少異なります。VxFS の i ノードは、10 個のエクステントを直接参照し、各エクステントは開始ブロックアドレスとブロック単位での長さの組み合わせで表現されています。

ディスク領域としては 512 バイトのセクタが割り当てられ、このセクタから論理ブロックが構成されます。VxFS では、1024、2048、4096、8192 バイトの論理ブロックサイズがサポートされています。デフォルトのブロックサイズは、2 TB 未満のファイルシステムに対しては 1 KB、2 TB 以上のファイルシステムに対しては 8 KB です。

ファイルシステムのディスクレイアウトについて

ディスクレイアウトは、ファイルシステムの情報がディスクに保存される方法です。VxFS (Veritas File System) では、新機能や特定の UNIX 環境を提供するため、さまざまなディスクレイアウトバージョンがサポートされています。

ディスクレイアウトバージョンをアップグレードするには、次のいずれかのコマンドを使用できます。

`vxupgrade`	ファイルシステムがオンラインのときに、既存の VxFS ファイルシステムを、サポート対象のディスクレイアウトバージョンにアップグレードします。 `vxupgrade`(1M) マニュアルページを参照してください。
`vxfsconvert`	ファイルシステムがマウントされていないときに、サポート対象外のディスクレイアウトバージョンを、サポート対象のバージョンにアップグレードします。 `vxfsconvert` コマンドは、ファイルシステムがマウントされていないときにネイティブファイルシステム (ext2、ext3、および ext4) を VxFS に変換する場合にも使用できます。 `vxfsconvert`(1M) マニュアルページを参照してください。

表 1-2は、サポート対象のディスクレイアウトバージョンを一覧表示します。

表 1-2　サポート対象のディスクレイアウトバージョン

バージョン	サポート対象機能
バージョン 12	バージョン 12 は CFS で 128 ノードクラスタをサポートします。
バージョン 13	■ WORM のサポートの追加 ■ 拡張ファイル属性使用時のクローン作成パフォーマンスの向上
バージョン 14	SmartIO FEL ベースのキャッシュをサポートします。
Version 15	■ SELinux 属性のストレージと取得のパフォーマンスの向上 ■ WORM ファイルの安全なクロックサポート
Version 16	WORM ファイルの監査ログのサポート

現在作成およびマウントできるのは、バージョン 12、13、14、15、16 のみです。バージョン 6、7、8、9、10、11 はマウントできますが、サポートされているバージョンにアップグレードする場合のみです。

Storage Foundation Cluster File System（SFCFS）について

1 つのクラスタの複数のシステム（またはノード）での使用に向けて VxFS ファイルシステムを拡張します。CFS では、同一のファイルシステムを複数のノードに同時にマウントで

きます。CFS は、管理の簡素化、パフォーマンスの向上、アプリケーションとデータベースの高速フェールオーバーを実現します。

クラスタファイルシステムでサポートされている Veritas File System 機能について

Storage Foundation Cluster File System High Availability は Veritas File System (VxFS) に基づきます。

VxFS ローカルファイルシステムのほとんどの主要機能をクラスタファイルシステムで利用できます。次の機能が含まれます。

- 最大 1 TB のサイズのファイルをマップするエクステントベースの領域管理
- インテントログを使ってファイルシステムメタデータの最近の更新を追跡することによるシステムクラッシュからの高速リカバリ
- ファイルシステムが使用中でもその拡張や断片化解消が可能なオンライン管理

VxFS のすべてのマニュアルページには、Storage Foundation Cluster File System の問題についてのセクションがあります。このセクションでは、クラスタマウントされたファイルシステムでコマンドが機能するかどうかについての情報が記載されています。また、ローカルマウントされたファイルシステムの場合との動作の違いを示しています。

クラスタファイルシステムでサポートされていない Veritas File System 機能

p.39 の 表 1-3 を参照してください。 はクラスタファイルシステムでサポートされない機能を一覧表示します。一覧表示された機能は使用できますが、正しく動作する保証はありません。

サポートされていない機能を SFCFSHA で使わないようにしてください。また、マウントしているファイルシステムを、ローカルマウントやクラスタマウントとしてこれらのオプションで置き換えないようにしてください。

表 1-3　クラスタファイルシステムでサポートされていない Veritas File System の機能

qlog	Quick Log はサポートしていません。
スワップファイル	スワップファイルはクラスタマウントされたファイルシステムでサポートされません。
mknod	クラスタマウントされたファイルシステムで、mknod コマンドを使ってデバイスを作成することはできません。

キャッシュアドバイザリ	キャッシュアドバイザリは個別のファイルシステム上で mount コマンドによって設定されますが、クラスタの他のノードには伝達されません。
Cached Quick I/O	ファイルシステムのキャッシュにデータをキャッシュする、この Quick I/O for Database 機能はサポートされていません。
ファイルアクセス時間に依存するコマンド	クラスタファイルシステムでは atime ファイル属性が厳密に同期されないため、ファイルアクセス時間の表示がノード間で異なる場合があります。そのため、アクセス時間のチェックに依存するユーティリティは確実に機能しない場合があります。

Veritas InfoScale Operations Manager について

Veritas InfoScale Operations Manager には、Veritas InfoScale 製品の集中型管理コンソールが用意されています。Veritas InfoScale Operations Manager を使って、ストレージリソースを監視、視覚化、管理したり、レポートを生成したりすることができます。

Veritas InfoScale Operations Manager を使って Storage Foundation と Cluster Server の環境を管理することをお勧めします。

Veritas InfoScale Operations Manager は https://sort.veritas.com/ からダウンロードできます。

インストール、アップグレード、設定の手順について詳しくは、Veritas InfoScale Operations Manager のマニュアルを参照してください。

Veritas Enterprise Administrator（VEA）のコンソールは Veritas InfoScale 製品に含まれなくなりました。VEA の使用を続けたい場合は、https://www.veritas.com/product/storage-management/infoscale-operations-manager からソフトウェアバージョンをダウンロードできます。Storage Foundation Management Server は非推奨です。

Cluster Manager（Java コンソール）を使ってシングルクラスタを管理したい場合は、https://www.veritas.com/product/storage-management/infoscale-operations-manager からダウンロードできます。Java コンソールを使ってこのリリースの新しい機能を管理することはできません。Cluster Server Management Console は非推奨です。

Veritas Replicator について

Veritas Replicator は、異種データレプリケーションの包括的なソリューションを組織に提供します。Storage Foundation に対する 1 つのオプションとして、Veritas Replicator はコスト効率の高いデータレプリケーションを IP ネットワーク上で有効にし、従来のアレイベースのレプリケーションアーキテクチャに代わって非常に柔軟なストレージハードウェ

ア非依存型のアーキテクチャを組織に提供します。Veritas Replicator は、VVR（Volume Replicator Option）を使ったブロックベースの継続的なレプリケーションと VFR （File Replicator Option）を使ったファイルベースの定期的なレプリケーションによって生まれる柔軟性を提供します。

VFR とは

VFR（Veritas File Replicator）は、コスト効率の高い定期的なデータレプリケーションを IP ネットワーク上で有効にし、ディザスタリカバリとオフホスト処理に対して非常に柔軟なストレージ非依存型のデータ可用性ソリューションを組織に提供します。ビジネスニーズに応じてレプリケーション間隔をスケジュール設定できる柔軟性により、Veritas File Replicator はファイルシステムに対するすべての更新を追跡し、設定された時間間隔で定期的にこれらの更新をレプリケートします。VFR は、ネットワークリソースが乏しい場合にレプリケーションへの影響を軽減するために、VxFS（Veritas File System）によって提供されるデータ重複排除機能を活用します。VFR は、Linux 上の Virtual Store 6.0 にデフォルトで含まれ、Linux 上の Storage Foundation と関連製品ではオプションとして使用可能です。

VFR の機能

Veritas File Replicator（VFR）には、次の機能が用意されています。

- 単一のファイルからファイルシステム全体まで及ぶファイルシステムのサブセットの定期的なレプリケーションをサポートします。
- 可逆データ転送をサポートします。レプリケーションのターゲットは、実行時にソースになることがあります（以前のソースシステムがターゲットになる）。
- 最後に正常にレプリケートされたポイントインタイムイメージからの自動リカバリをサポートします。
- 定期的に変更をレプリケートします。間隔は、ユーザーが設定できます。
- ターゲットシステムにおけるストレージの効率を高めるために、重複排除をサポートします。
- 偶発的な書き込みに対するターゲットファイルシステムの保護をサポートします。
- IPv4、IPv6、デュアルスタック構成をサポートします。

詳しくは、『Storage Foundation and High Availability Solutions Replication 管理者ガイド』を参照してください。

Storage Foundation Cluster File System High Availability の使用例

Storage Foundation Cluster File System High Availability のコンポーネントと機能は別々に使用できます。連携して使用することでパフォーマンスと耐障害性が向上し、ストレージやアプリケーションの管理が容易になります。Storage Foundation Cluster File System High Availability には次のような機能があります。

- データベースパフォーマンスの向上: Storage Foundation Cluster File System High Availability のデータベースアクセラレータを使った I/O のパフォーマンス向上。SFHA Solutions のデータベースアクセラレータを使用することで、ファイルシステムの管理機能と便利さを享受すると同時に RAW ディスクが高速化されます。
- シンストレージを設定、保守するために Storage Foundation Cluster File System High Availability のシンプロビジョニングとシン再生のソリューションを使用することによる、シンアレイの使用状況の最適化。
- データベースのバックアップとリカバリ: Storage Foundation Cluster File System High Availability Flashsnap、Storage Checkpoint、NetBackup の PITC の使用による、データのバックアップとリカバリ。
- データのオフホスト処理: Storage Foundation Cluster File System High Availability ボリュームスナップショットを使用した、生産ホストのパフォーマンスロスを防ぐための、データのオフホスト処理。
- テストと開発環境の最適化: Storage Foundation Cluster File System High Availability の PITC を使用した、テスト、決定モデリング、開発を目的とする実稼動データベースのコピーを使用することによる、テストと開発環境の最適化。
- 仮想デスクトップ環境の最適化: Storage Foundation Cluster File System High Availability FileSnap を使用した仮想デスクトップ環境の最適化。
- ストレージ使用の最大化: 経過時間、優先度、アクセス率の基準に基づくストレージ階層にデータを移動するために、Storage Foundation Cluster File System High Availability SmartTier を使用した、ストレージ使用の最大化。
- ストレージ使用の最大化: 物理的な共有ストレージを使わずに、データ冗長性、高可用性、およびディザスタリカバリに Storage Foundation Cluster File System High Availability Flexible Storage Sharing を使ったストレージ使用の最大化。
- データの移行: Storage Foundation Cluster File System High Availability Portable Data Containers を使用した、ある環境から別の環境への簡単かつ確実なデータの移行。

シナリオ例を使って Storage Foundation Cluster File System High Availability の使用例について文書化した補足マニュアル:『Veritas InfoScale ソリューションガイド』を参照してください。

2

Dynamic Multi-Pathing の動作

この章では以下の項目について説明しています。

DMP の動作方法

DMP（Dynamic Multi-Pathing）では、パスフェールオーバー機能と負荷分散機能を使って、可用性、信頼性、パフォーマンスを向上します。これらの機能は、さまざまなベンダーのマルチポートディスクアレイに対応しています。

ディスクアレイは、複数のパスを介して、ホストシステムに接続することができます。ディスクへのさまざまなパスを検出するために、DMP では、対応している各アレイに特有の機構を使います。また、DMP では、DMP に対応していて同じホストシステムに接続されているアレイの様々なエンクロージャを識別します。

p.291 の「新しく追加されたディスクデバイスの検出と設定」を参照してください。

DMP で使われるマルチパスポリシーは、ディスクアレイの特性によって異なります。

DMP では、次の標準アレイタイプをサポートします。

表 2-1

アレイタイプ	説明
アクティブ/アクティブ（A/A）	複数のパスを同時に使って I/O を行うことができます。また DMP により、I/O 負荷が LUN への複数のパス上に均等に分散されるので I/O スループットが向上します。1 つのパスが失われた場合、DMP は自動的に、そのアレイに対して使える他のパスを介して I/O を行います。
非対称アクティブ/アクティブ（A/A-A）	A/A-A または非対称アクティブ/アクティブアレイは、パフォーマンスをほとんど低下させずにセカンダリストレージパスからアクセスできます。動作は、ALUA のアレイがサポートする SCSI コマンドをサポートしない点以外は ALUA と同じです。
非対称論理ユニットアクセス（ALUA）	DMP は ALUA のすべてのバリアントをサポートします。
アクティブ/パッシブ（A/P）	通常の操作中に 1 つのコントローラ（アクセスポートまたはストレージプロセッサ）上のプライマリ（アクティブ）パス経由で LUN（論理ユニット番号。ハードウェアを使って作成される実際のディスクまたは論理ディスク）へのアクセスが可能です。 非明示的フェールオーバーモード（auto-trespass モード）では、プライマリパスに障害が発生した場合、別のコントローラ上のセカンダリ（パッシブ）パスに I/O をスケジューリングすることによって、A/P アレイが自動的にフェールオーバーします。このパッシブポートは、アクティブポートに障害が発生するまで I/O には使われません。A/P アレイでは、プライマリパスで I/O 障害が発生すると、単一の LUN でパスのフェールオーバーが実行されます。 このアレイモードは、複数のプライマリパスを 1 つのコントローラに持つことで、同時 I/O と負荷分散をサポートします。この機能は、複数のポートを持つコントローラにより、またはアレイとコントローラ間に SAN スイッチを挿入することによって、提供されます。セカンダリ（パッシブ）パスへのフェールオーバーは、すべてのアクティブなプライマリパスに障害が発生した場合にのみ実行されます。

アレイタイプ	説明
明示的フェールオーバーモードまたは非 auto-trespass モードのアクティブ/パッシブ（A/PF）	LUN のセカンダリパスへのフェールオーバーを実行するには、該当するコマンドをアレイに発行する必要があります。 このアレイモードは、複数のプライマリパスを 1 つのコントローラに持つことで、同時 I/O と負荷分散をサポートします。この機能は、複数のポートを持つコントローラにより、またはアレイとコントローラ間に SAN スイッチを挿入することによって、提供されます。セカンダリ（パッシブ）パスへのフェールオーバーは、すべてのアクティブなプライマリパスに障害が発生した場合にのみ実行されます。
LUN グループフェールオーバーが設定されたアクティブ/パッシブ（A/PG）	LUN グループフェールオーバーが設定されたアクティブ/パッシブアレイ（A/PG アレイ）の場合、1 つのコントローラを介して接続されている LUN のグループは単一のフェールオーバーエンティティとして扱われます。A/P アレイの場合と異なり、フェールオーバーは個々の LUN レベルではなくコントローラレベルで実行されます。プライマリコントローラとセカンダリコントローラは、それぞれ別の LUN グループに接続されます。プライマリコントローラの LUN グループ内の LUN の 1 つに障害が発生した場合、そのグループ内のすべての LUN に対して、セカンダリコントローラへのフェールオーバーが実行されます。 このアレイモードは、複数のプライマリパスを 1 つのコントローラに持つことで、同時 I/O と負荷分散をサポートします。この機能は、複数のポートを持つコントローラにより、またはアレイとコントローラ間に SAN スイッチを挿入することによって、提供されます。セカンダリ（パッシブ）パスへのフェールオーバーは、すべてのアクティブなプライマリパスに障害が発生した場合にのみ実行されます。

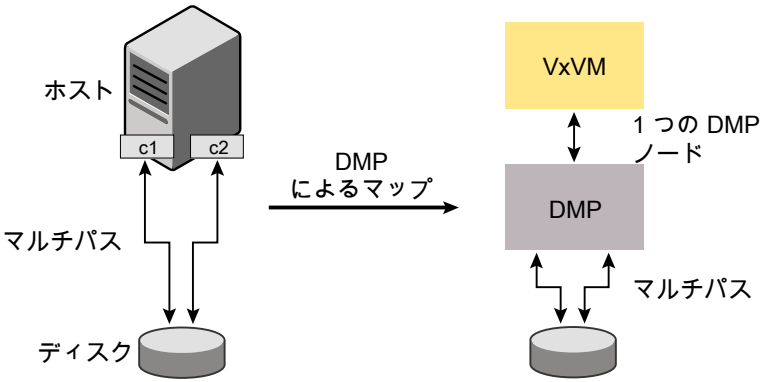
アレイポリシーモジュール（APM）では、DMP がサポートする標準タイプ以外のアレイタイプを DMP に定義できます。

Storage Foundation Cluster File System High Availability は、DMP メタノード（DMP ノード）を使って、システムに接続されているディスクデバイスにアクセスします。DMP に対応しているアレイ内のディスクの場合は、DMP により各ディスクに接続するパスセットに 1 つのノードがマップされます。さらに、DMP によりそのディスクアレイに適合するマルチパスポリシーがノードに関連付けられます。

DMP に対応していないアレイ内のディスクの場合は、DMP によりディスクに接続するパスそれぞれに、個別のノードがマップされます。ノードの raw デバイスおよびブロックデバイスは、ディレクトリ /dev/vx/rdmp および /dev/vx/dmp にそれぞれ作成されます。

図 2-1 では、DMP によりサポートされているディスクアレイ内のディスクにどのようにノードが設定されるかについて説明しています。

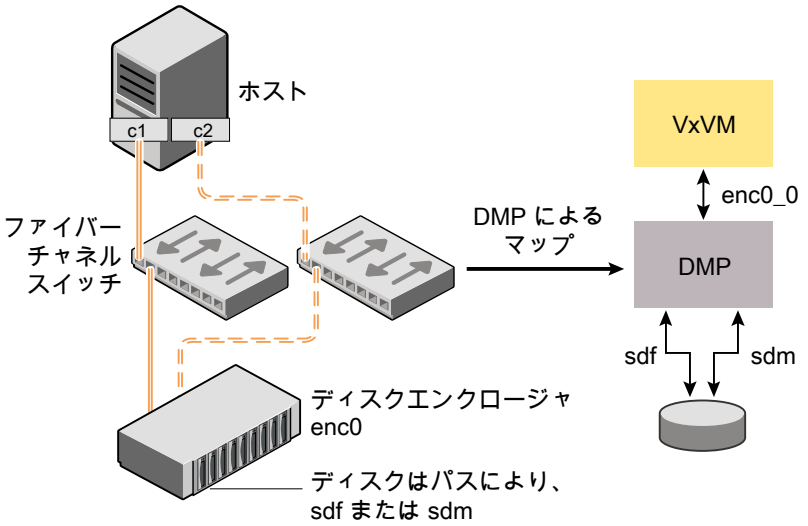
図 2-1 DMP で、ディスクに対する複数の物理パスを 1 つのノードで表す方法



DMP では、ディスクが属するアレイを識別することができるディスクデバイス名前の付け方を導入しました。

図 2-2には、エンクロージャ内の 1 つのディスクに 2 つのパス (sdf と sdm) が存在し、VxVM が 1 つの DMP ノード (enc0_0) を使ってディスクにアクセスする例が示されています。

図 2-2 SAN 環境における、ディスクエンクロージャに対するマルチパスの例



p.47 の「エンクロージャに基づく名前の付け方について」を参照してください。

p.382 の「ディスクデバイスの名前の付け方の変更」を参照してください。

p.291 の「新しく追加されたディスクデバイスの検出と設定」を参照してください。

デバイス検出

デバイス検出は、ホストに接続されているディスクを検出するプロセスを示すために使う用語です。この機能は DMP にとって重要です。DMP では多くのベンダーにより増加し続けるディスクアレイをサポートする必要があるためです。ホストに接続されているデバイスを検出する機能とともに、デバイス検出サービスでは、新しいディスクアレイのサポートを追加できます。デバイス検出はデバイス検出層（DDL）と呼ばれる機能を使います。

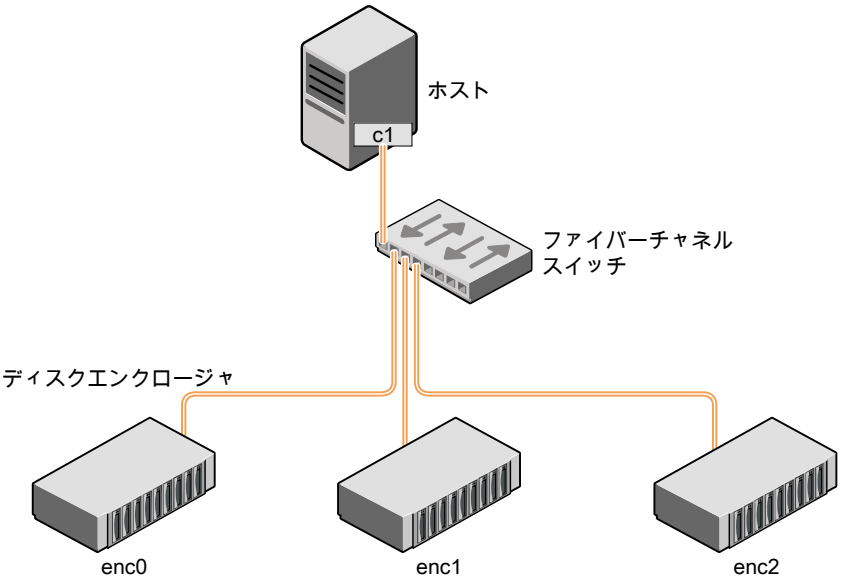
DDL により、再ブートすることなく新しいディスクアレイのサポートを追加できます。

エンクロージャに基づく名前の付け方について

オペレーティングシステムに基づくデバイスの名前の付け方の代わりに、エンクロージャに基づく名前の付け方を使うことができます。ファイバーチャネルスイッチを使うストレージエリアネットワーク（SAN）では、オペレーティングシステムからのディスクの配置情報で、ディスクの物理的位置が正しく示されない場合があります。エンクロージャに基づく名前の付け方では、SFCFSHA はエンクロージャに個々の物理エンティティとしてアクセスできます。個々のエンクロージャに、データの冗長性のあるコピーを設定することで、1 つ以上のエンクロージャの障害に対抗できます。

図 2-3 に、ホストコントローラがファイバーチャネルスイッチを使って複数のエンクロージャに接続されている通常の SAN 環境を示します。

図 2-3　ファイバーチャネルスイッチで接続されているディスクエンクロージャの設定例



このような設定では、エンクロージャに基づく命名を使って、エンクロージャ内の各ディスクを示すことができます。たとえば、エンクロージャ `enc0` 内のディスクのデバイス名は `enc0_0`、`enc0_1` のように設定されています。この規則の主な利点は、大規模な SAN 設定でディスクの物理的位置を迅速に特定できることです。

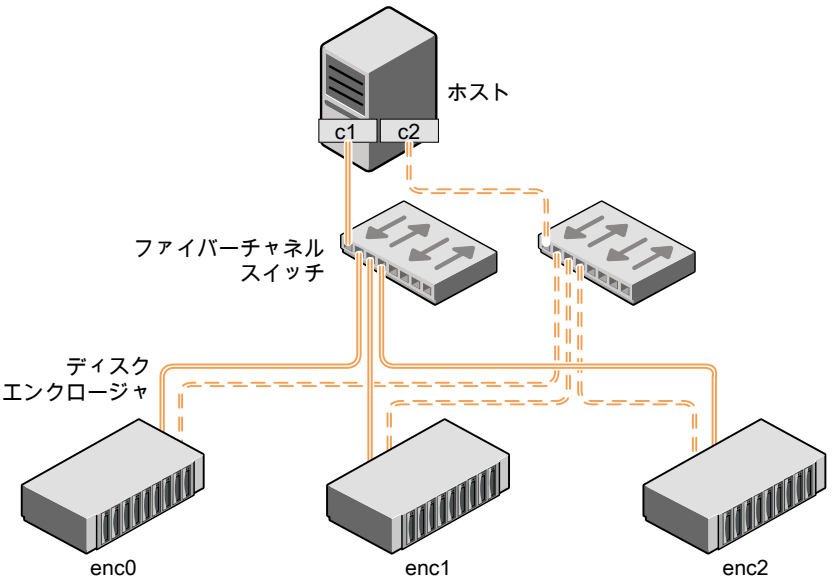
ほとんどのディスクアレイでは、ハードウェアベースのストレージ管理を使って、複数の物理ディスク 1 つの LUN としてオペレーティングシステムに提示できます。このような場合、VxVM でもコンポーネントディスクではなく、1 つの論理ディスクデバイスを認識します。このため、エンクロージャ内のディスクにリファレンスを作成する場合、ディスクは物理ディスクまたは LUN になります。

エンクロージャに基づく命名によるもう 1 つの重要な利点は、VxVM がデータの冗長なコピーを同じエンクロージャに配置するのを回避できることです。各エンクロージャは独立した障害のあるドメインと認識される可能性があるため、そのような配置を回避するのは好ましいことです。たとえば、ミラー化したボリュームがエンクロージャ `enc1` のディスク上にのみ設定された場合は、スイッチとエンクロージャ間のケーブル障害により、ボリューム全体が利用できなくなることがあります。

必要に応じて、SFCFSHA がエンクロージャに割り当てたデフォルト名を自分の設定に意味のある名前に交換することができます。

図 2-4 に、ホスト上の独立したコントローラをエンクロージャへの独立したパスを持つ個々のスイッチに接続して、ストレージへの冗長ループアクセスを実現する高可用性(HA)設定を示します。

図 2-4　冗長ループアクセスを実行するために複数のスイッチを使った HA の設定例



このような設定により、ホストコントローラ(c1 と c2)のいずれかに障害が発生したり、ホストといずれかのスイッチをつなぐケーブルに障害が発生しても可用性を維持することができます。この例では、VxVM がアクセスできるすべてのパスで、各ディスクは同じ名前になっています。たとえば、ディスクデバイス enc0_0 は 1 つのディスクを表しますが、オペレーティングシステムには 2 つの異なるパス、sdf と sdm が認識されています。

p.382 の「ディスクデバイスの名前の付け方の変更」を参照してください。

データの冗長性を設定するとき、ドメインに障害が発生することを考慮するために、ミラー化したボリュームを、エンクロージャをまたがってレイアウトする方法を制御できます。

パスでの I/O を DMP で監視する方法

リリース 5.0 より前の VxVM には、エラー処理を実行するカーネルデーモン(errord)と、パスリストアアクティビティを実行するカーネルデーモン(restored)がありました。

リリース 5.0 からは、DMP が、エラー処理、パスリストア、統計情報収集、SCSI 要求コールバックなどのタスクに使うカーネルスレッドのプールを保守します。restored の名前は、下位互換性のために引き続き維持されています。

1 つのカーネルスレッドは、パスで I/O エラーが発生すると、そのパスに対応する HBA の精査を開始することで応答します。続いて、別のスレッドが HBA からの応答に従って適切な処理を行います。適用した処理は、そのパスでの I/O 要求を再試行することも、そのパスを破棄して代替パスで I/O をスケジュール設定し直すこともできます。

リストアカーネルタスクは定期的(デフォルトでは 5 分間隔)に起動して、パスの健全性を調べ、リストアされたパスで I/O を再開します。パスによっては断続的にエラーが起きることがあるため、パスが一定期間(デフォルトでは 5 分)健全であり続けた場合にのみ、このパスで I/O を再開します。DMP ではパスのチェックに異なるポリシーを設定できます。

p.351 の「DMP パスリストアポリシーの設定」を参照してください。

統計情報収集タスクは、各 I/O 要求の開始時間と終了時間、各パスでの I/O エラー数と再試行回数を記録します。この情報を使って、I/O 要求によって SCSI ドライバのフラッドが起きないように、DMP を設定できます。この機能を I/O 調整と呼びます。

I/O 要求がミラーボリュームに関連する場合、VxVM は、FAILFAST フラグを指定します。この場合、DMP はエラーになった I/O 要求をそのパス上で再試行する代わりに、エラーが起きたというマークをパス上のディスクに付けます。

p.50 の「パスフェールオーバー機構」を参照してください。

p.51 の「I/O 調整」を参照してください。

パスフェールオーバー機構

複数パスを持つディスクアレイで DMP を使うと、システムの可用性が向上します。ディスクアレイへのパスの 1 つが失われた場合、DMP では、管理者が介入しなくても、I/O 要求に対して次に使用可能なパスが自動的に選択されます。

また、接続が修復または復元されたり、OS が正しくデバイスを認識している場合にシステムが完全に起動した後にデバイスの追加や削除が行われると、DMP に通知されます。

必要に応じて、パスでの I/O エラーに対する DMP の応答は、個々のアレイへのパスごとに調整できます。DMP では、I/O 要求が成功することなく一定時間が経過した場合、またはパスでの一定回数の再試行が失敗した場合に、I/O 要求を時間切れにするように設定できます。

p.347 の「I/O エラーに対する応答の設定」を参照してください。

サブパスフェールオーバーグループ(SFG)

サブパスフェールオーバーグループ(SFG)は、まとめて失敗およびリストアできるパスのグループを表します。SFG のパスで I/O エラーが発生した場合、DMP は SFG のパス以外に、その他のパスでプロアクティブなパスのプローブを行います。この動作により、パスのフェールオーバーのパフォーマンスが大幅に向上し、結果として I/O パフォーマンスが向上します。サブパスフェールオーバーグループを形成するために DMP が現在従っている基準は、ホストからアレイまで同じエンドポイントを持つパスを、1 つの論理的なストレージフェールオーバーグループにまとめることです。

p.349 の「サブパスフェールオーバーグループ（SFG）の設定」を参照してください。

LIPP（Low-Impact Path Probing）

DMP のリストアデーモンは、LUN パスを定期的にプローブし続けます。この動作は、パスで I/O が発生しない場合でも、DMP がパスを最新の状態に保つのに役立ちます。パスの状態がリストアデーモンによって更新されている間に実行されるプローブの数を最適化するため、LIPP（Low-Impact Path Probing）はリストアデーモンにロジックを追加します。この最適化は、論理的なサブパスのフェールオーバーグループを使うことで実現されます。LIPP のロジックが導入された DMP は、サブパスフェールオーバーグループ（SFG）内のすべてのパスをプローブする代わりに、SFG 内の限られた数のパスのみをプローブします。これらのプローブの結果に基づいて、DMP はその SFG 内のすべてのパスの状態を判断します。

p.349 の「LIPP（Low-Impact Path Probing）の設定」を参照してください。

I/O 調整

I/O 調整を有効にし、応答動作が低下したパスでの未処理の I/O 要求数が増加した場合、未処理の I/O 要求数が一定値に達したとき、またはそのパスで最後に I/O 要求が成功してから一定時間が経過したときに、新しい I/O 要求をそのパスに送らないように DMP を設定できます。調整がパスに適用されると、そのパスでの新しい I/O 要求は、別の使用可能なパス上にスケジュール設定されます。この調整は、パスにエラーがないと HBA から報告があった場合、またはパスでの未処理の I/O 要求が成功した場合に、パスから削除されます。

p.348 の「I/O 調整機構の設定」を参照してください。

負荷分散

デフォルトでは、DMP は最小キュー I/O ポリシーを使ってすべてのアレイタイプのパス全体に負荷を分散します。負荷分散が行われると、使用可能なパスすべての総帯域幅を使って、I/O スループットが最大化されます。I/O は未処理の I/O が最小のパスを使って送信されます。

アクティブ/パッシブ（A/P）のディスクアレイ場合は、I/O はプライマリパスで送信されます。すべてのプライマリパスに障害が発生した場合、I/O は使用可能なセカンダリパスでの送信に切り替えられます。あるコントローラから別のコントローラへ連続して LUN 制御が移動し I/O 処理が極端に遅くなると、A/P ディスクアレイに対するプライマリ/セカンダリパスの負荷分散は、複数 I/O の同時処理をサポートしていないかぎり行われません。

その他のアレイでは、負荷分散は現在アクティブなパスのすべてで実行されます。

エンクロージャやディスクアレイへのパスに適用する I/O ポリシーは変更できます。この操作はサーバーに影響しない、ダウンタイムを必要としないオンライン操作です。

クラスタ環境における DMP

A/P（アクティブ/パッシブ）タイプのディスクアレイを複数のホストで共有するクラスタ環境では、クラスタ内のすべてのノードが同一の物理ストレージコントローラポート経由でディスクにアクセスする必要があります。ディスクへのアクセスに複数のパスを同時に使うと、I/O パフォーマンスが大幅に低下します（ピンポン効果とも呼ばれます）。単一のクラスタノードでパスフェールオーバーが発生した場合でも、すべてのノードが継続して同一の物理パスを共有できるように、クラスタ全体が調整されます。

VxVM 4.1 より前のリリースでは、クラスタ化と DMP 機能は、A/P アレイでパスがリストアされたときに自動フェールバックを処理できず、明示的フェールオーバーモードアレイのフェールバックもサポートしていませんでした。フェールバックは、パス障害が修復された後に、各クラスタノードで `vxdctl enable` コマンドを実行することにより、手動で実行する必要がありました。リリース 4.1 からは、フェールバックは、マスターノードによって調整され、クラスタ全体で自動的に実行されるようになりました。明示的フェールオーバーモードアレイの自動フェールバックも、適切な下位コマンドを実行することにより処理できます。

メモ: A/P アレイの自動フェールバックのサポートには、システムへの適切な ASL（Array Support Library）のインストールが必要です。APM（Array Policy Module）も必要になることがあります。

---

p.293 の「ディスクの検出とディスクアレイの動的な追加について」を参照してください。

アクティブ/アクティブタイプのディスクアレイの場合、すべてのディスクは、接続されたすべての物理パスを通じて同時にアクセスできます。クラスタ環境では、ノードが同じ物理パスでディスクにアクセスする必要はありません。

p.295 の「デバイス検出層の管理方法」を参照してください。

p.353 の「アレイポリシーモジュール（Array Policy Modules）の設定」を参照してください。

共有ディスクグループでのコントローラの有効化と無効化について

VxVM（Veritas Volume Manager）5.0 より前のリリースでは、共有 Veritas Volume Manager ディスクグループの一部であるディスクに接続されているパスまたはコントローラを有効または無効にすることはできませんでした。VxVM 5.0 以降では、クラスタ内の共有 DMP ノードで、これらの操作をサポートします。

Veritas Volume Manager と Oracle ASM ディスクの共存

ASM（Automatic Storage Management）ディスクは、Oracle Automatic Storage Management ソフトウェアで使用されるディスクです。Veritas Volume Manager（VxVM）

と Oracle ASM ディスクは、Oracle ASM ディスクを Oracle ASM タイプのディスクとして認識することで共存します。VxVM は ASM ディスクが上書きされないように、上書きする可能性がある操作から保護します。VxVM は、ASM ディスクを ASM 形式のディスクとして分類、表示します。ASM ディスクは初期化できません。また、ASM ディスクを上書きする可能性がある VxVM 操作を実行することはできません。

ディスクが ASM ディスクとして認識されると、ディスクの初期化コマンドは失敗し、適切なエラーメッセージが表示されます。 force オプションが指定されてる場合でも、`vxdisk init` コマンドと `vxdisksetup` コマンドは失敗します。ASM デバイスのディスクが変更されるを防ぐため、ASM の制御下にあるディスクに対する `vxprivutil` コマンドも失敗します。

ターゲットディスクが ASM 制御下にある場合、ターゲットディスクを上書きするルータビリティ操作はすべて失敗します。メッセージが表示され、ディスクがすでに ASM ディスクとして使用されていることを示します。ルータビリティ操作には、VM のルートイメージを作成する作成する操作（`vxcp_lvmroot` コマンド）、VM のルートミラー（`vxrootmir` コマンド）、LVM のルートイメージをリストアする操作（`vxres_lvmroot` コマンド）が含まれます。ASM ディスクに対する `vxdestroy_lvmroot` コマンドも失敗します。ターゲットディスクが、想定されている LVM 制御下にないためです。

ASM が以前にアクセスしたが、現在は ASM ディスクグループに属していないディスクを FORMER ASM ディスクと呼びます。ASM ディスクを ASM 制御下から削除すると、VxVM はそのディスクを FORMER ASM ディスクとしてラベル付けします。ASM ディスクに対するこの制約は、FORMER ASM ディスクに対しても同じように実施されます。将来、ASM を有効にすることでディスクを再利用できます。FORMER ASM ディスクを VxVM で使用するには、ディスクを ASM 制御から削除した後で、ASM ディスクの情報を消去する必要があります。FORMER ASM ディスクに初期化コマンドを実行すると、コマンドは失敗します。メッセージが表示され、VxVM で使用するには初期化の前にディスクをクリーンアップする必要があることを示します。

VxVM で使用するために、FORMER ASM ディスクを ASM 制御から削除するには

1. `dd` コマンドでディスクをクリーンアップし、ディスク上の ASM 識別情報を削除します。次に例を示します。

   ```
   dd if=/dev/zero of=/dev/rdsk/<wholedisk|partition> count=1 bs=1024
   ```

   ここで、*wholedisk* は、`cxtydz` の形式のディスク名です。

   ここで *partition* は、`cxtydzsn` の形式のパーティション名です。

2. ディスクスキャンを実行します。

   ```
   # vxdisk scandisks
   ```

ASM ディスクを表示するには

◆ ASM ディスクを表示するには、次のいずれかのコマンドを使用します。

vxdisk list コマンドは、ディスクタイプ ASM として表示します。

```
# vxdisk list
DEVICE       TYPE            DISK         GROUP        STATUS
Disk_0s2     auto:LVM        -            -            LVM
Disk_1       auto:ASM        -            -            ASM
EVA4K6K0_0   auto            -            -            online
EVA4K6K0_1   auto            -            -            online
```

特定のディスクが ASM 制御下にあるかどうか調べるには

◆ 特定のディスクが ASM 制御下にあるかどうか調べるには vxmediadisc ユーティリティを使用します。

```
# /etc/vx/diag.d/vxmediadisc 3pardata0_2798
 3pardata0_2799   ACTIVE
```

または、ユーティリティを使用して、ディスクが LVM や ASM などの外部ソフトウェアの制御下にあるかどうか調べることもできます。

```
# /etc/vx/bin/vxisforeign 3pardata0_2799
 3pardata0_2799   ASM    ACTIVE

# /etc/vx/bin/vxisforeign 3pardata0_2798
 3pardata0_2798   ASM    FORMER
```

3

Veritas Volume Manager の動作

この章では以下の項目について説明しています。

Veritas Volume Manager のオペレーティングシステムでの動作

Veritas Volume Manager（VxVM）は、オペレーティングシステムと、ファイルシステムやデータベース管理システムなどのデータ管理システム間のサブシステムとして動作します。VxVM はオペレーティングシステムと強固に結合されます。ディスクまたは LUN を VxVM の制御下に置く以前に、ディスクがオペレーティングシステムデバイスのインターフェースを介してアクセスできる必要があります。VxVM は、オペレーティングシステムインターフェースサービスの最上部層に置かれ、オペレーティングシステムが物理ディスクにアクセスする方法に依存します。

VxVM の次の機能は、オペレーティングシステムに依存します

- オペレーティングシステム（ディスク）デバイス
- デバイスの処理
- VxVM Dynamic Multi-Pathing（DMP）メタデバイス

VxVM は、次の常時起動しているデーモンとカーネルスレッドに依存して動作します。

`vxconfigd`
: VxVM 設定デーモンは、ディスクとグループ設定を管理し、設定変更をカーネルに伝達し、ディスクに保存されている設定情報を変更します。

  `vxconfigd`（1M）マニュアルページを参照してください。

`vxiod`
: VxVM 入出力カーネルスレッドは、プロセスの呼び出しをブロックすることなく入出力処理を拡張できます。デフォルトでは、16 の入出力スレッドが起動時に開始し、少なくとも 1 つの入出力スレッドが常時実行している必要があります。

  `vxiod`（1m）のマニュアルページを参照してください。

`vxrelocd`
: ホットリロケーションデーモンは、冗長性に影響するイベントについて VxVM を監視し、ホットリロケーションを実行して冗長性を復元します。システム内にシンプロビジョニングディスクが設定されている場合、削除されたボリュームのストレージ領域は、ポリシーの設定どおりにこのデーモンによって再生利用されます。

  `vxrelocd`（1M）のマニュアルページを参照してください。

データの保存方法

物理ディスクにデータを保存するには、いくつかの方法が使われます。これらの方法では、データが効率よく保存され検索されるようにディスク上のデータを編成します。ディスク編成の基本的方法をフォーマットと呼びます。フォーマットによって、設定済みのストレー

ジパターンを使ってファイルをディスクに書き込んだり、ディスクから検索できるように、ハードディスクを整備します。

フォーマットされたハードディスクに情報を保存するには、物理ストレージレイアウトと論理的ストレージレイアウトの 2 つの方法が使われます。VxVM では、論理ストレージレイアウト方法を使います。

p.57 の「Veritas Volume Manager でストレージ管理を処理するには」を参照してください。

Veritas Volume Manager でストレージ管理を処理するには

Volume Manager（VxVM）では、次の 2 種類のオブジェクトを使ってストレージ管理を実現します。

物理オブジェクト
: 物理ディスクの LUN（ハードウェアに実装された仮想ディスク）またはデータ保存用のブロックデバイスインターフェースと raw デバイスインターフェースを備えた他のハードウェアを指します。

p.57 の「物理オブジェクト」を参照してください。

仮想オブジェクト
: 1 つ以上の物理ディスクが VxVM の制御下に置かれると、これらの物理ディスク上にボリュームと呼ばれる仮想オブジェクトが作成されます。各ボリュームは 1 つ以上の物理ディスクからデータを記録し、検索します。ファイルシステム、データベース、その他のアプリケーションは、物理ディスクと同じ方法で、ボリュームにアクセスします。ボリュームは、ボリューム設定の変更に使う他の仮想オブジェクト（プレックスとサブディスク）からも構成されます。ボリュームとその仮想コンポーネントは、仮想オブジェクトまたは VxVM オブジェクトと呼ばれます。

p.59 の「仮想オブジェクト」を参照してください。

物理オブジェクト

物理ディスクは、データが最終的に格納される基本的なストレージデバイス（メディア）です。デバイス名を使ってディスクを検索することで、物理ディスク上のデータにアクセスできます。物理ディスクのデバイス名は、使用するコンピュータシステムによって変わります。すべてのシステムですべてのパラメータが使われるわけではありません。

標準のデバイス名は、`sda` か `hdb` の形式です。`sda` は最初（`a`）の SCSI ディスクを参照し、`hdb` は 2 番目（`b`）の EIDEのディスクを参照します。

図 3-1 は物理ディスクおよびデバイス名（*devname*）が VxVM（Veritas Volume Manager）マニュアルでどのように説明されるか示します。

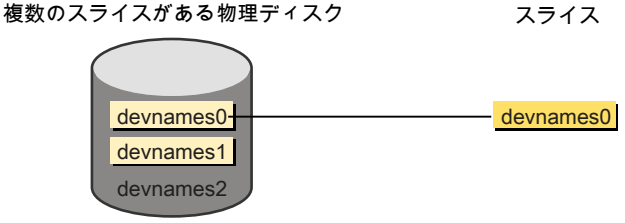
図 3-1　　物理ディスクの例

VxVM は VxVM 制御下（VM ディスク）の物理ディスクの識別情報を書き込みます。VxVM ディスクは物理ディスクの切断またはシステムの停止の後であっても識別されます。その後、VxVM は障害検出を提供し、システム回復を促進するためにディスクグループおよび論理オブジェクトを再形成できます。

ディスクパーティションについて

図 3-2 に、物理ディスクを 1 つ以上のパーティションに分割する方法を示します。

図 3-2　　パーティションの例



パーティション番号はデバイス名（*devname*）の最後に追加されます。

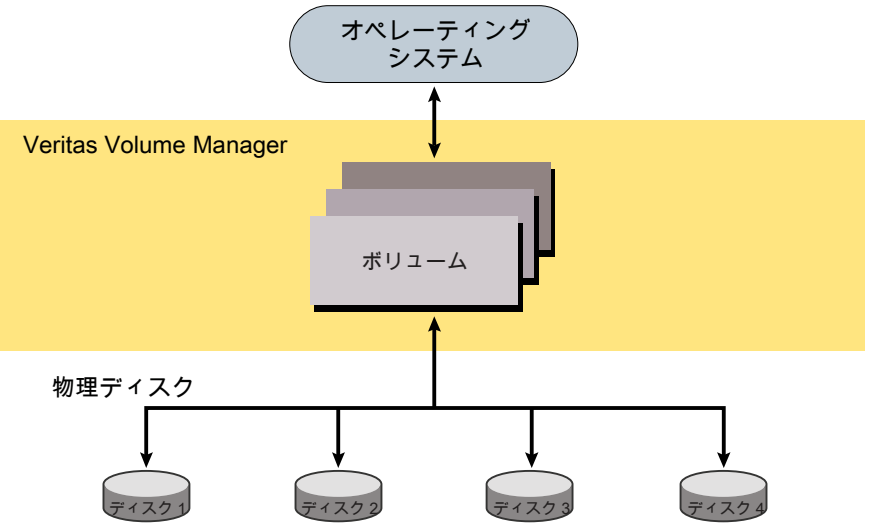
ディスクアレイ

ディスクは物理デバイスであり、読み取りや書き込みの前にディスク上の正しい位置にヘッドを移動する時間が必要になるため、ディスクへの I/O 操作は比較的遅い処理となります。すべての読み取りまたは書き込み操作が、個々のディスクに一度に 1 つずつ実行されると、読み書き時間が管理できなくなります。複数のディスクでこれらの操作を実行することで、この問題を軽減できます。

ディスクアレイは、VxVM がオペレーティングシステムに 1 つ以上の仮想ディスクまたはボリュームとして提示することができる物理ディスクの集合です。VxVM によって作成されたボリュームは、オペレーティングシステムで、物理ディスクのように認識され、動作します。ボリュームと対話するアプリケーションは、物理ディスクの場合と同様に動作します。

図 3-3は、VxVM がディスクアレイのディスクを複数のボリュームとしてオペレーティングシステムに提示する方法を示しています。

図 3-3 VxVM がディスクアレイのディスクをボリュームとしてオペレーティングシステムに提示する方法



データはアレイ内の複数のディスクまたは複数のアレイを使用するディスクに分散され、ディスク上の I/O 処理を分散または調整することができます。たとえば並列 I/O を複数のディスクで使うと、データ転送速度とアレイ全体のスループットの向上によって I/O 処理効率が改善されます。

仮想オブジェクト

VxVM（Veritas Volume Manager）は複数の仮想化層を使って、個別の機能を提供したり、物理的な制限を軽減したりします。物理ディスクを VxVM の制御下に置くと、物理オブジェクトと VxVM オブジェクトが接続されます。

表 3-1 では VxVM 内に仮想オブジェクトが記述されます。

表 3-1 VxVM 仮想オブジェクト

仮想オブジェクト	説明
ディスクグループ	ディスクグループは、共通の設定を共有するディスクの集合で、VxVM によって管理されます。ディスクグループ設定は、関連する VxVM オブジェクトとその属性と接続に関する詳細情報を格納する一連のレコードです。ディスクグループ名には、最大で 29 文字使えます。ディスクグループ名はピリオド（.）を含むことができません。

仮想オブジェクト	説明
VxVM ディスク	VxVM ディスクは、VxVM の制御下に物理ディスクを配置するときに、物理ディスクに割り当てられます。VxVM ディスクは通常ディスクグループ内にあります。VxVM は、VxVM のディスクスペースの連続した領域からストレージを割り当てます。 各 VxVM ディスクは、少なくとも 1 つの物理ディスクまたはディスクパーティションに対応します。 通常、VxVM ディスクにはパブリックリージョン（割り当てられたストレージ）と VxVM 内部設定情報が保存される小さいプライベートリージョンが存在します。
サブディスク	サブディスクは連続した一連のディスクブロックです。ブロックとは、ディスク上の領域の単位です。VxVM は、サブディスクを使ってディスク領域を割り当てます。VxVM ディスクは、1 つ以上のサブディスクに分割できます。各サブディスクは、VxVM ディスクの特定の部分で、物理ディスクの特定の領域にマップされます。
プレックス	プレックスは、1 つ以上の物理ディスクの 1 つ以上のサブディスクで構成されます。
ボリューム	ボリュームはアプリケーション、データベース、ファイルシステムでは物理ディスクデバイスと同様に認識されますが、その実体は物理ディスクデバイスに伴う物理的な制約は受けない仮想ディスクデバイスです。ボリュームは 1 つ以上のプレックスで構成され、それぞれのプレックスにはボリューム内で選択されたデータのコピーが保存されています。ボリュームは仮想ディスクであるため、特定のディスクまたはディスクの特定領域に制限されません。VxVM ユーザーインターフェースを使って、ボリュームの設定を変更できます。ボリュームを使っているアプリケーションやファイルシステムを混乱させることなく、設定を変更できます。たとえば、ボリュームを個別のディスク上にミラー化したり、別のディスクストレージを使うために移動することができます。

ホストシステムに VxVM をインストールした後、VxVM ディスクをディスクグループにまとめ、ディスクグループ領域を割り当てて論理ボリュームを作成し、物理ディスクの内容を VxVM の制御下に置く必要があります。

物理ディスクの内容を VxVM の制御下に置く処理は、VxVM が物理ディスクを制御していて、ディスクが LVM などの別のストレージマネージャの制御下にない場合にのみ行われます。

LVM ディスクと VxVM ディスクの共存方法や、LVM ディスクを VxVM ディスクに変換する方法について詳しくは、『Veritas InfoScale ソリューションガイド』を参照してください。

VxVM によって仮想オブジェクトが作成され、オブジェクト間に論理接続が形成されます。VxVM は、仮想オブジェクトを使ってストレージ管理タスクを実行します。

vxprint コマンドは、システムに存在する VxVM オブジェクトについての詳細情報を表示します。

vxprint（1M）マニュアルページを参照してください。

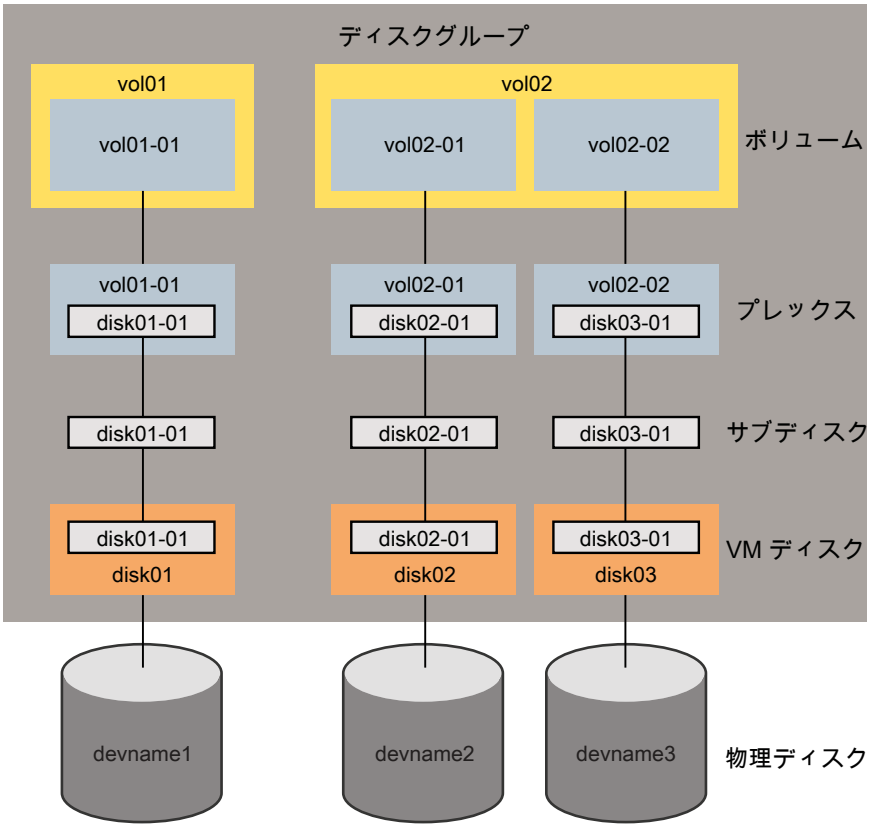
Veritas Volume Manager の仮想オブジェクトの結合

Veritas Volume Manager（VxVM）の仮想オブジェクトを結合してボリュームを構築できます。ボリューム内に含まれる仮想オブジェクトは、VxVM ディスク、ディスクグループ、サブディスクとプレックスです。VxVM のオブジェクトは、次のように構成されています。

- VxVM ディスクはディスクグループにグループ化されます。
- サブディスク（各サブディスクはディスクの特定領域を表す）は、結合されてプレックスを形成します。
- ボリュームは 1 つ以上のプレックスで構成されます。

図 3-4 に、VxVM の仮想オブジェクト間の関係と、物理ディスクとの対応関係を示します。

図 3-4 VxVM のオブジェクト間の関係



ディスクグループには、3 つの VxVM ディスクが含まれ、これらのディスクから 2 つのボリュームが作成されます。ボリューム `vol01` は単純な構成でプレックスは 1 つです。ボリューム `vol02` は、2 つのプレックスを持つミラーボリュームです。

各種の仮想オブジェクト(ディスクグループ、VM ディスク、サブディスク、プレックス、ボリューム)については、次の項で説明します。Veritas Volume Manager には、データ変更オブジェクト(DCO)やボリュームセットなどの拡張機能を提供するオブジェクトもあります。

Veritas Volume Manager の設定デーモンについて

Veritas Volume Manager(VxVM)の設定デーモン(`vxconfigd`)は、VxVM コマンドとカーネルデバイスドライバ間のインターフェースを提供します。`vxconfigd` は VxVM ユーティリティからの設定変更要求を処理し、その変更要求を VxVM カーネルに伝え、ディ

スクに保存されている設定情報を変更します。また、システムの起動時に VxVM を初期化します。

vxdctl コマンドは、vxconfigd デーモンに対するコマンドラインインターフェースです。

vxdctl は、次の目的に使えます

- vxconfigd デーモンの動作の制御
- システム全体でのデフォルトのディスクグループ定義の変更

VxVM 4.0 以降のリリースでは、ディスクアクセスレコードは /etc/vx/volboot ファイルに保存されません。非永続ディスクアクセスレコードは、システムの起動時にディスクをスキャンして作成されます。simple および nopriv ディスクの永続ディスクアクセスレコードは、root ファイルシステムの /etc/vx/darecs ファイルに保存されます。vxconfigd デーモンは、このファイルの内容を読み取って、そのディスクグループのディスクと設定データベースを検索します

/etc/vx/darecs ファイルには、自動設定できない外部デバイスの定義も保存されます。これらのエントリを追加するには、vxddladm addforeign コマンドを使います。

vxddladm(1M)マニュアルページを参照してください。

システムが DMP(Dynamic Multi-Pathing)を使うように設定されている場合は、vxdctl コマンドを次の目的にも使えます

- DMP データベースを再設定して、システムに新たに接続されたディスクデバイスまたは削除されたディスクデバイスを含める。
- /dev/vx/dmp と /dev/vx/rdmp ディレクトリに DMP デバイスノードを作成する。
- アクティブ/パッシブディスクアレイのパス型の変更を DMP データベースに反映する。プライマリとセカンダリ間のパス型の変更には、ディスクアレイのベンダーが提供するユーティリティを使います。

vxdctl(1M)マニュアルページを参照してください。

ディスクアレイへの複数パス

ディスクデバイスにアクセスする複数のポートを備えたディスクアレイもあります。これらのポートと、ホストバスアダプタ(HBA)コントローラとアレイにローカルなデータバスまたは I/O プロセッサを接続して、ディスクデバイスにアクセスする複数のハードウェアパスを作成できます。このようなディスクアレイをマルチパス化されたディスクアレイと呼びます。このタイプのディスクアレイは、多様な設定でホストシステムに接続できます(たとえば、シングルホスト上の異なるコントローラに接続された複数ポート構成、ホスト上の 1 つのコントローラを介したポートのチェーン構成、異なるホストに同時に接続されたポート構成など)。

p.43 の「DMP の動作方法」を参照してください。

Veritas Volume Manager のボリュームレイアウト

Veritas Volume Manager（VxVM）仮想デバイスは、ボリュームで定義されます。ボリュームは、ボリュームを 1 つ以上のプレックスへ関連付けることで定義されたレイアウトを持っています。ここで各プレックスは 1 つ以上のサブディスクにマップされています。ボリュームは、他のアプリケーションからデータにアクセスするための仮想デバイスインターフェースを提供します。これらの論理構成ブロックによって、ボリュームアドレス領域がリマップされます。これにより、I/O は実行時にリダイレクトされます。

可用性やパフォーマンスのレベルは、ボリュームレイアウトによって異なります。 必要なサービスのレベルに合わせて、ボリュームレイアウトを設定または変更できます。

非階層化ボリューム

非階層化ボリュームでは、サブディスクは VxVM ディスクに直接マップされます。これにより、VxVM ディスク上のパブリックリージョンに対応する連続したストレージ領域をサブディスクで定義することができます。アクティブな場合、VxVM ディスクは下位の物理ディスクに直接関連付けられます。このため、ボリュームレイアウトと物理ディスクの組み合わせによって、指定した仮想デバイスで利用できるストレージサービスが特定されます。

階層化ボリューム

階層化ボリュームは、サブディスクを下位となるボリュームにマップすることによって構成されます。この場合、下位ボリュームのサブディスクは、VxVM ディスクにマップすることで、接続されている物理ストレージにマップする必要があります。

階層化ボリュームを使うと、論理構成の組み合わせが豊富になります。そのいくつかは仮想デバイスの設定に適しています。たとえば、階層化ボリュームでは、ストライプ化を使うときの高可用性が可能になります。コマンドレベルで階層化ボリュームを自由に使えるようにすると管理が難しくなるため、定義済みの階層化ボリューム設定がいくつか VxVM に組み込まれています。

p.78 の「階層化ボリュームについて」を参照してください。

これらの定義済みの設定は、組み込まれたルールにより、指定された制約内でサービスの希望レベルを自動的に満たすように機能します。 現在の設定で動作する現在のコマンド起動に対して最も適した機能となるように、自動的に設定します。

一連の仮想デバイスから希望のストレージサービスを実現するためには、適切な一連の VxVM ディスクをディスクグループに編成し、複数の設定コマンドを実行する必要があります。

VxVM は可能なかぎり、管理インターフェースおよび一連のレイアウトとともに初期設定とオンライン再設定を処理し、このジョブを簡単で確定的なものにします。

レイアウト方法

仮想オブジェクトのデータは、次のレイアウト方法を使ってボリュームを作成するように編成されます。

- 連結、分散、およびカービング
  p.65 の「連結、分散、およびカービング」を参照してください。
- ストライプ化(RAID 0)
  p.67 の「ストライプ化(RAID 0)」を参照してください。
- ミラー化(RAID 1)
  p.71 の「ミラー化(RAID 1)」を参照してください。
- ストライプ化 + ミラー化(ミラー化ストライプ、RAID 0+1)
  p.71 の「ストライプ化 + ミラー化(ミラー化ストライプ、RAID 0+1)」を参照してください。
- ミラー化 + ストライプ化(ストライプ化ミラー、RAID 1+0 または RAID 10)
  p.72 の「ミラー化 + ストライプ化(ストライプ化ミラー、RAID 1+0 または RAID 10)」を参照してください。
- RAID 5(パリティ付きストライプ化)
  p.73 の「RAID 5(パリティ付きストライプ化)」を参照してください。

連結、分散、およびカービング

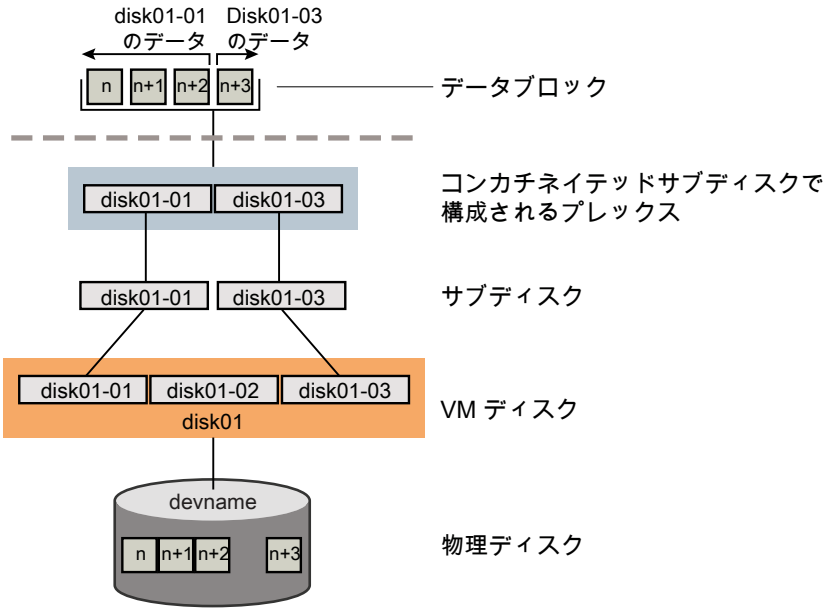
連結(コンカチネイト)によって、データはプレックス内の 1 つ以上のサブディスクに直線的にマップされます。コンカチネイテッドプレックス内のすべてのデータに連続的にアクセスする場合は、まず、最初のサブディスクのデータの先頭から末尾までアクセスします。その後、残りの各サブディスクのデータの先頭から末尾まで連続してアクセスし、最後のサブディスクまでアクセスします。

コンカチネイテッドプレックス内のサブディスクは物理的に連続している必要はなく、複数の VxVM ディスクに属することができます。複数の VxVM ディスクに存在するサブディスクを使った連結を分散(スパン)と呼びます。

図 3-5 に、同一の VxVM ディスクから 2 つのサブディスクを連結する様子を示します。

単一の LUN またはディスクが複数のサブディスクに分割され、各サブディスクが重複のないボリュームに属している場合は、これをカービングと呼びます。

図 3-5 連結ボリュームの例



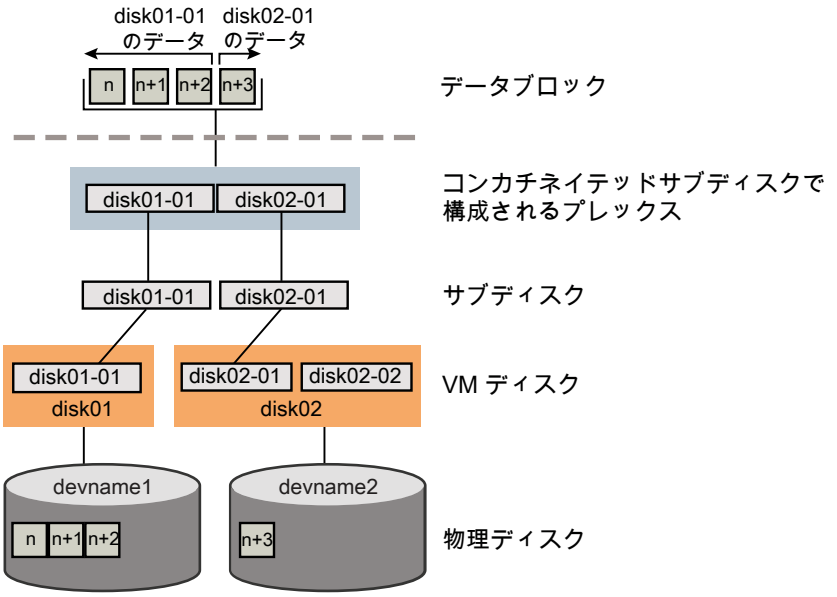
(プレックスの先頭から順に番号付けされた) n、n+1、n+2 および n+3 の各ブロックは、プレックス上では連続していますが、実際には同一の物理ディスク上の 2 つの異なるサブディスクから連結されたものです。

サブディスク disk01-01 と VxVM ディスク disk02 の残りの空き領域は、他の目的に使えます。

1 つのディスク上のプレックスの連続領域が十分でない場合、複数のサブディスクを用いて連結ディスクを使えます。この連結ディスク形式は、ディスク間の負荷調整および特定ディスク上のヘッド移動の最適化に使えます。

図 3-6 に、スパンプレックス内の 2 つのサブディスク上に分散しているデータを示します。

図 3-6 分散の例



(プレックスの先頭から順に番号付けされた) n、n+1、n+2 および n+3 の各ブロックは、プレックス上では連続していますが、実際には 2 つの異なる物理ディスク上の 2 つの異なるサブディスクから連結されたものです。

サブディスク disk02-02 と VxVM ディスク disk02 の残りの空き領域は、他の目的に使えます。

警告: 複数ディスクにまたがってプレックスを分散すると、ディスク障害により、割り当てられたボリュームに障害が発生する可能性が増大します。ミラー化または RAID 5 を使って、1 つのディスク障害によりボリューム障害が発生するリスクを軽減します。

ストライプ化(RAID 0)

ストライプ化(RAID 0)は、物理ディスクへ読み書きするデータ量が大きく、パフォーマンスが重要な場合に役立ちます。ストライプ化は、複数のディスクをまたぐマルチユーザーアプリケーションの I/O 負荷を調整するのにも役立ちます。複数ディスクとの並列データ転送を使うことにより、ストライプ化でデータアクセス性能を大幅に向上できます。

ストライプ化により、データは複数の物理ディスクにインタリーブする形でマップされます。ストライプ化プレックスには複数のサブディスクが含まれ、複数の物理ディスク上に分散されます。データは、ストライプ化プレックスのサブディスクに交互に均等に割り当てられます。

サブディスクは「カラム」にグループ化され、各物理ディスクのカラムは 1 つに限定されます。各カラムには 1 つ以上のサブディスクが含まれます。このサブディスクは 1 つ以上の物理ディスクから作成できます。カラムごとのサブディスクの数とサイズは様々です。必要に応じて、サブディスクをカラムに追加できます。

警告: 複数のディスクにまたがってボリュームのストライプ化や分割を行うと、ディスク障害によりボリューム障害が発生する可能性が増大します。

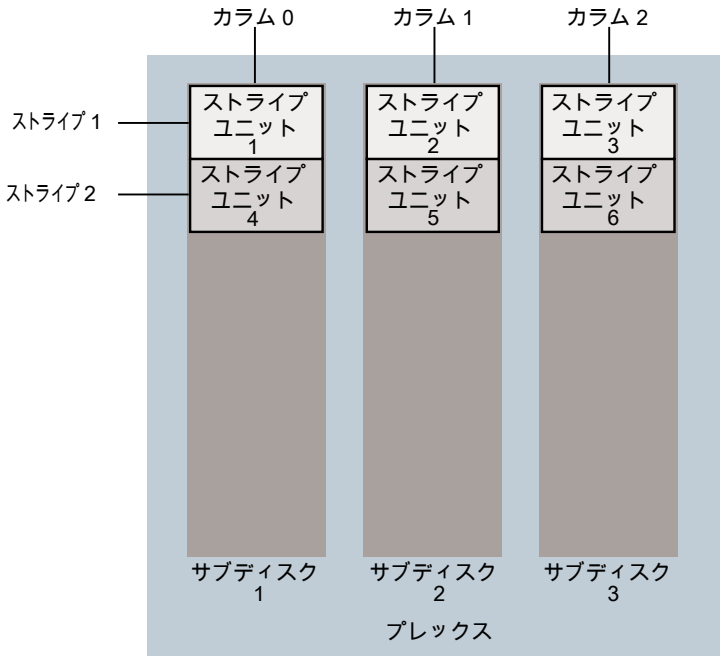
---

5 つのボリュームが同じ 5 つのディスク上にストライプ化された場合、5 つのディスクのどれか 1 つに障害が発生すると、5 つすべてのボリュームをバックアップから復元する必要があります。各ボリュームが個別のディスク上にある場合は、復元するボリュームは 1 つのみです（ストライプ化の代わりに、またはストライプ化と組み合わせてミラー化や RAID 5 を使うと、1 つのディスク障害により多数のボリューム障害が発生する可能性を大幅に軽減できます）。

データは同じサイズのストライプユニットに分けられ、カラム間にインタリーブされます。各ストライプユニットは、ディスク上の連続した一連のブロックです。デフォルトのストライプユニットサイズは 64 KB です。

図 3-7 に、ストライプ化プレックスの 3 つのカラム、6 つのストライプユニット、および 3 つのカラムにわたってストライプ化されたデータの例を示します。

図 3-7 3 つのカラムからなるストライプ化プレックス



ストライプは、すべてのカラムにまたがって同じ位置に存在する一連のストライプユニットから構成されます。この図では、ストライプユニット 1、2、3 によって 1 つのストライプが構成されています。

順番に見た場合、最初のストライプの構成要素は次のとおりです。

- カラム 0 のストライプユニット 1
- カラム 1 のストライプユニット 2
- カラム 2 のストライプユニット 3

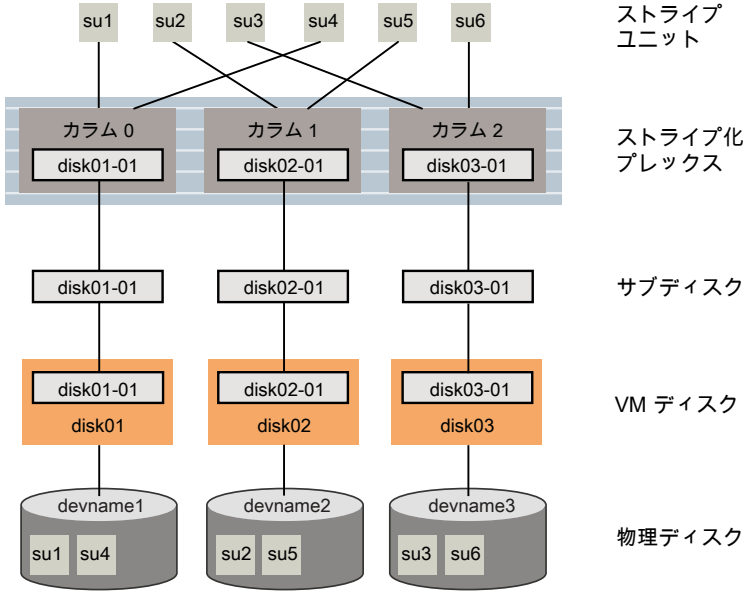
2 番目のストライプの構成要素は次のとおりです。

- カラム 0 のストライプユニット 4
- カラム 1 のストライプユニット 5
- カラム 2 のストライプユニット 6

ストライプ化は、カラムのサイズの終わりまで（すべてのカラムが同じサイズの場合）、サイズが違う場合は最も短いカラムの最後に到達するまで続行されます。長いカラムのサブディスクの最後に残っている領域は、未使用領域となります。

図 3-8 に、サブディスク 1 つを持つ等しいサイズのカラムが 3 つあるストライプ化プレックスを示します。

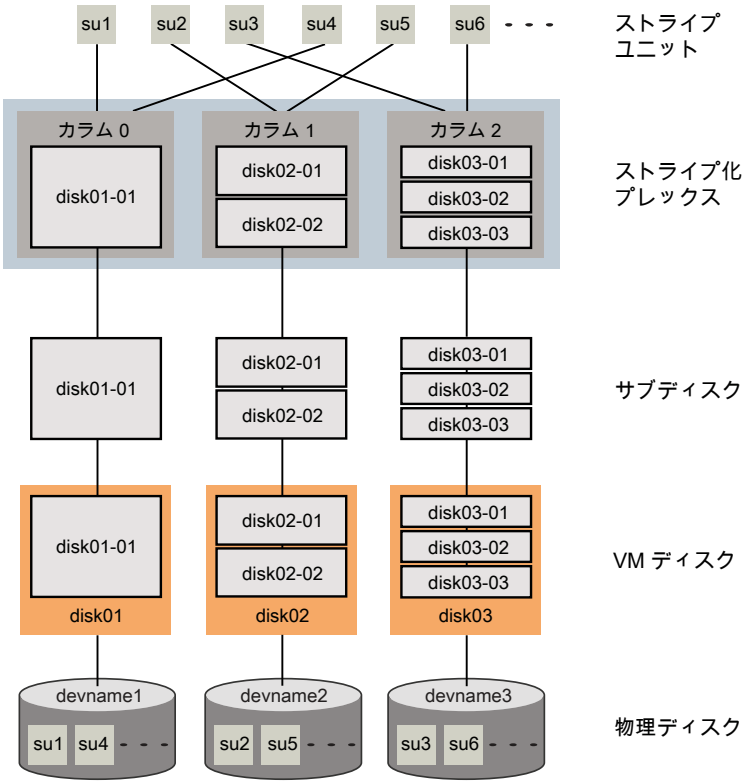
図 3-8　カラムごとに 1 つのサブディスクを持つストライプ化プレックスの例



物理ディスクごとに 1 つのカラムがあります。この例では、VM ディスクのすべての領域を使う 3 つのサブディスクを示しています。ストライプ化プレックスの各サブディスクが VM ディスクの一部のみを使うこともあります。この場合、空き領域を他のディスク管理タスクに使えます。

図 3-9 に、異なるサイズのサブディスクが含まれている 3 つのカラムのあるストライプ化プレックスを示します。

図 3-9 1 つのカラムにコンカチネイテッドサブディスクが存在するストライプ化プレックスの例



各カラムには、異なる数のサブディスクが含まれています。物理ディスクごとに 1 つのカラムがあります。ストライプ化プレックスは、ストライプ化する各 VM ディスクからサブディスクを 1 つずつ使って作成できます。同じディスクの異なる領域または別のディスクの領域から領域を割り当てることもできます（たとえば、プレックスのサイズが増加する場合）。カラムには異なる VM ディスクのサブディスクを含めることもできます。

p.244 の「ストライプボリュームの作成」を参照してください。

ミラー化（RAID 1）

ミラー化では、複数のミラー（プレックス）を使って、ボリュームに保存されている情報のコピーを複製します。物理ディスクの障害発生時、障害が発生したディスク上のプレックスは利用できなくなりますが、システムは影響を受けていないミラーを使って、動作を続行します。同様に、2 つの別々のコントローラの 2 つの LUN をミラー化すると、システムはコントローラ障害が発生した場合でも動作できます。

ボリュームには 1 つのプレックスのみ存在する場合もありますが、データの冗長性を確保するには、少なくとも 2 つのプレックスが必要です。冗長性を実現するには、これらの各プレックスが、異なるディスクからディスク領域を得る必要があります。

多数のディスクにまたがってストライプ化または分散を実行する場合、これらのディスクのいずれかに障害が発生すると、プレックス全体が使えなくなります。複数のディスクの 1 つに障害が発生する確率は相当に高いため、ストライプボリュームまたはスパンボリュームの信頼性（および可用性）を改善するためにミラー化を検討する必要があります。

p.242 の「ミラーボリュームの作成」を参照してください。

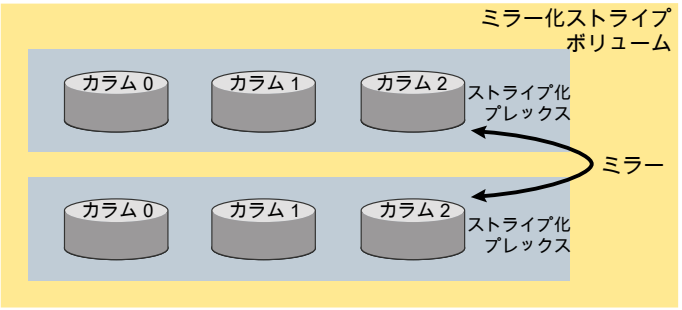
ストライプ化 + ミラー化（ミラー化ストライプ、RAID 0+1）

VxVM では、ストライプ化されたプレックスのミラー化をサポートしています。ストライプ化にミラー化を加えたレイアウトをミラー化ストライプレイアウトと呼びます。ミラー化ストライプレイアウトでは、複数ディスクにデータを分散するストライプ化の利点と、データの冗長性を確保するミラー化の利点の両方が得られます。

ストライプ化されたプレックスのミラー化を有効にするには、ストライプ化プレックスとそのミラーが異なるディスクから割り当てられる必要があります。

図 3-10 に、3 つのディスクにストライプ化された 2 つのプレックスがミラーとして同じボリュームに接続され、ミラー化ストライプボリュームを作成している例を示します。

図 3-10　　6 つのディスク上にレイアウトされたミラー化ストライプボリューム



p.245 の「ミラー化ストライプボリュームの作成」を参照してください。

各ミラーのデータプレックスのレイアウトタイプは、連結またはストライプにできます。ストライプ化されているミラーが 1 つのみであっても、そのボリュームをミラー化ストライプボリュームと呼びます。連結プレックスをミラー化したボリュームはミラー化連結ボリュームと呼びます。

ミラー化 + ストライプ化（ストライプ化ミラー、RAID 1+0 または RAID 10）

Veritas Volume Manager（VxVM）では、ミラー化されたプレックスのストライプ化をサポートしています。ミラー化にストライプ化を加えたレイアウトをストライプ化ミラーレイアウトと呼びます。プレックスをストライプ化する前にミラー化することで、ストライプの各カラムをミラー化します。各カラムにサブディスクが複数ある場合、各カラムではなく、各サブディスクを独立した状態でミラー化できます。

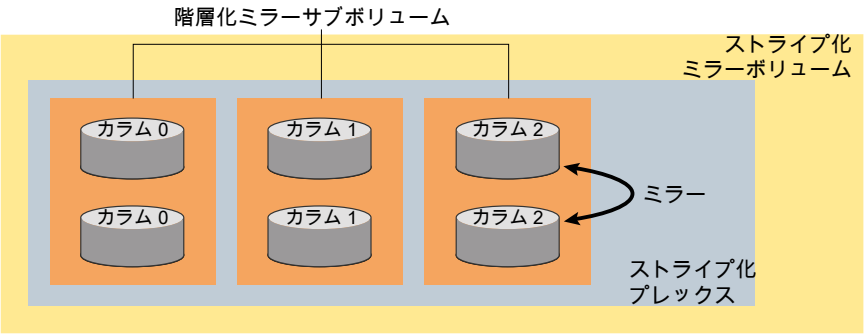
ストライプ化ミラーボリュームは、階層化ボリュームの一種です。

p.78 の「階層化ボリュームについて」を参照してください。

ミラー化ストライプボリュームと同様に、ストライプ化ミラーボリュームでは、複数ディスクにデータを分散するストライプ化の利点と、データの冗長性を確保するミラー化の利点の両方が得られます。さらに、冗長性が向上し、ディスク障害後の修復時間が短縮されます。

図 3-11 に、既存の 2 ディスク構成のミラーボリュームを 3 つ使ってストライプ化プレックス内に個別のカラムを形成し、ストライプ化ミラーボリュームを作成する例を示します。

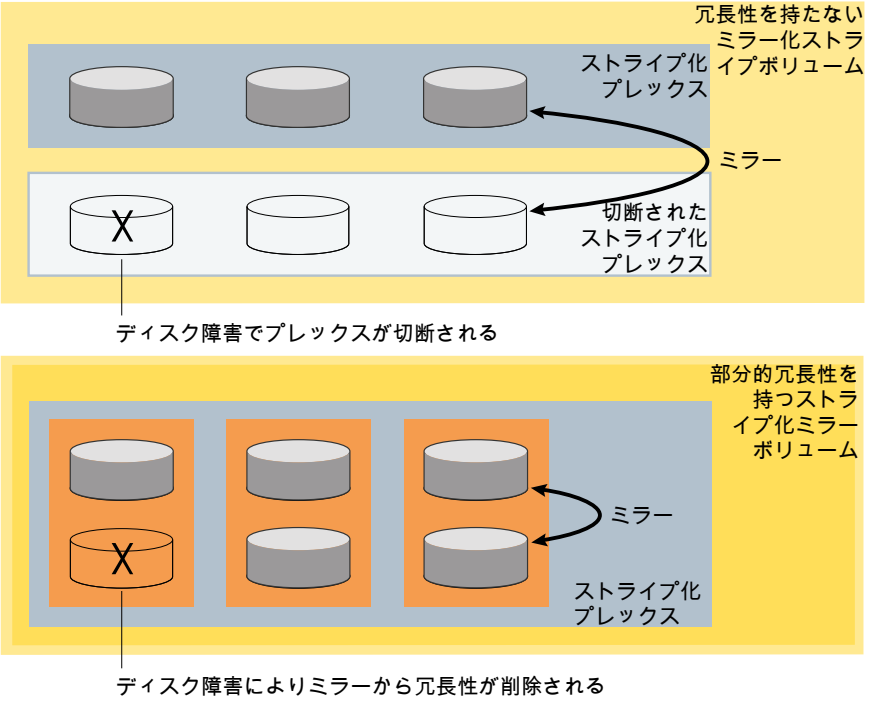
図 3-11　　6 つのディスク上にレイアウトされたストライプ化ミラーボリューム



p.245 の「ストライプ化ミラーボリュームの作成」を参照してください。

図 3-12 に、ミラー化ストライプレイアウト 1 つのディスク障害によってデータプレックス全体が切断され、ボリューム全体の冗長性が失われることを示します。

図 3-12　1 つのディスク障害がミラー化ストライプボリュームとストライプ化ミラーボリュームに与える影響



ディスク交換時に、プレックス全体を更新する必要があります。プレックス全体のリカバリには、膨大な時間を要することがあります。ストライプ化ミラーレイアウトの場合、ディスク障害の発生時に切断する必要があるのは障害の発生したサブディスクのみであり、その部分にかぎりボリュームの冗長性が失われます。ディスク交換時に修復を要するのは、ボリュームの一部のみです。さらに、ミラー化ストライプボリュームの場合、手動またはホットリロケーションで最初の障害ディスクが交換される前に 2 番目のディスクに障害が生じると、ボリューム全体が使えなくなります。

ミラー化ストライプボリュームと比較して、ストライプ化ミラーボリュームはディスク障害に強く、修復時間が短くなります。

下位のミラーボリュームをストライプ化する代わりに、階層化ボリュームを連結ボリュームから構成する場合、ボリュームは連結ミラーボリュームと呼ばれます。

RAID 5（パリティ付きストライプ化）

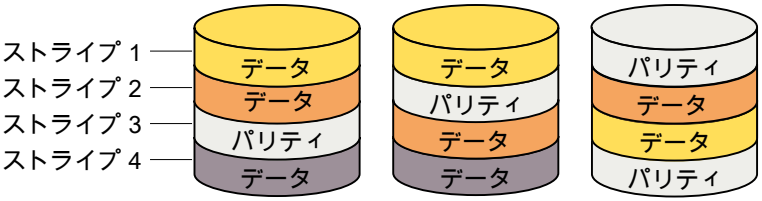
ミラー化（RAID 1）と RAID 5 はいずれもデータの冗長性を実現しますが、使う方法は異なります。ミラー化を行うと、ボリュームのデータの完全コピーが複数保持されるため、データの冗長性が確保されます。ミラーボリュームに書き込まれたデータはすべてのコピーに

反映されます。ミラーボリュームの一部に障害が生じても、システムはデータの他のコピーを使い続けます。

RAID 5 では、パリティを使ってデータの冗長性を確保します。パリティは、障害後にデータを復元するために使う計算値です。データが RAID 5 ボリュームに書き込まれている間、データに対し、排他的論理和（XOR）を使ってパリティが計算されます。計算されたパリティはボリュームに書き込まれます。データと計算されたパリティは、複数のディスクをまたがって「ストライプ化された」プレックスに保存されます。RAID 5 ボリュームの一部に障害が発生した場合、障害が発生したボリュームの該当する部分に存在していたデータは残りのデータとパリティ情報から再作成されます。また、このレイアウトでは、連結とストライプ化を混合することも可能です。

図 3-13 に、RAID 5 アレイ設定でのパリティの位置を示します。

図 3-13 RAID 5 モデルのパリティ位置



各ストライプには、パリティストライプユニットが含まれているカラムと、データが含まれているカラムが存在します。パリティはアレイ内のすべてのディスクに分散されます。この場合、書き込みをする際に 1 つのパリティディスクがデータを受け入れるまで待つ必要がないため、大規模な独立した書き込み時間が短縮されます。

RAID 5 ボリュームでは、修復時間を最短にするために、さらにログを記録できます。RAID 5 ボリュームでは、RAID 5 ログを使って現在書き込まれているデータとパリティのコピーを保存します。RAID 5 ログはオプションで、RAID 5 ボリュームと一緒に作成することも、後で追加することもできます。

p.75 の「Veritas Volume Manager の RAID 5 アレイ」を参照してください。

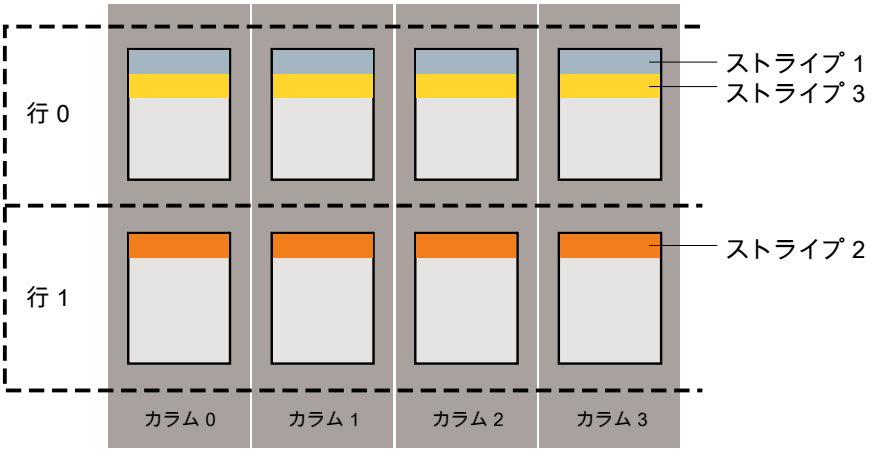
メモ: Veritas Volume Manager（VxVM）では、専用ディスクグループ上で構成するボリュームに対する RAID 5 レイアウトをサポートしていますが、Cluster Volume Manager（CVM）環境の共有ディスクグループ上で構成するボリュームに対するレイアウトとしてはサポートしていません。また、VxVM では VxVM ソフトウェアを使って設定された RAID 5 ボリュームのミラー化もサポートしていません。RAID 5 LUN ハードウェアはミラー化できます。

従来の RAID 5 アレイ

従来の RAID 5 アレイでは、複数のディスクが行とカラムで構成されています。カラムは、アレイ内で同じ順序に配置された複数のディスクです。行は、パリティストライプのストライプ幅全体を使ってサポートすることが可能な最小数のディスクです。

図 3-14 に、従来の RAID 5 アレイの行およびカラムの配列を示します。

図 3-14 従来の RAID 5 アレイ



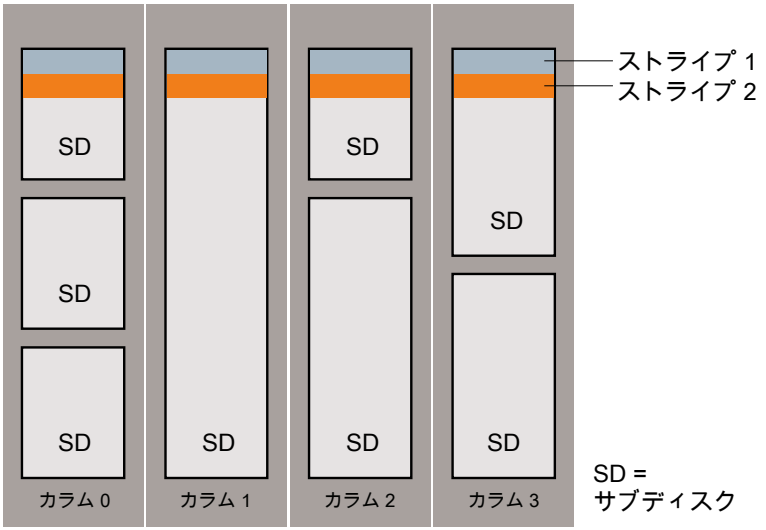
この従来のアレイ構造では、カラムごとに行を追加してアレイを拡張することができます。ストライプ化は、行 0 のディスクにまたがって最初のストライプを適用し、次に行 1 のディスクにまたがって 2 番目のストライプを適用し、次に、行 0 のディスクにまたがって 3 番目のストライプを適用する、というように実現します。このタイプのアレイでは、すべてのディスクカラムおよび行が同じサイズである必要があります。

Veritas Volume Manager の RAID 5 アレイ

Varitas Volume Manager（VxVM）の RAID 5 アレイ構造は、従来のアレイ構造とは異なります。ディスクと他のオブジェクトが仮想のものなので、VxVM では行を使いません。

図 3-15 に、各サブディスクがディスクの特定領域を表す、変数長サブディスクで構成されるカラムを VxVM が使う方法を示します。

図 3-15 Veritas Volume Manager の RAID 5 アレイ



VxVM では、RAID 5 プレックスの各カラムを、異なる数のサブディスクで構成できます。指定したカラムのサブディスクは、異なる物理ディスクをもとに作成できます。必要に応じて、サブディスクをカラムに追加できます。ストライプ化は、各カラムの最上部の各ディスクにまたがって最初のストライプを適用し、続いてその次にストライプを適用するというように繰り返して、カラムの最後まで続けます。各カラムには、同じサイズのストライプユニットを使います。RAID 5 の場合、デフォルトのストライプユニットサイズは 16 KB です。

p.67 の「ストライプ化（RAID 0）」を参照してください。

メモ: RAID 5 ボリュームのミラー化は、サポートされていません。

p.246 の「RAID 5 ボリュームの作成」を参照してください。

左対称レイアウト

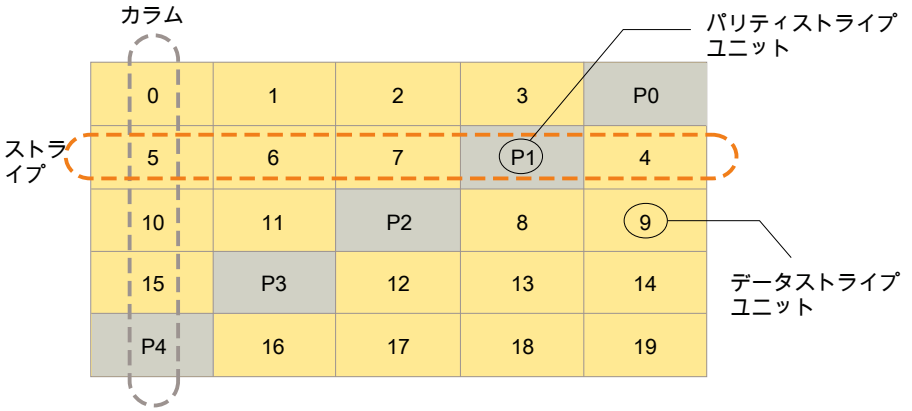
RAID 5 アレイの設定に使えるデータとパリティのレイアウトは複数あります。VxVM での RAID 5 の実現には、左対称レイアウトを使います。これによって、ランダム I/O 操作と大規模な連続 I/O 操作の両方の処理効率が最適化されます。ただし、レイアウト選択より、カラム数とストライプユニットサイズのほうが処理効率に大きな影響を与えます。

左対称レイアウトでは、カラムのデータとパリティの両方がストライプ化され、データの各ストライプについて、パリティは異なるカラムに配置されます。最初のパリティストライプユニットは、最初のストライプの最も右側のカラムに配置されます。各後続パリティストライプユニットは、前のパリティストライプユニットの位置から 1 つ左のカラムにシフトされた、次の

ストライプに配置されます。カラムよりストライプが多い場合は、パリティストライプユニットの配置は一番右のカラムから再度開始されます。

図 3-16 に、5 つのディスク(1 カラムに 1 つずつ)が存在する左対称パリティレイアウトを示します。

図 3-16　　左対称レイアウト



各ストライプに対して、データはパリティストライプユニットの右から開始されるように配置されます。図では、最初のストライプのデータ配置は P0 から、ストライプユニット 0 - 3 まで連続的に配置されます。2 番目のデータ配置は P1 から、ストライプユニット 4 までおよびストライプユニット 5 - 7 まで連続的に配置されます。残りのストライプについてもこのようにデータが配置されます。

各パリティストライプユニットには、同じストライプ内のデータストライプユニットのデータに実行された排他的論理和(XOR)演算の結果が保存されています。ハードウェアまたはソフトウェアの障害のため、1 つのカラムのデータにアクセスできない場合は、残りのカラムのデータストライプユニットの内容をそれぞれのパリティストライプユニットに対して XOR 演算することにより、各ストライプのデータを修復できます。

たとえば、一番左のカラムの一部または全部に対応するディスクに障害が生じた場合、ボリュームは縮退モードで配置されます。縮退モードでは、ストライプユニット 1 - 3 をパリティストライプユニット P0 に対して XOR 演算してストライプユニット 0 を再作成し、次にストライプユニット 4、6、7 をパリティストライプユニット P1 に対して XOR 演算してストライプユニット 5 を再作成するというように、障害のあるカラムのデータを再作成できます。

RAID 5 プレックスで複数のカラムに障害が生じると、ボリュームは切断されます。ボリュームでは、読み取りまたは書き込み要求を受け付けることができなくなります。障害のあるカラムが修復されると、バックアップからユーザーデータを復元する必要があります。

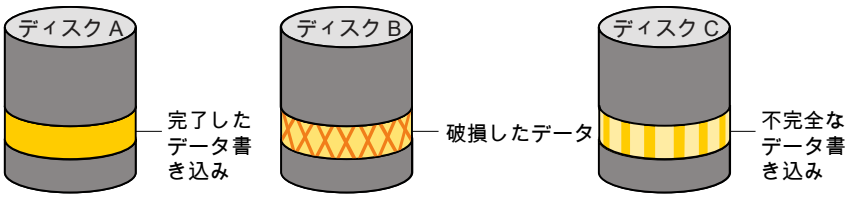
RAID 5 ログ

ログを使って、変更とパリティを永続デバイス(ディスク上のボリュームや非揮発性 RAM など)上のログ領域に即座に記録することにより、修復中のデータの破損を防止します。ログを記録した後に、新しいデータとパリティがディスクに書き込まれます。

ログを記録しないと、RAID 5 ボリュームのディスクとシステムの両方に障害が生じた場合、アクティブな書き込みに関与していないデータが損失したり、知らないうちに破損する可能性があります。このように 2 重に障害が生じた場合、ディスクのデータ部分に書き込まれたデータやパリティ部分に書き込まれたパリティが実際に書き込まれたかどうかを知ることができません。したがって、破損されたディスクのリカバリ自体も破損されている可能性があります。

図 3-17 に、3 つのディスク(A、B、C)上に設定された RAID 5 ボリュームを示します。

図 3-17 RAID 5 ボリュームへの不完全な書き込み



このボリュームで、ディスク B の破損したデータを修復できるかどうかは、ディスク A のデータとディスク C のパリティが完全であるかどうかで決まります。ただし、ディスク A へのデータ書き込みのみが完了しています。ディスク C へのパリティ書き込みが不完全なため、ディスク B のデータが不正確に復元される可能性があります。

この障害は、すべてのデータとパリティの書き込みをログに記録してからアレイに書き込むことで回避できます。この方法では、障害のあるドライブを復元する前に、ログを再生することにより、データとパリティの更新を完了できます。

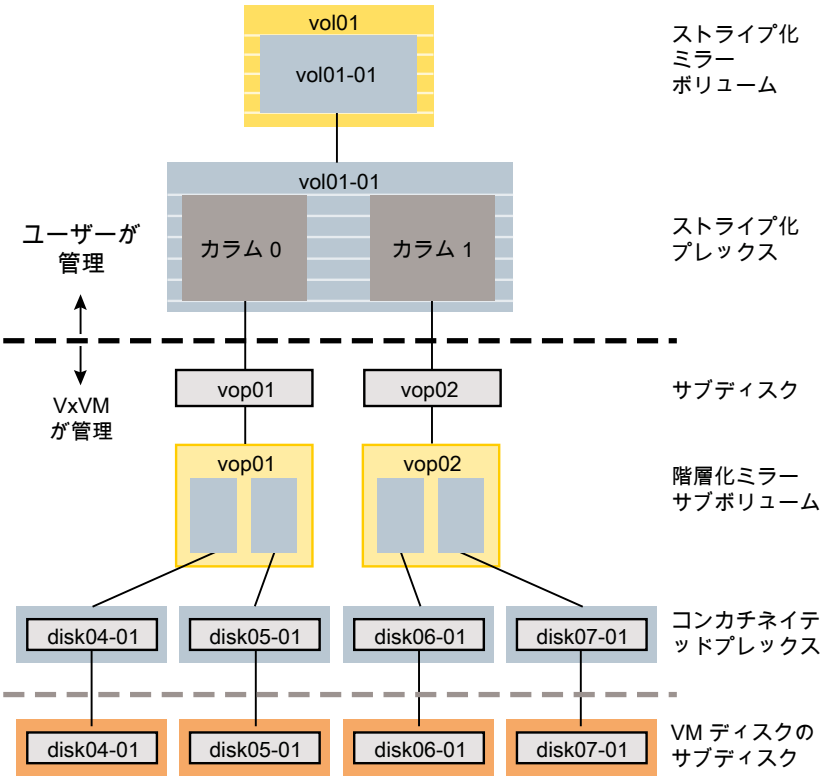
サブディスクをログプレックスとして追加することにより、ログは RAID 5 ボリュームに関連付けられます。各 RAID 5 ボリュームに、複数のログプレックスを作成できます。この場合、ログ領域はミラー化されます。

階層化ボリュームについて

階層化ボリュームは、他のボリュームの上に構築される Veritas Volume Manager(VxVM)仮想オブジェクトです。階層化ボリューム構造では、標準ボリューム構造より障害に対して強く、冗長性にも優れています。たとえば、ストライプ化ミラー階層化ボリュームの場合、各ミラー(プレックス)はより小さなストレージを対象とするため、リカバリは標準的なミラーボリュームより速くなります。

図 3-18 に、下位のミラーボリュームから構築されているサブディスクで各カラムが表される標準的なストライプ化ミラー階層化ボリュームを示します。

図 3-18 ストライプ化ミラー階層化ボリュームの例



ユーザーは、ユーザー管理領域にあるボリュームとストライプ化プレックスを使って、VxVM で通常のタスクを実行することができます。ユーザータスクは、階層化ボリュームの最上位ボリュームでのみ実行されます。

VxVM の管理領域にある下位ボリュームは VxVM により排他的に使われ、ユーザーによる操作はできません。内部構造を操作して階層化ボリュームを切断したり、下位ボリュームで他の操作を実行することはできません。必要な操作（ボリュームのサイズ変更、カラム幅の変更、カラムの追加など）はすべて、最上位ボリュームとストライプ化プレックスを含むユーザー管理領域で実行できます。

システム管理者は、トラブルシューティングやその他の処理（たとえば、特定のディスクにデータを配置するなど）のために、階層化ボリューム構造を操作できます。階層化ボリュームは、次のタスクと操作を実行するために VxVM で使われます。

ストライプ化ミラーの作成	p.245 の「ストライプ化ミラーボリュームの作成」を参照してください。 vxassist(1M)マニュアルページを参照してください。
連結ミラーの作成	p.244 の「連結ミラーボリュームの作成」を参照してください。 vxassist(1M)マニュアルページを参照してください。
オンライン再レイアウト	p.80 の「オンライン再レイアウト」を参照してください。 vxassist(1M)マニュアルページを参照してください。 vxrelayout(1M)マニュアルページを参照してください。
RAID 5 サブディスクの移動	vxsd(1M)マニュアルページを参照してください。
スナップショットの作成	p.87 の「ボリュームスナップショット」を参照してください。 vxassist(1M)マニュアルページを参照してください。 vxsnap(1M)マニュアルページを参照してください。

オンライン再レイアウト

オンライン再レイアウトを使うと、データアクセスを中断することなく、VxVM 内のストレージレイアウトを変換することができます。通常、ボリュームの冗長性またはパフォーマンス特性を変更する場合に、この操作を実行します。VxVM では、データを複製する(ミラー化)か、パリティを追加する(RAID 5)ことによって、ストレージの冗長性を確保します。VxVM のストレージのパフォーマンス特性は、ストライプ化のパラメータを変更することによって変更できます。このパラメータはカラム数とストライプ幅です。

p.966 の「オンライン再レイアウトの実行」を参照してください。

オンライン再レイアウトの動作方法

オンライン再レイアウトを使うと、データアクセスを妨げることなく、すでに作成したストレージレイアウトを変更することができます。特定のレイアウトの処理効率特性を、変更した必要条件に合うように変更できます。1 つのコマンドを呼び出して、あるレイアウトを別のレイアウトに変換できます。

たとえば、ストライプユニットサイズが 128 KB に設定されたストライプレイアウトで最適の処理効率が得られない場合は、再レイアウトを使って、ストライプユニットサイズを変更できます。

ボリューム上にマウントされたファイルシステム（Veritas File System など）がオンラインでの拡大操作と縮小操作をサポートしている場合は、この変換を実行するために、ファイルシステムのマウントを解除する必要はありません。

オンライン再レイアウトでは、既存のストレージ領域を再利用し、新しいレイアウトのニーズに応える領域割り当てポリシーがあります。レイアウト変換プロセスでは、ディスクグループで利用できる最小限の一時使用領域を使って、所定のボリュームが指定されたレイアウトに変換されます。

ソースレイアウトのデータを一度に一部ずつ指定されたレイアウトに移動して、変換が実行されます。データはソースボリュームから一時使用領域にコピーされ、その部分がソースボリュームストレージから削除されます。すると、ソースボリュームストレージは新しいレイアウトに変換され、一時使用領域に保存されているデータが新しいレイアウトに書き込まれます。この操作は、ソースボリュームのすべてのストレージとデータが新しいレイアウトに変換されるまで繰り返されます。

再レイアウトで使われる一時使用領域のデフォルトサイズは、ボリュームのサイズや再レイアウトのタイプによって異なります。50 MB を超えるボリュームの場合、必要な一時使用領域は通常、ボリュームのサイズの 10%（50 MB - 1 GB）になります。50 MB 未満のボリュームの場合は、ボリュームと同サイズの一時使用領域が必要です。

ディスクグループ内に一時使用領域用の空き領域が十分に存在しない場合は、必要なブロック数を示す次のエラーメッセージが表示されます。

```
tmpsize too small to perform this relayout (nblks minimum required)
```

`tmpsize` 属性を `vxassist` に使って、一時使用領域に使うデフォルトサイズを上書きできます。

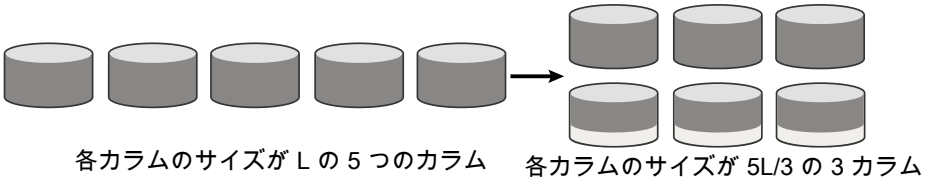
`vxassist`（1M）マニュアルページを参照してください。

一時使用領域と同様に、ストライプボリュームのカラムのサイズを拡大するには、一時使用中間ボリュームに領域が必要になります。ターゲットボリュームとソースボリュームのカラムのサイズの差に相当する領域が必要です。たとえば、長さが 50 GB の 3 つのカラムがあるため、長さが 30 GB の 5 つのカラムを持つ 150 GB のストライプボリュームを再レイアウトするときに、一時追加領域の 20 GB が必要になります。場合によっては、必要な一時使用領域はより大きくなります。たとえば、5 つのカラムで構成される 150 GB のストライプボリュームを、連結ボリューム（実際には 1 つのカラム）に再レイアウトするには、一時使用の中間ボリュームとして 120 GB が必要になります。

実行する再レイアウトのタイプによっては、再レイアウト先のボリュームに追加の永続ディスク領域が必要になります。たとえば、ストライプボリュームのカラム数を変更した場合がこれに該当します。

図 3-19 に、カラム数を減らすことによってボリュームに必要なディスク数が増加するケースを示します。

図 3-19　　ボリューム内のカラム数を減らす例

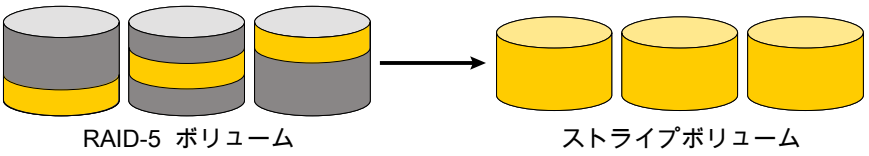


ボリュームのサイズは変わりませんが、カラムの 1 つの拡張先としてディスクを 1 つ追加する必要があるので注意してください。

オンライン再レイアウトを使って実行できる操作の例を次に示します。

- RAID 5 ボリュームを連結ボリューム、ストライプボリュームまたは階層化ボリュームから削除します。
  図 3-20 に、レイアウトを RAID 5 ボリュームに適用する例を示します。

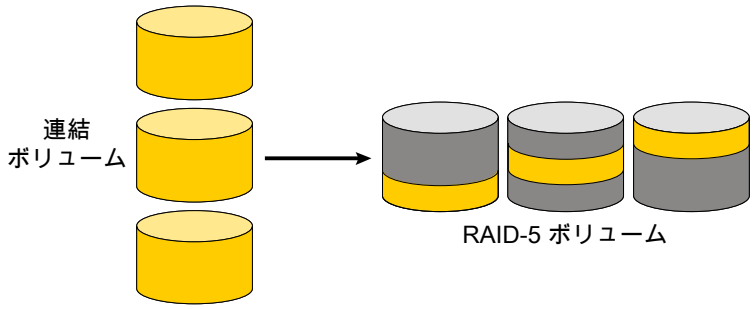
図 3-20　　RAID 5 ボリュームをストライプボリュームに再レイアウトする例



パリティを削除すると、ボリュームに必要なストレージ領域の総量が減少します。

- パリティをボリュームに追加して、RAID 5 ボリュームに変更します。
  図 3-21 に例を示します。

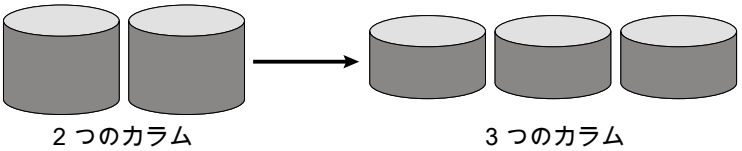
図 3-21　　連結ボリュームを RAID 5 ボリュームに再レイアウトする例



パリティを追加すると、ボリュームに必要なストレージ領域の総量が増加します。

- ボリューム内のカラム数を変更する場合。
  図 3-22 に、カラム数を変更する例を示します。

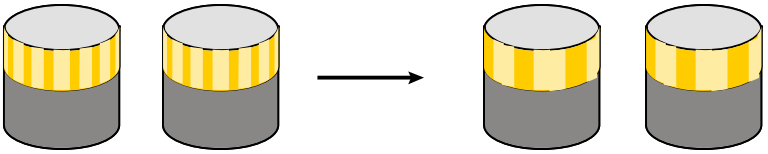
図 3-22　　ボリューム内のカラム数を増やす例



ボリュームのサイズを保つためにカラムサイズが縮小されます。

- ボリューム内のカラムストライプ幅を変更する場合。
  図 3-23 に、カラムのストライプ幅を変更する例を示します。

図 3-23　　ボリューム内のカラムのストライプ幅を増加する例



p.966 の「オンライン再レイアウトの実行」を参照してください。
p.967 の「可能な再レイアウト変換」を参照してください。

オンライン再レイアウトの制限

オンライン再レイアウトには次の制限があることに注意してください。

- ログプレックスは変換できません。
- ボリュームスナップショットは、ボリューム上でオンライン再レイアウト操作が実行されているときは作成できません。
- オンライン再レイアウトでは、単一の操作で非階層化ミラーボリュームが作成できません。vxassist relayout または vxrelayout コマンドに `mirror-stripe` または `mirror-concat` などの非階層化ミラーレイアウトを指定しても、常に階層化ミラーボリュームが作成されます。`vxassist convert` コマンドを使って、再レイアウトで作成された階層化ミラーボリュームを非階層化ボリュームに変換してください。

- 一般的な仕様上の制限事項として、変換先レイアウトを作成するために必要な最小数の物理ディスクが適用されます。たとえば、ミラーボリュームには少なくともミラーと同数のディスクが必要です。ストライプボリュームおよび RAID 5 ボリュームには少なくともカラムと同数のディスクが必要です。ストライプ化ミラーボリュームには少なくともミラー数にカラム数をかけたものと同数のディスクが必要です。
- レイアウト変換を適用するには、ミラーボリューム内のプレックスに同じストライプ幅とカラム数が必要です。各プレックスのレイアウトが一致しないかぎり、再レイアウトは実行できません。
- オンライン再レイアウトでは、スパースプレックスを変換できません。また、プレックスを分散させることもできません (スパースプレックスは、ボリュームと同じサイズのプレックスではありません。また、どのサブディスクにもマップされていない領域があるプレックスです)。
- ミラーボリューム内のミラー数は、再レイアウトでは変更できません。代わりに、`vxassist mirror` コマンドなどの代替コマンドを使用します。
- 一度にボリュームに適用できるレイアウトは 1 つだけです。

変換の特性

データをあるレイアウトから別のレイアウトへ変換するには、データを既存のレイアウトから新しいレイアウトに再配置する必要があります。変換中、オンライン再レイアウトでは、使う一時使用領域をミラー化して、データの冗長性を保持します。変換中は、データの読み取りおよび書き込みアクセスは割り込まれません。

変換中にシステムに障害が生じても、データは破損されません。システムが復元された後も変換は続行され、読み取りおよび書き込みアクセスは維持されます。

いつでもレイアウト変換プロセスをリバースできますが、データは以前の正確なストレージ位置に返されない場合もあります。進行中の変換をリバースする前に、その変換を停止する必要があります。

`vxrelayout status` *`volume`* コマンドを使って、変換方向を指定できます。

データを移動するための十分な冗長性と領域があれば、これらの変換処理は I/O 障害から保護されます。

変換とボリュームのサイズ

レイアウト変換によっては、ボリュームのサイズが増減します。このような状態が生じたら、オンライン再レイアウトで `vxresize` コマンドを使って、ファイルシステムを縮小または拡大します。

`vxresize`(1M)のマニュアルページを参照してください。

ボリュームの再同期

データを冗長性を持って保存したり、ミラーボリュームまたは RAID 5 ボリュームを使う場合、VxVM は、データのすべてのコピーが完全に一致するようにします。ただし、ある状況下では（通常、全体的なシステム障害のため）、ボリューム上の冗長データの一貫性が失われたり、同期化されないことがあります。ミラーデータは、もとのデータと厳密に同じではありません。このような状況は、通常の設定変更（プレックスの切断および再接続など）の他、データがボリュームに書き込まれている最中にシステムがクラッシュしたときなどに、生じることがあります。

データは、RAID 5 ボリュームのデータとパリティのように、並行してボリュームのミラーに書き込まれます。個々の書き込みが完了する前にシステムがクラッシュした場合は、完了している書き込みと完了していない書き込みが生じることがあります。このため、データが同期化されないことがあります。ミラーボリュームの場合、読み取り要求を満たすために異なるミラーが使われると、ボリュームの同じ領域からの 2 つの読み取りが異なる結果を返すことがあります。RAID 5 ボリュームの場合、パリティが破損し、不正なデータが復元されることになります。

VxVM は、すべてのミラーで完全に同じデータが保持され、RAID 5 ボリュームのデータとパリティが一致するようにします。このプロセスをボリュームの再同期と呼びます。 起動時に自動的にインポートされるディスクグループ（通常、システム全体で予約済みのディスクグループとしてエイリアスが設定されるディスクグループ bootdg）の一部であるボリュームでは、システムが再ブートすると、再同期が実行されます。

すべてのボリュームで、システム障害後の再同期を必要とするわけではありません。書き込みがされたことのないボリュームや休止状態の（すなわち、有効な I/O がない）ボリュームは、システム障害が生じたとき未処理の書き込みがないため、再同期の必要はありません。

ダーティフラグ

VxVM はボリュームが最初に書き込まれると、それを記録し、ダーティとして設定します。ボリュームについてすべてのプロセスが終了したか、ボリュームが管理者によってクリーンな状態で停止され、すべての書き込みが完了している場合、VxVM はそのボリュームのダーティフラグを削除します。 ダーティとして設定されているボリュームのみ再同期が必要です。

再同期プロセス

再同期のプロセスはボリュームタイプによって異なります。ミラーボリュームの場合は、ボリュームがリカバリモード（読み取り-ライトバックリカバリモードとも呼びます）となり再同期が実行されます。ボリューム内のデータの再同期は、バックグラウンドで実行されます。そのため、リカバリ実行中でもボリュームを使えます。RAID 5 ログを含む RAID 5 ボリュームは、これらのログを再生できます。ログが利用できない場合、ボリュームは復元リカバリモードとなり、すべてのパリティが再生成されます。

再同期は、システムの処理効率に影響を与える可能性があります。リカバリプロセスを実行すると、特定のディスクやコントローラにストレスを与えないように、リカバリが分散され、この影響が軽減されます。

大規模なボリュームや多数のボリュームの場合、再同期プロセスに時間がかかることがあります。これらの影響は、ミラーボリュームに DRL（dirty region logging）と FastResync（高速ミラー再同期）を使うか、または RAID 5 ボリュームに RAID 5 ログを使うことにより、最小限に抑えることができます。

p.86 の「DRL」を参照してください。

Oracle で使われるミラーボリュームについては、パフォーマンスをさらに向上する SmartSync 機能を使うことができます。

ホットリロケーション

ホットリロケーションは、システムが VxVM の冗長オブジェクト（ミラーボリュームまたは RAID 5 ボリューム）上の I/O 障害に自動的に対応し、冗長性を復元し、これらのオブジェクトにアクセスできるようにする機能です。VxVM はオブジェクトの I/O 障害を検出し、影響を受けたサブディスクを再配置します。影響を受けたサブディスクは、スペアディスクとして指定されているディスクまたはディスクグループ内の空き領域に再配置されます。その後、VxVM は次に障害発生前に存在したオブジェクトを復元し、それらに再度アクセスできるようにします。

部分的なディスク障害が発生した場合（すなわち、ディスク上の一部のサブディスクにのみ影響する障害の場合）は、障害発生部分の冗長データが再配置されます。影響を受けていないディスク部分の既存ボリュームには引き続きアクセスできます。

p.898 の「ホットリロケーションの動作方法」を参照してください。

DRL

DRL（dirty region logging）を有効にすると、システムクラッシュ後のミラーボリュームのリカバリが高速化されます。DRL では、ミラーボリュームへの I/O 書き込みのために変更された領域を追跡します。DRL ではこの情報を使って、ボリュームのこれらの部分のみを修復します。

DRL を使っておらずシステムに障害が生じた場合は、ボリュームのすべてのミラーを復元して整合性のある状態にする必要があります。ミラーからボリュームの全内容がコピーされて、リストアが完了します。このプロセスは長くかかり、I/O を集約的に使います。

メモ: DRL を使うと、大部分の書き込みアクセスパターンで I/O 負荷が少し増加します。このオーバーヘッドは、SmartSync を使うことによって削減されます。

インスタントスナップの DCO ボリュームをボリュームに関連付けた場合、DCO ボリュームの一部を使って DRL ログを保存できます。インスタントスナップの DCO ボリュームが存在するボリューム用に、DRL ログを個別に作成する必要はありません。

ログサブディスクとログプレックス

DRL ログサブディスクは、有効になったミラーボリュームの DRL を保存します。DRL が有効になったボリュームには、少なくとも 1 つのログサブディスクが存在します。複数のログサブディスクを使って DRL をミラー化できます。各ログサブディスクは、ボリュームのプレックスの 1 つと関連付けられます。各プレックスに配置できるログサブディスクは 1 つのみです。プレックスにログサブディスクのみがあり、データサブディスクがない場合、そのプレックスはログプレックスと呼ばれます。

ログサブディスクは、データサブディスクを含む通常のプレックスと関連付けることもできます。この場合、プレックスをそのデータサブディスクの 1 つでの障害のために切断する必要があるときは、ログサブディスクが利用できなくなる可能性があります。

`vxassist` コマンドを使って DRL を作成すると、デフォルトで 1 つのログサブディスクを含むログプレックスが作成されます。DRL は、ログサブディスクを作成し、それをプレックスに関連付けて、手動で設定することもできます。その場合、プレックスにはログとデータサブディスクの両方が含まれます。

シーケンシャル DRL

データベースログの再生に使うボリュームなどのボリュームは、連続して書き込まれるため、DRL ビットの遅延書き込みのパフォーマンスに対する効果はありません。これらのボリュームには、シーケンシャル DRL を使って、遅延書き込みを防ぎ、ダーティリージョン数を抑えることができます。これによって迅速に修復できます。ただし、シーケンシャル DRL をランダムに書き込まれたボリュームに適用すると、実行できる並行書き込み数が制限されるため、処理効率の障害となることがあります。

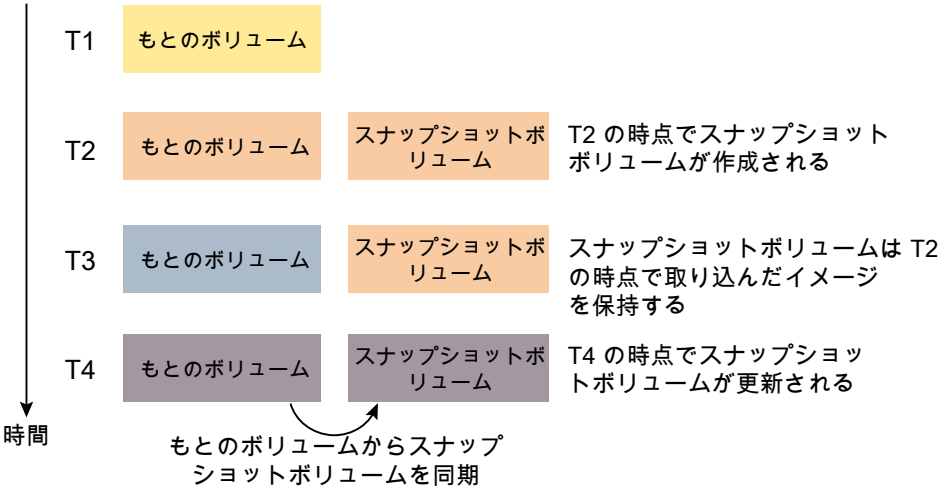
シーケンシャル DRL に対して許可されている最大ダーティリージョン数は、チューニングパラメータによって制御されます。詳しくは、「`voldrl_max_seq_dirty`」の項を参照してください。

ボリュームスナップショット

Veritas Volume Manager には、特定時点のボリュームのイメージを作成する機能があります。この特定時点でのボリュームのイメージを、ボリュームスナップショットと呼びます。ボリュームスナップショットとファイルシステムのスナップショットを混同しないように注意してください。後者は、Veritas File System のポイントインタイムイメージを指しています。

図 3-24 に、特定時点でのもとのボリュームのコピーがスナップショットボリュームにどのように反映されるかを示します。

図 3-24 ボリュームの point-in-time イメージとしてのボリュームスナップショット



元のボリュームの内容を変更しても、スナップショットボリュームでは、変更前の時点での元のボリュームの内容が保持されます。

スナップショットボリュームは、安定した独立の情報ベースとして、元のボリュームの内容のバックアップや、意思決定支援システムなどの他のアプリケーションに活用できます。この図では、後でスナップショットボリュームの内容が元のボリュームと再同期されます。

また、スナップショットボリュームは、元のボリュームの内容を復元することにも使えます。何らかの理由で元のボリュームの内容が破損した場合には、スナップショットボリュームを使うと便利です。

警告: スナップショットボリュームにデータを書き込むと、元のボリュームの内容の復元には使えなくなります。

VxVM のボリュームスナップショットの 1 つのタイプに、サードミラーブレークオフタイプがあります。サードミラーブレークオフという名称はミラーボリュームにスナップショットプレックス(サードミラー)を付加する実装方法に由来しています。付加されたスナップショットプレックスの内容はボリュームの元のプレックスと同期されます。スナップショットプレックスは、同期の完了後にスナップショットボリュームとして切断し、バックアップアプリケーションや意思決定支援アプリケーションで使えます。後でもとのボリュームにスナップショットプレックスを再接続するには、その内容を完全に再同期する必要があります。

もとのボリュームへの書き込みを追跡する FastResync 機能が導入されました。この追跡機能が導入されたおかげで、部分的な再同期を実行するだけで済むようになり、スナップショットプレックスの再接続が非常に短時間でできるようになりました。新しいリリースで

は、スナップショットモデルの機能が拡張され、スナップショットボリュームを複数のプレックスで構成したり、スナップショットボリュームのプレックスの一部を再接続したり、システムやクラスタの再ブート後も FastResync を維持できるようになっています。

VxVM のリリース 4.0 では、フルサイズインスタントスナップショットおよび領域最適化インスタントスナップショットが導入されています。これらのスナップショットは、すぐに使用可能で、設定と管理が容易であるなど、従来のサードミラースナップショットを上回る利点を備えています。フルサイズスナップショットではサードミラーブレークオフモデルも使えます。書き込みを集中的に行うアプリケーションではこのモデルを使う必要があります。

ボリュームスナップショットの使用方法といつ使用するかについて詳しくは、『Veritas InfoScale ソリューションガイド』を参照してください。

`vxassist`(1M)マニュアルページを参照してください。

`vxsnap`(1M)マニュアルページを参照してください。

スナップショット機能の比較

表 3-2 では、VxVM でサポートされる各種スナップショットの機能を比較します。

表 3-2　　サポートされるスナップショットの種類別の機能の比較

スナップショットの機能	フルサイズインスタントスナップショット(**vxsnap**)	領域最適化インスタントスナップショット(**vxsnap**)	ブレークオフスナップショット(**vxassist** または **vxsnap**)
作成後にすぐに使用可能	はい	はい	いいえ
必要なストレージ領域がもとのボリュームより少なくて済む	いいえ	はい	いいえ
もとのボリュームに再接続可能	はい	いいえ	はい
もとのボリュームの内容の復元に使用可能	はい	はい	はい
再接続せずに、すぐに更新可能	はい	はい	いいえ
スナップショット階層を分割可能	はい	いいえ	いいえ
もとのボリュームから別のディスクグループに移動可能	はい	いいえ	はい
独立したボリュームに変更可能	はい	いいえ	はい

スナップショットの機能	フルサイズインスタントスナップショット（vxsnap）	領域最適化インスタントスナップショット（vxsnap）	ブレークオフスナップショット（vxassist または vxsnap）
システムやクラスタを再ブートしても FastResync 機能が無効にならない	はい	はい	はい
同期処理を制御可能	はい	いいえ	いいえ
オフホストで移動可能	はい	いいえ	はい

フルサイズスナップショットは、従来のサードミラーブレークオフスナップショットよりも設定が容易で、柔軟に使えます。新しいボリュームには、vxassist コマンドを使うよりも vxsnap コマンドを使い設定されたスナップショットの方をお勧めします。インスタントスナップショットを使えるように従来のボリュームを再設定することもできますが、これには従来のサードミラースナップショットモデルを前提とした管理スクリプトを書き替える必要もあります。

アトミックな書き込みのサポート

Veritas InfoScale は、Fusion-io デバイス上でのアトミックな書き込み操作をサポートします。アトミックな書き込み対応デバイスは、書き込み I/O 操作（複数のセクターを拡張できる）のすべてのブロックが成功するか失敗することを保証します。書き込みが途中で失敗すると、ストレージは古いデータに戻ります。

アトミックな書き込みにより、更新ログバッファへの書き込みと実際のデータボリュームへのその他の書き込みという 2 つの部分の書き込みが必要になることが多い失敗した書き込みの予測できない状態の問題が解決されます。アトミックな書き込みを有効にすると、ログバッファへの書き込みがなくなり、これによりパフォーマンスが向上します。

Storage Foundation では、アトミックな書き込み対応のデバイスに Veritas Volume Manager（VxVM）ボリュームを作成するときに、アトミックな書き込みサポートを設定できます。アトミックな書き込み対応ボリュームの atomic write I/O サイズは 16KB です。

アトミックな書き込み対応ボリュームを作成している間、VxVM はすべての下位サブディスクが 16KB 境界に整列されることを保証します。アトミックな書き込み対応ボリュームを複数のアトミックな書き込み対応デバイスを拡張できますが、アトミックな書き込み境界にまたがる I/O はサポートされません。

アトミックな書き込みは、RAW VxVM ボリュームおよび VxVM ボリュームに設定された VxFS でサポートされています。

Storage Foundation の MySQL による atomic write I/O 機能について詳しくは、『Storage Foundation and High Availability Solutions ソリューションガイド』を参照してください。

FastResync

メモ: この機能を使う Veritas InfoScale Enterprise 製品のライセンスが必要です。

FastResync（以前の高速ミラー再同期、すなわち FMR）を実行すると、STALE 状態のミラー（同期化されていないミラー）が迅速かつ効率的に再同期されます。この機能により、Veritas Volume Manager（VxVM）のスナップショット機構の効率が改善され、バックアップや意思決定支援システムアプリケーションなどの操作処理速度が向上します。通常、これらの操作ではボリュームが休止状態である必要があり、システム上の他のアクティビティによるボリュームへの更新に操作が妨害されないようにする必要があります。これらの目標を達成するため、VxVM のスナップショット機構では、即座にプライマリボリュームの正確なコピーが作成されます。スナップショットが作成されると、元となるボリュームとは関係なくスナップショットにアクセスできます。

ストレージへのアクセスを共有する Cluster Volume Manager（CVM）環境では、異なるノードからスナップショットにアクセスすると、スナップショットのリソース競合とパフォーマンスの負荷を排除できます。

FastResync の動作方法

FastResync により、VxVM には次の機能強化が提供されます。

高速ミラー再同期

FastResync では、ミラー化されていなかった時の保存データの更新を記録することにより、ミラー再同期を最適化します（ミラーがボリュームから切断されて使えない状況は、エラーの結果 VxVM により自動的に切断される場合と、管理者が `vxplex` や `vxassist` などのユーティリティを使って、直接ボリュームから切断している場合があります。ミラーの回復は、`vxrecover` または `vxplex att` 操作により、以前切断された元のボリュームに再接続する処理です）。ミラーが使えるようになった任意の時点で、更新データのみを再適用して再同期させることができます。これは、保存データすべてをミラーにコピーする従来のミラー回復に比べて手間がかかりません。

FastResync がボリュームで有効になっても、ミラーの管理方法は変更されません。明らかな影響は、修復処理がより迅速に完了することのみです。

`vxplex`（1M）、`vxassist`（1M）および `vxrecover`（1M）マニュアルページを参照してください。

スナップショットの再利用	FastResync によって、スナップショットを破棄するのではなく、これを更新し再利用することができます。スナップショットプレックスを、元のボリュームに迅速に再度関連付けること（スナップバック）ができます。これによって、ボリュームスナップショットに依存するバックアップなどのサイクル処理を実行するために必要なシステムの負荷が軽減されます。

FastResync は次の 2 つの方法のいずれかで実装できます。

非永続 FastResync	非永続 FastResync は、メモリ上に FastResync で使う変更マップ（DCO マップ）を割り当てます。これらのマップは、ディスクや永続保存領域には保存されません。 p.92 の「非永続 FastResync とスナップショットの動作方法」を参照してください。
永続 FastResync	永続 FastResync はディスク上に FastResync マップを保持しているため、システムの再ブート、システムのクラッシュ、クラスタのクラッシュの場合も失われません。 p.93 の「永続 FastResync とスナップショットの連携」を参照してください。

非永続 FastResync とスナップショットの動作方法

スナップショットが作成される前に FastResync がボリューム上で有効になっている場合、VxVM のスナップショット機能は、FastResync の変更記録機能を利用して、スナップショットプレックスが作成された後、元のボリュームに対する更新を記録します。`snapback` オプションを使ってスナップショットプレックスを再接続すると、FastResync が記録する更新を適用して、スナップバック中にボリュームが再同期されます。この動作により、ボリュームの再同期に必要な時間が大幅に短縮されます。

非永続 FastResync は、メモリ内のマップを使って更新を記録します。マップは、ディスクや永続保存領域には保存されません。非永続 FastResync の利点は、ディスクが更新されないため、FastResync マップへの更新が I/O パフォーマンスに及ぼす影響を最小限に抑えられることです。ただし、FastResync はスナップショットが再接続されるまで有効であるため、システムを再起動できません。FastResync が無効になっている場合またはシステムを再起動した場合は、マップ内の情報が失われ、スナップバック実行時に全体を再同期させることが必要となります。

クラスタ共有のディスクグループ内のボリュームについては、クラスタ内の最低 1 つのノードがメモリ中に DCO マップを保存していればこのかぎりではありません。ただし、高可用性（HA）環境でノードがクラッシュした場合は、親ボリュームに再接続するときにスナップショットミラー全体を再同期させる必要があります。

FastResync マップ内の各ビットは、ボリューム内のアドレス領域の連続ブロック数を表します。マップのデフォルトサイズは 4 ブロックです。チューニングパラメータ vol_fmr_logsz を使って、マップの最大ブロックサイズを制限できます。

VxVM のチューニングについて詳しくは、『Storage Foundation and High Availability Solutions チューニングガイド』を参照してください。

永続 FastResync とスナップショットの連携

永続 FastResync はディスク上に FastResync マップを保持しているため、システムの再ブート、システムのクラッシュ、クラスタのクラッシュの場合も失われません。永続 FastResync は、ディスク上のデータ変更オブジェクト(DCO)ボリュームのマップを使って更新を記録します。マップ内の各ビットは、ボリューム内のアドレス領域の連続ブロック数を表します。

永続 FastResync は、ボリュームとそのスナップショットボリュームが異なるディスクグループに移された後も、その関係を記録できます。このため、ディスクグループが再結合された場合にはスナップショットプレックスの高速な再同期を実行できます。この機能は、非永続 FastResync ではサポートされていません。

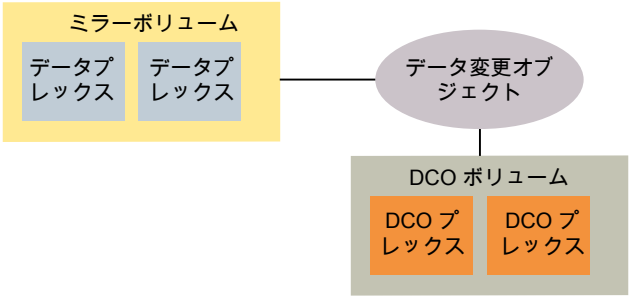
p.949 の「ディスクグループの内容の再編成」を参照してください。

永続 FastResync がボリュームまたはスナップショットボリュームで有効な場合、データ変更オブジェクト(DCO)と DCO ボリュームがボリュームに関連付けられます。

p.96 の「DCO ボリュームのバージョン管理」を参照してください。

図 3-25 に永続 FastResync が有効になっている 2 つのプレックスがあるミラーボリュームの例を示します。

図 3-25　永続 FastResync が有効になっているミラーボリューム

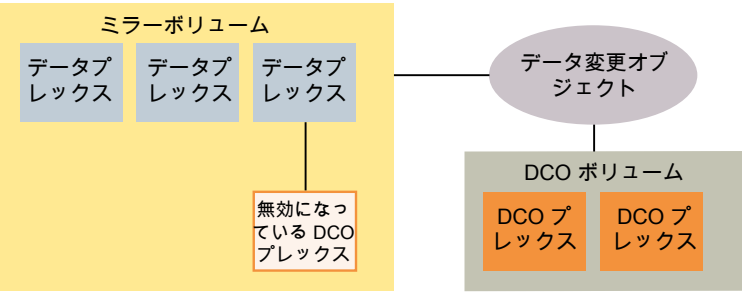


ボリュームには、DCO オブジェクトと 2 つのプレックスが存在する DCO ボリュームが関連付けられています。

vxsnap make コマンドを使ってインスタントスナップショットを作成するか、または vxassist snapstart コマンドを使って従来のサードミラースナップショットを作成します。

図 3-26 に、ボリュームでスナップショットプレックスを設定する方法と、そのスナップショットプレックスに無効な DCO プレックスを関連付ける方法を示します。

図 3-26　　スナップスタート処理完了後のミラーボリューム

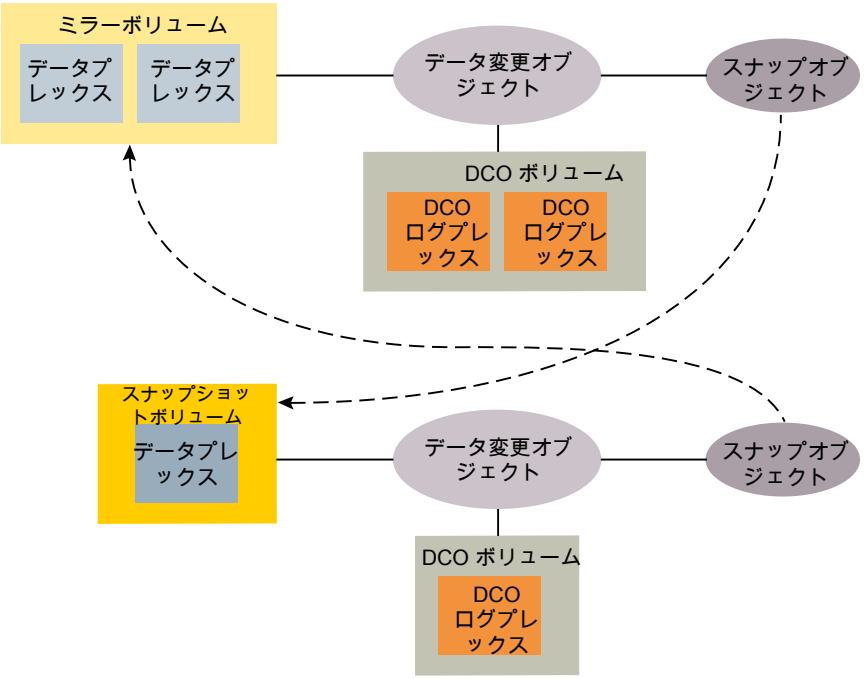


従来のスナップショットの場合は `vxassist snapstart` コマンド、領域最適化スナップショットの場合は `vxsnap make` コマンドをそれぞれ複数回実行することにより、複数のスナップショットプレックスおよび関連付けられた DCO プレックスをボリューム内に作成できます。ボリュームごとに、合計 32 個までのプレックス(データとログ)を作成できます。

`vxassist snapshot` 操作をボリュームに実行して、スナップショットプレックスから従来のスナップショットボリュームを作成できます。ただし、インスタントスナップショットでは、前手順において `vxsnap make` コマンドを使っている場合、インスタントスナップショットボリュームをすぐに使えます。コマンドを追加実行する必要はありません。

図 3-27 に、スナップショットボリュームの作成によりスナップショットボリュームに DCO オブジェクトと DCO ボリュームも設定する方法を示します。

図 3-27　スナップショット処理完了後のミラーボリュームとスナップショットボリューム



DCO ボリュームには、スナップショットプレックスと 1 対 1 で関連付けられた DCO プレックスが含まれます。スナップショットボリュームが 2 つのスナップショットプレックスから作成された場合は、DCO ボリュームに 2 つのプレックスが含まれます。領域最適化インスタントスナップショットの場合、DCO オブジェクトと DCO ボリュームは、VxVM ディスク上に作成されたボリュームではなく、キャッシュオブジェクト上で作成したスナップショットボリュームに関連付けられます。

元のボリュームとスナップショットボリュームの両方にスナップオブジェクトが関連付けられます。元のボリュームのスナップオブジェクトはスナップショットボリュームを指し、スナップショットボリュームのスナップオブジェクトは元のボリュームを指します。これによって、ボリュームとそのスナップショットが異なるディスクグループに移動されても、VxVM はその関係を記録できます。

元のボリューム内のスナップオブジェクトとスナップショットボリュームは、次のような状況では自動的に削除されます。

- 従来のスナップショットで `vxassist snapback` 操作を実行し、スナップショットボリュームのすべてのプレックスを元のボリュームに関連付ける場合。

- 従来のスナップショットで vxassist snapclear 操作を実行し、元のボリュームとスナップショットボリューム間の関連付けを解除する場合。ボリュームが異なるディスクグループにある場合、各ボリュームで個別にコマンドを実行する必要があります。
- フルサイズインスタントスナップショットで vxsnap reattach 操作を実行し、スナップショットボリュームのすべてのプレックスを元のボリュームに関連付ける場合。
- フルサイズインスタントスナップショットで vxsnap dis 操作または vxsnap split 操作をボリュームに対して実行し、元のボリュームとスナップショットボリューム間の関連付けを解除する場合。ボリュームが異なるディスクグループにある場合、各ボリュームで個別にコマンドを実行する必要があります。

メモ: vxsnap reattach、dis および split の各操作は、領域最適化インスタントスナップショットではサポートされていません。

---

vxassist(1M)マニュアルページを参照してください。

vxsnap(1M)マニュアルページを参照してください。

DCO ボリュームのバージョン管理

永続 FastResync はデータ変更オブジェクト(DCO)と DCO ボリュームを使って FastResync マップを保持します。

このリリースの Veritas Volume Manager(VxVM)は、次の DCO ボリュームバージョンをサポートします。

インスタントスナップ DCO ボリュームレイアウト	以前はバージョン 20 の DCO ボリュームレイアウトと呼ばれていましたが、このバージョンの DCO レイアウトは、ボリュームのインスタントスナップショットをサポートします。 このタイプの DCO は、FastResync マップを管理し、DRL リカバリマップとコピーマップと呼ばれる特殊なマップの管理にも使われます。コピーマップを使うと、システムがクラッシュしても、インスタントスナップショットの操作を正常に再開できます。
バージョン 0 の DCO ボリュームレイアウト	従来のスナップショット(vxassist スナップショット)をサポートするのは、このバージョンの DCO ボリュームレイアウトだけです。DCO オブジェクトは、FastResync マップに関する情報を管理します。これらのマップは、もとのボリュームと最後の snapshot 操作以降最大 32 個のスナップショットボリュームへの書き込みを記録します。ディスク上の DCO ボリュームの各プレックスには 33 個のマップが保持されます。各マップのサイズは、デフォルトでは 4 ブロックです。 VxVM ソフトウェアでは従来のバージョン 0(ゼロ)のレイアウトも引き続きサポートされます。

インスタントスナップ（バージョン 20）DCO ボリュームレイアウト

インスタントスナップデータ変更オブジェクト（DCO）は、フルサイズインスタントスナップショットと領域最適化インスタントスナップショットをサポートします。vxassist コマンドを使って管理される従来のサードミラーボリュームスナップショットはこの DCO のレイアウトではサポートされません。

Veritas Volume Manager（VxVM）4.0 で導入されたインスタントスナップ DCO ボリュームレイアウトは、バージョン 20 DCO ボリュームレイアウトともいいます。このタイプの DCO は、FastResync マップの管理だけでなく、DRL リカバリマップとコピーマップと呼ばれる特殊なマップの管理にも使われます。コピーマップを使うと、システムがクラッシュしても、インスタントスナップショットの操作を正常に再開できます。

p.86 の「DRL」を参照してください。

マップ内の各ビットは、ボリュームのアドレス空間内における領域（連続したブロックの数）と対応付けられています。各領域は、マップ内の変更を記録するボリュームの最小単位と対応付けられています。単位領域内の任意の場所で 1 バイトでも書き込みを行うと、その領域が対応付けられている空間全体に書き込んだ場合と同じように扱われます。

Storage Foundation Cluster File System High Availability 6.0 では、インスタントスナップショットの I/O パフォーマンスとスケーラビリティを向上させるため、インスタントスナップ DCO のボリュームレイアウトが変更されました。レイアウトが変更されても、インスタントスナップショットの管理方法は変わりません。明らかな影響は、I/O パフォーマンスが改善されたことです。DCO ボリュームのサイズが増加されることもあります。

インスタントスナップ DCO ボリュームのレイアウトは、事前に割り当てられたストレージ上に動的に作成されるマップを使います。DRL（Dirty Region Logging）マップのサイズは、ボリュームサイズに依存しません。DCO ボリュームを作成するとき、オプションの drlmapsz を使って DRL のサイズを設定できます。デフォルトでは、DRL のサイズは 1 MB に設定されます。

CVM 設定の場合、各ノードには、そのノードの最初書き込みのときに割り当てられる専用の DRL マップがあります。デフォルトで、DCO ボリュームには、32 個の DRL マップ、アキュムレータ、ボリュームあたり 16 個のマップ（DRL リカバリマップ、関連付けが解除されたプレックスを追跡するための切断マップ、スナップショット追跡用の残りの 14 個のマップ）を格納するだけのサイズが設定されます。

DCO プレックスのサイズは、次の式を使って見積もることができます。

```
DCO_volume_size = (32*drlmapsize + acmsize + 16*per-volume_map_size)
```

各オプションの説明

```
acmsize = (volume_size / (region_size*4))
```

```
per-volume_map_size = (volume_size/region_size*8)
```

```
drlmapsize = 1M, by default
```

100 GB ボリュームの場合、デフォルトの 64 KB の *領域サイズ*を設定した DCO ボリュームのサイズは、約 36 MB になります。

vxassist make コマンドでボリュームを作成するときに、vxsnap prepare コマンドを使って、またはオプションの logtype=dco dcoversion=20 を指定して、インスタントスナップショットの DCO を作成します。

バージョン 0 の DCO ボリュームレイアウト

バージョン 0 の DCO ボリュームレイアウトは、vxassist コマンドを使って管理される従来の(サードミラー)ボリュームスナップショットのみをサポートします。フルサイズインスタントスナップショットと領域最適化インスタントスナップショットは、この DCO レイアウトではサポートされません。

各マップのサイズは、ボリュームの作成時に vxassist コマンドに dcolen 属性を指定することによって変更できます。dcolen のデフォルトのサイズは 132 ブロックです(プレックスには 33 のマップが含まれ、各マップのサイズは 4 ブロックです)。より大きいマップサイズを使うには、希望のマップサイズに 33 をかけて、dcolen 値を計算します。たとえば、8 ブロックのマップを使うには、dcolen=264 と指定します。可能な最大マップサイズは 64 ブロックで、dcolen 値で言えば 2112 ブロックに相当します。

DCO プレックスのサイズは、ディスクグループアラインメントの整数倍に切り上げられます。CDS(Cross-platform Data Sharing)機能をサポートするディスクグループのアラインメントの値は 8 KB になります。それ以外のアラインメントの値は 1 ブロックです。

ボリュームの拡張による FastResync マップへの影響

スナップショットボリュームまたは元のボリュームを拡張した上で、FastResync を使う必要が生じることがあります。ボリュームを拡張すると、元のボリュームへの変更を記録するのに FastResync が使うマップが影響を受けますが、DCO ボリュームのレイアウトによってその影響が次のように異なってきます。

- インスタントスナップ DCO ボリュームの場合、マップサイズは拡大されますが、マップの各ビットが追跡する領域のサイズは変更されません。
- 旧形式の DCO ボリュームの場合、マップのサイズは変更されませんが、領域のサイズが拡大されます。

いずれの場合も、ボリュームの拡張エリアに対応するマップの一部は「dirty」と設定され、このエリアが再同期されます。たとえば snapback 処理は、領域不足から、不完全なスナップショットプレックスを作成しようとし、失敗することがあります。このような場合、スナップショットボリュームまたは元のボリュームを拡張してから、vxsnap reattach、vxsnap restore または vxassist snapback の各コマンドを呼び出す必要があります。2 つのボリュームを個別に拡張すると、別のミラーが存在する物理ディスクを使い、別ボリュームと物理ディスクを共有するスナップショットが作成されることになります。これを避けるには、snapback コマンドが完了した後に、ボリュームを拡張します。

vxsnap(1M)および vxassist(1M)の各マニュアルページを参照してください。

FastResync の仕様上の制限

次の仕様上の制限が FastResync に適用されます。

- RAID 5 ボリュームに対する永続 FastResync はサポートされていますが、永続 FastResync を使うと、DCO が関連付けられているボリュームでは再レイアウト操作やサイズ変更操作を使えなくなります。
- システムクラッシュ後のミラーの再同期には、非永続 FastResync、永続 FastResync のいずれも使えません。この場合、FastResync と共存できる DRL(Dirty Region Logging)を使います。VxVM リリース 4.0 以上では、DRL ログをインスタントスナップ DCO ボリューム内に保存できます。
- サブディスクが再配置されると、プレックス全体が「dirty」と設定され、全体の再同期が必要となります。
- スナップショットボリュームが別のディスクグループに分割されると、ディスクグループがもとのボリュームのディスクグループに再結合されたときに、非永続 FastResync を使ってスナップショットプレックスともとのボリュームを再同期できません。この場合は、永続 FastResync を使う必要があります。
- もとのボリューム(永続 FastResync が有効)が別のディスクグループに移動されたり分割され、スナップショットボリュームのディスクグループに移動または結合された場合は、vxassist snapback を使って従来型のスナップショットプレックスともとのボリュームを再同期できません。スナップショットボリュームが作成されたときに、スナップショットボリュームがレコード ID によってもとのボリュームを参照するため、この制限が生じます。もとのボリュームを異なるディスクグループに移動すると、ボリュームのレコード ID が変わり、関連付けが解除されます。ただし、この場合は、vxplex snapback コマンドと -f(強制)オプションを使ってスナップバックを実行できます。

  メモ: この制限は、従来型のスナップショットにのみ適用され、インスタントスナップショットには適用されません。

- スナップショットボリュームに対し、レイアウトを変更すると、そのスナップショットの FastResync 変更マップが「dirty」と設定され、スナップバック中に全体の再同期が必要となります。サブディスクの分割、サブディスクの移動、スナップショットボリュームのオンライン再レイアウトを実行するときに生じます。スナップショットが完了してからこれらの操作を実行した方が安全です。

  vxassist(1M)マニュアルページを参照してください。

  vxplex(1M)マニュアルページを参照してください。

  vxvol(1M)マニュアルページを参照してください。

オンライン再レイアウトの進行状況の制御

vxtaskコマンドを使って、再レイアウトを一時的に停止(pause)または取り消す(abort)ことができます。再レイアウト開始時に vxassist へのタスクタグを指定した場合、このタグを使って vxtask のタスクを指定できます。たとえば、myconv とタグを設定した再レイアウト操作を一時停止するには、次のように入力します。

```
# vxtask pause myconv
```

操作を再開するには、次のように vxtask コマンドを使います。

```
# vxtask resume myconv
```

vxtask pause コマンド以外の要因(vxtask abort コマンドによるタスクの停止、変換プロセスの停止、I/O エラーの発生など)で停止した再レイアウト操作を再開する場合は、次のように vxrelayout に start キーワードを指定します。

```
# vxrelayout -g mydg -o bg start vol04
```

vxtask pause コマンドで中断した再レイアウトを vxrelayout start コマンドで再開すると、操作を完了するためにタグのない新しいタスクが作成されます。このため、もとのタスクタグを使って再レイアウトを制御することはできなくなります。

-o bg オプションを指定すると、再レイアウトはバックグラウンドで再開されます。slow と iosize オプション修飾子を指定して、再レイアウトの速度やコピーされる各領域のサイズを制御することもできます。たとえば、次のコマンドでは各 10 MB 領域のコピーの間に 1000 ミリ秒(1 秒)の遅延を挿入します。

```
# vxrelayout -g mydg -o bg,slow=1000,iosize=10m start vol04
```

遅延時間のデフォルト値は 250 ミリ秒、また領域のサイズのデフォルト値は 1 MB です。

停止している再レイアウト操作の方向を逆にするには、次のように vxrelayout に reverse キーワードを指定します。

```
# vxrelayout -g mydg -o bg reverse vol04
```

これにより、これまでボリュームに対して行われた変更を元に戻し、もとのレイアウトに戻します。

vxtask abort を使って再レイアウトを取り消すことによっても、逆変換が行われ、ボリュームはもとの設定に戻ります。

vxrelayout(1M)マニュアルページを参照してください。

vxtask(1M)マニュアルページを参照してください。

VxVM のハードウェアクローンまたはスナップショットの処理方法

拡張ディスクアレイでは、ハードウェア側から物理ボリューム(ディスクまたは LUN)のコピーを作成することができます。

ハードウェアのスナップショット(EMC BCV™ または Hitachi ShadowImage™ など)、ハードウェアミラー、ハードウェアクローンを作成できます。また、dd または同じようなコマンドを使って、ディスクの内容のクローンを作成することができます。

物理ボリュームが VxVM ディスクである場合、ハードウェアコピー方式を使うと、VxVM で管理されたディスクのプライベートリージョンに格納された設定データもコピーされます。ハードウェアディスクのコピーが元の VxVM ディスクの複製になります。複製されたディスクイメージを VxVM で正しく処理するには、元のディスクイメージと複製ディスクイメージを VxVM が区別する必要があります。

VxVM はディスクがハードウェアコピーであることを検出して、複製ディスクが元のディスクと混同されていないことを確認します。この機能により、サーバーは一貫性のあるディスクセットをインポートできます。デフォルトでは VxVM によって元の物理ボリュームがインポートされますが、同じサーバー上のハードウェアコピーをユーザーが処理することもできます。VxVM にはクローンイメージを含むディスクグループをインポートして、一意の識別子を持つクローンディスクグループを作成するための特別なオプションがあります。ハードウェアコピーの複数のセットを管理する場合は、同じサーバーから管理する場合であっても、注意が必要です。

p.1000 の「ハードウェアクローンディスクを含むディスクグループのインポート」を参照してください。

VxVM はハードウェアコピーを処理する次の機能を備えています。

機能	説明
ハードウェアコピーと元のデータディスクを区別する。	VxVM はハードウェアディスクの属性から各ディスクの固有ディスク識別子(UDID)を検出して、この値を保存します。VxVM は検出された UDID と保存された値を比較して、ディスクがハードウェアコピーであるかどうかを検出します。
元の LUN の SAN を介して行われる誤った共有や、1 つ以上の PITC(Point-In-Time Copy)、ミラー、複製コピーの作成を防止する。	デフォルトでは、VxVM ディスクグループをインポートするときに、VxVM はクローンまたはコピーとして識別されたディスクのインポートを禁止します。この動作により、元のディスクとハードウェアコピーの組み合わせが誤ってインポートされることがなくなります。

機能	説明
ハードウェアコピーをクローンディスクグループまたは新しい標準ディスクグループとしてインポートする。	VxVM ディスクグループのディスクのハードウェアコピーをインポートするように選択した場合、VxVM はディスクをクローンディスクとして識別します。クローンディスクの状態を維持するのか、それとも新しい標準ディスクグループを作成するのかを選択できます。
アレイの LUN クラスを検出する。	VxVM は LUN クラスを含むアレイの拡張属性を検出します。LUN クラスは VxVM ディスクのハードウェアコピーであるディスクを識別するのに役立ちます。
ディスクセットにラベルを付けて管理するためのディスクタグ付けを行う。	同じボリュームセットの複数のコピーを作成する場合、管理者は一貫性のあるディスクセットを構成するディスクコピーを識別する必要があります。VxVM ディスクのタグを使うと、ディスクセットにラベルを付けることができます。たとえば、同じ LUN のポイントインタイムスナップショットが複数ある場合は、それぞれに異なるディスクタグを使ってラベルを付けることができます。タグの付いたスナップショット LUN をインポートする場合は、インポート操作時にタグを指定します。

VxVM の UDID の使用方法

VxVM（Veritas Volume Manager）は固有ディスク識別子（UDID）を使って VxVM ディスクのハードウェアコピーを検出します。物理ボリュームを使う前に、VxVM は必ず、ディスクに既存の UDID があるかどうか、そしてその UDID が予測値と一致するかどうかを確認します。

VxVM ディスクを初期化するときに、VxVM のデバイス検出層（DDL）はベンダーID（VID）、プロダクトID（PID）、キャビネットのシリアル番号、LUN のシリアル番号などのハードウェア属性から UDID を判別します。VxVM が UDID を持たないディスクを最初に認識するとき、または VxVM がディスクを初期化するときは、ディスクのプライベートリージョンに UDID が格納されます。UDID の正確な構成はアレイストレージライブラリ（ASL）によって決まります。VxVM の今後のバージョンでは、新しいアレイにさまざまな形式が使われる可能性があります。

UDID が設定されたディスクを検出すると、VxVM は現在の UDID 値（ハードウェア属性から判別された値）とディスクに格納済みの UDID を比較します。DDL で判別された UDID 値とディスク上の UDID の値が一致しない場合、VxVM はディスクに `udid_mismatch` フラグを設定します。

`udid_mismatch` フラグは一般に、ディスクが VxVM ディスクのハードウェアコピーであることを示します。ハードウェアコピーには、UDID を含む、元のディスクの VxVM プライ

ベートリージョンのコピーが含まれます。VxVM プライベートリージョンにすでに格納されている UDID は元のハードウェアディスクの属性と一致しますが、コピーであるハードウェアディスクの値とは一致しません。

UDID 一致機能があるため、VxVM はホストに提供されるディスクセットに一貫性がない状況を回避することができます。この機能を使うと、元の LUN と同じホスト上の LUN スナップショットで構成されるディスクグループをインポートすることができます。udid_mismatch フラグで識別されるディスクをインポートすると、VxVM はディスクに clone_disk フラグを設定します。元の LUN と同じホストで、元の LUN の複数のハードウェアイメージを同時に管理して、インポートする場合は、注意して行ってください。

p.1000 の「ハードウェアクローンディスクを含むディスクグループのインポート」を参照してください。

システムがコピー(クローン)デバイスのみを認識する場合は、clone_disk フラグを削除できます。リスクがないと確信できる場合のみ、clone_disk フラグを削除してください。たとえば、異なる時刻に同じ物理ベースボリュームのコピーである物理ボリュームが 2 つ存在しないことを確認する必要があります。

クローン以外のディスクに udid_mismatch フラグが正しく設定されていない場合は、udid_mismatch フラグを削除し、ディスクを標準ディスクとして処理することができます。

詳しくは、『Veritas InfoScale トラブルシューティングガイド』を参照してください。

ボリュームの暗号化

VxVM では、VxVM データボリュームの暗号化によって保存データに対する高度なセキュリティが提供されます。暗号化とは、データまたは情報を許可されたユーザーによってのみ解読可能なコードに変換する技術です。

Veritas InfoScale は、転送中ではないデータの暗号化と転送中のデータの暗号化をサポートします。これは、暗号化されたボリュームのバックアップを定期的に作成するために役立ちます。『Veritas InfoScale™ 7.3.1 レプリケーション管理者ガイド』を参照してください。

VxVM データボリュームは次の目的で暗号化することができます。

- 機密データを不正なアクセスから保護する
- ディスクの使用停止、またはディスク内容の安全な消去なしでの交換用出荷

暗号化の実装には、FIPS(Federal Information Processing Standard)Publication 140-2(FIPS PUB 140-2)セキュリティ基準によって検証された 256 ビットキーサイズの AES(Advanced Encryption Standard)暗号アルゴリズムを使用します。

ストレージ環境でボリュームまたはディスクグループを暗号化できます。VxVM では、ボリューム作成時にボリューム暗号化キーが生成されます。ボリューム暗号化キーは、キーラップによって保護 (ラップ) されます。ラップされたキーはボリュームレコードとともに保存されます。ボリューム暗号化キーは、ディスクには保存されません。

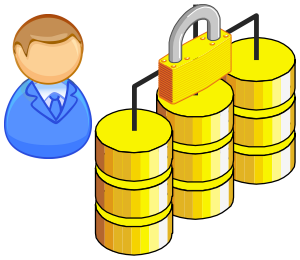
次の方法のいずれかを使って、ボリューム暗号化キーを保護できます。

パスフレーズ (PBE) の使用	p.108 の「パスフレーズを暗号化に使用」を参照してください。
Key Management Server (KMS) の使用	p.108 の「暗号化のための Key Management Server (KMS) の使用」を参照してください。

図 3-28 は暗号化プロセスを示しています。

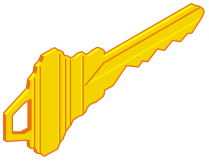
図 3-28 暗号化

1 ボリューム作成時に暗号化属性を設定。



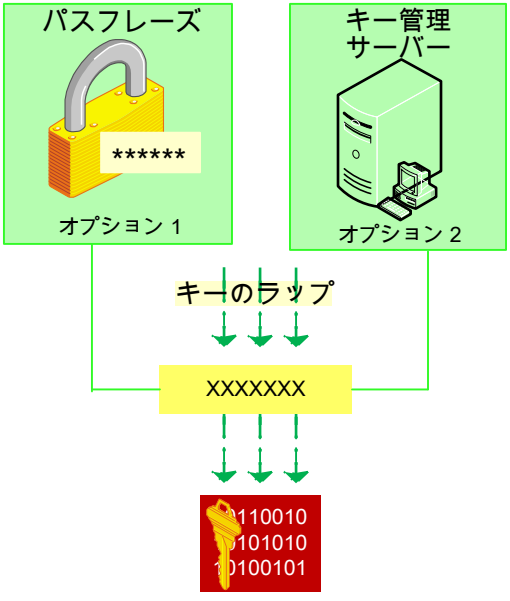
暗号化済みボリューム
encrypted=on

2 VxVM が暗号化キーを生成。



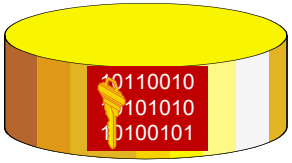
ボリューム暗号化キー

3 VxVM が次の 2 つのメカニズムのいずれかを用いて暗号化キーのセキュリティを確保。パスフレーズまたはキー管理サーバー。



ラップされたキー

4 VxVM がボリュームレコードでラップされたキーを保存。



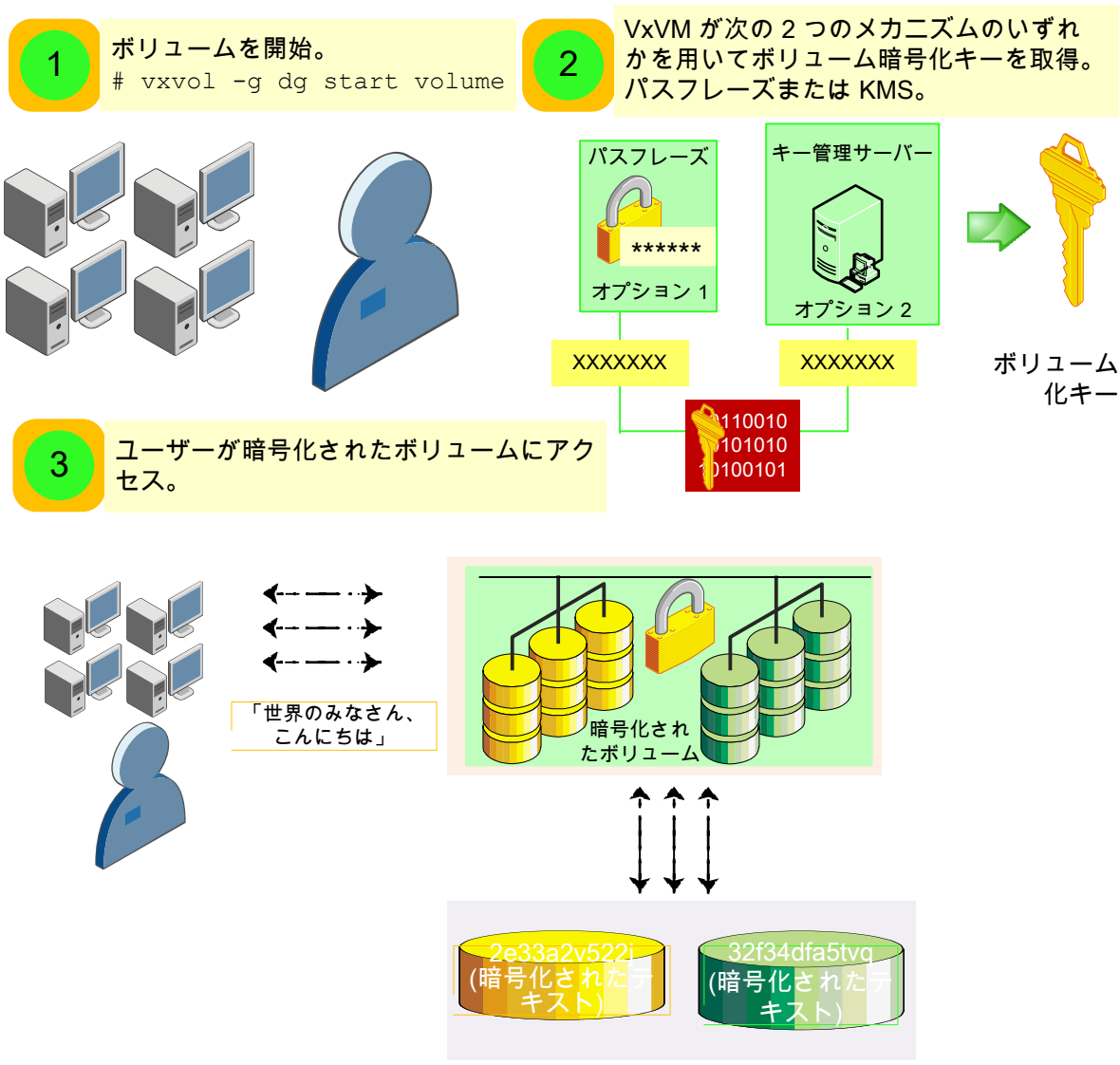
ディスクグループを暗号化する場合、ディスクグループ内のすべてのボリュームが暗号化されます。ディスクグループ内で後で作成されたボリュームもデフォルトで暗号化されます。

ディスクグループバージョン 220 以降を使用して作成された新しいボリュームのみを VxVM によって暗号化できます。

暗号化されたボリュームを開始すると、VxVM ではキーラップを使って、ボリューム暗号化キーが取得され、ボリュームへのアクセスが有効になります。

図 3-29 は解読プロセスを示しています。

図 3-29 解読



次の機能は、VxVM 暗号化でサポートされていません。

- ルートおよびスワップボリュームの暗号化
- RAID-5 レイアウトを使用するボリュームの暗号化
- 既存のボリュームの暗号化

- リンクされたブレークオフスナップショット

パスフレーズを暗号化に使用

ボリュームを暗号化する場合、VxVM でボリューム暗号化キーが生成されます。ボリューム暗号化キーはキーラップを使用して保護する必要があります。パスフレーズメカニズムを使用する場合、VxVM ではパスフレーズの入力を求めるメッセージが表示され、ハッシュアルゴリズムを使用して指定したパスフレーズからキーラップが派生されます。このメカニズムの使用には、追加のハードウェアまたはソフトウェアは不要です。パスフレーズはランダムに生成される必要があり、高エントロピーを含む必要があります。

パスフレーズを使用して暗号化されるボリュームは、システムのブートまたは再起動時に常に手動で起動する必要があります。これは、システム起動時に認証の入力を求めるメッセージがボリュームで表示されるためです。ただし、ファイルに必要なパスフレーズを入力し、暗号化されたボリュームに対し自動起動を有効にすることができます。

パスフレーズベースの暗号化は、自動設定に大きく依存しない環境に適しています。

暗号化のための Key Management Server (KMS) の使用

VxVM は、OASIS KMIP (Key Management Interoperability Protocol) 仕様に適合した Key Management Server (KMS) の使用をサポートしています。

暗号化されたボリュームの作成時:

- VxVM は KMIP を使って設定済みの KMS にキー生成の要求を送信します。
- KMS は一意の識別子で応答します。VxVM は KMS で生成したキーを取得するために識別子を KMS に送信します。
- KMS はキーで応答します。VxVM はランダムなボリューム暗号化キーを生成し、それを KMS が提供したキーを使って暗号化します。
- VxVM はボリュームレコードに暗号化したキーと KMS 識別子を格納します。

暗号化したボリュームの起動時:

- VxVM はボリュームレコードから暗号化したキーと KMS 識別子を取得します。
- VxVM はキーを取得するために識別子を KMS に送信します。
- KMS はキーで応答します。VxVM は (ボリュームレコードに格納されている) 暗号化したキーを KMS が提供したキーで解読します。

KMS ベースの暗号化は、高可用性および自動設定がサポートされている環境に適しています。

Key Management Server を使って、次のことを実行できます。

- 複雑なパスフレーズを覚える必要がない
- ディザスタリカバリ用のキーのバックアップまたはレプリケーション

- 単一の KMS キーを使用して、ディスクグループ内のすべてのボリュームのボリューム暗号化キーを暗号化するか取得します。
- ボリュームの KMS キーの変更またはローテーション

VxVM では、OASIS KMIP 仕様に適合した Key Management Server がサポートされています。

VxVM では、KMIP クライアントにあるファイル `/etc/vx/enc-kms-kmip.conf` の設定情報を使ってサーバーが設定されます。

p.252 の「Key Management Server の設定」を参照してください。

暗号化の推奨事項

パフォーマンス向上のため、Advanced Encryption Standard Instruction Set (または Intel Advanced Encryption Standard New Instructions (AES-NI)) をサポートするように設計された CPU を使用することをお勧めします。

次のコマンドを使って、プロセッサで暗号化アクセラレーションがサポートされているかどうかを確認できます。

```
$ grep -o aes /proc/cpuinfo
aes
aes
aes
aes
```

コマンドによって出力が生成されなければ、プロセッサで暗号化アクセラレーションがサポートされていません。

ディスクグループレベルの暗号化キーの管理とキーのローテーション (再キー)

InfoScale は、ディスクグループ内のすべてのボリュームに対して単一の KMS キーの使用をサポートします。したがって、各ボリュームの KMS キーを個別に維持するのではなく、ディスクグループレベルで共通の KMS キーを維持できます。ディスクグループと共通の KMS キーがある暗号化ボリュームを開始する場合、VxVM はボリュームへのアクセスを有効にするために 1 つのキーのみをフェッチする必要があります。したがって、共通 KMS キーは、ボリュームの数に基づいて複数の要求の形式で KMS に送信されるネットワーク負荷を軽減します。KMS に対する 1 つの要求によって、すべてのボリュームを 1 回の操作で開始できます。

この単一キーの使用をより安全にするために、InfoScale では、必要に応じて KMS キーを変更するボリュームを再キー化するオプションを提供します。このオプションは、キーローテーションとも呼ばれます。キーの再実行操作をスケジュールするために、ポリシーに基づいて外部スケジューラを使うことができます。

ディスクグループ内のすべての暗号化ボリュームに対して単一のキーを使用するには、same_enckey チューニングパラメータの値を次のように yes に設定します。

ディスクグループの作成時に、次のように設定します。

```
vxdg -o same_enckey=yes init DiskGroupName diskName1 diskName2 …
diskNameN
```

キーの再操作は次のように動作します。

- ボリューム暗号化キーは変更されません。
- 既存の KMS キーを使用して暗号化されたボリューム暗号化キーを取得し、新しい KMS キーを使用して再び暗号化します。
- 新しく暗号化されたボリューム暗号化キーと、変更された KMS キーの新しい KMS 識別子をボリュームレコードに格納します。

メモ: ディスクグループレベルの暗号化キー管理およびキーローテーション機能は、結合、分割、移動などの VVR 構成とディスクグループ操作をサポートしません。

4

Veritas File System の動作

この章では以下の項目について説明しています。

Veritas File System の機能

表 4-1には、Veritas File System (VxFS) 機能が一覧表示されます。

以下の表に Veritas File System (VxFS) の機能の一覧を示します。表の説明では各機能が SFCFSHA をサポートしているかどうかも示しています。

表 4-1 Veritas File System の機能

機能	説明
アクセス制御リスト	ACL (アクセス制御リスト) には、ユーザーやグループ、およびディレクトリやファイルに対するアクセス権限が定義されています。ACL はファイルごとに定義することもできれば、複数のファイルで共有することもできます。ACL では複数のユーザーやグループに対してアクセス権限を詳細に指定できます。 Linux のクラスタファイルシステムでは、ACL がサポートされています。 この機能は SFCFSHA でサポートされます。 `getfacl`(1) と `setfacl`(1) のマニュアルページを参照してください。

機能	説明
キャッシュアドバイザリ	キャッシュアドバイザリは個別のファイルシステム上で mount コマンドによって設定されますが、クラスタの他のノードには伝達されません。 mount コマンドだけではキャッシュアドバイザリを設定できません。キャッシュアドバイザリはファイルごとに設定できます (VX_SETCACHE ioctl を使います)。 p.620 の「Veritas File System I/O について」を参照してください。
ファイルアクセス時間に依存するコマンド	クラスタファイルシステムでは atime ファイル属性が厳密に同期されないため、ファイルアクセス時間の表示がノード間で異なる場合があります。そのため、アクセス時間のチェックに依存するユーティリティは確実に機能しない場合があります。
CDS (Cross-Platform Data Sharing)	CDS (Cross-platform data sharing) を使うと、異種システム間でデータを連続して共有し、そのデータを保持する物理デバイスに各システムが直接アクセスできるようになります。この機能は、VxVM (Veritas Volume Manager) とともに使う場合にかぎり、使えます。 この機能は SFCFSHA でサポートされます。 『Veritas InfoScale ソリューションガイド』を参照してください。
データの重複排除	継続的な費用をかけずに重複したデータを排除するために、ファイルシステムではプロセス後の定期的な重複排除を実行できます。データがオンデマンドで複製されたかどうかを確認し、その後に効率的かつ安全に重複を排除します。この機能は Veritas InfoScale Storage ライセンスと Veritas InfoScale Enterprise ライセンスの両方で使用できます。 この機能は SFCFSHA でサポートされます。 p.921 の「データの重複排除について」を参照してください。
断片化解消	ディレクトリから未使用領域を取り除き、すべての小さいファイルを連続させ、空きブロックを統合してファイルシステムで使用するために断片化解消を実行できます。 この機能は SFCFSHA でサポートされます。 p.126 の「断片化の解消について」を参照してください。

機能	説明
拡張データ整合性モード	VxFS には拡張データ整合性モードを有効にするための次の mount コマンドオプションがあります。 ■ blkclear p.267 の「blkclear マウントオプション」を参照してください。 ■ closesync p.267 の「mincache マウントオプション」を参照してください。 ■ log p.265 の「log マウントオプション」を参照してください。 この機能は SFCFSHA でサポートされます。
拡張パフォーマンスモード	VxFS のデフォルトログモード (mount -o delaylog) では、一部の構造上の変更に関するログを遅延させることで、処理効率が向上します。ただし、delaylog では拡張データ整合性モードと同じデータ整合性は提供されません。理由は、システムエラー時に直前の変更が失われることがあるためです。このオプションを使うことで、少なくとも従来の UNIX ファイルシステムと同程度のデータ整合性と、ファイルシステムの高速リカバリを実現できます。 mount_vxfs(1M) のマニュアルページを参照してください。 p.265 の「delaylog マウントオプション」を参照してください。
強化されたセキュリティ	RHEL は、ファイルシステムに対してユーザーレベルのセキュリティ機能を提供します。これらのセキュリティ機能は、OS レベルで SELinux を有効にすると利用できます。 mount_vxfs(1M) のマニュアルページを参照してください。
エクステント属性	VxFS では、エクステントと呼ばれる 1 つ以上の隣接するブロックから構成されるグループ単位で、ファイルにディスク領域が割り当てられます。VxFS は、プログラムが特定のファイルのエクステント割り当ての各種状態を制御できるようにするアプリケーションインターフェースを定義します。VxFS のアプリケーションインターフェースを使うと、プログラムで任意のファイルに対するエクステント割り当てをより細かく制御できます。 この機能は SFCFSHA でサポートされます。 p.283 の「エクステント属性について」を参照してください。

機能	説明
エクステントベースの割り当て	エクステントは、コンピュータファイルシステム内の連続したストレージ領域で、ファイル用に予約された領域です。ファイルに対する書き込みを開始すると、エクステント全体が割り当てられます。ファイルに再び書き込みを行うと、前回の書き込み場所に続けてデータが書き込まれます。これにより、ファイルの断片化を少なくするかまたは回避できます。エクステントは「アドレスと長さの組み合わせ」で表現され、これによって開始ブロックのアドレスと、(ファイルシステムまたは論理ブロックの) エクステントの長さが決まります。VxFS は、エクステントベースのファイルシステムのため、エクステント (複数ブロックで構成可能) を使ってアドレス指定します。単一ブロックセグメントにアドレス指定するのではありません。したがって、エクステントを使うとファイルシステムのスループットを向上できます。 この機能は SFCFSHA でサポートされます。 p.37 の「エクステントについて」を参照してください。
マウントの拡張オプション	VxFS ファイルシステムでは、`mount` コマンドが次のように強化されています。 ■ 拡張データ整合性モード ■ 拡張パフォーマンスモード ■ 一時ファイルシステムモード ■ 同期書き込みの改善 ■ 大容量ファイルサイズのサポート この機能は SFCFSHA でサポートされます。 p.262 の「VxFS ファイルシステムのマウント」を参照してください。
ファイルシステムの高速リカバリ	通常のファイルシステムはシステム障害からリカバリする際、そのための唯一の手段である `fsck` ユーティリティによるファイルシステム構造全体の検証に依存しています。ディスク構成の規模が大きい場合には、このユーティリティによる構造全体の調査、ファイルシステムの整合性の検証および不整合部分の修正の処理には、必然的に相当の時間が必要になります。VxFS は、VxFS インテントログと VxFS インテントログのサイズ変更機能で高速リカバリを実現します。 この機能は SFCFSHA でサポートされます。 p.36 の「Veritas File System のインテントログについて」を参照してください。

機能	説明
FCL (File Change Log)	VxFS FCL (File Change Log) は、ファイルシステム内のファイルおよびディレクトリへの変更を記録します。FCL は、バックアップ製品、Web 巡回ロボット、検索および索引エンジン、レプリケーションソフトウェアなど、ファイルシステム全体をスキャンして、前回のスキャン以降に変更された箇所を検出するアプリケーションによって使われます。FCL 機能は Veritas InfoScale™ Storage、Veritas InfoScale™ Availability、Veritas InfoScale™ Foundation、および Veritas InfoScale™ Enterprise の 4 つの Veritas InfoScale ライセンスで使用できます。 この機能は SFCFSHA でサポートされます。 p.1100 の「Veritas File System ファイルの変更ログについて」を参照してください。
ファイル圧縮	ファイルを圧縮すると、ファイルのアクセシビリティを維持しながら、アプリケーションに対して透過的に、使用する領域を減らすことができます。圧縮されたファイルの外観と動作は圧縮前のファイルとほとんど変わりません。同じファイル名で、圧縮前のファイルと同様に読み取りおよび書き込みできます。読み取りを行うと、メモリではデータが圧縮解除されます。ただし、ファイルのディスク上のコピーは圧縮された状態のままです。これに対し、書き込み後の新しいデータは、ディスク上で圧縮されていません。 この機能は SFCFSHA でサポートされます。 p.932 の「圧縮ファイルについて」を参照してください。
ファイルレプリケーション	コスト効率の高い定期的なデータレプリケーションを IP ネットワーク上で実行し、ディザスタリカバリとオフホスト処理に対して非常に柔軟なストレージ非依存型のデータ可用性ソリューションを組織に提供できます。 この機能は SFCFSHA でサポートされます。 『Veritas InfoScale レプリケーション管理者ガイド』を参照してください。

機能	説明
ファイルシステムのスナップショット	VxFS では、スナップショット機能を使ったオンラインデータバックアップが提供されています。この機能では、マウントされているファイルシステムのイメージに対して、特定時点で読み取り専用のコピーが瞬時に作成されます。元のファイルシステムは、スナップファイルシステムと呼ばれ、コピーは、スナップショットと呼ばれます。 スナップファイルシステムに変更が加えられると、古いデータはスナップショットにコピーされます。スナップショットの読み取りでは、変更されていないデータはスナップファイルシステムから読み取られ、変更されたデータはスナップショットから読み取られます。 バックアップは、次のいずれかの方法で実行します。 ■ スナップショットファイルシステムから選択したファイルをコピーする (`find` と `cpio` を使用)。 ■ ファイルシステム全体のバックアップを作成する (`fscat` を使用)。 ■ 完全または増分バックアップを実行する (`vxdump` を使用)。 この機能は SFCFSHA でサポートされます。 p.678 の「スナップショットファイルシステムについて」を参照してください。
FileSnap	FileSnap は、同じ名前空間のファイルの領域最適化コピーであり、同じファイルシステムに保存されます。VxFS は、ディスクレイアウトバージョン 8 以降のファイルシステムの FileSnap をサポートします。 この機能は SFCFSHA でサポートされます。 p.675 の「FileSnap について」を参照してください。
ファイルシステムのフリーズとアンフリーズ	ファイルシステムのフリーズ処理は、ファイルシステムの、ボリュームレベルで一貫性のある安定したイメージを取得するために必要な手順です。ボリュームレベルで一貫性のあるファイルシステムイメージは、ファイルシステムスナップショットツールを使って取得し、使うことができます。 この機能は SFCFSHA でサポートされます。ファイルシステムのフリーズとアンフリーズを必要とする同期操作はクラスタ全体に対して行われます。 p.624 の「ファイルシステムのフリーズとアンフリーズ」を参照してください。

機能	説明
同期書き込みの改善	VxFS では、同期書き込みを実行するアプリケーションのパフォーマンスを向上できます。mount -o datainlog オプションを使うと、同期書き込みが少量の場合のパフォーマンスが著しく向上します。 mount -o convosync=dsync オプションを使うと、同期データ書き込みが必要なアプリケーションのパフォーマンスは向上しますが、同期を行う i ノードアクセス時間の更新パフォーマンスは変わりません。 mount_vxfs(1M) のマニュアルページを参照してください。 **警告:** -o convosync=dsync オプションを使うと、POSIX セマンティクスに違反する場合があります。 p.269 の「convosync マウントオプション」を参照してください。
ロック	F_GETLK コマンドについては、競合ロックを保持しているプロセスがある場合、l_pid フィールドは競合ロックを保持しているプロセスのプロセス ID を返します。ノード ID からノード名への変換は、/etc/llthosts ファイルの検証または fsclustadm コマンドの使用で行うことができます。 この機能は必須のロックおよび従来の fcntl ロックでサポートされるデッドロック検出を除き、SFCFSHA でサポートされます。 fcntl(2) のマニュアルページを参照してください。

機能	説明

maxlink サポート	64K サブディレクトリよりも多くのサポートが追加されました。ファイルシステムで maxlink が無効になっている場合、サブディレクトリの制限はデフォルトでは 32K です。ファイルシステムで maxlink が有効になっている場合、4294967295(2^32 – 1) までのサブディレクトリを作成することが可能です。
	
	デフォルトでは maxlink は有効になっています。
	
	maxlink オプションを mkfs 時に有効にするには、次に例を示します。
	
	`# mkfs -t vxfs -o maxlink /dev/vx/rdsk/testdg/vol1`
	
	maxlink オプションを mkfs 時に無効にするには、次に例を示します。
	
	`# mkfs -t vxfs -o nomaxlink /dev/vx/rdsk/testdg/vol1`
	
	maxlink オプションをマウント済みファイルシステムの fsadm コマンドによって有効にするには、次に例を示します。
	
	`# fsadm -t vxfs -o maxlink /mnt1`
	
	maxlink オプションをマウント済みファイルシステムの fsadm コマンドによって無効にするには、次に例を示します。
	
	`# fsadm -t vxfs -o nomaxlink /mnt1`
	
	mkfs_vxfs(1M) マニュアルページと fsadm_vxfs(1M) マニュアルページを参照してください。

メモリマッピング	共有メモリのマッピングを確立するには mmap() 関数を使うことができます。 この機能は SFCFSHA でサポートされます。 mmap(2) のマニュアルページを参照してください。

機能	説明
MVS ファイルシステム	MVFS (multi-volume file system) 機能を使って、複数のボリュームを 1 つの論理オブジェクトで表すことができます。下位論理ボリュームへのすべての I/O は、ボリュームセットを介して指示されます。この MVS に 1 つの VxFS ファイルシステムを作成できます。この機能は、VxVM とともに使う場合にかぎり、使用可能です。MVFS 機能は Veritas InfoScale™ Storage、Veritas InfoScale™ Availability、Veritas InfoScale™ Foundation、および Veritas InfoScale™ Enterprise の 4 つの Veritas InfoScale ライセンスで使用できます。 p.815 の「MVS ファイルシステムについて」を参照してください。
ネストマウント	クラスタマウントまたはローカルマウントされたファイルシステム上のディレクトリを、ローカルファイルシステムまたは別のクラスタファイルシステムのマウントポイントとして使うことができます。 この機能は SFCFSHA でサポートされます。
NFS マウント	クラスタから NFS ファイルシステムをエクスポートします。分散型の高可用性の方法で CFS ファイルシステムを NFS エクスポートできます。 この機能は SFCFSHA でサポートされます。
パーティションディレクトリ	大きなボリュームにアクセスする、ファイルシステム内に存在するディレクトリにアクセスしアップデートを実行する並列スレッドでは、非常に長い待ち時間が発生します。 この機能では、ディレクトリをパーティション分割して、ファイルシステムのディレクトリのパフォーマンスを向上します。任意のディレクトリがチューニング可能なしきい値を超えると、ディレクトリの i ノードに排他ロックを実行して、エントリを異なるハッシュディレクトリにそれぞれ再分散します。これらのハッシュディレクトリは、ユーザーの名前空間ビューやオペレーティングシステムでは表示されません。この機能は、すべての新規作成、削除、検索スレッドに対して、それぞれのハッシュディレクトリ (ターゲットの名前により決定) を検索し、そのディレクトリで操作を実行します。これにより、親ディレクトリの inode およびその他のハッシュディレクトリへのアクセスが妨げられることがなくなり、ファイルシステムのパフォーマンスが大幅に向上します。 この機能はディスクレイアウトバージョン 8 以降のファイルシステムでのみ動作します。 この機能は SFCFSHA でサポートされます。 p.1113 の「パーティションディレクトリ」を参照してください。 `vxtunefs`(1M) と `fsadm_vxfs`(1M) のマニュアルページを参照してください。

機能	説明
クォータ	VxFS では、クォータがサポートされています。これにより、特定のユーザーとグループ単位でクォータを割り当てて、ファイルとデータブロックという 2 つの主要リソースの使用を制限します。これらのリソースごとにクォータを割り当てることができます。各クォータでは、それぞれのリソースについてハード制限とソフト制限という 2 タイプの制限を設けています。 ハード制限は、データブロックやファイルの絶対的な制限を表します。ユーザーは、どのような場合でも、ハード制限を超えたリソースを使うことはできません。 ソフト制限は、ハード制限より低い値であり、特定の時間内であれば、この制限を超えることができます。すなわち、特定の時間内であれば、ユーザーは一時的にこの制限を超えてリソースを使えます。 この機能は SFCFSHA でサポートされます。 p.1092 の「Veritas File System のクォータ制限について」を参照してください。
パス名の逆引きルックアップ	パス名の逆引きルックアップ機能を使うと、ファイルやディレクトリの i ノード番号から、完全パス名を取得できます。パス名の逆引きルックアップ機能は、VxFS FCL 機能のクライアント、バックアップユーティリティ、リストアユーティリティ、レプリケーション製品など、さまざまなアプリケーションで使えます。ファイルやディレクトリのパス名は非常に長くなることがあるため、これらのアプリケーションでは、情報を i ノード番号で保存することが普通です。そのため、パス名を簡単に取得する手段が必要になります。 この機能は SFCFSHA でサポートされます。 p.1109 の「パス名の逆引きルックアップについて」を参照してください。

機能	説明
SmartIO	Storage Foundation and High Availability Solutions (SFHA Solutions) の SmartIO 機能は、I/O キャッシュを使用して SSD または他のサポートデバイスのデータ効率を向上させます。効率を改善する SmartIO を使用して、IOPS ごとのコストを最適化できます。SmartIO は、高度でカスタマイズ可能なヒューリスティックを使って、キャッシュに保存するデータ、そのデータをキャッシュから削除する方法を決定します。このヒューリスティックでは、ワークフローの特性に関する SFHA Solutions の知識が活用されます。 SmartIO はターゲットデバイスまたはデバイスのキャッシュ領域を使います。キャッシュ領域は、SmartIO がキャッシュしたデータとそのデータに関するメタデータを格納するために使うストレージ領域です。キャッシュ領域のタイプに応じて、VxFS キャッシュまたは VxVM キャッシュのいずれかがサポートされるかが決定します。 この機能は SFCFSHA でサポートされます。 『Veritas InfoScale SmartIO for Solid-State Drives ソリューションガイド』を参照してください。
SmartTier	SmartTier オプションは、マルチボリュームファイルシステムに基づいています。SmartTier を使うと、2 つ以上のボリュームを 1 つのファイルシステムにマップできます。さらに、ファイルを別のボリュームに自動的に再配置するようにポリシーを設定したり、ファイル再配置コマンドを実行してファイルを再配置することができます。使う複数のボリュームに対して、ファイルの配置場所を選択できるようになります。これにより、特定のタイプのファイルにアクセスするアプリケーションのパフォーマンスを向上させることができます。SmartTier 機能は Veritas InfoScale Storage ライセンスと Veritas InfoScale Enterprise ライセンスの両方で使用できます。 **メモ:** 以前の VxFS 5.x リリースでは、SmartTier は DST (Dynamic Storage Tiering) と呼ばれていました。 この機能は SFCFSHA でサポートされます。 p.831 の「SmartTier について」を参照してください。

機能	説明
Storage Checkpoint	VxFS では、可用性、障害からのリカバリおよびパフォーマンスを向上させるために、ディスクによるバックアップとオンラインのバックアップおよびリストア機能が提供されており、頻繁に効率よくバックアップを実行することができます。バックアップアプリケーションやリストアアプリケーションでは、Storage Checkpoint を利用できます。これは、ディスクと I/O の効果的なコピー技術であり、ファイルシステムのフリーズしたイメージを定期的に作成します。Storage Checkpoint では、ある時点におけるファイルシステムのビュー (構造) を提示し、その後は元のファイルシステムブロックのコピーを識別して保持します。ディスクベースのミラー化を使う代わりに、Storage Checkpoint では、ファイルシステムで使用可能な空き領域プールを使うことによって、ディスク領域の消費を抑制し、I/O のオーバーヘッドを大幅に削減します。 Storage Checkpoint 機能は Veritas InfoScale Storage ライセンスと Veritas InfoScale Enterprise ライセンスの両方で使用できます。 この機能は SFCFSHA でサポートされます。 p.737 の「Storage Checkpoint について」を参照してください。
大容量ファイルのサポートと大容量ファイルシステム	VxFS は 2 GB 以上のファイルと、最大で 256 TB までの大容量ファイルシステムをサポートします。 **警告:** アプリケーションやユーティリティには、大容量ファイルに対応していないものもあります。 p.271 の「largefiles と nolargefiles マウントオプション」を参照してください。
スワップファイル	スワップファイルはクラスタマウントされたファイルシステムでサポートされません。
一時ファイルシステムモード	通常の UNIX システムでは、一時ファイルシステムディレクトリ (`/tmp` や `/usr/tmp` など) に、システムの再起動時に保持する必要がないファイルが保存されます。これらは一時的なディレクトリなので、その中に存在するファイルシステム構造の整合性を完全に保つ必要はありません。VxFS の `mount -o tmplog` オプションを使うと、多くの操作のログを遅延させることによって、一時ファイルシステムのパフォーマンスが向上します。 `mount_vxfs`(1M) のマニュアルページを参照してください。 p.266 の「tmplog マウントオプション」を参照してください。

機能	説明
シン再生	シン再生機能により、VxFS ファイルシステムの空きデータブロックを、シンストレージ LUN の空きストレージプールに解放できます。この機能は VxVM ボリュームに作成されたファイルシステム上でのみサポートされます。 p.783 の「ファイルシステムのシン再生について」を参照してください。

Veritas File System のパフォーマンスの向上

従来のファイルはブロック単位の割り当てスキームを使ったものです。この割り当てポリシーでは、小規模なファイルに対して十分なランダムアクセスと遅延が提供されますが、大容量ファイルのスループットに制限があります。そのため、従来のファイルは、ミッションクリティカルな処理環境では必ずしも最適なものではありません。

Veritas File System（VxFS）では、UFS とは異なる割り当て方法が使われています。そのため、割り当て、I/O およびキャッシュポリシーをより細かく制御できます。

p.126 の「Veritas File System の使用」を参照してください。

VxFS では、次の機能が強化されています。

- データ同期 I/O
  p.622 の「データ同期 I/O」を参照してください。
- ダイレクト I/O およびディスカバードダイレクト I/O
  p.621 の「ダイレクト I/O」を参照してください。
  p.622 の「ディスカバードダイレクト I/O」を参照してください。
- 拡張書き込み用の遅延割り当て
  p.125 の「拡張書き込み用の遅延割り当て」を参照してください。
- I/O パフォーマンスの向上
  p.124 の「I/O パフォーマンスの向上」を参照してください。
- キャッシュアドバイザリ
  p.624 の「キャッシュアドバイザリ」を参照してください。
- ディレクトリ拡張機能
- ファイルのアラインメント、エクステントサイズおよび事前割り当ての明示的な制御
  p.286 の「エクステント属性のアラインメント」を参照してください。
  p.284 の「固定エクステントサイズ」を参照してください。
  p.284 の「領域予約: ファイルへの事前領域割り当て」を参照してください。
- I/O チューニングパラメータ
  p.1112 の「VxFS ファイルシステムのチューニング」を参照してください。

- Veritas Volume Manager（VxVM）との統合
  p.34 の「Veritas Volume Manager の概要」を参照してください。
- 大容量ディレクトリのサポート

  メモ: VxFS では、大量のファイルが入ったディレクトリのファイルルックアップ時間を短縮します。

- パーティションディレクトリ
  `vxtunefs`(1M) と `fsadm_vxfs`(1M) のマニュアルページを参照してください。

I/O パフォーマンスの向上

Veritas File System（VxFS）では、I/O クラスタ化ポリシーを積極的に適用し、Veritas Volume Manager（VxVM）との統合を実行し、ファイルシステムごとにアプリケーション固有のパラメータ設定を許すことによって、I/O パフォーマンスが向上しています。

p.124 の「拡張 I/O クラスタ化機能」を参照してください。

p.124 の「拡張された I/O パフォーマンスのための Veritas Volume Manager と Veritas File System の統合」を参照してください。

p.125 の「I/O パフォーマンスの向上のためのアプリケーション固有のパラメータ」を参照してください。

拡張 I/O クラスタ化機能

I/O のクラスタ化とは、処理効率を向上させるために、複数の I/O 操作をグループ化するための手法の 1 つです。Veritas File System（VxFS）の I/O ポリシーは、他のファイルシステムより積極的にクラスタ化処理を実行することで、大容量ファイルに対して高い I/O スループットを実現しています。その結果、RAW ディスクと同レベルの処理効率が得られます。

拡張された I/O パフォーマンスのための Veritas Volume Manager と Veritas File System の統合

Veritas File System（VxFS）は Veritas Volume Manager（VxVM）と統合されています。そのため、下位ボリュームの I/O 特性を決定し、それに従って I/O を実行できます。また、VxFS では、この情報を使って、`mkfs` によって適切なアロケーションユニットを整列し、カーネルからの I/O 操作を効率よく実行します。

VxFS と VxVM の統合により、VxVM から I/O パラメータセットのエクスポートが可能になり、I/O の処理効率が向上します。このインターフェースにより、RAID 5 ボリューム、ストライプボリューム、ミラーボリュームなど、ボリューム設定に応じた高い処理効率を実現できます。RAID 5 ボリュームで I/O の処理効率を著しく高めるには、フルストライプ単位の

書き込みが重要です。VxFS では、これらのパラメータを使って適切な I/O 要求を VxVM に伝達します。

I/O パフォーマンスの向上のためのアプリケーション固有のパラメータ

ファイルシステム単位で、次のようなアプリケーション固有のパラメータを設定することで、I/O の処理効率を向上できます

- ディスカバードダイレクト I/O
  この値を超えるサイズはどれもダイレクト I/O として実行されます。
- ダイレクト I/O の最大サイズ
  この値は、1 つのダイレクト I/O の最大サイズを指定します。

`vxtunefs`(1M)と `tunefstab`(4)の各マニュアルページを参照してください。

拡張書き込み用の遅延割り当て

遅延割り当てでは、拡張書き込みの割り当てはスキップされ、バックグラウンドスレッドの割り当てを完了します。このアプローチによって、VxFS (Veritas File System) は、ファイルシステムの断片化を少なくする小さい割り当てを多数実行するのではなく、少数の割り当てを大きく割り当てます。動きが速い一時ファイルには、割り当てられているブロックがないため、ファイルシステムの断片化には追加されません。

ファイルが追記されると、ファイルへの割り当てはスキップされ、そのファイルは遅延割り当てリストに追加されます。割り当てがスキップされる範囲は、i ノードに記録されます。`write`() システムコールは、ユーザーのページがページキャッシュにコピーされるとすぐに返されます。ファイルへの実際の割り当ては、スケジューラスレッドが割り当てるファイルを選択すると実行されます。ファイルが切り捨てまたは削除されると、割り当ては不要になります。

遅延割り当ては、ローカルマウントファイルシステムではデフォルトで有効になっていますが、クラスタマウントファイルシステムでは無効になっています。この機能は、`vxtunefs` コマンドを使用して、クラスタマウントファイルシステムで有効にできます。`fsmap` コマンドを実行して、ファイル内の遅延割り当て範囲を表示できます。

詳しくは、`vxtunefs`(1M) と `fsmap`(1M) のマニュアルページを参照してください。

ファイルデータをディスクにすぐに書き込む必要があるインスタンスでは、遅延割り当てはファイルで無効になっています。ダイレクト I/O、同時 I/O、FDD/ODM アクセス、同期 I/O などのインスタンスの例を次に示します。遅延割り当ては、メモリマップファイルと BSD クォータではサポートされません。BSD クォータがファイルシステムで有効になると、遅延した割り当てはそのファイルシステムでは自動的に無効になります。

Veritas File System の使用

次のリストには、主な使用方法の VxFS の管理、修正、チューニングが含まれます。

- 「オンラインシステム管理」
- 「アプリケーションプログラミングインターフェース」

オンラインシステム管理

Veritas File System（VxFS）には、本マニュアル全般とマニュアルページ内で説明している CLI が用意されています。

VxFS を使うとファイルシステムがオンラインの間は多くの管理タスクを実行できます。管理タスクの中でさらに重要と思われる 2 つのタスクを次に示します。

- 「断片化の解消について」
- 「ファイルシステムのサイズ変更について」

断片化の解消について

フリーリソースには、最適な処理効率を達成できるような順序でアラインメントと割り当てが実行されます。ファイルシステムは、使用中にファイルの作成、削除、サイズ変更が実行されるに従って、フリーリソースの元の順序が失われていきます。ファイルシステムはディスク上に分散していき、使われない断片が使用中の領域に挟まれたまま残されてしまいます。このようなプロセスは「断片化」と呼ばれます。この断片化によって、ファイルシステム上でファイルへの空きエクステント（連続するデータブロックグループ）の割り当て方法が限定されてしまうため、処理効率は低下していきます。

VxFS では、断片化の問題を解消するためのオンラインの管理ユーティリティとして `fsadm` を提供しています。

`fsadm` ユーティリティは、次の方法でマウントされたファイルシステムの断片化を解消します。

- ディレクトリから未使用の領域を削除する。
- 小さなファイルをすべて連続するように配置する。
- ファイルシステムで使えるように、空きブロックを整理してまとめる。

このユーティリティは要求に応じて即時実行することができるため、cron を使って定期的に実行するようにします。

`fsadm_vxfs`（1M）のマニュアルページを参照してください。

ファイルシステムのサイズ変更について

ファイルシステムには、その作成時に特定のサイズが割り当てられます。ただし、ファイルシステムを使っていくうちにファイルシステムのサイズが適切ではなくなってくることがあります。

VxFS は、使用中にファイルシステムのサイズを変更できます。一般のコンピューティングファイルシステムでは、このようなサイズ変更は行えません。VxFS のユーティリティである fsadm を使うと、ファイルシステムのマウントを解除してユーザーの作業を中断することなく、ファイルシステムのサイズを拡張または縮小できます。ただし、ファイルシステムを拡張するには、ファイルシステムがマウントされているデバイスが拡張可能である必要があります。

VxVM では、使用中でもサイズを大きくすることが可能な仮想ディスクを使って拡張を行います。VxFS と VxVM の両方のコンポーネントの機能を使うことで、オンラインでの拡張が可能になります。ボリュームとファイルシステムのサイズを変更するには、vxresize コマンドを使ってください。vxresize コマンドを使うと、ファイルシステムはボリュームと一緒に縮小、拡張します。この操作のために vxassist コマンドと fsadm コマンドを組み合わせて使うこともできますが、代わりに vxresize コマンドを使うことをお勧めします。

vxresize(1M)のマニュアルページを参照してください。

p.214 の「新しい LUN の追加による既存のストレージの拡張」を参照してください。

アプリケーションプログラミングインターフェース

Veritas File System Developer's Kit(SDK)には、アプリケーションプログラミングインターフェース(API)を使用する場合や Veritas File System(VxFS)に用意されている数多くの機能やコンポーネントを修正したり、変更したりする場合に必要とされる情報が用意されています。

『Veritas File System プログラマリファレンスガイド』を参照してください。

VxFS は SVID(System V Interface Definition)必要条件に準拠し、NFS(Network File System)を使ったユーザーアクセスをサポートしています。他のファイルシステムでは利用できないパフォーマンスを向上させる機能を必要とするアプリケーションには、VxFS の拡張機能が有効です。

アプリケーションの拡張機能

Veritas File System(VxFS)では、商用アプリケーションと密接に関連付けられた API 機能を提供します。次は、このような商用アプリケーションで実行できる処理です。

- ファイル用領域の事前割り当て
- ファイル用の固定エクステントサイズの指定
- システムバッファキャッシュをバイパスしたファイル I/O

- ファイルに対して想定されるアクセス様式の指定

これらの拡張機能は VxFS 固有の IOCTL システムコールで実現しているため、既存の UNIX システムの多くではこれらの機能は使われません。アプリケーションでこれらの機能を使う場合は、可搬性の問題から、アプリケーションで使っているファイルシステムタイプを確認する必要があります。

5

Storage Foundation Cluster File System High Availability の動作

この章では以下の項目について説明しています。

Storage Foundation Cluster File System High Availability の動作方法

Storage Foundation Cluster File System High Availability（SFCFSHA）は、以下によるシステム管理タスクを単純化またはなくします。

- SFCFSHA の単一ファイルシステムのイメージ管理モデルでは、すべてのファイルシステム管理コマンドを任意のノードから実行できるようにし、管理が単純化されています。
- クラスタ内のすべてのサーバーが SFCFSHA のクラスタ共有ファイルシステムにアクセスできるため、複数のサーバー間でデータの一貫性が自動的に保たれます。すべてのクラスタノードは同じデータにアクセスでき、すべてのデータは、単一のサーバーファイルシステムのセマンティクスを使っているすべてのサーバーからアクセス可能です。
- すべてのファイルにすべてのサーバーからアクセスできるため、アプリケーションをサーバーに割り当てることで負荷を分散したり、その他の操作上の必要条件を満たすことができます。同様に、データへのアクセスのしやすさによって制約されないことから、フェールオーバーをより柔軟に行うことができます。
- 個々の SFCFSHA ファイルシステムをクラスタ内の任意のノードにすることができるため、ファイルシステムをクラスタノード間で均等に分散させることにより、*n* 個のノードクラスタでのフェールオーバー時間のうち、ファイルシステムのリカバリに占める時間を *n* 分の 1 に短縮できます。
- エンタープライズ RAID サブシステムのすべての容量がすべてのサーバーによってマウントでき、ハードウェア再構成の代わりに管理操作を行って容量を割り当てることができるため、より効率的に利用できます。

- より広範なストライプ化によるボリュームの拡張により、アプリケーション I/O の負荷分散が改善されます。各サーバーの I/O 負荷がストレージリソース間で分散されるだけでなく、SFCFSHA の共有ファイルシステムを利用することで、すべてのサーバーの負荷も互いに分散されます。
- 新しいサーバーはボリュームとファイルシステムに関するクラスタ全体の設定をそのまま取り入れるため、新しいサーバーのそれぞれに対してストレージ設定を行う必要はなく、サーバーの追加によるクラスタの拡張が容易になっています。
- ファイルベースのデータベースのパフォーマンスを RAW パーティションベースのデータベースと同等にするクラスタ化 Oracle Disk Manager（ODM）機能を、クラスタ内で動作中のアプリケーションで利用できます。

Storage Foundation Cluster File System High Availability を使う状況

SFCFSHA は、ホームディレクトリ、ブートサーバーファイル、Web ページなどを対象としたファイル共有が必要なアプリケーションや、クラスタ対応アプリケーションでの利用に適しています。また、大部分が読み取り専用の、データへのアクセスのみが必要な環境でいつでも使えるスタンバイデータが必要なときや、ファイル共有のために NFS を利用したくないときにも SFCFSHA を利用できます。

SFCFSHA はほとんどすべてのアプリケーションにとってメリットがあります。たとえば、「クラスタ対応」でないアプリケーションでも、クラスタ内の任意の場所にあるデータにアクセスし、操作できるようになります。異なるサーバー上で動作する複数のクラスタアプリケーションが、クラスタファイルシステム内のデータにアクセスしている場合、別々の下位ボリューム上に 1 つのクラスタファイルシステムを置く負荷分散効果によって、システム全体の I/O パフォーマンスが向上します。この処理は自動的に行われ、調整やその他の管理操作は不要です。

多くのアプリケーションは複数の同時実行スレッドで構成され、各スレッドのデータアクセスを調整する手段がある場合、それらのスレッドを複数の異なるサーバー上で実行できます。SFCFSHA はこの調整を行います。そのようなアプリケーションをクラスタ対応にし、アプリケーションのインスタンスどうしが協調してクライアント負荷とデータアクセス負荷を分散できるようになり、その結果として単一サーバーの容量を超えた拡張が可能になります。そのようなアプリケーションでは SFCFSHA が共有データアクセスを可能にし、クラスタノード間でのアプリケーションレベルの負荷分散ができるようにします。

SFCFSHA には以下の機能が用意されています。

- 常に利用可能でなければならない単一ホストアプリケーションの場合、SFCFSHA は、サーバーエラー発生後にアプリケーションが再起動できる稼働中のファイルシステム環境を提供することで、アプリケーションのフェールオーバー時間を短縮できます。

- 分散データベース管理システムや Web サーバーなどの並列アプリケーションの場合、SFCFSHA は、共有データをすべてのアプリケーションインスタンスに同時に提供します。また SFCFSHA は、サーバーの追加によってこれらのアプリケーションを拡張できるようにするとともに、サーバーエラー発生時にネットワークアドレスを再割り当てするだけで負荷を再分散できるようにすることでアプリケーションの可用性を向上させます。
- ビデオ制作など、非常に大きなファイルをステーション間で受け渡しするワークフローアプリケーションの場合、SFCFSHA によりすべてのステーションからファイルにアクセスできるようになるため、時間がかかってエラーになりやすいデータコピーは不要になります。
- バックアップに関しては、SFCFSHA では別々のサーバーで実行し、クラスタ共有ファイルシステム内のデータにアクセスすることによる操作上の影響を軽減できます。

以下に示すのは、SFCFSHA と連携した場合にアプリケーションがどのように機能するかの例です。

- ファイルサーバーで Storage Foundation Cluster File System High Availability を使用

  クラスタ設定で接続された（同じクライアントと同じストレージに接続された）2 台以上のサーバーが、別々のファイルシステムとして機能します。サーバーのうち 1 台にエラーが発生すると、他のサーバーがエラーを認識し、回復させ、プライマリシップを引き受け、エラーの発生したサーバーの IP アドレスを使ってクライアントへの応答を開始します。
- Web サーバーで Storage Foundation Cluster File System High Availability を使用

  Web サーバーのアプリケーションは通常は読み取り専用であるため、Web サーバーは共有クラスタ化に特に適しています。さらに、クライアント負荷分散フロントエンドを利用すると、サーバーとサイトのコピーを追加することで Web サーバークラスタの容量を拡張できます。SFCFSHA ベースのクラスタにより、このタイプのアプリケーションの拡張と管理は大幅に単純化されます。

Storage Foundation Cluster File System High Availability のアーキテクチャについて

Storage Foundation Cluster File System High Availability（SFCFSHA）を利用すると、ファイルシステムを使っているすべてのアプリケーションが同じサーバー上で動作しているかのように、複数のクラスタ化サーバーが 1 つのファイルシステムを同時にマウントして使うことができます。Veritas Volume Manager クラスタ機能（CVM）を使うと、論理ボリュームと RAW デバイスアプリケーションにクラスタ内のどこからでもアクセスできるようになります。

ここでは、次の内容について説明します

- 対称アーキテクチャについて
- SFCFSHA のプライマリ/セカンダリフェールオーバーについて
- Group Lock Manager を使用した単一ホストファイルシステムのセマンティクスについて

対称アーキテクチャについて

SFCFSHA では、クラスタ内のすべてのノードをメタデータサーバーとして同時に稼動することができる対称アーキテクチャを導入しています。SFCFSHA には、従来のマスター/スレーブまたはプライマリ/セカンダリという概念が一部に残っています。各クラスタファイルシステムを最初にマウントしたサーバーがそのプライマリになり、クラスタ内の他のノードはすべてセカンダリになります。アプリケーションは自らが動作しているサーバーから直接、ファイル内のユーザーデータにアクセスします。各 SFCFSHA ノードは、独自のインテントログを所有します。ファイルシステム操作(ファイルの割り当てや削除など)は、クラスタ内の任意のノードから行うことができます。

p.36 の「Veritas File System のインテントログについて」を参照してください。

Storage Foundation Cluster File System High Availability のプライマリ/セカンダリフェールオーバーについて

SFCFSHA プライマリが稼動するサーバーに障害が発生した場合は、残りのクラスタノードから新しいプライマリが選択されます。新しいプライマリは古いプライマリのインテントログを読み取り、障害時に処理中であったメタデータ更新を完了します。

SFCFSHA セカンダリ上で実行していたサーバーに障害が発生すると、プライマリは障害が発生したセカンダリのインテントログを読み取り、障害時に処理中であったメタデータ更新を完了します。

p.416 の「CFS プライマリノードに障害が発生した場合」を参照してください。

Group Lock Manager を使用した単一ホストファイルシステムのセマンティクスについて

SFCFSHA では、Veritas GLM(Group Lock Manager)を使って、各クラスタに UNIX の単一ホストファイルシステムのセマンティクスを複製します。これは書き込み動作で最も重要です。UNIX ファイルシステムでは、書き込み動作が原子的であるように見えます。つまり、あるアプリケーションがファイルに一連のデータを書き込むと、データがファイルシステムによってキャッシュされているがディスクに書き込まれていない場合でも、それ以降にそのファイルの同じ領域を読み取るアプリケーションはすべて、書き込まれた新しいデータを取り込みます。アプリケーションが整合性のないデータや、前回の書き込み結果の一部のみを取り込むことは一切できません。

単一ホスト書き込みセマンティクスを複製するために、書き込み元のクラスタノードに関係なく、システムのキャッシュは整合性を保持し、ノードごとにキャッシュされたデータへのすべての更新を瞬時に反映する必要があります。GLM はファイルをロックし、クラスタ内の他のノードがそのファイルを同時に更新できないようにしたり、更新が完了するまでそのファイルを読み取ることができないようにします。

クラスタファイルシステムでサポートされている Veritas File System 機能について

Storage Foundation Cluster File System High Availability は VxFS（Veritas File System）に基づきます。

VxFS ローカルファイルシステムのほとんどの主要機能をクラスタファイルシステムで利用できます。次の機能が含まれます。

- 最大 1 TB のサイズのファイルをマップするエクステントベースの領域管理
- インテントログを使ってファイルシステムメタデータの最近の更新を追跡することによるシステムクラッシュからの高速リカバリ
- ファイルシステムが使用中でもその拡張や断片化解消が可能なオンライン管理

VxFS のすべてのマニュアルページには、「Storage Foundation Cluster File System High Availability の問題」についてのセクションがあります。このセクションでは、クラスタマウントされたファイルシステムでコマンドが機能するかどうかについての情報が記載されています。また、ローカルマウントされたファイルシステムの場合との動作の違いを示しています。

p.111 の「Veritas File System の機能」を参照してください。

クラスタファイルシステムでサポートされていない Veritas File System 機能

表 5-1 はクラスタファイルシステムでサポートされない機能を一覧表示します。一覧表示された機能は使用できますが、正しく動作する保証はありません。

サポートされていない機能を SFCFSHA で使わないようにしてください。また、マウントしているファイルシステムを、ローカルマウントやクラスタマウントとしてこれらのオプションで置き換えないようにしてください。

表 5-1 クラスタファイルシステムでサポートされていない Veritas File System の機能

サポートされない機能	コメント
qlog	Quick Log はサポートしていません。
スワップファイル	スワップファイルはクラスタマウントされたファイルシステムでサポートされません。
mknod	クラスタマウントされたファイルシステムで、mknodコマンドを使ってデバイスを作成することはできません。
キャッシュアドバイザリ	キャッシュアドバイザリは個別のファイルシステム上で mount コマンドによって設定されますが、クラスタの他のノードには伝達されません。
ファイルアクセス時間に依存するコマンド	クラスタファイルシステムでは atime ファイル属性が厳密に同期されないため、ファイルアクセス時間の表示がノード間で異なる場合があります。そのため、アクセス時間のチェックに依存するユーティリティは確実に機能しない場合があります。

Cluster Server のアーキテクチャについて

GAB（Group Membership and Atomic Broadcast）と LLT（Low Latency Transport）は、イーサネットデータリンクに直接実装する VCS（Cluster Server）固有のプロトコルです。この 2 つは、クラスタ内のノードどうしを接続する冗長データリンク上で実行されます。単一点障害を回避するために、VCS には冗長クラスタ通信リンクが必要です。

GAB は、クラスタとクラスタアプリケーションにメンバーシップとメッセージサービスを提供します。また、GAB のメンバーシップにより、クラスタの起動と停止が正しい順序で実行されます。GAB を設定するには、/etc/gabtab ファイルを使います。このファイルには、起動時に GAB によって実行される gabconfig コマンドが含まれています。たとえば、-n <*number*> コマンドオプションで、クラスタのノード数を指定します。GAB は、SFCFSHA インストールスクリプトの実行時に自動的に設定されますが、クラスタにノードを追加する場合は、GAB を再設定する必要があります。

gabconfig（1M）のマニュアルページを参照してください。

LLT により、カーネル間の通信が可能になり、ネットワーク通信が監視されます。/etc/llthosts および /etc/llttab の各 LLT ファイルを設定して、クラスタ内のシステム ID の構成、複数のクラスタ ID の構成、ハートビートの頻度などのネットワークパラメータのチューニングを実行します。LLT により、クラスタメンバーシップの変化を即座に反映し、高速な応答が可能になります。

GAB と同様に、LLT は VCS のインストールスクリプトの実行時に自動的に設定されます。/etc/llttab、/etc/llthostsファイルはあなたがインストールの間に提供する情

報を含んでいます。クラスタにノードを追加する際に、LLT の再設定が必要になる場合があります。

llttab(4)と llthosts(4)の各マニュアルページを参照してください。

『Cluster Server 管理者ガイド』を参照してください。

SFCFSHA の各コンポーネントは、GAB メンバーシップポートに登録されます。ポートメンバーシップは、各コンポーネントに対して、クラスタを構成するノードを識別します。

表 5-2 にポートメンバーシップをまとめます。

表 5-2 ポートメンバーシップ

Port	説明
ポート a	GAB ハートビートメンバーシップ
ポート b	I/O フェンシングメンバーシップ
ポート d	Oracle Disk Manager (ODM)メンバーシップ
ポート f	クラスタファイルシステムメンバーシップ
ポート h	GAB と HAD(High Availability Daemon)との間の Cluster Server の通信
ポート m	SmartIO VxVM キャッシュの一貫性のための GLM(Group Lock Manager)通信。
ポート u	CVM スレーブから CVM マスターにコマンドをリダイレクトするための Cluster Volume Manager (CVM)ポート
ポート v	Cluster Volume Manager メンバーシップ
ポート y	I/O 転送のための CVM(Cluster Volume Manager)ポート
ポート w	各ノードの Cluster Volume Manager デーモンはこのポートを使って相互に通信するが、クラスタメンバーシップ情報は GAB(ポート v)経由で受け取る

Storage Foundation Cluster File System High Availability 名前空間について

マウントポイントの名前は、同じクラスタファイルシステムをマウントするすべてのノードで同じにする必要があります。この設定は、CFS マウントエージェント(オンライン、オフラインおよび監視)が正しく動作するために必要です。

非対称マウントについて

mount -o cluster オプション付きでマウントされた Veritas File System（VxFS）ファイルシステムは、ローカルマウント（非共有マウント）ではなく、クラスタマウント（共有マウント）です。共有モードでマウントされたファイルシステムはクラスタ環境で VxVM 共有ボリュームに配置される必要があります。mount -o remount オプションを使用するときは、ローカルマウントを共有モードで再マウントしたり、共有マウントをローカルモードで再マウントすることはできません。単一のクラスタ化されたファイルシステムは、異なるノードで異なる読み取りおよび書き込みオプションでマウントできます。このようなファイルシステムは非対称マウントと呼ばれます。

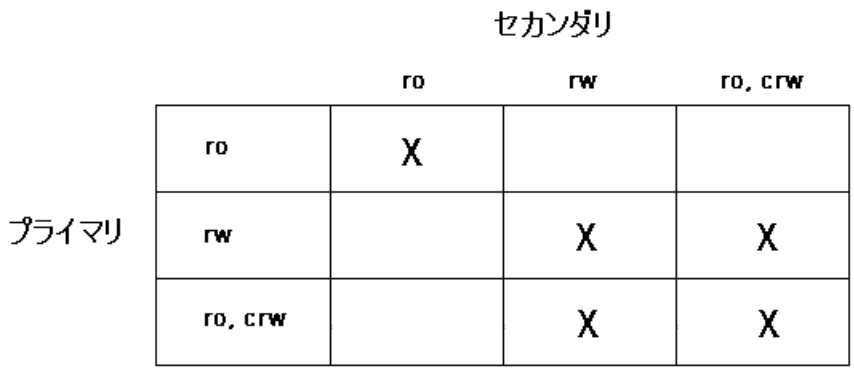
非対称マウントは、共有ファイルシステムを異なる読み取りおよび書き込み機能でマウントされます。たとえば、クラスタ内の 1 つのノードは読み取りおよび書き込みモードでマウントし、一方で、他のノードは読み取り専用モードでマウントすることができます。

プライマリが「ro」でマウントするとき、これはこのノードもその他のノードもファイルシステムの書き込みを許可されていないことを意味します。プライマリが「ro」をマウントする場合、セカンダリは「ro」のみをマウントできます。それ以外の場合、プライマリは「rw」または「ro,crw」をマウントし、セカンダリは同じ選択肢を持ちます。

クラスタ読み取りおよび書き込み（crw）オプションは、最初にファイルシステムをマウントするときに非対称マウントを行うことを指定する場合に使います。また、再マウント（mount -o remount）を実行して、オプションを変更する際に、指定することもできます。

図 5-1 に、プライマリノードとセカンダリノードのマウントが可能な、異なるモードを示します。

図 5-1　　プライマリマウントとセカンダリマウント



チェックマークは、所定のプライマリモードに対してセカンダリマウントが使用するモードを示します。

-o cluster,roオプションのみのプライマリをマウントすることは異なるモードの土台からセカンダリを防ぎます; つまり、読み書き両用。

詳しくは、mount_vxfs（1M）のマニュアルページを参照してください。

プライマリとセカンダリのクラスタノードについて

ファイルシステムクラスタは 1 つのプライマリと 63 個までのセカンダリで構成されます。プライマリとセカンダリの用語は、特定のノード（またはハードウェアプラットフォーム）ではなく 1 つのファイルシステムに適用されます。同一のクラスタノードを、ある共有ファイルシステムではプライマリにし、同時に別の共有ファイルシステムではセカンダリにすることができます。そのようにファイルシステムのプライマリシップを分散させて、クラスタ上の負荷を分散させる管理方針が推奨されます。

p.417 の「クラスタ内の作業負荷の分散について」を参照してください。

CVM の場合、単一のクラスタノードがクラスタ内のすべての共有ディスクグループと共有ボリュームのマスターになります。

プライマリシップの確認または移動

クラスタファイルシステムを最初にマウントしたノードはプライマリノードと呼ばれます。その他のノードはセカンダリノードと呼ばれます。プライマリノードで障害が発生すると、内部選択処理によって、どのセカンダリをプライマリファイルシステムにするかが決定されます。

プライマリシップを確認するには

- プライマリシップを確認するには、次のコマンドを入力します。

```
# fsclustadm -v showprimary mount_point
```

ノードをプライマリノードに設定するには

- ノードをプライマリノードに設定するには、ノードで次のコマンドを入力します。

```
# fsclustadm -v setprimary mount_point
```

クラスタファイルシステムの時間の同期について

SFCFSHA は、NTP（Network Time Protocol）デーモンなど何らかの外部コンポーネントを使って、すべてのノードのシステムクロックが同期されている必要があります。ノードを同期化しないと、i ノードのタイムスタンプ（ctime）とデータ修正時のタイムスタンプ（mtime）が実際の操作順序と一致しない場合があります。

ファイルシステムのチューニングパラメータ

/etc/vx/tunefstab ファイルを使用すると、調整可能なパラメータがファイルシステムのマウント時に更新されます。ファイルシステム /etc/vx/tunefstab のパラメータは、すべてのノードで同じになるように各クラスタノードに伝達されます。ファイルシステムがノードにマウントされる場合は、プライマリノードの /etc/vx/tunefstab パラメータが使われます。このファイルは各ノードで同じにしておくことをお勧めします。

メモ: /etc/vx/tunefstab ファイルがない場合は、手動で作成します。

デバイスのマウント中に lazy_copyonwrite を 1 に調整するには、ファイルを設定します。次に例を示します。

```
# cat /etc/vx/tunefstab

/dev/vx/dsk/sharedg/vol1 lazy_copyonwrite=1
```

詳しくは、tunefstab(4) および vxtunefs(1M) のマニュアルページを参照してください。

並列 fsck スレッドの数の設定について

この項では、並列 fsck スレッドの数を設定する方法について説明します。

リカバリ時にアクティブにできる並列 fsck スレッドの数は 4 に設定されていました。たとえば、1 つのノードが 12 のファイルシステムでフェールオーバーした場合は、その 12 のファイルシステムのログ再生が同時に完了しなくなります。数が 4 に設定されていたのは、メモリの少ないシステムでは、多数のファイルシステムの並列再生によってメモリに負荷がかかりすぎるからです。一方、大きなシステムでは、並列再生プロセスの数を 4 に限定する必要はありません。

この値はシステムの利用可能な物理メモリに従って調整されます。

並列 fsck スレッドの数を設定するには

- クラスタ内のすべてのノードで、/opt/VRTSvcs/bin/CFSfsckd/CFSfsckd.env ファイルを編集し、set FSCKD_OPTS="-n *N*" と設定します。

  ここで、*N* は必要な並列 fsck スレッドの数で、*N* の値は 4 と 128 の間でなければなりません。

Storage Checkpoint

SFCFSHA（Storage Foundation Cluster File System High Availability）には、Storage Checkpoint 機能が用意されており、特定時刻のファイルシステムの永続的なイメージを瞬時に作成できます。

p.737 の「Storage Checkpoint について」を参照してください。

ProductName; のバックアップ方法について

名前空間にアクセスするための API やコマンドが同じであるため、VxFS（Veritas File System）Standard Editionで使われているバックアップ方法と同じ方法を Storage Foundation Cluster File System High Availability（SFCFSHA）でも使えます。ファイルシステムの CheckPoint は、ファイルシステムのオンディスク PITC（point-in-time copy）です。クラスタファイルシステムの静止イメージを取得する際は、I/O パターンによって、チェックポイントを作成したファイルシステムの方がパフォーマンス上有効なため、クラスタファイルシステムの静止イメージを取得する際は、ファイルシステムのスナップショットより CheckPoint を使うことをお勧めします（下記を参照）。

ファイルシステムのスナップショットも、ファイルシステムのオンディスク静止イメージを取得する方法です。チェックポイント機能とは対照的に、この静止イメージは非永続的です。スナップショットに読み取り専用モードでマウントされたファイルシステムとしてアクセスすると、ファイルシステムのオンラインバックアップを効率よく実行できます。スナップショットは、スナップファイルシステムでデータブロックが上書きされるとデータブロックを逐次コピーするコピーオンライトセマンティクスを実装しています。クラスタファイルシステムのスナップショットは、クラスタノードから発生する I/O のコピーオンライト機構を拡張します。

バックアップ用にスナップショットファイルシステムをマウントすると、コピーオンライトを実行し、スナップショットからデータブロックを読み取るためにリソースが使われるため、システム上の負荷が増大します。このような場合、クラスタのスナップショットを使ってオフホストバックアップを行うことができます。オフホストバックアップは、プライマリサーバーでのバックアップアプリケーションによる負荷を軽減します。スナップショット全体のオーバーヘッドに比べて、リモートスナップショットからのオーバーヘッドは小さくて済みます。したがって、比較的負荷の少ないノードからスナップショットをマウントして、バックアップアプリケーションを実行すると、クラスタ全体のパフォーマンスに効果があります。

クラスタスナップショットには、次の特性があります。

- マウントしたクラスタファイルシステムのスナップショットは、クラスタ内の任意のノードにマウントできます。 マウントしたファイルシステムは、プライマリ、セカンダリ、またはセカンダリ専用にできます。ファイルシステムの安定したイメージを任意のノードからの書き込み用に提供できます。
  CFS マウントオプションであるセカンダリ専用（seconly）ファイルシステムについて詳しくは、`mount_vxfs` のマニュアルページを参照してください。

- クラスタファイルシステムの複数のスナップショットを同じクラスタノードまたは異なるクラスタノードにマウントできます。
- スナップショットは、スナップショットをマウントしているノードでのみアクセスできます。スナップショットデバイスを 2 つのノードに同時にマウントすることはできません。
- スナップショットをマウントするためのデバイスとしてローカルディスクまたは共有ボリュームを使えます。共有ボリュームは、スナップショットマウントによって排他的に使われ、スナップショットがそのデバイスでマウントされている間は、他のノードから使えません。
- スナップショットをマウントしているノードでは、スナップショットがマウントされている間はスナップショットを取られたオリジナルのファイルシステムのマウントを解除できません。
- SFCFSHA スナップショットは、スナップショットのマウントが解除された場合、またはスナップショットをマウントしているノードに障害が発生した場合、消失します。ただし、他のノードがクラスタから離れたり、クラスタに参加しても、スナップショットはその影響を受けません。
- 読み取り専用でマウントされたファイルシステムのスナップショットは作成できません。クラスタファイルシステムのスナップショットは、クラスタファイルシステムが `crw` オプションでマウントされている場合のみ、スナップショットを取得できます。

ファイルシステムの静止イメージの他に、Mirror Split および再結合を使って、共有ボリュームでボリュームレベルの静止イメージを使うこともできます。FastResync、Space Optimized スナップショットなどの機能も使えます。

並列 I/O について

一部の分散アプリケーションは、クラスタ内の 1 つ以上のノードから同じファイルに同時に読み取りや書き込みを行うことがあります。たとえば、1 つのスレッドがファイルに追加を行い、同時にいくつかのスレッドがそのファイルのさまざまな領域から読み取りを行うような分散アプリケーションがあります。一部の HPC（高性能コンピュータ）アプリケーションにも、利点として、このように 1 つのファイルに同時に I/O を行う機能があります。並列 I/O を使うためにアプリケーションを変更する必要はありません。

従来は、小さな領域への I/O を実行するためにファイル全体をロックしていました。並列 I/O をサポートするために、SFCFSHA は、I/O の要求に対応するファイル内の領域をロックします。ロックされる領域の単位はページです。2 つの I/O 要求があり、その少なくとも一方が書き込み要求で、その I/O 範囲がもう一方の I/O 範囲と重なる場合、それらの I/O 要求は競合します。

並列 I/O の機能により、要求が競合しないかぎり、複数のスレッドから 1 つのファイルへの同時 I/O が可能になります。I/O 要求を同時に発行する複数のスレッドは、同じノードで実行されていても、クラスタ内の異なるノードで実行されていてもかまいません。

メモリ割り当てが必要な I/O 要求を他の I/O 要求と同時に実行することはできません。書き込み側のファイルが拡張しており、読み取り側がこれより遅れている場合は、拡張部分に書き込むごとに必ずしもブロックの割り当てが行われないことに注意してください。

ファイルのサイズを事前に決定し、事前にファイルを割り当てて、I/O 時のブロックの割り当てを回避します。これにより、ファイルに並列 I/O を実行するアプリケーションの同時処理能力が向上します。また、並列 I/O により、クラスタ全体のキャッシュの整合性を損なわずに、ページキャッシュの無駄なフラッシュや無効な領域ロックの使用も回避します。

複数のノードから 1 つのファイルを更新するアプリケーションに対しては、マウントのオプション `-nomtime` によって同時処理能力がさらに向上します。クラスタ全体でファイルの変更と変更時刻が同期されないため、I/O とロックが増えることによるオーバーヘッドはありません。ノードのこれらのファイルに表示されるタイムスタンプは、過去 60 秒間に更新されていない可能性があります。

Cluster Volume Manager の I/O エラー処理ポリシーについて

I/O エラーは、ファイバーチャネルリンク、ホストバスアダプタ、ディスクの障害などの様々な原因で発生します。SFCFSHA は、I/O エラーが発生したノードでファイルシステムを無効にします。ファイルシステムは、他のノードからは引き続き使えます。

ハードウェアの障害が修正（ファイバーチャネルリンクの再確立など）されたら、クラスタ内のすべてのアクティブノードからファイルシステムのマウントを強制解除し、マウントリソースを無効のノードからオンラインにしてファイルシステムを再起動できます。

I/O 障害からのリカバリについて

無効になったファイルシステムは、マウントの強制解除によってリストアできます。また、再ブートしなくてもリソースはオンラインになり、これによって共有ディスクグループのリソースもオンラインになります。

メモ: jeopardy 条件が修正されていない場合、それらのクラスタは、その後のノードの障害でクラスタから切断されやすくなります。

『Cluster Server 管理者ガイド』を参照してください。

単一ネットワークリンクと信頼性について

環境によっては、ネットワーク障害に備えるための冗長性がなくなっても、クラスタ内のノードの接続に単一のプライベートリンクまたはパブリックネットワークを使う方がよい場合があ

ります。ハードウェアのトポロジーが単純で、費用も抑えられる利点はありますが、格段に可用性が低下します。

このような環境においては、SFCFSHA では、単一のプライベートリンクを使うか、もしくはパブリックネットワークをプライベートリンクとして使う選択ができます(I/O フェンシングが存在する場合)。スプリットブレインは、I/O フェンシングを使って処理されます。単一ネットワークは、インストール時に選択できます。

p.146 の「I/O フェンシングによるデータ破損の防止について」を参照してください。

優先度が低いリンクの設定

優先度の低いリンク(LLT)は、通常のハートビートチャネルのバックアップとして、優先度の低いリンクを使うように設定できます。通常、低優先度リンクはパブリックネットワークまたは管理ネットワーク上に設定します。通常、これは、クラスタのプライベートな相互接続とはまったく異なるネットワークインフラストラクチャになり、1 つの問題によってすべてのリンクが停止する可能性が減少します。優先度の低いリンクは、これだけが最後に残らないかぎり、クラスタメンバーシップのトラフィックには使われません。通常の稼動状態で優先度の低いリンクは、クラスタメンバーシップおよびリンク状態の保守のハートビートトラフィックのみを伝送します。ハートビートの頻度は、ネットワークのオーバーヘッドを減少させるために 50 % になります。優先度の低いリンクだけがネットワークリンクとして残った場合、LLT は、すべてのクラスタの状態に関するデータのやり取りも切り替えます。設定済みのプライベートリンクのいずれかが修復されると、LLT は、クラスタの状態に関するデータのやり取りを「優先度の高いリンク」に戻します。

クライアントが接続している間も、LLT リンクを追加または削除できます。GAB や高可用性デーモン had を終了する必要はありません。

リンクを追加するには

- リンクを追加するには、以下のコマンドを入力します。

  ```
  # lltconfig -d device -t device_tag
  ```

  ここで、*device_tag* は、後続コマンドで特定のリンクを識別するためのタグで、`lltstat`(1M) によって表示されます。

リンクを削除するには

- リンクを削除するには、以下のコマンドを入力します。

  ```
  # lltconfig -u device_tag
  ```

`lltconfig`(1M)のマニュアルページを参照してください。

変更はすぐに適用されますが、再ブートすると元に戻ります。再ブート後も変更を有効にするには、`/etc/lltab` ファイルを更新する必要があります。

メモ: LLT クライアントは、1 つの LLT のリンクのみが残され、GAB が jeopardy を宣言するまで、クラスタの状態を認識しません。

スプリットブレインと Jeopardy 処理

クラスタノード間でクラスタのメンバーシップビューが異なり、データの破損の可能性が増加した場合は、スプリットブレイン状態になります。I/O フェンシングにより、データ破損の可能性がなくなります。I/O フェンシングには、SCSI-3 PR をサポートするディスクが必要です。

コーディネーションポイントサーバー(CP サーバー)を使って I/O フェンシングを設定することもできます。SCSI-3 をサポートしない仮想環境では、非 SCSI-3 サーバーベースフェンシングを設定できます。

p.159 の「サーバーベースの I/O フェンシングについて」を参照してください。

p.146 の「SCSI-3 PR をサポートしない仮想マシンでの SFCFSHA 用 I/O フェンシングについて」を参照してください。

Jeopardy 状態

I/O フェンシングを使わない場合、SFCFSHA のインストールには 2 つのハートビートリンクが必要です。単一のハートビート接続に対してノードが停止した場合、SFCFSHA は、システムの障害か最終的なネットワーク接続の障害かを区別できなくなります。この状態は「jeopardy」と呼ばれます。

SFCFSHA は jeopardy を検出し、一部のスプリットブレイン状況でのデータ破損を防ぐ方法で対応します。ただし、データ破損は他の状況で起きる可能性があります

- すべてのリンクが同時に切断される。
- ノードが異常終了し、ハートビートメッセージに応答できなくなった。

このような状況でデータの破損の可能性をなくすには、I/O フェンシングが必要です。I/O フェンシングを使う場合、jeopardy 状態は、SFCFSHA スタックによる特別な処理を必要としません。

Jeopardy 処理

I/O フェンシングが設定されていないインストールでは、jeopardy 処理によって潜在的なスプリットブレイン状態が防止されます。jeopardy 状態の通知に続いていずれかのクラスタノードで障害が発生した場合、障害が発生したノードでマウントされたすべてのクラスタファイルシステムはすべての残りのノードで無効になります。jeopardy 状態の通知の後で切断の再設定が発生した場合、jeopardy 状態の通知を受信したノードはクラスタから切断されます。

I/O フェンシングについて

I/O フェンシングは、クラスタ内のノードがスプリットブレイン状態を示すクラスタメンバーシップの変更を検出するとき、共有ディスクのデータを保護します。

フェンシング操作で次のノードが決まります。

- 共有ストレージへのアクセスを保持しなければならないノード
- クラスタから取り出されなければならないノード

この決定によってデータ破損を防ぎます。インストーラは、Veritas InfoScale Enterprise をインストールするときに、I/O フェンシングドライバ、VRTSvxfen RPM の一部をインストールします。共有ディスクのデータを保護するには、Veritas InfoScale Enterprise をインストールして SFCFSHA を設定した後、I/O フェンシングを設定する必要があります。

I/O フェンシングモード - ディスクベースおよびサーバーベースの I/O フェンシング - ネットワーク分割の発生時にコーディネーションポイントを使用してアービトレーションを行います。一方、マジョリティベースの I/O フェンシングモードはアービトレーションにコーディネーションポイントを使用しません。マジョリティベースの I/O フェンシングでは、高可用性が失われる場合があります。ディスクベース、サーバーベースまたはマジョリティベースの I/O フェンシングを設定できます。

ディスクベースの I/O フェンシング	コーディネータディスクを使う I/O フェンシングはディスク型の I/O フェンシングと呼ばれます。 ディスク型の I/O フェンシングはシングルクラスタでデータ整合性を保証します。
サーバーベースの I/O フェンシング	少なくとも 1 つの CP サーバーシステムを使う I/O フェンシングはサーバー型の I/O フェンシングと呼ばれます。サーバーベースのフェンシングには、CP サーバーのみ、または CP サーバーとコーディネータディスクの組み合わせを含めることができます。 サーバー型の I/O フェンシングはクラスタでデータ整合性を保証します。 SCSI-3 PR をサポートしていない仮想化環境では、SFCFSHA は非 SCSI-3 の I/O フェンシングをサポートします。 p.146 の「SCSI-3 PR をサポートしない仮想マシンでの SFCFSHA 用 I/O フェンシングについて」を参照してください。

マジョリティベースの I/O フェンシング	マジョリティベースの I/O フェンシングでは、クラスタ環境でデータ破損に対する保護およびデータの整合性を提供するのにコーディネーションポイントを必要としません。 追加のサーバーやコーディネーションポイントとして使用できる共有 SCSI-3 ディスクがない場合はマジョリティベースの I/O フェンシングを使用します。

p.146 の「I/O フェンシングによるデータ破損の防止について」を参照してください。

メモ: I/O フェンシングを使ってスプリットブレインの状態からクラスタを保護することを推奨します。

『Storage Foundation Cluster File System High Availability 設定およびアップグレードガイド』を参照してください。

SCSI-3 PR をサポートしない仮想マシンでの SFCFSHA 用 I/O フェンシングについて

従来の I/O フェンシング実装では、コーディネーションポイントサーバー（CP サーバー）またはコーディネータディスクがコーディネーションポイントになっていて、CVM（Cluster Volume Manager）と Veritas I/O フェンシングモジュールによって SCSI-3 PR（SCSI-3 Persistent Reservation）ベースの保護がデータディスクに提供されます。この SCSI-3 PR 保護によって、接続を失ったノードからの I/O 操作が、テイクオーバー済みの残りのサブクラスタにアクセスできなくなります。

SCSI-3 PR をサポートしない仮想化環境では、SFCFSHA がデータディスクを適切に保護しようとします。このような環境では、SFCFSHA に非 SCSI-3 I/O フェンシングを設定する必要があります。非 SCSI-3 フェンシングでは、コーディネーションポイントとして CP サーバーのみを使用したサーバーベースの I/O フェンシング、またはコーディネーションポイントを使わないマジョリティベースの I/O フェンシングを使います。このような環境をサポートするために、一部の設定を変更します。

I/O フェンシングによるデータ破損の防止について

I/O フェンシングは、クラスタ内で通信障害が発生した場合にデータ破損を防止する機能です。

クラスタが高可用性を提供するには、ノードの障害時に修正アクションを実行する機能が必要です。このような場合、SFCFSHA は変更されたメンバーシップを反映するようにそのコンポーネントを設定します。

ノードの障害を検出する機構が壊れた場合も、ノードに障害が発生したときと同じ現象が起こるため、問題が発生します。たとえば、2 ノードクラスタ内のシステムにノード障害が発生した場合は、プライベートインターコネクト上でのハートビートの送信が停止します。次に、存在するノードが修正アクションを実行します。 実際のノードではなくプライベート相互接続に障害が起きた場合も同じ現象が起こり、これによって各ノードは自分の接続相手が切断されたと判断します。このような場合は、両方のノードが無秩序にデータストレージを制御しようとするため、通常はデータの破損につながります。

プライベートネットワークの障害の他にも、このような状況が生じる可能性があります。システムが非常にビジー状態で、応答がないか、または停止しているように見える場合、他のノードによりシステムがダウンしたと見なされることもあります。この宣言は、「中断」および「再開」機能をサポートするハードウェアを使用しているノードにも発生する場合があります。中断機能を使用してノードを PROM レベルに落としてから処理を再開すると、他のノードからシステム停止を宣言される場合があります。そのシステムが後で復旧して書き込み処理を再開しても、他のノードからはシステム停止を宣言される場合があります。

SFCFSHA は I/O フェンシングを使って、スプリットブレインに伴うリスクを除去します。I/O フェンシングは、アクティブなクラスタのメンバーへの書き込みアクセスを許可します。I/O フェンシングは、非メンバーからのストレージへのアクセスをブロックします。

SCSI-3 Persistent Reservation について

SCSI-3 PR（SCSI-3 Persistent Reservations）は、I/O フェンシングに必要で、クラスタ化された SAN 環境における SCSI 予約の使用に関する問題を解決します。SCSI-3 PR は、1 つのデバイスに複数のノードがアクセスできるようにするのと同時に、その他のノードのアクセスをブロックします。

SCSI-3 の予約は SCSI バスのリセット後も永続し、ホストからディスクへの複数のパスもサポートします。一方、SCSI-2 の予約では、1 つのパスと 1 つのホストによってのみ使えます。データの完全性のためにデバイスへのアクセスをブロックする必要がある場合は、1 つのホストと 1 つのパスのみがアクティブなまま残ります。複数のノードが制御された状態でストレージに読み書きする大規模クラスタでは、SCSI-II の予約では不十分です。

SCSI-3 PR は、登録と予約の概念を使います。各システムは、SCSI-3 デバイスにキーを登録します。キーを登録する複数のシステムはメンバーシップを形成し、予約（通常は WERO（Write Exclusive Registrants Only）と設定）を確立します。WERO 設定は、登録されたシステムだけが書き込み操作を実行できるようにします。特定のディスクに対して存在する予約は 1 つだけですが、登録は多数可能です。

SCSI-3 PR テクノロジを使うことにより、デバイスから登録を削除するだけで、簡単に書き込みアクセスをブロックできます。登録されたメンバーのみが別のメンバーの登録を解除できます。別のメンバーの登録を解除しようとするメンバーは、［獲得と中止（preempt and abort）］コマンドを発行します。ノード登録の解除は不可分であり、原子的なものです。登録を解除されたノードは別のノードの登録を解除することはできません。SFCFSHA では、1 つのノードは、デバイスのすべてのパスに対して同じキーを登録します。一度、［獲

得と中止（preempt and abort）]コマンドを実行すると、ストレージデバイスのすべてのパスから 1 つのノードを解除します。

I/O フェンシングの操作について

カーネルベースのフェンシングモジュール（vxfen）によって提供される I/O フェンシングは、ノードの障害と通信障害の発生時に同じように実行されます。ノード上のフェンシングモジュールは、GAB モジュールからクラスタメンバーシップの変更を通知されると、ただちにフェンシング操作を開始します。そのノードは、「獲得と中止（preempt and abort）」コマンドを使って、停止したノードの登録キーをコーディネータディスクから削除しようとします。 切断されたノードをコーディネータディスクから解除することに成功すると、ノードは、停止したノードの登録もデータディスクから解除します。スプリットブレインの状況で行われる場合は、スプリットブレインが発生した各ノードがコーディネータディスクの制御の取得をめぐって競合します。コーディネータディスクの定足数を取得した側が勝者になり、敗者に対するフェンシングを実行します。敗者は、パニックした後、システムを再起動します。

I/O フェンシングのコンポーネントについて

SFCFSHA の共有ストレージは、I/O フェンシングを有効にするために SCSI-3 Persistent Reservation をサポートしている必要があります。SFCFSHA には、次の 2 タイプの共有ストレージがあります。

- データディスク - 共有データを格納します。
  p.148 の「データディスクについて」を参照してください。
- コーディネーションポイント - メンバーシップの変更時にグローバルロックとして機能します。
  p.148 の「コーディネーションポイントについて」を参照してください。

データディスクについて

データディスクはデータ保存用の標準ディスクデバイスで、物理ディスクまたは RAID 論理ユニット（LUN）です。

これらのディスクは SCSI-3 PR をサポートし、標準の VxVM または CVM ディスクグループの一部である必要があります。CVM では、ディスクグループベースでデータディスクがフェンシングされます。ディスクグループに追加されたディスクと、デバイスで検出された新しいパスは、自動的にフェンシングされます。

コーディネーションポイントについて

コーディネーションポイントが提供するロック機構によって、他のノードからのデータドライブへのアクセスをブロックするノードが決定されます。ノードは、ピアをデータドライブからブロックする前に、コーディネーションポイントから削除できる必要があります。SFCFSHA

は、vxfen がコーディネーションポイントの制御権を獲得しようとし、制御権を獲得したパーティションは除外されたノードによるデータディスクへのアクセスをフェンシングするときにスプリットブレインを防ぎます。

メモ: クラスタのフェンシング設定には通常、3 つのコーディネーションポイントが必要です。Veritas Corporation では、1 つの CP サーバーをコーディネーションポイントとして使うサーバーベースフェンシングもサポートしていますが、この CP サーバーが単一障害点になる点に注意してください。

コーディネーションポイントはディスクまたはサーバーあるいはその両方の場合があります。

- コーディネータディスク
  コーディネーションポイントとして機能するディスクは、コーディネータディスクと呼ばれます。コーディネータディスクは、3 つの標準ディスクまたは LUN で、クラスタの設定時に I/O フェンシングで使うために予約されています。SFCFSHA の設定では、コーディネータディスクを他の保存目的に使うことはありません。
  コーディネータディスクは、Veritas Volume Manager の DMP（Dynamic Multi-pathing）機能を使用するように設定できます。DMP（Dynamic Multi-pathing）を実行すると、コーディネータディスクは、パスのフェールオーバーを利用したり、DMP 機能の動的な追加および削除を利用することができます。したがって、DMP デバイスを使うように I/O フェンシングを設定できます。I/O フェンシングは、使用するディスクデバイスで DMP ベースに基づく SCSI-3 ディスクポリシーを使います。

  メモ: I/O フェンシングの dmp ディスクポリシーでは、ノードからコーディネータディスクを参照するパスに単一のハードウェアパスと複数のハードウェアパスを使うことができます。いくつかのコーディネータディスクが複数のハードウェアパスを持ち、いくつかのコーディネータディスクが単一のハードウェアパスを持つ場合は、DMP ディスクポリシーのみをサポートします。新しいインストールの場合は、Veritas Corporation では単一のハードウェアパスでも IO フェンシングには DMP ディスクポリシーのみをサポートします。

  詳しくは、『Storage Foundation 管理者ガイド』を参照してください。

- コーディネーションポイントサーバー

  コーディネーションポイントサーバー（CP サーバー）はリモートシステムまたはクラスタで動作するソフトウェアソリューションです。CP サーバーは SFHA クラスタノードが次のタスクを実行できるようにすることによってアービトレーション機能を提供します。

  - （データドライブへのアクセスが CP サーバーに登録された）アクティブな SFCFSHA クラスタのメンバーになるための自己登録を行う
  - このアクティブな SFCFSHA クラスタのメンバーとして登録されている他のノードを調べる

- このアクティブな SFCFSHA クラスタから自己登録解除する
- このアクティブな SFCFSHA クラスタのメンバーである他のノードを強制的に登録解除する(獲得)

つまり、CP サーバーは既存の I/O フェンシングモジュール内で統合する別のアービトレーションメカニズムとして機能します。

メモ: CP サーバーでは、フェンシングアービトレーションのロジックは SFCFSHA クラスタ上に残ったままです。

異なるオペレーティングシステムを実行する複数の SFCFSHA クラスタ では、同時に CP サーバーにアクセスできます。TCP/IP ベースの通信が CP サーバーと SFCFSHA クラスタ の間で使用されます。

優先フェンシングについて

I/O フェンシングドライバは、コーディネーションポイントを使って VCS クラスタ内のスプリットブレインを防ぎます。デフォルトでは、フェンシングドライバはコーディネーションポイントの制御権を獲得するときに、ノード数が最大のサブクラスタを優先します。優先フェンシング機能を使うことで、存続させるサブクラスタをフェンシングドライバがどのように決定するかを指定できます。

優先フェンシングポリシーは、クラスタレベル属性 PreferredFencingPolicy を使って次のように設定できます。

- システムベース優先フェンシングポリシーを有効にして、処理能力の高いシステムを優先します。
- グループベース優先フェンシングポリシーを有効にして、優先度が高いアプリケーションのサービスグループを優先します。
- サイトベース優先フェンシングポリシーを有効にして、優先順位の高いサイトを優先します。
- 優先フェンシングポリシーを無効にして、デフォルトノード数ベースの制御権獲得ポリシーを使います。

p.150 の「優先フェンシングのしくみ」を参照してください。

p.526 の「優先フェンシングポリシーの有効化と無効化」を参照してください。

優先フェンシングのしくみ

I/O フェンシングドライバは、コーディネーションポイントを使って VCS クラスタでのスプリットブレインを防止します。ネットワーク分割時に、各サブクラスタのフェンシングドライバはコーディネーションポイントを獲得しようと競争します。コーディネーションポイントの大部分を獲得するサブクラスタが生存する一方で、フェンシングドライバが他のすべてのサブクラスタのノードでシステムパニックを引き起こします。デフォルトでは、コーディネー

ションポイントを獲得するための競争時に、フェンシングドライバは、最大数のノードが含まれるサブクラスタを優先します。

このデフォルトの獲得時の優先動作では、任意のノード上のオンラインであるアプリケーショングループまたは任意のサブクラスタのシステムの処理能力は考慮されません。たとえば、2 ノードのクラスタがあり、1 つのノードでアプリケーションを設定しており、もう 1 つのノードはスタンバイノードであるとします。ネットワーク分割が発生し、スタンバイノードが競争に勝った場合、アプリケーションが実行されているノードではパニックが発生し、VCS ではスタンバイノードでアプリケーションをオンラインにする必要が生じます。この動作により、中断が発生し、生存するノードにアプリケーションがフェールオーバーされ、再起動されるまでの時間がかかります。

優先フェンシング機能を使用することで、フェンシングドライバが生存するサブクラスタを判断する方法を指定できます。優先フェンシングソリューションでは、ノードの重みと呼ばれるフェンシングパラメータを利用します。VCS は、特定の VCS 属性を使ってユーザーが提供するオンラインアプリケーション、システムの処理能力、およびサイトの優先設定の詳細情報に基づいてノードの重みを計算し、フェンシングドライバに渡してコーディネーションポイントの獲得競争の結果に影響を与えます。競争時に、競争するノードはローカルサブクラスタと離脱するサブクラスタのすべてのノードの重みを集計します。離脱するサブクラスタの（ノードの重みの）合計の方が大きい場合、このサブクラスタの競争するノードは、コーディネーションポイントの獲得競争を遅らせます。したがって、重要なシステムまたは重要なアプリケーションがあるサブクラスタが競争に勝ちます。

優先フェンシング機能では、次の競争のポリシー値を取るクラスタレベルの属性 PreferredFencingPolicy が使われます。

- Disabled（デフォルト）: 優先フェンシングは無効です。
  PreferredFencingPolicy 属性の値を Disabled に設定すると、VCS は競争ポリシーに基づいてカウントを設定し、ノードの重みの値を 0 にリセットします。
- System: サブクラスタ内のシステムの処理能力に基づきます。
  アーキテクチャ、CPU 数、またはメモリという観点で、あるシステムが他のシステムより処理能力が高い場合、このシステムがフェンシングの競争において優先されます。
  PreferredFencingPolicy 属性の値を System に設定すると、VCS は、システムレベルの属性 FencingWeight に基づいてノードの重みを計算します。
- Group: サブクラスタ内の優先度が高いアプリケーションに基づきます。
  フェンシングドライバは、任意のサブクラスタのノード上のオンラインであるサービスグループを考慮に入れます。
  ネットワークパーティションでは、I/O フェンシングによってコーディネーションポイントの競争に参加するすべてのノード上で VCS エンジンが実行しているかどうかが特定されます。すべてのノードで VCS エンジンが実行している場合は、優先度が高いサービスグループがあるノードがフェンシング時に優先されます。
  ただし、優先度が高いサービスグループを持つノード上の VCS エンジンインスタンスが何らかの理由で強制終了されている場合は、I/O フェンシングはそのノードの優先フェンシングノードの重みをゼロにリセットします。I/O フェンシングは、メンバーシッ

プアービトレーションにそのノードを優先しません。その代わり、優先度の低いサービスグループがあるノードでも、VCS エンジンが実行しているノードを優先します。

VCS エンジンと I/O フェンシング間に同期がない場合は、VCS エンジンが実行していなくても優先度が高いサービスグループを持つノードが優先されます。つまり、優先されなかったノード上のサービスグループが、生存するノードにフェールオーバーできない状況であるということです。

PreferredFencingPolicy 属性の値を Group に設定すると、VCS は、アクティブであるそれらのサービスグループのグループレベルの属性 Priority に基づいてノードの重みを計算します。

- Site: サブクラスタ内のサイトに割り当てられた優先度に基づきます。
  クラスタ属性の SiteAware を 1 に設定した場合にのみ、サイトポリシーが利用できます。VCS は優先度が高いサイト内のノードに高い重みを設定し、優先度が低いサイト内のノードに低い重みを設定します。ノードの累積した重みが一番高いサイトが優先サイトになります。サイト間のネットワークパーティションでは、コーディネーションポイントの獲得競争で優先されたサイトのノードを持つサブクラスタを VCS は優先します。

VCS 属性について詳しくは、『Cluster Server 管理者ガイド』を参照してください。

p.526 の「優先フェンシングポリシーの有効化と無効化」を参照してください。

I/O フェンシングの設定ファイルについて

表 5-3 に、I/O フェンシングの設定ファイルのリストを示します。

表 5-3　I/O フェンシングの設定ファイル

ファイル	説明
/etc/sysconfig/vxfen	このファイルには I/O フェンシングの起動と停止の環境変数が格納されます。 ■ VXFEN_START - システムの再起動後における I/O フェンシングモジュールの起動動作を定義します。有効な値は次のとおりです。 1 - I/O フェンシングの起動が有効であることを示します。 0 - I/O フェンシングの起動が無効であることを示します。 ■ VXFEN_STOP - システムのシャットダウン中における I/O フェンシングモジュールのシャットダウン動作を定義します。有効な値は次のとおりです。 1 - I/O フェンシングのシャットダウンが有効であることを示します。 0 - I/O フェンシングのシャットダウンが無効であることを示します。 SFCFSHA の設定の終わりに、インストーラはこれらの変数の値を 1 に設定します。
/etc/vxfendg	このファイルにはコーディネータディスクグループの情報が含まれています。 このファイルはサーバーベースのフェンシングとマジョリティベースのフェンシングには適用できません。

ファイル	説明
/etc/vxfenmode	このファイルには次のパラメータがあります。 ■ vxfen_mode ■ scsi3 - ディスクベースのフェンシングの場合 ■ customized - サーバーベースのフェンシングの場合 ■ disabled - I/O フェンシングドライバを実行するが、フェンシング操作は行わない場合 ■ majority - コーディネーションポイントを使わないフェンシングの場合 ■ vxfen_mechanism このパラメータはサーバーベースのフェンシングにのみ適用できます。値は CP と設定します。 ■ scsi3_disk_policy ■ dmp - DMP デバイスを使うように vxfen モジュールを設定します。 このディスクポリシーは、デフォルトでは DMP です。iSCSI のデバイスを使う場合は、ディスクポリシーを DMP と設定する必要があります。 **メモ:** すべてのノードで同じ SCSI-3 ディスクポリシーを使う必要があります。 ■ コーディネーションポイントのリスト このリストはサーバーベースのフェンシング設定でのみ必要です。 サーバーベースのフェンシングのコーディネーションポイントには、コーディネータディスク、CP サーバー、または両方を含めることができます。コーディネータディスクを使う場合は、個々のコーディネータディスクを含むコーディネータディスクグループを作成する必要があります。 コーディネーションポイントと複数の IP アドレスを各 CP サーバーに指定する方法について詳しくは、サンプルファイル /etc/vxfen.d/vxfenmode_cps を参照してください。 ■ single_cp このパラメータは、1 つの高可用性 CP サーバーをコーディネーションポイントとして使うサーバーベースのフェンシングに適用できます。また、単一ディスクを持つコーディネータディスクグループを使う場合にも適用できます。 ■ autoseed_gab_timeout このパラメータを使うと、使用できないクラスタノードがある場合でも、GAB のクラスタの自動シーディングを使用できるようになります。 この機能は、SCSI3 とカスタマイズしたモードの I/O フェンシングに適用できます。 0 - GAB の自動シーディング機能を有効にします。0 より大きい値は、自動的にクラスタをシーディングする前の GAB の遅延時間（秒数）を示します。 -1 - GAB の自動シーディング機能をオフにします。この設定がデフォルト値です。 ■ detect_false_pesb 0 - 古いキーの検出を無効にします。 1 - 古いキーの検出を有効にして、すでに発生しているスプリットブレインが実際に発生している状態か、誤検出のアラームかを判断します。 デフォルト: 0 **メモ:** このパラメータは、`vxfen_mode=customized` の場合のみ考慮されます。

ファイル	説明
/etc/vxfentab	I/O フェンシングを起動すると、起動スクリプト vxfen で各ノード上にこの /etc/vxfentab ファイルが作成されます。この起動スクリプトは /etc/vxfendg および /etc/vxfenmode ファイルのコンテンツを使用します。システムが再起動されると必ず、フェンシングドライバによって、すべてのコーディネータポイントの最新リストで vxfentab ファイルが再初期化されます。 **メモ:** /etc/vxfentab ファイルは生成ファイルであるため、変更しないでください。 ディスクベースの I/O フェンシングの場合は、各ノード上の /etc/vxfentab ファイルに各コーディネータディスクへのすべてのパスと一意のディスク識別子の一覧が含まれます。パスと一意のディスク識別子はスペースで区切ります。あるノードでのディスクベースのフェンシングの設定における /etc/vxfentab ファイルの例を次に示します。 ■ DMP ディスク: /dev/vx/rdmp/sdx3 HITACHI%5F1724-100%20%20FAStT%5FDISKS%5F6 00A0B8000215A5D000006804E795D0A3 /dev/vx/rdmp/sdy3 HITACHI%5F1724-100%20%20FAStT%5FDISKS%5F6 00A0B8000215A5D000006814E795D0B3 /dev/vx/rdmp/sdz3 HITACHI%5F1724-100%20%20FAStT%5FDISKS%5F6 00A0B8000215A5D000006824E795D0C3 サーバーベースのフェンシングの場合は、/etc/vxfentab ファイルにセキュリティ設定の情報も含まれます。 1 つの CP サーバーのみのサーバーベースのフェンシングの場合は、/etc/vxfentab ファイルに single_cp 設定情報も含まれます。 このファイルはマジョリティベースのフェンシングには適用できません。

各種のイベントシナリオにおける I/O フェンシングの動作

表 5-4 に、障害イベントの各種シナリオで、データの破損を回避するために I/O フェンシングがどのように動作するかを示します。イベントごとに、オペレータの対処方法を確認します。

表 5-4 I/O フェンシングのシナリオ

イベント	ノード A の動作	ノード B の動作	オペレータのアクション
両方のプライベートネットワークで障害が発生。	ノードは、コーディネーションポイントの定足数を獲得しようとします。 ノード A は、コーディネーションポイントの競合に勝利すると、ノード B を共有ディスクの登録から除外して処理を続行します。	ノード B は、コーディネーションポイントの定足数を獲得しようとします。 ノード B は、コーディネーションポイントの競合に敗退すると、パニックになり、クラスタから自分自身を削除します。	ノード B がクラスタから切り離されたら、ノード B を復帰させる前にプライベートネットワークを修復します。
上記イベント後に、両方のプライベートネットワークが再び正常に動作する。	ノード A は処理を続行します。	ノード B はクラッシュしています。単に再起動しただけでは、ノード B はデータディスクに書き込めないため、データベースを起動できません。	プライベートネットワークが復元されたらノード B を再起動する。
一方のプライベートネットワークでエラーが発生する。	ノード A は、I/O フェンスのエラーメッセージをコンソールに表示し、処理を続行します。	ノード B は IOFENCE に関するメッセージをコンソールに表示するが、動作を続ける。	プライベートネットワークを修復します。ネットワークの修復後に、両方のノードは修復したネットワークを自動的に使います。

イベント	ノード A の動作	ノード B の動作	オペレータのアクション
ノード A がハングする。	ノード A は、何らかの理由で異常なビジー状態になっているか、カーネルデバッガに制御されています。 ノード A がハング状態またはカーネルデバッガの制御から解放されると、キューイングされていたデータディスクへの書き込みは、ノード A の登録が解除されているため失敗します。ノード A は、削除されていることに関するメッセージを GAB から受け取ると、パニックになり、自身をクラスタから削除します。	ノード B はノード A とのハートビートを失い、コーディネーションポイントの定足数を獲得しようとします。 ノード B はコーディネーションポイントの競合に勝利すると、ノード A の登録を共有ディスクから解除します。	ハングしたノードを修復またはデバッグし、クラスタを再結合するためにノードを再ブートします。

イベント	ノード A の動作	ノード B の動作	オペレータのアクション
ノード A、ノード B の電源およびプライベートネットワークが遮断される。コーディネーションポイントおよびデータディスクの電源は維持される。 ノードに電源が戻り再起動されるが、プライベートネットワークは依然として電源がオフになっている。	ノード A は再起動され、I/O フェンシングドライバ (vxfen) はノード B がコーディネーションポイントに登録されていることを検出する。プライベートネットワークがダウンしているため、ドライバはノード B がクラスタのメンバーとして一覧表示されていることを認識しません。このため、I/O フェンシングデバイスドライバはノード A がクラスタに参加することを拒否します。ノード A のコンソールは次の内容を表示します。 `Potentially a preexisting split brain. Dropping out of the cluster. Refer to the user documentation for steps required to clear preexisting split brain.`	ノード B は再起動され、I/O フェンシングドライバ (vxfen) はノード A がコーディネーションポイントに登録されていることを検出する。プライベートネットワークがダウンしているため、ドライバはノード A がクラスタのメンバーとして一覧表示されていることを認識しません。このため、I/O フェンシングデバイスドライバはノード B がクラスタに参加することを拒否します。ノード B のコンソールは次の内容を表示します。 `Potentially a preexisting split brain. Dropping out of the cluster. Refer to the user documentation for steps required to clear preexisting split brain.`	すでに発生しているスプリットブレイン状態を解決する。

イベント	ノード A の動作	ノード B の動作	オペレータのアクション
ノード B がダウンしている間にノード A がクラッシュする。ノード B は再起動されるが、ノード A は停止したままである。	ノード A はクラッシュしています。	ノード B は再起動され、ノード A がコーディネーションポイントに登録されていることを検出する。ドライバは、ノード A がクラスタのメンバーとして一覧表示されていることを認識しません。I/O フェンシングデバイスドライバはコンソールに次のメッセージを表示します。 `Potentially a preexisting split brain. Dropping out of the cluster. Refer to the user documentation for steps required to clear preexisting split brain.`	すでに発生しているスプリットブレイン状態を解決する。
3 つのコーディネーションポイントのうちの 2 つを含むディスクアレイの電源が遮断される。 いずれのノードもクラスタメンバーシップから切り離されません。	ノード A は、クラスタから切り離されるノードがないかぎり動作を続行します。	ノード B は、クラスタから切り離されるノードがないかぎり動作を続行します。	障害が発生したディスクアレイの電源を入れ、以降のネットワーク分割によってクラスタがシャットダウンしないようにするか、コーディネーションポイントを交換します。 p.489 の「クラスタがオンラインのときの I/O フェンシングコーディネータディスクの置き換え」を参照してください。

イベント	ノード A の動作	ノード B の動作	オペレータのアクション
3 つのコーディネーションポイントのうちの 2 つを含むディスクアレイの電源が遮断される。 ノード B はクラスタから正常に切り離され、ディスクアレイは電源が切れたままである。正常な切り離しはクリーンシャットダウンを意味するため、vxfen が適切に設定解除される。	ノード A はクラスタで動作を続行します。	ノード B はクラスタから切り離されました。	障害が発生したディスクアレイの電源を入れ、以降のネットワーク分割によってクラスタがシャットダウンしないようにするか、コーディネーションポイントを交換します。 p.489 の「クラスタがオンラインのときの I/O フェンシングコーディネータディスクの置き換え」を参照してください。
3 つのコーディネーションポイントのうちの 2 つを含むディスクアレイの電源が遮断される。 ノード B が不意にクラッシュするか、ノード A およびノード B 間でネットワーク分割が発生し、ディスクアレイは電源が切れたままである。	ノード A は、コーディネーションポイントの定足数を獲得しようとします。3 つのコーディネーションポイントのうち 1 つしか使えないため、ノード A は定足数の確保に失敗します。ノード A は、パニックになり、クラスタから自身を削除します。	クラッシュまたはネットワーク分割が原因で、ノード B はクラスタから切り離されました。	障害が発生したディスクアレイの電源を入れ、I/O フェンシングドライバを再起動して、ノード A をすべてのコーディネーションポイントに登録できるようにするか、コーディネーションポイントを交換します。

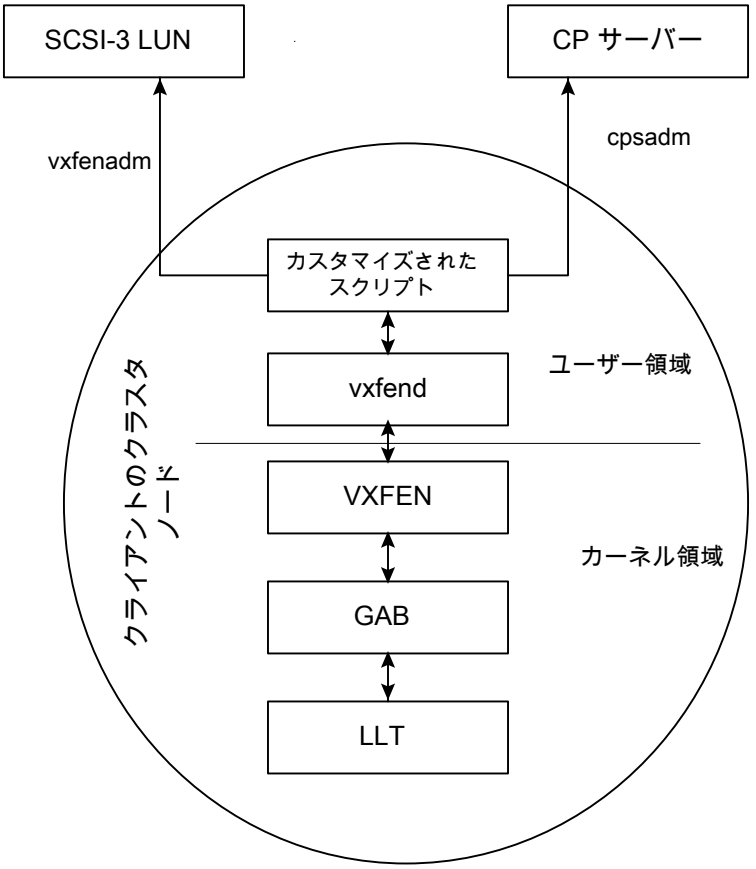
サーバーベースの I/O フェンシングについて

ディスクベースの I/O フェンシングの実装で、vxfen ドライバは SCSI-3 PR ベースのアービトレーション操作をすべてドライバ内部で処理します。I/O フェンシングはカスタマイズされたフェンシングと呼ばれるフレームワークも提供します。このフレームワークでは、アービトレーション操作はカスタムスクリプトとして実装されます。vxfen ドライバはカスタムスクリプトを呼び出します。

CP サーバーベースのコーディネーションポイントでは、カスタマイズされたフェンシングのフレームワークを使います。SCSI-3 PR 型のフェンシングアービトレーションをカスタマイズされたフェンシングフレームワークを使用して有効にすることもできます。これにより、ユーザーはカスタマイズされたフェンシングを使って、SCSI-3 LUN と CP サーバーの組み合わせをコーディネーションポイントとして指定できます。カスタマイズされたフェンシングは、`/etc/vxfenmode` ファイルで vxfen_mode=customized および vxfen_mechanism=cps を指定することによって有効にできます。

図 5-2 に、カスタマイズされたフェンシングオプションの概念図を示します。

図 5-2　　カスタマイズされたフェンシング



ユーザーレベルのデーモン vxfend は vxfen ドライバと対話し、vxfen ドライバは GAB と対話してノードのメンバーシップ更新を取得します。メンバーシップ更新を受信すると、vxfend は各種のスクリプトを呼び出してコーディネーションポイントの制御権を獲得し、データディスクを遮蔽します。vxfend デーモンは各種のフェンシングエージェントを管理します。カスタマイズされたフェンシングスクリプトは `/opt/VRTSvcs/vxfen/bin/customized/cps` ディレクトリにあります。

該当するスクリプトには次のものがあります。

- generate_snapshot.sh : コーディネータディスクの SCSI ID や CP サーバーの UUID を取得します。
  CP サーバーは、`/etc/VRTScps/db/current/cps_uuid` に格納されている UUID を使います。

UUID (全世界で一意の ID) については、『Cluster Server 管理者ガイド』を参照してください。

- join_local_node.sh: コーディネータディスクまたは CP サーバーにキーを登録します。
- race_for_coordination_point.sh: クラスタ再設定後、勝者を決定するために競合します。
- unjoin_local_node.sh : join_local_node.sh で登録されるキーを削除します。
- fence_data_disks.sh: 接続を失ったノードによるアクセスからデータディスクをフェンシングします。
- local_info.sh: vxfen ドライバによって使われるローカルノードの設定パラメータとコーディネーションポイントを表示します。
- validate_pesb_join.sh:
  - すでに発生しているスプリットブレインが実際に発生している状態か、コーディネーションポイントに古いキーが存在することによる誤検出のアラームなのかどうかを判断します。
  - すでに発生しているスプリットブレインが誤検出のアラームであることがわかるとコーディネーションポイントをクリアします。

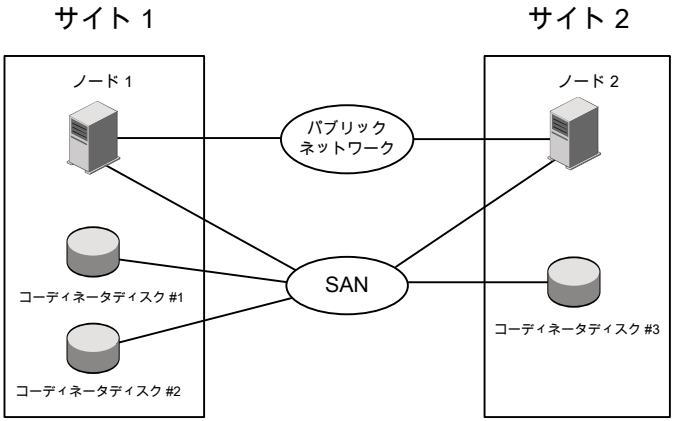
CP サーバーによって提供される I/O フェンシングの拡張

CP サーバーの設定は次の新しい機能の提供によってディスク型の I/O フェンシングを拡張します。

- CP サーバーの設定は拡張性があり、3 つの CP サーバーによる設定で複数 SFCFSHA クラスタに I/O フェンシングを提供できます。単一の CP サーバーの設定は多数の SFCFSHA クラスタに対応できるので複数の SFCFSHA クラスタの配備の費用は大幅に減らすことができます。
- 適切に配置された CP サーバーにより、I/O フェンシングプロセスのコーディネータディスクの場所バイアスを除去できます。たとえば、こうした位置的な偏りは、設備面の制限によって 3 つのコーディネータディスクのうち 2 つを単一サイトに配置し、コスト不足のため 3 つ目のコーディネータディスクの配置先を整備できないような状況で起きることがあります。
  p.162 の 図 5-3 を参照してください。
  このような設定では、2 つのコーディネータディスクのあるサイトがアクセス不能の場合、他のサイトはコーディネーションポイントの大半が不足するため存続しません。I/O フェンシングは 3 つ目のサイトへ SAN を拡張することを要求しますが、これは適切な解決策ではない場合があります。代替方法は、CP サーバーを 3 つ目のコーディネーションポイントとしてリモートサイトに配置することです。

メモ: CP サーバーは SCSI-3 準拠コーディネータディスクに依存する必要のない代替のアービトレーションメカニズムを提供します。Cluster Volume Manager（CVM）でのデータディスクフェンシングには、依然として SCSI-3 I/O フェンシングが必要です。

図 5-3　　サイト 1 でのコーディネータディスクの非対称配置



CP サーバーデータベースについて

CP サーバーは SFCFSHA クラスタ のノードの登録キーを格納するためにデータベースを必要とします。CP サーバーは処理に SQLite データベースを使います。デフォルトでは、データベースは /etc/VRTScps/db にあります。

CP サーバーのホストが単一ノード VCS クラスタの場合、データベースはローカルファイルシステムに配置できます。CP サーバーをホストする SFHA クラスタの場合、データベースは共有ファイルファイルシステムに配置する必要があります。ファイルシステムは SFHA クラスタの一部であるすべてのノード間で共有する必要があります。

CP サーバーのホストである SFHA クラスタで、共有データベースは SCSI-3 PR ベースの I/O フェンシングを設定することによって保護されます。SCSI-3 PR 型の I/O フェンシングはスプリットブレインシナリオから保護します。

警告: CP サーバーのデータベースは、直接編集してはならず、`cpsadm`（1M）を使ってのみアクセスする必要があります。データベースを手動で操作すると、システムパニックなど望ましくない結果が発生する場合があります。

CP サーバーのユーザータイプと権限について

CP サーバーは次のユーザー種類をサポートし、それぞれのユーザーが異なるアクセスレベルの権限を持ちます。

- CP サーバーの管理者（admin）
- CP サーバーのオペレータ

アクセスレベルの権限が異なるとユーザーが発行できるコマンドも異なります。ユーザーが CP サーバーの admin ユーザーでもなく CP サーバーのオペレータユーザーでもない場合、ユーザーはゲストステータスを持ち、限られたコマンドを発行できます。

ユーザー種類とアクセスレベルの権限は、フェンシング用の SFCFSHA クラスタ設定の間に個々のユーザーに割り当てられます。インストール処理中、ユーザー名、パスワード、およびアクセスレベルの権限（CP サーバーの admin または CP サーバーのオペレータ）を入力するよう求められます。

CP サーバーを管理し、操作するには、少なくとも CP サーバー の admin である必要があります。

CP サーバーのルートユーザーはすべての管理者権限を与えられ、すべての CP サーバー固有の操作を実行するためにこれらの管理者権限を使用できます。

SFCFSHA クラスタと CP サーバー間のセキュア通信について

データセンターでは、SFCFSHA クラスタ（アプリケーションクラスタ）と CP サーバー間の TCP/IP 通信のセキュリティを確保する必要があります。 通信チャネルのセキュリティには、暗号化、認証、および認可が含まれます。

CP サーバーのノードまたはクラスタは、コーディネーションポイントとして自身と通信する SFCFSHA クラスタノードの真正性を確認し、既知の SFCFSHA クラスタノードからの要求のみを受け入れるようにする必要があります。 不明なクライアントからの要求は非認証として拒否されます。 同様に、SFCFSHA クラスタ内のフェンシングフレームワークは、CP サーバーを使ってフェンシング操作を実行しているのが認証済みのユーザーであることを確認する必要があります。

CP サーバーと SFCFSHA クラスタ間の通信のセキュアモードは HTTPS 通信です。

HTTPS 通信: SSL インフラは通信が安全であることを確認するために、クライアントクラスタ証明書と CP サーバー証明書を使います。 HTTPS モードは、認証サーバーのクレデンシャルを作成するためにブローカーメカニズム使いません。

Veritas Product Authentication Services (AT) を使って CP サーバーと SFCFSHA クラスタ間の通信を保護する方法

Veritas Product Authentication Services (AT): 認証が行われるエンティティはプリンシパルと呼ばれます。SFCFSHA クラスタノード上で、現在の VCS インストーラはクラスタ内の各ノード上で認証サーバーの信用証明を作成します。また、信用証明を認証する

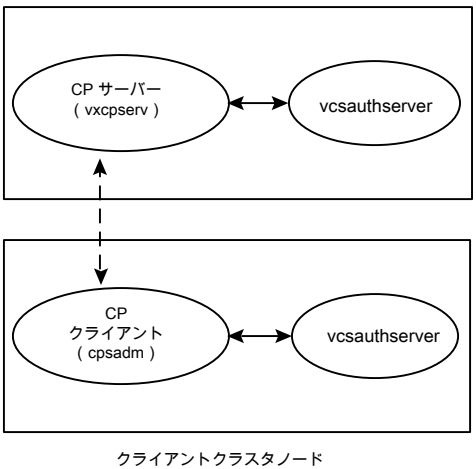
vcsauthserver を作成します。次にインストーラは、セキュアモードでの VCS の起動に進みます。通常、セキュリティが設定された既存の VCS クラスタでは、vcsauthserver は各クラスタノード上で動作します。

CP サーバーと SFCFSHA クラスタ(アプリケーションクラスタ)ノードの通信には、次のエンティティが関係します。

- CP サーバー用の vxcpserv
- SFCFSHA クラスタノード用の cpsadm

図 5-4 に、CP サーバーと SFCFSHA クラスタ上でセキュリティを有効にした場合の終端間通信フローの概略図を示します。

図 5-4 **CP サーバーと SFCFSHA クラスタ上でセキュリティを有効にした場合の終端間通信フロー**



CP サーバーと SFCFSHA クラスタノード上でセキュリティを設定した状態の、両者間の通信フローは次のようになります。

- 初期設定:
  CP サーバーと SFCFSHA クラスタノードの ID は、VCS インストーラによってそれぞれノードで設定されます。

メモ: フェンシングの設定時に、インストーラは CP サーバーとアプリケーションクラスタ間の信頼を確立して、vxcpserv プロセスでアプリケーションクラスタノードからの要求を認証できるようにします。I/O フェンシングを手動で設定した場合は、CP サーバーとアプリケーションクラスタ間の信頼を設定する必要があります。

---

cpsadm コマンドは環境変数 CPS_USERNAME、CPS_DOMAINTYPE からユーザー名、ドメインタイプを取得します。cpsadm コマンドを手動で実行する前に、これらの変数をエクスポートします。カスタマイズされたフェンシングフレームワークは cpsadm のコマンドを実行する前にこれらの環境変数を内部でエクスポートします。
CP サーバーのプロセス（vxcpserv）が使うそのプロセス自身のユーザー（CPSERVER）は、ローカルの vcsauthserver に追加されます。

- 認証ブローカーからのクレデンシャルの取得:
  cpsadm コマンドは、ローカルノードにある既存の信用証明を取得しようとします。インストーラはこれらの信用証明をフェンシングの設定中に生成します。
  vxcpserv プロセスは、ローカルノードにある既存の信用証明を取得しようとします。インストーラはこれらの信用証明をセキュリティを有効にするときに生成します。
- CP サーバーと SFCFSHA クラスタノード間の通信:
  CP サーバーが信用証明を確立し、稼動状態になると、CP サーバーでクライアントからデータを受信する準備が整います。cpsadm コマンドで信用証明を取得し、CP サーバーの信用証明を認証すると、cpsadm は CP サーバーに接続します。CP サーバーにデータが渡されます。
- 検証:
  特定の SFCFSHA クラスタノードからデータを受信すると、vxcpserv はその信用証明を検証します。検証が失敗した場合、接続要求データは拒否されます。

CP サーバーと SFCFSHA クラスタ上のセキュリティ設定の詳細

ここでは CP サーバーと SFCFSHA クラスタ（アプリケーションクラスタ）のセキュリティ設定の詳細を説明します。

Veritas Product Authentication Services (AT) セキュアモードの設定

以下は CP サーバーと SFCFSHA クラスタ間のセキュリティで保護された通信の設定です:

- CP サーバーの設定:
  インストーラにより、次の値を持つユーザーが作成されます:
  - username: CPSERVER
  - domainname: VCS_SERVICES@*cluster_uuid*
  - domaintype: vx

  設定を確認するには、CP サーバーで次のコマンドを実行します:

```
# export EAT_DATA_DIR=/var/VRTSvcs/vcsauth/data/CPSERVER

 # /opt/VRTScps/bin/cpsat showcred
```

メモ: CP サーバーの設定ファイル（/etc/vxcps.conf）は security=0 を指定する行を含みません。security パラメータを指定する行がない、または security=1 を指定する行がある場合、CP サーバーのセキュリティは有効（デフォルト値です）になります。

- SFCFSHA クラスタノードの設定:

  SFCFSHA クラスタで、インストーラは、フェンシングの設定中に次の値を持つ cpsadm のユーザーを作成します。

  - username: CPSADM
  - domainname: VCS_SERVICES@*cluster_uuid*
  - domaintype: vx

  セキュリティ設定を確認するには、SFCFSHA クラスタノードで次のコマンドを実行します:

```
# export EAT_DATA_DIR=/var/VRTSvcs/vcsauth/data/CPSADM

 # /opt/VRTScps/bin/cpsat showcred
```

上で説明したユーザーは、CP サーバーと SFCFSHA クラスタノード間の通信の認証のみに使用されます。

CP サーバーの認証では、セキュリティが設定されている場合、SFCFSHA クラスタ上のカスタマイズされたフェンシングフレームワークで次のユーザーが使用されます。

CPSADM@VCS_SERVICES@*cluster_uuid*

ここで、*cluster_uuid* はアプリケーションクラスタの汎用一意識別子です。

各 SFCFSHA クラスタノードで、SFCFSHA クラスタノードでのフェンシングを開始する前にこのユーザーが CP サーバーデータベースに登録される必要があります。これは次のコマンドを実行することによって確認できます。

```
# cpsadm -s cp_server -a list_users
```

コマンドの出力の例を次に示します。

```
Username/Domain Type
CPSADM@VCS_SERVICES@77a2549c-1dd2-11b2-88d6-00306e4b2e0b/vx

Cluster Name / UUID                          Role
cluster1/{77a2549c-1dd2-11b2-88d6-00306e4b2e0b} Operator
```

メモ: 各クライアントのノードの設定ファイル（/etc/vxfenmode）は security=0 を指定する行を含みません。security パラメータを指定する行がない、または security=1 を指定する行がある場合、クライアントノードはセキュリティが有効な状態（デフォルト値です）で起動します。

非セキュアモードの設定

非セキュアモードでは、認可のみが CP サーバーで提供されます。パスワードは要求されません。認証と暗号化は提供されません。「vx」ドメインタイプの「cpsclient@hostname」のユーザークレデンシャルは、CP サーバーまたは SFCFSHA クラスタノード間の通信のカスタマイズされたフェンシングフレームワークによって使われます。

各 SFCFSHA クラスタノードで、SFCFSHA クラスタノードでのフェンシングを開始する前にこのユーザーが CP サーバーデータベースに追加される必要があります。ユーザーは次のコマンドを発行することによって確認できます。

```
# cpsadm -s cpserver -a list_users
```

コマンドの出力の例を次に示します。

```
Username/Domain Type  Cluster Name / UUID                  Role
cpsclient@sys1/vx   cluster1 / {f0735332-e3709c1c73b9}  Operator
```

メモ: 非セキュアモードで、CP サーバーの設定ファイル（/etc/vxcps.conf）は security=0 を指定する行を含むはずです。同様に、各 SFCFSHA クラスタノードでは、設定ファイル（/etc/vxfenmode）は security=0 を指定する行を含むはずです。

Storage Foundation Cluster File System High Availability と Veritas Volume Manager のクラスタ機能エージェント

エージェントは、事前定義済みのリソースタイプを管理する VCS プロセスです。SFCFSHA と CVM は、VCS とやり取りするためにエージェントを必要とします。エージェントには、リソースをオンラインまたはオフラインにする、リソースを監視する、状態の変化を VCS に報告するなどの機能があります。VCS 付属エージェントは VCS の一部であり、VCS のインストール時にインストールされます。SFCFSHA エージェントと CVM エージェントは VCS のアドオンリソースであり、それぞれ Veritas File System と Veritas Volume Manager に固有です。

『Storage Foundation Cluster File System High Availability インストールガイド』を参照してください。

Veritas Volume Manager のクラスタ機能

Veritas Volume Manager クラスタ機能（CVM）を使うと、クラスタ全体のアプリケーションから論理ボリュームにアクセスできるようになります。CVM により、複数のホストから制御下にある論理ボリュームに並列アクセスできるようになります。VxVM クラスタは、一連のデバイスを共有するノードで構成されます。これらのノードは、1 つのネットワークを介して接続されます。1 つのノードに障害が発生しても、他のノードはデバイスにアクセスできます。VxVM クラスタ機能により、すべてのノードで、変更などのデバイス設定が同じ論理ビューとして表示されます。VCS によってクラスタの設定がセットアップされたら、CVM 共有ストレージを設定します。

6

Cluster Volume Manager の動作

この章では以下の項目について説明しています。

VxVM のクラスタ機能について

クラスタは、ディスクの集合を共有する多数のホストまたはノードで構成されます。クラスタ設定の主な利点は、次のとおりです。

可用性	1 つのノードが失敗しても、他のノードは共有ディスクに引き続きアクセスできます。適したソフトウェアが設定されている場合、ミッションクリティカルなアプリケーションは、実行をクラスタのスタンバイノードに転送することによって、動作し続けることができます。冗長ハードウェアに切り替えることで連続的で途切れることのないサービスを提供できるこの機能を、一般にフェールオーバーと呼びます。 フェールオーバーは、データベースやファイル共有の、ユーザーや上位レベルのアプリケーションに対して透過的です。Veritas Cluster Server（VCS）などのクラスタ管理ソフトウェアを使ってシステムおよびサービスを監視し、ハードウェア障害またはソフトウェア障害のいずれかが発生した場合に他のノード上でアプリケーションが再起動されるように設定しておく必要があります。VCS では、ノードのクラスタへの参加または切り離しなど、一般的な管理タスクの実行もできます。 スタンバイノードはアイドル状態のままになっている必要はないことに注意してください。同時に他のアプリケーションへのサービスの提供に利用できます。
オフホスト処理	クラスタ内のより低負荷のノードでバックアップ、意思決定支援、レポート生成などのアクティビティを実行することで、システムリソースの競合を減少させることができます。これにより、企業はクラスタシステムへの投資以上の価値を導出できます。

SFCFSHA（Storage Foundation Cluster File System High Availability）は、CVM（Cluster Volume Manager）をコンポーネントとして含んでいます。CVM は、VxVM（Veritas Volume Manager）の機能を拡張し、クラスタ環境のサポートを追加します。CVM では、クラスタノードが VxVM の制御下にある一連のディスクまたは LUN に同時にアクセスして、それを管理できます。すべてのノードでディスク構成とその論理表示に対するすべての変更について、同一の論理表示を利用できます。CVM 機能が有効なとき、すべてのクラスタノードが共有ディスクグループなどの VxVM オブジェクトを共有できます。専用ディスクグループはクラスタ以外の環境と同様にサポートされます。

クラスタ化の概要

企業規模のミッションクリティカルなデータ処理の分野では、密結合されたクラスタシステムが一般的に使われています。クラスタの第一のメリットは、ハードウェア障害に対する保護です。障害の発生やその他の理由でプライマリノードが使用できなくなっても、クラスタ内のスタンバイノードに実行制御を転送することによって、アプリケーションの実行を継続できます。冗長ハードウェアに切り替えることでサービスの連続的な可用性を提供するこの機能を、一般にフェールオーバーと呼びます。

また、クラスタ化されたシステムのもう 1 つの大きなメリットとして、バックアップ、意思決定支援、レポート生成などのアクティビティにより、システムリソースの競合を減少させる機能があります。このような処理を、サービスの要求に応答する負荷の重いノード上ではなく、クラスタ内の負荷の軽いノード上で行うことによって、クラスタシステムへの投資からビジ

ネスに付加価値をもたらすことができます。一部の操作を低負荷のノードで実行するこのような機能を、一般に負荷分散と呼びます。

クラスタボリューム管理の概要

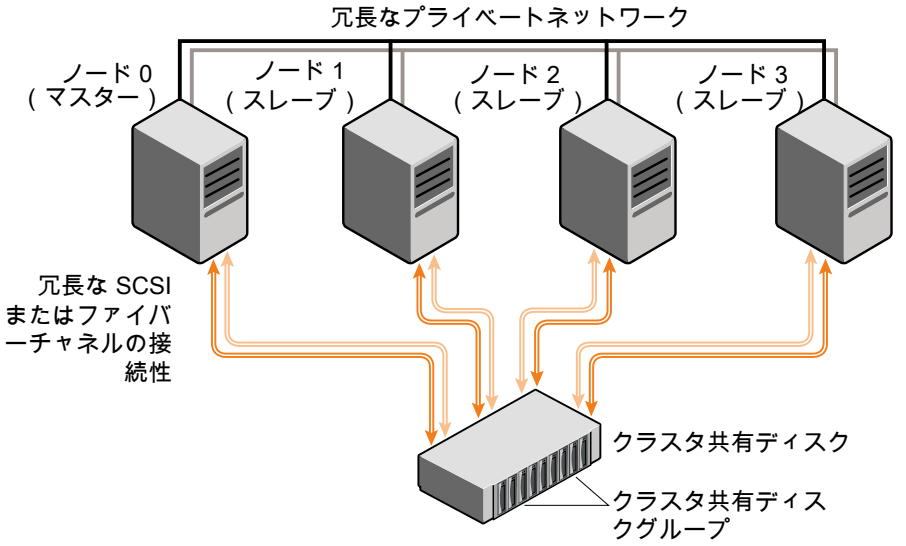
過去数年間に、共有データアクセスを使う並列アプリケーションがますます一般的になってきました。現在入手できる商用アプリケーションの例としては、Oracle Real Application Clusters™（RAC）、Sybase Adaptive Server®、Informatica Enterprise Cluster Edition が挙げられます。さらに、NFS（Network File System）、FTP（File Transfer Protocol）、NNTP（Network News Transfer Protocol）のセマンティクスでは、これらの作業負荷を共有データアクセスクラスタによって処理できます。そして多数の組織が、共有データアクセスクラスタを利用する社内アプリケーションを開発してきました。

VxVM のクラスタ機能（CVM）は、VCS またはホスト OS に装備されているクラスタモニタデーモンと連携します。クラスタモニタは、クラスタメンバーシップの変更を VxVM に通知します。VxVM の各ノードは独自に起動し、クラスタモニタに加え、OS と VxVM/CVM のコピーをそれぞれ独自に備えています。あるノードをクラスタに結合すると、そのノードは共有ディスクグループおよびボリュームにアクセスできるようになります。ノードがクラスタから切断されると、そのノードは共有ディスクにアクセスできなくなります。ノードに対して適切なコマンドを発行すると、そのノードはクラスタに参加します。

警告: VxVM の CVM 機能は、VxVM との連携を意図して正しく設定されたクラスタモニタと共に使う場合にかぎりサポートされます。

図 6-1 は、類似または同一のハードウェア特性（CPU、RAM およびホストアダプタ）を持ち、同一のソフトウェア（OS を含む）で設定されたノード構成の単純なクラスタを示しています。

図 6-1 4 ノード CVM クラスタの例



クラスタモニタにとって、すべてのノードが等価です。共有ディスクグループ内に設定された VxVM オブジェクトは、潜在的に、クラスタに参加しているすべてのノードからアクセスされる可能性があります。ただし、VxVM の CVM 機能では、ノードの 1 つがマスターノードとして機能し、クラスタ内の他のノードすべてがスレーブノードとして機能するメンバーシップが要求されます。 任意のノードがマスターノードになり、特定の VxVM のアクティビティの調整を担当することができます。

この例では、ノード 0 が CVM マスターノードとして設定されていて、ノード 1、2、3 は CVM スレーブノードとして設定されています。ノードはプライベートネットワークによって完全に接続され、ストレージエリアネットワーク(SAN: Storage Area Network)で共有する外部ストレージ(ディスクアレイまたは JBOD(just a bunch of disk)のいずれか)に、SCSI またはファイバーチャネルを介してそれぞれ接続されています。

図 6-1 に各ノードがディスク(1 つ以上のクラスタ共有ディスクグループで構成される)への 2 つの独立したパスを持つことを示します。パスが複数あると、一方のパスに障害が発生した場合でも可用性を維持することができます。ただし、これはクラスタ設定の必要条件ではありません。ディスクを接続するパスは 1 つでも構いません。

プライベートネットワークにより、ノードはシステムリソースと互いの状態についての情報を共有できます。プライベートネットワークを使って、すべてのノードから他の現在アクティブなノード、クラスタに参加しているノード、クラスタから切り離されているノード、障害が発生しているノードを認識できます。プライベートネットワークは、チャネルの 1 つで障害が発生した場合に備えた冗長性を確保するために、少なくとも 2 つの通信チャネルを必要とします。1 つのチャネルのみが使われていた場合、その障害はノードの障害と区別が付きません。これは「ネットワーク分割」として知られる状況です。

VxVM オブジェクトを設定または再設定するコマンドは、クラスタ内の任意のノードで実行できます。これらの作業には、共有ディスクグループの設定、ボリュームの作成と再設定、スナップショット操作の実行などがあります。

クラスタに最初に参加したノードがマスターノードの機能を実行します。マスターノードがクラスタから切断されると、スレーブノードの 1 つが新しいマスターとして選択されます。

p.424 の「CVM マスター選択を制御する方法」を参照してください。

専有および共有ディスクグループについて

次のタイプのディスクグループが定義されます。

専用ディスクグループ	1 つのノードだけに属します。専用ディスクグループのインポートは、1 つのシステムでのみ行えます。専用ディスクグループ内の LUN には、物理的には 1 つ以上のシステムからアクセスできますが、アクセスは一度に 1 つのシステムのみに制限されます。 ブートディスクグループ（通常、bootdg という予約済みディスクグループ名がエイリアスとして設定される）は常に、専用ディスクグループです。
共有ディスクグループ	すべてのノードで共有できます。共有（またはクラスタ共有）ディスクグループは、すべてのクラスタノードにインポートされます。共有ディスクグループ内の LUN は、クラスタに参加可能なすべてのシステムから、物理的にアクセスできる必要があります。

CVM クラスタでは、大多数のディスクグループが共有されます。共有ディスクグループ内の LUN は、クラスタ内のすべてのノードからアクセスでき、複数のクラスタノード上のアプリケーションが同じ LUN に同時にアクセスすることもできます。共有ディスクグループ内のボリュームは、ライセンスキーおよびディスクグループアクティブ化モードによる制約の範囲内で、クラスタ内の複数のノードから同時にアクセスできます。

vxdg コマンドを使って、ディスクグループをクラスタ共有に設定することができます。

p.448 の「共有ディスクグループのインポート」を参照してください。

ディスクグループを 1 つのノードに対してクラスタ共有としてインポートすると、各ディスクヘッダーにはクラスタ ID が設定されます。各ノードを順次クラスタに参加していくと、ディスクグループをクラスタ共有であると認識しインポートします。それに対して、専用ディスクグループのディスクヘッダーは個々のノードのホスト名でマーク付けされます。システム管理者は、共有ディスクのインポートまたはデポートを随時実行できます。この操作は、すべてのノードに対して分散方式で実行されます。

各 LUN は一意のディスク ID でマーク付けされます。マスターノード上で VxVM のクラスタ機能を起動すると、マスターノードはすべての共有ディスクグループを（autoimport 属性セットが設定されていない場合を除き）インポートします。スレーブノードがクラスタへの参加を試みると、マスターノードはインポートされているディスクレコードの一覧をスレーブノードに送信し、スレーブノードはそのすべてにアクセスできるかをチェックします。ス

レーブノードからアクセスできないディスクがある場合には、クラスタへの参加は中断されます。一覧内のディスクすべてにアクセスできる場合には、スレーブノードはクラスタに参加し、マスターノードと同じ共有ディスクグループをインポートします。ノードがクラスタから正常に切断されると、そのノードにインポートされていた共有ディスクグループはすべてデポートされますが、それらのディスクグループは残りのノードからはデポートされません。

共有ディスクグループの再設定は、すべてのノードで協調して実行されます。ディスクグループの設定の変更は、マスターノードによって開始され、すべてのノード上で同時に実行されて、各ノード上の変更は同一になります。これらの変更には原子的な性質があります。これは、この変更がすべてのノード上で同時に発生するか、あるいはまったく発生しないかのどちらかであることを意味します。

クラスタのすべてのメンバーがクラスタ共有ディスクグループに対して同時読み書きアクセス権限を持つかどうかは、アクティブ化モード設定によって異なります。

p.174 の「共有ディスクグループのアクティブ化モード」を参照してください。

クラスタ共有ディスクグループに含まれているデータは、クラスタ内のノードが少なくとも 1 つアクティブになっていれば利用できます。あるクラスタノードの障害が、残りのアクティブノードのアクセスに影響を与えることはありません。どのノードからアクセスしても、クラスタ共有ディスクグループの設定は同一に見えます。

警告: 各ノード上で稼動しているアプリケーションは、同時に VxVM ディスク上のデータにアクセスできます。VxVM は、複数のノードによる共有ボリュームへの同時書き込みに対する保護を行いません。アプリケーション側で(たとえば Cluster File System または分散ロックマネージャを使って)整合性を管理することが前提となっています。

共有ディスクグループのアクティブ化モード

共有ディスクグループのアクティブ化は、ノード上のアプリケーションからの I/O に、このディスクグループへのアクセスを許可するため、そのノード上で実行する必要があります。アプリケーションがボリュームに対して読み取りや書き込みができるかどうかは、共有ディスクグループのアクティブ化モードで指示されます。共有ディスクグループの有効なアクティブ化モードには、`exclusivewrite`(排他書き込み)、`readonly`(読み取り専用)、`sharedread`(共有読み取り)、`sharedwrite`(共有書き込み)、および `off`(非アクティブ化)があります。

共有ディスクグループのデフォルトのアクティブ化モードは `sharedwrite`(共有書き込み)です。

HA(high availability; 高可用性)アプリケーションおよびオフホストバックアップなどのクラスタの特殊な用途では、ディスクグループのアクティベーションを使って、クラスタ内の個別のノードからのボリュームアクセスを明示的に制御できます。

表 6-1 では、アクティベーションモードを説明しています。

表 6-1　共有ディスクグループのアクティブ化モード

アクティブ化モード	説明
exclusivewrite (ew)	ノードはディスクグループに対する排他書き込みのアクセス権を持ちます。書き込みアクセスを実行するためにディスクグループをアクティブ化できる他のノードはありません。
readonly(ro)	ノードはディスクグループに対する読み取りのアクセス権を持ち、クラスタ内の他のすべてのノードの書き込みアクセスを拒否します。ノードはディスクグループに対する書き込みのアクセス権は持ちません。書き込みモードのいずれかを他のノード上で実行するためにディスクグループのアクティブ化を試みると失敗します。
sharedread(sr)	ノードはディスクグループに対する読み取りのアクセス権を持ちます。ノードはディスクグループに対する書き込みのアクセス権は持ちませんが、他のノードは書き込みアクセスを取得できます。
sharedwrite (sw)	ノードはディスクグループに対する書き込みのアクセス権を持ちます。共有読み取りおよび書き込みアクセスのためのディスクグループのアクティブ化の試行は成功します。排他書き込みおよび読み取り専用アクセスのためのディスクグループのアクティブ化の試行は失敗します。
off	ノードはディスクグループに対する読み取りと書き込みのいずれのアクセス権も持ちません。ディスクグループに対するクエリー操作は許可されます。

表 6-2 に、共有ディスクグループのアクティブ化モードの対応関係の概略を示します。

表 6-2　アクティブ化モードの対応関係

クラスタでのディスクグループのアクティブ化モード	他のノード上でディスクグループをアクティベーションする試み			
	exclusive-write	**readonly**	**sharedread**	**sharedwrite**
exclusivewrite	失敗する	失敗する	成功する	失敗する
readonly	失敗する	成功する	成功する	失敗する
sharedread	成功する	成功する	成功する	成功する
sharedwrite	失敗する	失敗する	成功する	成功する

共有ディスクグループは、ディスクグループが作成またはインポートされるときに、指定したモードで自動的にアクティブ化できます。共有ディスクグループの自動アクティブ化を制御するには、次の行を含むデフォルトファイル /etc/default/vxdg を作成します。

```
enable_activation=true
default_activation_mode=activation-mode
```

activation-mode は、排他書き込み(exclusivewrite: ew)、読み取り専用(readonly: ro)、共有読み取り(sharedread: sr)、共有書き込み(sharedwrite: sw)、または非活性(off)のいずれかです。

共有ディスクグループが作成またはインポートされると、指定したモードにアクティブにされます。ノードがクラスタに参加すると、そのノードからアクセス可能なすべての共有ディスクグループが、指定したモードでアクティブにされます。

ディスクグループのアクティブ化モードは、クラスタ内の各ノードからのボリューム I/O を制御します。指定されたノードのディスクグループが、クラスタ内の別のノード上で競合するモードでアクティブにされている場合、そのディスクグループをアクティブにできません。デフォルトファイルを使ってアクティブ化を有効にする場合は、表 6-2 に示すように、クラスタ内のすべてのノードでこのファイルの一貫性を持たせることをお勧めします。そうしないと、アクティブ化の結果が予測不能になります。

vxconfigdデーモンがすでに動作している間デフォルトファイルが編集されたら、処理を再起動するすべてのノードの/sbin/vxconfigd -k -x syslogコマンドを実行してください。

デフォルトのアクティブ化モードが off 以外のディスクグループに対して、クラスタ内の別のノードが競合するモードでアクティブにすると、クラスタへの参加、またはディスクグループの作成やインポートに続いて行われるアクティブ化の適用に失敗することがあります。

共有ディスクグループのアクティベーションモードを表示するには、vxdg list *diskgroup* コマンドを使います。

p.446 の「共有ディスクグループの一覧表示」を参照してください。

また、vxdg コマンドを使って、共有ディスクグループのアクティブ化モードを変更することもできます。

p.450 の「共有ディスクグループ上のアクティベーションモードの変更」を参照してください。

クラスタ内の単一ノードによってのみ開くことができるようにボリュームを設定することも可能です。

p.456 の「排他的起動権限を持つボリュームの作成」を参照してください。

p.456 の「ボリュームへの排他的起動権限の設定」を参照してください。

共有ディスクグループの制限

CVM 経由では RAW デバイスのみがアクセスできます。Cluster File System などの適切なソフトウェアがインストールされ設定されている場合を除き、共有ボリューム内のファイルシステムへの共有アクセスはサポートされていません。

メモ: ブートディスクグループ（通常、bootdg というエイリアスが設定される）を、クラスタ共有にすることはできません。専用ディスクグループである必要があります。

VxVM のクラスタ機能は、RAID 5 ボリュームやクラスタ共有ディスクグループをサポートしていません。ただし、これらのボリュームは、クラスタの特定のノードに接続されている専用ディスクグループに対しては使えます。または、クラスタ内の他のノードに対してフェールオーバーすることもできます。

専用ディスクグループに共有可能にする必要がある RAID 5 ボリュームがある場合、最初に、このボリュームを stripe-mirror または mirror-stripe などのサポートされているボリュームタイプに再レイアウトする必要があります。共有ボリュームのオンライン再レイアウトは、RAID 5 ボリュームが関係しない場合にかぎりサポートされます。

ストレージ接続エラーへの Cluster Volume Manager（CVM）の耐性

Cluster Volume Manager（CVM）は共有ストレージモデルを使用します。共有ディスクグループ内のボリュームには、クラスタ内の各ノードから同時に読み書きアクセスを行うことができます。

クラスタ耐性は、1 つ以上のノードが共有ストレージへの接続を失った場合の中断が最小となるクラスタ機能を意味します。CVM は、オンラインディスクグループのストレージ接続の消失を検出すると、状況に応じて適切なエラー処理を実行します。たとえば、状況によって CVM は、ネットワーク上の I/O をリダイレクトする、プレックスを切断する、またはすべてのディスクのボリュームを無効にすることがあります。CVM の動作は、環境に応じた適切な処理を確実に行うようにカスタマイズできます。

CVM 耐性機能はまた新しいノードが共有ディスクグループのすべてのディスクに接続できない場合にもノードがクラスタに結合できるようにします。この動作により、オフライン化されたノードはクラスタに再結合できます。同様に、共有ディスクグループはノードでインポートできます。

メモ: クラスタ耐性機能は一時エラーの処理を目的としています。できるだけ早く接続をリストアしてください。

CVM は接続エラーに対するクラスタ耐性と耐障害性を次の方法で強化します。

表 6-3

機能	説明	設定可能かどうか
データプレックスの一貫性。	CVM はデータディスクの接続エラーを管理し、影響を受けていないディスクで I/O が続行できるようにします。 ■ すべてのノードでエラーが発生した場合、CVM は少なくとも 1 つのプレックスがアクセス可能である限りは、影響があるプレックスを切断します。 ■ エラーがすべてのノードに影響しない場合は、ディスク切断ポリシーが CVM のエラー処理方法を決定します。 p.184 の「ディスク切断ポリシーについて」を参照してください。	はい。切断のポリシーによって制御されます（ローカルまたはグローバル）。 p.451 の「共有ディスクグループの切断ポリシーの設定」を参照してください。
アプリケーション I/O の継続	エラーがすべてのノードに影響しない場合は、CVM はストレージへのアクセスがあるノードに I/O をネットワーク経由でリダイレクトできます。この動作は、一部のノードでストレージ接続性のエラーが起きた場合でもアプリケーション I/O を一部のノードで継続するよう有効化します。 I/O のリダイレクトにより、CVM は 1 つ以上のノードに下位のストレージへのアクセスがある場合に、ボリューム I/O のローカルエラーまたはプレックスの切断の必要性を避けることができます。したがって、ioship ポリシーはディスク切断ポリシーの動作を変更します。 p.181 の「CVM I/O 転送のあるアプリケーション I/O のリダイレクトについて」を参照してください。	はい。ディスクグループに設定されている `ioship` チューニングパラメータによって制御します。 p.451 の「共有ディスクグループでの I/O 転送の有効化」を参照してください。

機能	説明	設定可能かどうか
共有ディスクグループ設定の可用性。	マスターノードは共有ディスクグループ設定の変更を処理するため、CVM はマスターノードに設定のコピーへのアクセスがあることを確認します。 マスターノードが設定のコピーへの接続性を失った場合、CVM は設定のコピーにアクセスできるノードに I/O 要求をネットワーク経由でリダイレクトします。この動作によって、ディスクグループは使用可能な状態のままになります。 この動作はディスク切断ポリシーまたは ioship ポリシーの影響を受けません。 ディスクグループバージョンが 170 より前であると、CVM はディスクグループの障害ポリシー（dgfailpolicy）に従って切断を処理します。 p.180 の「共有ディスクグループ設定のコピーの可用性」を参照してください。	いいえ。デフォルトで有効。
スナップショットの可用性	CVM は内部 I/O を開始して Data Change Object（DCO）を更新します。 ノードがこれらのオブジェクトへの接続を失った場合、CVM はアクセスがあるノードに内部 I/O ネットワーク経由でリダイレクトします。 この動作はディスク切断ポリシーまたは ioship ポリシーの影響を受けません。	いいえ。デフォルトで有効。

機能	説明	設定可能かどうか
クラスタノードと共有ディスクグループの可用性	CVM は、ノードがすべての共有ストレージにアクセスできない場合にもクラスタノードが結合できるようにします。 同様にノードは、ストレージへのローカルエラーがあっても共有ディスクグループをインポートできます。 この動作はディスク切断ポリシーまたは ioship ポリシーの影響を受けません。 p.190 の「クラスタノードと共有ディスクグループの可用性」を参照してください。	はい。 storage_connectivity チューニングパラメータで制御します。 p.455 の「ストレージ切断に対する CVM 耐障害性の制御 」を参照してください。

共有ディスクグループ設定のコピーの可用性

クラスタの少なくとも 1 つのノードが設定コピーにアクセスできれば、CVM は共有ディスクグループへの接続性を維持します。マスターノードは設定の変更を行い、その変更をすべてのスレーブノードに伝搬します。マスターノードは設定のコピーにアクセスできなくなると、アクセスできるスレーブノードに設定変更の書き込みを送信します。スレーブノードは変更を実装します。マスターノードが共有ディスクグループの設定にアクセスできなくなった場合でも、この動作によって、ディスクグループはアクティブ状態のままになります。すべてのノードが共有ディスクグループにアクセスできなくなると、ディスクグループは無効になります。

リリース 6.0 よりも前のリリースでは、共有ディスクグループのディスクグループの障害ポリシーを設定できました。最新のディスクグループバージョンとクラスタプロトコルバージョンを備えるディスクグループでは、ディスクグループ障害ポリシーはサポートされません。

クラスタプロトコルバージョンが 110 より前であるか、またはディスクグループバージョンが 170 より前である場合、ディスクグループ障害ポリシー（dgfailpolicy）はエラー発生後に動作を決定します。

VxVM の関連するリリースのマニュアルを参照してください。

ディスクグループ設定は、ioship ポリシーの設定に関係なく、この方法で処理されます。ioship ポリシーはアプリケーション I/O のリダイレクトを制御します。

p.181 の「CVM I/O 転送のあるアプリケーション I/O のリダイレクトについて」を参照してください。

CVM I/O 転送のあるアプリケーション I/O のリダイレクトについて

Cluster Volume Manager（CVM）には、ノードがディスクへの接続を失った場合に、アプリケーション I/O をネットワークを介して接続の消失を処理するオプションを提供します。接続を失ったノードは、ディスクにアクセスできるノードにネットワークを介して I/O 要求を送信します。ノードはマスターノードまたはスレーブノードのいずれかです。I/O をリダイレクトする処理も、I/O 転送と呼ばれます。アプリケーション I/O は、ノードがローカル接続を失っ場合にのみ転送されます。

I/O 転送は、デフォルトでは無効化されている ioship ポリシーによって制御されます。I/O 転送は、共有ディスクグループごとに有効化できます。

p.451 の「共有ディスクグループでの I/O 転送の有効化」を参照してください。

CVM はディスク切断ポリシーとともに I/O 転送ポリシーを I/O エラーの処理方法を決定するものとみなします。

p.188 の「CVM 切断ポリシーの I/O 転送の相互関係」を参照してください。

I/O 転送ポリシーはアプリケーション I/O のみを処理します。

CVM は設定コピーへの I/O に必ず I/O のリダイレクトを使用します。

p.180 の「共有ディスクグループ設定のコピーの可用性」を参照してください。

ストレージ切断と CVM ディスク切断ポリシー

1 つ以上のクラスタノードで 1 つ以上のプレックスへの接続エラーを検出した Cluster Volume Manager（CVM）は、以下に基づいて接続エラーの処理を決定します。

- ストレージ切断エラーの種類。
  p.181 の「ストレージ接続エラーの種類について」を参照してください。
- ディスクグループの切断ポリシーの設定。
  p.184 の「ディスク切断ポリシーについて」を参照してください。

ストレージ接続エラーの種類について

CVM はエラーの範囲に基づいてストレージ接続エラーの種類を判断します。CVM はエラーがすべてのノードに影響する（グローバルエラー）か、特定のノードに影響する（ローカルエラー）かを判断します。CVM はエラーがボリュームの 1 つ以上のプレックスに影響するかどうかも判断します。すべてのプレックスに影響するエラーは、それは全体的なエラーであるとみなされます。それ以外のエラーは、部分的なエラーであるとみなされます。

CVM は次のストレージ切断の種類を定義します。

- グローバルで部分的なエラー
  図 6-2 にこのシナリオを示します。
- グローバルで全体的なエラー

図 6-3 にグローバルで全体的なエラーを示します。

- ローカルで部分的なエラー
  図 6-4 にローカルで部分的なエラーを示します。
- ローカルで全体的なエラー
  図 6-5 にローカルで全体的なエラーを示します。

図 6-2 にグローバルで部分的なエラーを示します。グローバルで部分的なエラーとは、すべてのノードが影響を受けますが、ボリュームのプレックスの一部は影響を受けないエラーです。例では、クラスタ内のすべてのノードが、ボリュームのプレックス B を持つアレイ B へのアクセスを失っています。

図 6-2　　グローバルで部分的なエラー

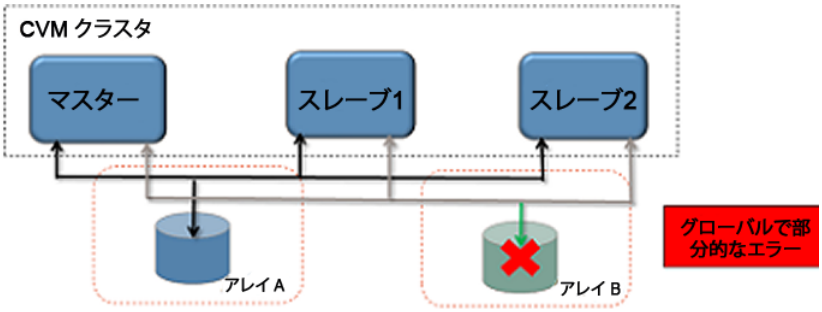


図 6-3 にグローバルで全体的なエラーを示します。グローバルで全体的なエラーとはすべてのノードが影響を受け、ボリュームのすべてのプレックスが影響を受けるエラーです。

図 6-3　　グローバルで全体的なエラー

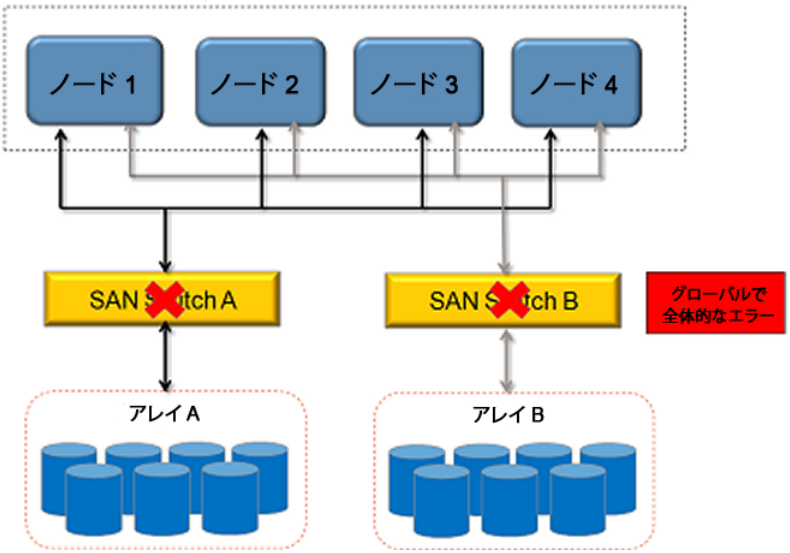


図 6-4 にローカルで部分的なエラーを示します。ローカルで部分的なエラーとは、一部のプレックスには影響せず、1 つ以上のノードで発生しますが、すべてのノードでは発生しないエラーです。

図 6-4　　ローカルで部分的なエラー

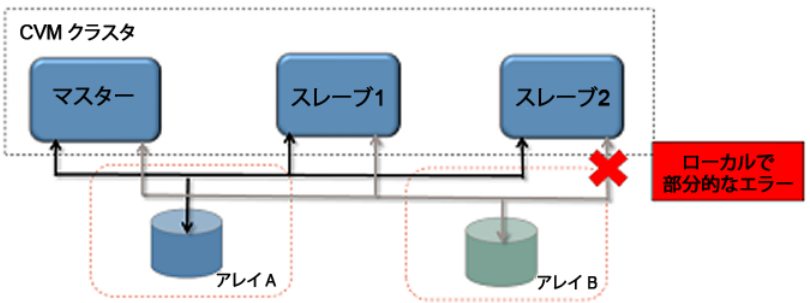
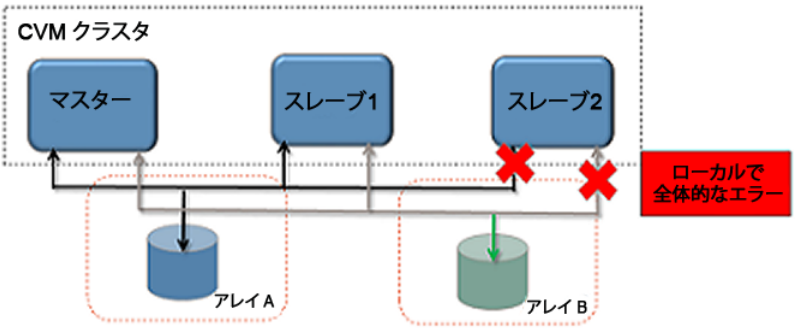


図 6-5 にローカルで全体的なエラーを示します。ローカルで全体的なエラーとは、すべてのプレックスに影響し、1 つ以上のノードで発生しますが、すべてのノードでは発生しないエラーです。

図 6-5　ローカルで全体的なエラー



ディスク切断ポリシーについて

ディスク切断ポリシーでは、CVM がストレージのエラーまたはストレージへの接続エラーを処理する方法を決定します。

サイズの小さいミラーボリューム、ミラー化されていないボリューム、ハードウェアミラーを使うボリュームまたは専用ディスクグループ内のボリュームの場合は、ローカル切断ポリシー（local detach policy）を設定してもメリットはありません。通常は、デフォルトのグローバル切断ポリシー（global detach policy）を使うことをお勧めします。

次のディスク切断ポリシーが使用可能です。

- グローバル切断ポリシー
  グローバル切断ポリシーでは、任意の I/O エラーで、エラーを参照するプレックスがボリュームのクラスタ全体で切断されるように指定します。この動作は、クラスタのすべてのノード上のボリュームへの対称型アクセスを保証します。グローバル切断ポリシー（global detach policy）は従来のポリシーであり、クラスタを設定するすべてのノードにデフォルトで適用されます。
  p.185 の「グローバル切断ポリシーによって CVM がローカルストレージ切断を処理する方法」を参照してください。
- ローカル切断ポリシー
  ローカル切断ポリシーは、ローカル接続の問題のためにノードにプレックスへの I/O エラーがある場合、ボリュームがローカルで無効になることを示します。プレックスは、クラスタ全体では切断されません。この動作は、すべてのプレックスが他のノードの I/O で使用可能であることを確認します。エラーがあったノードのみが影響を受けます。
  p.186 の「ローカル切断ポリシーによって CVM がローカルストレージ切断を処理する方法」を参照してください。

ディスク切断ポリシーは共有ディスクグループのために設定されます。切断ポリシーをデフォルトのグローバル切断ポリシーからローカルの切断ポリシーに変更する必要がある場合は、vxdg コマンドを使用します。

グローバル切断ポリシーによって CVM がローカルストレージ切断を処理する方法

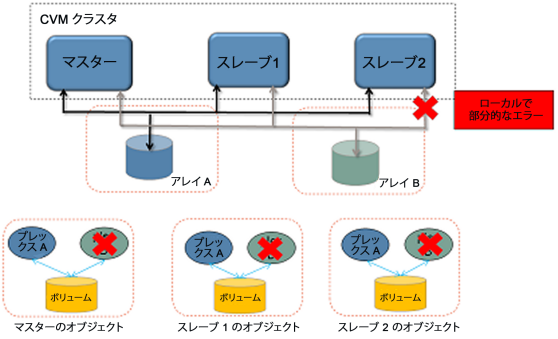
ローカルエラーに対する CVM 動作は、切断ポリシーの設定と影響されるプレックスの数によって決まります。

エラーがすべてのノードに影響しない場合、エラーはローカルであるとみなされます。ローカルエラーは 1 つ以上のノードで発生します（すべてのノードではありません）。I/O エラーは、少なくとも 1 つのノードがプレックスにアクセスできる場合は、ローカルであると見なされます。

切断ポリシーがグローバルに設定され、エラーが 1 つ以上のノードのボリュームの 1 つ以上のプレックスに影響する場合、CVM はプレックスを切断します。グローバル切断ポリシーは、CVM がボリュームのプレックス（ミラー）の一貫性を確保することを示します。プレックスの切断により、プレックスのデータはすべてのノードでまったく同じになります。接続が戻ると、CVM はボリュームにプレックスを再接続し、プレックスを再同期します。

図 6-6 に、切断のポリシーがグローバルの場合に CVM がローカルで部分的なエラーを処理する方法を示します。

図 6-6　グローバル切断ポリシーで CVM がローカルで部分的なエラーを処理する方法



このポリシーの利点は、ボリュームがすべてのノードの I/O でまだ使用できるという点です。スレーブノードに読み込みまたは書き込みの I/O エラーがある場合、マスターノードはエラーを修復するために、通常の I/O リカバリ操作を実行します。必要に応じて、プレックスはクラスタ全体のボリュームから切断されます。ノードはすべてクラスタ内にとどまり、I/O は引き続き実行されますが、ミラーの冗長性は低くなります。

欠点は、プレックスが切断されたために冗長性が失われた点です。クラスタの 1つ以上のノードがプレックスへの接続性を失うため、クラスタ全体はそのプレックスにアクセスで

きなくなります。この動作は、1 つのノードのローカルエラーが、クラスタのすべてのノードにグローバルな影響を与えることを意味します。

グローバル切断ポリシーでは、プレックスを再接続するオーバーヘッドも必要です。I/O エラーの原因となる問題が修正されたら、ディスクを再接続する必要があります。切断されたミラーは、データの冗長性がリカバリされる前に、リカバリする必要があります。

ミラーボリュームのすべてのプレックスでノードがエラーを検出すると、ローカルノードからボリュームへの I/O は失敗しますが、プレックスは切断されません。この動作により、各プレックスが次々と切断されたり、ボリュームがグローバルに無効になるなどの動作が回避されます。

ローカル切断ポリシーによって CVM がローカルストレージ切断を処理する方法

ローカル切断ポリシーは、ローカル接続の問題のためにノードにプレックスへの I/O エラーがある場合、I/O がローカルボリュームで失敗することを示します。プレックスは、クラスタ全体では切断されません。この動作は、すべてのプレックスが他のノードの I/O で使用可能であることを確認します。エラーがあったノードのみが影響を受けます。

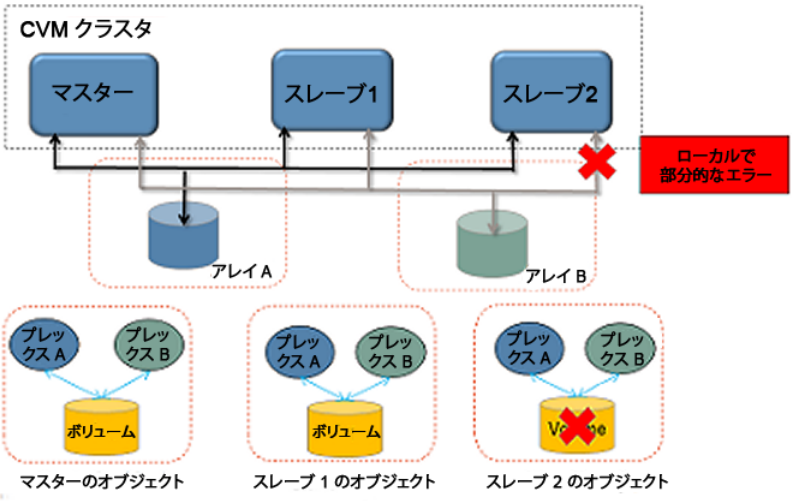
このポリシーの利点は、ボリュームの冗長性ができるかぎり保護される点です。ローカル切断ポリシー（local detach policy）は、ボリュームにアクセスできるノードの数よりもボリュームの冗長性が重要となる大規模なクラスタのフェールオーバーアプリケーションをサポートします。つまり、クラスタ内のすべてのボリュームのプレックスが失われるよりも、1 つ以上のノードがボリュームへの I/O アクセスを失うことを選択します。このシナリオは通常、ボリュームがミラー化される場合、および同じサービスを他のノードからシームレスに提供できる並列アプリケーションが使われている場合に適用されます。たとえば、このオプションは高速フェールオーバーの構成には適していません。

切断ポリシーがローカルに設定され、エラーがローカルで部分的なエラーである場合、CVM はボリュームへの書き込み I/O をローカルに失敗します。ローカル切断ポリシーは、CVM によってローカルの切断エラーの影響がローカルノードのみにとどまることを示します。ボリュームへの I/O がローカルに失敗している場合、アプリケーションは別のノードにフェールオーバーされる必要があります。

I/O 転送ポリシーが有効になっている場合、I/O はネットワークを介してクラスタの別のノードにリダイレクトされます。この場合、CVM は I/O を失敗しません。

図 6-7 にローカルで部分的なエラーを示します。

図 6-7 ローカルで部分的なエラー - ローカル切断ポリシー



切断ポリシーの選択に関するガイドライン

通常は、グローバル切断ポリシー（global detach policy）を使うことをお勧めします。これが特に重要になるのは、次の条件に該当する場合です。

- ミラー化されていないボリューム、サイズの小さいボリューム、ミラーボリュームまたはハードウェアミラーボリュームが設定されている場合。ローカル切断ポリシー（local detach policy）を適用した場合に別途必要になるメッセージ処理の分だけシステムオーバーヘッドを減らすことができます。

ローカル切断ポリシー（local detach policy）が適するケースとしては、次のような場合が考えられます。

- サイズの大きいミラーボリュームが設定されている場合。再接続されたプレックスの再同期により、システムの処理効率が低下することがあります。ローカル切断ポリシー（local detach policy）を使った場合には、プレックスを切断する必要が生じることはありません DRL（dirty region logging）機能も再同期に必要な処理量を減らすことができます。
- ノード数が 4 を超えるクラスタ。クラスタ内のノード数が多い場合は、特定のノード上でアプリケーションの稼動を維持することの重要性が比較的低くなります。クラスタ管理ソフトウェアの設定により、ボリュームへのアクセスが可能なノードにアプリケーションを移動できる可能性があります。また、負荷分散により、I/O の問題が発生したボリューム以外のボリュームにアプリケーションを移動できる可能性もあります。この動作はデータ冗長性を保持し、該当するディスク上のボリュームに対して他のノードから引き続き I/O を実行することができます。

表 6-4 では、異なる切断ポリシー下で I/O エラーが生じた場合の CVM の動作を比較します。

表 6-4　ミラーボリュームの I/O エラー発生時のディスク切断ポリシーのクラスタの動作

エラーのタイプ	ローカル切断ポリシー	グローバル切断ポリシー
ローカルで部分的なエラー	プレックスにアクセスできないノードからボリュームへの I/O が失敗します。	プレックスを切断します。
ローカルで全体的なエラー	プレックスにアクセスできないノードからボリュームへの I/O が失敗します。	ボリュームへの I/O が失敗します。
グローバルで部分的なエラー	プレックスを切断します。	プレックスを切断します。
グローバルで全体的なエラー	ボリュームを無効にします。	ボリュームを無効にします。

CVM 切断ポリシーの I/O 転送の相互関係

I/O 転送が有効な場合、CVM は I/O のリダイレクトを試行してからローカルにボリュームを無効化するかプレックスを切断します。したがって、I/O 転送が有効な場合には切断ポリシーの動作は異なります。

表 6-5 では、I/O 転送が有効な場合のミラーボリュームのディスクに対するクラスタの I/O エラーへの影響の概略を示します。

表 6-5　I/O 転送が有効な場合のミラーボリュームの I/O エラー発生時のディスク切断ポリシーのクラスタの動作

エラーのタイプ	ローカル切断ポリシー	グローバル切断ポリシー
ローカルで部分的なエラー	I/O を転送します。	プレックスを切断します。
ローカルで全体的なエラー	I/O を転送します。	I/O を転送します。
グローバルで部分的なエラー	プレックスを切断します。	プレックスを切断します。
グローバルで全体的なエラー	最後のプレックス以外のすべてのプレックスを切断します。	ボリュームを無効にします。

ポリシーの影響を受けない CVM のストレージ切断シナリオ

次のストレージ接続エラーの動作は切断ポリシーに関係なく同じになります。

- グローバルで部分的なエラー
  図 6-8 にこのシナリオを示します。
- グローバルで全体的なエラー。
  図 6-9 にグローバルで全体的なエラーを示します
- ローカルで全体的なエラー。
  図 6-10 にグローバルで全体的なエラーを示します

図 6-8 にグローバルで部分的なエラーを示します。グローバルで部分的なエラーとは、すべてのノードが影響を受けますが、ボリュームのプレックスの一部は影響を受けないエラーです。例では、クラスタ内のすべてのノードが、ボリュームのプレックス B を持つアレイ B へのアクセスを失っています。

プレックス B は切断されます。プレックスにアクセスできるノードがないため、ミラーの一貫性を維持するためにプレックスは削除される必要があります。ボリューム内の他のプレックスへの I/O は続行します。これは、ボリュームの冗長性を低減します。

図 6-8　　ポリシーの影響を受けないグローバルで部分的なエラー

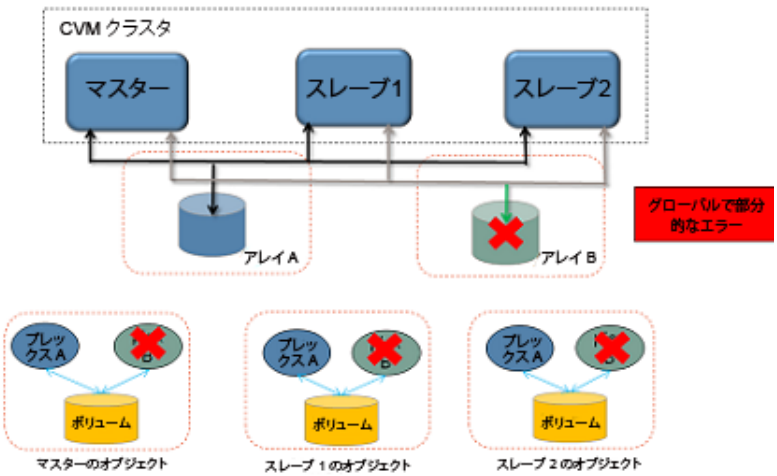
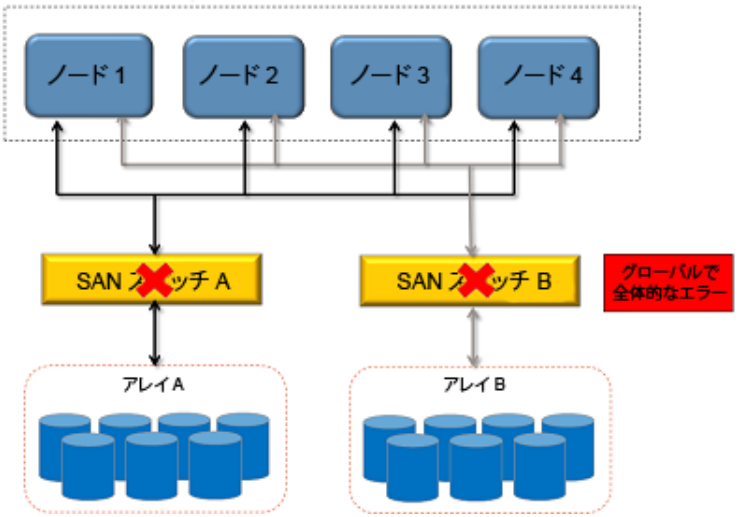


図 6-9 にグローバルで全体的なエラーを示します。これはすべてのノードが影響を受け、ボリュームのすべてのプレックスが影響を受けることを意味します。ボリュームは無効になります。使用できるプレックスがないため、ボリュームはいずれの I/O でも使用できません。すべてのノードでエラーが同時に発生した場合、プレックスは切断されません。

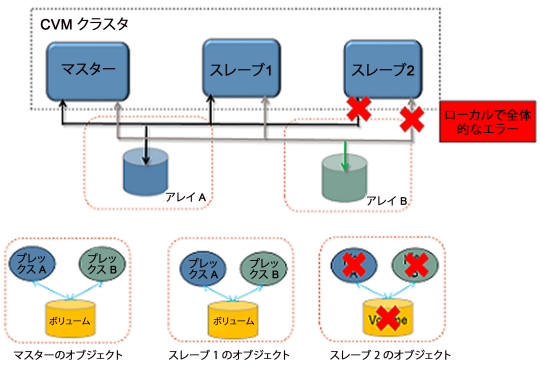
図 6-9 ポリシーの影響を受けないグローバルで全体的なエラー



ローカルエラーが全体的でボリュームのすべてのプレックスに影響する場合、CVM の動作はポリシーに関係なく同じになります。CVM はボリュームへの I/O をローカルに失敗します。プレックスにアクセスできないノードからボリュームへの I/O は失敗します。他のノードからボリュームへの I/O は続行します。

図 6-10 にローカルで全体的なエラーを示します。

図 6-10 ポリシーの影響を受けないローカルで全体的なエラー



クラスタノードと共有ディスクグループの可用性

デフォルトで、CVM では、クラスタ内の 1 つ以上のノードを介してディスクにアクセスできるノードはクラスタと結合できます。通常、ノードはマスターノードによってディスクにアクセ

スします。共有ディスクグループは、すべてのディスクにアクセスできないノードがある場合にもインポートできます。ノード結合操作やディスクグループのインポート操作の実行中、ノードがボリュームの下位のストレージにアクセスできないことを CVM が検出すると、ボリュームは LDISABLED 状態になります。この状況ではストレージの接続性は非対称になります。これは、共有ディスクグループですべてのディスクへの同じアクセスを持っていないノードがあることを意味しています。

メモ: 非対称的なディスクアクセスのサポートは、一時的な接続の問題を処理し、永続的な状態にならないようにすることを目的としています。接続はできるだけ早く復元する必要があります。CVM は、接続が復元されるまでクラスタを縮退モードにあるとみなします。

この動作はデフォルトで無効になっています。ストレージ接続性ポリシーを耐性に設定して、この機能を無効にできます。この機能の設定は、切断のポリシーと I/O 転送のポリシーの制約を受けません。ただし、ディスクグループのバージョンとクラスタプロトコルのバージョンは動作をサポートするレベルである必要があります。

ストレージ接続を耐性に設定すると、CVM（Cluster Volume Manager）はクラスタに結合するノードが共有ディスクグループ内のすべてのディスクにアクセスできることを要求します。この必要条件は I/O のエラーのためにクラスタから削除されたノードにも適用されます。ノードはディスクアクセスが復元されるまでクラスタに再結合できません。同様に、共有ディスクグループはクラスタのすべてのノードがディスクグループのすべてのディスクにアクセスないかぎりはインポートできません。

p.455 の「ストレージ切断に対する CVM 耐障害性の制御 」を参照してください。

CVM の初期化と設定

新しいクラスタにノードを結合するには、クラスタモニタの設定時に適切な設定情報を指定しておく必要があります。この情報は、通常、ある形式でクラスタモニタ設定データベースに保存されます。この情報の正確な内容および形式は、クラスタモニタの特性により異なります。VxVM に必要となる情報は、次のとおりです。

- クラスタ ID
- ノード ID
- ノードのネットワークアドレス
- ポート番号

ノードをクラスタに結合すると、ノードの起動時に、この情報が自動的にそのノード上の VxVM にロードされます。

メモ: VxVM の CVM 機能は、VxVM との連携を意図して正しく設定されたクラスタモニタと共に使う場合にかぎりサポートされます。

Cluster Service（VCS）内の GAB（Group Membership and Atomic Broadcast）などのクラスタモニタを使います。VCS 環境については、`vxcvmconfig` コマンドを任意のノード上で使って、VxVM の CVM 機能を使うよう設定します。`vxcvmconfig` コマンドは、Veritas Volume Manager には装備されていません。

クラスタモニタの起動手順はノードの初期化を行い、ノード上の各種のクラスタコンポーネント（クラスタをサポートする VxVM、クラスタモニタおよび分散ロックマネージャなど）を起動します。クラスタモニタの起動手順が完了すると、アプリケーションを起動できるようになります。クラスタモニタの起動手順は、クラスタに結合する各ノードで呼び出す必要があります。

クラスタ環境では、VxVM の初期化時にクラスタ設定情報がロードされ、クラスタにノードが結合されます。最初に結合されたノードがマスターノードになり、後のノードは（スレーブとして）マスターノードに結合されます。2 つのノードが同時に結合された場合は、VxVM がマスターノードを決定します。結合後のノードには、共有ディスクグループとボリュームへのアクセス権が与えられます。

クラスタの再設定

ノードの切断や結合が発生すると、クラスタの再設定が実行されます。各ノードのクラスタモニタは、継続的に、他のクラスタノードを監視します。クラスタのメンバーシップに変更があると、クラスタモニタがVeritas Volume Manager（VxVM）に通知して、変更に対する適切な処理が実行されます。

クラスタの再設定の間、VxVM は共有ディスクへの I/O を中断します。再構成が完了すると、I/O が再開されます。再構成中、アプリケーションが短時間停止しているように見えることがあります。

VxVM 操作またはリカバリなどの他の操作が進行中の場合は、それらの操作が完了するまで、クラスタの再設定が遅延することがあります。ボリュームの再設定は、クラスタの再設定と同時に実行されることはありません。環境によっては、一方の操作が保留され、後で再開されることがあります。多くの場合、クラスタの再設定が優先されます。ただし、ボリュームの再設定が（2PC の）コミットフェーズにある場合は、ボリュームの再設定が先に完了します。

p.195 の「ボリュームの再設定」を参照してください。

p.192 の「vxclustadm ユーティリティ」を参照してください。

vxclustadm ユーティリティ

`vxclustadm`コマンドはCVM機能にVCSがと同時にクラスタ監視使われるときインターフェースをのVxVM提供したものです。また、クラスタの起動時および停止時に呼び出さ

れます。クラスタモニタが存在しない場合、vxclustadm はクラスタ内の任意のノードで VxVM の CVM 機能を起動または停止する役割も担います。

vxclustadm の startnode キーワードは、クラスタ設定情報を VxVM カーネルに渡すことによって、クラスタノード上の CVM 機能を起動します。このコマンドを受信すると、カーネルおよび VxVM 設定デーモン vxconfigd は初期化を実行します。

stopnode キーワードを指定すると、ノード上の CVM 機能が停止します。未処理の I/O がすべて完了し、すべてのアプリケーションが共有ボリュームの使用を停止するまで待機します。

setmaster キーワードは、指定したノードに CVM マスターを移行します。移行はオンライン操作です。ベリタスでは、クラスタが VxVM 設定の変更またはクラスタの再設定操作を処理していないときマスターを切り替えることをお勧めします。

reinit キーワードを指定すると、クラスタを停止することなく、クラスタへのノードの追加や、クラスタからのノードの削除を行うことができます。このコマンドを実行する前に、クラスタ内でサポートされているノードに関する最新情報でクラスタ設定ファイルを更新しておく必要があります。

nidmap キーワードを指定すると、VxVM のクラスタサポートサブシステムの CVM ノード ID とクラスタモニタのノード ID の対応関係を示す表が出力されます。また、クラスタ内のノードの状態も出力されます。

nodestate キーワードを指定すると、次の例に示すように、クラスタノードの状態と最後の中止の原因が表示されます。

```
# vxclustadm nodestate

state: out of cluster
reason: user initiated stop
```

表 6-6 に、考えられるノード中止の理由を一覧表示します。

表 6-6　　ノードの中止メッセージ

原因	説明
スレーブノード上にディスクがありません（cannot find disk on slave node）	スレーブノード上にディスクがないか、ディスクが不良です。
設定データを取得できません（cannot obtain configuration data）	ディスク障害などのエラーのため、ノードが設定データを読み取れません。
クラスタデバイスの起動に失敗しました（cluster device open failed）	クラスタデバイスの起動に失敗しました。

原因	説明
ライセンスがマスターノードのライセンスと一致しません(clustering license mismatch with master node)	クラスタライセンスがマスターノードのライセンスと一致しません。
クラスタライセンスがありません(clustering license not available)	クラスタライセンスが見つかりませんでした。
接続がマスターによって拒否されました(connection refused by master)	ノードの結合がマスターノードによって拒否されました。
ディスクは別のクラスタが使用中です(disk in use by another cluster)	ディスクは、ノードを結合しようとしているクラスタ以外のクラスタに属しています。
再設定中のため結合がタイムアウトしました(join timed out during reconfiguration)	クラスタ内で再構成が実行されているため、ノードの結合がタイムアウトしました。
カーネルログの更新に失敗しました(klog update failed)	ノードの結合中にカーネルログのコピーを更新することはできません。
結合中にマスターが中止されました(master aborted during join)	ノードをクラスタに結合しているときにマスターノードが中止されました。
プロトコルのバージョンが不正です(protocol version out of range)	クラスタプロトコルのバージョンが一致しない、または未サポートのバージョンです。
修復しています(recovery in progress)	ノードによって起動されたボリュームの修復が完了していません。
役割の移行に失敗しました(transition to role failed)	ノードのマスターへの昇格に失敗しました。
ユーザーによる中止(user initiated abort)	ユーザーまたはクラスタモニタによって中止されたため、ノードがクラスタから切断されました。
ユーザーによる停止(user initiated stop)	ユーザーまたはクラスタモニタによって停止されたため、ノードがクラスタから切断されました。
vxconfigd **が有効になっていません**(vxconfigd is not enabled)	VxVM 設定デーモンが有効になっていません。

vxclustadm(1M)マニュアルページを参照してください。

ボリュームの再設定

ボリュームの再設定は、ディスクグループ、ボリューム、プレックスなどの VxVM オブジェクトを作成、変更または削除する処理です。クラスタ内では、すべてのノードが協調してこれらの操作を実行します。vxconfigd デーモンは、ボリュームの再設定において積極的な役割を果たします。再設定を成功させるには、vxconfigd デーモンが各ノード上で動作している必要があります。

p.196 の「vxconfigd デーモン」を参照してください。

ボリュームの再設定のトランザクションは、マスターノード上で VxVM ユーティリティを実行することによって開始されます。このユーティリティは、マスターノード上のローカルな vxconfigd デーモンにコンタクトします。このデーモンは、要求された変更を検証します。たとえば、既存のディスクグループと同じ名前で新しいディスクグループを作成しようとすると、vxconfigd デーモンはエラーを返します。この後、マスターノード上の vxconfigd デーモンは、スレーブノード上の vxconfigd デーモンに変更の詳細を送信します。スレーブノード上の vxconfigd デーモンは、続いて独自に検証を実行します。たとえば、各スレーブノードに作成されているものと同じ名前の専有ディスクが存在しないか検証します。新しいディスクが含まれる場合は、各ノードが新しいディスクにアクセスできることを検証します。すべてのノード上の vxconfigd デーモンが提案された変更が妥当であると合意すると、各ノードはそれぞれのカーネルに通知します。これに続いて、カーネルは、トランザクションをコミットするか破棄するかを協調して決定します。トランザクションをコミットする前に、すべてのカーネルに進行中の I/O がないことを確認し、再設定が完了するまでアプリケーションが発行した I/O をブロックします。マスターノードは、ボリュームの再設定、トランザクションの開始およびコミットの両方に対する調整役を担います。その結果、設定の変更は、すべてのノードで同時に発生したように見えます。

再設定トランザクション中に、いずれかのノード上の vxconfigd デーモンが終了すると、すべてのノードに通知され、そのトランザクションは破棄されます。いずれかのノードがクラスタから切断されると、マスターノードがすでにそれをコミットしている場合を除き、そのトランザクションは破棄されます。マスターノードがクラスタから切断されると、以前はスレーブノードであった新しいマスターノードが、以前のマスターノードからトランザクションのコミットの通知を受け取ったどうかによって、そのトランザクションをコミットまたは破棄させます。この通知は、新しいマスターノードが受け取っていない場合、他のすべてのスレーブノードも受け取っていないものとして動作します。

ボリュームの再設定トランザクション実行中に、ノードがクラスタへの結合を試みた場合の再設定の結果は、トランザクションのフェーズにより異なります。カーネルがまだ呼び出されていないフェーズでは、ノードがクラスタに結合するまで、ボリュームの再設定が中断されます。カーネルが呼び出されているフェーズでは、再設定が完了するまで、ノードはクラスタへの結合を待機します。

スレーブノード上での検証に失敗したり、ノードがクラスタから切断されたりといったエラーがと、ユーティリティにエラーが返され、マスターノード上のコンソールにエラーが起きたノードを識別するメッセージが送信されます。

vxconfigd デーモン

VxVM 設定デーモンの vxconfigd は、VxVM オブジェクトの設定を保持します。このデーモンは、クラスタを制御する指示をカーネルから受け取ります。vxconfigd デーモンのプロセスが各ノード上で実行されます。このデーモンプロセスはネットワークを介して相互に通信します。VxVM 各種ユーティリティは、起動されると同じノード上で稼動している vxconfigd デーモンと通信しますが、他のノード上の vxconfigd デーモンとは通信しません。クラスタの起動時、カーネルは vxconfigd デーモンにクラスタの制御操作を始めるように促し、それがマスターノードか、スレーブノードかを示します。

クラスタ制御操作のためにノードが初期化されると、vxconfigd デーモンにノードがクラスタに結合するところであることが通知され、クラスタモニタの設定データベースから次の情報が提供されます。

- クラスタ ID
- ノード ID
- マスターノード ID
- ノードの役割
- ノードのネットワークアドレス

マスターノードでは、vxconfigd デーモンが共有ディスクグループをインポートすることによって共有設定を構成し、スレーブノードがクラスタに結合する準備が完了すると、カーネルに通知します。

スレーブノードでは、クラスタに結合できるようになったときに、vxconfigd デーモンがこの通知を受けます。スレーブノードがクラスタに結合されると、共有設定を構成するために(スレーブノードの)vxconfigd デーモンおよび VxVM カーネルがマスターノード上の vxconfigd デーモンおよび VxVM カーネルと通信します。

あるノードがクラスタから切断されると、カーネルは他のすべてのノード上の vxconfigd デーモンに通知します。マスターノードは、続いて、必要なクリーンアップを実行します。マスターノードがクラスタから切断されると、カーネルは新しいマスターノードを選択し、すべてのノード上の vxconfigd デーモンにメンバーシップの決定を通知します。

また、vxconfigd デーモンは、ボリュームの再設定にも関与します。

p.195 の「ボリュームの再設定」を参照してください。

vxconfigd デーモンのリカバリ

クラスタでは、スレーブノード上の vxconfigd デーモンが、常に、マスターノード上の vxconfigd デーモンに接続されます。vxconfigd デーモンが停止していると、ボリュームの再設定トランザクションは実行できません。スレーブノードで vxconfigd デーモンが稼動していない場合でも、他のノードがクラスタに結合することは可能です。

vxconfigd デーモンが停止した場合に実行される処理は、どのノードのデーモンが停止したかによって異なります。

- vxconfigd デーモンがマスターノード上で停止された場合は、スレーブノード上の vxconfigd デーモンがマスターノードへの再結合を定期的に試みます。マスターノード上で vxconfigd デーモンが再起動するまで、この試みは成功しません。この場合には、スレーブノード上の vxconfigd デーモンは共有設定に関する設定情報を失うわけではないため、設定情報がすべて正しく表示されます。
- スレーブノード上の vxconfigd デーモンが停止されても、マスターノードは何の動作も行いません。vxconfigd デーモンがスレーブ上で再起動されると、スレーブの vxconfigd デーモンはマスターデーモンに再接続して共有設定に関する設定情報を再取得しようと試みます（カーネルが使っている共有設定も、共有ディスクへのアクセスも、影響を受けません）。スレーブノード上の vxconfigd デーモンがマスターノード上の vxconfigd デーモンに正常に再接続されるまでは、スレーブノード上のデーモンは共有設定に関する設定情報をほとんど持たないため、共有設定の表示や修正の試行がすべて失敗することがあります。たとえば、vxdg list コマンドを使って一覧表示された共有ディスクグループは disabled と設定され、再結合が正常に終了すると enabled と設定されます。
- vxconfigd デーモンがマスターノード上とスレーブノード上の両方で停止された場合、マスターノード上およびスレーブノード上で vxconfigd が再起動され再接続されるまで、スレーブノードは正確な設定情報を表示しません。

VCS（Cluster Server）の CVM（Cluster Volume Manager）エージェントが、クラスタの再構成時にノード上で vxconfigd デーモンが実行していないことを確認すると、vxconfigd デーモンは自動的に再起動されます。

警告: vxconfigd の -r リセットオプションは、vxconfigd デーモンを再起動し、すべての状態を一から再作成します。このときクラスタの共有設定に関する設定情報が破棄されるため、ノードがクラスタに結合してしている間は、このオプションを使って vxconfigd を再起動することはできません。

場合によっては、Veritas Volume Manager（VxVM）の問題を解決するために、VCS が制御するクラスタで vxconfigd を手動で再起動する必要があります。

ノードの停止

ノードのクラスタモニタの停止手順を呼び出すことによってノード上のクラスタを停止することは可能ですが、この手順は、共有ストレージにアクセスするノード上のアプリケーションをすべて停止した後にクラスタコンポーネントを終了させるためのものです。VxVM ではノードのクリーンシャットダウンがサポートされており、共有ボリュームへのアクセスがすべて停止したときに、ノードをクラスタから正常に切断できます。切断後、ホストは操作可能ですが、クラスタアプリケーションをホスト上で実行することはできません。

VxVM の CVM 機能は、各ボリュームの大域状態の設定情報を保持します。これにより、VxVM で、ノードがクラッシュしたとき、どのボリュームを修復する必要があるかを確認できます。クラッシュまたは正常ではない他の何らかの方法によってノードがクラスタから切断されると、VxVM は書き込みが完了していない可能性があるボリュームを確認し、マスターノードがこれらのボリュームを再同期します。このとき、いずれかのボリュームに対して DRL（dirty region logging）または FastResync が有効であるならば、再同期で使うことができます。

ノードのクリーンシャットダウンは、すべてのクラスタアプリケーションを停止する手順の実行後または実行時に行う必要があります。クラスタ化されたアプリケーションの特性およびその停止手順によっては、正常な停止に多くの時間（数分から数時間）を要することがあります。たとえば、多くのアプリケーションは、ドレイニングの概念を持っており、新しい作業を受け付けず、実行中の作業がすべて完了してから終了します。実行時間の長いトランザクションが実行中の場合は、この処理に時間がかかることがあります。

VxVM の停止手順が呼び出されると、停止するノード上にある共有ディスクグループ内のボリュームがすべて検証されます。この後、手順は、停止処理を続行するか、次のいずれかの理由により失敗します。

- 共有ディスクグループ内のすべてのボリュームが停止していると、VxVM がそれらをアプリケーションに対して使用不能にします。切断するノード上でこれらのボリュームが停止しているという情報はすべてのノードに通知されるため、再同期は実行されません。
- 共有ディスクグループ内に起動しているボリュームがあると、停止手順は失敗します。停止手順は、成功するまで繰り返し再試行される可能性があります。この操作ではタイムアウトはチェックされません。タイムアウトチェックは、クラスタ化されたアプリケーションが有効な状態にないことを確認する目的で用意されている機能です。

停止処理が正常に終了すると、ノードはクラスタから切断されます。ノードがクラスタに再び結合されるまで、共有ボリュームにアクセスすることはできません。

停止処理には長い時間がかかることがあるため、停止処理の進行中に他の再構成が実行される場合があります。通常、他の再構成が完了するまで、停止処理は中断されます。ただし、停止処理がすでに一定以上進んでいる場合は、停止処理が先に完了します。

クラスタの停止

すべてのノードをクラスタから切断する際に最後のノードが正常に切断されなかった場合や、正常に切断されなかった以前のノードからの再同期化が完了していない場合には、クラスタの次回の起動時に共有ボリュームを修復する必要があります。ノードがクラスタに参加するとき、CVM が自動的にリカバリと再同期のタスクを処理します。

クラスタ環境での DRL

DRL（Dirty Region Log）は、システムのクラッシュ後、ボリュームを迅速に修復するために使うボリュームのオプションの属性です。DRL は、クラスタ共有ディスクグループでサポートされています。この項では、クラスタ環境での DRL の動作を簡単に説明します。

クラスタ環境では、DRL の VxVM 実装は通常の実装とはわずかに異なります。

クラスタをサポートしていないシステム上の DRL には、リカバリマップとアクティブマップが 1 つずつ記録されます。ただし、CVM DRL には、クラスタごとに 1 つのリカバリマップと、クラスタノードごとに 1 つのアクティブマップが記録されます。

クラスタ環境下における DRL のサイズは、リカバリマップに加えクラスタ内の各ノードのアクティブマップを収容する必要があるため、通常、クラスタ化されていないシステム内のものより大きくなります。DRL 内の各マップの大きさは、ブロック 1 つ分または複数分です。`vxassist` コマンドは、ボリュームサイズとノード数に合わせて自動的に十分な大きさの DRL を割り当てます。

クラスタ環境における共有ディスクグループは、専用ディスクグループ（および、そのボリューム）と同様に再インポートすることができます。ただし、そのインポートされたディスクグループの DRL は無効状態と見なされ、結果として、完全なリカバリが実行されます。

クラスタをサポートしていないシステムの専用ディスクグループとして共有ディスクグループがインポートされる場合、VxVM は、共有ボリュームのログを無効状態であると見なし、ボリューム全体のリカバリを実行します。リカバリが完了すると、VxVM は DRL を使います。

VxVM のクラスタ機能では、共有されていないボリューム上で DRL リカバリを実行することができます。ただし、このようなボリュームをクラスタをサポートしている VxVM システムに移動し、共有としてインポートすると、DRL が小さすぎてすべてのクラスタノードのマップが収容できない可能性があります。このため、VxVM は、このログを無効に設定し、いずれにしても、ボリューム全体に対するリカバリを実行します。同様に、2 ノードクラスタから 4 ノードクラスタへの DRL ボリュームの移動を行うと、ログサイズが小さすぎるという結果になり、VxVM のクラスタ機能がボリューム全体のリカバリを行うことになります。どちらの場合にも、新しいログに十分な大きさを割り当てる必要があります。

p.86 の「DRL」を参照してください。

クラスタ環境での DRL の動作方法

クラスタ内の 1 つ以上のノードがクラッシュすると、DRL はクラッシュが発生した際にそれらのノードが使っていたすべてのボリュームのリカバリを実行します。最初のクラスタの起動時のボリュームの起動操作の間に、アクティブマップの内容すべてがリカバリマップに統合されます。

クラッシュしたノード（すなわち、ダーティとしてクラスタから切断されたノード）は、DRL アクティブマップが影響を受けたすべてのボリュームのリカバリマップに統合されるまで、クラスタへ再結合することができません。リカバリユーティリティは、クラッシュしたノードのア

クティブマップをリカバリマップと比較して、必要な更新をすべて行います。このノードはその後でのみ、クラスタに再結合してボリュームに対する I/O を再開(アクティブマップを上書き)できます。この間、他のノードは引き続き I/O を実行することができます。

VxVM は、クラッシュしたノードを追跡します。複数のノードリカバリが指定時刻にクラスタで進行中である場合、VxVM は DRL リカバリの状態の変化を追跡し、I/O の衝突を防ぎます。

マスターノードは、各ボリュームの DRL リカバリマップの更新を一時的に追跡し、複数のユーティリティがリカバリマップを同時に変更するのを回避します。

複数ホストのフェールオーバー設定

Cluster Volume Manager(CVM)のコンテキスト外では、Veritas Volume Manager(VxVM)ディスクグループは一度に 1 つのホストでのみインポート(使用可能に)できます。ホストが(専有)ディスクグループをインポートすると、ディスクグループのボリュームや設定にホストがアクセスできるようになります。システム管理者やシステムソフトウェアが別のホストと同じディスクグループを専有して使う場合、そのディスクグループをすでにインポートしたホスト(インポートを行うホスト)はディスクグループをデポートする(アクセスを放棄する)必要があります。一度デポートすると、ディスクグループは別のホストでインポートできます。

Oracle RAC のように、適切な同期を行わず 2 つのホストで同時に 1 つのディスクグループへのアクセスが許可されている場合、ディスクグループの設定や、場合によってはボリュームの内容が壊れる場合があります。同様の破損が生じる可能性があるのは、raw ディスクパーティション上のファイルシステムやデータベースが 2 つのホストにより同時にアクセスされた場合であり、VxVM に限定した問題ではありません。

インポートロック

非 Cluster Volume Manager(CVM)環境のホストがディスクグループをインポートすると、そのディスクグループのすべてのディスクに対してインポートロックが書き込まれます。インポートロックは、ホストがこのディスクグループをデポートすると解除されます。インポートロックにより、インポートを行っているホストがディスクグループをデポートするまでの間、他のホストがこのディスクグループをインポートするのは回避されます。

具体的には、ホストがディスクグループをインポートするとき、ディスクグループ内のディスクのいずれかが別のホストでロックされているとインポートは通常失敗します。これにより、再ブート後にディスクグループの再インポートが自動的に行われ(自動インポート)、最初のホストが停止している場合にも別のホストによるインポートは回避されます。インポートを行うホストがディスクグループをデポートせずに停止した場合、別のホストがこのディスクグループをインポートするには、ホスト ID のロックを先に解除する必要があります(後で説明します)。

インポートロックには、インポートを行うホストを識別しロックを実行するための、ホスト ID（ホスト名）参照が含まれています。そのため、2 つのホストが同じホスト ID を保有する場合問題が生じることがあります。

Veritas Volume Manager（VxVM）では（デフォルトでは）ホスト ID としてホスト名を使うため、あるマシンと別のマシンで同じホスト名を使っている場合はホスト名を変更することをお勧めします。ホスト名を変更するには、`vxdctl hostid new_hostname` コマンドを実行します。

フェールオーバー

インポートロック規則は、あるシステムから別のシステムへディスクグループが通常切り替えられない環境において正しく動作します。ただし、2 つのホスト Node A と Node B が同じディスクグループのドライブにアクセスできる設定について考えてみましょう。ディスクグループはまず Node A でインポートされますが、Node A がクラッシュした場合、システム管理者は Node B からディスクグループへアクセスする必要があります。このようなフェールオーバーのシナリオを使うと、1 つのノードで障害が発生しデータにアクセスできない場合に、手作業でデータの高可用性を得ることができます。フェールオーバーを高可用性モニタと組み合わせて、データに対する自動的な高可用性を得ることができます。Node B により Node A がクラッシュまたは停止したことが検出されると、Node B はディスクグループをインポート（フェールオーバー）し、ボリュームにアクセスできるようにします。

Veritas Volume Manager ではフェールオーバーをサポートできますが、ディスクグループを別のシステムにインポートする前に、最初のシステムが停止または使えないことの確認は、システム管理者が行うか、外部の高可用性モニタ（Cluster Server（VCS）など）で行う必要があります。

p.994 の「システム間でのディスクグループの移動」を参照してください。

`vxdg`（1M）マニュアルページを参照してください。

ディスクグループ設定の破損

`vxdg import` を `-C`（ロックの解除）や `-f`（強制的なインポート）と組み合わせて使って別のホストで使用中のディスクグループをインポートする場合、ディスクグループ設定が破損する可能性があります。一方のホストがクラッシュまたは停止する前に、インポートされたボリューム上でファイルシステムやデータベースを起動した場合、ボリューム内容の破損も起こることがあります。

このような破損が起きた場合、通常は始めから設定を再構築し、バックアップからすべてのデータを復元する必要があります。通常、各ディスクグループには数多くの設定コピーがありますが、ほとんどの場合、破損が生じるとすべての設定コピーに影響を及ぼすため、冗長性はこの場合役立ちません。

設定バックアップデーモンの vxconfigbackupd が実行されているかぎり、設定が変更されたときには常に Veritas Volume Manager（VxVM）によって設定のバックアップが作成されます。デフォルトでは、バックアップは /etc/vx/cbr/bk に格納されます。設定のバックアップは、vxconfigbackup ユーティリティを使って手動でも行えます。設定は vxconfigrestore ユーティリティを使って再構築できます。

vxconfigbackup、vxconfigbackupd、vxconfigrestoreマニュアルページを参照してください。

ディスクグループの設定が破損すると、通常、設定データベースのレコードが見つからなかったり、レコードが重複します。その結果、vxconfigd の様々なエラー メッセージが次の形式で表示されます。

```
VxVM vxconfigd ERROR
V-5-1-569 Disk group group,Disk disk:
Cannot auto-import group: reason
```

reason には次のようなエラーが示されます。

```
Association not resolved
Association count is incorrect
Duplicate record in configuration
Configuration records are inconsistent
```

通常、これらのエラーは特定のディスクグループの設定コピーと関連して報告されますが、すべてのコピーに適用されます。通常、エラーとともに次のようなメッセージが表示されます。

```
Disk group has no valid configuration copies
```

Cluster Server（VCS）を使うと、すべてのディスクグループのフェールオーバーに関する問題が正しく管理されます。VCS では高可用性モニタが組み込まれ、VxVM、Veritas File System（VxFS）および一般に知られているいくつかのデータベース用のフェールオーバースクリプトが組み込まれています。

-t オプションを vxdg に使うと再ブート時の自動的な再インポートは回避されます。このオプションは、VxVM による自動インポートに依存せず、独自にインポートを制御する（VCS などの）ホストモニタで使う場合に指定する必要があります。

詳しくは、『Veritas InfoScale トラブルシューティングガイド』を参照してください。

Flexible Storage Sharing について

FSS（Flexible Storage Sharing）は、クラスタ全体でローカルストレージのネットワーク共有を有効にします。ローカルストレージは DAS（Direct Attached Storage）または内部

ディスクドライブの形式にする必要があります。ネットワーク共有ストレージは、クラスタノード間のネットワーク相互接続を使って有効化します。

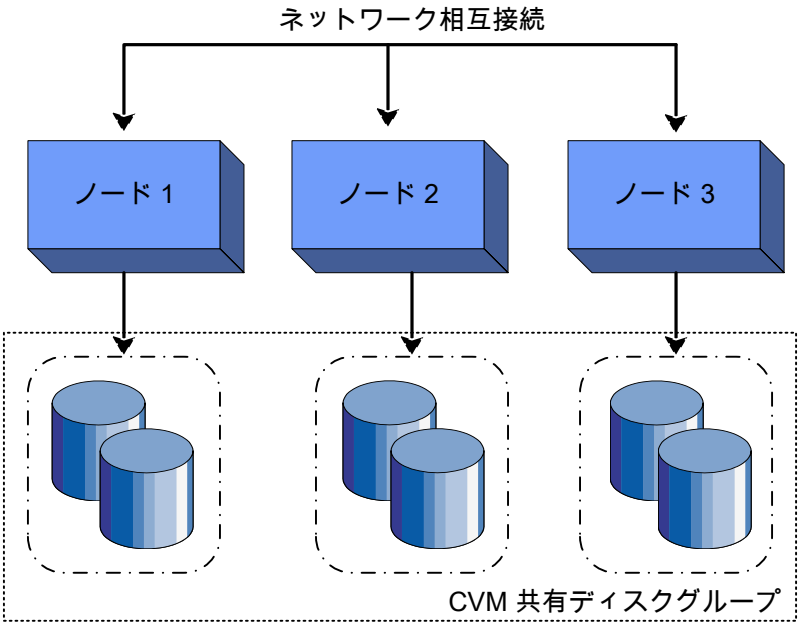
FSS を使うと、ネットワーク共有ストレージは物理共有ストレージと共存できます。論理ボリュームは、共通のストレージ名前空間を作成する両方のタイプのストレージを使って作成できます。ネットワーク共有ストレージを使う論理ボリュームにより、ファイルシステムとアプリケーションに透過的な物理共有ストレージを必要とすることなく、データ冗長性と高可用性、およびディザスタリカバリ機能が提供されます。

FSS は、ローカル SSD を搭載していない可能性があるサービスノードにリモートでキャッシュ保存するために SmartIO 技術とともに使うことができます。

FSS は CVM プロトコルバージョン 140 以降での最大 64 ノードまでを含んでいるクラスタでサポートされています。詳しくは、『Veritas InfoScale リリースノート』を参照してください。

図 6-11 に、Flexible Storage Sharing 環境を示します。

図 6-11 Flexible Storage Sharing 環境



Flexible Storage Sharing の使用例

次に、FSS 機能のいくつかの使用例を示します。

現在の使用例でのローカルストレージの使用	FSS 機能は、Storage Foundation Cluster File System High Availability（SFCFSHA）スタックの現在の使用例をすべてサポートします。SAN ベースのストレージは必要ありません。
オフホスト処理	データの移行: ■ 共有（SAN）ストレージからネットワーク共有ストレージ ■ ネットワーク共有ストレージから SAN ストレージ ■ ノード (DAS)/クラスタに接続したストレージから、ストレージを共有しない異なるノード (DAS)/クラスタに接続したストレージ バックアップ/スナップショット: 追加ノードのバックアップを作成するには、クラスタに参加し、バックアップを作成するホストではなくクラスタの 1 つ以上のノードに接続しているが DAS/共有ストレージでホストされるボリュームやスナップショットから読み込みます。
既存の SFCFSHA 機能で利用できる DAS SSD のメリット	■ クラスタの個々のノードに接続した DAS SSD にわたるミラー化。DAS SSD は、SAN ストレージより高いパフォーマンスを提供します (SSD を含む)。FSS には、クラスタにわたるこれらの SSD を共有する方法が用意されています。 ■ SSD で 1 つのミラー、SAN ストレージで別のミラーを保存すると、SSD が読み込みアクセスを高速化し、SAN ストレージがデータの高可用性を実現します。 ■ Storage Foundation で SSD を使う場合には複数のベストプラクティスがあります。すべての使用例はクラスタ環境で SSD に接続した SAN で実行できます。FSS を使うと、DAS SSD も同じような目的で使うことができます。
ファイルシステムのキャッシュ保存に SmartIO を使う FSS	クラスタのノードに内部 SSD も HDD もある場合は、HDD は FSS を使ってネットワークで共有できます。SSD を使ってリード/ライトバックキャッシュを設定するには、SmartIO を使います。読み込みキャッシュでは、ネットワーク共有 HDD を使って作成したボリュームを扱うことができます。

リモートのキャッシュ保存に SmartIO を使う FSS	FSS は、ローカル SSD デバイスを備えていないノードにキャッシュサービスを提供するために SmartIO と連携して働きます。 このシナリオでは、FSS（Flexible Storage Sharing）によって、ローカル SSD があるノードから SSD がエクスポートされます。FSS で、エクスポートされた SSD のプールがクラスタ内に作成されます。この共用プールから、クラスタの各ノードのキャッシュ領域を作成します。各キャッシュ領域には、作成される特定のノードからのみアクセス可能です。キャッシュ領域のタイプには VxVM または VxFS があります。 クラスタは CVM クラスタである必要があります。 リモート SSD のキャッシュ領域のボリュームレイアウトは、ホスト全体をミラー化するデフォルトの FSS 割り当てポリシーではなく単純なストライプレイアウトに従います。キャッシュに保存する操作により特定ボリュームのパフォーマンスが低下する場合は、そのボリュームのキャッシュを無効にします。キャッシュ領域の作成に使うボリュームは、ディスクグループバージョン 200 以降のディスクグループで作成する必要があります。ただし、ディスクグループバージョン 190 以降のディスクグループで作成したデータボリュームは、FSS によってエクスポートされたデバイスで作成したキャッシュ領域にアクセスできます。 **メモ:** CFS ライトバックキャッシュは、リモートの SSD に作成されたキャッシュ領域ではサポートされません。 詳しくは、『Veritas InfoScale SmartIO for Solid State Drives ソリューションガイド』を参照してください。
キャンパスクラスタ設定	キャンパスクラスタは、サイト間のファイバーチャネル（FC）SAN 接続が確立されていなくても設定できます。
クラウド環境での FSS	FSS (Flexible Shared Storage) 技術を使用して、クラウド環境の「シェアードナッシング」ストレージの制限に対処できます。FSS では、ネットワーク上でクラウドブロックストレージを共有して、シェアードナッシングクラスタを作成できます。 詳しくは、『Veritas InfoScale Solutions in Cloud Environments』のマニュアルを参照してください。

p.460 の「Flexible Storage Sharing の管理」を参照してください。

Flexible Storage Sharing の制限事項

FSS（Flexible Storage Sharing）を使う場合には次の制限事項があります。

- FSS は最大 64 個のノードのクラスタでのみサポートされます。
- ディスクの初期化操作は、ディスクにローカル接続するノードでのみ実行してください。
- FSS は共有ネットワークでブートディスク、非透過ディスク、非 VxVM ディスクの使用をサポートしません。
- ホットリロケーションは FSS ディスクグループでは無効です。
- VxVM クローンディスク操作は、FSS ディスクグループではサポートされていません。
- FSS は複数のホストに接続済みの非 SCSI3 ディスクをサポートしません。
- 動的 LUN 拡張（DLE）はサポートされません。
- FSS は、`vxsnap` 操作を使って作成、またはボリューム作成時に「logtype=dco dcoversion=20」属性を指定して作成したインスタントデータ変更オブジェクト（DCO）のみをサポートします。
- デフォルトでは、下位の mediatype が異なるので `vxassist` による SSD と HDD 間のミラーの作成はサポートされません。この問題を回避するには、特定の mediatype を持つボリューム（デフォルトの mediatype である HDD など）を作成してから、SSD にミラーを追加します。
  次に例を示します。

```
# vxassist -g diskgroup make volume size init=none

# vxassist -g diskgroup mirror volume mediatype:ssd

# vxvol -g diskgroup init active volume
```

p.466 の「vxassist を使用したミラー化ボリュームの管理」を参照してください。

ディスクグループのサブクラスタ化を使用した CVM 環境でのアプリケーションの分離

Veritas InfoScale には、非実働環境でのアプリケーション分離機能の技術プレビューが導入されています。これは、テスト環境のみを対象に先行導入された早期アクセス版です。

Veritas InfoScale は、ディスクグループのサブクラスタの作成時に CVM クラスタでアプリケーションの分離をサポートします。ディスクグループのサブクラスタは、共有ディスクグループのインポートまたはデポートを選択できるノードの論理グループで構成されます。従来の CVM 環境のように、クラスタ内のすべてのノードで共有ディスクグループがイン

ポートまたはデポートされることはありません。そのため、ノードのエラーやクラスタ内のアプリケーションでの設定変更の影響が最小限に抑えられます。

VCS 設定ファイルで CVMCluster リソースの CVMDGSubClust 属性を設定すると、アプリケーション分離機能を有効にできます。クラスタを再起動すると機能が有効になり、共有ディスクグループがクラスタ内のすべてのノードに自動インポートされることはありません。ディスクグループをインポートする最初のノードは、ディスクグループのサブクラスタを形成し、サブクラスタのディスクグループマスターとして選択されます。共有ディスクグループをインポートするクラスタのその他のノードはスレーブとして扱われます。ディスクグループのサブクラスタのマスターノードですべてのディスクグループレベルの操作を実行します。各ディスクグループのサブクラスタのマスターはいつでも切り替えることができます。ノードは、サブクラスタのマスターの役割とその他のサブクラスタのスレーブの役割を果たすことができます。

ノードが SAN に接続できなくなった場合、そのノードの I/O は従来の CVM 環境と同様にディスクグループのサブクラスタ内にある別のノードに送信されます。ディスクグループ内のすべてのディスクでの I/O の失敗が原因でエラーが発生した場合、ディスクグループは無効になるため、ディスクグループを共有するノードはデポートして再度インポートする必要があります。

ノードは、複数のディスクグループのサブクラスタに属することができます。各ディスクグループのサブクラスタは、一部の機能を除いて、クラスタ化された Veritas Volume Manager 環境のすべての機能を提供します。

次の CVM 機能は、ディスクグループのサブクラスタで利用できません。

- ローリングアップグレード
- CVM のキャンパスクラスタの設定
- 異なるディスクグループのサブクラスタマスター (ソースディスクグループとターゲットディスクグループ) を使用した移動と結合の操作
- クラスタ化した Volume Replicator
- クローンデバイス

アプリケーション分離機能は CVM プロトコルバージョン 160 以降でサポートされます。デフォルトでは、インストール後もアップグレード後も無効になります。

図 6-12 に、アプリケーション分離機能を適用するためのディスクグループのサブクラスタ化を示します。

図 6-12 アプリケーション分離機能を適用するためのディスクグループのサブクラスタ化

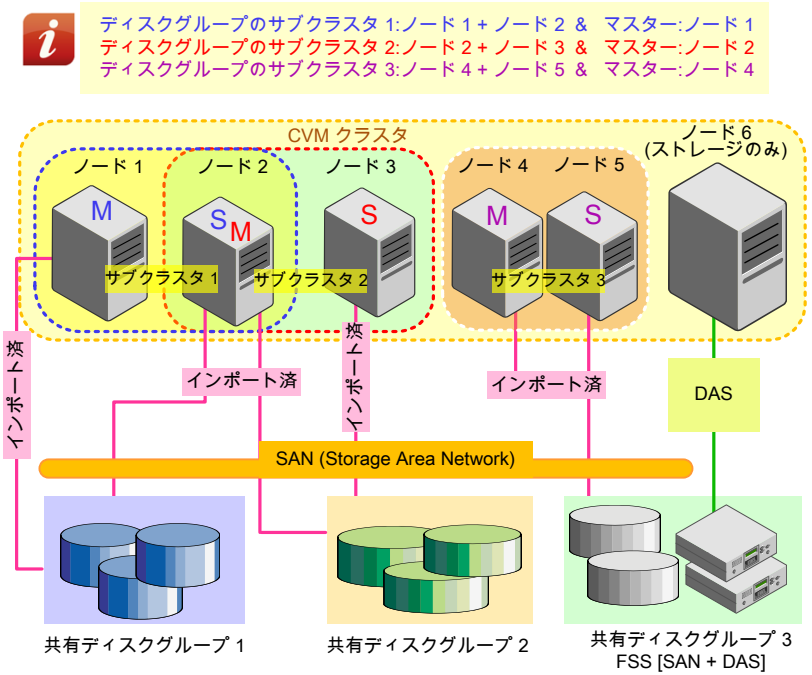
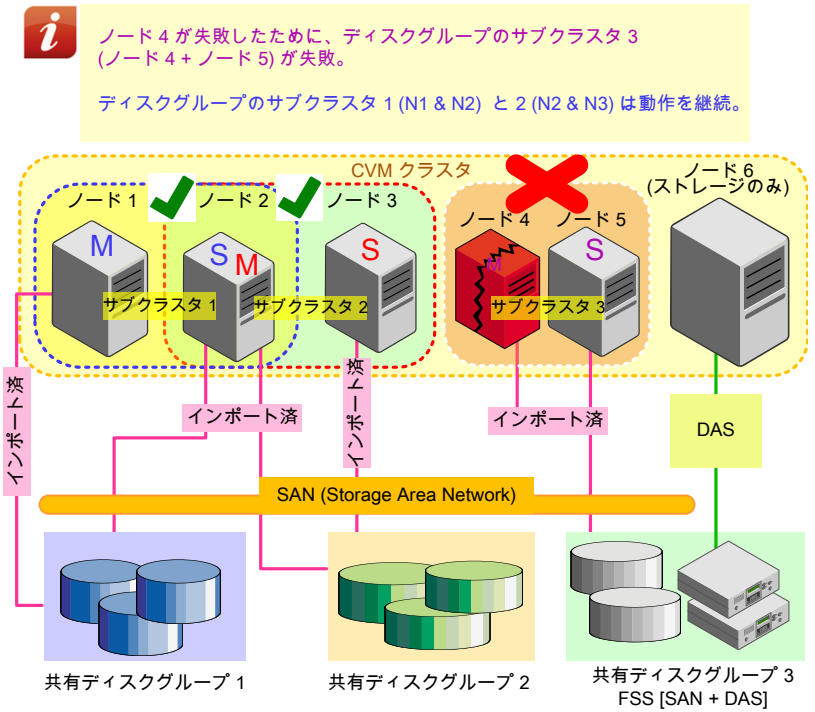


図 6-13 に、ディスクグループのサブクラスタ内のエラー管理を示します。

この図は次のことを示します。

- ディスクグループのサブクラスタを作成し、ディスクグループを選択してインポートする
- SAN と DAS の両方のストレージを含む構成
- ノード 6 は複数のディスクグループのサブクラスタに DAS ストレージをエクスポートするストレージ専用ノード
- ノードは、サブクラスタのマスターとその他のサブクラスタのスレーブという複数の役割を果たすことができる

図 6-13 ディスクグループのサブクラスタ内のエラー管理



ディスクグループのサブクラスタ内の動作変更

ディスクグループのサブクラスタ内の操作には、従来の CVM 環境の動作と異なるものがあります。

表 6-7 に、ディスクグループのサブクラスタにおける特定の操作の動作変更を示します。

表 6-7 ディスクグループのサブクラスタ内の動作変更

操作	動作の変更
共有ディスクグループの自動インポート	デフォルトでは、CVM クラスタの起動時に共有ディスクグループはインポートされません。ノードで手動でインポートする必要があります。 クラスタの一部のディスクグループにストレージがなくても CVM クラスタは正常に起動します。

操作	動作の変更
共有ディスクグループをクラスタ設定に追加する、またはクラスタ設定から削除する (cfsdgadm)	クラスタ内の一部のノードで共有ディスクグループを自動インポートできます。クラスタの起動時にクラスタのファイルシステム環境に必要なディスクグループが VCS によって自動的にインポートされます。
共有ディスクグループの作成	共有ディスクグループを作成するときに、コマンドを実行するノードでのみインポートされます。
共有ディスクグループのインポート	`vxdg -s import` コマンドを使って共有ディスクグループをインポートすると、コマンドを実行するノードでのみディスクグループがインポートされます。 同じディスクグループをインポートするノードは、そのディスクグループのサブクラスタの一部になります。
共有ディスクグループのデポート	`vxdg deport` コマンドを使って共有ディスクグループをデポートすると、コマンドを実行するノードでのみディスクグループがデポートされます。 共有ディスクグループをデポートするノードは、ディスクグループのサブクラスタから切り離され、ディスクグループのサブクラスタのリカバリを開始することがあります。
CVM マスターとディスクグループマスター	各ディスクグループのサブクラスタに、VxVM の設定変更を処理するディスクグループマスターがあります。すべてのディスクグループレベルの操作は、ディスクグループのマスターノードで実行します。 ディスクグループのマスターノードは、ディスクグループのサブクラスタ内にある任意のノードに切り替えることができます。ノードは、ディスクグループに含まれる複数のサブクラスタのディスクグループマスターとして設定できます。
FSS	FSS ディスクグループを作成すると、VxVM はクラスタノードの DAS ストレージをクラスタ内の他のノードに自動エクスポートします。 `vxdisk export` コマンドを使って FSS ディスクグループを作成する前に、ストレージを手動でエクスポートすることもできます。ストレージをすでにエクスポートしている場合、VxVM は自動エクスポート操作をスキップします。 FSS ディスクグループは、任意のノードでインポートできるので、ディスクグループのサブクラスタ外からからストレージを利用できます。ノードは、複数のディスクグループのサブクラスタに DAS ストレージをエクスポートできます。
ディスクグループのサブクラスタ内でのコマンド送信	ディスクグループの操作はディスクグループのマスターノードで実行する必要があります。ディスクグループのサブクラスタのスレーブノードで実行する `vxdg` 以外のすべてのコマンドは、ディスクグループのサブクラスタのマスターノードに送信されます。ディスクグループのサブクラスタ外で実行するディスクグループの操作はサポートされていないので失敗します。

データベースエージェントに関する変更

表 6-8 に、アプリケーション分離機能をサポートするために導入された属性を一覧表示します。

表 6-8 CVM エージェントの属性

CVM エージェント	属性	説明
CVMCluster	CVMDGSubClust	アプリケーション分離機能を有効 (1) または無効 (0) にできます。
CVMVolDG	NodeList	ディスクグループのサブクラスタに属するノードのリストを指定できます。

2

ストレージのプロビジョン

7

新しいストレージのプロビジョニング

この章では以下の項目について説明しています。

新しいストレージのプロビジョニング

次の手順では、新しいストレージをプロビジョニングする方法について説明します。シンストレージの Storage Foundation をプロビジョニングする場合、Storage Foundation とシンストレージの動作について理解する必要があります。

p.771 の「Storage Foundation Cluster File System High Availability のシン最適化ソリューションについて 」を参照してください。

この手順はボリュームとファイルシステムが基本設定であることを想定しています。設定をカスタマイズするためにより多くのオプションを利用可能です。

p.217 の「割り当て動作のカスタマイズ」を参照してください。

p.259 の「VxFS ファイルシステムの作成」を参照してください。

新しいストレージをプロビジョニングするには

1 LUN を設定します。LUN の作成、マスク、バインドを行う方法については、ストレージアレイのマニュアルを参照してください。

2 次のいずれかのコマンドを使って、Veritas Volume Manager（VxVM）で使う LUN を初期化します。

vxdisksetup コマンドを使う方法をお勧めします。

```
# vxdisksetup -i 3PARDATA0_1
# vxdisk init 3PARDATA0_1
```

3 ディスクグループに LUN を追加します。

- LUN 用のディスクグループがなければ、ディスクグループを作成します。

```
# vxdg init dg1 dev1=3PARDATA0_1
```

- すでに LUN 用のディスクグループがある場合は、そのディスクグループに LUN を追加します。

```
# vxdg -g dg1 adddisk 3PARDATA0_1
```

4 LUN にボリュームを作成します。

```
# vxassist -b -g dg1 make vol1 100g 3PARDATA0_1
```

5 ボリュームに Veritas File System（VxFS）ファイルシステムを作成します。

```
# mkfs -t vxfs /dev/vx/rdsk/dg1/vol1
```

6 ファイルシステムにマウントポイントを作成します。

```
# mkdir /mount1
```

7 次のコマンドを実行して、ファイルシステムをマウントします。

```
# mount -t vxfs /dev/vx/dsk/dg1/vol1 /mount1
```

新しい LUN の追加による既存のストレージの拡張

次の手順では、新しい LUN を追加することによって既存のストレージを拡張する方法について説明します。

新しい LUN を追加して既存のストレージを拡張するには

1 LUN を作成し、設定します。

2 ディスクグループに LUN を追加します。

```
# vxdg -g dg1 adddisk 3PARDATA0_2
```

3 必要なサイズまでボリュームとファイルシステムを拡張します。次に例を示します。

```
# vxresize -b -F vxfs -g dg1 vol1 200g
```

LUN の拡張による既存のストレージの拡張

次の手順では、LUN を拡張することによって既存のストレージを拡張する方法について説明します。

LUN を拡張して既存のストレージを拡張するには

1 既存の LUN を拡張します。LUN の作成、マスク、バインドを行う方法については、ストレージアレイのマニュアルを参照してください。

2 Veritas Volume Manager（VxVM）が新しい LUN サイズを認識するようにします。

```
# vxdisk -g dg1 resize 3PARDATA0_1
```

3 新しい最大ボリュームサイズを計算します。

```
# vxassist -g dg1 -b maxgrow vol1
```

4 必要なサイズまでボリュームとファイルシステムを拡張します。

```
# vxresize -b -F vxfs -g dg1 vol1 200g
```

vxlist で SFCFSHA の情報を表示する

vxlist コマンドは、SFCFSHA の設定を統合して表示する、表示コマンドです。vxlist コマンドは、Veritas Volume Manager（VxVM）と Veritas File System（VxFS）からの情報を統合します。vxlist コマンドは情報表示のために各種のオプションを提供しています。たとえば、ファイルシステム情報にボリューム、ディスクグループおよびその他の情報を含めて表示するには、コマンドを次の形式で使用します。以前のリリースでは、次の情報を取得するには、少なくとも 2 つのコマンドを実行する必要がありました。

```
# /opt/VRTSsfmh/bin/vxlist fs
```

```
TY FS  FSTYPE SIZE   FREE   %USED DEVICE_PATH          MOUNT_POINT
fs /   ext3   65.20g 51.70g 17%   /dev/sda1                /
fs mnt vxfs   19.84g  9.96g 49%   /dev/vx/dsk/bardg/vol1   /mnt
```

vxlist コマンドのヘルプを表示するには、次のコマンドを入力してください。

```
# vxlist -H
```

vxlist(1m) マニュアルページを参照してください。

8

ストレージの設定のための高度な割り当て方法

この章では以下の項目について説明しています。

割り当て動作のカスタマイズ

デフォルトでは、vxassist コマンドは基本必要条件を満たす使用可能なストレージにボリュームを作成します。vxassist は使用可能なディスク領域を検索します。その領域

は、レイアウト指定に適合し、かつ空き領域を最大限に活用する設定で割り当てられます。vxassist コマンドでは、目的のボリュームの基本属性を入力するだけで、必要なプレックスとサブディスクを作成できます。

シンストレージの Storage Foundation Cluster File System High Availability をプロビジョニングする場合、Storage Foundation Cluster File System High Availability とシンストレージの動作について理解する必要があります。

p.771 の「Storage Foundation Cluster File System High Availability のシン最適化ソリューションについて 」を参照してください。

また、既存のボリュームを vxassist コマンドを使用して変更すると、vxassist コマンドは下位のオブジェクトまたは関連付けられたオブジェクトを自動的に変更します。vxassist コマンドは、ユーザーがコマンドラインで特定の値を指定しないかぎり、多くのボリューム属性に対しデフォルト値を使います。デフォルト値を変更することで、vxassist コマンドのデフォルトの動作をカスタマイズできます。

p.219 の「vxassist のデフォルト値の設定」を参照してください。

vxassist コマンドは、デフォルトのルールに従ってデフォルトのディスクグループにボリュームを作成します。別のディスクグループを使うには、vxassist コマンドに -g diskgroup オプションを指定します。

p.946 の「デフォルトのディスクグループの名前の付け方」を参照してください。

特定のボリュームに特定の特性を割り当てる場合は、vxassist コマンドラインに追加の属性を指定できます。これらは、割り当てるディスクのタイプ、またはその他の属性(ストライプユニットサイズ(ストライプ幅)、RAID 5 またはストライプボリュームのカラム数、ミラー数、ログ数、ログタイプなど)を選択するためのストレージ指定となります。

vxassist で使用可能なキーワードと属性について詳しくは、vxassist(1M) マニュアルページを参照してください。

割り当ての属性を使用して、表 8-1 で示す割り当ての動作のタイプを指定できます。

表 8-1　　割り当て動作のタイプ

割り当て動作	手順
ボリュームのレイアウト	p.240 の「特定のレイアウトのボリュームの作成」を参照してください。
メディアタイプ	p.249 の「特定のメディアタイプのボリュームの作成」を参照してください。
特定のディスク、サブディスク、プレックスの場所	p.248 の「指定したディスクにおけるボリュームの作成」を参照してください。
順次ディスク割り当て	p.253 の「ボリュームのストレージに対する順次ディスク割り当て」を参照してください。

割り当て動作	手順
サイトベースの割り当て	p.256 の「サイトベースの割り当て」を参照してください。
読み込みのポリシーの設定	p.256 の「ミラーボリュームの読み取りポリシーの変更」を参照してください。

さらに、vxassist ユーティリティは、効率的かつ柔軟にボリューム割り当てを定義して管理できるように、さまざまなしくみを提供します。

p.219 の「vxassist のデフォルト値の設定」を参照してください。

p.220 の「ルールを使った、ボリューム割り当ての効率向上」を参照してください。

p.223 の「永続的な属性について」を参照してください。

p.226 の「割り当て用のディスククラスのカスタマイズ」を参照してください。

p.228 の「use 節と require 節を使った vxassist 操作のための割り当て制約の指定」を参照してください。

p.236 の「永続的な属性の use および require タイプの管理 」を参照してください。

vxassist のデフォルト値の設定

vxassist コマンドが使うデフォルト値は、ファイル /etc/default/vxassist で指定できます。このファイルに記述されているデフォルト値は、コマンドラインまたは -d オプションを使って指定した別のデフォルトファイルで上書きされないかぎり有効です。デフォルト値は、コマンドラインで指定された値が常に優先されます。また、vxassist には、他で指定した値が見つからない場合に使う一連のデフォルト値も組み込まれています。

/etc/default ディレクトリと vxassist デフォルトファイルがシステムに存在していない場合は、新たに作成する必要があります。

デフォルトファイルのエントリの形式は、改行で区切られた属性値ペアのリストです。これらの属性値ペアは、vxassist コマンドラインでオプションとして指定したものと同じです。

vxassist(1M)マニュアルページを参照してください。

/etc/default/vxassist ファイルに指定されているデフォルト属性を表示するには、次の形式の vxassist コマンドを使います。

```
# vxassist help showattrs
```

次に、vxassist デフォルトファイルの例を示します。

```
#  By default:
#  create unmirrored, unstriped volumes
#  allow allocations to span drives
```

```
#  with RAID-5 create a log, with mirroring don't create a log
#  align allocations on cylinder boundaries
    layout=nomirror,nostripe,span,nocontig,raid5log,noregionlog,
    diskalign

#  use the fsgen usage type, except when creating RAID-5 volumes
    usetype=fsgen
#  allow only root access to a volume
     mode=u=rw,g=,o=
     user=root
     group=root

#  when mirroring, create two mirrors
     nmirror=2
#  for regular striping, by default create between 2 and 8 stripe
#  columns
     max_nstripe=8
     min_nstripe=2

#  for RAID-5, by default create between 3 and 8 stripe columns
     max_nraid5stripe=8
     min_nraid5stripe=3

#  by default, create 1 log copy for both mirroring and RAID-5 volumes

    nregionlog=1
    nraid5log=1

#  by default, limit mirroring log lengths to 32Kbytes
    max_regionloglen=32k

#  use 64K as the default stripe unit size for regular volumes
    stripe_stwid=64k

#  use 16K as the default stripe unit size for RAID-5 volumes
    raid5_stwid=16k
```

ルールを使った、ボリューム割り当ての効率向上

vxassist コマンドを使うと、一組のボリューム割り当てルールを作成し、1 つの名前を付けて定義できます。ボリュームの割り当て要求でこの名前を指定すると、このルールで定義されているすべての属性は、vxassist でボリュームを作成するときに引き継がれます。

ボリュームの割り当てルールを作成することには、次の利点があります。

- ルールによって入力が合理化され、エラーが減ります。比較的複雑な割り当てルールを 1 つの場所で一度定義すると、ルールを再利用できます。
- ルールを使うと、一連のサーバー全体を含めて、環境内の動作を標準化できます。

たとえば、一連のサーバーでストレージの階層を標準化できるように、割り当てルールを作成できます。次の必要条件があったと仮定します。

階層 1	特定の一組のアレイタイプ間でのエンクロージャのミラー化
階層 2	特定の一組のアレイタイプ間でのミラー化されていないストライプ化
階層 0	ソリッドステートドライブ（SSD）ストレージを選択する

ボリューム割り当ての各必要条件に対してルールを作成し、ルール tier1、tier2、tier0 と名前を付けることができます。

特定目的のボリュームを作成するたびに同じ属性を持つボリュームが作成されるように、ルールを定義することもできます。たとえば、実稼動データベース用のボリュームを作成する場合は、productiondb といった名前のルールを作成できます。ホームディレクトリ用の標準化されたボリュームを作成する場合は、homedir といった名前のルールを作成できます。高パフォーマンスのインデックスボリュームを標準化する場合は、dbindex といった名前のルールを作成できます。

ルールファイルの形式

ルールを作成するとき、/etc/default/vxassist ファイルではそれらを定義しません。別のファイルにルールを作成し、/etc/default/vxassist にはパス情報を追加します。デフォルトでは、ルールファイルは /etc/default/vxsf_rules からロードされます。この場所は、/etc/default/vxassist の属性 rulefile=/**パス**/*rule_file_name* で上書きできます。また、コマンドラインで追加のルールファイルを指定できます。

ルールファイルでは、次の規則が使われます。

- 空白行は無視されます。
- シャープ記号 # を使ってコメントを始めます。
- 埋め込みスペース、復帰改行、タブを含む可能性のある文字列には、C 言語スタイルで引用符を使います。たとえば、description 属性のテキストを引用符で囲みます。
- トークンはスペースで分けます。
- 1 行よりも長いルールは波カッコ（{}）で囲みます。

ルールファイル内では、ボリュームの割り当てルールは次の形式になっています。

```
volume rule rulename vxassist_attributes
```

この構文では、*rulename* という名前のルールを定義しています。これは、列挙する vxassist 属性を簡潔に表すものにします。ルールは rule=*rulename*[,*rulename*,...] の属性を使って他のルールを参照できます。これらの記述では、そのルールから現在定義中のルールにすべての属性を追加します。ルール定義で指定する属性は、参照によって指定したルール内に競合している属性が含まれている場合、それらを上書きします。description=*description_text* 属性を使うと、ルールに説明を追加できます。

基本的なルールファイルを次に示します。ファイルに含まれる最初のルールである base では、logtype 属性と persist 属性を定義しています。このファイルの残りのルールである tier0、tier1、tier2 は、base ルールを参照し、自身の階層に固有の属性も定義しています。ルールを参照すると、1 つの場所で定義した属性を、他のルールで再利用できます。

```
# Create tier 1 volumes mirrored between disk arrays, tier 0 on SSD,

# and tier 2 as unmirrored. Always use FMR DCO objects.
volume rule base { logtype=dco persist=yes }
volume rule tier0 { rule=base mediatype:ssd tier=tier0 }
volume rule tier1 { rule=base mirror=enclosure tier=tier1 }
volume rule tier2 { rule=base tier=tier2 }
```

次のルールファイルには、複数行にわたって記述された複雑な定義が含まれています。

```
volume rule appXdb_storage {
    description="Create storage for the database of Application X"
    rule=base
    siteconsistent=yes
    mirror=enclosure
}
```

ボリュームを作成するルールの使用

vxassist コマンドを使ってボリュームを作成するときには、コマンドラインにルール名を含めることができます。たとえば、vxsf_rules ファイルの内容は次のとおりです。

```
volume rule basic { logtype=dco }
volume rule tier1 {
        rule=basic
        layout=mirror
        tier=tier1
}
```

次の例では、ディスクグループ dg3 内にボリューム vol1 を作成するときには、コマンドラインで tier1 ルールを指定できます。コマンドラインで入力する属性に加えて、vol1 には tier1 で定義した属性が付与されます。

```
vxassist -g dg3 make vol1 200m rule=tier1
```

次の vxprint コマンドでは、ディスクグループ dg3 の属性が表示されます。出力には新しいボリュームの vol1 が含まれています。

```
vxprint -g dg3
TY NAME         ASSOC        KSTATE   LENGTH   PLOFFS   STATE    TUTIL0  PUTIL0
dg dg3          dg3          -        -        -        -        -       -

dm ibm_ds8x000_0266 ibm_ds8x000_0266 - 2027264 -        -        -       -
dm ibm_ds8x000_0267 ibm_ds8x000_0267 - 2027264 -        -        -       -
dm ibm_ds8x000_0268 ibm_ds8x000_0268 - 2027264 -        -        -       -

v  vol1         fsgen        ENABLED  409600   -        ACTIVE   -       -
pl vol1-01      vol1         ENABLED  409600   -        ACTIVE   -       -
sd ibm_ds8x000_0266-01 vol1-01 ENABLED 409600  0        -        -       -
pl vol1-02      vol1         ENABLED  409600   -        ACTIVE   -       -
sd ibm_ds8x000_0267-01 vol1-02 ENABLED 409600  0        -        -       -
dc vol1_dco     vol1         -        -        -        -        -       -
v  vol1_dcl     gen          ENABLED  144      -        ACTIVE   -       -
pl vol1_dcl-01  vol1_dcl     ENABLED  144      -        ACTIVE   -       -
sd ibm_ds8x000_0266-02 vol1_dcl-01 ENABLED 144 0        -        -       -
pl vol1_dcl-02  vol1_dcl     ENABLED  144      -        ACTIVE   -       -
sd ibm_ds8x000_0267-02 vol1_dcl-02 ENABLED 144 0        -        -       -
```

次の vxassist コマンドでは、vol1 が tier1 内にあることを確認しています。ルール tier1 は正常に適用されました。

```
vxassist -g dg3 listtag
TY   NAME             DISKGROUP        TAG
==========================================================
v    vol1             dg3              vxfs.placement_class.tier1
```

永続的な属性について

vxassist コマンドでは、ボリュームについてボリューム割り当ての特定の属性を記録できます。これらの属性は永続的な属性と呼ばれます。後でボリュームに対して割り当て操作を行う場合に便利な、ボリュームの拡大やエンクロージャのミラー化などの属性を記録できます。また、特定のプロパティ(エンクロージャタイプ、ディスクタグ、メディアタイプな

ど)を持つストレージへの割り当てを制限できます。一方で、ボリュームの長さは有用ではなく、通常は特定ディスクのリストも有用ではありません。

永続的な属性は、次の操作の割り当て要求に対して、取得して適用できます(変更の指定も可能)。

- ボリュームの拡大または縮小
- 移動
- 再レイアウト
- ミラー(mirror)
- ログの追加

永続的な属性を使うと、注意深く記述した割り当て属性をボリュームの作成時に記録し、ボリュームに対する以後の割り当て操作のために保存できます。永続的な属性は修正や拡張も、破棄も可能です。たとえば、当初はミラー化されていなかったボリュームの分離ルールを追加して保存できます。また、制限が厳しすぎることが判明したボリュームの割り当てルールを一時的に停止したり、必要な割り当てが行えるようにルールを破棄したりできます。

永続的な属性を使って、コマンドラインまたはルールファイルの割り当て属性を記録できます。

p.224 の「永続的な属性の使用」を参照してください。

インテント管理操作(setrule、changerule、clearrule、listrule)を使って、永続的な属性の使用や必要なタイプを管理できます。

p.236 の「永続的な属性の use および require タイプの管理 」を参照してください。

永続的な属性の使用

ボリューム割り当ての属性を定義できるため、その後の操作でそれらの属性を再利用できます。これらの属性は永続的な属性と呼ばれ、一組の隠しボリュームタグに保存されます。`persist` 属性は、属性が永続的かどうかと、現在のコマンドがどのように既存の永続的な属性を使ったり修正したりできるかを決定します。永続性のルールは、デフォルトファイル、ルール、コマンドラインで指定できます。詳しくは、`vxassist` マニュアルページを参照してください。

永続的な属性がどのように機能するかを具体的に示すため、次の `vxsf_rules` ファイルを使います。このファイルには `mediatype` 属性を定義するルールの rule1 が含まれています。このルールでは `persist` 属性も使って `mediatype` 属性を永続的な属性にしています。

```
# cat /etc/default/vxsf_rules
volume rule rule1 { mediatype:ssd persist=extended }
```

次のコマンドでは、LUN の ibm_ds8x000_0266 と ibm_ds8x000_0268 がソリッドステートディスク(SSD)デバイスであることを確認しています。

```
# vxdisk listtag
DEVICE               NAME                VALUE
ibm_ds8x000_0266     vxmediatype         ssd
ibm_ds8x000_0268     vxmediatype         ssd
```

次のコマンドでは、ディスクグループ dg3 内にボリューム vol1 を作成しています。rule1 はコマンドラインで指定されるため、それらの属性も vol1 に適用されます。

```
# vxassist -g dg3 make vol1 100m rule=rule1
```

次のコマンドでは、ボリューム vol1 が rule1 で指定されているとおりに、SSD デバイスの ibm_ds8x000_0266 以外の場所に作成されることが示されています。

```
# vxprint -g dg3
TY NAME         ASSOC        KSTATE   LENGTH   PLOFFS   STATE    TUTIL0  PUTIL0
dg dg3          dg3          -        -        -        -        -       -

dm ibm_ds8x000_0266 ibm_ds8x000_0266 - 2027264 -        -        -       -
dm ibm_ds8x000_0267 ibm_ds8x000_0267 - 2027264 -        -        -       -
dm ibm_ds8x000_0268 ibm_ds8x000_0268 - 2027264 -        -        -       -

v  vol1         fsgen        ENABLED  204800   -        ACTIVE   -       -
pl vol1-01      vol1         ENABLED  204800   -        ACTIVE   -       -
sd ibm_ds8x000_0266-01 vol1-01 ENABLED 204800  0        -        -       -
```

次のコマンドでは、rule1 で定義されている属性が表示されます。

```
# vxassist -g dg3 help showattrs rule=rule1
alloc=mediatype:ssd
persist=extended
```

永続的な属性が定義されていない場合は、次のコマンドによってハードディスクドライブ(HDD)デバイス上で vol1 が拡張されます。しかし、このセクションの始めで、mediatype:ssd は永続的な属性として定義されていました。したがって、次のコマンドはこの最初の意図に従って、SSD デバイス上のボリュームを拡張します。

```
# vxassist -g dg3 growby vol1 1g
```

次の vxprint コマンドでは、ボリュームが SSD デバイス上で拡張されたことを確認しています。

```
# vxprint -g dg3
TY NAME         ASSOC        KSTATE   LENGTH   PLOFFS   STATE    TUTIL0  PUTIL0
```

```
dg dg3          dg3          -        -        -        -        -       -

dm ibm_ds8x000_0266 ibm_ds8x000_0266 - 2027264 -        -        -       -
dm ibm_ds8x000_0267 ibm_ds8x000_0267 - 2027264 -        -        -       -
dm ibm_ds8x000_0268 ibm_ds8x000_0268 - 2027264 -        -        -       -

v  vol1         fsgen        ENABLED  2301952  -        ACTIVE   -       -
pl vol1-01      vol1         ENABLED  2301952  -        ACTIVE   -       -
sd ibm_ds8x000_0266-01 vol1-01 ENABLED 2027264 0        -        -       -
sd ibm_ds8x000_0268-01 vol1-01 ENABLED 274688  2027264  -        -       -
```

割り当て用のディスククラスのカスタマイズ

vxassist コマンドは、割り当て用のストレージ仕様を表示するためのディスククラスを受け入れます。ディスククラスは、ディスクと自動的に関連付けられる、内部で検出される属性です。vxassist を使って、割り当て要求に対してディスククラスを指定し、割り当て用のストレージのタイプを表示できます。

ディスククラスについて詳しくは、vxassist（1M）マニュアルページの「ストレージの仕様」セクションを参照してください。

ディスククラスを次のようにカスタマイズできます。

- カスタマイズされたエイリアス名を作成します。
  p.226 の「ユーザー定義ディスククラスのエイリアス名」を参照してください。
- ディスククラスの優先順位をカスタマイズします。
  p.227 の「ユーザー定義ディスククラスの優先順位」を参照してください。

また、カスタマイズされたユーザー定義ディスククラスも作成できます。

p.228 の「ユーザー定義ディスククラス」を参照してください。

ユーザー定義ディスククラスのエイリアス名

便宜上、既存のストレージ仕様のディスククラスのエイリアス名を定義できます。通常エイリアスは、短く、よりユーザーにわかりやすい名前にします。対応するディスククラスの代わりにエイリアスを使って、vxassist 割り当ての制約を指定できます。ルールファイルのエイリアス名を定義します。

たとえば、基本ディスククラス「arraytype」にエイリアスとして「atyp」を定義するには、ルールファイルに次の文を含めます。

```
class alias atyp=arraytype
```

上のルールファイルが使われる場合、割り当て用にエイリアス「atyp」を指定できます。たとえば、次の制約の仕様はボリューム作成のために A/A アレイからストレージを割り当てます。

```
# vxassist -g dgname make volnamevolsize use=atyp:A/A
```

ユーザー定義ディスククラスの優先順位

vxassist コマンドは、ミラー制限（mirrorconfine、wantmirrorconfine）、ミラー分離（mirror、wantmirror）およびストライプ分離（stripe、wantstripe）制約で指定されたディスククラスのデフォルトの優先順位を適用します。優先度が高いクラスは、ミラー化またはストライプ化するときに割り当てに使われます。別の優先順位が必要な場合は、これらのディスククラスのデフォルトの順位を変更できます。

メモ: 「site」クラスは常に優先度が最上位であり、その順序は上書きできません。

ルールファイルのカスタマイズされた優先順位を定義します。番号が大きいほど、クラスの優先度はより高くなります。

次に、ミラーおよびストライプ分離または制限制約でサポートされるクラス名のデフォルトの優先順位を示します。

site	order=1000
vendor	order=900
arrayproduct	order=800
array	order=700
arrayport	order=600
hostport	order=400

優先順位の許容範囲は 0 ～ 1000 です。

たとえば、デフォルトでは array クラスの優先度が hostport クラスより高く設定されています。hostport クラスの優先度を高くするには、より大きい番号を割り当てます。クラスの順位を定義するには、ルールファイルに次の文を含めます。

```
class define array order=400
class define hostport order=700
```

上のルールが使われる場合、次のコマンドは array クラスよりも hostport クラス全体をミラーリングします。

```
# vxassist -g dgname make volnamevolsize mirror=array,hostport
```

ユーザー定義ディスククラス

vxassist コマンドのストレージ仕様で使用するためにカスタマイズされたディスククラスを定義できます。カスタマイズされたディスククラスは、ユーザー定義デバイスの分類とグループ化を可能にします。これらのディスククラスを制御割り当てに使うことができます。カスタマイズされたディスククラスは、一連のディスクと関連付けられるユーザー定義プロパティです。このプロパティはディスククラスとして特定の制約を満たすディスクに接続されます。

vxassist 割り当ての制約を指定するために、他のストレージ仕様のディスククラスのようなカスタムディスククラスを使うことができます。ルールファイルのカスタムディスククラスを定義します。

例

ルールファイルの次の定義により、ユーザー定義プロパティ「poolname」は参照されたディスクに関連付けられます。日立または IBM として定義されたアレイベンダープロパティを持つすべてのデバイスは、poolname「finance」としてマーク付けされます。DGC または EMC として定義されたアレイベンダープロパティを持つすべてのデバイスは、poolname「admin」としてマーク付けされます。

```
disk properties vendor:HITACHI {
     poolname:finance
}
disk properties vendor:IBM {
     poolname:finance
}
disk properties vendor:DGC {
     poolname:admin
}
disk properties vendor:EMC {
     poolname:admin
}
```

ユーザー定義ディスククラス「poolname」は、割り当てに使うことができます。たとえば、次の制約の仕様はボリューム作成のために poolname「admin」からディスクを割り当てます。

```
# vxassist -g dgname make volnamevolsize poolname:admin
```

use 節と require 節を使った vxassist 操作のための割り当て制約の指定

vxassist コマンドは、割り当て用のさまざまなストレージ仕様を受け入れます。require 制約と use 制約は、割り当て用の詳細なストレージ仕様を指定する方法です。これらの制約は、意図されたプロパティのインターセクションセットまたはユニオンセットからのディ

スクの選択を可能にします。既存のメソッド `alloc` と `logdisk` 節よりも正確な割り当てによって、ディスクのセットを指定できます。use および require 制約は、データ、ログ、またはその両方に適用できます。

制約は、次のいずれかのタイプになります。

- require 制約
  制約の仕様のすべてを満たす必要があります。そうでない場合、割り当ては失敗します。require 制約はインターセクションセットとして動作します。たとえば、特定のアレイタイプの特定のアレイベンダー AND からディスクを割り当てます。
- use 制約
  制約の仕様の少なくとも 1 つを満たす必要があります。そうでない場合、割り当ては失敗します。use 制約はユニオンセットとして動作します。たとえば、指定されたエンクロージャ(enclrA または enclrB)のいずれかからディスクを割り当てます。

ディスクグループのバージョンが 180 またはそれ以上の場合は、use および require タイプの制約は、デフォルトでボリューム用に永続的です。これらの節のデフォルトの保持は、指定済みのインテントを破壊せずに、それ以上の割り当て操作(拡大など)を可能にします。

`vxassist` コマンドラインの use または require 節で、複数のストレージ仕様をカンマで区切って指定できます。また、`vxassist` コマンドラインで複数の use または require 節を指定できます。

p.230 の「複数の require および use 制約の相互関係」を参照してください。

`vxassist` インテント管理操作(`setrule`、`changerule`、`clearrule`、`listrule`)を使って、永続的な require および use 制約を管理します。

p.236 の「永続的な属性の use および require タイプの管理 」を参照してください。

require 制約について

制約の「require」タイプは、割り当てが、制約内のストレージ仕様と一致するストレージを選択するように指定します。したがって、require 制約はインターセクションセットまたは論理的な AND 操作のように動作します。仕様のいずれかが満たされていないと、操作は失敗します。require 制約を指定する属性名は次のとおりです。

- `require`
  制約はデータとログの割り当ての両方に適用されます。
- `logrequire`
  制約はログの割り当てにのみ適用されます。
- `datarequire`
  制約はデータの割り当てにのみ適用されます。

ストレージ仕様が「!」によって取り消された場合、割り当てはそのストレージ仕様と一致するストレージを除外します

メモ: 制約の require タイプがクラスが同じでインスタンスが異なる状態で提供された場合、これらのインスタンスはインターセクトされずにユニオナイズされます。つまり、割り当てはこれらのストレージ仕様をすべて満たすストレージを選択します（制約の use タイプでも同様）。

---

p.230 の「複数の require および use 制約の相互関係」を参照してください。

use 制約について

制約の「use」タイプは、割り当てが、制約内のストレージ仕様と少なくとも 1 つが一致するストレージを選択するように指定します。したがって、use 制約はユニオンセットまたは論理的な OR 操作のように動作します。仕様のいずれかが満たされていないと、操作は失敗します。use 制約を指定する属性名は次のとおりです。

- `use`
  制約はデータとログの割り当ての両方に適用されます。
- `loguse`
  制約はログの割り当てにのみ適用されます。
- `datause`
  制約はデータの割り当てにのみ適用されます。

p.230 の「複数の require および use 制約の相互関係」を参照してください。

ストレージ仕様が「!」によって取り消された場合、割り当てはそのストレージ仕様と一致するストレージを除外します

複数の require および use 制約の相互関係

`vxassist` コマンドラインで複数の use または require 節を指定できます。すべての組み合わせがサポートされているわけではありません。ただし、サポート対象の制約仕様を組み合わせることで、サポート対象にすることはできます。

制約の範囲には、data-specific（datause または datarequire）、log-specific（loguse または logrequire）、またはデータとログの両方（use または require）に適用される general があります。

メモ: Veritas は use または require 制約を、直接ストレージ仕様や `alloc` または `logdisk` のような他の節と組み合わせないことを推奨します。

次のルールは、複数の use または require 節が指定されたときに適用されます。

- 同じ範囲の複数の use 制約は、ストレージ仕様の少なくとも 1 つが満たされるようにユニオナイズされます。つまり、複数の use 節、複数の datause 節、または複数の loguse 節です。
- 同じ範囲の複数の require 制約は、すべてのストレージの仕様が満たされるようにインターセクトされます。つまり、複数の require 節、複数の datarequire 節、または複数の logrequire 節です。
- 同じ範囲の require および use 制約は相互にインターセクトされます。つまり、require 節と use 節、datarequire 節と datause 節、または logrequire 節と loguse 節です。use ストレージ仕様の少なくとも 1 つが満たされ、require ストレージ仕様はすべて満たされている必要があります。たとえば、datause 節と datarequire 節が併用された場合、データの割り当ては datause 仕様の少なくとも 1 つ、および datarequire 仕様のすべてを満たす必要があります。
- data-specific 制約と log-specific 制約は併用できます。これらはデータとログにそれぞれ独立して適用されます。つまり、datause 節と loguse 節または logrequire 節、datarequire 節と loguse 節または logrequire 節です。たとえば、datarequire 節をデータの割り当て制御に使うと同時に logrequire 節をログの割り当て制御に使うことができます。
- vxassist コマンドは、data-specific または log-specific 制約と general 範囲の制約の組み合わせをサポートしません。たとえば、require 節は logrequire 節または datarequire 節とともに使うことができません。ただし、サポート対象の制約仕様を組み合わせることで、サポート対象にすることはできます。

表 8-2 に、複数の制約が指定された場合の制約の各タイプの相互関係のルールの概略を示します。

表 8-2 require および use 制約の組み合わせ

範囲	相互にユニオナイズされる	相互にインターセクトされる	独立して適用される
データ	datause - datause	datarequire - datause datarequire - datarequire	datause - loguse datause - logrequire datarequire - loguse datarequire - logrequire

範囲	相互にユニオナイズされる	相互にインターセクトされる	独立して適用される
ログ	loguse - loguse	logrequire - loguse logrequire - logrequire	loguse - datause loguse - datarequire logrequire -datause logrequire - datarequire
全般 - ログとデータ	use - use	use - require require - require	N/A

require および use 制約の例

次に、ストレージ割り当て用の use および require 制約の例を示します。

例 1 - require 制約

この例は、2 つのアレイ(emc_clariion0 と ams_wms0)からのディスクを持つディスクグループの require 制約を示します。アレイは両方とも同じ HBA hostportid(06-08-02)で接続されていますが、それぞれ異なるアレイタイプ(A/A と A/A-A)を持っています。

次の出力はディスクグループの情報を示します。

```
# vxprint -g testdg
TY NAME         ASSOC        KSTATE   LENGTH   PLOFFS   STATE    TUTIL0  PUTIL0
dg testdg       testdg       -        -        -        -        -       -

dm ams_wms0_359 ams_wms0_359 -        2027264  -        -        -       -
dm ams_wms0_360 ams_wms0_360 -        2027264  -        -        -       -
dm ams_wms0_361 ams_wms0_361 -        2027264  -        -        -       -
dm ams_wms0_362 ams_wms0_362 -        2027264  -        -        -       -
dm emc_clariion0_0 emc_clariion0_0 -  4120320  -        -        -       -
dm emc_clariion0_1 emc_clariion0_1 -  4120320  -        -        -       -
dm emc_clariion0_2 emc_clariion0_2 -  4120320  -        -        -       -
dm emc_clariion0_3 emc_clariion0_3 -  4120320  -        -        -       -
```

特定の HBA に接続され、アレイタイプ A/A を持つディスクで、データとログの両方を割り当てるには:

```
# vxassist -g testdg make v1 1G logtype=dco dcoversion=20 ¥
require=hostportid:06-08-02,arraytype:A/A
```

次の出力は上のコマンドの結果を示します。コマンドは、require 制約のすべてのストレージ仕様を満たす emc_clariion0 アレイディスクのデータとログにディスク領域を割り当てます:

```
# vxprint -g testdg
TY NAME         ASSOC        KSTATE   LENGTH   PLOFFS   STATE    TUTIL0  PUTIL0
dg testdg       testdg       -        -        -        -        -       -

dm ams_wms0_359 ams_wms0_359 -        2027264  -        -        -       -
dm ams_wms0_360 ams_wms0_360 -        2027264  -        -        -       -
dm ams_wms0_361 ams_wms0_361 -        2027264  -        -        -       -
dm ams_wms0_362 ams_wms0_362 -        2027264  -        -        -       -
dm emc_clariion0_0 emc_clariion0_0 -  4120320  -        -        -       -
dm emc_clariion0_1 emc_clariion0_1 -  4120320  -        -        -       -
dm emc_clariion0_2 emc_clariion0_2 -  4120320  -        -        -       -
dm emc_clariion0_3 emc_clariion0_3 -  4120320  -        -        -       -

v  v1           fsgen        ENABLED  2097152  -        ACTIVE   -       -
pl v1-01        v1           ENABLED  2097152  -        ACTIVE   -       -
sd emc_clariion0_0-01 v1-01  ENABLED  2097152  0        -        -       -
dc v1_dco       v1           -        -        -        -        -       -
v  v1_dcl       gen          ENABLED  67840    -        ACTIVE   -       -
pl v1_dcl-01    v1_dcl       ENABLED  67840    -        ACTIVE   -       -
sd emc_clariion0_0-02 v1_dcl-01 ENABLED 67840  0        -        -       -
```

例 2 - use 制約

この例は、3 つのアレイ(ams_wms0、emc_clariion0、および hitachi_vsp0)からのディスクを持つディスクグループの use 制約を示します。

次の出力はディスクグループの情報を示します。

```
# vxprint -g testdg
TY NAME         ASSOC        KSTATE   LENGTH   PLOFFS   STATE    TUTIL0  PUTIL0
dg testdg       testdg       -        -        -        -        -       -

dm ams_wms0_359 ams_wms0_359 -        2027264  -        -        -       -
dm ams_wms0_360 ams_wms0_360 -        2027264  -        -        -       -
dm ams_wms0_361 ams_wms0_361 -        2027264  -        -        -       -
dm ams_wms0_362 ams_wms0_362 -        2027264  -        -        -       -
dm emc_clariion0_0 emc_clariion0_0 -  4120320  -        -        -       -
dm hitachi_vsp0_3 hitachi_vsp0_3 -    4120320  -        -        -       -
```

アレイ ams_wms0 またはアレイ emc_clariion0 に属するディスクにデータとログの両方を割り当てるには:

```
# vxassist -g testdg make v1 3G logtype=dco dcoversion=20 ¥
use=array:ams_wms0,array:emc_clariion0
```

次の出力は上のコマンドの結果を示します。コマンドは、use 制約で指定されたアレイを満たすディスクのデータとログにディスク領域を割り当てます。

```
# vxprint -g testdg
TY NAME         ASSOC        KSTATE   LENGTH   PLOFFS   STATE    TUTIL0  PUTIL0
dg testdg       testdg       -        -        -        -        -       -

dm ams_wms0_359 ams_wms0_359 -        2027264  -        -        -       -
dm ams_wms0_360 ams_wms0_360 -        2027264  -        -        -       -
dm ams_wms0_361 ams_wms0_361 -        2027264  -        -        -       -
dm ams_wms0_362 ams_wms0_362 -        2027264  -        -        -       -
dm emc_clariion0_0 emc_clariion0_0 -  4120320  -        -        -       -
dm hitachi_vsp0_3 hitachi_vsp0_3 -    4120320  -        -        -       -

v  v1           fsgen        ENABLED  6291456  -        ACTIVE   -       -
pl v1-01        v1           ENABLED  6291456  -        ACTIVE   -       -
sd ams_wms0_359-01 v1-01     ENABLED  2027264  0        -        -       -
sd ams_wms0_360-01 v1-01     ENABLED  143872   2027264  -        -       -
sd emc_clariion0_0-01 v1-01  ENABLED  4120320  2171136  -        -       -
dc v1_dco       v1           -        -        -        -        -       -
v  v1_dcl       gen          ENABLED  67840    -        ACTIVE   -       -
pl v1_dcl-01    v1_dcl       ENABLED  67840    -        ACTIVE   -       -
sd ams_wms0_360-02 v1_dcl-01 ENABLED  67840    0        -        -       -
```

例 3: datause と logrequire の組み合わせ

この例は、datause 制約と logrequire 制約の組み合わせを示します。ディスクグループは、異なるアレイタイプを持つ 3 つのアレイ(ams_wms0、emc_clariion0、および hitachi_vsp0)からのディスクを持っています。

次の出力はディスクグループの情報を示します。

```
# vxprint -g testdg
 TY NAME         ASSOC        KSTATE   LENGTH   PLOFFS   STATE    TUTIL0  PUTIL0
 dg testdg       testdg       -        -        -        -        -       -
dm ams_wms0_359 ams_wms0_359 -        2027264  -        -        -       -
dm ams_wms0_360 ams_wms0_360 -        2027264  -        -        -       -
dm ams_wms0_361 ams_wms0_361 -        2027264  -        -        -       -
dm ams_wms0_362 ams_wms0_362 -        2027264  -        -        -       -
dm emc_clariion0_0 emc_clariion0_0 -  4120320  -        -        -       -
dm emc_clariion0_1 emc_clariion0_1 -  4120320  -        -        -       -
dm emc_clariion0_2 emc_clariion0_2 -  4120320  -        -        -       -
```

```
dm emc_clariion0_3 emc_clariion0_3 -  4120320  -        -        -       -
dm hitachi_vsp0_3 hitachi_vsp0_3 -    4120320  -        -        -       -
```

ams_wms0 または emc_clariion0 アレイからのディスクにデータを割り当て、アレイタイプ A/A-A からのディスクにログを割り当てるには:

```
# vxassist -g testdg make v1 1G logtype=dco dcoversion=20 ¥
datause=array:ams_wms0,array:emc_clariion0 logrequire=arraytype:A/A-A
```

次の出力は上のコマンドの結果を示します。コマンドはデータとログにそれぞれ独立したディスク領域を割り当てます。データ領域は datause 制約を満たす emc_clariion0 ディスクに割り当てられます。ログ領域は、A/A-Aアレイタイプで logrequire 制約を満たす ams_wms0 ディスクに割り当てられます。

```
# vxprint -g testdg
TY NAME         ASSOC        KSTATE   LENGTH   PLOFFS   STATE    TUTIL0  PUTIL0
dg testdg       testdg       -        -        -        -        -       -
dm ams_wms0_359 ams_wms0_359 -        2027264  -        -        -       -
dm ams_wms0_360 ams_wms0_360 -        2027264  -        -        -       -
dm ams_wms0_361 ams_wms0_361 -        2027264  -        -        -       -
dm ams_wms0_362 ams_wms0_362 -        2027264  -        -        -       -
dm emc_clariion0_0 emc_clariion0_0 -  4120320  -        -        -       -
dm emc_clariion0_1 emc_clariion0_1 -  4120320  -        -        -       -
dm emc_clariion0_2 emc_clariion0_2 -  4120320  -        -        -       -
dm emc_clariion0_3 emc_clariion0_3 -  4120320  -        -        -       -
dm hitachi_vsp0_3 hitachi_vsp0_3 -    4120320  -        -        -       -

v  v1           fsgen        ENABLED  2097152  -        ACTIVE   -       -
pl v1-01        v1           ENABLED  2097152  -        ACTIVE   -       -
sd emc_clariion0_0-01 v1-01  ENABLED  2097152  0        -        -       -
dc v1_dco       v1           -        -        -        -        -       -
v  v1_dcl       gen          ENABLED  67840    -        ACTIVE   -       -
pl v1_dcl-01    v1_dcl       ENABLED  67840    -        ACTIVE   -       -
sd ams_wms0_359-01 v1_dcl-01 ENABLED  67840    0        -        -       -
```

例 4: use と require の組み合わせ

この例は、use 制約と require 制約の組み合わせを示します。ディスクグループは、3 つのアレイ(ams_wms0、emc_clariion0、およびhitachi_vsp0)からのディスクを持っています。ams_wms0 アレイからのディスクのみマルチパスです。

次の出力はディスクグループの情報を示します。

```
# vxprint -g testdg
TY NAME         ASSOC        KSTATE   LENGTH   PLOFFS   STATE    TUTIL0  PUTIL0
dg testdg       testdg       -        -        -        -        -       -
```

```
dm ams_wms0_359 ams_wms0_359 -        2027264  -        -        -       -
dm ams_wms0_360 ams_wms0_360 -        2027264  -        -        -       -
dm ams_wms0_361 ams_wms0_361 -        2027264  -        -        -       -
dm ams_wms0_362 ams_wms0_362 -        2027264  -        -        -       -
dm emc_clariion0_0 emc_clariion0_0 -  4120320  -        -        -       -
dm emc_clariion0_1 emc_clariion0_1 -  4120320  -        -        -       -
dm emc_clariion0_2 emc_clariion0_2 -  4120320  -        -        -       -
dm emc_clariion0_3 emc_clariion0_3 -  4120320  -        -        -       -
dm hitachi_vsp0_3 hitachi_vsp0_3 -    4120320  -        -        -       -
```

emc_clariion0 または ams_wms0 アレイからのディスク、およびマルチパスのディスクにデータ領域とログ領域を割り当てるには:

```
# vxassist -g testdg make v1 1G logtype=dco dcoversion=20 ¥
use=array:emc_clariion0,array:ams_wms0 require=multipathed:yes
```

次の出力は割り当ての結果を示します。データ領域とログ領域は、use 制約と require 制約を満たす ams_wms0 ディスクにあります。

```
# vxprint -g testdg
TY NAME         ASSOC        KSTATE   LENGTH   PLOFFS   STATE    TUTIL0  PUTIL0
dg testdg       testdg       -        -        -        -        -       -
dm ams_wms0_359 ams_wms0_359 -        2027264  -        -        -       -
dm ams_wms0_360 ams_wms0_360 -        2027264  -        -        -       -
dm ams_wms0_361 ams_wms0_361 -        2027264  -        -        -       -
dm ams_wms0_362 ams_wms0_362 -        2027264  -        -        -       -
dm emc_clariion0_0 emc_clariion0_0 -  4120320  -        -        -       -
dm emc_clariion0_1 emc_clariion0_1 -  4120320  -        -        -       -
dm emc_clariion0_2 emc_clariion0_2 -  4120320  -        -        -       -
dm emc_clariion0_3 emc_clariion0_3 -  4120320  -        -        -       -
dm hitachi_vsp0_3 hitachi_vsp0_3 -    4120320  -        -        -       -
v  v1           fsgen        ENABLED  2097152  -        ACTIVE   -       -
pl v1-01        v1           ENABLED  2097152  -        ACTIVE   -       -
sd ams_wms0_359-01 v1-01     ENABLED  2027264  0        -        -       -
sd ams_wms0_360-01 v1-01     ENABLED  69888    2027264  -        -       -
dc v1_dco       v1           -        -        -        -        -       -
v  v1_dcl       gen          ENABLED  67840    -        ACTIVE   -       -
pl v1_dcl-01    v1_dcl       ENABLED  67840    -        ACTIVE   -       -
sd ams_wms0_360-02 v1_dcl-01 ENABLED  67840    0        -        -       -
```

永続的な属性の use および require タイプの管理

永続的な属性は、そのボリュームための後続の割り当て操作で使われることになる、保存されたボリュームインテントです。インテント管理操作は、ボリュームの永続的なインテント

の use および require タイプの管理を可能にします。これらの操作は、ボリューム作成後のインテントの独立した管理を可能にします。ボリュームの永続的なインテントを変更する場合は、変更されたインテントは有効性を確認されず、ボリュームの現在の割り当てを強制されません。

次の vxassist 操作を使ってボリュームの永続的なインテントを設定、変更、消去、または一覧表示できます。

- setrule
  既存の保存されたインテントを、指定されたボリュームの指定されたインテントに置き換えます。
- changerule
  指定されたインテントを指定されたボリュームの既存の保存されたインテントに追加します。
- clearrule
  指定されたボリュームの既存の保存されたインテントを削除します。
- listrule
  指定されたボリュームの保存されたインテントを一覧表示します。ボリューム名が指定されていない場合、このコマンドはすべてのボリュームのインテントを表示します。

インテント管理操作は、永続的な制約の use または require タイプにのみ適用されます。永続的な制約の他のタイプは、persist 属性によって管理されます。

p.224 の「永続的な属性の使用」を参照してください。

ボリュームに現在関連付けられているインテントを表示するには

◆ ボリュームに現在関連付けられているインテントを表示するには、次のコマンドを使います。

```
# vxassist [options] listrule [volume]
```

たとえば、ボリューム v1 の既存の保存されたインテントを表示するには、次のようにします。

```
# vxassist -g testdg listrule v1
volume rule v1 {
     require=array:ams_wms0
}
```

ボリュームに現在関連付けられているインテントを置き換えるには

1 ボリュームに現在関連付けられているインテントを表示します。

```
# vxassist [options] listrule [volume]
```

この例では、アレイが ams_wms0 であるように要求する既存の保存されたインテントがボリューム v1 にあります。たとえば、ボリューム v1 の既存の保存されたインテントを表示するには、次のようにします。

```
# vxassist -g testdg listrule v1
volume rule v1 {
     require=array:ams_wms0
}
```

2 次のコマンドを使って、新しいインテントを指定します。

```
# vxassist [options] setrule volumeattributes...
```

たとえば、アレイを ds4100-0 アレイと置き換えるには、次のコマンドを使って新しいインテントを指定します。

```
# vxassist -g testdg setrule v1 require=array:ds4100-0
```

3 表示コマンドを使って新しいインテントを検証します。

たとえば、次のコマンドはインテントが変更されたことを示します。

```
# vxassist -g testdg listrule v1
volume rule v1 {
     require=array:ds4100-0
}
```

ボリュームに現在関連付けられているインテントに追加するには

1 ボリュームに現在関連付けられているインテントを表示します。

```
# vxassist [options] listrule [volume]
```

この例では、アレイが ds4100-0 であるように要求する既存の保存されたインテントがボリューム v1 にあります。たとえば、ボリューム v1 の既存の保存されたインテントを表示するには、次のようにします。

```
# vxassist -g testdg listrule v1
volume rule v1 {
     use=array:ds4100-0
}
```

2 次のコマンドを使って、新しいインテントを追加します。

```
# vxassist [options] changerule volumeattributes...
```

たとえば、use 制約に ams_wms0 アレイを追加するには、次のコマンドを使って新しいインテントを指定します。

```
# vxassist -g testdg changerule v1 use=array:ams_wms0
```

3 表示コマンドを使って新しいインテントを検証します。

たとえば、次のコマンドはインテントが変更されたことを示します。

```
# vxassist -g testdg listrule v1
volume rule v1 {
     use=array:ds4100-0,array:ams_wms0
}
```

ボリュームに現在関連付けられているインテントをクリアするには

1 ボリュームに現在関連付けられているインテントを表示します。

```
# vxassist [options] listrule [volume]
```

たとえば、ボリューム v1 の既存の保存されたインテントを表示するには、次のようにします。

```
# vxassist -g testdg listrule v1
volume rule v1 {
     require=multipathed:yes
     use=array:emc_clariion0,array:ams_wms0
}
```

2 次のコマンドを使って、既存のインテントをクリアします。

```
# vxassist [options] clearrule volume
```

たとえば、ボリューム v1 のインテントをクリアするには、次のようにします。

```
# vxassist -g testdg clearrule v1
```

3 ボリュームに保存されたインテントがないこと確認します。

たとえば、次のコマンドはボリューム v1 に保存されたインテントがないことを示します。

```
# vxassist -g testdg listrule v1
volume rule v1 {}
```

特定のレイアウトのボリュームの作成

VxVM（Veritas Volume Manager）を使用して、さまざまなレイアウトのボリュームの作成が可能になります。作成したいレイアウトタイプを示すために属性を指定できます。次のセクションでは、それぞれのタイプについて説明します。

- ミラーボリューム
  p.242 の「ミラーボリュームの作成」を参照してください。
- ストライプボリューム
  p.244 の「ストライプボリュームの作成」を参照してください。
- RAID 5 ボリューム
  p.246 の「RAID 5 ボリュームの作成」を参照してください。

ボリュームレイアウトのタイプ

VxVM（Veritas Volume Manager）では、異なるレイアウトタイプでボリュームを作成できます。表 8-3 では、VxVM ボリュームのレイアウトタイプについて説明します。

表 8-3　ボリュームレイアウトのタイプ

レイアウトタイプ	説明
連結	サブディスクがプレックス内で順番に連続して配列されているボリューム。1 つのディスク領域ではボリュームのための領域が足りない場合、連結により 1 つまたは複数のディスクの複数の領域からボリュームを作成できます。単一の LUN またはディスクが複数のサブディスクに分割され、各サブディスクが重複のないボリュームに属している場合は、これをカービングと呼びます。 p.65 の「連結、分散、およびカービング」を参照してください。
ストライプ	データを複数のディスクに均等に分散したボリューム。ストライプは、1 つのプレックスの複数のサブディスクに交互に均等に割り当てられる同じサイズの断片です。ストライプ化プレックスには少なくとも 2 つのサブディスクが含まれ、それぞれが別のディスクに存在する必要があります。ストライプ化プレックスに使うディスク数が多いほど、スループットが向上します。ストライプ化は、特定のサブディスクにトラフィック量が高い領域が存在する場合に、I/O 負荷を分散させる役割も持っています。 p.67 の「ストライプ化（RAID 0）」を参照してください。
ミラー	ボリュームに含まれる情報を複製する複数のデータプレックスを持つボリューム。ボリューム内のデータプレックスは 1 つでもかまいませんが、真の意味でのミラー化を実現し、データの冗長性を確保するには、少なくとも 2 つのデータプレックスが必要です。冗長性を有効に利用するには、各データプレックスが異なるディスクのディスク領域で構成されている必要があります。 p.71 の「ミラー化（RAID 1）」を参照してください。
RAID-5	ストライプ化を使ってアレイ内の複数のディスクにデータとパリティを均等に分散したボリューム。各ストライプには、パリティストライプユニットとデータストライプユニットが含まれています。ディスクの 1 つに障害が発生しても、パリティを使ってデータを復元することができます。データ変更が発生するたびにパリティ情報を更新する必要があるため、RAID 5 ボリュームの書き込みスループットはストライプボリュームの処理効率に比べて低くなります。ただし、データの冗長性の実装にパリティを使うため、ミラーボリュームに比べて必要なディスク領域は少なくなります。 p.73 の「RAID 5（パリティ付きストライプ化）」を参照してください。

レイアウトタイプ	説明
ミラー化ストライプ	ストライプ化プレックスと、それをミラー化した別のプレックスで設定されるボリューム。このレイアウトでは、少なくとも 2 つのストライプ化用ディスクと、1 つまたは複数のミラー化用ディスク(プレックスが単一プレックスかストライプ化プレックスかによる)が必要です。このレイアウトの利点は、データを複数のディスクに分散させることにより処理効率が向上し、かつデータの冗長性が確保されることです。 p.71 の「ストライプ化 + ミラー化(ミラー化ストライプ、RAID 0+1)」を参照してください。
階層化ボリューム	別のボリュームで構成されるボリューム。非階層化ボリュームは、サブディスクを VxVM ディスクにマッピングすることによって構成されます。階層化ボリュームは、サブディスクをより下位のボリューム(ストレージボリューム)にマッピングすることによって、より複雑な形式の論理レイアウトを構成できます。 p.78 の「階層化ボリュームについて」を参照してください。 次のレイアウトは階層化ボリュームの例です。 ■ ストライプ化ミラー ストライプ化ミラーボリュームは、複数のミラーボリュームをストライプボリュームのカラムとして設定することによって作成されます。このレイアウトには、非階層化ミラー化ストライプボリュームと同じ利点があります。さらに、このレイアウトでは、1 つのディスクに障害が発生してもストライプ化プレックス全体がオフラインになるわけではないため、リカバリ時間が短縮されます。 p.72 の「ミラー化 + ストライプ化(ストライプ化ミラー、RAID 1+0 または RAID 10)」を参照してください。 ■ 連結ミラー 連結ミラーボリュームは、複数のミラーボリュームを連結して作成されます。このレイアウトでは、1 つのディスクに障害が発生してもミラー全体がオフラインになるわけではないため、リカバリ時間が短縮されます。

ミラーボリュームの作成

ミラーボリュームは、複数のデータコピーを保持することにより、データの冗長性を確保します。それぞれのコピー(すなわちミラー)は、ボリュームの元のコピーや他のミラーとは別のディスクに保管されます。ボリュームをミラー化しておくと、コンポーネントミラー内のいずれかのディスクに障害が発生しても、データは消失しません。

ミラーボリュームを作成するには、ディスクグループにおいて、少なくともボリューム内のミラー数と同じ数のディスク上に使用可能な領域が存在する必要があります。

layout=mirror指定すれば、ミラー化されたボリュームのための最もよいレイアウトを判断しますvxassist。レイアウトの利点がボリュームのサイズと関連しているので、vxassist はボリュームのサイズに基づいてレイアウトを選択します。より小さいボリュームの場合、vxassist は、より単純なミラー化された連結（ミラー化連結）のレイアウトを使います。より大きいボリュームの場合、vxassist は、より複雑な連結されたミラー（連結ミラー）のレイアウトを使います。属性 stripe-mirror-col-split-trigger-pt によって、この選択が制御されます。stripe-mirror-col-split-trigger-pt よりも小さいボリュームはミラー化連結として作成され、それよりも大きいボリュームは連結ミラーとして作成されます。デフォルトでは、属性 stripe-mirror-col-split-trigger-pt は 1 GB に設定されています。この値は、/etc/default/vxassist で設定できます。特定のレイアウトを実装する理由があれば、望ましいレイアウトを実装するために layout=mirror-concat か layout=concat-mirror を指定できます。

新しいミラーボリュームを作成するには

◆ 新しいミラーボリュームを作成するには、次のコマンドを使います。

```
# vxassist [-b] [-g diskgroup] make volumelength ¥
  layout=mirror [nmirror=number] [init=active]
```

ボリュームをすぐに使用可能にする必要がある場合は、-b オプションを指定します。

たとえば、ディスクグループ mydg にミラーボリューム volmir を作成するには、次のコマンドを使います。

```
# vxassist -b -g mydg make volmir 5g layout=mirror
```

次の例に、デフォルトの 2 つのミラーの代わりに、3 つのミラーがあるボリュームを作成する方法を示します。

```
# vxassist -b -g mydg make volmir 5g layout=mirror nmirror=3
```

ミラー化連結ボリュームの作成

ミラー化連結ボリュームは、複数のコンカチネイテッドプレックスをミラー化します。

ミラー化連結ボリュームを作成するには

◆ 次のコマンドを使って、ミラー化連結されたボリュームとしてボリュームを作成します。

```
# vxassist [-b] [-g diskgroup] make volumelength ¥
  layout=mirror-concat [nmirror=number]
```

ボリュームをすぐに使用可能にする必要がある場合は、-b オプションを指定します。

または、まず連結ボリュームを作成し、次にそれをミラー化することもできます。

p.972 の「ボリュームへのミラーの追加」を参照してください。

連結ミラーボリュームの作成

連結ミラーボリュームは、下位の複数のミラーボリュームを結合した階層化ボリュームの一種です。

連結ミラーボリュームを作成するには

- 連結ミラーボリュームを作成するには、次のコマンドを使います。

```
# vxassist [-b] [-g diskgroup] make volume length ¥
  layout=concat-mirror [nmirror=number]
```

ボリュームをすぐに使用可能にする必要がある場合は、-b オプションを指定します。

ストライプボリュームの作成

ストライプボリュームは、2 台以上の物理ディスク上に配置された 2 つ以上のサブディスクからなる 1 つ以上のプレックスで構成されます。ストライプボリュームを作成するには、ディスクグループにおいて、少なくともボリューム内のカラム数と同じ数のディスク上に使用可能な領域が存在する必要があります。

p.67 の「ストライプ化（RAID 0）」を参照してください。

ストライプボリュームを作成するには、次のコマンドを使います。

```
# vxassist [-b] [-g diskgroup] make volume length layout=stripe
```

ボリュームをすぐに使用可能にする必要がある場合は、-b オプションを指定します。

たとえば、ディスクグループ mydg に 10 GB のストライプボリューム volzebra を作成するには、次のコマンドを使います。

```
# vxassist -b -g mydg make volzebra 10g layout=stripe
```

このコマンドを実行すると、デフォルトのストライプユニットサイズ（64 KB）とデフォルトのストライプ数（2）でストライプボリュームが作成されます。

コマンドラインにディスク名を挿入して、ボリュームを作成するディスクを指定できます。たとえば、3 つの特定のディスク mydg03、mydg04、mydg05 上に 30 GB のストライプボリュームを作成するには、次のコマンドを使います。

```
# vxassist -b -g mydg make stripevol 30g layout=stripe ¥
  mydg03 mydg04 mydg05
```

カラム数を変更する場合やストライプユニットサイズ（ストライプ幅）を変更する場合は、修飾子 ncolumn または stripeunit を指定して vxassist を実行します。たとえば、次のコマンドを実行すると、カラム数が 5 でストライプサイズが 32 KB のストライプボリュームが作成されます。

```
# vxassist -b -g mydg make stripevol 30g layout=stripe ¥
  stripeunit=32k ncol=5
```

ミラー化ストライプボリュームの作成

ミラー化ストライプボリュームは、複数のストライプデータプレックスをミラー化したものです。ミラー化ストライプボリュームを作成するには、ディスクグループにおいて、少なくともボリューム内のミラー数にカラム数をかけたものと同数のディスク上に使用可能な領域が存在する必要があります。

ストライプ化ミラーボリュームを作成するには、次のコマンドを使います。

```
# vxassist [-b] [-g diskgroup] make volume length ¥
  layout=mirror-stripe [nmirror=number_of_mirrors] ¥
  [ncol=number_of_columns] [stripewidth=size]
```

ボリュームをすぐに使用可能にする必要がある場合は、-b オプションを指定します。

または、まずストライプボリュームを作成し、次にそれをミラー化することもできます。その場合、追加のデータプレックスは、ストライプ化プレックスまたはコンカチネイテッドプレックスのいずれかになります。

p.972 の「ボリュームへのミラーの追加」を参照してください。

ストライプ化ミラーボリュームの作成

ストライプ化ミラーボリュームは、下位の複数のミラーボリュームをストライプ化した階層化ボリュームの一種です。ストライプ化ミラーボリュームを作成するには、ディスクグループにおいて、少なくともボリューム内のミラー数にカラム数をかけたものと同数のディスク上に使用可能な領域が存在する必要があります。

ストライプ化ミラーボリュームを作成するには、次のコマンドを使います。

```
# vxassist [-b] [-g diskgroup] make volume length ¥
  layout=stripe-mirror [nmirror=number_of_mirrors] ¥
  [ncol=number_of_columns] [stripewidth=size]
```

ボリュームをすぐに使用可能にする必要がある場合は、-b オプションを指定します。

各カラムのサイズが vxassist デフォルトファイルで定義された stripe-mirror-col-split-trigger-pt 属性の値よりも大きい場合、VxVM（Veritas Volume Manager）は、デフォルトでは、カラムではなくサブディスク単位でミラー化して下位のボリュームを作成しようとします。

1 つのカラムに複数のサブディスクが存在する場合は、各カラムの代わりに各サブディスクを個別にミラー化できます。サブディスク単位でのミラー化を明示的に指定するには、レイアウトを stripe-mirror ではなく、stripe-mirror-sd と指定します。一方、カラ

ム単位でのミラー化を明示的に指定するには、レイアウトを stripe-mirror ではなく、stripe-mirror-col と指定します。

RAID 5 ボリュームの作成

RAID 5 ボリュームを作成するには、ディスクグループにおいて、少なくともボリューム内のカラム数と同じ数のディスク上に使用可能な領域が存在する必要があります。RAID 5 ログを作成する場合は、それ以上のディスクが必要になります。

メモ: VxVM（Veritas Volume Manager）では、専用ディスクグループでの RAID 5 ボリュームの作成をサポートしていますが、クラスタ環境の共有ディスクグループでの作成はサポートしていません。

---

vxassist コマンド（推奨）または vxmake コマンドを使って RAID 5 ボリュームを作成できます。ここでは、優先される方法、vxassist コマンドを使う場合について説明します。

vxmake コマンドの使用について詳しくは、vxmake(1M) のマニュアルページを参照してください。

RAID 5 ボリュームには、3 台以上の物理ディスクに配置されている 3 つ以上のサブディスクで構成された RAID 5 データプレックスがあります。各ボリュームの RAID 5 データプレックスは 1 つのみです。RAID 5 ボリュームには、ボリュームに書き込まれるデータとパリティに関するログ情報を作成する 1 つまたは複数の RAID 5 ログプレックスも作成することができます。

p.73 の「RAID 5（パリティ付きストライプ化）」を参照してください。

警告: 複数のディスクで障害が発生するとボリュームがリカバリ不能になるため、8 個を超えるカラムを持つ RAID 5 ボリュームは作成しないでください。

RAID 5 ボリュームを作成するには、次のコマンドを使います。

```
# vxassist [-b] [-g diskgroup] make volumelength layout=raid5 ¥
  [ncol=number_of_columns] [stripewidth=size] [nlog=number] ¥
  [loglen=log_length]
```

ボリュームをすぐに使用可能にする必要がある場合は、-b オプションを指定します。

たとえば、ディスクグループ mydg に 2 つの RAID 5 ログを持つ RAID 5 ボリューム volraid を作成するには、次のコマンドを使います。

```
# vxassist -b -g mydg make volraid 10g layout=raid5 nlog=2
```

これにより、デフォルトのストライプユニットサイズの RAID 5 ボリュームがデフォルト数のディスク上に作成されます。さらに、デフォルトの 1 つのログではなく、2 つの RAID 5 ログが作成されます。

RAID 5 ログにディスクを割り当てる必要がある場合、ログプレックスで使うディスクを指定するには、logdisk 属性を使う必要があります。

RAID 5 ログはコンカチネイテッドプレックスやストライプ化プレックスにすることができ、RAID 5 ボリュームに関連付けられた各 RAID 5 ログには、ボリュームのログ情報の全コピーが含まれます。RAID 5 アレイへの同時アクセスをサポートするには、ログのサイズは、RAID 5 プレックスのストライプサイズよりも数倍大きくする必要があります。

すなわち、RAID 5 ボリュームごとに少なくとも 2 つの RAID 5 ログプレックスを設定する必要があります。これらのログプレックスは、それぞれ別のディスクに配置します。各 RAID 5 ボリュームに 2 つのログプレックスを設定することにより、1 つのディスクに障害が発生してもログ情報の消失を防止できます。

特定のストレージに RAID 5 ボリュームを作成するときに、順次ディスク割り当てを使う場合、オプションの logdisk 属性を使って、RAID 5 ログプレックスを作成するディスクを指定できます。ログの領域を割り当てるディスクを指定するには、次の形式の vxassist コマンドを使います。

```
# vxassist [-b] [-g diskgroup] -o ordered make volumelength ¥
  layout=raid5 [ncol=number_columns] [nlog=number] ¥
  [loglen=log_length] logdisk=disk[,disk,...] ¥
  storage_attributes
```

たとえば、次のコマンドは、カラム数が 3 でデフォルトのストライプユニットサイズの RAID 5 ボリュームを mydg04、mydg05 および mydg06 に作成します。さらに、2 つの RAID 5 ログをディスク mydg07 と mydg08 上に作成します。

```
# vxassist -b -g mydg -o ordered make volraid 10g layout=raid5 ¥
  ncol=3 nlog=2 logdisk=mydg07,mydg08 mydg04 mydg05 mydg06
```

ログの数は、logdisk に指定したディスク数と一致する必要があります。

p.253 の「ボリュームのストレージに対する順次ディスク割り当て」を参照してください。

vxassist（1M）マニュアルページを参照してください。

後から RAID 5 ボリュームにログを追加できます。

RAID 5 ログを既存のボリュームに追加するには、次のコマンドを使います。

```
# vxassist [-b] [-g diskgroup] addlog volume [loglen=length]
```

-b オプションを指定すると、新しいログの追加はバックグラウンドタスクになります。

ボリュームに初めてログを追加する場合は、ログのサイズを指定できます。その後に追加するログは、既存のログと同じサイズで設定されます。

たとえば、ディスクグループ mydg 内の RAID 5 ボリューム volraid にログプレックスを 1 つ作成する場合は、次のコマンドを使います。

```
# vxassist -g mydg addlog volraid
```

指定したディスクにおけるボリュームの作成

VxVM（Veritas Volume Manager）は、ユーザーが別途指定しないかぎり、各ボリュームを作成するディスクを自動的に選択します。特定のボリュームに対して特定のタイプのディスクを選択する場合は、ストレージの割り当てに関するストレージ指定を vxassist に提供できます。

詳しくは、vxassist（1M）マニュアルページのストレージ指定のセクションを参照してください。

p.226 の「割り当て用のディスククラスのカスタマイズ」を参照してください。

p.228 の「use 節と require 節を使った vxassist 操作のための割り当て制約の指定」を参照してください。

指定したディスク上にボリュームを作成する場合は、それらのディスクを VxVM に指定する必要があります。このとき、複数のディスクを指定することができます。

指定したディスクにボリュームを作成するには、次のコマンドを使います。

```
# vxassist [-b] [-g diskgroup] make volumelength ¥
  [layout=layout] diskname ...
```

ボリュームをすぐに使用可能にする必要がある場合は、-b オプションを指定します。

たとえば、ディスク mydg03 と mydg04 上に 5 GB のボリューム volspec を作成するには、次のコマンドを使います。

```
# vxassist -b -g mydg make volspec 5g mydg03 mydg04
```

vxassist コマンドでは、ストレージ属性を指定できます。ストレージ属性を指定することにより、vxassist がボリュームの設定に使うディスクやコントローラなどのデバイスを制御できます。

たとえば、次のように指定してディスク mydg05 を除外することができます。

メモ: ! 文字は一部のシェルでの特殊文字です。次の例では、bash シェルでこの文字をエスケープ処理する方法を示します。

```
# vxassist -b -g mydg make volspec 5g ¥!mydg05
```

次の例では、c2 コントローラ上のすべてのディスクが除外されます。

```
# vxassist -b -g mydg make volspec 5g ¥!ctlr:c2
```

特定のディスクグループのディスクのみを使ってボリュームを作成する場合は、次の例のように vxassist に -g オプションを指定します。

```
# vxassist -g bigone -b make volmega 20g bigone10 bigone11
```

または、diskgroup 属性を使うこともできます。

```
# vxassist -b make volmega 20g diskgroup=bigone bigone10 ¥
   bigone11
```

ストレージ属性で指定されるディスクは、ディスクグループに属している必要があります。ディスクグループに属していない場合、vxassist はその指定されたディスクを使わずにボリュームを作成します。

ストレージ属性を使って、ボリュームの最大サイズを計算したり、ボリュームを拡張したり、ボリュームからミラーやログを削除する場合などに vxassist が使うディスクを制御することもできます。次の例では、vxassist がディスクグループ mydg 内のディスクを使って作成できる RAID 5 ボリュームの最大サイズを計算するときに、ディスク mydg07 と mydg08 が除外されます。

```
# vxassist -b -g mydg maxsize layout=raid5 nlog=2 ¥!mydg07 ¥!mydg08
```

指定したストレージ上のボリュームのレイアウトも制御できます。

p.253 の「ボリュームのストレージに対する順次ディスク割り当て」を参照してください。

vxassist では、ディスクタグに基づいてディスクを選択することもできます。次のコマンドは tier1 というディスクタグがあるディスクのみを含めます。

```
# vxassist -g mydg make vol3 1g disktag:tier1
```

vxassist(1M)マニュアルページを参照してください。

特定のメディアタイプのボリュームの作成

ボリュームを作成するとき、ボリュームのメディアタイプを指定できます。サポートされているメディアタイプは、HDD(ハードディスクドライブ)またはSSD(ソリッドステートデバイス)です。SSD メディアタイプは、ディスクグループ 150 以上を必要とします。デフォルトは HDD です。

メディアタイプを指定するには、mediatype 属性で vxassist コマンドを指定します。mediatype が指定されていない場合、ボリュームでは HDD デバイスからのみストレージを割り当てます。

暗号化ボリュームの作成

vxassist を使って属性 encrypted を on に設定し、暗号化ボリュームを作成します。

ボリュームをパスワードまたはパスフレーズで暗号化した場合、VxVM ではボリュームに対し「encrypted with password」というボリュームステータスが表示されます。Key Management Server が設定されており、それを使用してボリュームを暗号化した場合、VxVM ではボリュームに対し「encrypted」というボリュームステータスが表示されます。

以下に、Key Management Server を使ったボリュームの暗号化の例を示します。

```
# vxassist -g mydg make vol01 1g encrypted=on
# vxassist -g mydg make vol02 1g
# vxencrypt list
Disk group: mydg

 VOLUME       STATUS
 vol01         encrypted
 vol02         not encrypted
```

暗号化パスワードの変更

暗号化パスワードはいつでも vxencrypt コマンドを使って変更します。

```
# vxencrypt -g mydg passwd vol01
Enter current password: xxxx
Enter new password: xxxx
Confirm new password: xxxxx
```

キーの再設定操作を使った KEK の変更

InfoScale には、必要に応じて、ボリュームのキーを再設定して KMS キーを変更するオプションが用意されています。このオプションは、キーローテーションとも呼ばれます。キーの再実行操作をスケジュールするために、ポリシーに基づいて外部スケジューラを使うことができます。

- 1 つのボリュームでは、vxencrypt コマンドを使用してキー暗号化キーをいつでも変更できます。

```
vxencrypt -g DiskGroupName rekey volumeName
```

何らかの理由によってボリュームのキーの再設定操作が失敗した場合は、vxencrypt コマンドを再度実行します。vxencrypt コマンドを実行すると、InfoScale は暗号化されたボリュームの新しい識別子の KMS を要求します。

暗号化ボリュームの表示

vxencrypt コマンドを使って、暗号化ボリュームの一覧を表示します。

```
# vxencrypt list
Disk group: mydg
        VOLUME       STATUS
        vol01         encrypted
        vol02         encrypted

Disk group: mydg1
        VOLUME       STATUS
        vol03         encrypted
        vol04         not encrypted

# vxencrypt -g mydg1 list
        VOLUME       STATUS
        vol03         encrypted
        vol04         not encrypted
```

暗号化ボリュームの自動スタートアップ

デフォルトで、暗号化ボリュームはシステムのブート時に自動的に起動できません。ユーザーがアクセスパスフレーズを入力する必要があるためです。ただし、/etc/vx/encryption/password_file ファイル内に必要なパスフレーズを保存し、暗号化されたボリュームの起動を自動化することができます。

注意: パスワードファイルは、ディスクに保存されます。権限のないユーザーにファイルが読み取られないように、セキュアファイル権限を設定します。ファイルがあるホストおよびストレージへの物理アクセスも保護する必要があります。

パスフレーズファイルには、暗号化ボリュームごとに 1 行が含まれています。各行には、次の情報がテキストの 3 列に含まれます。

最初の列	ディスクグループの名前またはディスクグループ ID * ワイルドカード文字を使って、ディスクグループを示します。
2 番目の列	暗号化ボリュームの名前 * ワイルドカード文字を使って、ボリュームを示します。

3 番目の列	パスフレーズ パスフレーズは、password_fileファイル内でテキスト形式で指定する必要があります。 VxVM では パスフレーズのサイズまたはパスフレーズの文字数には制限がありませんが、パスフレーズには改行または NULL 文字は含まれてはいけません。

システムスタートアップ時に、VxVM は暗号化ボリュームに対しパスフレーズを問い合わせます。ボリュームがファイルに一覧表示されている場合、ユーザーに手動入力を求める代わりに、ボリュームの対応するパスフレーズが使用されます。

次に、サンプルのパスフレーズファイルを示します。

```
datadg1    datavol1   mypassphrase1
datadg1    datavol2   mypassphrase2
```

Key Management Server の設定

Key Management Server をボリューム暗号化用に設定するには、設定ファイル /etc/vx/enc-kms-kmip.conf を KMIP クライアント上に作成します。

設定ファイルには、次の情報が含まれている必要があります。

host	Key Management Server のホスト名または IP アドレス
port	Key Management Server が Key Management Interoperability Protocol (KMIP) クライアントを受け入れるポート番号
keyfile	KMIP クライアントによって使用されるプライベートキーの場所 (PEM 形式)
certfile	KMIP クライアントによって使用される証明書の場所 (PEM 形式)
cacerts	相互認証に使用されるルート証明書の場所 (PEM 形式)
ssl_version	KMIP クライアントによって使用される SSL バージョン。

次に、設定ファイルの例を示します。

```
[client]
host = kms-enterprise.example.com
port = 5696
keyfile= /etc/vx/client-key.pem
certfile= /etc/vx/client-crt.pem
cacerts= /etc/vx/cacert.pem
ssl_version = PROTOCOL_TLSv1
```

ボリュームのストレージに対する順次ディスク割り当て

順次ディスク割り当てを行うと、領域の割り当てを完全に制御できます。この操作を実行する場合、vxassist コマンドに指定するディスク数は、ボリュームの作成に必要なディスク数と一致させる必要があります。また、vxassist に指定するディスクの順番も重要です。

ボリューム作成時に、-o ordered オプションを vxassist に指定すると、指定したストレージが次の順序で割り当てられます。

- ディスクの連結
- カラムの形成
- ミラーの形成

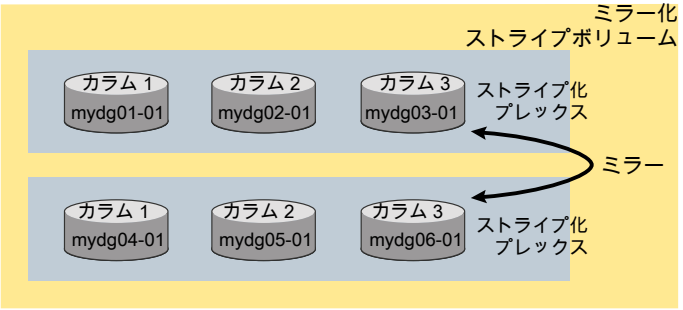
たとえば、次のコマンドを実行すると、ディスクグループ mydg 内の 6 つのディスク上に 3 つのカラムと 2 つのミラーを持つミラー化ストライプボリュームが作成されます。

```
# vxassist -b -g mydg -o ordered make mirstrvol 10g ¥
  layout=mirror-stripe ncol=3 mydg01 mydg02 mydg03 ¥
  mydg04 mydg05 mydg06
```

このコマンドは、最初のミラーのカラム 1、2、3 をディスク mydg01、mydg02、mydg03 に、2 番目のミラーのカラム 1、2、3 をディスク mydg04、mydg05、mydg06 にそれぞれ配置します。

図 8-1 に、順次ディスク割り当てを使ったミラー化ストライプボリュームの作成例を示します。

図 8-1　順次ディスク割り当てを使ったミラー化ストライプボリュームの作成例



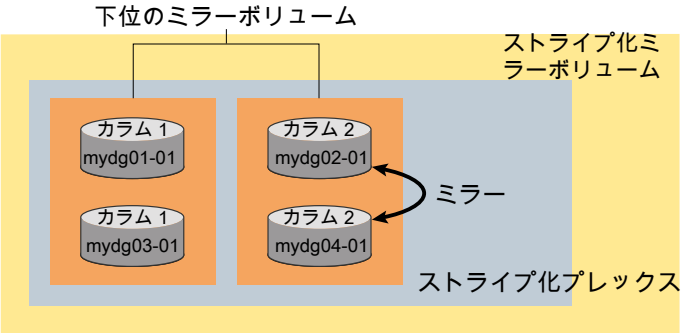
階層化ボリュームに対しても、vxassist は、非階層化ボリュームと同じルールを適用してストレージを割り当てます。たとえば、次のコマンドは、2 つのカラムを持つストライプ化ミラーボリュームを作成します。

```
# vxassist -b -g mydg -o ordered make strmirvol 10g ¥
  layout=stripe-mirror ncol=2 mydg01 mydg02 mydg03 mydg04
```

このコマンドは、カラム 1 のミラーをディスク mydg01 と mydg03 をまたがって作成し、カラム 2 のミラーをディスク mydg02 と mydg04 をまたがって作成します。

図 8-2 に、順次ディスク割り当てを使ったストライプ化ミラーボリュームの作成例を示します。

図 8-2　順次ディスク割り当てを使ったストライプ化ミラーボリュームの作成例



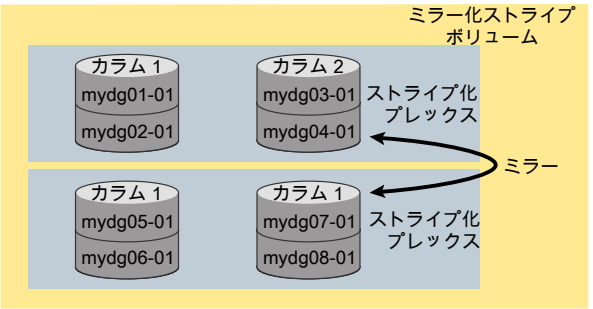
さらに、col_switch 属性を使って、ディスク上の領域をカラムに連結する方法を指定できます。たとえば、次のコマンドは、2 つのカラムを持つミラー化ストライプボリュームを作成します。

```
# vxassist -b -g mydg -o ordered make strmir2vol 10g ¥
  layout=mirror-stripe ncol=2 col_switch=3g,2g ¥
  mydg01 mydg02 mydg03 mydg04 mydg05 mydg06 mydg07 mydg08
```

このコマンドは、mydg01 の 3 ギガバイトと mydg02 の 2 ギガバイトをカラム 1 に割り当て、mydg03 の 3 ギガバイトと mydg04 の 2 ギガバイトをカラム 2 に割り当てます。これらのカラムのミラーも同じように mydg05 から mydg08 までに形成されます。

図 8-3 に、連結ディスク領域を使ったミラー化ストライプボリュームの作成例を示します。

図 8-3　　連結ディスク領域を使ったミラー化ストライプボリュームの作成例



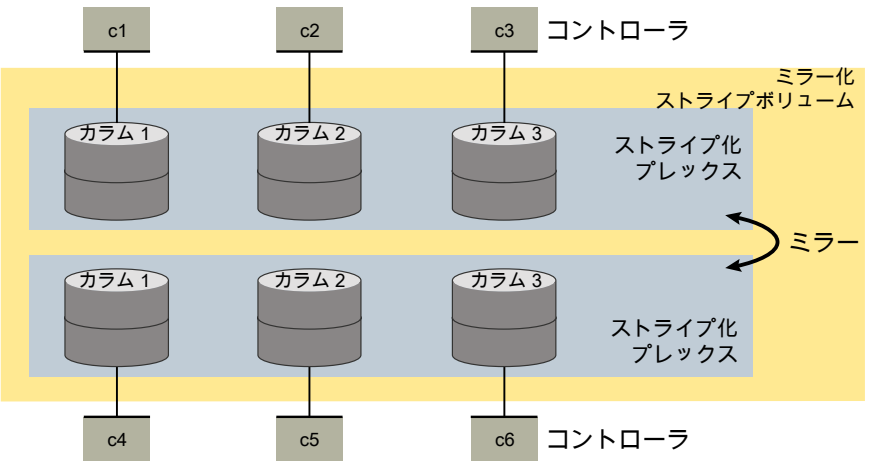
コントローラ、エンクロージャ、ターゲットおよびトレイ用の他のストレージ指定クラスも順次ディスク割り当てと同時に使えます。たとえば、次のコマンドは、指定のコントローラ間に 3 つのカラムを持つミラー化ストライプボリュームを作成します。

```
# vxassist -b -g mydg -o ordered make mirstr2vol 80g ¥
  layout=mirror-stripe ncol=3 ¥
  ctlr:c1 ctlr:c2 ctlr:c3 ctlr:c4 ctlr:c5 ctlr:c6
```

このコマンドは、コントローラ `c1` 上のディスクの領域をカラム 1 に、コントローラ `c2` 上のディスクの領域をカラム 2 に、というように割り当てていきます。

図 8-4 に、ストレージ指定属性を使った、複数のコントローラ間でのミラー化ストライプボリュームの作成例を示します。

図 8-4　　ストレージ指定属性を使った、複数のコントローラ間でのミラー化ストライプボリュームの作成例



別の方法でも、複数のコントローラ間でのミラーボリュームのレイアウトに関して vxassist を制御できます。

サイトベースの割り当て

リモートミラー設定（別名キャンパスクラスタまたはストレッチクラスタ）では、クラスタのホストとストレージが 2 つ以上のサイトに分割されます。通常、これらのサイトには、ストレージへのアクセスとクラスタノード間でのプライベートリンク通信を提供する、冗長高キャパシティネットワークを介して接続します。

リモートミラーサイトのディスクグループをサイトの一貫性があるように設定します。このようなディスクグループにボリュームを作成すると、デフォルトで、ボリュームはすべてのサイトにミラー化されます。

p.588 の「サイトとリモートミラーについて」を参照してください。

ミラーボリュームの読み取りポリシーの変更

ミラー化ボリュームに対して、Veritas Volume Manager（VxVM）は読み取りポリシーを使用してボリューム内のどのデータプレックスを読み取りに使用するかを決定します。デフォルトでは、VxVM は次の基準を次の順序で使用してプレックスを選択します:

- サイト
  同じサイト上のプレックスが他のサイト上のプレックスよりも優先的に選択されます。
- 接続性
  ローカル接続されたプレックスが、リモート接続されたプレックスよりも優先的に選択されます。この基準は共有ディスクグループに適用されます。
- メディアの種類
  SSD デバイスが HDD よりも優先的に選択されます。
- レイアウト
  ストライプ化プレックスが他のレイアウトよりも優先的に選択されます。

読み取りポリシーをカスタマイズするには、次の VxVM 読み取りポリシーのいずれかを選択します:

`prefer`	特定の名前のプレックスを使用して読み取りリクエストを処理します。`prefer` ポリシーを設定する際に希望するプレックスを 1 つ指定します。 指定されたプレックスでリクエストを処理できない場合、VxVM は `select` ポリシーのプレックス順序を適用します。

round
: 非シーケンシャル読み取り操作を利用可能なすべてのプレックスに対して「ラウンドロビン」方式で配分します。たとえば、3 つのプレックスが使用可能な場合、VxVM は読み取り要求が各プレックスに 1/3 ずつ配分されるようにプレックスを切り替えます。シーケンシャル読み取り操作では 1 つのプレックスにのみアクセスします。このアプローチでは、ドライブまたはコントローラの先読みキャッシュポリシーを利用します。

select
: プレックスの性質に基づいてプレックスを選択します。サイトの整合性が有効になっていなければ、select ポリシーがデフォルトの読み取りポリシーです。サイトが構成されている場合、VxVM は内部でサイト読み取りポリシーを切り替えます。

  select ポリシーは以下の順序でプレックスを選択します。

  - ローカル接続されたストライプ化 SSD プレックス
  - ローカル接続された SSD プレックス
  - ローカル接続されたストライプ化プレックス
  - ローカル接続されたプレックス
  - リモート接続されたストライプ化 SSD プレックス
  - リモート接続された SSD プレックス

  VxVM が上記の特性を持つプレックスを発見できなかった場合、VxVM は round ポリシーを使用します。

siteread
: ローカルで定義されたサイトでプレックスから優先的に読み取ります。この方法は、サイトの整合性が有効になっているディスクグループ内のボリュームのデフォルトのポリシーです。

  siteread ポリシーは以下の順序でプレックスを選択します。

  - ローカルサイト、ローカル接続されたストライプ化 SSD プレックス
  - ローカルサイト、ローカル接続された SSD プレックス
  - ローカルサイト、ローカル接続されたストライプ化プレックス
  - ローカルサイト、ローカル接続されたプレックス
  - ローカルサイト、リモート接続されたストライプ化 SSD プレックス
  - ローカルサイト、リモート接続された SSD プレックス
  - ローカルサイト、リモート接続されたストライプ化プレックス
  - ローカルサイト、リモート接続されたプレックス

  VxVM が上記の特性を持つプレックスを発見できなかった場合、VxVM は select ポリシーを順番に使用します。

split
: 読み取った要求を分割し、それらの要求を利用可能なすべてのプレックスに分散します。

メモ: RAID 5 ボリュームには読み取りポリシーを設定できません。

読み取りポリシーを round に設定するには、次のコマンドを使います。

```
# vxvol [-g diskgroup] rdpol round volume
```

たとえば、ディスクグループ mydg 内のボリューム vol01 の読み取りポリシーをラウンドロビンに設定する場合は、次のコマンドを使います。

```
# vxvol -g mydg rdpol round vol01
```

読み取りポリシーを prefer に設定するには、次のコマンドを使います。

```
# vxvol [-g diskgroup] rdpol prefer volumepreferred_plex
```

vol01 のポリシーをプレックス vol01-02 から優先的に読み取るように設定する場合は、次のコマンドを使います。

```
# vxvol -g mydg rdpol prefer vol01 vol01-02
```

読み取りポリシーを select に設定するには、次のコマンドを使います。

```
# vxvol [-g diskgroup] rdpol select volume
```

9

VxFS ファイルシステムの作成とマウント

この章では以下の項目について説明しています。

VxFS ファイルシステムの作成

`mkfs` コマンドは、特殊キャラクタデバイスファイルに書き込むことで、VxFS ファイルシステムを作成します。特殊キャラクタデバイスは、VxVM (Veritas Volume Manager) ボリュームである必要があります。`mkfs` コマンドは、ルートディレクトリと `lost+found` ディレクトリが組み込まれたファイルシステムを構築します。

`mkfs`を実行する前に対象デバイスを作成する必要があります。

ご使用のオペレーティングシステムのマニュアルを参照してください。

論理デバイス (VxVM ボリュームなど) を使っている場合は、VxVM のマニュアルを参照してください。

メモ: このリリースでは、LVM (Logical Volume Manager) または MD (Multiple Device) ドライバボリュームでの VxFS ファイルシステムの作成はサポートしていません。ext2 または ext3 ファイルシステムを VxFS ファイルシステムに変換する前に、基盤となる LVM を VxVM ボリュームに変換する必要もあります。vxvmconvert(1M) マニュアルページを参照してください。

mkfs(1M) と mkfs_vxfs(1M) のマニュアルページを参照してください。

mkfs コマンドでファイルシステムを作成する場合は、以下の特性を選択できます。

- 「ファイルシステムのブロックサイズ」
- 「インテントログサイズ」

ファイルシステムを作成するには

◆ ファイルシステムを作成するには、mkfs コマンドを使います。

```
mkfs [-t vxfs] [generic_options]
[-o specific_options] -m special [size]
```

`-t vxfs`	VxFS ファイルシステムタイプを指定します。
`-m`	ファイルシステムの作成に使ったコマンドラインを表示します。ファイルシステムはすでに作成された状態である必要があります。このオプションを使うと、ファイルシステムの構築に使うパラメータが判断できます。
generic_options	他の多くのファイルシステムと共通なオプションです。
`-o`*specific_options*	VxFS 固有のオプションです。
`-o N`	ファイルシステムのジオメトリを表示しますが、デバイスには書き込みません。
`-o largefiles`	ユーザーが 2 GB を超えるファイルを作成できるようになります。デフォルトのオプションは largefiles です。
`-o nomaxlink`	64K より大きいサブディレクトリのためにサポートが追加されます。ファイルシステムで maxlink が無効になっている場合は、サブディレクトリの制限はデフォルトでは 32K です。
special	特殊デバイスのファイル位置または特定ストレージデバイスのキャラクタデバイスノードを指定します。デバイスは、Veritas Volume Manager ボリュームである必要があります。
size	ファイルシステムで 512 バイトのセクタを指定します。*サイズ*を指定していない場合は mkfs により、特殊デバイスのサイズが決定されます。

次の例では、サイズが 12288 セクタの VxFS ファイルシステムを VxVM ボリューム上に作成します。

VxFS ファイルシステムを作成するには

1 ファイルシステムを作成します。

```
# /opt/VRTS/bin/mkfs /dev/vx/rdsk/diskgroup/volume 12288

version 16 layout
12288 sectors, 6144 blocks of size 1024, log size 256 blocks
rcq size 1024 blocks
largefiles supported
maxlink supported
WORM not supported
```

2 新規作成したファイルシステムをマウントします。

```
# mount -t vxfs /dev/vx/dsk/diskgroup/volume /mnt1
```

ファイルシステムのブロックサイズ

VxFS では、エクステント単位で割り当てが実行されます。他の UNIX ファイルシステムとは異なり、VxFS は断片が存在するブロックは割り当てに使いません。VxFS では、1 つ以上のブロックで構成されるエクステント単位で格納領域が割り当てられるからです。ファイルシステムの作成時に `mkfs -o bsize` オプションでブロックサイズを指定します。ブロックサイズは、ファイルシステムを作成した後では変更できません。VxFS の最小ブロックサイズは 1K です。

デフォルトのブロックサイズは、1 TB 未満のファイルシステムに対しては 1024 バイト、1 TB 以上のファイルシステムに対しては 8192 バイトです。

実行するアプリケーションの特性に応じてブロックサイズを選択します。たとえば、小さなサイズのファイルを多く作成するアプリケーションの場合、1 K ブロックサイズを指定することで領域が節約できます。大容量ファイルを作成するアプリケーションの場合は、1 K より大きいブロックサイズが適切です。大きいブロックサイズを使うと、ファイルシステムのオーバーヘッドとなる領域は減少しますが、ブロックサイズの倍数でないファイルが使う領域は増加します。システムの効率を最大限にするブロックサイズを特定する簡単な方法は、システムの負荷を各サイズで試してみて、最速となるサイズを選択することです。

インテントログサイズ

ファイルシステムの作成時に、`mkfs -o logsize` オプションでインテントログサイズを指定します。`fsadm` コマンドの `logsize` オプションを使って、インテントログのサイズを動的に増減できます。`mkfs` ユーティリティは、デフォルトのインテントログサイズ 64 MB を使

います。デフォルトのサイズで大部分の作業負荷に対応できます。システムが NFS サーバーとして使われる場合、またはシステムに書き込みが集中するような作業負荷に対して使われる場合、大きいログサイズを使うとパフォーマンスが向上する場合があります。

fsadm_vxfs(1M)および mkfs_vxfs(1M)のマニュアルページを参照してください。

インテントログサイズが大きい場合、リカバリ時間も比例して長くなり、通常の操作時にもファイルシステムでメモリなど多くのシステムリソースが消費される可能性があります。

大きいログサイズでも VxFS で高いパフォーマンスを得るために、数タイプのシステムパフォーマンスベンチマークが用意されています。ログサイズを選択する最適な方法は、ブロックサイズと同様に、システムの標準的な負荷を各サイズで試してみて、最速となるサイズを選択することです。

ファイルシステムの VxFS への変換

ストレージのニーズに対応するには、ファイルシステムを Veritas File System (VxFS) に変換する必要がある場合があります。

InfoScale には、ネーティブファイルシステム (Ext2、Ext3、Ext4) を VxFS に変換するためのオフラインおよびオンラインのソリューションが用意されています。

ファイルシステムのオンライン変換は、リアルタイムの移行プロセスです。このプロセスには、アプリケーションデータを既存の LVM ボリュームから VxVM ボリュームにコピーするために、追加の独立したストレージが必要です。この変換プロセスには、ファイルシステムの移行にある程度のアプリケーション停止時間が伴います。ネーティブファイルシステムの VxFS へのオンライン移行を実行するには、fsmigadm コマンドを使います。

ファイルシステムのオフライン変換は、アプリケーションがオフラインの間に既存のファイルシステムを変換できるプロセスです。オフライン変換には 2 つの手順があります。

1. Logical Volume Manager (LVM) ボリュームグループとオブジェクトを同等の VxVM ディスクグループおよびオブジェクトに変換する。vxvmconvert ユーティリティを使用し、LVM グループ VxVM ディスクグループに変換します。
2. ファイルシステムを VxFS に変換する。vxfsconvert ユーティリティを使用し、ネーティブファイルシステムを VxFS に変換します。

ファイルシステムの変換について詳しくは、『Veritas InfoScale ソリューションガイド』を参照してください。

VxFS ファイルシステムのマウント

VxFS ファイルシステムをマウントするには、mount コマンドを使います。mount コマンドを入力すると、この汎用的な mount コマンドは引数の構文を解析し、-t FSType オプションで指定されたファイルシステムに固有の mount コマンドを実行します。-t オプションを指定していない場合、コマンドはファイルシステムの /etc/fstab ファイル、および

特殊なファイルに合った FSType やマウントポイントを検索します。ファイルシステムタイプが指定されていない場合、mount は、デフォルトのファイルシステムを使います。

mount コマンドは、ファイルシステムでダーティログを検出すると、VxFS の fsck コマンドを自動的に実行してインテントログをクリーンアップします。この機能は、VxVM（Veritas Volume Manager）ボリュームにマウントされたファイルシステム上でのみサポートされます。

RHEL では、context、defcontext、fscontext、rootcontext の各マウントオプションを使用して、VxFS 操作モードを指定できます。

VxFS（Veritas File System）では、標準マウントモード（delaylog モード）に加えて、他の操作モードを指定できる次の mount オプションを用意しています。

- 「log マウントオプション」
- 「delaylog マウントオプション」
- 「tmplog マウントオプション」
- 「logiosize マウントオプション」
- 「nodatainlog マウントオプション」
- 「blkclear マウントオプション」
- 「mincache マウントオプション」
- 「convosync マウントオプション」
- 「ioerror マウントオプション」
- 「largefiles と nolargefiles マウントオプション」
- 「cio マウントオプション」
- 「mntlock マウントオプション」
- 「ckptautomnt マウントオプション」

キャッシング動作は mincache オプションで、O_SYNC および D_SYNC の書き込み動作は convosync オプションで変更できます。

fcntl（2）のマニュアルページを参照してください。

delaylog および tmplog モードでは、パフォーマンスが大幅に向上します。delaylog モードを使うと、通常 log モードの約 15 から 20% パフォーマンスが向上し、tmplog モードを使うとさらにパフォーマンスが向上します。パフォーマンスの向上は、実行される操作と作業負荷によって異なります。読み書きが集中する作業負荷に対しては、あまりパフォーマンスが向上しませんが、ファイルシステム構造に集中する作業負荷（mkdir、create、rename など）に対しては 100% 以上の向上が見込まれる可能性があります。モードを選択するには、各ログモードでシステムの標準的な負荷をテストして、パフォーマンスを比較することをお勧めします。

多くのモードは組み合わせて使えます。たとえば、デスクトップコンピュータでは、blkclear と mincache=closesync の両方のモードを使うことが考えられます。

mount コマンドは、ファイルシステムでダーティログを検出すると、VxFS の fsck コマンドを自動的に実行してインテントログをクリーンアップします。この機能は、VxVM（Veritas Volume Manager）ボリュームにマウントされたファイルシステム上でのみサポートされます。

mount_vxfs (1M) のマニュアルページを参照してください。

ファイルシステムをマウントするには

◆ ファイルシステムをマウントするには、mount コマンドを使います。

```
mount [-t vxfs] [generic_options] [-r] [-o specific_options] ¥

special mount_point
```

vxfs	ファイルシステムのタイプです。
generic_options	他の多くのファイルシステムと共通なオプションです。
specific_options	VxFS 固有のオプションです。
-o ckpt=*ckpt_name*	Storage Checkpoint をマウントします。
-o cluster	ファイルシステムを共有モードでマウントします。VxFS Cluster File System 機能を使う場合にのみ利用できます。
special	VxFS 特殊ブロックデバイスです。
mount_point	ファイルシステムをマウントするディレクトリです。
-r	読み取り専用のファイルシステムとしてマウントします。

次の例では、読み書きのアクセス権限と遅延ログモードを指定して、/dev/vx/dsk/fsvol/vol1 ファイルシステムを /mnt1 ディレクトリにマウントします。

ファイルシステムのマウントの例

◆ 読み書きのアクセス権限と遅延ログモードを指定して、/dev/vx/dsk/fsvol/vol1 ファイルシステムを /mnt1 ディレクトリにマウントします。

```
# mount -t vxfs -o delaylog /dev/vx/dsk/fsvol/vol1 /mnt1
```

log マウントオプション

ファイルシステムでは、ファイルシステムに対する構造上の変更はすぐにはディスクに書き込まれないため、非同期と言えます。これにより、処理効率が向上します。ただし、システムに対する直前の変更内容をシステム障害の発生時に失うことがあります。たとえば、ファイルに加えた属性に関する変更や直前に作成したファイルが失われる場合があります。log モードでは、write(2)、writev(2)、および pwrite(2)を除くすべてのシステムコールは、アプリケーションに返された後は永続性が保証されます。

rename(2)システムコールでは、ファイルデータの永続性を保証するためにソースファイルをディスクにフラッシュしてから、名前が変更されます。log と delaylog の両モードで、システムコールが復帰するときに名前の変更も永続的であることが保証されます。これは、新しいファイルの内容を一時ファイルに書き込んでからターゲットファイルに重ねて名前を変更することにより、原子性を維持してファイルの更新を試みるシェルスクリプトやプログラムにとって有益です。

delaylog マウントオプション

デフォルトのログモードである delaylog では、ファイルへの書き込みは遅延(バッファ)になります。つまり、書き込まれるデータは、いったんファイルシステムキャッシュにコピーされ、その後ディスクにフラッシュされます。delaylog モードでは、write(2)、writev(2)、pwrite(2)を除くほとんどのシステムコールの影響は、システムコールがアプリケーションに戻って約 3 秒後に永続性が保証されます。それに対して、他のほとんどのファイルシステムでは、復帰してから約 30 秒以上経過するまで大部分のシステムコールが永続性を持ちません。高速ファイルシステムリカバリはこのモードで動作します。

遅延書き込みでは、データをディスクに同期的に書き込むよりも、パフォーマンスは向上します。ただし、システム障害が起きた場合、障害発生の直前に書き込まれていたデータは、ディスクにフラッシュされていないため、失われることがあります。さらに、書き込み要求の一部としてファイルに領域が割り当てられており、システム障害の発生前に対応するデータがディスクにフラッシュされていなかった場合、初期化されていないデータがファイルに含まれます。

最も一般的な書き込みである遅延拡張書き込み(ファイルサイズを拡張する遅延書き込み)の場合、VxFS は、データがディスクにフラッシュされるまで待機してから、ディスクのファイルサイズを更新することにより、初期化されていないデータがファイルに含まれる問題を回避しています。 ディスクにデータがフラッシュされる前にシステム障害が起きた場合、ファイルサイズがまだ更新されていないため、初期化されていないデータがファイルに含まれません。 割り当てられた未使用ブロックは再利用されます。

rename(2)システムコールでは、ファイルデータの永続性を保証するためにソースファイルをディスクにフラッシュしてから、名前が変更されます。log と delaylog の両モードで、システムコールが復帰するときに rename も永続的であることが保証されます。これは、新しいファイルの内容を一時ファイルに書き込んでからターゲットファイルに重ねて名

前を変更することにより、原子性を維持してファイルの更新を試みるシェルスクリプトやプログラムにとって有益です。

tmplog マウントオプション

tmplog モードでのシステムコールの影響は、delaylog モードの場合と類似した永続性の保証を備えています。さらに、遅延拡張書き込みの拡張フラッシュが無効にされることによってもパフォーマンスは向上しますが、システム障害の発生時に書き込みが進行中であったファイルで、データが失われたり、初期化されていないデータが出現する可能性が高まります。このモードは一時ファイルシステムのみに使うことをお勧めします。高速ファイルシステムリカバリはこのモードで動作します。

メモ:「システムコールの影響」という用語は、st_atime の変更を除く、システムコールによって引き起こされたファイルシステムのデータおよびメタデータに対する変更を指します。

stat(2)のマニュアルページを参照してください。

ログモードの永続性の保証

VxFS はすべてのログモードで POSIX に完全準拠しています。fsync(2)および fdatasync(2)システムコールの影響は、復帰後に永続性が保証されます。write(2)、writev(2)または pwrite(2)によって変更されたデータやメタデータの永続性の保証は、ログのマウントオプションによる影響を受けません。これらのシステムコールの影響は、O_SYNC、O_DSYNC、VX_DSYNC または VX_DIRECT フラグが、convosync= mount オプションによって変更されてファイル記述子に指定されている場合のみ、永続性が保証されます。

VxFS ファイルシステムにおいて、NFS サーバーの動作は、log および tmplog マウントオプションによって影響されませんが、delaylog マウントオプションには影響されます。tmplog の使用時を除き、VxFS はどのような場合でも、NFSv3 および NFSv4 標準の永続性必要条件に準拠しています。VxFS ファイルシステムの log モード専用に開発された場合を除き、UNIX アプリケーションは他のほとんどのファイルシステムが提供する程度の永続性保証を予期しているため、delaylog モードでマウントされた VxFS ファイルシステムで使っても堅ろう性が向上します。log、mincache= および closesync の各 mount オプションは、他のほとんどのファイルシステムで得られる保証よりも高度な永続性の保証を必要とするアプリケーションにとっては有益です。ただし、現在入手できる大部分の商用アプリケーションは、delaylog モードの場合でも、デフォルトの VxFS mount オプションで良好に動作します。

mount_vxfs (1M) のマニュアルページを参照してください。

logiosize マウントオプション

logiosize=size オプションは、読み取り、変更および書き込み機能を使うストレージデバイスのパフォーマンスを向上させるために提供されています。ファイルシステムのマウント時に logiosize を指定すると、少なくとも *size* バイトまたは *size* の倍数の容量のインテントログが書き込まれて、デバイスから最高のパフォーマンスを実現できます。

mount_vxfs(1M)のマニュアルページを参照してください。

size の値には、512、1024、2048、4096 または 8192 を指定できます。

nodatainlog マウントオプション

破損ブロックの再ベクトル化をサポートしないディスクを使うシステムには、nodatainlog モードを使います。通常、VxFS では、同期書き込みにインテントログを使います。i ノード更新とデータの両方が 1 つのトランザクションとしてログに記録されるため、ディスクへの書き込みは二度ではなく一度で実行します。同期書き込みからアプリケーションに復帰する際には、ファイルシステムはデータがすでに書き込まれていることをアプリケーションに通知します。このとき、ディスク障害が原因でメタデータの更新に失敗すると、ファイルには破損マークが付けられ、ファイル全体が失われることになります。

破損ブロックの再ベクトル化をサポートしているディスクの場合は、データの更新に失敗する可能性が低いため、同期書き込みが正常にログに記録されます。破損ブロックの再ベクトル化をサポートしていないディスクの場合は、データの更新に失敗する可能性があるため、nodatainlog モードを使う必要があります。

nodatainlog モードを使ったファイルシステムの同期書き込みは、Veritas File System の標準モードよりも約 50% 処理速度が低下します。他の操作では性能に影響しません。

blkclear マウントオプション

データの安全保護を重視する環境では blkclear モードを使います。blkclear モードは、初期化されていない記憶域がファイル内に存在しないようにします。ファイルにエクステントを割り当てるときにディスクのエクステントを消去して、整合性を向上させます。このモードでは書き込み範囲を広げても影響はありません。blkclear モードを使うと、VxFS の標準モードよりも約 10 % 処理速度が低下しますが、どの程度の作業負荷がかかるかによっても異なります。

mincache マウントオプション

mincache モードには、次のサブオプションが用意されています。

- mincache=closesync
- mincache=direct
- mincache=dsync

- mincache=unbuffered
- mincache=tmpcache

ユーザーがシステムを終了する前に電源を切断する可能性があるデスクトップ環境では、mincache=closesync モードが有効です。このモードでは、ファイルを閉じるときにファイルの変更がすべてディスクにフラッシュされます。

パフォーマンスを向上させるために、通常のファイルシステムでは、データと i ノードの変更をディスクに同期を取って書き込むことは実行しません。システムクラッシュが発生した場合、発生前の数分間に更新したファイルのデータが失われる可能性があります。mincache=closesync モードを使うと、システムクラッシュが発生したり、電源が切断された場合でも、その時点で開いていたファイルのデータしか失われません。mincache=closesync モードを使うと、VxFS の標準モードよりも約 15 % 処理速度が低下しますが、どの程度の作業負荷がかかるかによっても異なります。

次に、mincache モードを使う環境について説明します。

- mincache=direct、mincache=unbuffered および mincache=dsync の各モードは、I/O のカーネルバッファリングや非同期 I/O の遅延フラッシュによってアプリケーションが信頼性の問題を抱えている場合に使います。
- mincache=direct モードおよび mincache=unbuffered モードを使うと、すべての非同期 I/O 要求は VX_DIRECT または VX_UNBUFFERED キャッシュアドバイザリが指定されているときと同様に処理されます。
- mincache=dsync モードを使うと、すべての非同期 I/O 要求は VX_DSYNC キャッシュアドバイザリが指定されているときと同様に処理されます。
  VX_DIRECT、VX_UNBUFFERED および VX_DSYNC の説明と、ダイレクト I/O の必要条件については、vxfsio(7)のマニュアルページを参照してください。
- mincache=direct、mincache=unbuffered および mincache=dsync の各モードでも mincache=closesync モードと同様にファイルデータが close 実行時にフラッシュされます。

mincache=direct、mincache=unbuffered および mincache=dsync の各モードでは非同期 I/O が同期 I/O に変更されるため、小規模から中規模のファイルを扱うアプリケーションではスループットが著しく低下する可能性があります。VX_DIRECT および VX_UNBUFFERED キャッシュアドバイザリではデータのキャッシングが実行されないため、読み取りキャッシュを利用しているアプリケーションでは mincache=dsync モードを使うことでパフォーマンスの低下が抑えられます。mincache=direct および mincache=unbuffered は、バッファ I/O よりも CPU 時間を必要としません。

データの整合性よりもパフォーマンスを優先させる場合は、mincache=tmpcache モードを使います。mincache=tmpcache モードでは、遅延拡張書き込み操作に対する特殊な処理が無効になるので、整合性が低下する分、パフォーマンスが向上します。他の mincache モードとは異なり、tmpcache モードでは閉じられているファイルをディスクに

フラッシュしません。mincache=tmpcache オプションを使うと、クラッシュの発生時に拡張されていたファイルに不良なデータが含まれてしまうことがあります。

mount_vxfs(1M)のマニュアルページを参照してください。

convosync マウントオプション

convosync(convert osync)モードには、次のサブオプションが用意されています。

- convosync=closesync

  メモ: convosync=closesync モードでは同期書き込みおよびデータ同期書き込みを非同期書き込みに変換し、ファイルの変更はファイルを閉じるときにディスクにフラッシュします。

- convosync=delay
- convosync=direct
- convosync=dsync

  メモ: convosync=dsync オプションを使うと、POSIX の同期 I/O 規格に対し非準拠になります。

- convosync=unbuffered

convosync=delay モードでは、同期書き込みおよびデータ同期書き込みは即座に処理されず、遅延されます。ファイルを閉じるときに特別なアクションは実行されません。このオプションでは、O_SYNC を指定して開かれたファイルで通常保証されるデータの整合性は実質的に取り消されます。

open(2)、fcntl(2)、vxfsio(7)の各マニュアルページを参照してください。

警告: 同期 I/O が非同期 I/O に事実上変換されるため、convosync=closesync または convosync=delay モードを使うときには細心の注意を払ってください。データの信頼性を保持するために同期 I/O を使うアプリケーションでは、システムクラッシュが発生して同期書き込み用データが消失すると処理に失敗することがあります。

convosync=dsync モードでは、同期書き込みがデータ同期書き込みに変換されます。

closesync と同様に、direct、unbuffered および dsync の各モードを使うと、ファイルを閉じるときにファイルの変更がディスクにフラッシュされます。これらのモードを使って、同期 I/O を使うアプリケーションの処理を高速化できます。データの整合性を重視する多くのアプリケーションでは、O_SYNC fcntl を指定してファイルデータを同期的に書き込

んでいます。ただし、これにより、i ノードの更新回数が増えるため、パフォーマンスが低下します。convosync=dsync、convosync=unbuffered および convosync=direct のモードを使うと、アプリケーションで i ノードの時刻を修正することなく同期書き込みが可能となるため、この問題は軽減されます。

convosync=dsync、convosync=unbuffered および convosync=direct を使う前に、ファイルシステムを利用するアプリケーションすべてが O_SYNC での書き込み時に i ノード時刻の同期更新を必要としていないことを確認してください。

ioerror マウントオプション

マウントされているファイルシステムで I/O エラーを処理するためのポリシーを設定します。I/O エラーが発生するのは、ファイルデータの読み取りや書き込みの最中、またはメタデータの読み取りや書き込みの最中です。これらの I/O エラーにより、ファイルシステムは、停止したり、パフォーマンスが低下します。ioerror オプションは、ファイルシステムがさまざまなエラーにどのように対応するかを決める 5 つのポリシーを提供します。5 つのすべてのポリシーは、ファイルシステムを停止するか、破損した i ノードを不良であるとマークすることで、データの破損を抑えます。

各ポリシーは次のとおりです。

- 「ポリシーの無効化」
- 「wdisable ポリシーと mwdisable ポリシー」
- 「mdisable ポリシー」

ポリシーの無効化

disable を選択した場合、VxFS は、I/O エラーの検出後にファイルシステムを無効にします。この場合は、ファイルシステムのマウントを解除し、その I/O エラーの原因となった状態を修正する必要があります。修復が終わったら、fsck を実行し、ファイルシステムを再びマウントします。ファイルシステムを修復するには、多くの場合、fsck を再生します。完全な fsck が必要となるのは、ファイルシステムのメタデータに構造上の損傷がある場合に限られます。RAID 5 やミラー化ディスクなど、下位ストレージが冗長である環境では、disable を選択してください。

wdisable ポリシーと mwdisable ポリシー

wdisable（書き込み無効化）または mwdisable（メタデータ書き込み無効化）を選択した場合は、発生したエラーのタイプによって、ファイルシステムが無効にされたり、縮退します。非冗長ストレージを使う場合など、書き込みエラーよりも読み取りエラーの方が多く繰り返される可能性がある環境では、wdisable または mwdisable を選択してください。mwdisable が、ローカルマウントに対するデフォルトの ioerror マウントオプションです。

メモ: プレックスが 1 つのみのときに問題が発生した場合、wdisable または mwdisable が選択されているときは、ミラーボリュームファイルシステムは無効になりません。

mount_vxfs (1M) のマニュアルページを参照してください。

メモ: nodisable オプションが選択されている場合、動作は mwdisable ioerror ポリシーと同じになります。詳しくは、mwdisable オプションを参照してください。

mdisable ポリシー

mdisable (メタデータの無効化) を選択した場合、メタデータの読み取りまたは書き込みの失敗によりファイルシステムが無効になります。ただし、このような処理の失敗がデータエクステントに限定される場合はファイルシステムの処理が継続されます。mdisable は、クラスタをマウントするためのデフォルトの ioerror マウントオプションとなっています。

largefiles と nolargefiles マウントオプション

Veritas File System (VxFS) では、最大 16 TB のスパースファイルと、1 KB - 2 TB のサイズの非スパースファイルをサポートします。

メモ: バックアップなどのアプリケーションやユーティリティで大容量ファイルを認識できない場合、障害が発生する可能性があります。そのような場合は、大容量ファイル機能を有効にしないでファイルシステムを作成します。

p.271 の「大容量ファイル機能を有効にしたファイルシステムの作成」を参照してください。

p.272 の「大容量ファイル機能を有効にしたファイルシステムのマウント」を参照してください。

p.272 の「大容量ファイルでのファイルシステムの管理」を参照してください。

大容量ファイル機能を有効にしたファイルシステムの作成

大容量ファイル機能を有効にしたファイルシステムを作成するには

```
# mkfs -t vxfs -o largefiles special_devicesize
```

largefiles を指定して、largefiles フラグを設定します。これにより、ファイルシステムでは 2 GB 以上のファイルを取り扱うことができるようになります。これはデフォルトのオプションです。

フラグを解除し、大容量ファイルを作成できないようにするには

```
# mkfs -t vxfs -o nolargefiles special_devicesize
```

largefiles は永続型フラグであり、ディスクに保存されます。

大容量ファイル機能を有効にしたファイルシステムのマウント

nolargefiles オプションを指定した mount コマンドが成功した場合、ファイルシステムでは大容量ファイルを含んだり、作成することはできません。largefiles オプションを指定した mount コマンドが成功した場合、ファイルシステムでは大容量ファイルを含めたり作成することができます。

指定された largefiles|nolargefiles オプションとディスク上のフラグが一致しない場合は、mount コマンドが成功しません。

largefiles または nolargefiles オプションが指定されていない場合、mount コマンドではデフォルトでオンディスクフラグの現在の設定値に一致するように定義されているため、どちらのオプションも指定しないことをお勧めします。ファイルシステムのマウント後、fsadm ユーティリティを使って largefiles オプションを変更できます。

大容量ファイルでのファイルシステムの管理

大容量ファイル機能を有効にしたファイルシステムの管理には、次のようなものがあります。

- 大容量ファイルフラグの現在の状態の確認
- マウントされたファイルシステム上で使う機能の切り替え
- マウント解除されたファイルシステム上で使う機能の切り替え

largefiles フラグの現在の状態を確認するには、次のコマンドのいずれかを入力します。

```
# mkfs -t vxfs -m special_device
 # /opt/VRTS/bin/fsadm mount_point | special_device
```

マウントされたファイルシステムで機能を切り替えるには、次のコマンドを入力します。

```
# /opt/VRTS/bin/fsadm -o [no]largefiles mount_point
```

マウント解除されたファイルシステムで機能を切り替えるには、次のコマンドを入力します。

```
# /opt/VRTS/bin/fsadm -o [no]largefiles special_device
```

大容量ファイルをすでに保持しているファイルシステムの場合は、nolargefiles に変更できません。

mount_vxfs(1M)、fsadm_vxfs(1M)および mkfs_vxfs(1M)の各マニュアルページを参照してください。

cio マウントオプション

cio（同時 I/O）オプションで、同時読み取りや同時書き込みを行うためにマウントされているファイルシステムを指定します。cio は指定しているが、ライセンスがない場合、mount コマンドを実行するとエラーメッセージが発行され、ファイルシステムがマウントされないまま処理が終了します。cio オプションは、再マウントで無効にはできません。cio オプションを無効にするには、cio オプションを指定しない状態でファイルシステムをマウント解除し、再マウントする必要があります。

mntlock マウントオプション

mntlock オプションを使うと、ファイルシステムがアプリケーションによってマウント解除されなくなります。アプリケーションでファイルシステムを監視しているときに、そのファイルシステムが他のアプリケーションや管理者によって間違ってマウント解除されないように設定する必要がある場合に、このオプションは便利です。

以前にロックしたファイルシステムについては、vxumount コマンドの mntunlock オプションを使うと、mntlock オプションが取り消されます。

ckptautomnt マウントオプション

ckptautomnt オプションは、Storage Checkpoint へのアクセスを容易にする Storage Checkpoint への可視機能を有効にします。

p.671 の「Storage Checkpoint の可視性」を参照してください。

マウントコマンドオプションの組み合わせ

mount オプションは任意に組み合わせることができますが、意味をなさない組み合わせもあります。一般的で適切な mount オプションの組み合わせ例を次に示します。

オプションを使ってデスクトップファイルシステムをマウントするには、次のように入力します。

```
# mount -t vxfs -o log,mincache=closesync ¥
 /dev/vx/dsk/diskgroup/volume /mnt
```

この組み合わせにより、ファイルが閉じられるときに、データが確実にディスクに同期化され、データは失われません。このため、アプリケーションが終了し、ファイルが閉じられたときに、システムの電源を即座に落とした場合でもデータは失われません。

一時ファイルシステムをマウントする場合、またはバックアップからリストアする場合は、次のように入力します。

```
# mount -t vxfs -o tmplog,convosync=delay,mincache=tmpcache ¥
/dev/vx/dsk/diskgroup/volume /mnt
```

この組み合わせは、データの絶対整合性よりパフォーマンスを重視する一時ファイルシステムに有効です。O_SYNC 書き込みは遅延書き込みとして処理され、遅延拡張書き込みは処理されません。このため、システムクラッシュが発生した場合、ファイルに破損したデータが含まれることがあります。この mount コマンドを組み合わせると、システムクラッシュが発生する 30 秒以内に書き込まれたファイルには壊れたデータが含まれたり、ファイルが失われることがあります。ただし、このようなファイルシステムは、アプリケーションによっても異なりますが、log モードを使ったファイルシステムと比べてディスク書き込みが極めて少なくなり、パフォーマンスが著しく向上します。

同期書き込みを行うファイルシステムをマウントするには、次のように入力します。

```
# mount -t vxfs -o log,convosync=dsync ¥
/dev/vx/dsk/diskgroup/volume /mnt
```

O_SYNC 書き込みを実行するが、データ同期書き込みセマンティクスのみが必要なアプリケーションのパフォーマンスを向上させる場合に、この組み合わせが使えます。ファイルシステムが convosync=dsync を使ってマウントされている場合、データの整合性を保持したまま、パフォーマンスが著しく向上します。

ファイルシステムのマウント解除

現在マウントされているファイルシステムのマウントを解除するには、umount コマンドを使います。

vxumount（1M）のマニュアルページを参照してください。

ファイルシステムをマウント解除するには

- ファイルシステムのマウントを解除するには、umount コマンドを使います。

  マウントを解除するファイルシステムを *mount_point* または *special* として指定します。*special* は、ファイルシステムが存在する VxFS 特殊ブロックデバイスです。

次に、ファイルシステムのマウント解除の例を示します。

ファイルシステムのマウント解除の例

- ファイルシステム /dev/vx/dsk/fsvol/vol1 のマウントを解除します。

  ```
  # umount /dev/vx/dsk/fsvol/vol1
  ```

ファイルシステムサイズの変更

マウントされた VxFS ファイルシステムサイズを拡張または縮小するには、fsadm コマンドを使います。ファイルシステムが増加できるサイズは、ファイルシステムディスクのレイアウトバージョンに依存します。バージョン 7 以降のファイルシステムは、256 テラバイトま

で増やすことができます。バージョン 7 以降のディスクレイアウトのファイルシステムが増やせるサイズは、ファイルシステムのブロックサイズに依存します。

fsadm_vxfs(1M)および fdisk(8)の各マニュアルページを参照してください。

fsadm を使ったファイルシステムの拡張

fsadm コマンドを使ってファイルシステムのサイズを変更できます。

VxFS ファイルシステムのサイズを変更するには

- VxFS ファイルシステムを拡張するには、fsadm コマンドを使います。

```
fsadm [-t vxfs] [-b newsize] [-r rawdev] ¥
mount_point
```

vxfs	ファイルシステムタイプです。
newsize	ファイルシステムの拡張後のサイズです。デフォルトの単位はセクタですが、k または K でキロバイト、m または M でメガバイト、g または G でギガバイトをそれぞれ指定できます。
mount_point	ファイルシステムのマウントポイントです。
-r rawdev	/etc/fstab にエントリがなく、fsadm で RAW デバイスを確認できない場合、RAW デバイスのパス名を指定します。

次の例では、/mnt1 にマウントされたファイルシステムサイズを 22528 セクタに拡張します。

ファイルシステムを 22528 セクタに拡張する例

- /mnt1 上にマウントされた VxFS ファイルシステムサイズを 22528 セクタに拡張します。

```
# fsadm -t vxfs -b 22528 /mnt1
```

次の例では、/mnt1 にマウントされたファイルシステムサイズを 500 GB に拡張します。

ファイルシステムを 500 GB に拡張する例

- /mnt1 上にマウントされた VxFS ファイルシステムサイズを 500 GB に拡張します。

```
# fsadm -t vxfs -b +500g /mnt1
```

ファイルシステムの縮小

ファイルシステムがマウント済みの場合でも、fsadm を使うとファイルシステムサイズを縮小できます。

警告: この操作後に、デバイスの最後に未使用領域ができます。この時点で、デバイスのサイズを変更できます。ただし、デバイスの容量がファイルシステムの新しいサイズより大きくなるように設定してください。

VxFS ファイルシステムのサイズを縮小するには

◆ VxFS ファイルシステムのサイズを縮小するには、fsadm コマンドを使います。

```
fsadm  [-t vxfs] [-b newsize] [-r rawdev] mount_point
```

vxfs	ファイルシステムタイプです。
newsize	ファイルシステムの縮小後のサイズですデフォルトの単位はセクタですが、k または K でキロバイト、m または M でメガバイト、g または G でギガバイトをそれぞれ指定できます。
mount_point	ファイルシステムのマウントポイントです。
-r rawdev	/etc/fstab にエントリがなく、fsadm で RAW デバイスを確認できない場合、RAW デバイスのパス名を指定します。

次の例では、/mnt1 にマウントされた VxFS ファイルシステムを 20480 セクタに縮小します。

ファイルシステムを 20480 セクタに縮小する例

◆ /mnt1 上にマウントされた VxFS ファイルシステムを 20480 セクタに縮小します。

```
# fsadm -t vxfs -b 20480 /mnt1
```

次の例では、/mnt1 にマウントされたファイルシステムを 450 GB に縮小します。

ファイルシステムを 450 GB に縮小する例

◆ /mnt1 上にマウントされた VxFS ファイルシステムを 450 GB に縮小します。

```
# fsadm -t vxfs -b 450g /mnt1
```

ファイルシステムの再構成

ファイルシステムがマウント済みの場合でも、fsadm を使うと、断片化したファイルシステムを再構成することによって縮小できます。この方法は、以前に縮小できなかったファイルシステムサイズを縮小する場合に使えます。

VxFS ファイルシステムを再構成するには

- VxFS ファイルシステムを再構成するには、fsadm コマンドを使います。

```
fsadm [-t vxfs] [-e] [-d] [-E] [-i] [-D] [-H] [-r rawdev]
mount_point
```

vxfs	ファイルシステムタイプです。
-d	サブディレクトリエントリの後に、他のすべてのエントリが最終アクセス日時を基準に降順で並ぶように、ディレクトリエントリの順序を変更します。また、ディレクトリを縮小して空き領域を削除します。
-D	ディレクトリの断片化を報告します。
-e	ファイルシステムの断片化を最小限に抑えます。エクステント数を最小限に抑えるためにファイルが再構成されます。
-E	エクステントの断片化を報告します。
i	過去 60 秒間に使用されたファイルを再構成できないことを示します。 ファイルシステムの再構成中に、アプリケーションが任意のファイルにアクティブにアクセスしているか、過去 60 秒間にアクセスしていた場合、ファイルシステムの再構成によって該当ファイルの読み取りと書き込み操作が影響を受けることはなく、ファイルは再構成から除外されます。
-H	-E オプションと -D オプションで使用したときに、ストレージサイズをヒューマンフレンドリな単位 (KB/MB/GB/TB/PB/EB) で表示します。
mount_point	ファイルシステムのマウントポイントです。
-r *rawdev*	/etc/fstab にエントリがなく、fsadm で RAW デバイスを確認できない場合、RAW デバイスのパス名を指定します。

空き領域の断片化の解消を実行するには

- VxFS ファイルシステムの空き領域の断片化の解消を実行するには、fsadm コマンドを使います。

```
fsadm [-t vxfs] [-C] mount_point
```

vxfs	ファイルシステムタイプです。
-C	ファイルシステムの空き領域の断片化を最小化します。これにより、デバイスの空き領域でより大きいチャンクを生成するように試行されます。
mount_point	ファイルシステムのマウントポイントです。

次の例では、/mnt1 にマウントされた VxFS ファイルシステムを再構成します。

VxFS ファイルシステムの再構成の例

- /mnt1 にマウントされた VxFS ファイルシステムを再構成します。

```
# fsadm -t vxfs -EeDd /mnt1
```

次の例では、/mnt1 にマウントされたファイルシステムの空き領域の断片化を最小化します。

空き領域の断片化の解消の実行例

- /mnt1 にマウントされた VxFS ファイルシステムの空き領域を最小化します。

```
# fsadm -t vxfs -C /mnt1
```

マウントされているファイルシステムの情報の表示

現在マウントされているファイルシステムの一覧を表示するには、mount コマンドを使います。

mount_vxfs(1M)と mount(8)のマニュアルページを参照してください。

マウントされたファイルシステムの状態を表示するには

- マウントされているファイルシステムの状態を表示するには、mount コマンドを使います。

```
mount
```

  これにより、マウントされたすべてのファイルシステムのタイプと mount オプションが表示されます。

次の例は、オプションを使用せずに mount コマンドを呼び出すことにより、マウント済みファイルシステムの情報を表示したものです。

マウントされているファイルシステムの情報を表示するには

◆ オプションなしで mount コマンドを実行します。

```
# mount
/dev/sda3 on / type ext3 (rw,acl,user_xattr)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
/dev/vx/dsk/testdg/vol01 on /vol01_testdg type vxfs
 (rw,delaylog,largefiles,ioerror=mwdisable)
```

ファイルシステムタイプの識別

特定のファイルシステムのタイプを確認するには、fstyp コマンドを使います。これは、ファイルシステムが他の環境で作成されており、そのタイプを確認する場合に使えます。

fstyp_vxfs(1M) のマニュアルページを参照してください。

ファイルシステムタイプを確認するには

◆ ファイルシステムタイプを確認するには、fstyp コマンドを使います。

```
fstyp -v special
```

special ブロックまたはキャラクタ (RAW) デバイス。

-v 調べる必要があるデバイスを指定します。

次の例では、fstyp コマンドを使って、/dev/vx/dsk/fsvol/vol1 デバイスのファイルシステムタイプを確認します。

ファイルシステムのタイプを確認するには

- `fstyp` コマンドを使って、`/dev/vx/dsk/fsvol/vol1` デバイスのファイルシステムタイプを確認します。

```
# fstyp -v /dev/vx/dsk/fsvol/vol1
```

出力は、ファイルシステムタイプが `vxfs` であることを示しており、次のようなファイルシステム情報が表示されます。

```
 vxfs
magic a501fcf5  version 16  ctime Mon 04 May 2020 04:05:56 PM IST
volguid 3f03b5b2-8c55-11ea-b66f-abd4f2521f81
logstart 0  logend 0
bsize  1024 size  41943040 dsize  41943040  ninode 0  nau 0
defiextsize 0  ilbsize 0  immedlen 96  ndaddr 10
aufirst 0  emap 0  imap 0  iextop 0  istart 0
bstart 0  femap 0  fimap 0  fiextop 0  fistart 0  fbstart 0
nindir 2048  aulen 32768  auimlen 0  auemlen 8
auilen 0  aupad 0  aublocks 32768  maxtier 15
inopb 4  inopau 0  ndiripau 0  iaddrlen 8  bshift 10
inoshift 2  bmask fffffc00  boffmask 3ff  checksum 103b34fd9
oltext1 32  oltext2 4714  oltsize 1  checksum2 0
free 41862017  ifree 0
efree  1 2 3 2 2 0 1 2 2 2 1 1 1 1 2 0 0 1 1 1 1 1 1 2 1 0 0 0 0
 0 0 0
```

空き領域の監視

通常、Veritas File System（VxFS）はファイルシステムの空き領域が 10% 未満にならない限り、最高のパフォーマンスを実現します。これは、ファイルシステムの空き領域が 10% 以上であれば、断片化を解消でき、十分なエクステント領域を確保できるからです。`df` コマンドを定期的に使って、空き領域を監視することをお勧めします。

`df_vxfs`（1M）のマニュアルページを参照してください。

空き領域のないファイルシステムは、ファイルシステムのパフォーマンスに悪影響を与えます。このため、空き領域のないファイルシステムの場合、一部のファイルを削除するか、または領域を拡張する必要があります。

`fsadm_vxfs`（1M）のマニュアルページを参照してください。

VxFS はシンストレージ LUN 上の空きストレージの再生をサポートします。

p.783 の「ファイルシステムのシン再生について」を参照してください。

断片化の監視

断片化により、パフォーマンスや可用性が低下します。fsadm コマンドの断片化チェック機能や再編成機能を定期的に使うことをお勧めします。

断片化で問題が発生しないようにするための最も簡単な方法は、cron コマンドを使って定期的な断片化解消の実行のスケジュールを設定することです。

断片化を解消するスケジューリングは、週単位(ファイルシステムの使用頻度が高い場合)または月単位(ファイルシステムの頻度が低い場合)に設定する必要があります。エクステントの断片化は、fsadm コマンドを使って監視する必要があります。

断片化の程度を確認するには、次の要因をチェックします。

- 8 ブロック未満のエクステントの空き領域の割合
- 64 ブロック未満のエクステントの空き領域の割合
- 64 ブロック以上のエクステントの空き領域の割合

断片化が解消されているファイルシステムには、次の特性があります。

- 8 ブロック未満のエクステントの空き領域が 1% 未満
- 64 ブロック未満のエクステントの空き領域が 5% 未満
- 64 ブロック以上の空きエクステントとして利用可能な領域が全ファイルシステムサイズの 5% 以上

断片化が解消されていないファイルシステムには、次の特性が 1 つ以上あります。

- 8 ブロック未満のエクステントの空き領域が 5% 以上
- 64 ブロック未満のエクステントの空き領域が 50% 以上
- 64 ブロック以上の空きエクステントとして利用可能な領域が全ファイルシステムサイズの 5% 未満

断片化は、断片化インデックスに基づいて判断することもできます。2 タイプのインデックス(ファイルの断片化インデックスと空き領域の断片化インデックス)は、fsadm コマンドによって生成されます。これらのインデックスはいずれも範囲は 0 ～ 100 で示されます。ファイルの断片化と空き領域の断片化の状態についてある程度判断できます。断片化インデックスの値が 0 である場合、ファイルシステムに断片化は存在しないことを意味します。値が 100 である場合、ファイルシステムは最大限に断片化されていることを意味します。インデックスに基づいて、fsadm コマンドを使って適切な断片化の解消(defragmentation)オプションを使う必要があります。たとえば、ファイルの断片化インデックスの値が大きい場合、fsadm コマンドは -e オプションを指定して実行する必要があります。空き領域の断片化インデックスの値が大きい場合、fsadm コマンドは -C オプションを指定して実行する必要があります。-e オプションを指定して fsadm コマンドを実行すると、内部的には、ファイルの断片化の解消の前に、空き領域の断片化の解消が実行されます。

fsadm の実行を適切な間隔でスケジュールし、エクステントを再構成する前後に fsadm のエクステントの断片化チェック機能を実行することで、最適なエクステントの再構成を実行するスケジュールを決定できます。

事前チェック結果は、再構成前の断片化の程度を示します。断片化の度合いが不適切な値に近づいている場合は、fsadm の実行間隔を短くする必要があります。断片化の度合いが低い場合は、fsadm の実行間隔を長くできます。

事後チェック結果は、再構成後の結果を示します。断片化の度合いは、断片化されていないファイルシステムの特性と同程度である必要があります。同程度でない場合、ファイルシステムのサイズを変更することをお勧めします。空き領域のないファイルシステムでは断片化が発生しやすく、その解消も困難です。また、その問題のあるファイルシステムで、使用頻度があまり高くない時間帯に再構成を実行していない可能性もあります。

ディレクトリ再構成は、エクステント再構成ほど重要ではありませんが、定期的にディレクトリ再構成を実行するとパフォーマンスが向上します。エクステント再構成のスケジューリングに合わせてファイルシステムのディレクトリ再構成のスケジュールも設定することをお勧めします。次のスクリプト例では、複数のファイルシステムで cron コマンドによって午前 3 時に定期的に実行されます。

```
outfile=/var/spool/fsadm/out.`/bin/date +'%m%d'`
for i in /home /home2 /project /db
do
  /bin/echo "Reorganizing $i"
  /usr/bin/time /opt/VRTS/bin/fsadm -t vxfs -e -E -s $i
  /usr/bin/time /opt/VRTS/bin/fsadm -t vxfs -s -d -D $i
done > $outfile 2>&1
```

10

エクステント属性

この章では以下の項目について説明しています。

エクステント属性について

Veritas File System（VxFS）では、隣接する 1 つ以上のブロックを使って、ファイルにディスク領域が割り当てられます。このようなブロックは、エクステントと呼ばれます。VxFS のアプリケーションインターフェースを使うと、プログラムで任意のファイルに対するエクステント割り当てをより細かく制御できます。ファイルに対するエクステント割り当てポリシーは、エクステント属性と呼ばれます。

VxFS の `getext` および`setext` コマンドを使うと、ファイルのエクステント属性を表示および設定できます。

`setext`（1）と `getext`（1）のマニュアルページを参照してください。

ファイルに対応付けられた基本的なエクステント属性には、領域予約と固定エクステントサイズがあります。ファイルの領域予約属性を設定することにより、ファイルに領域を事前に割り当てることができます。または、固定エクステントサイズを設定して、ファイルシステムのデフォルトの割り当てポリシーを無効にすることもできます。

p.284 の「領域予約: ファイルへの事前領域割り当て」を参照してください。

p.284 の「固定エクステントサイズ」を参照してください。

これらの属性の方針を決めるほかのポリシーは、割り当ての処理の中で決定します。

次の基準を指定できます。

- ファイル用に予約された領域が連続すること。
- 現在予約された領域を超えるファイルには、割り当てを実行しないこと。
- ファイルを閉じた時点で未使用の予約領域を解放すること。

- 領域は割り当てるが、領域の予約は実行しないこと。
- 割り当てられた領域を即座に反映して、ファイルのサイズを変更すること。

エクステント属性には、永続的なものと一時的なものがあります。永続的な属性は、そのファイルに関する情報としてディスク上に保存されます。一時的な属性は、ファイルを閉じた時点、またはシステムを再ブートした時点で失われます。永続的な属性はファイルのアクセス権限と類似しており、ファイルの i ノードに書き込まれます。ファイルのコピー、移動またはアーカイブを実行すると、ソースファイルの永続的な属性のみが新しいファイルに保持されます。

p.285 の「その他のエクステント属性の制御」を参照してください。

通常、ユーザーは領域を予約するためのエクステント属性のみを指定します。多くの属性は、特定の I/O パターンやアラインメントが実行されたディスク上のファイルシステム向けのアプリケーションで使われます。

p.620 の「Veritas File System I/O について」を参照してください。

領域予約: ファイルへの事前領域割り当て

VxFS（Veritas File System）では、データがファイルに書き込まれるときではなく、書き込みが要求されたときに、領域をファイルに事前に割り当てることができます。事前に割り当てられた領域は、ファイルシステム内の他のファイルに割り当てることはできません。VxFS では、ファイルに必要な領域が要求される前に、その領域がファイルに対応付けられます。そのため、ファイルシステムが予想外に領域不足になることを回避できます。

永続的な予約領域は、ファイルが切り捨てられても解放されません。解放する場合、予約の設定を解除するか、ファイルを削除して予約した領域を解放する必要があります。

固定エクステントサイズ

Veritas File System（VxFS）のデフォルトの割り当てポリシーでは、書き込み時に追加領域が必要になった場合、各種の方法に基づいて割り当て方法が決定されます。このポリシーでは、大規模な領域を割り当てて I/O パフォーマンスを最適化すること、およびファイルシステムの断片化を最小限に抑えることの両方を可能にします。VxFS は、ファイルシステムでそのデータに最も適した使用可能な領域から割り当てを行うことでこれらを実現します。

固定エクステントサイズを設定すると、ファイルのデフォルトの割り当てポリシーが無効になり、常に永続的な属性として機能します。固定エクステントサイズを使う場合は、アプリケーションに適したエクステントサイズを選択してください。VxFS のエクステントに基づく割り当てポリシーでは、ブロック単位のファイルシステムと比べて、間接ブロックの割り当てがかなり少なくなります。そのため、間接参照に必要なディスクアクセスの多くを省略できるという利点があります。ただし、小さいエクステントサイズでは、この利点を活かすことができません。

エクステントが大きいファイルでは、領域が連続するため、I/O 機能が向上する傾向にあります。ただし、未使用領域が大きいエクステントを細分化することにより、空き領域が断片化されるので、ファイルシステムの全般的なパフォーマンスが低下します。ファイルシステムの断片化を最小限に抑えられないと、ファイルの領域が連続しなくなるため、ファイルの I/O 機能が低下する場合があります。

固定エクステントサイズは、特に次のような状況で使います。

- ファイルが大きく、スパースファイルになっており、書き込みサイズが固定している場合。固定エクステントサイズを書き込みサイズの倍数に設定すると、ユーザーデータが含まれていないブロックによって無駄になる領域を最小限に抑えることができます。このような領域は、書き込み領域とエクステント領域が整列するように調整されていないために発生します(スパースファイルのデフォルトのエクステントサイズは 8K です)。
- ファイルが大きくて連続している場合。固定エクステントサイズを大きく設定すると、ファイル内のエクステント数を最小限に抑えることができます。

カスタムアプリケーションでも、ディスク上のシリンダまたはストライプ境界に揃えるようにエクステントを整列させる場合など、特定の理由から固定エクステントサイズを使う場合があります。

固定エクステントサイズと共有エクステントの連携方法

VxFS(Veritas File System)では、ファイルの最小割り当てサイズを制御するファイルの固定エクステントサイズオプションを設定できます。共有解除する必要がある共有エクステントがファイルにある場合、共有解除操作の一部である割り当ては、ファイルに設定されている固定エクステントサイズオプションを無視します。共有解除操作での割り当てサイズは、共有領域の書き込み操作のサイズによって決まります。

その他のエクステント属性の制御

エクステント属性のその他の制御機能を使うと、次の条件を設定できます

- アロケーションユニットを整列させるかどうか。
p.286 の「エクステント属性のアラインメント」を参照してください。
- アロケーションユニットを連続させるかどうか。
p.286 の「エクステント属性の連続性」を参照してください。
- 予約領域を超えてファイルを書き込めるようにするかどうか。
p.286 の「エクステント属性の予約領域を超えた書き込み操作」を参照してください。
- ファイルを閉じた時点で未使用の予約領域を解放するかどうか。
p.286 の「エクステント属性の予約領域の解放」を参照してください。
- 領域の予約をファイルの永続的な属性にするかどうか。
p.286 の「エクステント属性の予約領域の永続性」を参照してください。
- ファイルの予約領域をどの時点でそのファイルの一部にするか。

p.287 の「エクステント属性の予約領域を含むファイル」を参照してください。

エクステント属性のアラインメント

割り当てに特定のアラインメント条件を設定すると、特定の I/O パターンまたはディスクアラインメントに基づいてファイルの割り当てを実行できます。アラインメントを指定できるのは、固定エクステントサイズが設定されている場合のみです。割り当てにアラインメント条件を使う場合、専用のアプリケーションがそのファイルシステムを使うように設定する必要があります。

`setext`(1)のマニュアルページを参照してください。

p.620 の「Veritas File System I/O について」を参照してください。

エクステント属性の連続性

領域を予約する際に、割り当てが連続するように(同じエクステントになるように)指定できます。領域の連続性が最大限まで達成すると、I/O 機能が最適化されます。

メモ: ファイルシステムに固定エクステントサイズまたはアラインメントを設定すると、エクステントが適切なサイズではない場合(または適切に整列されていない場合)、領域が不足していることを示すエラーメッセージが返されます。ファイルシステムに十分な空き領域がある場合でも、固定エクステントサイズが大きいと、同様のエラーメッセージが返されます。

エクステント属性の予約領域を超えた書き込み操作

領域を予約する際に、書き込み操作によって予約領域の最後の使用可能なブロックが使われた場合、それ以上割り当てが実行されないように指定できます。この設定を`ulimit`コマンドの機能と同様に使って、ファイルのサイズが大きくなりすぎることを防ぐことができます。

エクステント属性の予約領域の解放

領域を予約する際に、ファイルを閉じた時点で未使用の予約領域を解放するように指定できます。そのファイルを開いているプロセスが複数存在する場合、そのすべてのプロセスがファイルを閉じるまで、予約領域は解放されません。

エクステント属性の予約領域の永続性

領域を予約する際に、予約領域をファイルの永続的な属性にしないことを指定できます。その場合、ファイルを閉じた時点で未使用の予約領域が破棄されます。

エクステント属性の予約領域を含むファイル

領域を予約する際に、予約領域を含めるようにファイルサイズを調整できます。通常、書き込み操作によって追加領域が必要になるまで、予約領域はファイルサイズには含まれません。即座にファイルサイズを変更するように予約領域を設定すると、数多くの一時ファイルが生成されます。ftruncate を使ってファイルサイズを増やす場合とは異なり、このような予約ではファイルに含まれているブロックのデータは初期化されません。したがって、以前別のファイルに含まれていたデータがそのファイルに含まれる可能性があります。そのため、この機能は適切な権限のあるユーザーに制限されます。適切な権限のないユーザーには、初期化されていないデータを表示しないようにする変形要求があります。

エクステント属性に関連するコマンド

Veritas File System（VxFS）には、エクステント属性を操作するコマンドとして、setext と getext があります。これらのコマンドを使って、任意のエクステント属性をファイルに設定したり、すでにファイルに対応付けられた属性を表示したりできます。

setext（1）と getext（1）のマニュアルページを参照してください。

VxFS 固有の vxdump および vxrestore コマンドでは、ファイルのバックアップ、リストア、移動またはコピーを実行する際にエクステント属性が保持されます。

これらのほとんどのコマンドには、ファイルのエクステント属性を操作するためのコマンドラインオプション（-e）が用意されています。予約された領域、固定エクステントサイズおよびエクステントのアラインメントといったエクステント属性情報を含む VxFS ファイルにこのオプションを使います。書き込み先のファイルシステムでエクステント属性がサポートされていない場合、書き込み元のファイルシステムと異なるブロックサイズを持つ場合、またはエクステント属性必要条件を満たす空きエクステントが不足している場合は、これらの属性情報が失われる可能性があります。

-e オプションを次の引数とともに使います。

warn	エクステント属性情報を保持できない場合、警告メッセージを出力します（デフォルト）。
force	エクステント属性情報を保持できない場合、処理を中断します。
ignore	エクステント属性情報を永久に破棄します。

次の例では、file1 という名前のファイルを作成し、ファイルに 2 GB のディスク領域を事前に割り当てます。

エクステント属性の設定の例

1 ファイル file1 を作成します。

```
# touch file1
```

2 ファイル file1 に 2 GB のディスク領域を事前に割り当てます。

```
# setext -t vxfs -r 2g -f chgsize file1
```

例では、-f chgsize オプションが指定されているため、VxFS はすぐに予約をファイルに組み込み、予約領域が組み込まれて増加したサイズとブロック数の情報を使って、ファイルの i ノードを更新します。root 権限を持つユーザーのみが -f chgsize オプションを使用できます。

次の例では、file1 という名前のファイルのエクステント属性の情報を取得します。

エクステント属性情報の取得の例

◆ ファイル file1 のエクステント属性の情報を取得します。

```
# getext -t vxfs file1
file1: Bsize  1024  Reserve 2097152  Extent Size  0
```

ファイル file1 には、1024 バイトのブロックサイズ、36 の予約済みブロック、3 ブロックの固定エクステントサイズが指定されており、3 ブロックの境界にすべてのエクステントが整列されています。現在の予約領域が完全に消費された後、ファイルサイズを増やすことはできません。領域の予約と固定エクステントサイズは、ファイルシステムのブロックサイズ単位で割り当てられます。

エクステント属性の保存の失敗について

エクステント属性を保持するコマンドを使ってファイルのコピー、移動またはアーカイブを実行すると、属性を失う場合があります。

その原因には、次のいずれかが考えられます。

- アーカイブからコピー、移動またはリストアしたファイルを受け取るファイルシステムが VxFS ファイルシステムではない場合。他のファイルシステムでは VxFS ファイルシステムのエクステント属性がサポートされていないため、ソースファイルの属性が移行中に失われます。
- コピー、移動またはリストアしたファイルを受け取るファイルシステムが VxFS タイプであるが、エクステント属性を保持できるだけの十分な空き領域がない場合。たとえば、50 K のファイルに対し 1 MB の領域を予約していたとします。受け取り先のファイルシステムの空き領域が 500 K の場合、ファイルは保存できても予約領域を保持できません。

- アーカイブからコピー、移動またはリストアしたファイルを受け取るファイルシステムは VxFS タイプであるが、ブロックサイズが異なる場合。元のファイルシステムと受け取り側のファイルシステムでブロックサイズが異なると、エクステント属性は保持できません。たとえば、元のファイルシステムのブロックサイズが 1024 バイト、受け取る側のファイルシステムのブロックサイズが 4096 バイト、ソースファイルに 3 ブロック(3072 バイト)の固定エクステントサイズが指定されているとします。この固定エクステントサイズは元のファイルシステムに適用されていますが、受け取り側のファイルシステムには変換できません。

  この例と同じ送り側および受け取り側のファイルシステムを使った場合でも、ファイルに 4 ブロックの固定エクステントサイズを指定すると属性を保持できます。送り側のファイルシステムで 4 ブロック(4096 バイト)のエクステントは、受け取り側で 1 ブロックエクステントに変換できるためです。

  さまざまな大きさのブロックサイズが使われたファイルシステムでは、コピー、移動またはリストアで属性が保持されない場合があります。任意のシステムのすべてのファイルシステムには、同じブロックサイズを使うことをお勧めします。

3

DMP を使ったマルチパスの管理

11

Dynamic Multi-Pathing の管理

この章では以下の項目について説明しています。

新しく追加されたディスクデバイスの検出と設定

新しいディスクをホストに物理的に接続する場合、または新しいファイバーチャネルデバイスをホストにゾーン化する場合は、`vxdctl enable` コマンドを使ってボリュームデバイスノードディレクトリを再構築し、DMP (Dynamic Multi-Pathing) の内部データベースを更新してシステムの新しい状態を反映できます。

DMP データベースを再構築するには、まず Linux に新しいディスクを認識させてから `vxdctl enable` コマンドを呼び出します。

`vxdisk scandisks` コマンドを使ってオペレーティングシステムのデバイスツリー内のデバイスをスキャンし、マルチパス化されたディスクの動的再構成を開始することもできます。

システムに追加された新しいデバイスのみを SFCFSHA でスキャンし、有効または無効になっているデバイスをスキャンしない場合は、次に示すように、どちらかのコマンドに `-f` オプションを指定します。

```
# vxdctl -f enable
# vxdisk -f scandisks
```

ただし、次の構成要素に変更がありシステム構成が修正になった場合は、完全なスキャンを開始します。

- インストール済み ASL（Array Support Library）。
- VxVM による使用から除外になっているデバイスの一覧。
- DISKS（JBOD）、SCSI3、外部デバイスの定義。

`vxdctl` (1M) マニュアルページを参照してください。

`vxdisk` (1M) マニュアルページを参照してください。

部分的なデバイス検出

DMP（Dynamic Multi-Pathing）では部分的なデバイス検出をサポートしており、検出プロセスから物理ディスクへのパスを有効または無効にすることができます。

`vxdisk scandisks` コマンドは、OS デバイスツリー内のデバイスを再スキャンして、DMP 再設定を実行します。`vxdisk scandisks` コマンドにパラメータを指定すると、部分的なデバイス検出を実行できます。たとえば、次のコマンドを指定すると、SFCFSHA は以前には認識されなかった新しく追加されたデバイスを検出します。

```
# vxdisk scandisks new
```

次の例では、ファブリックデバイスを検出します。

```
# vxdisk scandisks fabric
```

次のコマンドでは、デバイス `sdm` と `sdn` をスキャンします。

```
# vxdisk scandisks device=sdm,sdn
```

または、接頭辞 **!** を使って指定したデバイス以外をすべてスキャンすることもできます。

メモ: **!** 文字は一部のシェルでの特殊文字です。次の例では、bash シェルでこの文字をエスケープ処理する方法を示します。

```
# vxdisk scandisks ¥!device=sdm,sdn
```

また、論理コントローラまたは物理コントローラの一覧に接続している（または接続していない）デバイスをスキャンすることもできます。たとえば、次のコマンドを実行すると、指定した論理コントローラに接続しているデバイスを除く、すべてのデバイスを検出して設定できます。

```
# vxdisk scandisks ¥!ctlr=c1,c2
```

次のコマンドを実行すると、指定した物理コントローラに接続しているデバイスのみを検出できます。

```
# vxdisk scandisks pctlr=c1+c2
```

物理コントローラの一覧の各項目は、+ 文字で区切られます。

`vxdmpadm getctlr all` コマンドを使うと、物理コントローラの一覧を取得できます。

`vxdisk scandisks` コマンドに指定する引数は 1 つのみにする必要があります。複数のオプションを指定すると、エラーになります。

`vxdisk`(1M)マニュアルページを参照してください。

ディスクの検出とディスクアレイの動的な追加について

DMP(Dynamic Multi-Pathing)は ASL(Array Support Library)を使って、アレイ固有のマルチパスサポートを提供します。ASL は、動的にロード可能な共有ライブラリ(DDL のプラグイン)です。ASL は、デバイス検出中にデバイス属性を検出するためのハードウェア固有のロジックを実装します。DMP は、各ディスクアレイに関連付ける必要のある ASL を決定するためのデバイス検出層(DDL)を提供します。

場合によって、DMP は、LUN をディスク(JBOD)として処理することで、基本的なマルチパスとフェールオーバーの機能性を提供することもできます。

DMP がデバイスを要求する方法

あらゆるアレイに対して完全に最適化されたサポートを提供したり、より複雑なアレイタイプをサポートするために、DMP(Dynamic Multi-Pathing)はアレイ固有の ASL(Array Support Library)の使用を要求します。ASL は APM(Array Policy Module)とともに提供される場合があります。ASL と APM は、事実上、特定のアレイモデルと DMP の抱き合わせ販売を可能にするアレイ固有のプラグインです。

サポート対象のアレイの一覧については、ハードウェア互換性リストを参照してください。

https://www.veritas.com/support/en_US/article.000126344

デバイス検出中に、DDL はインストールされている各デバイスの ASL を調べて、どの ASL がデバイスを要求するかを見つけます。

デバイスを要求する ASL がなければ、DDL は対応する JBOD 定義があるかどうかを調べます。サポートされていないアレイの JBOD 定義を追加すれば、DMP はそのアレイに対してマルチパス機能を提供できるようになります。JBOD 定義があれば、DDL は DISKS カテゴリでデバイスを要求します。これにより、DMP が使う JBOD(物理ディスク)デバイスのリストに LUN が追加されます。JBOD 定義にキャビネット番号が含まれていれば、DDL はそのキャビネット番号を使って LUN をエンクロージャに分類します。

p.304 の「DISKS カテゴリへのサポートされていないディスクアレイの追加」を参照してください。

DMP は、ASL または JBOD 定義がなくても、非対称論理ユニットアクセス(ALUA)対応アレイに対して基本的なマルチパス機能を提供できます。DDL は ALUA ディスクのエンクロージャの一部として LUN を要求します。アレイタイプは ALUA として示されます。JBOD 定義を追加すれば、LUN をエンクロージャに分類することもできます。

ディスクカテゴリ

DMP (Dynamic Multi-Pathing) を使用して認識されるディスクアレイは ASL によってサポートされており、ディスクから返されるベンダー ID 文字列 (「HITACHI」など) に応じて分類されます。

DMP によってマルチパス化できる JBOD のディスクは DISKS カテゴリに分類されます。サポート外のアレイにあるディスクも DISKS カテゴリに分類されることがあります。

p.304 の「DISKS カテゴリへのサポートされていないディスクアレイの追加」を参照してください。

サポート対象のどのカテゴリにも属さず、DMP によるマルチパス化もできない JBOD のディスクは、OTHER_DISKS カテゴリに分類されます。

新しいディスクアレイの DMP サポートの追加

新しいタイプのディスクアレイに対するサポートパッケージを動的に追加できます。ベリタスが開発したサポートパッケージは ASL(アレイサポートライブラリ)の形式で提供されます。ベリタスは VRTSaslapm RPM に対する更新を通して新しいディスクアレイのサポートを提供します。更新済みの VRTSaslapm RPM がダウンロード可能かどうかを判断するには、ハードウェア互換性リストのテクニカルノートを参照してください。ハードウェア互換性リストには、VRTSaslapm RPM をインストールするための、最新のダウンロード用RPM と手順へのリンクが記載されています。システムがオンラインの間に VRTSaslapm RPM をアップグレードできます。アプリケーションを停止する必要はありません。

ハードウェア互換性リストにアクセスするには、次の URL に移動します。

https://www.veritas.com/support/en_US/article.000126344

各 VRTSaslapm RPM は、Storage Foundation Cluster File System High Availability のバージョンごとに固有です。インストールしたバージョンの Storage Foundation Cluster File System High Availability をサポートする VRTSaslapm RPM をインストールする必要があります。

VRTSaslapm RPM のインストール時に、新しいディスクアレイがシステムに接続されている必要はありません。

最新の VRTSaslapm RPM を削除する必要がある場合は、以前にインストールしたバージョンに戻せます。手順について詳しくは、『Veritas InfoScale トラブルシューティングガイド』を参照してください。

新しいディスクアレイ検出の有効化

vxdctl enable コマンドは、すべてのディスクデバイスおよびその属性をスキャンし、SFCFSHA のデバイスリストを更新し、新しいデバイスデータベースを使って DMP を再設定します。ホストを再ブートする必要はありません。

警告: このコマンドによって、DMP がアレイについて正しく設定されます。この操作を行わないと、VxVM でディスクへの独立したパスが別々のデバイスとして処理され、データの破損を引き起こす可能性があります。

新しいディスクアレイ検出を有効化するには、次の手順を実行します。

- 次のコマンドを入力します。

```
# vxdctl enable
```

サードパーティドライバの共存について

Storage Foundation Cluster File System High Availability (SFCFSHA) のサードパーティ製ドライバ (TPD) 共存機能を使うと、DMP の監視する能力を保持したまま、いくつかのサードパーティ製マルチパス化ドライバによって制御される I/O に DMP (Dynamic Multi-Pathing) をバイパスさせることができます。適切な ASL (Array Support Library) が使用可能でインストールされている場合は、仕様ファイルの設定や特殊コマンドの実行を行うことなく、TPD を使うデバイスを検出できます。SFCFSHA の TPD 共存機能では、サードパーティ製マルチパス化ドライバを一切変更せずに共存させることが可能です。

p.386 の「サードパーティ製ドライバ制御のエンクロージャに対するデバイスの命名の変更」を参照してください。

p.325 の「サードパーティ製のドライバにより制御されるデバイスに関する情報の表示」を参照してください。

デバイス検出層の管理方法

デバイス検出層 (DDL) を使うと、ディスクアレイの動的な追加が可能になります。DDL は、Storage Foundation Cluster File System High Availability (SFCFSHA) の操作に必要なディスクとその属性を検出します。

DDL は、次のタスクを実行する vxddladm ユーティリティを使って管理します。

- iSCSI デバイスなどの DDL で検出したすべてのデバイスの階層の一覧表示

- iSCSI を含むすべてのホストバスアダプタの一覧表示.
- ホストバスアダプタ上で設定されたポートの一覧表示.
- ホストバスアダプタから設定されたターゲットの一覧表示
- ホストバスアダプタから設定されたデバイスの一覧表示
- iSCSI 操作パラメータの取得または設定.
- サポートされているアレイタイプの一覧表示
- アレイのサポートの DDL への追加
- アレイのサポートの DDL からの削除
- 無効にされたディスクアレイに関する情報の一覧表示
- DISKS（JBOD）カテゴリ内の認識されているディスクの一覧表示
- DISKS カテゴリへの特定ベンダーのディスクの追加
- DISKS カテゴリからのディスクの削除
- 外部デバイスとしてのディスクの追加

次の項で、これらのタスクの詳細を説明します。

vxddladm（1M）マニュアルページを参照してください。

iSCSI を含むすべてのデバイスの一覧表示

iSCSI デバイスなどの DDL で検出したすべてのデバイスの階層を一覧表示できます。

iSCSI を含むすべてのデバイスを一覧表示するには

◆ 次のコマンドを入力します。

```
# vxddladm list
```

出力例を次に示します。

```
HBA fscsi0 (20:00:00:E0:8B:19:77:BE)
    Port fscsi0_p0 (50:0A:09:80:85:84:9D:84)
       Target fscsi0_p0_t0 (50:0A:09:81:85:84:9D:84)
           Device sda
. . .
HBA iscsi0 (iqn.1986-03.com.sun:01:0003ba8ed1b5.45220f80)
    Port  iscsi0_p0 (10.216.130.10:3260)
       Target  iscsi0_p0_t0 (iqn.1992-08.com.netapp:sn.84188548)

           Device sdb
           Device sdc
       Target  iscsi0_p0_t1 (iqn.1992-08.com.netapp:sn.84190939)
. . .
```

iSCSI を含むすべてのホストバスアダプタの一覧表示

iSCSI のアダプタを含めて、システムで設定されたすべてのホストバスアダプタ(HBA)についての情報を入手できます。

表 11-1 は HBA の情報を示します。

表 11-1 HBA の情報

フィールド	説明
ドライバ	HBA を制御するドライバ。
ファームウェア(firmware)	ファームウェアのバージョン。
検出(Discovery)	対象の検出で採用された方法。
状態(State)	デバイスがオンラインまたはオフラインのどちらであるか。
アドレス(Address)	ハードウェアアドレス。

iSCSI を含むすべてのホストバスアダプタを一覧表示するには

◆ 次のコマンドを使い、iSCSI デバイスを含めて、システムで設定されたすべての HBA を一覧表示します。

```
# vxddladm list hbas
```

ホストバスアダプタ上で設定されたポートの一覧表示

HBA に設定されたすべてのポートについての情報を取得できます。画面に次の情報が表示されます。

HBA-ID	親 HBA。
状態(State)	デバイスがオンラインまたはオフラインのどちらであるか。
アドレス(Address)	ハードウェアアドレス。

ホストバスアダプタ上で設定されたポートを一覧表示するには

◆ 次のコマンドを使って、HBA で設定されたポートを取得します。

```
# vxddladm list ports

PORT-ID   HBA-ID    STATE      ADDRESS
-----------------------------------------------------
c2_p0     c2        Online     50:0A:09:80:85:84:9D:84
c3_p0     c3        Online     10.216.130.10:3260
```

ホストバスアダプタまたはポートから設定されたターゲットの一覧表示

ホストバスアダプタまたはポートから設定されたすべてのターゲットについての情報を取得できます。

表 11-2 はターゲットの情報を示します。

表 11-2 ターゲットの情報

フィールド	説明
エイリアス(Alias)	エイリアス名(設定されている場合)。
HBA-ID	親 HBA またはポート。
状態	デバイスがオンラインまたはオフラインのどちらであるか。

フィールド	説明
アドレス	ハードウェアアドレス。

ターゲットを一覧表示するには

- すべてのターゲットを一覧表示するには、次のコマンドを使います。

```
# vxddladm list targets
```

出力例を次に示します。

```
TARGET-ID  ALIAS HBA-ID STATE  ADDRESS
-----------------------------------------------------------------------
c2_p0_t0   -     c2     Online 50:0A:09:80:85:84:9D:84
c3_p0_t1   -     c3     Online iqn.1992-08.com.netapp:sn.84190939
```

ホストバスアダプタまたはポートから設定されたターゲットを一覧表示するには

- 次のコマンドを使えば、HBA またはポートに基づいてフィルタ処理できます。

```
# vxddladm list targets [hba=hba_name|port=port_name]
```

たとえば、特定の HBA から設定されたターゲットを取得するには、次のコマンドを使います。

```
# vxddladm list targets hba=c2

TARGET-ID  ALIAS  HBA-ID   STATE      ADDRES
-------------------------------------------------------------
c2_p0_t0   -      c2       Online     50:0A:09:80:85:84:9D:84
```

ホストバスアダプタとターゲットから設定されたデバイスの一覧表示

ホストバスアダプタから設定されたすべてのデバイスについての情報を取得できます。

表 11-3 はデバイスの情報を示します。

表 11-3　　デバイスの情報

フィールド	説明
Device	デバイス名。
Target-ID	親ターゲット。
状態	デバイスがオンラインまたはオフラインのどちらであるか。

フィールド	説明
DDL の状況	デバイスが DDL によって要求されるかどうか。要求される場合、出力には ASL 名も表示されます。

ホストバスアダプタから設定されたデバイスを一覧表示するには

- 設定されたデバイスを取得するには、次のコマンドを使います。

```
# vxddladm list devices

Device    Target-ID    State    DDL status (ASL)
-------------------------------------------------------
sda       fscsi0_p0_t0 Online   CLAIMED (libvxemc.so)
sdb       fscsi0_p0_t0 Online   SKIPPED (libvxemc.so)
sdc       fscsi0_p0_t0 Offline  ERROR
sdd       fscsi0_p0_t0 Online   EXCLUDED
sde       fscsi0_p0_t0 Offline  MASKED
```

ホストバスアダプタとターゲットから設定されたデバイスを一覧表示するには

- 特定の HBA とターゲットから設定されたデバイスを取得するには、次のコマンドを使います。

```
# vxddladm list devices target=target_name
```

iSCSI 操作パラメータの取得または設定

DDL は、iSCSI デバイスパスのパフォーマンスに影響を与える特定のパラメータを設定および表示するためのインターフェースを提供します。ただし、基盤となる OS フレームワークがこの値の設定をサポートする必要があります。OS のサポートがない場合、`vxddladm set` コマンドはエラーを返します。

表 11-4 iSCSI のデバイスのパラメータ

パラメータ	デフォルト値	最小値	最大値
DataPDUInOrder	yes	no	yes
DataSequenceInOrder	yes	no	yes
DefaultTime2Retain	20	0	3600
DefaultTime2Wait	2	0	3600
ErrorRecoveryLevel	0	0	2

パラメータ	デフォルト値	最小値	最大値
FirstBurstLength	65535	512	16777215
InitialR2T	yes	no	yes
ImmediateData	yes	no	yes
MaxBurstLength	262144	512	16777215
MaxConnections	1	1	65535
MaxOutStandingR2T	1	1	65535
MaxRecvDataSegmentLength	8182	512	16777215

特定の iSCSI ターゲットのイニシエータ上の iSCSI 操作パラメータを取得するには

◆ 次のコマンドを入力します。

```
# vxddladm getiscsi target=tgt-id {all | parameter}
```

このコマンドを使えば、すべての iSCSI 操作パラメータを取得できます。

```
# vxddladm getiscsi target=c2_p2_t0
```

出力例を次に示します。

```
PARAMETER               CURRENT  DEFAULT    MIN      MAX
-------------------------------------------------------
DataPDUInOrder              yes      yes      no      yes
DataSequenceInOrder         yes      yes      no      yes
DefaultTime2Retain           20       20       0     3600
DefaultTime2Wait              2        2       0     3600
ErrorRecoveryLevel            0        0       0        2
FirstBurstLength          65535    65535     512 16777215
InitialR2T                  yes      yes      no      yes
ImmediateData               yes      yes      no      yes
MaxBurstLength           262144   262144     512 16777215
MaxConnections                1        1       1    65535
MaxOutStandingR2T             1        1       1    65535
MaxRecvDataSegmentLength   8192     8182     512 16777215
```

特定の iSCSI ターゲットのイニシエータ上で iSCSI 操作パラメータを設定するには

◆ 次のようにコマンドを入力します。

```
# vxddladm setiscsi target=tgt-idparameter=value
```

サポートされているすべてのディスクアレイの一覧表示

この手順を使って、vxddladm コマンドの他の形とともに使う vid 属性と pid 属性の値を取得します。

サポートされているすべてのディスクアレイを一覧表示するには、次の作業を実行します。

- 次のコマンドを実行します。

```
# vxddladm listsupport all
```

Array Support Library（ASL）の詳細の表示

DMP（Dynamic Multi-Pathing）により、Array Support Library（ASL）の詳細を表示できます。

Array Support Library（ASL）の詳細を表示するには

- 次のようにコマンドを入力します。

```
# vxddladm listsupport libname=library_name.so
```

このコマンドを実行すると、ベンダー ID（VID）、アレイのプロダクト ID（PID）、アレイタイプ（A/A または A/P など）とアレイの名前が表示されます。出力例を次に示します。

```
# vxddladm listsupport libname=libvxfujitsu.so
ATTR_NAME             ATTR_VALUE
================================================
LIBNAME               libvxfujitsu.so
VID                   vendor
PID                   GR710, GR720, GR730
                      GR740, GR820, GR840
ARRAY_TYPE            A/A, A/P
ARRAY_NAME            FJ_GR710, FJ_GR720, FJ_GR730
                      FJ_GR740, FJ_GR820, FJ_GR840
```

ディスクアレイライブラリのサポートの無効化

特定のディスクアレイライブラリに依存するディスクアレイのサポートを無効にすることができます。また、特定のベンダーからディスクアレイのサポートを無効にすることもできます。

ディスクアレイライブラリのサポートを無効化するには

- ディスクアレイライブラリのサポートを無効化するには、次のコマンドでアレイライブラシを指定します。

```
# vxddladm excludearray libname=libname
```

次の例に示すように、特定ベンダーのディスクアレイのサポートを無効にすることもできます。

```
# vxddladm excludearray vid=ACME pid=X1

# vxdisk scandisks
```

無効にされたディスクアレイライブラリのサポートの有効化

特定のディスクアレイライブラリに依存するすべてのアレイのサポートを以前に無効にした場合は、この手順を使ってそれらのアレイのサポートを有効にします。この手順では、エクスクルードリストからライブラリを削除します。

無効にされたディスクアレイライブラリのサポートを有効化するには

- 特定のディスクアレイライブラリに依存するすべてのアレイのサポートを無効にした場合は、`includearray` キーワードを使って除外ファイルリストのエントリを削除できます。

```
# vxddladm includearray libname=libname
```

このコマンドは、アレイライブラリをデータベースに追加し、そのライブラリをデバイスの検出に使えるようにします。

```
# vxdisk scandisks
```

無効にされたディスクアレイの一覧表示

現在 VxVM（Veritas Volume Manager）による使用が無効化されているすべてのディスクアレイを一覧表示するには

- 次のコマンドを入力します。

```
# vxddladm listexclude
```

DISKS カテゴリで認識されているディスクの一覧表示

DISKS（**JBOD**）カテゴリ内の認識されているディスクを一覧表示するには

- 次のようにコマンドを入力します。

```
# vxddladm listjbod
```

DISKS カテゴリへのサポートされていないディスクアレイの追加

アレイで利用できる Array Support Library（ASL）がない場合、JBOD デバイスとしてディスクアレイを追加する必要があります。

JBOD は、指定されていないかぎり、アクティブ/アクティブ（A/A）であると見なされます。適切な ASL が利用できない場合、A/A-A、A/P または A/PF アレイは、パス遅延や I/O エラーを避けるため、アクティブ/パッシブ（A/P）JBOD として認識される必要があります。JBOD が ALUA 対応であれば、JBOD は ALUA アレイとして追加されます。

p.43 の「DMP の動作方法」を参照してください。

警告: この手順は、VxVM（Veritas Volume Manager）ではサポートされていないアレイに DMP を正しく設定するためのものです。この操作を行わないと、VxVM でディスクへの独立したパスが別々のデバイスとして処理され、データの破損を引き起こす可能性があります。

DISKS カテゴリへのサポートされていないディスクアレイを追加するには、次の手順を実行します。

1 次のコマンドを使って、アレイ内にあるディスクのベンダー ID とプロダクト ID を確認します。

```
# /etc/vx/diag.d/vxscsiinq device_name
```

device_name は、アレイ内にあるいずれかのディスクのデバイス名です。このコマンドで出力されるベンダー ID (VID) とプロダクト ID (PID) の値を書き留めてください。Fujitsu 製のディスクの場合は、表示されるシリアル番号の文字数も書き留めておきます。

次の出力例では、ベンダー ID は SEAGATE、プロダクト ID は ST318404LSUN18G です。

```
Vendor id (VID)      : SEAGATE
Product id (PID)     : ST318404LSUN18G
Revision             : 8507
Serial Number        : 0025T0LA3H
```

2 データベースなど、アレイ上で設定された VxVM ボリュームにアクセスしているアプリケーションをすべて停止し、アレイに設定されているすべてのファイルシステムと Storage Checkpoint のマウントを解除します。

3 アレイがタイプ A/A-A、A/P または A/PF の場合、auto-trespass モードで設定する必要があります。

4 次のコマンドを入力して、新しい JBOD カテゴリを追加します。

```
# vxddladm addjbod vid=vendorid [pid=productid] ¥
[serialnum=opcode/pagecode/offset/length] ¥
[cabinetnum=opcode/pagecode/offset/length] policy={aa|ap}]
```

vendorid および *productid* は、前の手順で検出した VID および PID の値です。たとえば、*vendorid* は FUJITSU、IBM、または SEAGATE になります。Fujitsu 製のデバイスの場合は、length 引数にシリアル番号の文字数 (10 など) も指定する必要があります。アレイがタイプ A/A-A、A/P、または A/PF の場合は、policy=ap 属性も指定する必要があります。

前述の例に続けて、このタイプのディスクアレイを JBOD として定義するコマンドは次のようになります。

```
# vxddladm addjbod vid=SEAGATE pid=ST318404LSUN18G
```

5 アレイを VxVM 制御下に置くには、vxdctl enable コマンドを使います。

```
# vxdctl enable
```

p.295 の「新しいディスクアレイ検出の有効化」を参照してください。

6 アレイがサポートされたことを確認するには、次のコマンドを入力します。

```
# vxddladm listjbod
```

前述の例のアレイに対する、このコマンドの出力例を次に示します。

```
VID      PID          SerialNum            CabinetNum    Policy
                  (Cmd/PageCode/off/len) (Cmd/PageCode/off/len)
==============================================================
SEAGATE ALL PIDs  18/-1/36/12           18/-1/10/11    Disk
SUN     SESS01    18/-1/36/12           18/-1/12/11    Disk
```

7 アレイが認識されたことを確認するには、以下のサンプルアレイの出力例に示されているように vxdmpadm listenclosure コマンドを使います。

```
# vxdmpadm listenclosure
```

```
ENCLR_NAME ENCLR_TYPE ENCLR_SNO STATUS    ARRAY_TYPE LUN_COUNT FIRMWARE
============================================================================
Disk       Disk       DISKS     CONNECTED Disk       2         -
```

アレイのエンクロージャ名およびエンクロージャタイプの両方が「Disk」に設定されていることがわかります。アレイ内にあるディスクを表示するには、vxdisk list コマンドを使います。

```
# vxdisk list
DEVICE       TYPE            DISK         GROUP        STATUS
punr710vm04_disk_1 auto:none       -            -            online invalid
punr710vm04_disk_2 auto:none       -            -            online invalid
punr710vm04_disk_3 auto:none       -            -            online invalid
punr710vm04_disk_4 auto:none       -            -            online invalid
sda                auto:none       -            -            online invalid
xiv0_9148          auto:none       -            -            online invalid thinrclm
...
```

8 DMP のパスが認識されたことを確認するには、以下のサンプルアレイの出力例に示されているように vxdmpadm getdmpnode コマンドを使います。

```
# vxdmpadm getdmpnode enclosure=Disk
NAME                STATE        ENCLR-TYPE   PATHS  ENBL  DSBL  ENCLR-NAME
================================================================================
punr710vm04_disk_1   ENABLED      Disk         1      1     0     disk
punr710vm04_disk_2   ENABLED      Disk         1      1     0     disk
punr710vm04_disk_3   ENABLED      Disk         1      1     0     disk
punr710vm04_disk_4   ENABLED      Disk         1      1     0     disk
sda                  ENABLED      Disk         1      1     0     disk
...
```

この例の出力では、アレイ内のディスクへのパスが 2 つあることが示されています。

詳しくは、vxddladm help addjbod コマンドを入力してください。

vxddladm (1M) マニュアルページを参照してください。

vxdmpadm (1M) マニュアルページを参照してください。

DISKS カテゴリからのディスクの削除

DISKS カテゴリからディスクを削除するには、このセクションの手順を使います。

ディスクを DISKS カテゴリから削除するには

◆ vxddladm コマンドに rmjbod キーワードを付けて使います。次の例は、ベンダー ID が SEAGATE のディスクを削除するコマンドを示しています。

```
# vxddladm rmjbod vid=SEAGATE
```

外部デバイス

DDL（Device Discovery Layer）では、RAM ディスクなど、自動検出不能な一部のデバイスについては検出できないことがあります。このような外部デバイスは、vxddladm addforeign コマンドを使うことにより、VxVM（Veritas Volume Manager）で simple ディスクとして使えるようになります。このコマンドを使うと、I/O 処理に DMP を使う必要もなくなります。次の例は、指定したディレクトリにブロックデバイスおよびキャラクタデバイスのエントリを追加する方法を示しています。

```
# vxddladm addforeign blockdir=/dev/foo/dsk chardir=/dev/foo/rdsk
```

ブロックデバイスまたはキャラクタデバイスがドライバによってサポートされていない場合は、次のようにコマンドから省略できます。

```
# vxddladm addforeign blockdir=/dev/foo/dsk
```

このコマンドを実行すると、デフォルトでは、OS によって管理されるデバイスツリー内の、自動検出機構で検出されたデバイスと一致するエントリがすべて無効になります。この動作は、vxddladm(1M) マニュアルページの説明に従い、-f オプションおよび -n オプションを使って上書きできます。

エントリを追加した外部デバイスは、vxdisk scandisks コマンドまたは vxdctl enable コマンドのいずれかを使うと simple ディスクとして検出されるようになります。検出されたこれらのディスクは、自動設定されたディスクと同じ方法で使えます。

外部デバイスのサポートには、次の制限があります。

- 外部デバイスは常に単一パスのディスクと見なされます。自動検出ディスクとは異なり、DMP ノードがありません。
- クラスタ環境の共有ディスクグループについてはサポートされていません。スタンドアロンホストシステムのみがサポートされています。
- PGR（Persistent Group Reservation）操作についてはサポートされていません。
- DMP の制御下にないため、障害が発生したディスクを自動的に有効にすること、および DMP 管理コマンドを使うことができません。
- エンクロージャ情報を VxVM から確認できません。これらのデバイスを使って作成されたディスクグループは可用性が低くなります。

- I/O フェンシング機能とクラスタファイルシステム機能は、外部デバイスではサポートされません。

デバイスを VxVM で非表示にする

VxVM（Veritas Volume Manager）からデバイスを無効にするには、次の手順を使います。DMP（Dynamic Multi-Pathing）ドライバ（vxdmp）によってデバイスがマルチパス化されないようにするオプションは非推奨です。

VxVM からのデバイスを無効化するには

1 vxdiskadm コマンドを実行し、メインメニューで[VxVM **が使用するデバイスのパス、またはマルチパスの無効化**(Prevent multipathing/Suppress devices from VxVM's view)]を選択します。続行するかどうかの確認を求めるプロンプトが表示されます。

2 次のオプションから、実行する操作を選択します。

オプション 1	VxVM から特定のコントローラ上のすべてのパスを無効にします。
オプション 2	VxVM から特定のパスを無効にします。
オプション 3	特定のベンダー ID とプロダクト ID の組み合わせに一致するディスクを VxVM から無効にします。 ルートディスクは無効にできません。 外部ディスクのベンダー ID とプロダクト ID が、ルートディスクと同じベンダー ID とプロダクト ID で、ルートディスクが VxVM でカプセル化されている場合、操作は失敗します。
オプション 4	ディスクへのパスをすべて無効にします。
オプション 5 非推奨	VxVM による特定のコントローラ上のすべてのディスクのマルチパス化を無効にします。 この操作は、サポート外の設定になる可能性があるため非推奨です。
オプション 6 非推奨	VxVM によるディスクのマルチパス化を無効にします。指定したパスに対応するディスクは、OTHER_DISKS カテゴリで認識され、マルチパス化されません。 この操作は、サポート外の設定になる可能性があるため非推奨です。
オプション 7 非推奨	特定のベンダー ID とプロダクト ID の組み合わせに一致するディスクのマルチパス化を無効にします。特定のベンダー ID とプロダクト ID の組み合わせに対応するディスクは、OTHER_DISKS カテゴリで認識され、マルチパス化されません。 この操作は、サポート外の設定になる可能性があるため非推奨です。
オプション 8	現在無効になっているデバイスを一覧表示します。

デバイスの VxVM での表示

デバイスを VxVM(Veritas Volume Manager)で再び表示するには、この手順を実行します。Dynamic Multi-Pathing(DMP)ドライバ(vxdmp)によるマルチパス化を許可するオプションは廃止されました。

デバイスを VxVM で表示するには、次の手順を行います。

1 `vxdiskadm` コマンドを実行し、メインメニューで[`VxVM` **が使用するデバイスのパス、またはマルチパスの有効化**(`Allow multipathing/Unsuppress devices from VxVM's view`)]を選択します。続行するかどうかの確認を求めるプロンプトが表示されます。

2 次のオプションから、実行する操作を選択します。

オプション 1	VxVM から特定のコントローラ上のすべてのパスを有効にします。
オプション 2	VxVM から特定のパスを有効にします。
オプション 3	特定のベンダー ID とプロダクト ID の組み合わせに一致するディスクを VxVM から有効にします。
オプション 4	ディスクへのパスをすべて有効にします。
オプション 5 非推奨	特定のコントローラ上のパスを持つすべてのディスクのマルチパス化を許可します。 この操作は非推奨です。
オプション 6 非推奨	VxVM によるディスクのマルチパス化を許可します。 この操作は非推奨です。
オプション 7 非推奨	特定のベンダー ID とプロダクト ID の組み合わせに一致するディスクのマルチパス化を許可します。 この操作は非推奨です。
オプション 8	現在無効になっているデバイスを一覧表示します。

コントローラとストレージプロセッサに対する I/O の有効化と無効化について

DMP を使うと、ホストバスアダプタ(HBA)コントローラ、またはストレージプロセッサのアレイポートを介して I/O を無効にして、管理上の操作を実行できるようになります。この機能は、ホスト上の HBA コントローラ、または SFCFSHA に対応しているディスクアレイに接続したアレイポートでメンテナンスを実施するときに使うことができます。HBA コントローラまたはアレイポートの I/O 操作は、管理タスクが完了した後で有効に戻すことができます。この一連の操作は、`vxdmpadm` コマンドを使って実行できます。

アクティブ/アクティブタイプのディスクアレイの場合は、HBA コントローラまたはアレイポートを介した I/O を無効にすると、I/O は残りのパス上で継続されます。アクティブ/パッシブタイプのディスクアレイの場合は、HBA コントローラまたはアレイポートを介した I/O を無

効にするとすべてのプライマリパスが無効になるため、DMP はセカンダリパスにフェールオーバーし、I/O はそれらのパス上で継続されます。

管理操作が終了したら、vxdmpadm コマンドを使って HBA コントローラまたはアレイポートを経由したパスを再び有効にしてください。

p.344 の「パス、コントローラ、アレイポート、DMP ノードに対する I/O の無効化」を参照してください。

p.345 の「パス、コントローラ、アレイポート、DMP ノードに対する I/O の有効化」を参照してください。

一定の再設定操作は動的にオンラインでも実行できます。

DMP データベース情報の表示について

vxdmpadm コマンドを使って DMP データベース情報の一覧表示し、他の管理タスクを実行できます。このコマンドを実行すると、ディスクに接続されているすべてのコントローラおよび DMP データベースに保存されている他の関連情報を一覧表示することができます。この情報は、システムのハードウェアの配置および有効化や無効にする必要のあるコントローラの判定に役立てることができます。

vxdmpadm コマンドは、ディスクアレイのシリアル番号、ディスクアレイに接続されている DMP デバイス (ディスク)、特定のコントローラ、エンクロージャ、またはアレイポートに接続されているパスなど、有用な情報も示します。

p.315 の「vxdmpadm ユーティリティを使った DMP の管理」を参照してください。

ディスクへのパスの表示

vxdisk コマンドは、特定のメタデバイスに関するマルチパス情報を表示するのに使われます。メタデバイスは、システムの HBA コントローラを通じた物理パスを複数持つ物理ディスクを表すデバイスです。DMP（Dynamic Multi-Pathing）では、システム内のすべての物理ディスクを、1 つ以上の物理パスを持つメタデバイスとして表します。

システム上のマルチパス情報を表示するには

◆ vxdisk path コマンドを実行すると、次に示すように、システム上のデバイスパス、ディスクアクセス名、ディスクメディア名、ディスクグループの関係が表示されます。

```
# vxdisk path

SUBPATH      DANAME     DMNAME     GROUP    STATE
sda          sda        mydg01     mydg     ENABLED
sdi          sdi        mydg01     mydg     ENABLED
sdb          sdb        mydg02     mydg     ENABLED
sdj          sdj        mydg02     mydg     ENABLED
 .
 .
 .
```

この出力例では、2 つのディスク(mydg01 と mydg02)にそれぞれ 2 つのパスが存在し、各ディスクが ENABLED 状態であることが示されています。

特定のメタデバイスのマルチパス情報を表示するには

1 次のコマンドを実行します。

```
# vxdisk list devicename
```

たとえば、デバイス sdl のマルチパス情報を表示するには、次のコマンドを使います。

```
# vxdisk list sdl
```

vxdisk list コマンドからの出力では、次の例に示すようにマルチパスの情報が表示されます。

```
Device:    sdl
devicetag: sdl
type:      sliced
hostid:    sys1
.
.
.
Multipathing information:
numpaths:   2
sdl    state=enabled     type=primary
sdp    state=disabled    type=secondary
```

numpaths 行では、デバイスに対して 2 つのパスがあることが示されています。出力の[Multipathing information]セクションの次の 2 行では、1 つのパスがアクティブ(state=enabled)で、もう 1 つのパスでエラーが発生している(state=disabled)ことを示します。

type フィールドは、EMC CLARiiON、Hitachi HDS 9200 および 9500、Sun StorEdge 6xxx および Sun StorEdge T3 アレイといったアクティブ/パッシブタイプのディスクアレイ上のディスクの場合に表示されます。このフィールドでは、ディスクへのパスがプライマリパスであるか、セカンダリパスであるかが表示されます。

type フィールドは、EMC Symmetrix、Hitachi HDS 99xx、Sun StorEdge 99xx シリーズおよび IBM ESS シリーズといったアクティブ/アクティブタイプのディスクアレイ上のディスクの場合には表示されません。このタイプのディスクアレイに、プライマリパスおよびセカンダリパスといった概念は存在しません。

2 また、次のコマンドを使ってマルチパス情報を表示できます。

```
# vxdmpadm getsubpaths dmpnodename=devicename
```

たとえば、emc_clariion0_431 のマルチパス情報を表示するには、次のコマンドを使います。

```
# # vxdmpadm getsubpaths dmpnodename=emc_clariion0_431
```

vxdmpadm getsubpaths コマンドの通常の出力例は、次のとおりです。

```
NAME   STATE[A]    PATH-TYPE[M]          CTLR-NAME  ENCLR-TYPE    ENCLR-NAME    ATTRS
PRIORITY
=============================================================================================
sdac   ENABLED     Active/Non-Optimized  c6         EMC_CLARiiON  emc_clariion0  -
-
sdam   ENABLED(A)  Active/Optimized(P)   c6         EMC_CLARiiON  emc_clariion0  -
-
sdi    ENABLED     Active/Non-Optimized  c1         EMC_CLARiiON  emc_clariion0  -
-
sds    ENABLED(A)  Active/Optimized(P)   c1         EMC_CLARiiON  emc_clariion0  -
-
```

vxdmpadm ユーティリティを使った DMP の管理

vxdmpadm ユーティリティは、DMP（Dynamic Multi-Pathing）に対するコマンドライン管理インターフェースです。

vxdmpadm ユーティリティを使って、次のタスクを実行できます。

- 特定のパスに対する DMP デバイス名の取得
  p.317 の「DMP ノードに関する情報の取得」を参照してください。
- DMP ノードについての統合された情報の表示.
  p.318 の「DMP ノードについての統合された情報の表示」を参照してください。
- LUN グループのメンバーの表示
  p.319 の「LUN グループのメンバーの表示」を参照してください。
- DMP デバイスノード、HBA コントローラ、エンクロージャ、アレイポート下のすべてのパスの一覧表示
  p.319 の「DMP ノード、コントローラ、エンクロージャ、アレイポートによって制御されるパスの表示」を参照してください。
- ホスト上の HBA コントローラに関する情報の表示
  p.322 の「コントローラに関する情報の表示」を参照してください。

- エンクロージャに関する情報の表示
  p.324 の「エンクロージャに関する情報の表示」を参照してください。
- エンクロージャのストレージプロセッサに接続したアレイポートに関する情報の表示
  p.324 の「アレイポートに関する情報の表示」を参照してください。
- ALUA アレイの非対称アクセス状態の表示。
- 他社のマルチパス化ドライバにより制御されるデバイスに関する情報の表示
  p.325 の「サードパーティ製のドライバにより制御されるデバイスに関する情報の表示」を参照してください。
- 拡張されるデバイスの属性の表示
  p.326 の「拡張デバイス属性の表示」を参照してください。
- VxVM の制御下におけるデバイスの無効化と有効化
  p.328 の「VxVM の制御下におけるデバイスの無効化と有効化」を参照してください。
- DMP ノード、エンクロージャ、パス、コントローラの I/O 統計情報の収集
  p.329 の「I/O 統計情報の収集と表示」を参照してください。
- エンクロージャへのパスに関する属性の設定
  p.335 の「エンクロージャへのパスに関する属性の設定」を参照してください。
- デバイスまたはエンクロージャの冗長レベルの表示.
  p.336 の「デバイスまたはエンクロージャの冗長レベルの表示」を参照してください。
- アクティブパスの最小数の指定.
  p.337 の「アクティブパスの最小数の指定」を参照してください。
- エンクロージャへのパスに対して使われる I/O ポリシーの表示または設定
  p.338 の「I/O ポリシーの指定」を参照してください。
- システム上のパス、HBA コントローラ、アレイポートに対する I/O の有効化または無効化
  p.344 の「パス、コントローラ、アレイポート、DMP ノードに対する I/O の無効化」を参照してください。
- エンクロージャ名の変更
  p.346 の「エンクロージャ名の変更」を参照してください。
- I/O 要求エラーに対する DMP の応答方法の設定
  p.347 の「I/O エラーに対する応答の設定」を参照してください。
- I/O 調整機構の設定
  p.348 の「I/O 調整機構の設定」を参照してください。
- DMP パスリストアスレッドの動作の制御
  p.351 の「DMP パスリストアポリシーの設定」を参照してください。
- アレイポリシーモジュール（Array Policy Modules）の設定.

p.353 の「アレイポリシーモジュール（Array Policy Modules）の設定」を参照してください。

- DMP で使われる各種のチューニングパラメータの値の取得または設定
  p.1115 の「DMP チューニングパラメータ」を参照してください。

vxdmpadm(1M) のマニュアルページを参照してください。

DMP ノードに関する情報の取得

次のコマンドを実行すると、指定した物理パスを制御する DMP（Dynamic Multi-Pathing）ノードが表示されます。

```
# vxdmpadm getdmpnode nodename=pathname
```

nodename 属性には物理パスを指定できます。この物理パスはデバイスのディレクトリに一覧表示された有効なパスである必要があります。

デバイスのディレクトリは /dev ディレクトリです。

このコマンドの出力は、次の出力例のようになります。

```
# vxdmpadm getdmpnode nodename=sdbc

NAME               STATE   ENCLR-TYPE   PATHS ENBL DSBL ENCLR-NAME
==========================================================================
emc_clariion0_89 ENABLED EMC_CLARiiON 6     6    0    emc_clariion0
```

LUN のシリアル番号とアレイのボリューム ID を表示するには、-v オプションを使います。

```
# vxdmpadm -v getdmpnode nodename=sdbc

NAME             STATE  ENCLR-TYPE  PATHS ENBL DSBL ENCLR-NAME SERIAL-NO ARRAY_VOL_ID
=============================================================================================
emc_clariion0_89 ENABLED EMC_CLARiiON 6   6    0   emc_clariion0 600601601 893
```

指定したエンクロージャのすべての DMP ノードのリストを取得するには、getdmpnode で enclosure 属性を使います。

```
# vxdmpadm getdmpnode enclosure=emc_clariion0

NAME               STATE    ENCLR-TYPE   PATHS  ENBL  DSBL  ENCLR-NAME
=============================================================================
emc_clariion0_429  ENABLED  EMC_CLARiiON 4      4     0     emc_clariion0
emc_clariion0_430  ENABLED  EMC_CLARiiON 4      4     0     emc_clariion0
emc_clariion0_431  ENABLED  EMC_CLARiiON 4      4     0     emc_clariion0
emc_clariion0_432  ENABLED  EMC_CLARiiON 4      4     0     emc_clariion0
```

特定の DMP ノードの DMP 情報を表示するには、getdmpnode で dmpnodename 属性を使います。

```
# vxdmpadm getdmpnode dmpnodename=emc_clariion0_158

NAME               STATE   ENCLR-TYPE   PATHS ENBL DSBL ENCLR-NAME
===================================================================
emc_clariion0_158 ENABLED EMC_CLARiiON 1     1    0    emc_clariion0
```

DMP ノードについての統合された情報の表示

vxdmpadm list dmpnode コマンドは DMP（Dynamic Multi-Pathing）ノードの詳細情報を表示します。情報には、エンクロージャ名、LUN シリアル番号、ポート ID 情報、デバイス属性などが含まれます。

次のコマンドは、システムのすべての DMP ノードに関する統合された情報を表示します。

```
# vxdmpadm list dmpnode all
```

指定したエンクロージャのすべての DMP ノードのリストを取得するには、list dmpnode で enclosure 属性を使います。

```
# vxdmpadm list dmpnode enclosure=enclosurename
```

たとえば、次のコマンドを実行すると、enc0 エンクロージャ内のすべての DMP ノードに関する統合された情報が表示されます。

```
# vxdmpadm list dmpnode enclosure=enc0
```

特定の DMP ノードの DMP 情報を表示するには、list dmpnode で dmpnodename 属性を使います。DMP ノードは名前またはパス名で指定できます。指定した DMP ノードの詳細情報には、一覧表示された DMP ノードの各サブパスについてのパス情報が含まれています。

パスの状態は、障害により無効になったパスと、管理上の目的で手動で無効にされたパスでは異なります。vxdmpadm disable コマンドを使って手動で無効にされたパスは、disabled(m) として表示されます。

```
# vxdmpadm list dmpnode dmpnodename=dmpnodename
```

たとえば、次のコマンドを実行すると、DMP ノード emc_clariion0_158 についての統合された情報が表示されます。

```
# vxdmpadm list dmpnode dmpnodename=emc_clariion0_158

dmpdev      = emc_clariion0_158
state       = enabled
```

```
enclosure    = emc_clariion0
cab-sno      = CK200070400359
asl          = libvxCLARiiON.so
vid          = DGC
pid          = DISK
array-name   = EMC_CLARiiON
array-type   = CLR-A/PF
iopolicy     = MinimumQ
avid         = 158
lun-sno      = 600601601A141B001D4A32F92B49DE11
udid         = DGC%5FDISK%5FCK200070400359%5F600601601A141B001D4A32F92B49DE11
dev-attr     = lun
###path      = name state type transport ctlr hwpath aportID aportWWN attr
path         = sdck enabled(a) primary FC c2 c2 A5 50:06:01:61:41:e0:3b:33 -
path         = sdde enabled(a) primary FC c2 c2 A4 50:06:01:60:41:e0:3b:33 -
path         = sdcu enabled secondary FC c2 c2 B4 50:06:01:68:41:e0:3b:33 -
path         = sdbm enabled secondary FC c3 c3 B4 50:06:01:68:41:e0:3b:33 -
path         = sdbw enabled(a) primary FC c3 c3 A4 50:06:01:60:41:e0:3b:33 -
path         = sdbc enabled(a) primary FC c3 c3 A5 50:06:01:61:41:e0:3b:33 -
```

LUN グループのメンバーの表示

次のコマンドを実行すると、指定した Dynamic Multi-Pathing（DMP）ノードと同じ LUN グループに属する DMP ノードが表示されます。

```
# vxdmpadm getlungroup dmpnodename=dmpnode
```

次に例を示します。

```
# vxdmpadm getlungroup dmpnodename=sdq

NAME      STATE    ENCLR-TYPE  PATHS     ENBL    DSBL    ENCLR-NAME
=============================================================
sdo       ENABLED  ACME        2         2       0       enc1
sdp       ENABLED  ACME        2         2       0       enc1
sdq       ENABLED  ACME        2         2       0       enc1
sdr       ENABLED  ACME        2         2       0       enc1
```

DMP ノード、コントローラ、エンクロージャ、アレイポートによって制御されるパスの表示

vxdmpadm getsubpaths コマンドを実行すると、DMP（Dynamic Multi-Pathing）が認識するすべてのパスが一覧表示されます。vxdmpadm getsubpaths コマンドには、特定の DMP ノード、コントローラ、エンクロージャ、またはアレイポート上のサブパスを一覧表示

するオプションもあります。アレイポート上のパスを一覧表示するには、エンクロージャ名とアレイポート ID の組み合わせ、またはアレイポートのワールドワイドネーム(WWN)を指定します。

DMP が認識するすべてのサブパスを一覧表示するには、次のコマンドを実行します。

```
# vxdmpadm getsubpaths
```

```
NAME    STATE[A]    PATH-TYPE[M]          DMPNODENAME        ENCLR-NAME    CTLR    ATTRS
==============================================================================
sdaf    ENABLED(A)  PRIMARY               ams_wms0_130       ams_wms0      c2        -
sdc     ENABLED     SECONDARY             ams_wms0_130       ams_wms0      c3        -
sdb     ENABLED(A)    -                   vm04_disk_24       disk          c0        -
sda     ENABLED(A)    -                   vm04_disk_25       disk          c0        -
sdaa    ENABLED     Active/Non-Optimized emc_clariion0_438 emc_clariion0 c1        -
sdak    ENABLED(A)  Active/Optimized(P)   emc_clariion0_438 emc_clariion0 c6        -
```

vxdmpadm getsubpaths コマンドを dmpnodename 属性と組み合わせると、/dev/vx/dmp ディレクトリで指定した DMP ノード名によって制御される LUN すべてのパスが表示されます。

```
# vxdmpadm getsubpaths dmpnodename=sdby
```

```
NAME    STATE[A]    PATH-TYPE[M] CTLR-NAME  ENCLR-TYPE  ENCLR-NAME  ATTRS    PRIORITY
==============================================================================
sdbp    ENABLED       -          c1          EMC         emc0          -        -
sdbs    ENABLED       -          c1          EMC         emc0          -        -
sdbv    ENABLED       -          c6          EMC         emc0          -        -
sdby    ENABLED(A)    -          c6          EMC         emc0          -        -
```

A/A アレイの場合、I/O に使える有効パスはすべて ENABLED(A) として表示されます。

I/O ポリシーが singleactive に設定されている A/P アレイの場合、1 つのパスのみが ENABLED(A) として表示されます。その他のパスは ENABLED でも使えません。I/O ポリシーが singleactive に設定されていない場合、DMP は、状態が ENABLED(A) と表示されているパスのグループ(すべてプライマリまたはすべてセカンダリ)を使えます。

p.338 の「I/O ポリシーの指定」を参照してください。

DISABLED 状態のパスは I/O 操作には使えません。

システム管理者によって手動で無効にされたパスは DISABLED(M) と表示されます。障害が起きたパスは DISABLED と表示されます。

getsubpaths を使うと、特定の HBA コントローラに接続しているすべてのパスに関する情報を入手できます。

```
# vxdmpadm getsubpaths ctlr=c1

NAME STATE[A]   PATH-TYPE[M]         DMPNODENAME        ENCLR-TYPE   ENCLR-NAME
ATTRS  PRIORITY
==========================================================================================
sdh  ENABLED    Active/Non-Optimized emc_clariion0_429  EMC_CLARiiON  emc_clariion0
  -       -
sdr  ENABLED(A) Active/Optimized(P)  emc_clariion0_429  EMC_CLARiiON  emc_clariion0
  -       -
sdm  ENABLED(A) Active/Optimized(P)  emc_clariion0_430  EMC_CLARiiON  emc_clariion0
  -       -
sdw  ENABLED    Active/Non-Optimized emc_clariion0_430  EMC_CLARiiON  emc_clariion0
  -       -
```

また、getsubpaths を使うと、アレイ上のポートに接続しているすべてのパスに関する情報を入手できます。アレイポートは、エンクロージャ名とアレイポート ID で指定するか、またはアレイポートの WWN（ワールドワイド名）識別子で指定します。

```
# vxdmpadm getsubpaths enclosure=enclosure portid=portid
# vxdmpadm getsubpaths pwwn=pwwn
```

たとえば、エンクロージャとアレイポート ID を指定してアレイポート上のサブパスを一覧表示するには、次のコマンドを実行します。

```
# vxdmpadm getsubpaths enclosure=emc_clariion0 portid=A7

NAME STATE[A]   PATH-TYPE[M]         DMPNODENAME       ENCLR-NAME   CTLR ATTRS PRIORITY
==========================================================================================
sdal ENABLED(A) Active/Optimized     emc_clariion0_429 emc_clariion0 c6   -      -
sdr  ENABLED(A) Active/Optimized     emc_clariion0_429 emc_clariion0 c1   -      -
sdaq ENABLED    Active/Non-Optimized emc_clariion0_430 emc_clariion0 c6   -      -
sdw  ENABLED    Active/Non-Optimized emc_clariion0_430 emc_clariion0 c1   -      -
```

たとえば、WWN を通してアレイポート上のサブパスを一覧表示するには、次のコマンドを実行します。

```
# vxdmpadm getsubpaths pwwn=50:06:01:67:3e:a0:75:95

NAME STATE[A]   PATH-TYPE[M]        CTLR-NAME ENCLR-TYPE   ENCLR-NAME     ATTRS PRIORITY
==========================================================================================
sdal ENABLED(A) Active/Optimized     c6        EMC_CLARiiON  emc_clariion0  -       -
sdr  ENABLED(A) Active/Optimized(P)  c1        EMC_CLARiiON  emc_clariion0  -       -
sdaq ENABLED    Active/Non-Optimized c6        EMC_CLARiiON  emc_clariion0  -       -
sdw  ENABLED    Active/Non-Optimized c1        EMC_CLARiiON  emc_clariion0  -       -
```

```
# vxdmpadm getsubpaths pwwn=20:00:00:E0:8B:06:5F:19
```

getsubpaths を使うと、エンクロージャのすべてのサブパスに関する情報を入手できます。

```
# vxdmpadm getsubpaths enclosure=enclosure_name [ctlr=ctlrname]
```

エンクロージャ上のすべてのサブパスを一覧表示するには、次のコマンドを実行します。

```
# vxdmpadm getsubpaths enclosure=emc_clariion0
NAME         STATE[A]    PATH-TYPE[M] DMPNODENAME   ENCLR-NAME    CTLR    ATTRS
================================================================================
sdav         ENABLED(A)  PRIMARY      emc_clariion0_1017 emc_clariion0 c3    -
sdbf         ENABLED     SECONDARY    emc_clariion0_1017 emc_clariion0 c3    -
sdau         ENABLED(A)  PRIMARY      emc_clariion0_1018 emc_clariion0 c3    -
sdbe         ENABLED     SECONDARY    emc_clariion0_1018 emc_clariion0 c3    -
```

エンクロージャ上のコントローラのすべてのサブパスを一覧表示するには、次のコマンドを実行します。

```
# vxdmpadm getsubpaths enclosure=emc_clariion0
```

デフォルトでは、vxdmpadm getsubpaths コマンドの出力は、エンクロージャ名、DMP ノード名によってソートされ、さらにその中ではパス名でソートされます。

パス名、DMP ノード名、エンクロージャ名、またはホストコントローラ名に基づいて出力をソートするには、-s オプションを使います。

サブパスの情報をソートするには、次のコマンドを使います。

```
# vxdmpadm -s {path | dmpnode | enclosure | ctlr} getsubpaths ¥
[all | ctlr=ctlr_name | dmpnodename=dmp_device_name | ¥
enclosure=enclr_name [ctlr=ctlr_name | portid=array_port_ID] | ¥
pwwn=port_WWN | tpdnodename=tpd_node_name]
```

p.384 の「DMP ノードのカスタム名の設定」を参照してください。

コントローラに関する情報の表示

次の Dynamic Multi-Pathing（DMP）コマンドを実行すると、システム上のすべての HBA コントローラの属性が一覧表示されます。

```
# vxdmpadm listctlr all

CTLR-NAME  ENCLR-TYPE     STATE    ENCLR-NAME      PATH_COUNT
==============================================================
c1         OTHER          ENABLED  other0          3
c2         X1             ENABLED  jbod0           10
```

```
c3          ACME            ENABLED  enc0            24
c4          ACME            ENABLED  enc0            24
```

この出力では、コントローラ c1 はエンクロージャタイプが OTHER であるため、認識される DMP カテゴリに含まれないディスクに接続されていることがわかります。

その他のコントローラは、認識される DMP カテゴリに含まれるディスクに接続されています。

すべてのコントローラが、I/O 操作に利用可能であることを示す ENABLED 状態になっています。

状態が DISABLED であれば、コントローラが I/O 操作に利用不能であるという意味です。利用不能の場合は、ハードウェア障害が起きているか、vxdmpadm disable コマンドを使ってそのコントローラ上での I/O 操作が無効にされている可能性があります。

この形式のコマンドを実行すると、特定のエンクロージャまたは特定のエンクロージャタイプに属するコントローラが一覧表示されます。

```
# vxdmpadm listctlr enclosure=emc0
```

または

```
# vxdmpadm listctlr type=EMC
```

```
# vxdmpadm listctlr type=EMC
CTLR_NAME   ENCLR_TYPE    STATE      ENCLR_NAME  PATH_COUNT
============================================================
c1          EMC           ENABLED    emc0            6
c6          EMC           ENABLED    emc0            6
```

vxdmpadm getctlr コマンドを実行すると、HBA ベンダーの詳細とコントローラ ID が表示されます。iSCSI デバイスでは、コントローラ ID は IQN または IEEE 形式に基づく名前になります。FC デバイスの場合、コントローラ ID は WWN です。WWN は ESD から取得されます。したがって、ESD が動作していなければこのフィールドはブランクになります。ESD は、イベントの発生を DDL に通知するために使われるデーモンプロセスです。「コントローラ ID」として表示される WWN は、ホストコントローラに関連付けられた HBA ポートの WWN にマップされます。

```
# vxdmpadm getctlr c5
```

```
LNAME     PNAME VENDOR  CTLR-ID
=================================================
c5        c5    qlogic      20:07:00:a0:b8:17:e1:37
```

エンクロージャに関する情報の表示

DMP（Dynamic Multi-Pathing）は、可能な場合、エンクロージャタイプ、エンクロージャシリアル番号、状態、アレイタイプ、LUN 数、ファームウェアバージョンといったエンクロージャの属性を表示できます。

指定したエンクロージャの属性を表示するには、次の DMP コマンドを使います。

```
# vxdmpadm listenclosure emc0
ENCLR_NAME ENCLR_TYPE ENCLR_SNO            STATUS    ARRAY_TYPE LUN_COUNT FIRMWARE
==========================================================================================
emc0       EMC        000292601383         CONNECTED A/A        30        5875
```

システムのすべてのエンクロージャの属性を表示するには、次の DMP コマンドを使います。

```
# vxdmpadm listenclosure all
```

```
ENCLR_NAME       ENCLR_TYPE     ENCLR_SNO     STATUS    ARRAY_TYPE LUN_COUNT FIRMWARE
==========================================================================================
Disk             Disk           DISKS         CONNECTED Disk       6         -
emc0             EMC            000292601383  CONNECTED A/A        1         5875
hitachi_usp-vm0 Hitachi_USP-VM 25847         CONNECTED A/A        1         6008
emc_clariion0    EMC_CLARiiON   CK20007040035 CONNECTED CLR-A/PF   2         0324
```

アレイポートに関する情報の表示

アレイポートについての情報を表示するには、このセクションの DMP（Dynamic Multi-Pathing）コマンドを使います。アレイポートに関して表示される情報には、そのエンクロージャの名前、ID、WWN 識別子が含まれます。

パス、DMP ノード、HBA コントローラを介してアクセス可能なアレイポートの属性を表示するには、次のいずれかのコマンドを使います。

```
# vxdmpadm getportids path=path_name
# vxdmpadm getportids dmpnodename=dmpnode_name
# vxdmpadm getportids ctlr=ctlr_name
```

次のコマンド形式を実行すると、特定のエンクロージャ内のすべてのアレイポートに関する情報を表示します。

```
# vxdmpadm getportids enclosure=enclr_name
```

次の例では、DMP ノード sdg を介してアクセス可能なアレイポートに関する情報が表示されます。

```
# vxdmpadm getportids dmpnodename=sdg

NAME        ENCLR-NAME  ARRAY-PORT-ID pWWN
============================================================
sdg         HDS9500V0   1A            20:00:00:E0:8B:06:5F:19
```

サードパーティ製のドライバにより制御されるデバイスに関する情報の表示

サードパーティ製ドライバ(TPD)共存機能を使うと、DMP(Dynamic Multi-Pathing)の監視機能を残したまま、サードパーティ製マルチパス化ドライバによって制御されている I/O に DMP をバイパスさせることができます。次のコマンドは、指定した TPD デバイスに対して、DMP が検出したパスを表示するコマンドおよび指定した TPD 制御ノードに対して DMP が検出した対応する TPD デバイスを表示するコマンドです。

```
# vxdmpadm getsubpaths tpdnodename=TPD_node_name
# vxdmpadm gettpdnode nodename=TPD_path_name
```

p.386 の「サードパーティ製ドライバ制御のエンクロージャに対するデバイスの命名の変更」を参照してください。

たとえば、EMC Symmetrix アレイ内の PowerPath によって制御され、DMP にも認識される次のようなディスクを想定します。

```
# vxdisk list

DEVICE       TYPE            DISK         GROUP        STATUS
emcpowerp    auto:cdsdisk    -            -            online
emcpowerq    auto:cdsdisk    -            -            online
emcpowerr    auto:cdsdisk    -            -            online
emcpowers    auto:cdsdisk    -            -            online
emcpowert    auto:cdsdisk    -            -            online
```

次のコマンドを実行すると、DMP が PowerPath 制御のノード emcpowerp に対応するパスを検出し、表示します。

```
# vxdmpadm getsubpaths tpdnodename=emcpowerp

NAME         TPDNODENAME  PATH-TYPE[-] DMPNODENAME  ENCLR-TYPE   ENCLR-NAME
=============================================================================
sdt          emcpowerp        -        emcpowerp    PP_EMC_CLARiiON  pp_emc_clariion0
sdo          emcpowerp        -        emcpowerp    PP_EMC_CLARiiON  pp_emc_clariion0
sdj          emcpowerp        -        emcpowerp    PP_EMC_CLARiiON  pp_emc_clariion0
sde          emcpowerp        -        emcpowerp    PP_EMC_CLARiiON  pp_emc_clariion0
```

逆に次のコマンドは、DMP がパス sdt に対応する PowerPath ノードを検出し、PowerPath ノードに関する情報を表示します。

```
# vxdmpadm gettpdnode nodename=sdt
NAME                    STATE      PATHS      ENCLR-TYPE    ENCLR-NAME
============================================================================
emcpowerp               ENABLED      4          PP_EMC_CLARiiON   pp_emc_clariion0
```

拡張デバイス属性の表示

DDL（Device Discovery Layer の略でデバイス検出層の意味）拡張属性は、DDL によって検出される VxVM（Veritas Volume Manager）、DMP（Dynamic Multi-Pathing）LUN またはディスクに対応する属性またはフラグです。これらの属性によって、LUN が特定のハードウェアカテゴリに識別されます。

表 11-5 は、カテゴリの一覧です。

表 11-5　　拡張属性のカテゴリ

カテゴリ	説明
ハードウェア RAID のタイプ	LUN が属するストレージ RAID グループのタイプを表示します。
シンプロビジョニングの検出と再生	LUN のシン再生機能を表示します。
デバイスメディアのタイプ	メディアのタイプについて、ソリッドステートドライブ（SSD）かどうかを表示します。
ストレージベースのスナップショット/クローン	LUN がプライマリ LUN のスナップショットまたはクローンのどちらであるかを表示します。
ストレージベースのレプリケーション	LUN が、リモートサイト全体にわたってレプリケートされるグループの一部であるかどうかを表示します。
トランスポート	この LUN への接続に使われる HBA のタイプ（FC、SATA、iSCSI）を表示します。

各 LUN には、これらの拡張属性が 1 つ以上存在する場合があります。DDL は、ASL（Array Support Library）からのデバイス検出中に拡張属性を検出します。また、VOM（Veritas Operations Manager）がある場合、DDL は、管理対象ホストとして設定されているホストの VOM Management Server から拡張属性を取得することもできます。

vxdisk -p list コマンドを実行すると、DDL 拡張属性が表示されます。たとえば、次のコマンドを実行すると、この LUN の std、fc、RAID_5 属性が表示されます。

```
# vxdisk -p list
DISK           : tagmastore-usp0_0e18
```

```
DISKID          : 1253585985.692.rx2600h11
VID             : HITACHI
UDID            : HITACHI%5FOPEN-V%5F02742%5F0E18
REVISION        : 5001
PID             : OPEN-V
PHYS_CTLR_NAME : 0/4/1/1.0x50060e8005274246
LUN_SNO_ORDER   : 411
LUN_SERIAL_NO   : 0E18
LIBNAME         : libvxhdsusp.sl
HARDWARE_MIRROR: no
DMP_DEVICE      : tagmastore-usp0_0e18
DDL_THIN_DISK   : thick
DDL_DEVICE_ATTR: std fc RAID_5
CAB_SERIAL_NO   : 02742
ATYPE           : A/A
ARRAY_VOLUME_ID: 0E18
ARRAY_PORT_PWWN: 50:06:0e:80:05:27:42:46
ANAME           : TagmaStore-USP
TRANSPORT       : FC
```

vxdisk -x attribute -p list コマンドを実行すると、プロパティリストと属性の 1 行リストが表示されます。次の例は、hdprclm 属性を使ってシン再生をサポートする 2 つの日立製 LUN を示しています。

```
# vxdisk -x DDL_DEVICE_ATTR -p list
DEVICE                  DDL_DEVICE_ATTR
tagmastore-usp0_0a7a    std fc RAID_5
tagmastore-usp0_065a    hdprclm fc
tagmastore-usp0_065b    hdprclm fc
```

ユーザーは、同じコマンド内に複数の **-x** オプションを指定することによって複数のエントリを表示できます。次に例を示します。

```
# vxdisk -x DDL_DEVICE_ATTR -x VID -p list

DEVICE               DDL_DEVICE_ATTR  VID
tagmastore-usp0_0a7a std fc RAID_5    HITACHI
tagmastore-usp0_0a7b std fc RAID_5    HITACHI
tagmastore-usp0_0a78 std fc RAID_5    HITACHI
tagmastore-usp0_0a79 std fc RAID_5    HITACHI
tagmastore-usp0_065a hdprclm fc       HITACHI
tagmastore-usp0_065b hdprclm fc       HITACHI
tagmastore-usp0_065c hdprclm fc       HITACHI
tagmastore-usp0_065d hdprclm fc       HITACHI
```

vxdisk -e list コマンドを使うと、ATTR という名前の最後の列に DLL_DEVICE_ATTR プロパティが表示されます。

```
# vxdisk -e list
DEVICE               TYPE   DISK   GROUP   STATUS   OS_NATIVE_NAME   ATTR
tagmastore-usp0_0a7a auto   -      -       online   c10t0d2          std fc RAID_5
tagmastore-usp0_0a7b auto   -      -       online   c10t0d3          std fc RAID_5
tagmastore-usp0_0a78 auto   -      -       online   c10t0d0          std fc RAID_5
tagmastore-usp0_0655 auto   -      -       online   c13t2d7          hdprclm fc
tagmastore-usp0_0656 auto   -      -       online   c13t3d0          hdprclm fc
tagmastore-usp0_0657 auto   -      -       online   c13t3d1          hdprclm fc
```

拡張属性をサポートする ASL の一覧とこれらの属性の説明については、次の URL にあるハードウェア互換性リスト (HCL) を参照してください。

https://www.veritas.com/support/en_US/article.000126344

VxVM の制御下におけるデバイスの無効化と有効化

vxdmpadm exclude コマンドを実行すると、指定する基準に基づいてデバイスが VxVM (Veritas Volume Manager) 制御から除外されます。デバイスは、無効化されると DMP (Dynamic Multi-Pathing) によって要求されず、結果として VxVM で使えなくなります。vxdmpadm include コマンドを実行すると、VxVM の制御下にデバイスを追加し直すことができます。デバイスは、VID:PID の組み合わせ、パス、コントローラ、またはディスクに基づいて追加または除外できます。感嘆符 (!) を使えば、指定したもの以外のパスまたはコントローラを除外または追加できます。

ルートディスクは無効にできません。外部ディスクのベンダー ID とプロダクト ID が、ルートディスクと同じベンダー ID とプロダクト ID で、ルートディスクが VxVM でカプセル化されている場合、操作は失敗します。

メモ: ! 文字は一部のシェルでの特殊文字です。次の構文は、bash シェルでこの文字をエスケープ処理する方法を示しています。

```
# vxdmpadm exclude { all | product=VID:PID |
ctlr=[¥!]ctlrname | dmpnodename=diskname [ path=[¥!]pathname] }

# vxdmpadm include { all | product=VID:PID |
ctlr=[¥!]ctlrname | dmpnodename=diskname [ path=[¥!]pathname] }
```

各オプションの説明

all	すべてのデバイス
product=*VID:PID*	指定した VID:PID を持つすべてのデバイス

ctlr=*ctlrname*	指定したコントローラ上のすべてのデバイス
dmpnodename=*diskname*	DMP ノード下のすべてのデバイス
dmpnodename=*diskname* path=¥!*pathname*	指定したものを除く、DMP ノード下のすべてのデバイス

I/O 統計情報の収集と表示

vxdmpadm iostat コマンドを使って、指定した DMP ノード、エンクロージャ、パス、ポート、コントローラについての I/O 統計情報を収集し、表示できます。

表示される統計情報は、統計情報の収集に使われた CPU ごとの CPU 使用率とメモリサイズ、読み取り操作および書き込み操作の数、読み取りおよび書き込みが行われた KB 数、読み取りまたは書き込みが行われた KB あたりの平均時間(ミリ秒)です。

統計情報の収集を有効にするには、次のコマンドを入力します。

```
# vxdmpadm iostat start [memory=size]
```

memory 属性は各 CPU の I/O 統計情報の記録に使うメモリの最大サイズを制限します。デフォルトでは、各 CPU に対して 32k(32 KB)に制限されます。

I/O のカウンタを 0 にリセットするには、次のコマンドを入力します。

```
# vxdmpadm iostat reset
```

収集した統計情報を一定の間隔で表示するには、次のコマンドを入力します。

```
# vxdmpadm iostat show {filter} [interval=seconds [count=N]]
```

上記のコマンドは ***filter*** によって指定されたデバイスの I/O 統計情報を表示したものです。***filter*** は次のいずれかになります。

- all
- ctlr=*ctlr-name*
- dmpnodename=*dmp-node*
- enclosure=*enclr-name* [portid=*array-portid*] [ctlr=*ctlr-name*]
- pathname=*path-name*
- pwwn=*array-port-wwn*[ctlr=*ctlr-name*]

interval 属性および count 属性を使うと、I/O 統計情報を表示する間隔(秒)と表示する行数をそれぞれ指定できます。統計情報の記録に使えるメモリが十分でない場合は、指定した値よりも間隔が短くなることがあります。

DMP には、指定した条件で収集した I/O 累積統計情報を表示する *groupby* オプションもあります。

p.330 の「累積 I/O 統計情報の表示」を参照してください。

統計情報の収集を無効にするには、次のコマンドを入力します。

```
# vxdmpadm iostat stop
```

累積 I/O 統計情報の表示

`vxdmpadm iostat` コマンドにより、さまざまな I/O チャネルまたは I/O チャネルの一部にわたる I/O 負荷の分散を分析する機能が提供されます。適切な *filter* を選択して、DMP ノード、コントローラ、アレイエンクロージャ、パス、ポート、仮想マシンの I/O 統計情報を表示します。次に、*groupby* 節を使って、分析する基準に従って累積統計を表示します。`groupby` 節を指定しない場合、統計はパスごとに表示されます。

filter と *groupby* 節を組み合わせるときに、必須の使用例のシナリオの I/O 負荷を分析できます。次に例を示します。

- HBA、エンクロージャ、アレイポートにわたる I/O 負荷を比較するには、指定した属性を持つ *groupby* 節を使います。
- 指定の I/O チャネル（ポートリンクを配列する HBA）をわたる I/O 負荷を分析するには、HBA と PWWN またはエンクロージャとポートで *filter* を使います。
- HBA へのリンクにわたる I/O 負荷の分散を分析するには、HBA による *filter* と *groupby* アレイポートを使います。

`iostat` コマンドの次の形式を使って I/O 負荷を分析します。

```
# vxdmpadm [-u unit] iostat show [groupby=criteria] {filter}  ¥
  [interval=seconds [count=N]]
```

上記のコマンドは *filter* によって指定されたデバイスの I/O 統計情報を表示したものです。*filter* は次のいずれかになります。

- `all`
- `ctlr=ctlr-name`
- `dmpnodename=dmp-node`
- `enclosure=enclr-name [portid=array-portid ] [ctlr=ctlr-name]`
- `pathname=path-name`
- `pwwn=array-port-wwn[ctlr=ctlr-name]`

次の `groupby` 基準に従って統計情報を集計できます。

- `arrayport`

- ctlr
- dmpnode
- enclosure

デフォルトでは、読み書き時間はミリ秒単位で小数点以下 2 桁まで表示されます。スループットデータは「ブロック」単位で表示され、その出力は拡大縮小されます。つまり、有効数字を一定に保ったまま、小さい値は小さい単位で表示され、大きい値は大きい単位で表示されます。統計データの表示単位を指定できます。-u オプションには次のオプションを指定できます。

h または H	スループットを最も大きい単位で表示します。
k	スループットをキロバイト単位で表示します。
m	スループットをメガバイト単位で表示します。
g	スループットをギガバイト単位で表示します。
bytes| b	スループットを正確なバイト数で表示します。
us	読み取り/書き込みの平均時間をマイクロ秒単位で表示します。

DMP ノードでグループ化するには、次のコマンドを使います。

```
# vxdmpadm [-u unit] iostat show groupby=dmpnode ¥
[all | dmpnodename=dmpnodename | enclosure=enclr-name]
```

コントローラでグループ化するには、次のコマンドを使います。

```
# vxdmpadm [-u unit] iostat show groupby=ctlr [ all | ctlr=ctlr ]
```

次に例を示します。

```
# vxdmpadm iostat show groupby=ctlr ctlr=c5

            OPERATIONS        BLOCKS       AVG TIME(ms)
CTLRNAME  READS   WRITES  READS  WRITES  READS   WRITES
c5        224      14     54      7       4.20    11.10
```

アレイポートでグループ化するには、次のコマンドを使います。

```
# vxdmpadm [-u unit] iostat show groupby=arrayport [ all ¥
| pwwn=array_pwwn | enclosure=enclr portid=array-port-id ]
```

次に例を示します。

```
# vxdmpadm -u m iostat show groupby=arrayport ¥
enclosure=HDS9500-ALUA0 portid=1A
```

```
             OPERATIONS         BYTES        AVG TIME(ms)
PORTNAME   READS    WRITES   READS   WRITES   READS    WRITES
1A         743      1538     11m     24m      17.13    8.61
```

エンクロージャでグループ化するには、次のコマンドを使います。

```
# vxdmpadm [-u unit] iostat show groupby=enclosure [ all ¥
| enclosure=enclr ]
```

次に例を示します。

```
# vxdmpadm -u h iostat show groupby=enclosure enclosure=EMC_CLARiiON0

OPERATIONS        BLOCKS        AVG TIME(ms)
ENCLOSURENAME    READS   WRITES   READS   WRITES   READS    WRITES
EMC_CLARiiON0    743     1538     11392k  24176k 17.13    8.61
```

すべてのデータエントリがゼロであるエンティティはフィルタで除外することもできます。このオプションは多くのフェールオーバーデバイスを含んでいるクラスタ環境に特に有用です。アクティブパスの統計のみを表示できます。

iostat show コマンドの出力からすべてがゼロのエントリをフィルタ処理するには、次のコマンドを使います。

```
# vxdmpadm [-u unit] -z iostat show [all|ctlr=ctlr_name |
dmpnodename=dmp_device_name | enclosure=enclr_name [portid=portid]
|
pathname=path_name|pwwn=port_WWN][interval=seconds [count=N]]
```

次に例を示します。

```
# vxdmpadm -z iostat show dmpnodename=emc_clariion0_893

cpu usage = 9852us    per cpu memory = 266240b
              OPERATIONS       BLOCKS           AVG TIME(ms)
PATHNAME   READS   WRITES    READS    WRITES    READS     WRITES
sdbc       32      0         258      0          0.04       0.00
sdbw       27      0         216      0          0.03       0.00
sdck       8       0         57       0          0.04       0.00
sdde       11      0         81       0          0.15       0.00
```

読み取り/書き込みの平均時間をマイクロ秒単位で表示するには

```
# vxdmpadm -u us iostat show pathname=sdck

cpu usage = 9865us    per cpu memory = 266240b
               OPERATIONS       BLOCKS           AVG TIME(us)
```

```
PATHNAME    READS     WRITES   READS     WRITES    READS     WRITES
sdck        8         0        57        0         43.04       0.00
```

キューに入れられた I/O または無効な I/O の統計の表示

指定した DMP ノード、または指定したパスまたはコントローラについて、DMP（Dynamic Multi-Pathing）内のキューに入れられた I/O を表示するには、-q オプションを付けた vxdmpadm iostat show コマンドを使います。DMP ノードの場合、-q オプションを付けると、基盤となるレイヤーに送信された、指定した DMP ノード上の I/O が表示されます。パスまたはコントローラを指定した場合、-q オプションを付けると、指定したパスまたはコントローラに送信され、DMP にまだ返されていない I/O が表示されます。

vxdmpadm iostat コマンドの詳細については、vxdmpadm（1m）マニュアルページを参照してください。

DMP ノード上のキューに入れられた I/O 件数を表示するには、次のコマンドを使います。

```
# vxdmpadm -q iostat show [filter] [interval=n [count=m]]
```

次に例を示します。

```
# vxdmpadm -q iostat show dmpnodename=emc_clariion0_352

cpu usage = 338us    per cpu memory = 102400b
                        QUEUED I/Os     PENDING I/Os
DMPNODENAME          READS    WRITES
emc_clariion0_352    0        0          0
```

DMP ノード、パス、またはコントローラ上でエラーで返された I/O の件数を表示するには

```
# vxdmpadm -e iostat show [filter] [interval=n [count=m]]
```

たとえば、パス上でエラーを返された I/O の件数を表示するには、次を実行します。

```
# vxdmpadm -e iostat show pathname=sdo

cpu usage = 637us    per cpu memory = 102400b
                  ERROR I/Os
PATHNAME      READS    WRITES
sdo           0          0
```

vxdmpadm iostat コマンドの使用例

DMP（Dynamic Multi-Pathing）では、vxdmpadm iostat コマンドを使って I/O 統計情報を収集し、表示できます。ここでは、vxdmpadm iostat コマンドを使ったセッションの例を示します。

最初に次のコマンドを実行すると I/O 統計情報の収集が有効になります。

```
# vxdmpadm iostat start
```

続いて次のコマンドを実行すると、読み取りおよび書き込み操作の総数、読み取りおよび書き込みが行われたキロバイト数など、すべてのパスに関する現在の統計情報が表示されます。

```
# vxdmpadm -u k iostat show all
                    cpu usage = 7952us    per cpu memory = 8192b
             OPERATIONS            BYTES          AVG TIME(ms)
PATHNAME READS     WRITES    READS     WRITES   READS    WRITES
sdf          87          0    44544k          0    0.00      0.00
sdk           0          0         0          0    0.00      0.00
sdg          87          0    44544k          0    0.00      0.00
sdl           0          0         0          0    0.00      0.00
sdh          87          0    44544k          0    0.00      0.00
sdm           0          0         0          0    0.00      0.00
sdi          87          0    44544k          0    0.00      0.00
sdn           0          0         0          0    0.00      0.00
sdj          87          0    44544k          0    0.00      0.00
sdo           0          0         0          0    0.00      0.00
sdj          87          0    44544k          0    0.00      0.00
sdp           0          0         0          0    0.00      0.00
```

次のコマンドを実行すると、vxdmpadm で統計情報の収集に使われるメモリのサイズが変更されます。

```
# vxdmpadm iostat start memory=4096
```

統計情報は、次のようにパス名、DMP ノード名およびエンクロージャ名ごとにフィルタして表示することができます(各 CPU のメモリサイズは前のコマンドに従って変更されています)。

```
# vxdmpadm -u k iostat show pathname=sdk
                    cpu usage = 8132us    per cpu memory = 4096b
               OPERATIONS            BYTES             AVG TIME(ms)
PATHNAME    READS     WRITES      READS     WRITES     READS    WRITES
sdk             0          0          0          0      0.00      0.00

# vxdmpadm -u k iostat show dmpnodename=sdf
                    cpu usage = 8501us    per cpu memory = 4096b
               OPERATIONS            BYTES             AVG TIME(ms)
PATHNAME    READS     WRITES      READS     WRITES     READS    WRITES
sdf          1088          0    557056k          0      0.00      0.00

# vxdmpadm -u k iostat show enclosure=Disk
```

```
                    cpu usage = 8626us    per cpu memory = 4096b
                OPERATIONS              BYTES              AVG TIME(ms)
PATHNAME    READS     WRITES      READS     WRITES     READS    WRITES
sdf          1088          0    557056k          0      0.00      0.00
```

統計情報を表示する回数および間隔を指定することもできます。次のコマンドを実行すると、特定のパスについての統計情報の増分が表示されます。この例では、回数は2回、間隔は 2 秒と指定しています。

```
# vxdmpadm iostat show pathname=sdk interval=2 count=2
                    cpu usage = 9621us    per cpu memory = 266240b
                OPERATIONS              BLOCKS           AVG TIME(ms)
PATHNAME    READS     WRITES     READS     WRITES     READS     WRITES
sdk         0          0          0          0          0.00       0.00

sdk         0          0          0          0          0.00       0.00
```

エンクロージャへのパスに関する属性の設定

vxdmpadm setattr コマンドを使って、エンクロージャまたはディスクアレイへのパスに関する属性を設定できます。

パスに設定された属性は再起動後も、製品アップグレード後も保持されます。

設定できる属性は次のとおりです。

active　スタンバイ(フェールオーバー)パスをアクティブパスに変更します。アレイへのアクティブパスを指定するには、次のコマンドを実行します。

```
# vxdmpadm setattr path sde pathtype=active
```

nomanual　パスの元のプライマリ属性またはセカンダリ属性を復元します。JBOD ディスクへのパスを復元するには、次のコマンドを実行します。

```
# vxdmpadm setattr path sdm pathtype=nomanual
```

nopreferred　パスの通常の優先順位を復元します。パスに対するデフォルトの優先順位を復元するには、次のコマンドを実行します。

```
# vxdmpadm setattr path sdk ¥
  pathtype=nopreferred
```

`preferred [priority=N]`

特定のパスを優先パスとして指定し、オプションで優先順位番号を割り当てます。優先順位番号を指定する場合は、1 以上の整数にする必要があります。パスの優先順位の高さは、I/O 負荷の伝送能力の高さを表します。

p.338 の「I/O ポリシーの指定」を参照してください。

アクティブ/アクティブディスクアレイの I/O ポリシーに `priority` 属性を設定し、パスの優先順位を 2 に指定するには、次のコマンドを実行します。

```
# vxdmpadm setattr enclosure enc0 ¥
  iopolicy=priority
# vxdmpadm setattr path sdk pathtype=preferred ¥
  priority=2
```

`primary`

特定のパスを JBOD ディスクアレイのプライマリパスとして定義します。JBOD ディスクアレイのプライマリパスを指定するには、次のコマンドを実行します。

```
# vxdmpadm setattr path sdm pathtype=primary
```

`secondary`

特定のパスを JBOD ディスクアレイのセカンダリパスとして定義します。JBOD ディスクアレイのセカンダリパスを指定するには、次のコマンドを実行します。

```
# vxdmpadm setattr path sdn pathtype=secondary
```

`standby`

通常の I/O スケジュールには使わないスタンバイパスを設定します。このパスは、I/O に使えるアクティブパスが存在しない場合に使われます。A/P-C アレイのスタンバイパスを指定するには、次のコマンドを実行します。

```
# vxdmpadm setattr path sde pathtype=standby
```

デバイスまたはエンクロージャの冗長レベルの表示

必要な冗長レベルを下回っているデバイスを一覧表示するには、`vxdmpadm getdmpnode` コマンドを使います。

指定するエンクロージャで、有効パス数が特定の値よりも少ないデバイスを一覧表示するには、次のコマンドを使います。

```
# vxdmpadm getdmpnode enclosure=encl_name redundancy=value
```

たとえば、有効パス数が 3 つよりも少ないデバイスを一覧表示するには、次のコマンドを使います。

```
# vxdmpadm getdmpnode enclosure=EMC_CLARiiON0 redundancy=3

NAME              STATE    ENCLR-TYPE   PATHS ENBL DSBL ENCLR-NAME
==========================================================================
```

```
emc_clariion0_162 ENABLED  EMC_CLARiiON 3     2    1    emc_clariion0
emc_clariion0_182 ENABLED  EMC_CLARiiON 2     2    0    emc_clariion0
emc_clariion0_184 ENABLED  EMC_CLARiiON 3     2    1    emc_clariion0
emc_clariion0_186 ENABLED  EMC_CLARiiON 2     2    0    emc_clariion0
```

特定のデバイスの最小冗長レベルを表示するには、次のように vxdmpadm getattr コマンドを使います。

```
# vxdmpadm getattr enclosure|arrayname|arraytype ¥
 component-name redundancy
```

たとえば、エンクロージャ HDS9500-ALUA0 の最小冗長レベルを表示するには、次のコマンドを実行します。

```
# vxdmpadm getattr enclosure HDS9500-ALUA0 redundancy

ENCLR_NAME  DEFAULT  CURRENT
============================================
 HDS9500-ALUA0    0      4
```

アクティブパスの最小数の指定

デバイスまたはエンクロージャの最小冗長レベルを設定できます。最小冗長レベルは、デバイスまたはエンクロージャに対してアクティブにする必要のあるパスの最小数です。パス数がエンクロージャの最小冗長レベルを下回ると、メッセージがシステムコンソールに送信され、DMP（Dynamic Multi-Pathing）ログファイルにも記録されます。また、通知が vxnotify クライアントに送信されます。

最小冗長レベルに設定された値は再起動や製品アップグレード後も保持されます。最小冗長レベルを設定しない場合、デフォルト値は 0 です。

最小冗長レベルを設定するには、vxdmpadm setattr コマンドを使うことができます。

アクティブパスの最小数を指定するには

- 次のように、redundancy 属性を使って vxdmpadm setattr コマンドを実行します。

  ```
  # vxdmpadm setattr enclosure|arrayname|arraytype component-name
   redundancy=value
  ```

  value はアクティブパス数です。

  たとえば、エンクロージャ HDS9500-ALUA0 の最小冗長レベルを設定するには、次のコマンドを実行します。

  ```
  # vxdmpadm setattr enclosure HDS9500-ALUA0 redundancy=2
  ```

I/O ポリシーの表示

エンクロージャ、アレイまたはアレイタイプに現在設定されている I/O ポリシーおよびデフォルトで設定される I/O ポリシーを表示するには、vxdmpadm getattr コマンドを使います。

たとえば、JBOD ディスクに設定されているデフォルトおよび現在の I/O ポリシー(iopolicy)を表示するには、次のコマンドを実行します。

```
# vxdmpadm getattr enclosure Disk iopolicy

ENCLR_NAME        DEFAULT         CURRENT
---------------------------------------
Disk              MinimumQ        Balanced
```

分散 I/O ポリシーでパーティションサイズ 2 MB に設定されているエンクロージャ enc0 のパーティションサイズ(partitionsize)を表示するには、次のコマンドを実行します。

```
# vxdmpadm getattr enclosure enc0 partitionsize

ENCLR_NAME      DEFAULT         CURRENT
---------------------------------------
enc0            512             4096
```

I/O ポリシーの指定

vxdmpadm setattr コマンドを使って、ディスクアレイまたはエンクロージャへの複数のパスに I/O 負荷を分散する DMP(Dynamic Multi-Pathing) I/O ポリシーを変更することができます。ポリシーは、特定のエンクロージャ(HDS01 など)、特定のタイプのすべてのエンクロージャ(HDS など)、または特定のアレイタイプのすべてのエンクロージャ(アクティブ/アクティブの場合は A/A、アクティブ/パッシブの場合は A/P など)に対して設定できます。

メモ: I/O ポリシーはシステムの再起動後にも保持されます。

表 11-6 は設定される可能性がある I/O ポリシーを記述します。

表 11-6 DMP I/O ポリシー

ポリシー	説明

adaptive	I/O をパスに動的にスケジュールしてディスクに対する全体的な I/O スループットを最大化するためのポリシーです。このポリシーは、I/O 負荷が場合によって異なるような環境で使うと便利です。たとえば、I/O 転送が長い場合（テーブルのスキャン）や短い場合（無作為検索）があるデータベースなどで使います。また、このポリシーはパスによってホップ数が異なる SAN 環境でも有効です。このポリシーは DMP で自動的に管理されるため、設定を変更することはできません。
	たとえば、エンクロージャ enc1 に対して I/O ポリシーを adaptive に設定するには、次のコマンドを実行します。
	`# vxdmpadm setattr enclosure enc1 ¥`
	`  iopolicy=adaptive`

adaptiveminq	I/O が各パスの I/O キューの長さに従ってスケジュールされること以外は adaptive ポリシーと同様です。最も短いキューが付いているパスは最高優先度に割り当てられます。

balanced [partitionsize=*size*]	ディスクドライブおよび RAID コントローラでのキャッシュ処理を最適化するためのポリシーです。キャッシュのサイズは通常 120 から 500 KB 以上で、各ハードウェアの特性によって異なります。通常の処理時は、ディスク（LUN）は複数の領域（パーティション）に論理的に分割され、指定した領域に対する I/O が 1 つのアクティブパスにのみ送出されます。そのパスに障害が発生した場合は、作業負荷は自動的に別のアクティブパスに分散されます。
	パーティションのためにサイズを指定するのに partitionsize 属性を使うことができます。パーティションサイズのブロック数は、2 の累乗の値（2 から 231）に調整できます。2 の累乗以外の値は、自動的に適切な値に切り下げられます。
	0 のパーティションサイズを指定することは、デフォルトのパーティションサイズを指定することと同じです。
	パーティションサイズのデフォルト値は 512 ブロック（256 KB）です。パーティションサイズに 0 を指定すると、デフォルトのパーティションサイズである 512 ブロック（256 KB）が使われます。
	デフォルト値を変更するには、dmp_pathswitch_blks_shift チューニングパラメータの値を調整します。
	p.1115 の「DMP チューニングパラメータ」を参照してください。
	メモ: キャッシュサイズより大きい値を設定する場合には、このポリシーを使う利点はありません。
	たとえば、日立製 HDS 9960 A/A アレイの推奨パーティションサイズは、I/O 処理パターンが主に順次読み取りまたは書き込みである場合、32,768 から 131,072 ブロック（16 MB から 64 MB）です。
	たとえば、パーティションサイズが 4096 ブロック（2 MB）のエンクロージャ enc0 の balanced I/O ポリシーを設定するには、次のコマンドを実行します。
	`# vxdmpadm setattr enclosure enc0 ¥`
	`  iopolicy=balanced partitionsize=4096`

ポリシー	説明

minimumq	LUN のキューに残っている未処理の I/O 要求の数が最も少ないパスに I/O を送出するポリシーです。キューが最も短いパスは DMP で自動的に判別されるため、設定を変更することはできません。
	
	たとえば、JBOD に対して I/O ポリシーを minimumq に設定するには、次のコマンドを実行します。
	
	`# vxdmpadm setattr enclosure Disk ¥`
	`  iopolicy=minimumq`
	
	これはすべてのアレイでデフォルトの I/O ポリシーです。

priority	SAN 内のパスによって処理効率が異なるため負荷分散を手動で強制的に行う場合に便利なポリシーです。使用可能なパスの設定や処理効率特性、およびシステムのその他の要素にも考慮して、各パスに優先順位を割り当てることができます。
	
	p.335 の「エンクロージャへのパスに関する属性の設定」を参照してください。
	
	たとえば、すべての SENA アレイに対して I/O ポリシーを priority に設定するには、次のコマンドを実行します。
	
	`# vxdmpadm setattr arrayname SENA ¥`
	`  iopolicy=priority`

round-robin	I/O をラウンドロビンシーケンスのパス間で同等に共有するポリシーです。たとえば、3 つのパスが存在する場合、最初の I/O 要求で 1 つのパスが使われると、2 番目では別のパス、3 番目では残っている 3 つ目のパスが使われ、4 番目の I/O 要求では再度最初のパスというように割り当てられていきます。このポリシーは DMP で自動的に管理されるため、設定を変更することはできません。
	
	たとえば、すべてのアクティブ/アクティブアレイに対して I/O ポリシーを round-robin に設定するには、次のコマンドを実行します。
	
	`# vxdmpadm setattr arraytype A/A ¥`
	`  iopolicy=round-robin`

singleactive	I/O を単一のアクティブパスに送信するポリシーです。このポリシーは、1 つのコントローラに 1 つのアクティブパスが存在し、他のパスはフェールオーバーを実行する場合に使われる、A/P アレイ用に設定できます。A/A アレイ用に設定した場合、パス間での負荷分散は実行されず、代替パスは高可用性(HA)を得る場合にのみ使われます。現在のアクティブパスに障害が発生した場合、I/O は代替アクティブパスに切り替えられます。単一のアクティブパスは DMP で選択されるため、設定を変更することはできません。
	
	たとえば、JBOD ディスクに I/O ポリシー singleactive を設定するには、次のコマンドを実行します。
	
	`# vxdmpadm setattr arrayname Disk ¥`
	`  iopolicy=singleactive`

非対称アクティブ/アクティブまたは ALUA アレイのパスでの I/O のスケジュール設定

adaptive、balanced、minimumq、priority、round-robin I/O ポリシーとともに use_all_paths 属性を指定して、I/O 要求が、非対称アクティブ/アクティブ（A/A-A）アレイまたは ALUA アレイのプライマリパスに加え、セカンダリパスでもスケジュール設定するかどうかを指定できます。アレイの特性によっては、負荷分散の向上の結果として、総 I/O スループットが増加することがあります。ただし、この機能は、アレイベンダーが推奨している場合にかぎり有効にしてください。A/A-A または ALUA 以外のアレイタイプには効果がありません。

たとえば、次のコマンドでは、パーティションサイズが 4096 ブロック（2 MB）のエンクロージャ enc0 の balanced I/O ポリシーを設定し、セカンダリパスで I/O 要求をスケジュール設定できるようにしています。

```
# vxdmpadm setattr enclosure enc0 iopolicy=balanced ¥
      partitionsize=4096 use_all_paths=yes
```

この属性のデフォルト設定は use_all_paths=no です。

エンクロージャ、アレイ名、アレイタイプに対する use_all_paths の現在の設定を表示できます。これを行うには、use_all_paths オプションを vxdmpadm gettattr コマンドに指定します。

```
# vxdmpadm getattr enclosure HDS9500-ALUA0 use_all_paths

ENCLR_NAME    ATTR_NAME     DEFAULT CURRENT
============================================
HDS9500-ALUA0 use_all_paths no      yes
```

use_all_paths 属性が適用されるのは A/A-A アレイと ALUA アレイのみです。他のアレイの場合、上記コマンドを実行すると次のメッセージが表示されます。

```
Attribute is not applicable for this array.
```

SAN 環境における負荷分散の適用例

この例では、複数の SAN スイッチを経由した、アクティブ/パッシブデバイスへのプライマリパスが複数ある SAN 環境において、DMP（Dynamic Multi-Pathing）を使って負荷分散を設定する方法について説明します。

vxdisk listコマンドからのこのサンプル出力に示すように、sdmデバイスは 8 つの一次パスを備えています:

```
# vxdisk list sdq

Device: sdq
```

```
    .
    .
    .
numpaths: 8
sdj state=enabled type=primary
sdk state=enabled type=primary
sdl state=enabled type=primary
sdm state=enabled type=primary
sdn state=enabled type=primary
sdo state=enabled type=primary
sdp state=enabled type=primary
sdq state=enabled type=primary
```

さらに、このデバイスはエンクロージャ ENC0 内にあり、ディスクグループ mydg に属し、単純な連結ボリューム myvol1 を含んでいます。

まず、次のコマンドを入力して、DMP 統計情報の収集を有効にします。

```
# vxdmpadm iostat start
```

次に dd コマンドを使って、ボリュームからの入力作業負荷を適用します。

```
# dd if=/dev/vx/rdsk/mydg/myvol1 of=/dev/null &
```

デバイスの DMP 統計情報を表示する vxdmpadm iostat コマンドを実行すると、すべての I/O が 1 つのパス sdq に対して行われていることがわかります。

```
# vxdmpadm iostat show dmpnodename=sdq interval=5 count=2
    .
    .
    .
cpu usage = 11294us per cpu memory = 32768b
              OPERATIONS           KBYTES          AVG TIME(ms)
PATHNAME   READS    WRITES    READS     WRITES    READS       WRITES
sdj         0        0         0         0          0.00         0.00
sdk         0        0         0         0          0.00         0.00
sdl         0        0         0         0          0.00         0.00
sdm         0        0         0         0          0.00         0.00
sdn         0        0         0         0          0.00         0.00
sdo         0        0         0         0          0.00         0.00
sdp         0        0         0         0          0.00         0.00
sdq         10986    0         5493      0          0.41         0.00
```

次の vxdmpadm コマンドを使って、このデバイスを含むエンクロージャの I/O ポリシーを表示します。

```
# vxdmpadm getattr enclosure ENC0 iopolicy

ENCLR_NAME      DEFAULT          CURRENT
============================================
ENC0            MinimumQ      Single-Active
```

この出力から、このエンクロージャのポリシーが singleactive に設定されており、その結果、すべての I/O が 1 つのパスで行われていることがわかります。

I/O 負荷を複数のプライマリパスに分散するために、次のようにポリシーを round-robin に設定します。

```
# vxdmpadm setattr enclosure ENC0 iopolicy=round-robin
# vxdmpadm getattr enclosure ENC0 iopolicy

ENCLR_NAME    DEFAULT            CURRENT
============================================
ENC0          MinimumQ           Round-Robin
```

次に DMP 統計情報をリセットします。

```
# vxdmpadm iostat reset
```

作業負荷をかけたままの状態にして、I/O ポリシーを複数のプライマリパスへの負荷分散に変更した効果を確認することができます。

```
# vxdmpadm iostat show dmpnodename=sdq interval=5 count=2
    .
    .
    .
cpu usage = 14403us per cpu memory = 32768b
             OPERATIONS             KBYTES          AVG TIME(ms)
PATHNAME   READS    WRITES     READS      WRITES    READS       WRITES
sdj        2041     0          1021       0          0.39         0.00
sdk        1894     0          947        0          0.39         0.00
sdl        2008     0          1004       0          0.39         0.00
sdm        2054     0          1027       0          0.40         0.00
sdn        2171     0          1086       0          0.39         0.00
sdo        2095     0          1048       0          0.39         0.00
sdp        2073     0          1036       0          0.39         0.00
sdq        2042     0          1021       0          0.39         0.00
```

次のコマンドを入力すると、エンクロージャを single active I/O ポリシーに戻すことができます。

```
# vxdmpadm setattr enclosure ENC0 iopolicy=singleactive
```

パス、コントローラ、アレイポート、DMP ノードに対する I/O の無効化

パス、HBA コントローラ、アレイポート、DMP（Dynamic Multi-Pathing）ノードを介した I/O を無効にすると、DMP は、指定したパスを介して、または指定のコントローラ、アレイポートまたは DMP ノードに接続したパスを介して I/O 要求を発行できなくなります。指定したパスに保留中の I/O がある場合、vxdmpadm disable コマンドは、そのパスを無効にする前に、I/O が完了するまで待機します。

メモ: VxVM（Veritas Volume Manager）のリリース 5.0 からは、この操作は、クラスタ共有ディスクグループが設定されているディスクアレイへのアクセスに使われているコントローラでサポートされます。

DMP は、マルチパスのためにサードパーティ製ドライバ (TPD) を使うコントローラの I/O を無効にする操作をサポートしません。

1 つ以上のパスに対する I/O を無効にするには、次のコマンドを使います。

```
# vxdmpadm [-c|-f] disable path=path_name1[,path_name2,path_nameN]
```

1 つ以上の HBA コントローラに接続されているパスに対する I/O を無効にするには、次のコマンドを使います。

```
# vxdmpadm [-c|-f] disable ctlr=ctlr_name1[,ctlr_name2,ctlr_nameN]
```

アレイポートに接続したパスに対する I/O を無効にするには、次のいずれかのコマンドを使います。

```
# vxdmpadm [-c|-f] disable enclosure=enclr_name portid=array_port_ID
# vxdmpadm [-c|-f] disable pwwn=array_port_WWN
```

ここで、アレイポートは、エンクロージャ名とアレイポート ID で指定することも、アレイポートの WWN（World Wide Name）識別子で指定することもできます。

次に、アレイポートで I/O を無効にする方法の例を示します。

```
# vxdmpadm disable enclosure=HDS9500V0 portid=1A
# vxdmpadm disable pwwn=20:00:00:E0:8B:06:5F:19
```

特定のパスに対する I/O を無効にするには、ファブリックの 2 つの末端を表すコントローラとポート ID を両方とも指定します。

```
# vxdmpadm [-c|-f] disable ctlr=ctlr_name enclosure=enclr_name ¥
portid=array_port_ID
```

特定の DMP ノードの I/O を無効にするには、DMP のノード名を指定します。

```
# vxdmpadm [-c|-f] disable dmpnodename=dmpnode
```

-c オプションを使うと、ディスクへの有効なパスが 1 つだけであるかどうかを確認できます。

デバイスが使用中かどうかにかかわらず、最後のパスを無効化するには、-f オプションを使用します。

disable 操作が単一のパスを介してルートディスクに接続されたコントローラに対して発行され、代替パス上に設定されたルートディスクミラーがない場合、その操作は失敗します。そのようなミラーが存在する場合は、コマンドは成功します。disable 操作は、1 本のパスのみでスワップ デバイスに接続されているコントローラーに対して発行されると失敗します。

パス、コントローラ、アレイポート、DMP ノードに対する I/O の有効化

コントローラを有効にすると、以前に無効にされたパス、HBA コントローラ、アレイポート、DMP（Dynamic Multi-Pathing）ノードで I/O を再び受け入れられるようになります。この操作は、パス、コントローラ、アレイポート、DMP ノードがホストにアクセス可能であり、このホスト上で I/O を実行できる場合にのみ成功します。アクティブ/パッシブディスクアレイを接続している場合に、enable 操作を行うと、プライマリパスに対する I/O のフェールバックが実行されます。また、enable 操作により、以前に切断されたシステムボード上のコントローラに対する I/O を行うこともできるようになります。

メモ: この操作は、クラスタ共有ディスクグループが設定されているディスクアレイへのアクセスに使われるコントローラでサポートされます。

DMP は、マルチパスのためにサードパーティ製ドライバ（TPD）を使うコントローラの I/O を有効にする操作をサポートしません。

1 つ以上のパスに対する I/O を有効にするには、次のコマンドを使います。

```
# vxdmpadm enable path=path_name1[,path_name2,path_nameN]
```

1 つ以上の HBA コントローラに接続されているパスに対する I/O を有効にするには、次のコマンドを使います。

```
# vxdmpadm enable ctlr=ctlr_name1[,ctlr_name2,ctlr_nameN]
```

アレイポートに接続したパスに対する I/O を有効にするには、次のいずれかのコマンドを使います。

```
# vxdmpadm enable enclosure=enclr_name portid=array_port_ID
# vxdmpadm enable pwwn=array_port_WWN
```

ここで、アレイポートは、エンクロージャ名とアレイポート ID で指定することも、アレイポートの WWN（World Wide Name）識別子で指定することもできます。

次に、アレイポートで I/O を有効にするコマンドの使用例を示します。

```
# vxdmpadm enable enclosure=HDS9500V0 portid=1A
# vxdmpadm enable pwwn=20:00:00:E0:8B:06:5F:19
```

特定のパスに対する I/O を有効にするには、ファブリックの 2 つの末端を表すコントローラとポート ID を両方とも指定します。

```
# vxdmpadm enable ctlr=ctlr_name enclosure=enclr_name ¥
  portid=array_port_ID
```

特定の DMP ノードの I/O を有効にするには、DMP のノード名を指定します。

```
# vxdmpadm enable dmpnodename=dmpnode
```

エンクロージャ名の変更

vxdmpadm setattr コマンドを使って、既存のエンクロージャに意味のある名前を設定することができます。例を次に示します。

```
# vxdmpadm setattr enclosure emc0 name=GRP1
```

この例では、エンクロージャ名を emc0 から GRP1 に変更しています。

メモ: エンクロージャ名の接頭辞の長さは、最大 23 文字です。

次のコマンドは、変更された名前を表示します。

```
# vxdmpadm listenclosure all

ENCLR_NAME      ENCLR_TYPE     ENCLR_SNO     STATUS    ARRAY_TYPE
LUN_COUNT FIRMWARE
=============================================================================
Disk            Disk           DISKS         CONNECTED Disk       6
         -
GRP1            EMC            000292601383  CONNECTED A/A        1
         5875
hitachi_usp-vm0 Hitachi_USP-VM 25847         CONNECTED A/A        1
         6008
emc_clariion0   EMC_CLARiiON   CK20007040035 CONNECTED CLR-A/PF   2
         0324
```

I/O エラーに対する応答の設定

指定したエンクロージャ、ディスクアレイ名、またはアレイタイプへのパスでエラーの発生した I/O 要求に対して DMP（Dynamic Multi-Pathing）がどのように応答するかを設定できます。デフォルトでは、DMP は、エラーになった I/O 要求を各種アクティブパス上で最大 5 回再試行する設定になっています。

エンクロージャ、アレイ名、アレイタイプへのパスに適用されている I/O 要求エラー時の処理に対する現在の設定を表示するには、`vxdmpadm getattr` コマンドを使います。

p.350 の「リカバリオプション値の表示」を参照してください。

DMP がパス上で I/O 要求の送信を再試行する回数について制限を設定するには、次のコマンドを使います。

```
# vxdmpadm setattr ¥
  {enclosure enc-name|arrayname name|arraytype type} ¥
  recoveryoption=fixedretry retrycount=n
```

`retrycount` に対する引数の値には、DMP が別の利用可能なパスで I/O 要求を再度スケジュール設定するまで、またはすべての要求を失敗するまでに再試行する回数を指定します。

固定した試行回数を指定する代わりに、DMP が I/O 要求を再試行できる回数を指定できます。I/O 要求がその回数内に成功しない場合、DMP は I/O 要求を失敗します。`iotimeout` 値を指定するには、次のコマンドを使用します。

```
# vxdmpadm setattr ¥
  {enclosure enc-name|arrayname name|arraytype type} ¥
  recoveryoption=timebound iotimeout=seconds
```

`iotimeout` のデフォルト値は、300 秒です。Oracle などの一部のアプリケーションでは、`iotimeout` をもっと大きい値に設定する方が望ましい場合があります。DMP の `iotimeout` 値は下位オペレーティングシステム層の I/O 処理時間より大きい値にします。

メモ: `fixedretry` 設定と `timebound` 設定は相互に排他的です。

次の例では、エンクロージャ `enc0` に対するリカバリを期限付きで設定し、`iotimeout` の値を 360 秒に設定します。

```
# vxdmpadm setattr enclosure enc0 recoveryoption=timebound ¥
  iotimeout=360
```

次の例では、すべてのアクティブ/アクティブアレイへのパスに対して固定再試行限度を 10 回に設定します。

```
# vxdmpadm setattr arraytype A/A recoveryoption=fixedretry ¥
  retrycount=10
```

recoveryoption=default を指定すると DMP をリカバリのデフォルト設定にリセットします。

たとえば、次のコマンドはデフォルト設定を設定します:

```
# vxdmpadm setattr arraytype A/A recoveryoption=default
```

PCI デバイスのデフォルト設定は recoveryoption=fixedretry retrycount=5 です。

その他のデバイスのデフォルト設定は recoveryoption=timebound iotimeout=300 です。

recoveryoption=default を指定すると、I/O 調整がデフォルトの設定になるという影響もあります。

p.348 の「I/O 調整機構の設定」を参照してください。

メモ: I/O エラーへの応答の設定はシステムの再起動後にも保持されます。

I/O 調整機構の設定

デフォルトでは、DMP（Dynamic Multi-Pathing）はすべてのパスの I/O 調整がオフになるように設定されます。エンクロージャ、アレイ名、アレイタイプへのパスに適用されている I/O 調整の現在の設定を表示するには、vxdmpadm getattr コマンドを使います。

p.350 の「リカバリオプション値の表示」を参照してください。

I/O 調整を有効にすると、統計情報収集デーモンのアクティビティのために、CPU とメモリに対して少しのオーバーヘッドがかかります。I/O 調整を無効にすると、デーモンは統計情報を収集せず、I/O 調整が再度有効になるまで活動なしのままになります。

I/O 調整をオフにするには、vxdmpadm setattr コマンドを次の形式で使います。

```
# vxdmpadm setattr ¥
  {enclosure enc-name|arrayname name|arraytype type} ¥
  recoveryoption=nothrottle
```

次の例は、エンクロージャ enc0 へのパスに対して I/O 調整を無効にする方法を示します。

```
# vxdmpadm setattr enclosure enc0 recoveryoption=nothrottle
```

vxdmpadm setattr コマンドを使うと、指定したエンクロージャ、ディスクアレイ名、アレイタイプへのパスで I/O 調整を有効にできます。

```
# vxdmpadm setattr ¥
  {enclosure enc-name|arrayname name|arraytype type}¥
  recoveryoption=throttle [iotimeout=seconds]
```

iotimeout 属性が指定されている場合、その引数は、DMP がそのパス上の I/O 調整を呼び出すまでに未処理の I/O 要求の成功を待機する時間を秒単位で指定します。iotimeout のデフォルト値は、10 秒です。iotimeout をもっと大きい値に設定すると、I/O 調整が呼び出される前に、潜在的に SCSI ドライバ内でより多くの I/O 要求がキューに入れられる原因になります。

たとえば次の例では、エンクロージャ enc0 に対して、iotimeout の値を 60 秒に設定します。

```
# vxdmpadm setattr enclosure enc0 recoveryoption=throttle ¥
  iotimeout=60
```

I/O 調整をデフォルト設定にリセットするには、次のように recoveryoption=default を指定します。

```
# vxdmpadm setattr arraytype A/A recoveryoption=default
```

上記のコマンドは、recoveryoption=nothrottle に対応するデフォルトの動作を設定します。上記のコマンドは、I/O エラーに対する応答のデフォルト動作も設定します。

p.347 の「I/O エラーに対する応答の設定」を参照してください。

メモ: I/O 調整設定はシステムの再起動後にも保持されます。

LIPP（Low-Impact Path Probing）の設定

LIPP（Low-Impact Path Probing）機能は vxdmpadm settune コマンドを使って有効または無効にできます。

```
# vxdmpadm settune dmp_low_impact_probe=[on|off]
```

パスのプローブは、同じ HBA とアレイポートに接続されているパスのサブセットをプローブすることによって最適化されます。パスのサブセットのサイズは dmp_probe_threshold チューニングパラメータによって制御できます。デフォルト値は 5 に設定されています。

```
# vxdmpadm settune dmp_probe_threshold=N
```

サブパスフェールオーバーグループ（SFG）の設定

サブパスフェールオーバーグループ（SFG）機能は、dmp_sfg_threshold チューニングパラメータを使って有効または無効にできます。チューニングパラメータのデフォルト値は 1 で、これは機能が有効であることを表します。

機能を無効にするには、dmp_sfg_threshold チューニングパラメータの値を 0 に設定します。

```
# vxdmpadm settune dmp_sfg_threshold=0
```

機能を有効にするには、dmp_sfg_threshold の値を、SFG をトリガするのに必要なパスのエラー数に設定します。

```
# vxdmpadm settune dmp_sfg_threshold=N
```

サブパスフェールオーバーグループの ID を表示するには、次のコマンドを使います。

```
# vxdmpadm getportids {ctlr=ctlr_name | dmpnodename=dmp_device_name
 ¥
    | enclosure=enclr_name | path=path_name}
```

リカバリオプション値の表示

エンクロージャ、アレイ名、アレイタイプへのパスに適用されている I/O 要求エラー時の処理に対する現在の設定を表示するには、次の Dynamic Multi-Pathing（DMP）コマンドを使います。

```
# vxdmpadm getattr ¥
  {enclosure enc-name|arrayname name|arraytype type} ¥
  recoveryoption
```

次の例では、vxdmpadm getattr コマンドを使って、エンクロージャで設定された recoveryoption オプション値を表示します。

```
# vxdmpadm getattr enclosure HDS9500-ALUA0 recoveryoption
ENCLR-NAME     RECOVERY-OPTION  DEFAULT[VAL]    CURRENT[VAL]
===============================================================
HDS9500-ALUA0  Throttle         Nothrottle[0]   Nothrottle[0]
HDS9500-ALUA0  Error-Retry      Timebound[300] Timebound[300]
```

このコマンドの出力は、デフォルトと現在のポリシーオプションとその値を示します。

表 11-7 に、エラー後の I/O 再試行に関するリカバリオプションの設定の概略を示します。

表 11-7　エラー後の I/O 再試行に関するリカバリオプション

リカバリオプション	使用可能な設定	説明
recoveryoption=fixedretry	Fixed-Retry (retrycount)	I/O がエラーになった場合、DMP はエラーとなった I/O 要求を指定回数再試行します。

リカバリオプション	使用可能な設定	説明
recoveryoption=timebound	Timebound (iotimeout)	I/O がエラーになった場合、DMP はエラーとなった I/O 要求を指定時間(秒単位)再試行します。

表 11-8 に、I/O 調整に関するリカバリオプションの設定の概略を示します。

表 11-8 I/O 調整に関するリカバリオプション

リカバリオプション	使用可能な設定	説明
recoveryoption=nothrottle	なし	I/O 調整は使われません。
recoveryoption=throttle	Timebound (iotimeout)	指定時間(秒単位)以内に I/O 要求が戻らない場合、DMP はパスを調整します。

DMP パスリストアポリシーの設定

DMP(Dynamic Multi-Pathing)は、指定した時間間隔に基づきパスの状態を監視するカーネルタスクを保守します。パスに対して実行される分析の種類は、設定されたチェックポリシーに応じて変わります。

メモ: DMP パスリストアタスクでは、`vxdmpadm disable` を使って無効にしたコントローラを介するパスに対する無効化の状態は変更されません。

DMP パスリストアポリシーを設定する場合は、パスリストアスレッドを停止してから、新しい属性でそのスレッドを再起動する必要があります。

p.353 の「DMP パスリストアスレッドの停止」を参照してください。

次のいずれかのリストアポリシーを設定するには、`vxdmpadm settune dmp_restore_policy` コマンドを使います。ポリシーは、リストアスレッドが停止されるか、または `vxdmpadm settune` コマンドを使ってこれらの値が変更されるまで有効なままです。

- `check_all`
  パスリストアスレッドは、システム上のすべてのパスを分析し、オンライン状態に戻っているパスを有効にするとともに、アクセスできないパスを無効にします。このポリシーを設定するコマンドは、次のとおりです。

  ```
  # vxdmpadm settune dmp_restore_policy=check_all
  ```

- `check_alternate`

パスリストアスレッドは、オンラインに戻っているパスを有効にするとともに、少なくとも 1 つの代替パスが正常であるかどうかをチェックします。この条件が満たされない場合は、通知が生成されます。このポリシーを使うと、正常なすべてのパスで inquiry コマンドを実行する必要がないため、使用可能なパスが多数存在する場合には、check_all に比べて無駄が少なくなります。各 DMP ノードのパスが 2 つのみの場合、このポリシーは check_all と同じです。このポリシーを設定するコマンドは、次のとおりです。

```
# vxdmpadm settune dmp_restore_policy=check_alternate
```

- check_disabled
  デフォルトのパスリストアポリシーです。パスリストアスレッドは、ハードウェアの故障が原因で以前に無効にされたパスの状態をチェックし、オンライン状態に戻っている場合はそれらのパスを再び有効にします。このポリシーを設定するコマンドは、次のとおりです。

  ```
  # vxdmpadm settune dmp_restore_policy=check_disabled
  ```

- check_periodic
  パスリストアスレッドは、特定のサイクル数ごとに 1 回 check_all を実行し、残りのサイクルで check_disabled を実行します。使用可能なパスが多数存在する場合にこのポリシーを使うと、check_all の実行に伴って定期的に処理速度が低下することがあります。このポリシーを設定するコマンドは、次のとおりです。

  ```
  # vxdmpadm settune dmp_restore_policy=check_periodic
  ```

  check_all ポリシーの実行間隔のデフォルト値は 10 サイクルです。

dmp_restore_interval チューニングパラメータは、パスリストアスレッドがパスを検査する頻度を指定します。たとえば、次のコマンドを実行すると、ポーリング間隔が 400 秒に設定されます。

```
# vxdmpadm settune dmp_restore_interval=400
```

これらの設定はすぐに適用され、再ブート後も保持されます。現在の設定を表示するには、vxdmpadm gettune コマンドを使います。

p.1115 の「DMP チューニングパラメータ」を参照してください。

ポリシーまたは間隔を指定せずに vxdmpadm start restore コマンドを実行すると、パスリストアスレッドは、以前管理者が vxdmpadm settune コマンドを使って設定したポリシーと間隔の永続的な設定で起動します。管理者がポリシーまたは間隔を設定していない場合は、システムのデフォルト値が使われます。システムのデフォルトリストアポリシーは、check_disabled です。システムのデフォルト間隔は 300 秒です。

警告: システムのデフォルト値よりも短い間隔を指定すると、システムパフォーマンスに悪影響を与える可能性があります。

DMP パスリストアスレッドの停止

Dynamic Multi-Pathing（DMP）パスリストアスレッドを停止するには次のコマンドを使います。

```
# vxdmpadm stop restore
```

警告: パスリストアスレッドを停止すると、自動パスフェールバックは停止します。

DMP パスリストアスレッドの状態の表示

DMP（Dynamic Multi-Pathing）パスリストアスレッドの状態を示すチューニングパラメータ値を表示するには、`vxdmpadm gettune` コマンドを使用します。チューニングパラメータには次のものがあります。

dmp_restore_state　自動パスリストアカーネルスレッドの状態。

dmp_restore_interval　DMPパスリストアスレッドのポーリング間隔。

dmp_restore_policy　パスの状態の確認のために DMP が使うポリシー。

DMP パスリストアスレッドの状態を表示するには

◆ 次のコマンドを実行します。

```
# vxdmpadm gettune dmp_restore_state

# vxdmpadm gettune dmp_restore_interval

# vxdmpadm gettune dmp_restore_policy
```

アレイポリシーモジュール（Array Policy Modules）の設定

DMP（Dynamic Multi-Pathing）は、アレイと組み合わせて使うための APM（Array Policy Module）を提供します。APM は、次の操作を行うためのアレイ固有の手順とコマンドを定義する、動的にロード可能なカーネルモジュール（プラグイン）です。

- アレイ内のディスクへの複数のパスが有効な場合に I/O パスを選択する。
- パスフェールオーバー機構を選択する。
- パスに障害が発生した場合の代替パスを選択する。

- パスの変更を有効にする。
- SCSI 予約または予約解除要求に応答する。

DMP には、アレイを登録する際、これらの機能のデフォルトが設定されています。APM では、DMP によって提供される、または APM の別のバージョンによって提供される既存の設定値の一部または全部を変更できます。

次のコマンドを使って、システムに設定されたすべての APM を表示することができます。

```
# vxdmpadm listapm all
```

このコマンドの出力には各モジュールのファイル名、サポートされるアレイタイプ、APM 名、APM のバージョン、モジュールが現在ロードされ使われているかどうかが含まれます。

各モジュールの詳細情報を確認するには、モジュール名をコマンドの引数として指定します。

```
# vxdmpadm listapm module_name
```

APM を追加および設定するには、次のコマンドを使います。

```
# vxdmpadm -a cfgapm module_name [attr1=value1 ¥
      [attr2=value2 ...]]
```

オプションの設定属性およびその値は、各アレイの APM に特有のものです。詳しくは、アレイベンダーが提供するドキュメントを参照してください。

メモ: デフォルトでは、DMP は最新の APM を使います。DMP で以前の APM のバージョンを強制的に使うには、-a オプションの代わりに -u オプションを指定します。APM の最新バージョンが使用中でなければ、以前のバージョンに切り替わります。

-r オプションを指定すると、現在ロードされていない APM を削除することができます。

```
# vxdmpadm -r cfgapm module_name
```

vxdmpadm(1M) のマニュアルページを参照してください。

12

デバイスの動的再構成

この章では以下の項目について説明しています。

オンラインの Dynamic Reconfiguration について

次の種類のオンラインの Dynamic Reconfiguration を実行できます。

- DMP の制御下にある LUN のオンラインでの再設定
  p.364 の「DMP の制御下にある LUN のオンラインでの手動での再設定」を参照してください。
- アレイコントローラファームウェアの更新（無停止アップグレードとも呼ばれる）
  p.375 の「アレイコントローラファームウェアのオンラインでのアップグレード」を参照してください。

Dynamic Reconfiguration ツールでの DMP の制御下にある LUN のオンラインでの再設定

Dynamic Reconfiguration ツールを使用して DMP の制御下にある LUN のオンラインでの再設定を行うには次のタスクを実行します。

表 12-1

タスク	トピック
既存のターゲット ID からの LUN の動的削除	p.356 の「既存のターゲット ID からの LUN の動的削除」を参照してください。
新しいターゲット ID への LUN の動的追加	p.359 の「ターゲット ID への新しい LUN の動的追加」を参照してください。
既存のターゲット ID の LUN の置き換え	p.362 の「既存のターゲット ID からの LUN の置換」を参照してください。
LUN の特性の変更	p.373 の「アレイ側からの LUN の特性の変更」を参照してください。

既存のターゲット ID からの LUN の動的削除

DMP（Dynamic Multi-Pathing）には、既存のターゲット ID からの LUN の削除を簡素化する DR（Dynamic Reconfiguration）ツールがあります。各 LUN はホストからマッピングが解除されます。DMP はオペレーティングシステムのデバイススキャンを実行し、オペレーティングシステムのデバイスのツリーをクリーンアップします。

警告: Dynamic Reconfiguration ツールの外部のデバイス検出操作は、デバイス操作が完了するまで実行しないでください。

クラスタのすべてのノードに手順を実行します。

既存のターゲット ID から LUN を動的に削除するには

1 削除される LUN にホストされているすべてのアプリケーションとボリュームを停止します。

デバイスが Veritas Volume Manager（VxVM）によって使用中の場合は、次の手順を実行します。

- デバイスがディスクグループに属する場合、ディスクグループから削除します。

```
# vxdg -g dgname rmdisk daname
```

- vxdiskリストからディスクを削除します。
クラスタ内のすべてのノードでこの手順を実行します。

```
# vxdisk rm da-name
```

次に例を示します。

```
# vxdisk rm eva4k6k0_0
```

DMP デバイス上で Linux LVM を使っている LUN の場合は、LVM ボリュームグループからデバイスを削除します。

```
# vgreduce vgnamedevicepath
```

2 vxdiskadm ユーティリティを開始します。

```
# vxdiskadm
```

3 vxdiskadm メニューから[Dynamic Reconfiguration 機能]オプションを選択します。

4 [LUN を削除(Remove LUNs)]オプションを選択します。

5 list と入力するか、Return キーを押し、削除できる LUN のリストを表示します。使用中でない LUN は削除できます。

出力例を次に示します。

```
Select disk devices to remove: [<pattern-list>,all,list]: list
LUN(s) available for removal:
eva4k6k0_0
eva4k6k0_1
eva4k6k0_2
eva4k6k0_3
eva4k6k0_4
emc0_0119
```

6 LUN の名前、カンマ(,)で区切った LUN のリスト、正規表現を入力し、削除する LUN を指定します。

たとえば、emc0_0119. と入力します。

```
Select disk devices to Remove: [<pattern-list>,all,list,
file=<filename>,q] (default:list): emc0_0119
```

7 プロンプトで LUN の選択を確認します。

DMP は VxVM による使用から LUN を削除します。

8 次のプロンプトで、アレイ/ターゲットから LUN を削除します。

```
Remove Luns
Menu:
VolumeManager/Disk/DynamicReconfigurationOperations/RemoveLuns

INFO: Removing Lun [emc0_0119] from VxVM
INFO: LUN [emc0_0119] removed successfully from VxVM.
-----------------------------------------------------
Enclosure=emc0 AVID=0119
Device=emc0_0119 Serial=2200119000
PATH=sdad ctlr=c11 port=16c-0 [-]
PATH=sdah ctlr=c12 port=16c-0 [-]
PATH=sdaj ctlr=c12 port=16c-1 [-]
PATH=sdaf ctlr=c11 port=16c-1 [-]
-----------------------------------------------------
Please remove LUNs with Above details from array and press 'y'
to
continue removal or 'q' to quit :
```

9 次は、EMC Symmetrix のサンプルコマンドです。

```
# symmask -sid 822 -wwn 2001000e1ec307de -dir 16c -p 0 remove
devs 0119
# symmask -sid 822 -wwn 2001000e1ec307de -dir 16c -p 1 remove
devs 0119
# symmask -sid 822 -wwn 2001000e1ec307df -dir 16c -p 0 remove
devs 0119
# symmask -sid 822 -wwn 2001000e1ec307df -dir 16c -p 1 remove dev
 0119

# symmask -sid 822 refresh -nopr

Symmetrix FA/SE directors updated with contents of SymMask
Database 000290300822

When complete, respond to previous array prompt.

Please remove LUNs with Above details from array and
press 'y' to continue removal or 'q' to quit : y
```

10 DMP は VxVM による使用からのデバイスの削除を完了します。以下のような出力が表示されます。

```
Remove Luns
Menu:
VolumeManager/Disk/DynamicReconfigurationOperations/RemoveLuns

INFO: Checking/Removing stale device entries (if any).
INFO: Refreshing OS device Tree
INFO: Updating VxVM device tree
------------------------------------------------------------
Luns Removed
------------------------------------------------------------
emc0_0119
------------------------------------------------------------

Press <Enter> or <Return> to continue:
```

11 実行する動的再構成操作を指定します。

```
Specify Dynamic Reconfiguration Operation to be done:
Menu: VolumeManager/Disk/DynamicReconfigurationOperations

 1 Add Luns
 2 Remove Luns
 3 Replace Luns
 4 Replace HBA


 ? Display help about menu
 ?? Display help about the menuing system
 q Exit
```

Dynamic Reconfiguration ツールを終了するには、q を入力します。

ターゲット ID への新しい LUN の動的追加

DMP（Dynamic Multi-Pathing）には、新しいまたは既存のターゲット ID への新しい LUN の追加を簡素化する DR（Dynamic Reconfiguration）ツールがあります。1 つ以上の新しい LUN が複数の HBA ポートを介してホストにマップされます。認識する LUN に対してオペレーティングシステムのデバイススキャンが実行され、DMP 制御に追加されます。

警告: Dynamic Reconfiguration ツールの外部のデバイス検出操作は、デバイス操作が完了するまで実行しないでください。

クラスタの場合、クラスタ内のすべてのノードでこの手順を実行します。

ターゲット ID に新しい LUN を動的に追加するには

1 vxdiskadm ユーティリティを開始します。

```
# vxdiskadm
```

2 vxdiskadm メニューから[Dynamic Reconfiguration 機能]オプションを選択します。

3 [LUN を追加(Add LUNs)]オプションを選択します。

以下のような出力が表示されます。

```
Add Luns
Menu: VolumeManager/Disk/DynamicReconfigurationOperations/AddLuns

INFO: Refreshing OS device Tree
INFO: Updating VxVM device tree
Add LUNs from array, once done then press 'y' to continue
or 'q' to quit. :
```

4 次は、EMC Symmetrix のサンプルコマンドです。

```
# symmask -sid 822 -wwn 2001000e1ec307de -dir 16c -p 0 add devs
0119 -nopr
# symmask -sid 822 -wwn 2001000e1ec307de -dir 16c -p 1 add devs
0119 -nopr
# symmask -sid 822 -wwn 2001000e1ec307df -dir 16c -p 0 add devs
0119 -nopr
# symmask -sid 822 -wwn 2001000e1ec307df -dir 16c -p 1 add devs
0119 -nopr
# symmask -sid 822 refresh -nopr

Symmetrix FA/SE directors updated with contents of SymMask
Database 000290300822
```

5 プロンプトが表示されたら、アレイから LUN を追加します。

以下のような出力が表示されます。

```
Add LUNs from array, once done then press 'y' to continue
or 'q' to quit. : y

Add Luns
Menu: VolumeManager/Disk/DynamicReconfigurationOperations/AddLuns

INFO: Refreshing OS device Tree
INFO: Updating VxVM device tree
INFO: Updating partition table information and disk size
INFO: Number of Paths for Lun [emc0_0119] presented=4
INFO: Updating VxVM device tree
```

6 [y]を選択して DMP への LUN の追加を続行します。

DMP はオペレーティングシステムのデバイスツリーと VxVM デバイスツリーを更新します。新しく検出されたデバイスが認識可能になります。

```
-------------------------------------------------------
Luns Added
-------------------------------------------------------
Enclosure=emc0 AVID=0119
Device=emc0_0119 Serial=2200119000
PATH=sdaf ctlr=c11 port=16c-1 [-]
PATH=sdah ctlr=c12 port=16c-0 [-]
PATH=sdaj ctlr=c12 port=16c-1 [-]
PATH=sdad ctlr=c11 port=16c-0 [-]
-------------------------------------------------------

Press <Enter> or <Return> to continue:
```

7 実行する動的再構成操作を指定します。

```
Specify Dynamic Reconfiguration Operation to be done:
Menu: VolumeManager/Disk/DynamicReconfigurationOperations

 1 Add Luns
 2 Remove Luns
 3 Replace Luns
 4 Replace HBA


 ? Display help about menu
 ?? Display help about the menuing system
 q Exit

Select an operation to perform : q
```

Dynamic Reconfiguration ツールを終了するには、q を入力します。

既存のターゲット ID からの LUN の置換

DMP（Dynamic Multi-Pathing）には、既存のターゲット ID からの新しい LUN の交換を簡素化する DR（Dynamic Reconfiguration）ツールがあります。各 LUN はホストからマッピングが解除されます。DMP はオペレーティングシステムのデバイススキャンを実行し、オペレーティングシステムのデバイスのツリーをクリーンアップします。

警告: Dynamic Reconfiguration ツールの外部のデバイス検出操作は、デバイス操作が完了するまで実行しないでください。

クラスタの場合、クラスタ内のすべてのノードでこの手順を実行します。

既存のターゲット ID から LUN を動的に置換するには

1. 削除される LUN にホストされているすべてのアプリケーションとボリュームを停止します。

   デバイスが Veritas Volume Manager（VxVM）によって使用中の場合は、次の手順を実行します。

   - デバイスがディスクグループに属する場合、ディスクグループから削除します。

     ```
     # vxdg -g dgname rmdisk daname
     ```

   - vxdiskリストからディスクを削除します。
     クラスタ内のすべてのノードでこの手順を実行します。

     ```
     # vxdisk rm da-name
     ```

     次に例を示します。

     ```
     # vxdisk rm eva4k6k0_0
     ```

   DMP デバイス上で Linux LVM を使っている LUN の場合は、LVM ボリュームグループからデバイスを削除します。

   ```
   # vgreduce vgnamedevicepath
   ```

2. vxdiskadm ユーティリティを開始します。

   ```
   # vxdiskadm
   ```

3. vxdiskadm メニューから［Dynamic Reconfiguration 機能］オプションを選択します。
4. ［LUN を置換（Replace LUNs）］オプションを選択します。

   出力が置換可能な LUN の一覧を表示します。LUN 上に開いているものがなく、状態がオンラインまたは nolabel の場合は、その LUN を交換に利用できます。
5. 置換するLUN を 1 つ以上選択します。
6. プロンプトで LUN の選択を確認します。
7. アレイ/ターゲットから LUN を削除します。

8 削除を続行するために Dynamic Reconfiguration のツールに戻って[y]を選択します。

削除が正常に完了すると、Dynamic Reconfiguration ツールで LUN を追加するためのメッセージが表示されます。

9 プロンプトが表示されたら、アレイ/ターゲットから LUN を追加します。

10 [y]を選択して LUN の追加を続行します。

DMP はオペレーティングシステムのデバイスツリーと VxVM デバイスツリーを更新します。新しく検出されたデバイスが認識可能になります。

ホストバスアダプタのオンラインでの交換

DMP（Dynamic Multi-Pathing）には、既存のシステムからのホストバスアダプタの削除を簡素化する DR（Dynamic Reconfiguration）ツールがあります。

ホストバスアダプタをオンラインで交換するには

1 vxdiskadm ユーティリティを開始します。

```
# vxdiskadm
```

2 vxdiskadm メニューから[Dynamic Reconfiguration 機能]オプションを選択します。

3 [HBA を追加（Replace HBAs）]オプションを選択します。

出力が DMP に利用可能な HBA の一覧を表示します。

4 交換する HBA を 1 つ以上選択します。

5 プロンプトで HBA の選択を確認します。

6 ホストバスアダプタを交換します。

7 置換処理を続行するために Dynamic Reconfiguration のツールに戻って [y] を選択します。

DMP がオペレーティングシステムのデバイスツリーを更新します。

DMP の制御下にある LUN のオンラインでの手動での再設定

LUN の動的再設定では、アレイ設定コマンド、OS コマンド、Veritas Volume Manager コマンドが必要です。処理を正確に完了するには、コマンドをホストで正しい順序で実行する必要があります。

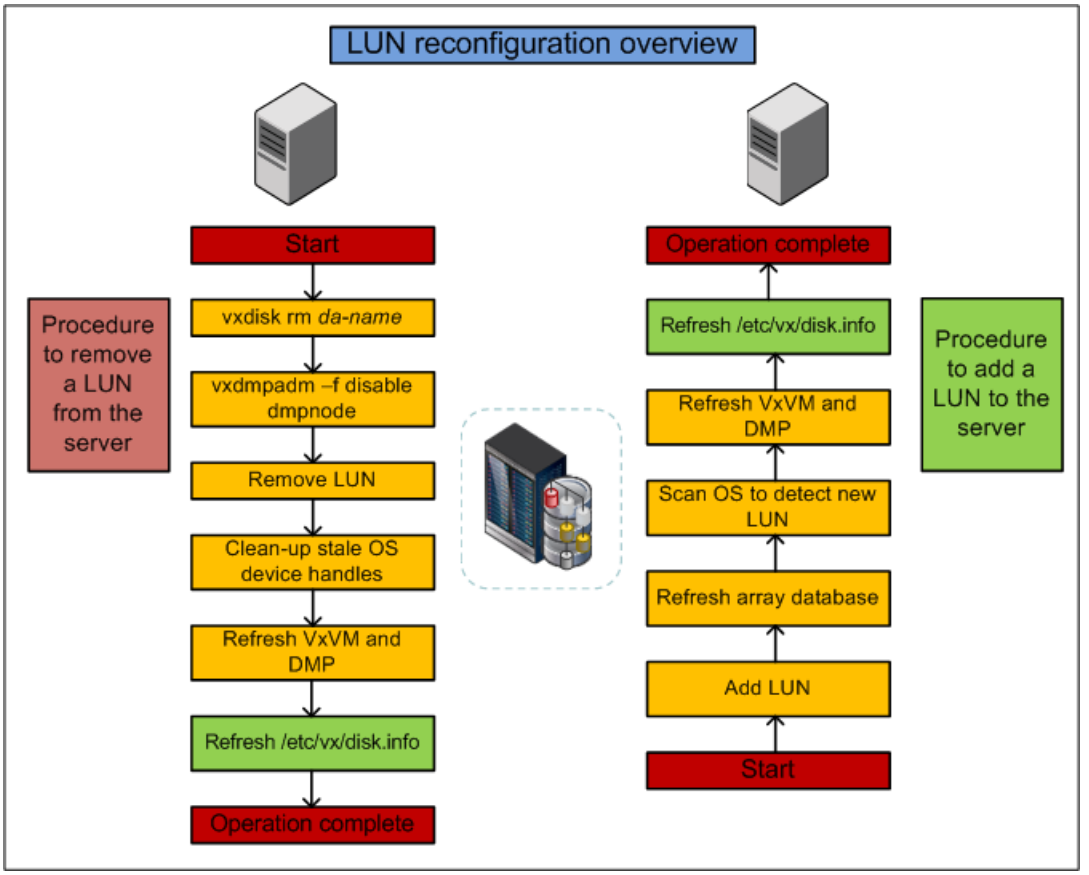
LUN の手動での再設定の概要

このセクションでは、LUNを手動で追加または削除するためのプレチェックと手順の概要のみを説明します。詳細な手順については、次の表に表示されたトピックで説明しています。

表 12-2

タスク	トピック
既存のターゲット ID からの LUN の動的削除	p.368 の「既存のターゲット ID から LUN を動的に手動での削除」を参照してください。
LUN の削除後のオペレーティングシステムデバイスツリーのクリーンアップ	p.373 の「LUN の削除後のオペレーティングシステムデバイスツリーの手動でのクリーンアップ」を参照してください。
LUN の追加または削除後のオペレーティングシステムデバイスツリーのスキャン	p.372 の「LUN の追加または削除後のオペレーティングシステムデバイスツリーのスキャン」を参照してください。
新しいターゲット ID への LUN の動的追加	p.370 の「新しいターゲット ID に新しい LUN を動的に手動での追加」を参照してください。
LUN の特性の変更	p.373 の「アレイ側からの LUN の特性の変更」を参照してください。

図 12-1 LUN の再設定の概要



プレチェック

LUN を手動で再設定する前に、次のプレチェックを実行します。

表 12-3 プレチェック

タスク	コマンド
/etc/vx/disk.info ファイルを確認する	# grep “0xffff” /etc/vx/disk.info
OS 層を更新する	# echo '- - -' > /sys/class/scsi_host/host$i/scan
OS デバイスハンドルをリストする	# lsscsi

タスク	コマンド
VxVM と DMP を更新する	`# vxdisk scandisks`
DDL layer/dev_t (デバイス番号)リストを更新する	`# vxddladm assign names`

メモ: 新しい LUN をプロビジョニングする前に、OS と VxVM がどちらもクリーンアップされていることを確認します。

LUN を手動で削除する

LUN を手動で削除する前に、次の手順を実行します。

表 12-4 LUN の削除手順

タスク	検証
ファイルシステムをマウント解除する	ディスクがディスクグループから削除されているかを確認します。
VxVM デバイスを閉じます。 `# vxdisk rm da-name`	VxVM デバイスが閉じているかを確認します。 `# vxdisk list`
DMP パスを無効にします。 `# vxdmpadm -f disable dmpnodename=da-name`	DMP パスが無効になっているかを確認します。 `# vxdmpadm getsubpaths dmpnodename=da-name`
LUN をサーバーからマスクする	LUN がアレイレベルで削除されているかを確認します。
OS デバイスハンドルをクリーンアップします。 `# echo 1 > /sys/block/device_name/device/delete`	OS デバイスハンドルがクリーンであるかを確認します。 `# lsscsi`
VxVM と DMP を更新します。 `# vxdisk scandisks`	
DDL layer/dev_t (デバイス番号) リストを更新する: `# vxddladm assign names`	

LUN を手動で追加する

LUN を手動で追加するには

1. LUN をサーバーの HBA WWN (Worldwide Name) にマスクします。
2. アレイのデータベースを更新します。
3. OS デバイスハンドルを更新します。
4. VxVM と DMP を更新します。
5. `/etc/vx/disk.info` ファイルを更新します。

既存のターゲット ID から LUN を動的に手動での削除

この場合、LUN のグループはホスト HBA ポートからマッピングが解除され、オペレーティングシステムのデバイススキャンが実行されます。それ以降 LUN をシームレスに追加するには、追加の手順でオペレーティングシステムのデバイスツリーをクリーンアップします。

上位レベルの手順と SFCFSHA コマンドは共通です。ただし、オペレーティングシステムのコマンドは Linux のバージョンによって変わることがあります。次の手順では例として Linux Suse10 を使います。

既存のターゲット ID から LUN を動的に削除するには

1. Dynamic Reconfiguration の実行前に、`dmp_cache_open` チューニングパラメータが `on` に設定されていることを確認します。この設定がデフォルト値です。

   ```
   # vxdmpadm gettune dmp_cache_open
   ```

   チューニングパラメータが `off` に設定されている場合、`dmp_cache_open` チューニングパラメータを `on` に設定します。

   ```
   # vxdmpadm settune dmp_cache_open=on
   ```

2. ホストから削除する LUN を識別します。次のいずれかを実行します。
   - ストレージアレイ管理を使い、LUN のアレイボリューム ID（AVID）を識別します。
   - アレイから AVID が報告されない場合は、LUN のインデックスを使います。
3. VxVM の制御下にある LUN の場合、次の手順を実行します。
   - `vxevac` コマンドを使って、LUN からデータを退避します。
     `vxevac`（1M）のオンラインマニュアルページを参照してください。
     データが待避されたら、次のコマンドを入力し、ディスクグループから LUN を削除します。

     ```
     # vxdg -g diskgroup rmdisk da-name
     ```

- データが待避されておらず、LUN がサブディスクまたはディスクグループの一部になっている場合は、次のコマンドを入力し、ディスクグループから LUN を削除します。ディスクが共有ディスクグループの一部ならば、-k オプションを使って強制的に削除する必要があります。

  ```
  # vxdg -g diskgroup -k rmdisk da-name
  ```

4 DMP デバイス上で Linux LVM を使っている LUN の場合は、LVM ボリュームグループからデバイスを削除します。

```
# vgreduce vgnamedevicepath
```

5 AVID または LUN のインデックスを使う場合は、ストレージアレイ管理を使い、手順 2 で識別した LUN のマップやマスクを解除します。

6 vxdisk リストから LUN を削除します。クラスタ内のすべてのノードで次のコマンドを入力します。

```
# vxdisk rm da-name
```

この手順は必須です。この手順を実行しない場合は、DMP デバイスツリーにゴーストパスが表示されます。

7 手順「6」で削除したデバイスの Linux SCSI デバイスツリーをクリーンアップします。

p.373 の「LUN の削除後のオペレーティングシステムデバイスツリーの手動でのクリーンアップ」を参照してください。

この手順は必須です。後から新しい LUN をホストに追加する場合は、オペレーティングシステムの SCSI デバイスツリーをクリーンアップして SCSI ターゲット ID を再利用のために解放する必要があります。

8 オペレーティングシステムのデバイスツリーをスキャンします。

p.372 の「LUN の追加または削除後のオペレーティングシステムデバイスツリーのスキャン」を参照してください。

9 SFCFSHA を使ってデバイススキャンを実行します。この操作はクラスタのすべてのノードで実行する必要があります。次のいずれかのコマンドを入力します。

- `# vxdctl enable`
- `# vxdisk scandisks`

10 次のコマンドを使って DMP のデバイス名データベースを更新します。

```
# vxddladm assign names
```

11 次の質問の回答により、LUN がクリーンに削除されたことを検証します。

- デバイスツリーがクリーンであるかどうか。
  オペレーティングシステムのメタノードが /sys/block ディレクトリから削除されていることを確認します。
- すべての該当する LUN が削除されたかどうか。
  vxdisk list コマンド出力などの DMP ディスクレポート作成ツールで、LUN が正常にクリーンアップされたかどうかを判断します。
- vxdisk list 出力は正しいかどうか。
  vxdisk list の出力に正しい数のパスが表示されていて、ゴーストディスクが含まれていないことを確認します。

これらの質問の回答に「いいえ」がある場合は、手順 5 に戻り、必要な手順を実行します。

すべての質問の回答が「はい」である場合は、LUN は正常に削除されています。

新しいターゲット ID に新しい LUN を動的に手動での追加

この場合、新しい LUN グループが複数の HBA ポートを介してホストにマップされます。認識する LUN に対してオペレーティングシステムのデバイススキャンが実行され、DMP 制御に追加されます。

上位レベルの手順と SFCFSHA コマンドは共通です。ただし、オペレーティングシステムのコマンドは Linux のバージョンによって変わることがあります。次の手順では例として Linux Suse10 を使います。

新しいターゲット ID に新しい LUN を動的に追加するには

1. Dynamic Reconfiguration の実行前に、dmp_cache_open チューニングパラメータが on に設定されていることを確認します。この設定がデフォルト値です。

   ```
   # vxdmpadm gettune dmp_cache_open
   ```

   チューニングパラメータが off に設定されている場合、dmp_cache_open チューニングパラメータを on に設定します。

   ```
   # vxdmpadm settune dmp_cache_open=on
   ```

2. ホストに追加する LUN を識別します。次のいずれかを実行します。
   - ストレージアレイ管理を使い、LUN のアレイボリューム ID（AVID）を識別します。
   - アレイから AVID が報告されない場合は、LUN のインデックスを使います。
3. 複数のホスト上の新しいターゲット ID に対して、LUN をマップするかマスクします。

4 オペレーティングシステムのデバイスをスキャンします。

p.372 の「LUN の追加または削除後のオペレーティングシステムデバイスツリーのスキャン」を参照してください。

すべての LUN が追加されたことが確認されるまで、手順 2 と手順 3 を繰り返します。

5 SFCFSHA を使ってデバイススキャンを実行します。この操作はクラスタのすべてのノードで実行する必要があります。次のいずれかのコマンドを入力します。

- `# vxdctl enable`
- `# vxdisk scandisks`

6 次のコマンドを使って DMP のデバイス名データベースを更新します。

```
# vxddladm assign names
```

7 次の質問の回答により、LUN が正しく追加されたことを検証します。

- 新しくプロビジョニングした LUN が `vxdisk list` の出力に表示されるかどうか。
- 各 LUN に設定されたパスがあるかどうか。

これらの質問の回答に「いいえ」がある場合は、手順 2 に戻り、もう一度手順を開始します。

すべての質問の回答が「はい」である場合は、LUN は正常に追加されています。ここで LUN をディスクグループに追加し、新しいボリュームを作成または既存のボリュームを拡張できます。

dmp_native_support チューニングパラメータが ON に設定されていて、新しい LUN に VxVM ラベルがない場合、または新しい LUN が TPD ドライバによって要求されていない場合、LUN は LVM で使用できます。

オペレーティングシステムのデバイスツリーがクリーンアップされていない場合のターゲット ID 再利用の検出について

以前有効であったオペレーティングシステムのデバイスエントリがクリーンアップされていない LUN または LUN セットのプロビジョニングを再度試みると、次のメッセージが表示されます。また、DMP デバイススキャンと DMP の再設定の間には、DMP の再設定が一時的に禁止されます。

p.373 の「LUN の削除後のオペレーティングシステムデバイスツリーの手動でのクリーンアップ」を参照してください。

```
VxVM vxdisk ERROR V-5-1-14519 Data Corruption Protection Activated
- User Corrective Action Needed
```

```
VxVM vxdisk INFO V-5-1-14521 To recover, first ensure that the OS
device tree is up to date (requires OS specific commands).

VxVM vxdisk INFO V-5-1-14520 Then, execute 'vxdisk rm' on the
following devices before reinitiating device discovery. <DA names>
```

前のメッセージの意味は新しい LUN が古い LUN のターゲット ID の再利用を試みるということです。デバイスエントリはクリーンされていません。したがって新しい LUN はターゲット ID を利用できません。DMP ではオペレーティングシステムのデバイスツリーがクリーンアップされるまでこの処理は実行されません。

LUN の追加または削除後のオペレーティングシステムデバイスツリーのスキャン

LUN を追加または削除したら、オペレーティングシステムのデバイスツリーをスキャンし、操作が正常に完了したことを確認します。

Linux には、SCSI バスを再スキャンし、バスにマップされているデバイスを識別するための方法が複数用意されています。これらには次の方法が含まれます。

- /sys ディレクトリでの SCSI スキャン機能
- HBA ベンダーのユーティリティ

SCSI スキャン機能を使ってスキャンするには

◆ 次のコマンドを入力します。

```
# echo '- - -' > /sys/class/scsi_host/host$i/scan
```

3 つのハイフンはチャネル、ターゲット、LUN 番号を指し、*host$i* はホストバスアダプタのインスタンスです。この例では、このホストバスアダプタのインスタンス経由で参照できるすべてのチャネル、ターゲット、LUN をスキャンします。

HBA ベンダーのユーティリティを使ってスキャンするには

◆ HBA ユーティリティのベンダーの指示に従います。次のような例があります。

- QLogic 社は新しく追加された LUN を動的にスキャンするスクリプトを提供しています。このスクリプトは QLogic 社の Web サイトからダウンロードできます。スクリプトを実行するには、次のコマンドを入力します。

  ```
  # ./ql-dynamic-tgt-lun-disc.sh
  ```

- Emulex 社は HBAnywhere スクリプトを提供しています。このスクリプトは Emulex 社の Web サイトからダウンロードできます。このスクリプトには、新しく追加された LUN を動的にスキャンする LUN スキャンユーティリティが含まれます。ユーティリティを実行するには、次のコマンドを入力します。

  ```
  # lun_scan all
  ```

LUN の削除後のオペレーティングシステムデバイスツリーの手動でのクリーンアップ

LUN を削除した後は、オペレーティングシステムのデバイスツリーをクリーンアップする必要があります。

オペレーティングシステムのコマンドは Linux のバージョンによって変わることがあります。次の手順では SUSE 10. を使います。これらの手順のいずれかで必要な結果が得られなければ、Novell 社のサポートに連絡してください。

LUN の削除後にオペレーティングシステムのデバイスツリーをクリーンアップするには

1. オペレーティングシステムのデータベースからデバイスを削除します。次のコマンドを入力します。

   ```
   # echo 1 > /sys/block/$PATH_SYS/device/delete
   ```

   PATH_SYS は削除するデバイスの名前です。

2. 次のコマンドを入力したときにデバイスが表示されてはいけません。この手順で、LUN が削除されたことを確認します。

   ```
   # lsscsi | grep PATH_SYS
   ```

3. LUN を削除したら、デバイスをクリーンアップします。次のコマンドを入力します。

   ```
   # echo "- - -" > /sys/class/scsi_host/host$I/scan
   ```

   3 つのハイフンはチャネル、ターゲット、LUN 番号を指し、*host$i* はホストバスアダプタのインスタンスです。この例では、このホストバスアダプタのインスタンス経由で参照できるすべてのチャネル、ターゲット、LUN をクリーンアップします。

アレイ側からの LUN の特性の変更

アレイで LUN の特性を変更できる場合があります。ほとんどの場合、デバイスに変更された性質が適用される前に、完全にデバイスの使用を停止する必要があります。LUN のプロパティを変更する前にデバイスをオフラインにし、設定後にデバイスを再びオンラインにすることをお勧めします。

EMC BCV や SRDF 操作など特定の場合、処理中にデバイスをオンラインの状態に保つことができます。

クラスタの場合、クラスタ内のすべてのノードでこの手順を実行します。

LUN のプロパティを変更するには

1. デバイスでホストされているすべてのアプリケーションおよびボリュームを停止します。

デバイスが Veritas Volume Manager（VxVM）によって使用中の場合は、次の手順を実行します。

- デバイスがディスクグループに属する場合、ディスクグループから削除します。

  ```
  # vxdg -g dgname rmdisk da_name
  ```

- ディスクをオフラインにします。
  クラスタ内のすべてのノードでこの手順を実行します。

  ```
  # vxdisk offline da_name
  ```

  次に例を示します。

  ```
  # vxdisk offline eva4k6k0_0
  ```

DMP デバイス上で Linux LVM を使っている LUN の場合は、LVM ボリュームグループからデバイスを削除します。

```
# vgreduce vgnamedevicepath
```

2 LUN の特性を変更します。

3 デバイスをオンラインにします。

Veritas Volume Manager ディスクの場合:

- デバイスをオンラインにします。

  ```
  # vxdisk online da_name
  ```

- ディスクグループにディスクを追加します。

  ```
  # vxdg -g dgname adddisk da_name
  ```

DMP デバイス上で Linux LVM を使っている LUN の場合は、LVM ボリュームグループにデバイスを再び追加します。

```
# vgreduce vgnamedevicepath
```

4 DMP を使ってデバイススキャンを実行します。

クラスタ内のすべてのノードでこのコマンドを実行します。

```
# vxdisk scandisks
```

アレイコントローラファームウェアのオンラインでのアップグレード

ストレージアレイサブシステムには、修正、パッチ、機能のアップグレードとして、コードのアップグレードが必要です。ファイルシステムがマウントされ、I/O サービスがストレージに提供されている場合は、これらのアップグレードをオンラインで実行できます。

ストレージサブシステムには、冗長性用に複数のコントローラが搭載されています。オンラインアップグレードは一度に 1 つのコントローラで行われます。コントローラのいずれかでオンラインコントローラアップグレードを実行している間は、DMP（Dynamic Multi-Pathing）がすべての I/O を代替コントローラにフェールオーバーします。最初のコントローラが完全にコードを展開した後、新しいバージョンのコードを使って再ブート、リセットするとオンラインになります。他のコントローラは同じ処理を実行し、I/O を代替コントローラにフェールオーバーします。

メモ: この処理を通して、アプリケーションの I/O は影響を受けません。

アレイベンダーはこの処理にさまざまな名前を付けています。たとえば、EMC は CLARiiON アレイの無停止アップグレード（NDU）と呼んでいます。

A/A タイプのアレイでは、このオンラインアップグレード処理の間に特別な処理は必要ありません。A/P、A/PF、ALUA タイプのアレイでは、オンラインのコントローラコードアップグレード中に、ベンダ固有のアレイポリシーモジュール（APM）を通して、DMP がアレイ固有の処理を実行します。

コントローラがコードアップグレード時にリセットされ、再ブートされるときに、DMP は SCSI 状態を通してこの状態を検出します。DMP はすべての I/O をすぐに次のコントローラにフェールオーバーします。

アレイが完全に NDU をサポートしていない場合は、コントローラへのすべてのパスが短い間 I/O 用に使用できなくなる場合があります。アップグレードを始める前に、`dmp_lun_retry_timeout` チューニングパラメータを、I/O 用にコントローラを使用できなくなると予想される時間よりも長い時間に設定します。DMP では `dmp_lun_retry_timeout` で指定した時間が終わるまで、または I/O が成功するまで、どちらかの状態になるまでは I/O はエラーになりません。したがって、アプリケーション I/O を中断しないでファームウェアのアップグレードを実行できます。

たとえば、I/O のためにパスを 300 秒間利用できないと予想している場合は、次のコマンドを使います。

```
# vxdmpadm settune dmp_lun_retry_timeout=300
```

DMP は I/O の再試行を 300 秒間、または I/O が成功するまで実行します。

オンラインコントローラアップグレードまたは NDU をサポートしているアレイを確認するには、次の URL にあるハードウェア互換性リスト（HCL）を参照してください。

https://www.veritas.com/support/en_US/article.000126344

NVMe デバイスの手動での再フォーマット

NVMe デバイスのセクタサイズは、デバイスを VxVM から削除し、再フォーマットすることで変更できます。

NVMe デバイスを手動で再フォーマットするには

1. ディスクをオフラインにします。

```
# vxdisk offline r720xd-114217_intel_nvme0_0
```

2. デバイスを VxVM から削除します。

```
# vxdisk rm r720xd-114217_intel_nvme0_0
```

3. NVMe デバイスをオペレーティングシステムから削除します。

```
# echo 1 > /sys/block/nvme0n1/device/device/remove
```

4. VxVM デバイスツリーを更新します。

```
# vxdisk scandisks
```

5. デバイスが存在しないことを確認します。

```
# vxdisk list | grep nvme
```

6. NVMe PCI デバイスを再スキャンすることで、オペレーティングシステムのデバイスツリーに追加します。

```
# echo 1 > /sys/bus/pci/rescan
# echo 1 > /sys/bus/pci/drivers/nvme/0000¥:05¥:00.0/rescan
```

7 ISDCT (Intel® SSD Data Center Tool) を使用して、NVMe デバイスを必要なセクタサイズにフォーマットします。

```
# isdct start -intelssd 0 -nvmeformat LBAFormat=3
SecureEraseSetting=0 ¥
ProtectionInformation=0 MetadataSettings=0
WARNING! You have selected to format the drive!
Proceed with the format? (Y|N): y
Formatting...

- Intel SSD DC P3700 Series CVFT5456000V2P0EGN -

Status : NVMeFormat successful.
```

8 VxVM デバイスツリーを更新します。

```
# vxdisk scandisks
```

9 デバイスを確認します。

```
# vxdisk list | grep nvme
r720xd-114217_intel_nvme0_0 auto:none  -    - online invalid
```

13

デバイスの管理

この章では以下の項目について説明しています。

ディスク情報の表示

ディスクを使う前に、そのディスクが初期化され VxVM（Veritas Volume Manager）制御下に置かれているかどうかを知る必要があります。ディスクがディスクグループに属しているかどうかも確認する必要があります。ディスクグループに属していないディスク上には、ボリュームを作成できないためです。`vxdisk list` コマンドは、認識しているすべてのディスクのデバイス名、そのディスク名、各ディスクに関連するディスクグループ名および各ディスクの状態を表示します。

VxVM で認識されているすべてのディスクに関する情報を表示するには

◆ 次のコマンドを実行します。

```
# vxdisk list
```

VxVM により次のような出力が表示されます。

```
DEVICE        TYPE            DISK    GROUP   STATUS
emc_clariion0_26    auto:cdsdisk      -       -     online
emc_clariion0_27    auto:cdsdisk      -       -     online
emc_clariion0_107   auto:cdsdisk    dsk0    tcdg    online
emc_clariion0_108   auto:cdsdisk    dsk1    tcdg    online
emc_clariion0_110   auto:cdsdisk    dsk2    tcdg    online
emc_clariion0_111   auto:cdsdisk    dsk3    tcdg    online
emc_clariion0_144   auto:none       -       -     online invalid
```

[STATUS]列の[online invalid]という語句は、ディスクがまだ VxVM の制御下に追加されていないことを示します。これらのディスクは、以前 VxVM によって初期化された可能性も初期化されなかった可能性もあります。[online]として一覧表示されているディスクは、すでに VxVM の制御下に置かれています。

個々のディスクに関する情報を表示するには

◆ 次のコマンドを実行します。

```
# vxdisk [-v] list diskname
```

-v オプションを指定すると、ディスクに定義されているすべてのタグとタグ値が追加で表示されます。デフォルトでは、タグは表示されません。

VxVM は メディア形式の検出 (-o mfd オプションで既存の vxdisk list コマンドを使用) を使って、4K セクタのデバイスのサポートを利用します。

メディア形式の検出について

メディア形式の検出または -o mfd オプションは、ディスクの正しい形式の識別や、異なるオペレーティングシステム環境内でディスクを移動するときに発生する偶発的なディスクの消去の防止に役立ちます。

オペレーティングシステムのネーティブレイアウトに関する情報の表示

オペレーティングシステム間のネーティブレイアウトに関する情報を表示するには、次のコマンドを使います。

```
# vxdisk -o mfd list
```

異なるオペレーティングシステム環境で -o mfd オプションに vxdisk list を使用する重要性を次の例で図解します。

例 1: Linux 環境

-o mfd オプションを使用する場合:

```
# vxdisk -o mfd list sda | egrep "^info:|flags:"

info:       format=linux:ext4

flags:      unusable online ready private autoconfig invalid
```

-o mfd オプションを使用しない場合:

```
# vxdisk list sda | egrep "^info:|flags:"

info:       format=none

flags:      online ready private autoconfig invalid
```

例 2: Solaris 環境

-o mfd オプションを使用する場合:

```
# vxdisk -o mfd list disk_0 | egrep "^info:|flags:"

info:       format=solaris:ZFS

flags:      unusable online ready private autoconfig invalid
```

-o mfd オプションを使用しない場合:

```
# vxdisk list disk_0 | egrep "^info:|flags:"

info:       format=ZFS

flags:      ZFS online ready private autoconfig invalid
```

メモ: Veritas Volume Manager は、使用不可を示すフラグが付いているディスクを直接初期化できません。この場合、vxdisk init コマンドはエラーメッセージが表示されて失敗し、外部形式の削除が要求されます。

次に例を示します。

```
#vxdisk init disk_0

VxVM vxdisk ERROR V-5-1-0 Device disk_0 is in use.
If it is still desired to initialize this device for VxVM use, please
```

```
 remove
the foreign format signatures from each of the following partition(s)

using 'dd' command or some other tool.
PARTITION  TYPE
0          solaris:zfs
```

ディスクセクタサイズに関する情報の表示

ディスクセクタサイズに関する情報を表示する場合は、vxdisk list コマンドと vxmediadisc コマンドを使用できます。

詳しくは、次の例を参照してください。

例 1: vxdisk list コマンドの使用

```
# vxdisk list sdz | grep "^iosize"

iosize:    min=4096 (bytes) max=1024 (blocks)
```

例 2: vxmediadisc コマンドの使用

```
# /etc/vx/diag.d/vxmediadisc -p /dev/sdz | grep "Sector size"

vxmediadisc: Sector size:   4096
```

サポート対象のオペレーティングシステム

オペレーティングシステムとディスクセクタサイズのネイティブレイアウトに関する情報の表示は、次のオペレーティングシステムでサポートされます。

1. Linux (RHEL、SLES、サポート対象の RHEL 互換配布)

vxdiskadm を使ったディスク情報の表示

VxVM（Veritas Volume Manager）では、vxdiskadm プログラムを使ってディスク情報を表示できます。ディスク情報には、初期化済みかどうか、属するディスクグループ、ディスクの状態が表示されます。list コマンドは、認識しているすべてのディスクのデバイス名、そのディスク名、各ディスクに関連するディスクグループ名および各ディスクの状態を表示します。

ディスク情報を表示するには、次の手順を実行します。

1 vxdiskadm プログラムを起動し、メインメニューから、[**ディスク情報の一覧表示**(List disk information)]を選択します。

2 次のプロンプト画面で、表示するデバイスの名前を入力するか、すべてのデバイスを一覧表示するために「all」を入力します。

```
List disk information
Menu: VolumeManager/Disk/ListDisk

VxVM INFO V-5-2-475 Use this menu operation to display a list of

disks. You can also choose to list detailed information about
the disk at a specific disk device address.

Enter disk device or "all" [<address>,all,q,?] (default: all)
```

- 「all」を入力した場合は、VxVM により、すべてのデバイスのデバイス名、ディスク名、グループ、状態が表示されます。
- デバイスの名前を入力した場合は、VxVM により、完全なディスク情報(デバイス名、ディスクのタイプ、ディスクのパブリックリージョンおよびプライベートリージョンに関する情報を含む)が表示されます。

この情報を確認した後、Return キーを押すと、メインメニューに戻ります。

ディスクデバイスの名前の付け方の変更

ディスクには、エンクロージャに基づく命名規則、またはオペレーティングシステムの名前の付け方を使えます。DMP コマンドでは、現在の名前の付け方に従ってデバイス名が表示されます。

デフォルトの名前の付け方は、エンクロージャに基づく命名規則(EBN)です。

ネーティブボリュームで DMP (Dynamic Multi-Pathing) を使う場合、ディスクの名前の付け方は EBN にする必要があり、use_avid 属性を yes、永続性の属性を yes にする必要があります。

ディスクの名前の付け方を変更するには、次の作業を実行します。

- vxdiskadm メインメニューで Change the disk naming scheme を選択して、SFCFSHA で使うディスクの名前の付け方を変更します。確認のメッセージが表示されたら、名前の付け方を変更する場合は y を入力します。

  または

  コマンドラインから名前の付け方を変更します。エンクロージャに基づく命名規則を選択するには、次のコマンドを使います。

  ```
  # vxddladm set namingscheme=ebn [persistence={yes|no}] ¥
  [use_avid={yes|no}] [lowercase={yes|no}]
  ```

  オペレーティングシステムに基づく名前の付け方を選択するには、次のコマンドを使います。

  ```
  # vxddladm set namingscheme=osn [persistence={yes|no}] ¥
  [lowercase=yes|no]
  ```

  オプションの persistence 引数を使うと、ディスクハードウェアを再設定してシステムを再ブートした後に、SFCFSHA で表示するディスクデバイスの名前を変更しないでおくかどうかを選択できます。デフォルトでは、エンクロージャに基づく命名規則は永続的です。オペレーティングシステムに基づく名前の付け方はデフォルトでは永続的ではありません。

  名前の付け方を変更しないで名前の永続性のみ変更するには、現在の名前の付け方に vxddladm set namingscheme コマンドを実行し、永続性の属性を指定します。

  デフォルトでは、ASL によって指定された名前に大文字が含まれていても、エンクロージャ名は小文字に変換されます。したがって、エンクロージャに基づくデバイス名は小文字になります。小文字への変換を無効にするには lowercase=no オプションを設定します。

  エンクロージャに基づく命名規則では、use_avid オプションを使って、デバイス名のインデックス番号にアレイのボリューム ID を使うかどうかを指定します。デフォルトは use_avid=yes です。これは、デバイスに *enclosure_avid* のように名前を付けることを示します。use_avid を no に設定すると、DMP デバイスに *enclosure_index* のように名前が付けられます。インデックス番号は、デバイスが LUN シリアル番号でソートされた後に割り当てられます。

  どちらの方法でも変更は即座に反映されます。

  p.385 の「永続的なデバイス名の再生成」を参照してください。

ディスクの名前の付け方の表示

DMP（Dynamic Multi-Pathing）でのディスクの名前の付け方は、オペレーティングシステムに基づく名前の付け方またはエンクロージャに基づく名前の付け方に設定できます。

このコマンドは SFCFSHA ディスクの名前の付け方が現在設定されているかどうかを表示します。また、永続性が有効になっているかどうかなど、ディスクの名前の付け方に関する属性が表示されます。

現在のディスクの名前の付け方とその操作モードを表示するには、次のコマンドを使います。

```
# vxddladm get namingscheme
NAMING_SCHEME    PERSISTENCE LOWERCASE USE_AVID
==============================================
Enclosure Based  Yes         Yes       Yes
```

DMP ノードのカスタム名の設定

DMP（Dynamic Multi-Pathing）ノード名はディスクへの複数のパスを表すメタデバイス名です。DDL（Device Discovery Layer）は、Storage Foundation Cluster File System High Availability（SFCFSHA）の名前の付け方に従ってデバイス名から DMP ノード名を生成します。

DMP ノードのカスタム名を指定できます。ユーザーが指定した名前は、名前の永続性が無効になっても永続的に保持されます。

すでにデバイスで使用中のカスタム名は、割り当てることができません。ただし、DDL が生成する名前と同じ名前の付け方に従って名前を割り当てると、デバイスが追加されるときに名前の衝突が起きる可能性があります。DMP デバイスのユーザー定義の名前が、DDL によって生成された別の DMP デバイスの名前と同じ場合、`vxdisk list` コマンドの出力では、デバイスの 1 つがエラーとして表示されます。

DMP のノードのカスタム名を指定するには

- 次のコマンドを実行します。

  ```
  # vxdmpadm setattr dmpnode dmpnodename name=name
  ```

名前は入力ファイルから割り当てることもできます。これにより、意味のある名前を使ってシステムの DMP ノードをカスタマイズできます。

エンクロージャのカスタム名を指定するには

- 次のコマンドを実行します。

  ```
  # vxdmpadm setattr enclosure enc_name name=custom_name
  ```

ファイルから DMP ノードを割り当てるには

1 設定のデバイスの名前で設定されるファイルを取得するには、次のコマンドを使います。

```
# vxddladm -l assign names  > filename
```

サンプルファイルは必要な形式を示し、カスタム名を指定するためのテンプレートとして機能します。

スクリプト vxgetdmpnames を使って、設定内のデバイスから設定されたサンプルファイルを取得します。

2 必要に応じてファイルを修正します。ファイルの正しい形式を必ず保持してください。

3 名前を割り当てるには、次のコマンドにファイルの名前とパスを指定してください。

```
# vxddladm assign names file=pathname
```

カスタム名を消去するには

◆ 名前を消去し、デフォルトのオペレーティングシステムに基づく名前付けまたはエンクロージャに基づく名前付けを使うには、次のコマンドを使います。

```
# vxddladm -c assign names
```

永続的なデバイス名の再生成

永続的なデバイス命名機能では、ディスクデバイスの名前はシステムの再ブート後も維持されます。デバイス検出層（DDL）は永続的なデバイス名のデータベースに従ってデバイス名を割り当てます。

オペレーティングシステムに基づく名前の付け方を選択した場合、各ディスク名には通常、ディスクへのパスのうちいずれかの名前が付けられます。ハードウェアを再構成して再ブートすると、再構成前と異なる名前がディスクへのパスに対して生成されます。従って、永続的なデバイス名は実際のパスに対応しなくなる場合があります。これにより、ディスクが使用不可になることはありませんが、ディスク名がそのパスのいずれかと関連することはなくなります。

同様にエンクロージャベースの命名法を選択すると、デバイス名はエンクロージャ名とインデックス番号によって決まります。アレイによって開示される LUN の順序がハードウェア構成により変更される場合、永続的なデバイス名は現在のインデックスを反映しないことがあります。

永続的なデバイス名を再生成するには、次の手順を実行します。

- 永続的な名前のリポジトリを再生成するには、次のコマンドを使います。

  ```
  # vxddladm [-c] assign names
  ```

  -c オプションは、ユーザーが指定したすべての名前を消去し、自動生成された名前と置換します。

  -c オプションを指定しない場合、ユーザーが指定した既存の名前は維持されますが、オペレーティングシステムベースの名前とエンクロージャベースの名前は再生成されます。

サードパーティ製ドライバ制御のエンクロージャに対するデバイスの命名の変更

デフォルトでは、サードパーティ製ドライバ(TPD)制御のエンクロージャは TPD が割り当てたノード名に基づいた疑似デバイス名を使います。デバイスの命名をネーティブに変更すると、デバイスには他の Storage Foundation Cluster File System High Availability (SFCFSHA)デバイスと同じ形式で名前が付けられます。デバイスは命名規則がどちらに設定されているかに応じて、オペレーティングシステムの名前(OSN)またはエンクロージャに基づく名前(EBN)を使います。

p.384 の「ディスクの名前の付け方の表示」を参照してください。

TPD 制御のエンクロージャに対するデバイスの命名を変更するには

- ディスクエンクロージャを制御するサードパーティ製ドライバ(TPD)が、適切な ASL (Array Support Library)によって共存をサポートしている場合、デフォルトの動作では、TPD が割り当てたノード名に基づいてデバイス名が割り当てられます。vxdmpadm コマンドを使うと、TPD 割り当ての名前とオペレーティングシステムが認識しているデバイス名を切り替えることができます。

  ```
  # vxdmpadm setattr enclosure エンクロージャ名 (Enclosure name)
  tpdmode=native|pseudo
  ```

  tpdmode 属性の引数では、オペレーティングシステムが使う名前 (native) または TPD 割り当てのノード名 (pseudo) を指定します。

  次の例は、pp_emc_clariion0 という名前のエンクロージャに対して、TPD に基づく命名とオペレーティングシステムに基づく命名を切り替えるときのコマンドの使い方を示しています。この例では、デバイスの名前の付け方を OSN に設定しています。

  ```
  # vxdisk list

  DEVICE       TYPE            DISK         GROUP        STATUS
  emcpowerp    auto:cdsdisk    -            -            online
  emcpowerq    auto:cdsdisk    -            -            online
  ```

```
emcpowerr    auto:cdsdisk    -            -            online
emcpowers    auto:cdsdisk    -            -            online
emcpowert    auto:cdsdisk    -            -            online

# vxdmpadm setattr enclosure pp_emc_clariion0 tpdmode=native

# vxdisk list

DEVICE       TYPE            DISK         GROUP        STATUS
sde          auto:cdsdisk    -            -            online
sdf          auto:cdsdisk    -            -            online
sdg          auto:cdsdisk    -            -            online
sdh          auto:cdsdisk    -            -            online
sdi          auto:cdsdisk    -            -            online
```

tpdmode を native に設定すると、最小のデバイス番号のパスが表示されます。

アレイボリューム識別子（AVID）の属性について

DMP はアレイボリューム ID（AVID）と呼ばれるアレイ固有の属性を使って、エンクロージャに基づく名前を DMP メタデバイスに割り当てます。AVID はアレイによって提供される、LUN 用の一意識別子です。アレイに対応する ASL（Array Support Library）が AVID プロパティを提供します。アレイエンクロージャ内で、DMP は DMP メタノード名内のインデックスとして、AVID を使います。DMP メタノード名は、*enclosureID_AVID* という形式になっています。

vxdmpadm getdmpnode などの SFCFSHA ユーティリティは、AVID プロパティを含む DMP メタノード名を表示します。アレイ管理インターフェース（GUI または CLI）に表示される LUN に DMP メタノード名を関連付けるには、AVID を使います。

ASL からアレイボリューム ID プロパティが提供されない場合は、DMP がインデックス番号を生成します。DMP はアレイから認識されたデバイスを LUN のシリアル番号でソートしてから、インデックス番号を割り当てます。この場合、DMP メタノード名は、*enclosureID_index* という形式になります。

エンクロージャに基づく命名規則とアレイボリューム ID（AVID）属性

DMP（Dynamic Multi-Pathing）はデフォルトで、アレイボリューム ID（AVID）と呼ばれるアレイ固有の属性を使って、エンクロージャベースの名前を DMP メタデバイスに割り当てます。AVID は、アレイによって提供される LUN の重複のない ID を提供します。アレイに対応する ASL が AVID プロパティを提供します。アレイエンクロージャ内で、DMP は DMP メタノード名内のインデックスとして、AVID を使います。DMP メタノード名は、*enclosureID_AVID* という形式になっています。

エンクロージャに基づく命名規則(EBN)に AVID が導入されたことにより、ストレージデバイスの識別がずっと容易になりました。アレイボリューム ID(AVID)により、同じストレージに接続された複数のノード間で一貫したデバイスの命名が可能になります。ディスクアクセス名はアレイ自体によって定義された名前がベースになるため、変化することはありません。

メモ: DMP はサードパーティ製のドライバでの AVID をサポートしません。

DMP がデバイスの AVID にアクセスできない場合、LUN シリアル番号と呼ばれる、重複のない別の LUN 識別子を取得します。DMP は LUN シリアル番号(LSN)に基づいてデバイスをソートしてから、インデックス番号を割り当てます。すべてのホストが同じデバイス集合を参照するため、すべてのホストでソート済みのリストが同じになり、クラスタ間でデバイスのインデックスの一貫性が保たれます。この場合、DMP メタノード名は、*enclosureID_index* の形式です。

DMP は拡張性のあるフレームワークもサポートします。このフレームワークでは、カスタムの名前をキャビネットおよび LUN のシリアル番号と関連付けるデバイス命名ファイルを適用することによって、ホスト上でデバイス名を完全にカスタマイズすることが可能です。

CVM(Cluster Volume Manager)クラスタが対称的である場合、クラスタ内の各ノードは同じディスクセットにアクセスします。エンクロージャに基づく名前は一貫性のある名前付けシステムを提供するため、各ノードでデバイス名は同じになります。

`vxdisk list` などの Storage Foundation Cluster File System High Availability (SFCFSHA) ユーティリティは、AVID プロパティを含む DMP メタノード名を表示します。アレイ管理インターフェース (GUI または CLI) に表示される LUN に DMP メタノード名を関連付けるには、AVID を使います。

たとえば、エンクロージャが emc_clariion0 で、ASL によって提供されたアレイボリューム ID が 91 である EMC CX アレイでは、DMP メタノード名は emc_clariion0_91 です。次の出力例は DMP メタノード名を示しています。

```
$ vxdisk list
emc_clariion0_91  auto:cdsdisk  emc_clariion0_91   dg1  online shared
emc_clariion0_92  auto:cdsdisk  emc_clariion0_92   dg1  online shared
emc_clariion0_93  auto:cdsdisk  emc_clariion0_93   dg1  online shared
emc_clariion0_282 auto:cdsdisk  emc_clariion0_282  dg1  online shared
emc_clariion0_283 auto:cdsdisk  emc_clariion0_283  dg1  online shared
emc_clariion0_284 auto:cdsdisk  emc_clariion0_284  dg1  online shared

# vxddladm get namingscheme
NAMING_SCHEME       PERSISTENCE    LOWERCASE      USE_AVID
============================================================
Enclosure Based     Yes            Yes            Yes
```

ディスクのインストールとフォーマットについて

ディスクおよびシステムのハードウェア機能によって、ディスクをインストールする前にシステムを停止して電源を切る必要がある場合や、システムの稼動中にディスクをホットインサートできる場合があります。多くのオペレーティングシステムで、再起動時に新しいディスクの存在を検出します。システムの稼動中にディスクを挿入する場合は、オペレーティングシステム固有のコマンドを入力して、システムに通知する必要があることがあります。

使用前にディスクを低レベルまたは中間レベルでフォーマットする必要がある場合は、オペレーティングシステム固有のフォーマットコマンドを使ってフォーマットします。

メモ: SCSI ディスクは通常、フォーマット済みです。既存のフォーマットに障害がある場合にのみ、再フォーマットを行う必要があります。

---

p.389 の「VxVM へのディスクの追加」を参照してください。

ディスクの追加と削除

ここでは、ディスクの追加と削除について説明します。

VxVM へのディスクの追加

メモ: Linux 7.3.1 以降では、ルートディスクのカプセル化 (RDE) はサポートされません。

VxVM（Veritas Volume Manager）の制御下に置くフォーマット済みディスクには、新しいディスクまたは VxVM の制御外で以前に使われたディスクを使えます。

ディスクのセットは、コントローラ上のすべてのディスク、選択したディスク、またはこれらの組み合わせで構成できます。

状況によっては、すべてのディスクが同じ方法で処理されない場合があります。

たとえば、あるディスクは初期化され、別のディスクはディスク上の既存データを保持するためにカプセル化されることがあります。

一度に複数のディスクを初期化するとき、特定のディスクまたは特定のコントローラを無効にできます。

また、一度に複数のディスクをカプセル化するときにも、特定のディスクまたは特定のコントローラを無効にできます。

VxVM からデバイスを無効にするには、`vxdiskadm` のメインメニューから[VxVM が使うデバイスのパス、またはマルチパスの無効化(`Prevent multipathing/Suppress devices from VxVM's view`)]を選択します。

警告: 初期化では、ディスク上の既存データは保持されません。

使える有効なパーティションテーブルのないディスクは初期化できません。ディスク上に空のパーティションテーブルを作成するには、fdisk コマンドを次のように使います。

```
# fdisk /dev/sdX

   Command (m for help): o
   Command (m for help): w
```

/dev/sdX には、ディスクデバイスの名前(/dev/sdi など)を指定します。

警告: fdisk コマンドを実行すると、ディスク上のデータが破棄されることがあります。ディスク上のデータを保持する必要がある場合は、このコマンドを実行しないでください。

p.309 の「デバイスを VxVM で非表示にする」を参照してください。

VxVM 用にディスクを初期化するには、次の手順を実行します。

1 vxdiskadm メインメニューから[1 つ以上のディスクの追加または初期化(Add or initialize one or more disks)]を選択します。

2 次のプロンプトで、VxVM 制御に追加するディスクのディスクデバイス名を入力します(またはディスクの一覧を表示するために、「list」を入力します)。

```
Select disk devices to add:
[<pattern-list>,all,list,q,?]
```

pattern-list には、単一ディスクまたは一連のディスクを指定できます。*pattern-list* が複数の項目で構成されている場合は、スペースで区切って項目を切り離します。たとえば、次のように **4** つのディスクを表します。

```
sde sdf sdg sdh
```

プロンプトで、list と入力した場合は、vxdiskadm プログラムによって、システムで使えるディスクの一覧が表示されます。

```
DEVICE       DISK         GROUP        STATUS
sdb          mydg01       mydg         online
sdc          mydg02       mydg         online
sdd          mydg03       mydg         online
sde          -            -            online
sdf          mydg04       mydg         online
sdg          -            -            online invalid
```

[ステータス(STATUS)]列の[online invalid]という語句は、ディスクがまだ追加されていないか、VxVM の制御用に初期化されていないことを示します。ディスク名およびディスクグループとともに「online」と表示されているディスクは、すでに VxVM 制御下にあります。

プロンプトで、初期化するディスクのデバイス名またはパターンを入力し、リターンキーを押します。

3 操作を続行するには、次のプロンプトで y を入力します(またはリターンキーを押します)。

```
Here are the disks selected. Output format: [Device]
list of device names

Continue operation? [y,n,q,?] (default: y) y
```

4 次のプロンプトで、ディスクの追加先ディスクグループを指定するか、none を入力して後で使うためにディスクを予約します。

```
You can choose to add these disks to an existing disk group,
a new disk group, or you can leave these disks available for use

by future add or replacement operations. To create a new disk
group, select a disk group name that does not yet exist. To
leave the disks available for future use, specify a disk group
name of none.

Which disk group [<group>,none,list,q,?]
```

5 存在しないディスクグループ名を指定した場合は、vxdiskadm によって、新しいディスクグループを作成する確認を求めるプロンプトが表示されます。

```
There is no active disk group named disk group name.

Create a new group named disk group name? [y,n,q,?]
(default: y)y
```

ディスクグループでクロスプラットフォームデータシェアリング（CDS）機能をサポートするかどうか確認するように求められます。

```
Create the disk group as a CDS disk group? [y,n,q,?]
(default: y)
```

新しいディスクグループを異なるオペレーティングシステムプラットフォーム間で移動させる場合は、y を入力します。それ以外の場合は、n を入力します。

6 次のプロンプトで、リターンキーを押してデフォルトのディスク名を使うか、n を入力して任意のディスク名を定義します。

```
Use default disk names for the disks? [y,n,q,?] (default: y) n
```

7 ディスクをホットリロケーションのスペアにするかどうかを尋ねるプロンプトが表示されたら、n を入力します（または、リターンキーを押します）。

```
Add disks as spare disks for disk group name? [y,n,q,?]
(default: n) n
```

8 ディスクをホットリロケーションへの適用対象から除外するかどうかを尋ねるプロンプトが表示されたら、n を入力します（または、リターンキーを押します）。

```
Exclude disks from hot-relocation use? [y,n,q,?}
(default: n) n
```

9 次のプロンプトでサイトタグをディスクに追加するかどうかを選択します。

```
Add site tag to disks? [y,n,q,?] (default: n)
```

サイトタグは、常にディスクアレイまたはエンクロージャに適用され、リモートミラー機能を使わないかぎり必要ありません。

y を入力してサイトタグを追加する場合、ステップ 11 でサイト名の入力を求めるメッセージが表示されます。

10 操作を続行するには、次のプロンプトで y を入力します（またはリターンキーを押します）。

```
The selected disks will be added to the disk group
disk group name with default disk names.
list of device names
Continue with operation? [y,n,q,?] (default: y) y
```

11 ステップ 9 でディスクにサイトタグを追加することにした場合、ここで、各エンクロージャのディスクに適用するサイト名の入力を求めるメッセージが表示されます。

```
The following disk(s):
list of device names

belong to enclosure(s):
list of enclosure names

Enter site tag for disks on enclosure enclosure name
[<name>,q,?] site_name
```

12 次のプロンプトが表示された場合、これにはすでに VxVM 用に初期化されているディスクがすべて一覧表示されます。

```
The following disk devices appear to have been initialized
already.
The disks are currently available as replacement disks.
Output format: [Device]

list of device names

Use these devices? [Y,N,S(elect),q,?] (default: Y) Y
```

このプロンプトにより、これらすべてのディスクに対して「はい(yes)」または「いいえ(no)」を示したり(Y または N)、個別ベースで各ディスクを処理する方法を選択したり(S)できます。

ディスクすべてを再初期化する場合は、次のプロンプトで Y を入力します。

```
VxVM NOTICE V-5-2-366 The following disks you selected for use
appear to already have been initialized for the Volume
Manager. If you are certain the disks already have been
initialized for the Volume Manager, then you do not need to
reinitialize these disk devices.
Output format: [Device]

list of device names

Reinitialize these devices? [Y,N,S(elect),q,?] (default: Y) Y
```

13 vxdiskadm によって、1 つ以上のディスクがカプセル化の候補にあることが示されます。カプセル化では、アクティブなディスクを VxVM 制御に追加して、そのディスク上のデータを保持できます。ディスク上のデータを保持する場合は、y を入力します。保持する必要のあるデータがディスク上にないことが確実な場合は、n を入力し、カプセル化を回避します。

```
VxVM NOTICE V-5-2-355 The following disk device has a valid
partition table, but does not appear to have been initialized
for the Volume Manager. If there is data on the disk that
should NOT be destroyed you should encapsulate the existing
disk partitions as volumes instead of adding the disk as a new
disk.
Output format: [Device]

device name

Encapsulate this device? [y,n,q,?] (default: y)
```

14 ディスクのカプセル化を選択すると、vxdiskadm によってデバイス名が確認され、処理を続行する許可を求めるプロンプトが表示されます。y を入力して（または、リターンキーを押して）、カプセル化を続行します。

```
VxVM NOTICE V-5-2-311 The following disk device has been
selected for encapsulation.
Output format: [Device]

device name

Continue with encapsulation? [y,n,q,?] (default: y) y
vxdiskadm now displays an encapsulation status and informs you
that you must perform a shutdown and reboot as soon as
possible:
VxVM INFO V-5-2-333 The disk device device name will be
encapsulated and added to the disk group disk group name with the

disk name disk name.
```

これで、フォーマットするディスクとして、異なるオペレーティングシステム間で移動できる CDS ディスク、移動できない sliced ディスク、simple ディスクのいずれかを選択できます。

```
Enter the desired format [cdsdisk,sliced,simple,q,?]
(default: cdsdisk)
```

使用目的に適した形式を入力します。多くの場合、デフォルトの cdsdisk を選択します。

次のプロンプトで、vxdiskadm により、デフォルトの 65536 ブロック（32 MB）をプライベートリージョンサイズに使うかどうかを尋ねるメッセージが表示されます。リターンキーを押して、デフォルト値の使用を確認するか、別の値を入力します（指定できる最大値は、524288 ブロックです）。

```
Enter desired private region length [<privlen>,q,?]
(default: 65536)
```

cdsdisk 形式を指定した場合は、この形式に変換できなかった場合に実行するアクションの入力を求めるプロンプトが表示されます。

```
 Do you want to use sliced as the format should cdsdisk fail?
[y,n,q,?] (default: y)
```

y を入力した場合、CDS ディスクとしてカプセル化できないディスクは sliced ディスクとしてカプセル化されます。それ以外の場合は、カプセル化が失敗します。

この後、vxdiskadm によって、ディスクのカプセル化が続行されます。次のようなコマンドを実行して、できるだけ早い段階でシステムを再ブートしてください。

```
# shutdown -r now
```

/etc/fstab ファイルが更新され、カプセル化されたファイルシステムをマウントするためのボリュームデバイスが追加されます。場合によっては、バックアップスクリプト、データベース、手動作成したスワップデバイスなどの参照項目を更新する必要があります。元の /etc/fstab ファイルは、/etc/fstab.b4vxvm として保存されます。

15 ディスクのカプセル化を選択しなかった場合、vxdiskadm によって、カプセル化の代わりに、ディスクの初期化を行うかどうかを尋ねるメッセージが表示されます。y を入力し、これを確認します。

カプセル化せずに、初期化しますか?(Instead of encapsulating, initialize?) [y,n,q,?] (default: n) y 初期化し、VxVM の制御下に追加したディスクは、次の例のようなメッセージとともに vxdiskadm によって確認されます。さらに、表面分析の実行を求めるプロンプトが表示されることがあります。

```
VxVM INFO V-5-2-205 Initializing device device name.
```

16 これで、フォーマットするディスクとして、異なるオペレーティングシステム間で移動できる CDS ディスク、移動できない sliced ディスク、simple ディスクのいずれかを選択できます。

```
Enter the desired format [cdsdisk,sliced,simple,q,?]
(default: cdsdisk)
```

使用目的に適した形式を入力します。多くの場合、デフォルトの cdsdisk を選択します。

17 次のプロンプトで、vxdiskadm により、デフォルトの 65536 ブロック(32 MB)をプライベートリージョンサイズに使うかどうかを尋ねるメッセージが表示されます。リターンキーを押して、デフォルト値の使用を確認するか、別の値を入力します(指定できる最大値は、524288 ブロックです)。

```
Enter desired private region length [<privlen>,q,?]
(default: 65536)
```

この後、vxdiskadm によって、ディスクの追加が続行されます。

```
VxVM INFO V-5-2-88 Adding disk device device name to disk group
disk group name with disk name disk name.
.
.
.
```

18 デフォルトのディスク名を使わない場合は、vxdiskadm からディスク名を入力するよう求められます。

19 次のプロンプトで、続けてディスクを初期化する(y)か、vxdiskadm メインメニューに戻る(n)かを指定します。

```
Add or initialize other disks? [y,n,q,?] (default: n)
```

vxdisk コマンドか vxdiskadm ユーティリティを使用してディスクのための既定のレイアウトを変更できます。

vxdisk (1M) マニュアルページを参照してください。

vxdiskadm(1M)マニュアルページを参照してください。

ディスクの再初期化

VxVM 用に以前に初期化されたディスクを、新しいディスクと同様に VxVM(Veritas Volume Manager)の制御下に置くことで、再初期化できます。

p.389 の「VxVM へのディスクの追加」を参照してください。

警告: 再初期化では、ディスク上のデータは保持されません。ディスクの再初期化を行う場合は、保持する必要のあるデータが含まれていないことを確認してください。

追加するディスクを以前 Volume Manager の制御外で使っていた場合は、ディスクをカプセル化して、ディスク情報を保持できます。追加するディスクを以前 LVM の制御下で使っていた場合は、そのデータを変換プロセスによって VxVM ディスク上で保持できます。

移行ボリュームについて詳しくは、『Veritas InfoScale ソリューションガイド』を参照してください。

vxdiskadd を使った VxVM の制御下へのディスクの配置

vxdiskadd コマンドを使うと、VxVM(Veritas Volume Manager)の制御下にディスクを追加できます。

vxdiskadd コマンドを使ってディスクを VxVM の制御下に配置するには

◆ 次のようにコマンドを入力します。

```
# vxdiskadd disk
```

たとえば、ディスク sdb を初期化するには、次の手順を実行します。

```
# vxdiskadd sdb
```

vxdiskaddコマンドは、ディスクを調査して初期化されているかどうかを確認します。また、VxVM に追加されたディスクやその他の状態もチェックします。

また、vxdiskaddコマンドは、ディスクをカプセル化できるかどうかもチェックします。

p.1063 の「ディスクのカプセル化」を参照してください。

未初期化ディスクを追加すると、vxdiskadd コマンドにより、警告およびエラーメッセージがコンソールに表示されます。これらのメッセージは無視してください。ディスクの初期化が完了すると、これらのメッセージは表示されません。初期化が完了すると、vxdiskadd コマンドによって、成功メッセージが表示されます。

vxdiskaddのディスク追加用の対話的ダイアログは、vxdiskadmのものと類似しています。

p.389 の「VxVM へのディスクの追加」を参照してください。

ディスクの削除

ここでは、VxVM（Veritas Volume Manager）ディスクを削除する方法について説明します。

ディスクグループの最後のディスクを削除するには、そのグループを無効にする必要があります。

p.1015 の「ディスクグループの無効化」を参照してください。

ディスクグループを無効にする代わりに、ディスクグループを破棄することができます。

p.1015 の「ディスクグループの破棄」を参照してください。

ディスクに障害が発生しているまたは発生した場合は、システムからディスクを削除し、別のシステムに移動することができます。

ディスクを削除するには、次の手順を実行します。

1 削除するディスク上に設定されているボリュームを対象とする、アプリケーションの動作をすべて停止します。ボリューム上に設定されているファイルシステムのマウントを解除し、データベースを停止します。

2 ボリュームを停止するには、次のコマンドを使います。

```
# vxvol [-g diskgroup] stop vol1vol2 ...
```

3 ボリュームを他のディスクに移動するか、ボリュームのバックアップを作成します。ボリュームを移動するには、vxdiskadmを使って複数のディスク上のボリュームをミラー化し、続いてボリュームの元のコピーを削除します。必要のなくなったボリュームは、移動する代わりに削除することができます。

4 ディスク上のデータが他のディスクに移動されているか不要であることを確認します。

5 vxdiskadmメインメニューで[ディスクの削除(Remove a disk for replacement)]を選択します。

6 次のプロンプトで、削除するディスクのディスク名を入力します。

```
Enter disk name [<disk>,list,q,?] mydg01
```

7 ディスク上にボリュームが存在する場合は、ディスクから退避させるかどうかを尋ねるメッセージが VxVM より表示されます。ボリュームを保持する場合は、y を入力します。それ以外の場合は、n を入力します。

8 次の確認プロンプトで、Return キーを押して操作を続行します。

```
VxVM NOTICE V-5-2-284 Requested operation is to remove disk
mydg01 from group mydg.


Continue with operation? [y,n,q,?] (default: y)
```

vxdiskadm ユーティリティは、ディスクグループからディスクを削除し、次の成功メッセージを表示します。

```
VxVM INFO V-5-2-268 Removal of disk mydg01 is complete.
```

これで、ディスクを削除するか、交換用ディスクとしてシステム上に残しておくことができます。

9 次のプロンプトで、他のディスクを削除する(y)か、vxdiskadmメインメニューに戻る(n)かを指定します。

```
Remove another disk? [y,n,q,?] (default: n)
```

サブディスクのあるディスクの削除

サブディスクが定義されている VxVM (Veritas Volume Manager)ディスクを削除することができます。たとえば、1 つのディスク上にすべてのボリュームを統合することができます。vxdiskadm プログラムを使って、ディスクの削除を行う場合は、そのディスク外へボリュームを移動することも選択できます。

移動できないサブディスクもあります。次の理由のいずれかにより、サブディスクを移動できないことがあります。

- サブディスクの属するディスクグループで残っているディスク上に、十分な領域がない場合。
- ボリューム上の既存のプレックスやストライプサブディスクから異なるディスク上に、プレックスまたはストライプサブディスクを割り当てることができない場合。

vxdiskadm プログラムを使ってサブディスクを移動できない場合は、ディスク削除の操作を進める前に、いくつかのディスクからプレックスを削除して空き領域を増やしてください。

p.1052 の「ボリュームの削除」を参照してください。

サブディスクのあるディスクを削除するには、次の手順を実行します。

1 vxdiskadm プログラムを実行して、メインメニューで[ディスクの削除(Remove a disk)]を選択します。

ディスクを使っているサブディスクがある場合は、次のメッセージが表示されます。

```
VxVM ERROR V-5-2-369 The following volumes currently use part of

disk mydg02:

home usrvol

Volumes must be moved from mydg02 before it can be removed.

Move volumes to other disks? [y,n,q,?] (default: n)
```

2 y を選択すると、可能であれば、すべてのサブディスクがディスクから移動されます。

サブディスクのないディスクの削除

サブディスクを含んでいない VxVM (Veritas Volume Manager) ディスクを削除することができます。

サブディスクを含んでいないディスクをディスクグループから削除するには、次の手順を実行します。

- vxdiskadm プログラムを実行してメインメニューで[ディスクの削除(Remove a disk)]を選択し、この例に示されているようにプロンプトに応答して mydg02 を削除します。

```
Enter disk name [<disk>,list,q,?] mydg02

VxVM NOTICE V-5-2-284 Requested operation is to remove disk
mydg02 from group mydg.

Continue with operation? [y,n,q,?] (default: y) y
VxVM INFO V-5-2-268 Removal of disk mydg02 is complete.
Clobber disk headers? [y,n,q,?] (default: n) y
```

y を入力すると、ディスクは VxVM の制御下から完全に削除されます。ディスクを VxVM の制御下から完全に削除しない場合は、n を入力してください。

ディスク名の変更

VxVM ディスク名を指定しないと、VxVM 制御にディスクを追加するときに、VxVM(Veritas Volume Manager)によってデフォルト名が設定されます。VxVM では、VxVM ディスク名はディスクまたはディスクタイプの位置を識別するために使われます。

ディスク名を変更するには

◆ 次のコマンドを入力します。

```
# vxedit [-g diskgroup] rename old_diskname new_diskname
```

デフォルトでは VxVM により、サブディスクオブジェクトの配置された VxVM ディスクの名前がサブディスクオブジェクトに設定されます。VxVM ディスクの名前を変更しても、そのディスク上のサブディスクの名前は自動的に変更されません。

たとえば、vxdisk list の出力で次のように表示される場合、ディスク mydg03 の名前を mydg02 に変更できます。

```
# vxdisk list

DEVICE       TYPE            DISK         GROUP        STATUS
sdb          auto:sliced     mydg01       mydg         online
sdc          auto:sliced     mydg03       mydg         online
sdd          auto:sliced     -            -            online
```

次のコマンドを使って、ディスクの名前を変更します。

```
# vxedit -g mydg rename mydg03 mydg02
```

名前が変更されていることを確認するには、再度 vxdisk list コマンドを使います。

```
# vxdisk list

DEVICE       TYPE            DISK         GROUP        STATUS
sdb          auto:sliced     mydg01       mydg         online
sdc          auto:sliced     mydg02       mydg         online
sdd          auto:sliced     -            -            online
```

14

イベント監視

この章では以下の項目について説明しています。

- Dynamic Multi-Pathing（DMP）のイベントソースデーモン（vxesd）について
- ファブリック監視と予防的なエラー検出
- Dynamic Multi-Pathing（DMP）の iSCSI および SAN ファイバーチャネルトポロジーの検出
- DMP イベントログ
- Dynamic Multi-Pathing（DMP）のイベントソースデーモンの起動と停止

Dynamic Multi-Pathing（DMP）のイベントソースデーモン（vxesd）について

イベントソースデーモン（vxesd）は Dynamic Multi-Pathing（DMP）コンポーネントプロセスで、適切な処理を実行するために使われるデバイス関連イベントの通知を受信します。vxesd のメリットには次のものがあります。

- SAN ファブリックイベントの監視と予防的なエラー検出（SAN イベント）
  p.405 の「ファブリック監視と予防的なエラー検出」を参照してください。
- トラブルシューティングのために DMP イベントをログに記録（DMP イベント）
  p.406 の「DMP イベントログ」を参照してください。
- 自動デバイス検出（OS イベント）
- SAN コンポーネントと HBA アレイポートの接続性の検出（ファイバーチャネルと iSCSI）
  p.406 の「Dynamic Multi-Pathing（DMP）の iSCSI および SAN ファイバーチャネルトポロジーの検出」を参照してください。

p.407 の「Dynamic Multi-Pathing（DMP）のイベントソースデーモンの起動と停止」を参照してください。

ファブリック監視と予防的なエラー検出

DMP は、パスのエラー検出で予防的な対応を取ります。

DMP イベントソースデーモン vxesd は、SNIA (Storage Networking Industry Association) HBA API ライブラリを使って SAN ファブリックイベントを HBA から受信します。

DMP は、アクティブな I/O がない場合でも、この情報に基づいて問題が疑われるデバイスを SAN イベントからチェックします。DMP が疑わしいデバイスを検証する間、新しい I/O は正常なパスに転送されます。

起動中、vxesd は (SNIA ライブラリ経由で) HBA をクエリーして SAN トポロジーを取得します。vxesd デーモンは、オペレーティングシステムが認識する個々のデバイスパスに対応する PWWN (Port World Wide Name) を確認します。vxesd デーモンがトポロジーを取得した後に、vxesd は SAN イベントの通知を受けるために HBA に登録します。LUN が SAN から切断されると、HBA は vxesd に SAN イベントを通知し、影響を受ける PWWN を指示します。vxesd デーモンはこのイベント情報を以前のトポロジー情報と関連付けて、どのデバイスパスの集合が影響を受けたかを特定します。

vxesd デーモンは、影響を受けたパスの集合を vxconfigd デーモン (DDL) に送信し、デバイスパスを疑わしい状態としてマーク付けできるようにします。

パスが疑わしい状態としてマーク付けされると、そのパスがデバイスへの最終パスである場合を除き、DMP は新しい I/O をそのパスに送信しません。バックグラウンドでは、DMP リストアタスクが SCSI 照会プローブを使って次の定期サイクルでパスのアクセシビリティを確認します。SCSI 照会に失敗した場合、DMP は影響を受けた LUN へのパスを無効にし、そのパスはイベントログにも記録されます。

後から LUN が再接続された場合、HBA は vxesd に SAN イベントを通知します。DMP リストアタスクが次のテストサイクルを実行する際に、無効にされたパスが SCSI プローブによって確認され、成功した場合はパスが再び有効化されます。

メモ: vxesd が HBA LINK UP イベントを受信すると、DMP リストアタスクが再開され、次の定期サイクルを待たずに SCSI プローブがすぐに実行されます。DMP リストアタスクが再開されると、このタスクによって新しい定期サイクルが開始されます。無効化されたパスが最初の SCSI プローブの時点でアクセス不能な場合、次回のサイクル（デフォルトの周期は 300 秒）でそのパスが再度テストされます。

ファブリック監視機能はデフォルトで有効です。dmp_monitor_fabric チューニングパラメータの値は、再起動の前後で変化しません。

dmp_monitor_fabric チューニングパラメータの現在値を表示するには、次のコマンドを使います。

```
# vxdmpadm gettune dmp_monitor_fabric
```

ファブリック監視機能を無効にするには、次のコマンドを使います。

```
# vxdmpadm settune dmp_monitor_fabric=off
```

ファブリック監視機能を有効にするには、次のコマンドを使います。

```
# vxdmpadm settune dmp_monitor_fabric=on
```

Dynamic Multi-Pathing（DMP）の iSCSI および SAN ファイバーチャネルトポロジーの検出

vxesd は、ホストが認識する iSCSI および FC（Fibre Channel の略でファイバーチャネルデバイスの意味）のトポロジーを構築します。vxesd デーモンは、SNIA Fibre Channel HBA API を使って SAN トポロジーを取得します。IMA が利用できない場合、iSCSI SAN トポロジーを取得するために iSCSI 管理 CLI が使われます。

ファイバーチャネルおよび iSCSI デバイスの階層リストを表示するには、次のコマンドを使います。

```
# vxddladm list
```

vxddladm(1M)のマニュアルページを参照してください。

DMP イベントログ

p.404 の「Dynamic Multi-Pathing（DMP）のイベントソースデーモン（vxesd）について」を参照してください。

イベントソースデーモン (vxesd) は DMP (Dynamic Multi-Pathing) コンポーネントプロセスで、適切な処理を実行するために使うデバイス関連イベントの通知を受信します。

DMP はメジャーイベントを vxesd に通知し、vxesd はイベントをログファイルに記録します。記録されるイベントには次のものがあります。

- パスまたは dmpnode を有効としてマーク付け
- パスまたは dmpnode を無効としてマーク付け
- パスの調整
- I/O エラー解析
- HBA および SAN イベント

DMP イベントについてシステムログまたはコンソールログに表示される詳細レベルを変更できます。チューニングパラメータ dmp_log_level を使用します。有効な値は 1 から 9 までです。デフォルトレベルは 1 です。

```
# vxdmpadm settune dmp_log_level=x
```

dmp_log_level の現在の値は次のものを使って表示できます。

```
# vxdmpadm gettune dmp_log_level
```

各種のログレベルについて詳しくは、vxdmpadm (1M) のマニュアルページを参照してください。

Dynamic Multi-Pathing（DMP）のイベントソースデーモンの起動と停止

デフォルトでは、DMP（Dynamic Multi-Pathing）はブート時にイベントソースデーモン vxesd を起動します。

vxesd デーモンを停止するには、次のように vxddladm ユーティリティを使います。

```
# vxddladm stop eventsource
```

vxesd デーモンを起動するには、次のように vxddladm ユーティリティを使います。

```
# vxddladm start eventsource [logfile=logfilename]
```

vxesd デーモンの状態を表示するには、vxddladm ユーティリティを使います。

```
# vxddladm status eventsource
```

4

Storage Foundation Cluster File System High Availability の管理

15

Storage Foundation Cluster File System High Availability とそのコンポーネントの管理

この章では以下の項目について説明しています。

Storage Foundation Cluster File System High Availability の管理について

SFCFSHA（Storage Foundation Cluster File System High Availability）は、複数のホストが同じファイルを同時にマウントし、そのファイルに対して同時にファイル操作を実行できるようにする共有ファイルシステムです。クラスタ設定で SFCFSHA を操作するに

は、Storage Foundation Cluster File System High Availability に含まれる一連の統合 Veritas 製品が必要です。

SFCFSHA でクラスタを設定するには Cluster Server（VCS）が必要です。VCS は SFCFSHA に不可欠な 2 つの主要コンポーネントを提供します。LLT RPM は、ノード間通信を可能にし、ネットワーク通信を監視します。GAB RPM は、クラスタの状態、設定、メンバーシップサービスを提供し、システム間のハートビートリンクを監視してシステムがアクティブであることを確実にします。SFCFSHA をインストールする際には、アプリケーションのフェールオーバーをサポートする VCS 提供のその他の RPM が複数あります。

『Storage Foundation Cluster File System High Availability インストールガイド』を参照してください。

VCS について詳しくは、『Cluster Server』のマニュアルを参照してください。

CFS の管理

ここでは、CFS（Cluster File System）の主な管理作業について説明します。

CFS の管理中に問題が発生した場合について詳しくは、『Veritas InfoScale トラブルシューティングガイド』を参照してください。

VCS 設定への新しい CFS システムの追加

エディタを使って、VCS（ Cluster Server）の `main.cf` ファイルに CFS（Cluster File System）ファイルシステムを追加するには、次のコマンドを実行します。

次に例を示します。

```
# cfsmntadm add oradatadg oradatavol ¥
/oradata1 all=suid,rw

Mount Point is being added...
  /oradata1 added to the cluster-configuration
```

cfsmount と cfsumount を使った CFS ファイルシステムのマウントとマウント解除

CFS ファイルシステムをマウントするには

- CFS ファイルシステムをマウントします。

```
# cfsmntadm add sdg vol1 /oradata1 all=
# cfsmount /oradata1
Mounting...
[/dev/vx/dsk/sdg/vol1]
mounted successfully at /oradata1 on sys1
[/dev/vx/dsk/sdg/vol1]
mounted successfully at /oradata1 on sys2
```

CFS ファイルシステムをマウント解除するには

- CFS ファイルシステムをマウント解除します。

```
# cfsumount /oradata1
Unmounting...
/oradata1 got successfully unmounted from sys1
/oradata1 got successfully unmounted from sys2
```

VCS 設定からの CFS ファイルシステムの削除

VCS 設定から CFS ファイルシステムを削除するには

- エディタを使わずに、VCS（Cluster Server）の main.cf ファイルから CFS（Cluster File System）を削除するには、次のコマンドを実行します。

```
# cfsmntadm delete /oradata1
Mount Point is being removed...
/oradata1 deleted successfully from cluster-configuration
```

CFS ファイルシステムのサイズ変更

CFS（Cluster File System）ファイルシステムがいっぱいであることを示すメッセージがコンソールに表示されている場合、ファイルシステムのサイズを変更することができます。vxresize コマンドを使うと、CFS ファイルシステムをサイズ変更できます。ファイルシステムと下位のボリュームが拡張されます。

さまざまなオプションについては vxresize（1M）マニュアルページを参照してください。

次のコマンドを使うと、Oracle データ CFS を含むファイルシステム（この Oracle データボリュームは CFS マウント済みです）をサイズ変更できます。

```
# vxresize -g oradatadg oradatavol +2G
```

ここで *oradatadg* は CVM ディスクグループ

ここで *oradatavol* はボリューム

ここで *+2G* はボリュームのサイズを 2 GB 増やすことを意味します。

次のコマンドを使うと、Oracle データ CFS を含むファイルシステム（この Oracle データボリュームは CFS マウント済みです）を縮小できます。

```
# vxresize -g oradatadg oradatavol -2G
```

-2G はボリュームのサイズを 2 GB 減らすことを意味します。

CFS ファイルシステムノードと各ノードのマウントポイントの状態の確認

`cfscluster status` コマンドを実行し、ノードと各ノード上にあるマウントポイントの状態を表示します。

```
# cfscluster status

Node             :  sys2
  Cluster Manager  :  not-running
  CVM state        :  not-running
  MOUNT POINT    SHARED VOLUME  DISK GROUP        STATUS

  /ocrvote       ocrvotevol     sys1_ocr        NOT MOUNTED
  /oracle        ora_vol        sys1_ora        NOT MOUNTED
  /crshome       ora_crs_vol    sys1_crs        NOT MOUNTED
  /oradata1      ora_data1_vol  sys1_data1      NOT MOUNTED
  /arch          archivol       sys1_data1      NOT MOUNTED

  Node             :  sys1
  Cluster Manager  :  running
  CVM state        :  running
  MOUNT POINT    SHARED VOLUME  DISK GROUP        STATUS

  /ocrvote       ocrvotevol     sys1_ocr        MOUNTED
  /oracle        ora_vol        sys1_ora        MOUNTED
  /crshome       ora_crs_vol    sys1_crs        MOUNTED
  /oradata1      ora_data1_vol  sys1_data1      MOUNTED
  /arch          archivol       sys1_data1      MOUNTED
```

CFS ポートの状態の確認

CFS（Cluster File System）は、ノード間の通信にポート f を使います。ポート f は、CFS メンバーシップに使用する GAB ポートです。

CFS ポートの状態は、次のように確認できます。

```
# gabconfig -a | grep "Port f"
```

CFS エージェントおよび AMF サポート

CFS（Cluster File System）エージェント（CFSMount と CFSfsckd）は、AMF（Asynchronous Monitoring Framework）対応です。このリリースでは、CFS エージェントは V51 フレームワークを使います。

CFS エージェントログファイル

/var/VRTSvcs/log ディレクトリにある CFS（Cluster File System）エージェントログファイルを使って、CFS の問題をデバッグすることができます。

```
# cd /var/VRTSvcs/log
# ls
CFSMount_A.log
CFSfsckd_A.log
engine_A.log
CmdServer-log_A.log
healthview_A.log
uuidconfig.log
HostMonitor_A.log
hastart.log
tmp
vxfen
```

エージェントフレームワーク情報は engine_A.log ファイルに、エージェントエントリポイント情報は CFSMount_A.log ファイルと CFSfsckd_A.log ファイルにあります。

CFS コマンド

表 15-1 で CFS（Cluster File System）のコマンドを説明します。

表 15-1 CFS コマンド

コマンド	説明
cfscluster	クラスタ設定コマンド 詳しくは、cfscluster(1M)マニュアルページを参照してください。
cfsmntadm	クラスタマウントされたファイルシステム上でポリシーを追加、削除、変更、設定する 詳しくは cfsmntadm(1M)のマニュアルページを参照してください。
cfsdgadm	共有ディスクグループをクラスタ設定に追加したり、クラスタ設定から削除したりする 詳しくは、cfsdgadm(1M)マニュアルページを参照してください。
cfsmount	共有ボリュームにクラスタファイルシステムをマウントする 詳しくは cfsmount(1M)のマニュアルページを参照してください。
cfsumount	共有ボリュームのクラスタファイルシステムをマウント解除する 詳しくは cfsumount(1M)のマニュアルページを参照してください。
cfsshare	CNFS(クラスタ化された NFS)および CIFS(Common Internet File System)設定コマンド 詳しくは、cfsshare(1M)マニュアルページを参照してください。

mount、fsclustadm、fsadm コマンドについて

fsadm コマンドは、VxFS ファイルシステム、Storage Checkpoint、個々のファイルとディレクトリでオンライン管理機能を実行します。fsadm コマンドは、ファイルシステムのサイズ変更、エクステントの再構成、ディレクトリの再構成、largefiles フラグのクエリーまたは変更、Thin Storage Reclamation をサポートします fsadm コマンドは、読み書きアクセスのためにマウントされたファイルシステムで動作します。ただし、-o オプションは、マウント解除されたクリーンなファイルシステムを含んでいる特殊デバイスでも動作できます。特権ユーザーのみが、マウントされたファイルシステムで largefiles フラグを変更したり、ファイルシステムをサイズ調整または再構成したりすることができます。ファイルシステムごとに呼び出すことができる fsadm のインスタンス数は、一度に 1 つのみです。

mount_point が Storage Checkpoint の場合、fsadm は、ファイルシステムのすべての Storage Checkpoint を含むファイルシステム全体で指定の操作を実行します。

fsadm_vxfs(1M)のマニュアルページを参照してください。

mount コマンドと fsclustadm コマンドは、クラスタファイルシステムを設定する上でも重要です。

mount コマンドについて

mount コマンドで -o cluster オプションを使うと、共有ファイルシステムにアクセスできます。

mount_vxfs（1M）のマニュアルページを参照してください。

fsclustadm コマンドについて

fsclustadm コマンドは、CFS の各種属性を報告します。fsclustadm を使ってできる作業には、クラスタ内のプライマリノードの表示と設定、ノード ID とホスト名の相互変換、指定されたファイルシステムマウントポイントのクラスタマウントを現在持つ全ノードの一覧表示、マウントがローカルマウントかクラスタマウントかの判別などがあります。fsclustadm コマンドは、ファイルシステムがマウントされているクラスタ内の任意のノードから実行でき、指定されたマウントポイントに対するプライマリノードの場所を制御できます。

fsclustadm（1M）のマニュアルページを参照してください。

fsadm コマンドについて

fsadm コマンドは、VxFS ファイルシステム、Storage Checkpoint、個々のファイルとディレクトリでオンライン管理機能を実行します。 fsadm コマンドは、ファイルシステムのサイズ変更、エクステントの再構成、ディレクトリの再構成、largefiles フラグのクエリーまたは変更、シンストレージの再生、空き領域の断片化をサポートします。

fsadm コマンドはプライマリノードまたはセカンダリノードから実行することができます。

fsadm_vxfs（1M）のマニュアルページを参照してください。

共有環境での UNIX コマンドの安全な実行

共有環境では、データの破損を避けるため、RAW デバイスへ書き込み可能な UNIX コマンドを慎重に使う必要があります。 共有 VxVM ボリュームの場合、SFCFSHA には、fsck や mkfs などの VxFS コマンドにより、クラスタ内の別のノードからマウントされたファイルシステムを誤って破損することを防止するために、クラスタ内のボリュームを占有して他のノードから保護する機能が用意されています。ただし、dd などのコマンドは他のノードから保護せず実行されるため、他のノードからマウントされたファイルシステムを破損する可能性があります。ファイルシステムでこれらの VxFS 以外のコマンドを実行する前に、ファイルシステムがクラスタにマウントされていないことを確認してください。mount コマンドを実行すると、ファイルシステムのマウントが共有か、ローカルかを確認できます。

すべてのノードでのシステムクロックの同期

NTP（Network Time Protocol）など何らかの外部コンポーネントを使って、すべてのノードのシステムクロックが同期されている必要があります。ノードを同期化しないと、ファイル

作成時のタイムスタンプ（ctime）と同じファイル修正時のタイムスタンプ（mtime）が実際の操作順序と一致しない場合があります。

CFS ファイルシステムの拡張

CVM（Cluster Volume Manager）用のマスターノードと、CFS（Cluster File System）用のプライマリノードがあります。ファイルシステムを拡張するときは、CVM マスターからボリュームとスレーブを拡張し、続いて任意の CFS ノードからファイルシステムを拡張します。CVM マスターと CFS ノードは異なるノードであってもかまいません。

クラスタ内のプライマリファイルシステムを確認するには

- クラスタ内のプライマリファイルシステムを確認するには、次のコマンドを入力します。

```
# fsclustadm -v showprimary mount_point
```

現在のノードが CVM マスターノードかどうかを確認するには

- 現在のノードが CVM マスターノードかどうかを確認するには、次のコマンドを入力します。

```
# vxdctl -c mode
```

ファイルシステムのサイズを実際に大きくするには

1. CVM マスターノードで、次のコマンドを入力します。

```
# vxassist -g shared_disk_group growto volume_namenewlength
```

2. SFCFSHA ノードで、次のコマンドを入力します。

```
# fsadm -t vxfs -b newsize -r device_namemount_point
```

/etc/fstab ファイルについて

fstab により開始されるマウントはクラスタ設定の開始前に行われるため、/etc/fstab ファイルではどのクラスタファイルシステムについても起動時のマウントを指定しないでください。クラスタマウントの場合は、再ブートの後に、VCS（Cluster Server）設定ファイルを使って有効にするファイルシステムを特定してください。

CFS プライマリノードに障害が発生した場合

CFS（Cluster File System）プライマリノードが稼動するサーバーに障害が発生した場合は、残りのクラスタノードから新しいプライマリが選択されます。新しいプライマリノードはファイルシステムのインテントログを読み取り、障害発生時に処理中であったメタデータの更新を完了します。この処理中は、他のノードからのアプリケーション I/O がブロックさ

れて、遅延が発生することがあります。ファイルシステムの整合性が回復すると、アプリケーションの処理が再開されます。

SFCFS を使うセカンダリノードのノードでは、ファイルシステムのメタデータを直接には更新しないため、セカンダリノードの障害が発生してもどのメタデータも修復する必要はありません。そのため、セカンダリノードの障害からの CFS のリカバリは、プライマリノードの障害からのリカバリより高速です。

詳しくは、`fsclustadm`(1M)マニュアルページを参照してください。

クラスタ内の作業負荷の分散について

クラスタ内の作業負荷の分散により、パフォーマンスおよびフェールオーバーの機能が向上します。

たとえば、8 つのファイルシステムと 4 つのノードがある場合、ノードごとに 2 つのファイルシステムをプライマリとして指定すると効果的です。プライマリシップは、どのノードが最初にファイルシステムをマウントするかによって決まります。また、`fsclustadm` を使って、CFS プライマリノードを指定することもできます。`fsclustadm setprimary` コマンドでも、現在のプライマリで障害が発生した場合にプライマリシップを引き受ける順序を定義できます。セットアップ後、クラスタ内の 1 つ以上のノードでファイルシステムがマウントされている間、このポリシーは有効です。

SFCFSHA での Storage Checkpoint について

クラスタファイルシステムでの Storage Checkpoint 作成のしくみはローカルマウントファイルシステムの場合と同じです。

Storage Checkpoint を作成および保持する方法については、より詳細な情報を入手できます。

p.737 の「Storage Checkpoint について」を参照してください。

SFCFSHA のスナップショットについて

スナップショットは、VxFS ファイルシステムの PIT(ポイントインタイム)イメージを提供します。スナップショットに読み取り専用モードでマウントされたファイルシステムとしてアクセスすると、ファイルシステムのオンラインバックアップを効率よく実行できます。スナップショットは、スナップファイルシステムでデータブロックが上書きされるとデータブロックを逐次コピーするコピーオンライトセマンティクスを実装しています。

p.678 の「スナップショットファイルシステムについて」を参照してください。

クラスタファイルシステムのスナップショットは、クラスタ内の任意のノードから発生する I/O のコピーオンライト機構を拡張します。

クラスタスナップショットの特性について

クラスタスナップショットには以下のような特性があります。

- マウントしたクラスタファイルシステムのスナップショットは、クラスタ内の任意のノードにマウントできます。マウントしたファイルシステムは、プライマリ、セカンダリ、またはセカンダリ専用にできます。ファイルシステムの安定したイメージを任意のノードからの書き込み用に提供できます。
- クラスタファイルシステムの複数のスナップショットを、クラスタ内の同じノードまたは異なるノードにマウントできます。
- スナップショットは、スナップショットをマウントしているノードでのみアクセスできます。スナップショットデバイスを 2 つの異なるノードに同時にマウントすることはできません。
- スナップショットをマウントするためのデバイスとしてローカルディスクまたは共有ボリュームを使えます。共有ボリュームは、スナップショットマウントによって排他的に使われ、スナップショットがそのデバイスでアクティブである間は、クラスタ内の他のノードから使えません。
- スナップショットをマウントしているノードでは、スナップショットがマウントされている間はスナップショットを取られたオリジナルのファイルシステムのマウントを解除できません。
- クラスタファイルシステムのスナップショットは、スナップショットのマウントが解除された場合、またはスナップショットをマウントしているノードに障害が発生した場合は消失します。ただし、他のどのノードがクラスタから離れたり、クラスタに参加したりしても、スナップショットはその影響を受けません。
- 読み取り専用でマウントされたファイルシステムのスナップショットは作成できません。クラスタファイルシステムのスナップショットは、クラスタファイルシステムが `crw` オプションでマウントされている場合のみ、スナップショットを取得できます。

パフォーマンスの注意事項

バックアップ用にスナップショットファイルシステムをマウントすると、コピーオンライトを実行し、スナップショットからデータブロックを読み取るためにリソースが使われるため、システム上の負荷が増大します。このような場合、クラスタのスナップショットを使ってオフホストバックアップを行うことができます。オフホストバックアップは、プライマリサーバーでのバックアップアプリケーションによる負荷を軽減します。スナップショット全体のオーバーヘッドに比べて、リモートスナップショットからのオーバーヘッドは小さくて済みます。したがって、比較的負荷の少ないノードからスナップショットをマウントして、バックアップアプリケーションを実行すると、クラスタ全体のパフォーマンスに効果があります。

クラスタでのスナップショットの作成

ここでは、クラスタファイルシステム管理インターフェースのコマンドを使って 2 ノードクラスタでスナップショットを作成し、マウントする方法を説明します。

クラスタファイルシステムでスナップショットを作成するには

1 共有 VxVM ボリューム上で VxFS ファイルシステムを作成するために、次のコマンドを入力します。

```
# mkfs -t vxfs /dev/vx/rdsk/cfsdg/vol1

version 16 layout
104857600 sectors, 52428800 blocks of size 1024, log size
16384 blocks unlimited inodes, largefiles not supported
52428800 data blocks, 52399152 free data blocks 1600
allocation units of 32768 blocks, 32768 data blocks
```

2 すべてのノードでファイルシステムをマウントするために、次のコマンドを入力します。

```
# cfsmntadm add cfsdg vol1 /mnt1 all=cluster
# cfsmount /mnt1
```

`cfsmntadm` コマンドで Cluster Manager 構成にエントリを追加してから、`cfsmount` コマンドですべてのノードにファイルシステムをマウントします。

3 以前に作成したボリューム (この例では `snapvol`) のスナップショットを Cluster Manager 構成に追加します。次に例を示します。

```
# cfsmntadm add snapshot cfsdg snapvol /mnt1 /mnt1snap sys1=ro
```

クラスタファイルシステムのスナップショットには、そのスナップショットが作成されたノード上でのみアクセスできます。スナップショットファイルシステム自体はクラスタマウントできません。

4 スナップショットファイルシステムを sys1 上で作成してローカルにマウントします。次のコマンドを入力します。

```
# cfsmount /mnt1snap
```

5 スナップされたファイルシステムは、そのスナップショットがすべてマウント解除されるまではマウント解除できません。スナップされたクラスタファイルシステムをマウント解除する前に、スナップショットをマウント解除して破棄します。次のコマンドを入力します。

```
# cfsumount /mnt1snap
```

VCS の管理

この項では、次の VCS 管理タスクの手順について説明します。

- 「指定した Pfile で Oracle を開始するように VCS を設定する」
- 「VCS 設定の確認」
- 「VCS の起動と停止」

VCS の管理中に問題が発生した場合について詳しくは、トラブルシューティングの項を参照してください。

指定した Pfile で Oracle を開始するように VCS を設定する

Oracle を指定した Pfile で開始するように VCS（Cluster Server）を設定したい場合、次のように Oracle グループの `main.cf` ファイルを変更します。

```
Oracle ora1 (
                Sid @sys1 = vrts1
                Sid @sys2 = vrts2
                Owner = oracle
                Home = "/app/oracle/orahome"
                StartUpOpt = SRVCTLSTART
                ShutDownOpt = SRVCTLSTOP
                pfile="/app/oracle/orahome/dbs/initprod1.ora"
                )
```

VCS 設定の確認

VCS 設定を確認するには

```
# cd /etc/VRTSvcs/conf/config
# hacf -verify .
```

VCS の起動と停止

この節は VCS を起動方法と停止方法を説明します。

VCS を起動するには

◆ 各ノードで、次のように入力して VCS を起動します。

```
# hastart
```

VCS を停止するには

◆ 各ノードで、次のように入力して VCS を停止します。

```
# hastop -local
```

また hastop -all コマンドを使用して、クラスタ内のすべてのノードの VCS クラスタを同時に停止させることもできます。ただし VCS を再起動する前に、ポート「h」が閉じたことを確認してください。

LLT の宛先ベースの負荷分散の設定

LLT（Low Latency Transport）の宛先ベースの負荷分散はデフォルトではオフになります。ベリタスは、クラスタセットアップに 2 つ以上のノードとよりアクティブな LLT ポートがあるときは、宛先ベースの負荷分散を推奨します。

LLT の宛先ベースの負荷分散を設定するには

◆ 次のコマンドを実行して、宛先ベースの負荷分散を設定します。

```
lltconfig -F linkburst:0
```

CVM の管理

ここでは、CVM （Cluster Volume Manager）を管理するタスクについて説明します。

すべての CVM 共有ディスクの一覧表示

次のコマンドを使って、すべての Cluster Volume Manager 共有ディスクを一覧表示します。

```
# vxdisk -o alldgs list |grep shared
```

クラスタ内で利用可能なすべてのディスクの表示

-o cluster オプションと vxdisk コマンドを使って、クラスタ内のローカルおよび共有ディスクのグローバルビューを取得できます。コマンドにより、ディスクのサイズ、LUN で物理的に割り当てられているストレージサイズおよびアロケーションユニットサイズ、ディスクグループ、ディスクのメディアタイプ、および各ディスクが接続されているノードの数が表示されます。物理割り当てがない場合は、コマンド出力に N/A とマークされます。

```
# vxdisk -o cluster list
DEVICE                GROUP TYPE       SIZE(MB)    STATE
sys1_disk_0    -      hdd        2048       online
sys1_disk_1    -      hdd        2048       online
```

```
sys1_disk_2   -     hdd        2048       online
sys1_disk_3   -     hdd        1024       online
sys1_disk_4   -     hdd        1024       online
```

クラスタ内の特定のディスクの詳細ビューを取得するには

```
# vxdisk -o cluster list disk_name
device      :  sys1_disk_0
dg          :  -
udid        :
VMware%5FVirtual%20disk%5FDISKS%5F6000C291A7E43F02BF3E8CB067061130
dgid        :
mediatype   :  hdd
site        :  -
status      :  online
size        :  2147483648
connectivity:  sys1.example.com
```

また、vxsan ヘルパーユーティリティでは、クラスタ全体だけでなく、個別のノードおよびディスクグループのストレージの概略ビューも提供されます。

```
# vxsan list
nodes:            total=16 storage=16
diskgroups:       total=15 imported=0
hdd:              total=67 capacity=187392 MB free=166912 MB
ssd:              total=0 capacity=0 MB free=0 MB

# vxsan list devices
DEVICE                          MEDIA      SIZE(MB)   GROUP
 NODES
STATE
emc0_019b                       hdd        30720      -
 8
online
emc0_019c                       hdd        30720      -
 8
online
emc0_019d                       hdd        30720      -
 8

# vxsan list nodes
             HDD                                SSD
  ------------------------------ ---------------------------------
NODE   COUNT TOTAL(MB) FREE(MB)       COUNT TOTAL(MB) FREE(MB)
sys1    4     4096     2048           0      0         0
```

```
sys2    8      126976    125952          0       0          0
sys3    4      4096      2048            0       0          0
sys4    4      4096      3072            0       0          0

# vxsan list devices node=sys1
DEVICE                            MEDIA       SIZE(MB)    GROUP
NODES
STATE
sys1_vmdk0_0                      hdd         1024        sdg2
1
online
sys1_vmdk0_1                      hdd         1024        -
1
online
sys1_vmdk0_2                      hdd         1024        -
1
online
sys1_vmdk0_3                      hdd         1024        fssdg1
1
online

# vxsan list devices dg=testdg
DEVICE                             MEDIA       SIZE(MB)    GROUP
 NODES
STATE
sys2_vmdk0_0                       hdd         1024        testdg
 1
online
sys2_vmdk0_4                       hdd         1024        testdg
 1
online
```

vxsan（1M）のマニュアルページを参照してください。

手動による CVM クラスタメンバーシップの確立

ほとんどの場合 CVM（Cluster Volume Manager）を手動で起動する必要はありません。VCS（ Cluster Server）が起動されるときに通常は起動します。

次のコマンドを実行して、CVM を手動で起動します。

```
#vxclustadm -m vcs -t gab startnode
```

vxclustadm は、クラスタの設定情報のある main.cf 設定ファイルを読み取るため、実行されている VCS には依存しないことに注意します。通常は hastart（VCS start）コマ

ンドによって CVM が自動的に起動されるため、vxclustadm startnode コマンドを実行する必要はありません。

CVM が正しく起動されているかどうかを確認するには、次のコマンドを実行します。

```
# vxclustadm nidmap
Name          CVM Nid     CM Nid      State
sys1          0           0           Joined: Master
sys2          1           1           Joined: Slave
```

CVM マスター選択を制御する方法

マスターノードが切断されるとき、CVM（Cluster Volume Manager）はマスターロールをクラスタ内の別のノードにフェールオーバーします。CVM は、マスターロールを引き継ぐために最も適したクラスタのノードを選択します。CVM では、共有ディスクグループで最も多いディスクへの接続性があるノードを優先します。この動作は、CVM の以前のリリースの強化機能です。

通常の操作の間に、CVM は各ノードにオフセット優先設定値を動的に割り当てます。優先設定の割り当ては自動で行われ、通常はストレージ管理者は何も行う必要がありません。ただし、マスター選択に対してより大きい制御が必要な場合、カスタマイズされた優先設定値も設定できます。マスターのフェールオーバーが発生すると、CVM は新しいマスターノードを選択するために、オフセット優先設定値と一緒にカスタムノード優先設定を使用します。

p.424 の「マスターフェールオーバーへのクラスタノードの優先設定の設定について」を参照してください。

スケジュールされた保守をマスターノードで実行するには、マスターロールをクラスタ内の別のノードに手動で移行します。

p.429 の「CVM マスターの手動での変更について」を参照してください。

マスターフェールオーバーへのクラスタノードの優先設定の設定について

Cluster Volume Manager（CVM）は、ディスクグループのディスクへの接続性などの条件に基づいて、各ノードにオフセット優先設定値を動的に割り当てます。優先設定の割り当ては自動で行われ、通常はストレージ管理者は何も行う必要がありません。

マスター選択に対してより大きい制御が必要な場合、カスタマイズされた優先設定値を設定できます。CVM クラスタのどのノードがマスターロールを実行する最も適切な候補であるか判断します。CVM が最適なノードのプールからマスターノードを選択するように、高い優先設定値または低い優先設定値をノードに割り当てます。CVM は新しいマスターノードを選択するために、オフセット優先設定値と一緒にカスタムノード優先設定を使用します。CVM は、優先されたノードにマスターロールをフェールオーバーします。

クラスタノードのマスターフェールオーバーへの優先設定

Cluster Volume Manager（CVM）は、ディスク接続性などの内部条件に基づいて、クラスタの各ノードに重みの値を割り当てます。CVM が割り当てる重みの値は -100 から 100 です。負の値は、フェールオーバーの場合にノードが新しいマスターノードになる可能性が低いことを意味します。正の値は、ノードが新しいマスターノードになる可能性が高いことを意味します。

CVM のデフォルト値を使用すると望ましい動作になる場合、優先設定値を調整する必要はありません。場合によっては、管理者は、フェールオーバーの場合にどのノードをマスターにするか、よりきめ細やかな制御が必要になる場合があります。特定のノードに対して、カスタムの重み値として正の値または負の値を割り当てることができます。カスタマイズされた優先設定値は、静的で永続的です。

カスタムの重み値は、CVM が重みを割り当てる場合に影響しません。CVM は、デフォルト値の 0 ではなくカスタム値を開始点として、重み値を増加または減少させます。選択した優先設定値が望み通りの効果があるかどうか確認するには、CVM が生成する値の結果を考慮します。

たとえば、フェールオーバーの場合に NodeA が常に NodeB よりも優先されることを確認したいとします。NodeA から一部のディスクへの接続が切断された場合、CVM は NodeA の重み値を減少させます。また、CVM は NodeB の重み値を増加させることがあります。CVM が割り当てた優先設定値の重みを上書きするには、NodeA のカスタム優先設定値は、NodeB のカスタム値より少なくとも 200 以上小さくする必要があります。たとえば、CVM は NodeA に -100、NodeB に 100 を割り当てたとします。NodeA がすべてのディスクへの接続性を失った場合に、NodeB をマスターフェールオーバーの対象にする場合は、NodeA に 99 という値を割り当てます。

CVM ノードの優先設定を行う場合の注意事項

Cluster Volume Manager（CVM）クラスタのどのノードがマスターロールを実行する最も適切な候補であるか判断できます。CVM が最適なノードのプールからマスターノードを選択するように、高い優先設定値または低い優先設定値をノードに割り当てます。

CVM リソースエージェントまたは `vxclustadm` コマンドの優先設定値を設定します。優先設定値の範囲は -2147483648 ～ 2147483647 です。

カスタム優先設定を指定しない場合、CVM は最も多くのストレージを表示するノードに CVM マスターノードの優先設定を与えます。

マスターフェールオーバーの CVM 優先設定を行うには、クラスタがクラスタプロトコルバージョン 110 以上で実行されている必要があります。

次のシナリオでは、次の優先設定値を指定します。

- I/O を集約的に使用するアプリケーションを実行するクラスタ
  フェールオーバー後にアプリケーションが新しいマスターノードで実行するように、マスターフェールオーバーの優先設定を行います。マスターノードのパニックが原因で

アプリケーションがクラッシュした後、アプリケーションは新しいマスターにフェールオーバーします。リカバリ中、I/O を集約的に使用するアプリケーションがマスター上の同じ場所に配置される場合、フェールオーバーではスレーブノードとマスターノード間のメッセージ交換は行われません。この動作によって、フェールオーバーとリカバリの時間が短縮されます。

- 頻繁に管理操作が行われるクラスタでは、負荷の高い内部 I/O が作成されます。
  フェールオーバー後にアプリケーションがスレーブノードで実行するように、マスターフェールオーバーの優先設定を行います。
  Storage Foundation は、ボリュームやスナップショットの作成などの、管理操作の I/O を発行します。このリリースでは、VxVM (Veritas Volume Manager) はアプリケーション I/O の負荷が高い場合に管理 I/O を調整します。スロットルにより、アプリケーション I/O に I/O を生成した Storage Foundation の影響を軽減します。
  使用している環境で頻繁な管理操作が必要な場合、管理 I/O の効果を最小化するために、マスターフェールオーバーの優先設定も行う必要があります。アプリケーションがフェールオーバーする可能性があるノードのマスターフェールオーバーに、より低い優先設定値を設定します。

- クラスタには、ストレージ容量や処理能力の異なるノードが存在します。
  クラスタに高い処理容量を持つノードがいくつか含まれている場合、それらのノードに、マスターロールを実行するため、またはアプリケーションを実行するための優先設定を CVM に指定させたい場合があります。アプリケーション配置オプションを考慮したら、クラスタノードの優先順位を決定します。より高い処理容量またはスループットを実現するノードに、高い優先設定値を割り当ててください。
  ただし、通常のデータバックアップまたは内部の低優先度ジョブに使用される少数の低容量ノードがクラスタに含まれることがあります。これらの低容量ノードからは、すべての共有ストレージを参照することはできません。そのため、これらのノードでマスターロールを実行しない方がよい場合があります。CVM が、より適切な候補の代わりにこれらのノードを CVM マスターとして設定しないように、これらのノードには負の優先設定値を設定します。

- マスターノードがアプリケーションと同じサイトにある必要があるキャンパスクラスタ。
  マスターノードを、アプリケーションが動作しているサイトと同じ場所に配置しておきたい場合があります。アプリケーションフェールオーバーがサイト内で起こる方法を定義した場合、CVM マスターがサイト内のノードにフェールオーバーするように設定したい場合もあります。マスター障害が発生している間はマスターがサイト内にとどまるように、サイト内のすべてのノードに高い優先設定値を設定します。現在、CVM はサイトに基づいて優先設定を自動的に設定しません。

CVMCluster エージェントを使用したクラスタノードの優先設定

クラスタノードで、CVMCluster エージェントを使用してマスターフェールオーバーの優先設定値を設定できます。この方法で設定した優先設定値は、再起動後も保持されます。

CVMCluster のエージェントを使ってクラスタノードの優先設定を設定するには

1 VCS の設定を読み書き両用にします。

```
# haconf -makerw
```

2 現在の時刻のノードの優先設定を表示します。

```
# hares -display cvm_clus -attribute CVMNodePreference
```

コマンドでは、次のマスターにするローカルノードの優先設定を指定する整数値を表示します。すべてのノードでコマンドを実行し、どのノードが次のマスターになるための高い優先設定を持っているか確認します。

3 新しい優先設定値を設定します。

```
# hares -modify cvm_clus CVMNodePreference ¥
"node1=weight1, node2=weight2, ..."
```

マスターフェールオーバーの場合に vxclustadm コマンドを使用したクラスタノードの優先設定値の設定

このセクションの手順では、vxclustadm コマンドを使用してマスターフェールオーバーでクラスタノードのユーザー優先設定値を設定する方法を説明します。この方法で設定した優先設定値は、再起動後は保持されません。

vxclustadm コマンドを使ってクラスタノードの優先設定値を設定するには

1 既存の優先設定を表示するには、次のコマンドを使用します。

```
# vxclustadm getpreference
```

2 新しい優先設定値を設定するには、次のコマンドを実行します。

```
# vxclustadm setpreference value
```

マスターフェールオーバーへのクラスタノードの優先設定値の設定例

この例では、クラスタ環境に 3 つのタイプのノードが含まれているケースを説明します。最適な候補にCVM（Cluster Volume Manager）がマスターロールをフェールオーバーするように、ノードの各タイプにマスター優先設定を定義できます。

次のプールのように、3 つのタイプのノードを考慮します。

- Pool1（Node1、Node2、Node3）

これらのノードは大容量(ストレージとプロセス)で、完全なストレージを表示できます。I/O を集約的に使用するアプリケーションは、クラスタのこれらのノードで稼働します。ディスク容量の 30% が失われても、他のプールからのノードではなく、このプールからノードを選択してもかまいません。

- Pool2(Node4、Node5)
  クラスタには、少数の低容量(ストレージとプロセス)ノードがあります。内部(社内)アプリケーションは、データに対して後処理を行うためにこれらのノードを使用します。これらのノードの一部は、スナップショットとバックアップにも使用されます。次の場合に、このプールからノードを選択できます。
  Pool1 に選択するノードがない、または
  Pool1 のすべてのノードが、優先設定値を Pool2 からのノードの優先設定値以下に減らす多くのディスクにアクセスできなくなった。

- Pool3(Node6、Node7)
  これらのノードは、すべてのボリューム上で稼働する必要がないアプリケーションを実行します。これらのノードでは、ストレージ(限られたストレージ表示)の表示が制限されている場合があります。CVM はすべてのストレージを表示しないため、このプール内のノードの優先設定値を内部でオフセットします。これらのノードのいずれかがマスターノードになる可能性を減らすため、負の優先設定値を割り当てることができます。

カスタム優先設定を定義しない場合、ディスクの可用性によっては、CVM は優先設定をオフセットとして設定します。時間 A で、現在のマスターである Node1 がクラスタから切断されたと仮定します。CVM は優先設定を計算し、最も高い接続性がある Node5 を選択します。Node5 が切断された場合、CVM は Node4 を選択します。Pool2 のノードは Pool1 のノードよりも優先されます。

Pool	ノード	時間 A での CVM オフセット
Pool1	Node1(現在のマスター)	0
Pool1	Node2	-30
Pool1	Node3	-25
Pool2	Node4	-20
Pool2	Node5	0
Pool3	Node6	-50
Pool3	Node7	-50

この例では、Pool1 のノードにより高い優先設定値を指定し、Pool3 のノードに低い優先設定値を指定しようとしています。次の図に、クラスタ内のノードに設定可能な優先設定値を示します。

クラスタノードの優先設定値を設定するには

1 Pool1 の各ノードでは、優先設定を 30 に設定します。

```
# hares -modify cvm_clus CVMNodePreference ¥
"node1=30, node2=30, node3=30"
```

2 Pool3 の各ノードでは、優先設定を -50 に設定します。

```
# hares -modify cvm_clus CVMNodePreference "node6=-50, node7=-50"
```

前のように優先設定値を指定したら、フェールオーバーの場合には望ましい動作が反映されます。Node1 が失敗した場合、Pool1 の他のノードはマスターを切り替える可能性が最も高い候補です。Node3 はディスクの 25% を失いましたが、引き続き他のプールのノードよりも優先されます。Node5 は高い接続性がありますが、Pool1 のノードの方が Node5 よりも優先されます。

ノード	時間 A での CVM オフセット	カスタム優先設定	優先設定の合計
Node1（現在のマスター）	0	30	30
Node2	-29	30	1
Node3	-25	30	5
Node4	-20	0	-20
Node5	0	0	0
Node6	-50	-50	-100
Node7	-50	-50	-100

CVM マスターの手動での変更について

マスターのロールを手動で移行するときに、マスターロールの引き継ぎ先のノードを指定する必要があります。マスターロールを引き継ぐ最適なノードを決定できるように、ノードで設定される優先設定値を表示できます。

マスターノードを手動で変更するとき、クラスタはオンライン状態のままとなり、クラスタから切断するためのノードは必要ありません。ただし、CVM（Cluster Volume Manager）は

アプリケーション I/O を静止します。したがって、パフォーマンスに対する効果を最小化するために、マスター切り替え操作を適切な時間にスケジュールします。

マスターフェールオーバーの後、CVM がマスターとして適切でないノードを選択したことが分かったら、マスターロールを手動で変更できます。この場合、クラスタノードのカスタマイズされたフェールオーバー優先設定値を変更したい場合もあります。

p.424 の「マスターフェールオーバーへのクラスタノードの優先設定の設定について」を参照してください。

マスターの手動での変更に関する注意事項

マスターがクラスタのマスターに最適なノード上で動作していない場合、マスターを手動で変更できます。考えられるシナリオを次に示します。

- 現在実行中のマスターがいくつかのディスクにアクセスできなくなりました。
  デフォルトでは、CVM は I/O 転送を使用してこのシナリオを処理します。ただし、ディスクへのアクセス権を持つノードにアプリケーションをフェールオーバーする方がよい場合があります。また、アプリケーションを移動するときに、新しいノードにマスターロールを再配置する方がよい場合もあります。たとえば、マスターノードとアプリケーションを同じノード上に配置したい場合があります。
  元のマスターノードをクラスタから切断されないようにしながら、マスターロールを移動するためにマスターの切り替え操作を使用できます。マスターロールとアプリケーションが両方とも他のノードに切り替えられたら、クラスタから元のノードを削除したい場合があります。ファイルシステムをマウント解除し、正常にノードを終了できます。その後にノードの保守を行うことができます。
- マスターノードは、アプリケーション負荷と内部生成された管理 I/O が重複すると、適切に縮小されません。
  配置の手法を再度評価してからマスターノードを再配置するように選択できます。

p.425 の「CVM ノードの優先設定を行う場合の注意事項」を参照してください。

CVM マスターの手動での変更

CVM（Cluster Volume Manager）マスターは、クラスタがオンライン状態の間に、クラスタ内の 1 つのノードから別のノードに手動で変更できます。CVM は、マスターノードを移行し、クラスタを再設定します。

クラスタが Veritas Volume Manager（VxVM）の設定変更やクラスタの再設定操作を処理していないときには、マスターを切り替えることを推奨します。ほとんどの場合、CVM は、VxVM またはクラスタで設定の変更が行われていることを検出した場合、マスター変更操作を中止します。マスター変更操作でのクラスタの再設定が開始された後は、設定の変更を必要とする他のコマンドは、マスターの切り替えが完了するまでは失敗します。

p.432 の「CVM マスター切り替え時のエラー」を参照してください。

オンラインのマスターを変更するには、クラスタプロトコルのバージョンが 100 以上である必要があります。

CVM マスターを手動で変更するには

1 現在のマスターを表示するため、次のいずれかのコマンドを使います。

```
# vxclustadm nidmap
Name               CVM Nid    CM Nid    State
sys1               0          0         Joined: Slave
sys2               1          1         Joined: Master
```

```
# vxdctl -c mode
mode: enabled: cluster active - MASTER
master: sys2
```

この例では、CVM マスターは sys2 です。

2 CVM マスターを変更するには、クラスタ内の任意のノードから次のコマンドを実行します。

```
# vxclustadm setmaster nodename
```

nodename で、新しい CVM マスターの名前を指定します。

次の例では、クラスタのマスターを sys2 から sys1 に変更します。

```
# vxclustadm setmaster sys1
```

3 マスターの切り替えを監視するには、次のコマンドを使います。

```
# vxclustadm -v nodestate
 state: cluster member
        nodeId=0
        masterId=0
        neighborId=1
        members[0]=0xf
        joiners[0]=0x0
        leavers[0]=0x0
        members[1]=0x0
        joiners[1]=0x0
        leavers[1]=0x0
        reconfig_seqnum=0x9f9767
        vxfen=off
state: master switching in progress
reconfig: vxconfigd in join
```

この例の状態は、マスターの切り替えが進行中であることを示します。

4 マスターが正常に変更されたかどうかを確認するには、次のコマンドの 1 つを使います。

```
# vxclustadm nidmap
Name               CVM Nid    CM Nid    State
sys1             0          0         Joined: Master
sys2             1          1         Joined: Slave
```

```
# vxdctl -c mode
mode: enabled: cluster active - MASTER
master: sys1
```

CVM マスター切り替え時のエラー

クラスタが VxVM（Veritas Volume Manager）やクラスタの設定変更を処理していないときには、マスターを切り替えることを推奨します。

ほとんどの場合、進行中の設定変更あると、CVM（Cluster Volume Manager）はそれを検出し、マスターの変更操作を中止します。CVM は障害の理由をシステムログに記録します。場合によって、障害は `vxclustadm setmaster` の出力で次のように表示されます。

```
# vxclustadm setmaster sys1
VxVM vxclustadm ERROR V-5-1-15837 Master switching, a reconfiguration
 or
```

```
a transaction is in progress.
Try again
```

場合によって、別の再設定操作によってマスターの切り替え操作が中断されると、マスターの変更が失敗します。この場合、既存のマスターがクラスタのマスターのままになります。再設定が完了したら、vxclustadm setmaster コマンドを再発行し、マスターを変更します。

マスターの切り替え操作が再設定を開始すると、設定変更を開始するコマンドはすべて、次のエラーで失敗します。

```
Node processing a master-switch request. Retry operation.
```

このメッセージが表示された場合は、マスターの切り替えが完了した後にコマンドを再試行します。

CVM 環境でのアプリケーション分離機能の有効化

アプリケーションの分離機能を有効にすると、アプリケーションがオフラインになり、アプリケーションの停止時間が生じることがあります。CVMCluster リソースの CVMDGSubClust 属性を 1 に設定して VCS 設定ファイルを更新します。変更はクラスタの再ブート後も変化しません。

アプリケーションの分離機能を有効にするには、VCS ポート「h」がアクティブである必要があります。

アプリケーションの分離機能を有効にするには

1. GAB ポート h がアクティブであることを確認します。

   ```
   # hastatus
   ```

2. クラスタ内のすべてのノードで CVM グループをオフラインにます。

   ```
   # hagrp -offline cvm -sys sys_name
   ```

3. CVMCluster リソースの CVMDGSubClust 属性を有効にするには、次の操作をします。

   ```
   # haconf -makerw
   # hares -modify cvm_clus CVMDGSubClust 1
   # haconf -dump -makero
   ```

4 すべてのノードで CVM グループをオンラインにします。

```
# hagrp -online cvm -sys sys_name
```

すべてのノードで CVM グループをオンラインにしたら、アプリケーションの分離機能を有効にして使用できるようにします。

5 クラスタ内のノードにディスクグループをインポートします。

ディスクグループはクラスタ内のノードに自動インポートされないので、クラスタ内の必要なノードにディスクグループをインポートするように VCS を設定する必要があります。必要に応じて、次のコマンドのいずれかを使用します。

`cfsdgadm` ブロックデバイスを使用している場合は、このオプションを使用します。

`cfsmntadm` ブロックデバイスにクラスタのファイルシステムを作成する場合は、このオプションを使用します。

```
# cfsdgadm add dgname sys1 sys2
```

これにより、VCS 設定ファイルが次のように更新されます。

```
group vrts_vea_cfs_int_cvmvoldg2 (
   SystemList = { sys1 = 5, sys2 = 6 }
   AutoFailOver = 0
   Parallel = 1
   AutoStartList = { sys1, sys2 }
   )
   CVMVolDg cvmvoldg2 (
      Critical = 0
      CVMDiskGroup = bdg
      CVMActivation @sys1 = sw
      CVMActivation @sys2 = sw
      NodeList = { sys1, sys2 }
      )
      requires group cvm online local firm
        // resource dependency tree
        //
        //      group vrts_vea_cfs_int_cvmvoldg2
        //      {
        //      CVMVolDg cvmvoldg2
        //      }
```

```
# cfsmntadm add dgname volname /mnt1 ¥
sys1=cluster sys2=cluster
```

これにより、VCS 設定ファイルが次のように更新されます。

```
group vrts_vea_cfs_int_cfsmount1 (
  SystemList = { sys1 = 5, sys2 = 7 }
```

```
AutoFailOver = 0
Parallel = 1
AutoStartList = { sys1, sys2 }
)
CFSMount cfsmount1 (
   Critical = 0
   MountPoint = "/mnt1"
   BlockDevice = "/dev/vx/dsk/adg/avol1"
   MountOpt @sys1 = "cluster"
   MountOpt @sys2 = "cluster"
   NodeList = { sys1, sys2 }
   )
   CVMVolDg cvmvoldg1 (
      Critical = 0
      CVMDiskGroup = adg
      CVMVolume = { avol1 }
      CVMActivation @sys1 = sw
      CVMActivation @sys2 = sw
      NodeList = { sys1, sys2 }
      )
      requires group cvm online local firm
      cfsmount1 requires cvmvoldg1
      // resource dependency tree
      //
      //      group vrts_vea_cfs_int_cfsmount1
      //      {
      //      CFSMount cfsmount1
      //          {
      //          CVMVolDg cvmvoldg1
      //          }
      //      }
```

6 アプリケーションの分離機能が有効になっていることを確認します。

```
# hares -display cvm_clus | grep CVMDGSubClust
cvm_clus     CVMDGSubClust          global        1
```

CVM クラスタでのアプリケーション分離機能の無効化

アプリケーション分離機能を無効にすると、従来の CVM の動作が復元されます。VCS ポート「h」がアクティブである必要があります。

アプリケーション分離機能を無効にするには

1 GAB ポート h がアクティブであることを確認します。

```
# hastatus
```

2 クラスタ内のすべてのノードで CVM グループをオフラインにます。

```
# hagrp -offline cvm -sys sys_name
```

3 CVMCluster リソースの CVMDGSubClust 属性を無効にするには、次の操作をします。

```
# haconf -makerw
# hares -modify cvm_clus CVMDGSubClust 0
# haconf -dump -makero
```

4 すべてのノードで CVM グループをオンラインにします。

```
# hagrp -online cvm -sys sys_name
```

5 アプリケーション分離機能が無効になっていることを確認します。

```
# hares -display cvm_clus | grep CVMDGSubClust
cvm_clus     CVMDGSubClust          global        0
```

手動でのディスクグループマスターの変更

サブクラスタがオンラインである場合、ディスクグループマスターをサブクラスタにあるノードから別のノードに手動で変更できます。CVM は、マスターノードを移行し、クラスタを再設定します。

ディスクグループマスターを手動で変更するには

1 現在のマスターを表示して次のコマンドを実行します。

```
# vxdg nidmap
Nidmap of DG cluster dg1

Name                            CVM Nid    CM Nid     State

sys1                            0          0          Joined:
Master
sys2                            1          1          Joined:
Slave
sys3                            2          8          Joined:
Slave
sys4                            3          10         Joined:
Slave

Nidmap of DG cluster dg2

Name                            CVM Nid    CM Nid     State

sys3                            1          0          Joined:
Slave
sys2                            0          1          Joined:
Master

Nidmap of DG cluster dg3

Name                            CVM Nid    CM Nid     State

sys1                            1          0          Joined:
Slave
sys4                            0          8          Joined:
Master
sys3                            2          10         Joined:
Slave
```

2 ディスクグループのサブクラスタ内にある任意のノードからマスターを変更するには、次のコマンドを実行します。

```
# vxdg -g dgname setmaster nodename
```

nodename には、新しいディスクグループマスターの名前を指定します。

次の例では、ディスクグループ dg2 のマスターを sys2 から sys3 に変更します。

```
# vxdg -g dg2 setmaster sys3
```

3 マスターが正常に変更されたことを確認します。

```
# vxdg nidmap dg2
Nidmap of DG cluster dg2

Name                                CVM Nid     CM Nid      State

sys3                                1           0           Joined:
Master
sys2                                0           1           Joined:
Slave
```

例: マスターフェールオーバーへのサブクラスタノードの優先設定値の設定

優先設定値は、ディスクグループサブクラスタ内のマスターノードのフェールオーバーに最適な候補を決定します。この値は、再ブート後は保持されません。優先設定機構では、従来の CVM 環境の値を保持します。

マスターフェールオーバーのサブクラスタノードの優先設定値を設定するには

1 既存の優先設定を表示するには、次のコマンドを使用します。

```
# vxdg getpreference
preference = 10
preference delta = -0
```

2 新しい優先設定値を設定するには、次のコマンドを実行します。

```
# vxdg -g dgname setpreference value
```

共有ディスクグループの手動インポート

次のコマンドを使って、共有ディスクグループを手動でインポートできます。

```
# vxdg -s import dg_name
```

共有ディスクグループの手動デポート

次のコマンドを使って、共有ディスクグループを手動でデポートできます。

```
# vxdg deport dg_name
```

共有ディスクのデポートにより、ディスク上の SCSI-3 PGR が削除されるので注意してください。

クラスタ内のノードへのリモートストレージのマッピング

リモートノードで利用可能なディスクまたはディスクグループをローカルノードのクラスタにマッピングすることができます。リモートストレージはノードに直接接続されてはいませんが、ローカルに接続されているディスクと同じように使用できます。必要に応じて、同じストレージをクラスタ内の複数のノードにマッピングすることができます。マッピングされたストレージは、ローカルノードで共有または専用ディスクグループを作成する場合にも使用できます。デバイスが正常にマッピングされると、リモートストレージはノードで表示可能な他のローカルストレージと同じように表示されます。

ディスクグループをマッピングすると、ディスクグループに存在するすべてのディスクがノードにマッピングされます。

メモ: リモートの個別のディスクのマッピングは、クラスタ再ブート後は持続されません。ただし、ディスク上で作成したディスクグループをインポートすると、自動的にノードにマップされます。マップされたディスクグループでの分割操作または結合操作は、マッピングに影響しません。前述のように、同じディスクグループを複数のノードにインポートするには、関連付けられているディスクがクラスタ内のすべてのノードにマップされる共有モードでインポートする必要があります。

リモートストレージをクラスタ内のノードにマップするには

1 クラスタ内の CVM ノードの一覧を表示します。

```
[root@sys2~]# /etc/vx/bin/vxclustadm nidmap
Name  CVM Nid    CM Nid       State
sys1    1          0          Joined: Slave
sys2    3          1          Joined: Slave
sys3    2          2          Joined: Slave
sys4    0          3          Joined: Master
```

2 クラスタ内で利用可能なディスクの一覧を表示します。

```
[root@sys2~]# vxdisk -o cluster list
DEVICE                          MEDIA      SIZE(MB)    GROUP
    NODES      STATE
emc_clariion0_105               hdd        2048        -
    4          online
emc0_03ce                       hdd        2048        -
    4          online
sys1_disk_0             hdd        200        -           1
      online
sys1_disk_1             hdd        200        -           1
      online
sys2_disk_0             hdd        200        -           1
      online
sys2_disk_1             hdd        200        -           1
      online
sys2_disk_2             hdd        2048       -           4
      online
sys2_disk_3             hdd        2048       -           4
      online
sys3_disk_0             hdd        200        -           1
      online
sys3_disk_1             hdd        200        -           1
      online
sys4_disk_0             hdd        200        -           1
      online
sys4_disk_1             hdd        200        -           1
      online
```

3 ディスクをローカルノードにマップするには、次のコマンドを実行します。

```
[root@sys2~]# vxdisk map sys1_disk_1
```

sys1_disk_1 はリモートディスクの名前です

ディスクグループをローカルノードにマップするには、次のコマンドを実行します。

```
[root@sys2~]# vxdisk map -G sys1_dg_1
```

sys1_dg_1 はリモートディスクグループの名前です

4 マッピングを確認します。

```
[root@sys2 ~]# vxdisk list | grep sys1_disk_1
sys1_disk_1 auto:cdsdisk    -            -            online
remote
```

クラスタ内のノードからのリモートストレージマッピングの削除

リモートストレージマッピングをノードから削除するには、vxdisk unmap コマンドを使います。ストレージがノードのローカルストレージ一覧から削除されます。ディスクグループマッピングを削除すると、ディスクグループ内のすべてのディスクがノードのローカルストレージ一覧から削除されます。

ディスクマッピングを削除するには:

```
# vxdisk unmap sys1_disk_1
```

sys1_disk_1 はリモートディスクの名前です

ディスクグループマッピングを削除するには:

```
# vxdisk unmap -G sys1_disk_1
```

sys1_dg_1 はリモートディスクグループの名前です

手動による共有ボリュームの起動

手動による CVM（Cluster Volume Manager）共有ディスクグループのインポートの後には、次のようにディスクグループのボリュームを手動で起動する必要があります。

```
# vxvol -g dg_name startall
```

ボリュームが起動されているかどうかを確認するには、次のコマンドを実行します。

```
# vxprint -htrg dg_name | grep ^v
```

起動されるボリュームに、状態「enabled」が表示されます。

CVM ポートの状態の評価

CVM（Cluster Volume Manager）カーネル（vxio ドライバ）は、カーネルメッセージにポート「v」、クラスタノード間の vxconfigd 通信にポート「w」、SmartIO VxVM キャッシュの一貫性を保つための GLM（Group Lock Manager）通信にポート「m」、スレーブノードからマスターノードへのコマンド転送にポート「u」を使います。次のコマンドでは、CVM ポートの状態が表示されます。

```
# gabconfig -a | egrep "Port [vwmu]"
```

CVM が SFCFSHA クラスタで実行されているかどうかの確認

次のオプションを使って、SFCFSHA クラスタで Cluster Volume Manager が起動されているか、または起動されていないかを確認できます。

次の出力が、クラスタのメンバーになっていないノードに表示されます。

```
# vxdctl -c mode
mode: enabled: cluster inactive
# vxclustadm -v nodestate
state: out of cluster
```

マスターノードには、次の出力が表示されます。

```
# vxdctl -c mode

mode: enabled: cluster active - MASTER
master: sys1
```

スレーブノードには、次の出力が表示されます。

```
# vxdctl -c mode

mode: enabled: cluster active - SLAVE
master: sys2
```

次のコマンドを実行すると、すべての CVM ノードを同時に表示することができます。

```
# vxclustadm nidmap

Name      CVM Nid    CM Nid     State
sys1         0          0     Joined: Master
sys2         1          1     Joined: Slave
```

CVM メンバーシップの状態の確認

CVM の状態は、次のように確認できます。

```
# vxclustadm -v nodestate
state: joining
        nodeId=0
        masterId=0
        neighborId=1
        members=0x3
        joiners=0x0
        leavers=0x0
        reconfig_seqnum=0x72a10b
        vxfen=on
```

状態は、CVM がカーネルレベルの参加を完了し、vxconfigd レベルの参加を行っている最中であることを示します。

vxdctl -c mode コマンドは、ノードが CVM マスターか CVM スレーブかを示しています。

CVM 共有ディスクグループの状態の確認

次のコマンドを使って、現在 SFCFSHA クラスタにインポートされている共有ディスクグループを一覧表示できます。

```
# vxdg list |grep shared

oradatadg enabled,shared 1052685125.1485.sys1
```

アクティブ化モードの確認

SFCFSHA クラスタでは、共有ディスクグループの有効化は、クラスタノードそれぞれで「shared-write」に設定される必要があります。

「共有書き込み」のアクティブ化が設定されるかどうかを確かめるには:

```
# vxdg list dg_name |grep activation

local-activation: shared-write
```

「共有書き込み」のアクティブ化が設定されていない場合は、次のコマンドを実行します。

```
# vxdg -g dg_name set activation=sw
```

CVM ログファイル

/var/VRTSvcs/log ディレクトリには、エージェントログファイルが格納されています。

```
# cd /var/VRTSvcs/log
# ls -l *CVM* engine_A.log
```

```
CVMCluster_A.log      # CVM Agent log
CVMVolDg_A.log        # CVM VolDg Agent log
CVMVxconfigd_A.log    # CVM vxconfigd Agent log
engine_A.log          # VCS log
```

cmdlog ファイルを使って、すでに実行された CVM(Cluster Volume Manager)コマンドからなるリストを表示できます。このファイルは、/var/adm/vx/cmdlog にあります。

ノードの状態の要求とマスターノードの検出

vxdctl ユーティリティは、vxconfigd ボリューム設定デーモンの動作を制御します。-c オプションを指定すると、クラスタ情報を参照し、どのノードがマスターであるかを確認することができます。vxconfigd デーモンのクラスタ機能が有効かどうかを確認するには、次のコマンドを使います。

```
vxdctl -c mode
```

表 15-2 に、クラスタノードの現在の状態に応じて出力される可能性があるメッセージを示します。

表 15-2　　クラスタの状態メッセージ

状態メッセージ	説明
mode: enabled: cluster active - MASTER master: mozart	このノードはマスターノードです。
mode: enabled: cluster active - SLAVE master: mozart	このノードはスレーブノードです。
mode: enabled: cluster active - role not set master: mozart state: joining reconfig: master update	このノードにはまだ役割が割り当てられていません。現在は、クラスタへの結合処理の実行中です。
mode: enabled: cluster active - SLAVE master: mozart state: joining	このノードはスレーブノードとして設定されています。現在は、クラスタへの結合処理の実行中です。

状態メッセージ	説明
mode: enabled: cluster inactive	クラスタはこのノードでアクティブではありません。
mode: booted: master: ts4200-04	ルートディスクのカプセル化を有効にしますが、トランザクションは有効にしません。
mode: disabled:	トランザクションを無効にします。

vxconfigd デーモンが DISABLED 状態の場合は、クラスタ情報は表示されません。

vxdctl(1M)マニュアルページを参照してください。

LUN が共有ディスクグループの一部であるかどうかの判別

vxdisk ユーティリティは、VxVM(Veritas Volume Manager)ディスクを管理します。vxdisk ユーティリティを使って、LUN がクラスタ共有ディスクグループの一部であるかどうかを判別するには、次のコマンドを使います。

```
# vxdisk list accessname
```

ここで、*accessname* はディスクアクセス名(またはデバイス名)です。

例として、このコマンド(デバイス sde に対する)からの出力の一部を次に示します。

```
Device:     sde
devicetag:  sde
type:       auto
clusterid:  cvm2
disk:       name=shdg01 id=963616090.1034.cvm2
timeout:    30
group:      name=shdg id=963616065.1032.cvm2
flags:      online ready autoconfig shared imported
...
```

clusterid フィールドを cvm2(クラスタの名前)を設定し、flags フィールドが shared のエントリを含んでいることに注意してください。imported フラグは、ノードがクラスタの一部で、ディスクグループをインポートする場合のみ設定します。

共有ディスクグループの一覧表示

vxdg は、共有ディスクグループに関する情報の一覧表示に使われます。すべてのディスクグループに関する情報を表示するには、次のコマンドを入力します。

```
# vxdg list
```

このコマンドの出力例を次に示します。

```
NAME        STATE               ID
group2      enabled,shared      774575420.1170.teal
group1      enabled,shared      774222028.1090.teal
```

共有ディスクグループは、共有(shared)フラグで示されます。

共有ディスクグループに関する情報のみを表示するには、次のコマンドを入力します。

```
# vxdg -s list
```

このコマンドの出力例は、次のとおりです。

```
NAME      STATE             ID
group2    enabled,shared    774575420.1170.teal
group1    enabled,shared    774222028.1090.teal
```

特定のディスクグループ 1 つに関する情報を表示するには、次のコマンドを入力します。

```
# vxdg list diskgroup
```

マスターノード上で vxdg list group1 コマンドを実行した場合の出力例を次に示します。

```
Group:     group1
dgid:      774222028.1090.teal
import-id: 32768.1749
flags:     shared
version:   140
alignment: 8192 (bytes)
ssb:       on
local-activation: exclusive-write
cluster-actv-modes: node0=ew node1=off
detach-policy: local
dg-fail-policy: leave
copies:    nconfig=2 nlog=2
config:    seqno=0.1976 permlen=1456 free=1448 templen=6
loglen=220
config disk sdk copy 1 len=1456 state=clean online
config disk sdk copy 1 len=1456 state=clean online
log disk sdk copy 1 len=220
log disk sdk copy 1 len=220
```

[flags]フィールドには、shared が設定されることに注意してください。同じコマンドをスレーブノード上で実行した場合、出力は多少異なります。[local-activation]フィールドおよび[cluster-actv-modes]フィールドには、このノードおよびクラスタ内の各ノードのアクティベーションモードが表示されます。 detach-policy フィールドと dg-fail-policy フィールドはそれぞれ、クラスタがディスクへの接続を失った場合、およびディスク上の設定とログのコピーへの接続を失った場合のクラスタの動作を示します。

共有ディスクグループの作成

VxVM（Veritas Volume Manager）を使うと、共有ディスクグループを作成できます。共有ディスクグループを作成する vxdg コマンドは、マスターノードまたはスレーブノードで実行できます。スレーブノードでディスクグループを作成すると、vxdg コマンドはマスターに送られ、マスターで実行されます。

クラスタを設定するためにクラスタソフトウェアを実行している場合は、次のコマンドを使って共有ディスクグループを作成することができます。

```
# vxdg -s init diskgroup [diskname=]devicenames
```

ここで、*diskgroup* にはディスクグループ名を、*diskname* には VxVM ディスクに対して選択された管理名を、*devicename* にはデバイス名（または、ディスクアクセス名）を指定します。

警告: OS は、ディスクが共有されているか判定できません。複数のシステムによってアクセスされる可能性のあるディスクを扱う場合にデータの一貫性を確保するには、ディスクグループにディスクを追加する際に正確な指定を行ってください。VxVM は、そのディスクからアクセス可能なノードがクラスタ内の唯一のノードである場合に、物理的に共有されていないディスクの共有ディスクグループへの追加を許可します。ただし、これは他のノードがクラスタに結合できないことを意味します。さらに、2 つのノード上の別のディスクグループへ同時に同じディスクの追加を試みると、その結果は未定義になります。すべての設定は、1 つのノードでのみ、可能であればマスターノードで行ってください。

共有ディスクグループのインポート

共有ディスクグループは、マスターノードまたはスレーブノードでインポートできます。共有ディスクグループをインポートする vxdg コマンドをスレーブノードで実行すると、コマンドはマスターに送られ、マスターで実行されます。

ディスクグループは、vxdg -s import コマンドを使って、共有ディスクグループとしてインポートすることができます。クラスタソフトウェアが実行される前にディスクグループが設定された場合、そのディスクグループは、次のコマンドを使って、クラスタにインポートすることができます。

```
# vxdg -s import diskgroup
```

diskgroup にはディスクグループの名前または ID を指定します。クラスタの次の再起動以降は、ディスクグループは自動的に共有ディスクグループとしてインポートされます。vxdg *ユーティリティを実行する前に、ディスクグループのデポートが*(vxdg deport diskgroup コマンドを使って)必要になる可能性があることに、注意してください。

ディスクグループの強制インポート

vxdg コマンドに、-f オプションを指定して、ディスクグループを強制的にインポートすることができます。

警告: 強制オプション(-f)の使用にあたっては注意が必要です。データの破損などの結果を招く可能性があることを十分に認識したうえで使ってください。

クラスタの再起動時、VxVM (Veritas Volume Manager)は、次の理由のいずれかにより、ディスクグループの自動インポートを拒否することがあります。

- ディスクグループ内のあるディスクが、ハードウェアエラーのため、アクセス不能である。この場合には、次のコマンドを使って、そのディスクグループを強制的に再インポートします。

  ```
  # vxdg -s -f import diskgroup
  ```

  メモ: 強制インポート後は、ボリューム上のデータを利用できない場合や、一部のボリュームが無効な状態になっている場合があります。

- 共有ディスクグループの一部のディスクにアクセスできないため、ディスクグループがすべてのディスクにはアクセスできません。この場合には、結果としてミラーに不整合を起こすことがあり、強制インポートは安全ではないため実行してはいけません。

共有ディスクグループから専用ディスクグループへの変換

共有ディスクグループは、マスターノードまたはスレーブノードで変換できます。共有ディスクグループを変換する vxdg コマンドをスレーブノードで実行すると、コマンドはマスターに送られ、マスターで実行されます。

共有ディスクグループを専用ディスクグループに変換するには、まず、次のコマンドを使ってマスターノードからその共有ディスクグループをデポートします。

```
# vxdg deport diskgroup
```

その後、次のコマンドを使って、任意のクラスタノード上にそのディスクグループを再インポートします。

```
# vxdg import diskgroup
```

共有ディスクグループ間のオブジェクト移動

共有ディスクグループ間のオブジェクトの移動は、マスターノードまたはスレーブノードで実行できます。共有ディスクグループ間でオブジェクトを移動する vxdg move コマンドをスレーブノードで実行すると、コマンドはマスターに送られ、マスターで実行されます。

vxdg move コマンドを使って、ディスクや最上位ボリュームなどの自己完結型の VxVM (Veritas Volume Manager) オブジェクトのセットをディスクグループ間で移動できます。クラスタ環境では、ディスクグループがインポートされている任意のクラスタノードから専用ディスクグループ間で、このようなオブジェクトを移動することができます。

p.956 の「ディスクグループ間のオブジェクト移動」を参照してください。

共有ディスクグループの分割

vxdg split コマンドを使って、自己完結型の VxVM (Veritas Volume Manager) オブジェクトのセットをインポートしたディスクグループから削除し、新しく作成したディスクグループに移動できます。

p.959 の「ディスクグループの分割」を参照してください。

専用ディスクグループを分割すると専用ディスクグループが 1 つ作成され、また共有ディスクグループを分割すると共有ディスクグループが 1 つ作成されます。クラスタ環境では任意のクラスタノードから専用ディスクグループを分割することができます。

共有ディスクグループを分割するか、または共有されるターゲットディスクグループをマスターノードまたはスレーブノードに作成できます。共有ディスクグループを分割するコマンド、または共有されるターゲットディスクグループをスレーブノードに作成するコマンドを実行すると、コマンドはマスターに送られ、マスターで実行されます。

共有ディスクグループの結合

専用ディスクグループと共有ディスクグループを結合することはできません。

vxdg join コマンドを使って、2 つのインポートしたディスクグループの内容を結合できます。クラスタ環境では、任意のクラスタノードから 2 つの専用ディスクグループを結合することができます。

ソースディスクグループとターゲットディスクグループの両方が共有されている場合は、マスターノードまたはスレーブノードから結合を実行できます。結合を実行するコマンドをスレーブノードで実行すると、コマンドはマスターに送られ、マスターで実行されます。

p.961 の「ディスクグループの結合」を参照してください。

共有ディスクグループ上のアクティベーションモードの変更

クラスタノードによる共有ディスクグループへのアクセスに対するアクティベーションモードは、そのノード上で直接設定されます。

共有ディスクグループのアクティベーションモードは、次のコマンドを使って、変更することができます。

```
# vxdg -g diskgroup set activation=mode
```

アクティベーションモードには、排他書き込み(exclusivewrite、ew)、読み取り専用(readonly、ro)、共有読み取り(sharedread、sr)、共有書き込み(sharedwrite、sw)、非活性(off)のいずれかを指定します。

このコマンドを使って共有ディスクグループのアクティベーションモードを変更する場合は、次に示すように、最初にアクティベーションモードを off に変更してからそれ以外の値に設定する必要があります。

```
# vxdg -g myshdg set activation=off
# vxdg -g myshdg set activation=readonly
```

p.174 の「共有ディスクグループのアクティブ化モード」を参照してください。

共有ディスクグループでの I/O 転送の有効化

I/O 転送ポリシーのデフォルトは[**オフ**]です。すべてのノードで I/O 転送を有効にできます。

共有ディスクグループでの I/O 転送を有効化する方法

- 指定したディスクグループの I/O 転送ポリシーを[オン]に設定します。

  ```
  # vxdg -g diskgroup set ioship=on
  ```

共有ディスクグループでの I/O 転送を無効化する方法

- 指定したディスクグループの I/O 転送ポリシーを[オフ]に設定します。

  ```
  # vxdg -g diskgroup set ioship=off
  ```

共有ディスクグループの切断ポリシーの設定

切断ポリシーのデフォルト値は global です。

共有ディスクグループの切断ポリシーを変更するには

- 切断ポリシーを共有ディスクグループで local に設定するには、次を実行します。

  ```
  # vxdg -g diskgroup set diskdetpolicy=local
  ```

  切断ポリシーを global (デフォルト)に設定するには、次を実行します。

  ```
  # vxdg -g diskgroup set diskdetpolicy=global
  ```

ボリュームレベルの I/O 転送

DRL (dirty region logging) を有効にすると、システムクラッシュ後のミラーボリュームのリカバリが高速化されます。アプリケーションがミラーボリュームにデータのストリームを書き込むと、アプリケーションの遅延時間に影響を与えるシリアル方式で DRL ログとデータの書き込みが行われます。クラスタ相互接続でのネットワークの遅延によって、I/O パフォーマンスの遅延がさらに長くなる可能性があります。

VxVM (Veritas Volume Manager) は、ボリュームレベルの I/O 転送機能と、強化された I/O パフォーマンスを提供します。ボリュームレベルの I/O 転送では、ネットワーク経由で送信する前に DRL ログとデータの書き込みを収集します。ただし、この収集によって、DRL ログとデータの書き込みの実行順序は変更されることはありません。これにより書き込み I/O 操作に関連するネットワークホップ数が削減され、書き込み操作の待機時間が短縮されます。

ボリュームレベルの I/O 転送のサポートには、次のような制限があります。

- ローカルディスク
- ディスクグループバージョン 260 以降
- CVM プロトコルバージョン 210 以降

ボリュームレベルの I/O 転送の有効化または無効化

ボリュームレベルの I/O 転送中にネットワークホップ数を減らすには、データボリュームと関連付けられた DCO ボリュームを同じノードに接続する必要があります。ボリュームでボリュームレベルの I/O 転送が有効である場合、必要に応じて、DCO ボリュームの構成を変更できます。たとえば、データボリュームと関連付けられた DCO ボリュームの接続が一致しない場合、接続を一致させるために DCO ボリュームにミラーが追加されます。ただし、DCO ボリュームへのミラーの追加操作が失敗すると、ボリュームレベルの I/O 転送は有効になりません。

データボリュームまたは関連付けられた DCO ボリュームのプレックスが切断されているか無効になっている場合、ボリュームレベルの I/O 転送は自動的に無効になります。ボリュームにストレージを提供するノードがクラスタから離脱した場合、または基盤となるストレージで障害が発生した場合、ボリュームまたは関連付けられたプレックスが切断されているか無効になっている可能性があります。ただし、切断されたプレックスが再接続され、接続が復元されると、この機能は自動的に有効になります。

ボリュームレベルの I/O 転送を手動で無効にすると、この機能が有効になったときに追加された DCO ミラーが存在する場合は削除されます。

FSS 環境でボリュームレベルの I/O 転送を有効にするには

1 特定のボリュームでボリュームレベルの I/O 転送を有効にするには、次のコマンドを使用します。

```
# vxvol -g <dgname> set obj_ioship=on <volname>
```

2 vxkprint コマンドを使用し、有効にしたボリュームの kflag3 値を確認して、ボリュームレベルの I/O 転送が有効になっていることを確認します。

メモ: ボリュームレベルの I/O 転送を有効にすると、指定されたボリュームに対して、ioship_capable フラグと ioship_enable フラグが設定されます。

次に例を示します。

```
# /etc/vx/diag.d/vxkprint
...
# Group-Objects: (cnt: 10)
Mirrorvol vol1: rid=0.1032 assoc=0.0
update_tid=0.1063

  len=1024000 poolid=1 cdsrecover=0/0 (clean) ap_recover_seqno:
0
      ap_recover_seqno_done: 0

  kflag=(enabled|rdwr|fastresync|except-det-sparse|writeback|
         writecopy|unknown=0x200000)


kflag2=(instant-ready|init-drl|enable-drl|cache-implicit|iomode-req)


  kflag3=(write_ack_auto|ioship_capable|ioship_enable)
  sflag=()
  guid = {723f4aa2-b8ce-11e7-9ae4-db37e6453020}
  vvr_tag = 0
  proxy rid = 0.0  mediatype = hdd
  iocount = 0
  maxiops = 0
  volgrp =
```

FSS 環境でボリュームレベルの I/O 転送を無効にするには

1 特定のボリュームでボリュームレベルの I/O 転送を無効にするには、次のコマンドを使用します。

```
# vxvol -g <dgname> set obj_ioship=off <volname>
```

2 vxkprint コマンドを使用し、無効化されたボリュームの kflag3 値を確認して、ボリュームレベルの I/O 転送が無効になっていることを確認します。

メモ: ボリュームレベルの I/O 転送を無効にすると、そのボリュームに対して、ioship_capable フラグと ioship_enable フラグは設定されません。

次に例を示します。

```
# /etc/vx/diag.d/vxkprint
...
# Group-Objects: (cnt: 10)
Mirrorvol vol1: rid=0.1032 assoc=0.0
  update_tid=0.1063
  len=1024000 poolid=1 cdsrecover=0/0 (clean)  ap_recover_seqno:
 0
      ap_recover_seqno_done: 0

  kflag=(enabled|rdwr|fastresync|except-det-sparse|writeback|
         writecopy|unknown=0x200000)


kflag2=(instant-ready|init-drl|enable-drl|cache-implicit|iomode-req)


  kflag3=(write_ack_auto)
  sflag=()
  guid = {723f4aa2-b8ce-11e7-9ae4-db37e6453020}
  vvr_tag = 0
  proxy rid = 0.0  mediatype = hdd
  iocount = 0
```

メモ: ボリュームレベルの I/O 転送が自動的に無効になると、ioship_capable フラグのみが設定されます。

```
# /etc/vx/diag.d/vxkprint
...
kflag3=(write_ack_auto|ioship_capable)
```

ストレージ切断に対する CVM 耐障害性の制御

デフォルトでは、CVM を使用して別のノードを通じてすべてのディスクにアクセスできるノードがクラスタに結合できるように設定できます。同様に、共有ディスクグループのディスクにアクセスできるノードが 1 つでもある場合は、CVM は共有ディスクグループをインポートできます。この動作はチューニングパラメータの storage_connectivity が[非対称(asymmetric)]に設定された場合に起こります。

必要に応じて、ノードをクラスタに結合する条件として、ノードが共有ディスクグループ内のすべてのディスクにアクセスすることを要求するように CVM で設定できます。接続は共有ディスクグループのインポート前にも必要です。この動作はチューニングパラメータの storage_connectivity が[耐性がある(resilient)]に設定された場合に起こります。

ディスクグループのバージョンとクラスタプロトコルのバージョンは非対称的な動作をサポートするレベルに設定する必要があります。

ストレージの接続パラメータを耐障害性に設定するには

1 ストレージ切断に対する CVM 耐障害性の現在の設定を表示します。

```
# vxtune storage_connectivity
KEYWORD                         CURRENT-VALUE   DEFAULT-VALUE
storage_connectivity            asymmetric      asymmetric
```

2 出力に現在の値が非対称であることが示される場合、次のコマンドを実行して耐障害性の動作を有効化できます。

```
# vxtune storage_connectivity resilient
```

3 変更された設定を確認します。

```
# vxtune storage_connectivity
KEYWORD                         CURRENT-VALUE   DEFAULT-VALUE
storage_connectivity            resilient       asymmetric
```

共有ディスクグループでのクローンディスクの扱い方

ディスクのクローンを作成したり、重複したディスク ID が作成されるような方法でコピーを作成した場合、ディスクを VxVM（Veritas Volume Manager）にインポートするには特別なアクションを実行する必要があります。共有ディスクグループの場合と専用ディスクグループの場合の手順は同じです。ディスクをインポートする準備が整ったら、vxdg import コマンドに -s オプションを指定します。

```
# vxdg -s import diskgroup
```

p.1000 の「ハードウェアクローンディスクを含むディスクグループのインポート」を参照してください。

排他的起動権限を持つボリュームの作成

vxassist コマンドを使ってボリュームを作成する際に exclusive=on 属性を指定すると、そのボリュームを同時に起動できるノードをクラスタ内で 1 つに制限することができます。たとえば、ディスクグループ dskgrp 内にミラーボリューム volmir を作成し、排他的起動権限を設定するには、次のコマンドを使います。

```
# vxassist -g dskgrp make volmir 5g layout=mirror exclusive=on
```

また、同じノードによる複数のボリュームの起動もサポートされています。ボリュームを起動したノードがそのボリュームを完全に停止するまで、他のノードはそのボリュームを起動できません。

代わりに exclusive=off を指定すると、1 つのボリュームをクラスタ内の複数のノードで同時に起動できるようになります。これがデフォルトの動作です。

ボリュームへの排他的起動権限の設定

ボリューム上の排他的起動権限は、クラスタ内の任意のノードから設定できます。また、この属性を設定する際には、ボリュームを起動しているノードがクラスタ内に存在しないことを確認してください。

vxvol コマンドに exclusive=on 属性を指定すると、既存のボリュームを同時に起動できるノードをクラスタ内で 1 つに制限することができます。

たとえば、ディスクグループ dskgrp 内のボリューム volmir に排他的起動権限を設定するには、次のコマンドを使います。

```
# vxvol -g dskgrp set exclusive=on volmir
```

また、同じノードによる複数のボリュームの起動もサポートされています。ボリュームを起動したノードがそのボリュームを完全に停止するまで、他のノードはそのボリュームを起動できません。

代わりに exclusive=off を指定すると、1 つのボリュームをクラスタ内の複数のノードで同時に起動できるようになります。これがデフォルトの動作です。

クラスタプロトコルのバージョンの表示

次のコマンドは、ノード上で稼動しているクラスタプロトコルのバージョンを表示します。

```
# vxdctl list
```

このコマンドの出力は、次のようになります。

```
Volboot file
version: 3/1
seqno: 0.19
cluster protocol version: 160
hostid: giga
hostguid:  {2d7702ba-eba8-11e5-bf2d-6def043d7adc}
```

次のコマンドを使って既存のクラスタプロトコルバージョンを確認することもできます。

```
# vxdctl protocolversion
```

このコマンドの出力は、次のようになります。

```
Cluster running at protocol 160
```

サポートされているクラスタプロトコルのバージョン範囲の表示

次のコマンドは、ノードがサポートしている最大と最小のプロトコルバージョンおよび現在のプロトコルバージョンを表示します。

```
# vxdctl support
```

このコマンドの出力は、次のようになります。

```
Support information:
vxconfigd_vrsn:     39
dg_minimum:         20
dg_maximum:        220
kernel:             39
protocol_minimum:   90
protocol_maximum:  160
protocol_current:  160
```

また、次のコマンドを使って、最新の VxVM（Veritas Volume Manager）リリースがサポートしているクラスタプロトコルのバージョンの範囲を表示することができます。

```
# vxdctl protocolrange
```

このコマンドの出力は、次のようになります。

```
minprotoversion: 90, maxprotoversion: 160
```

共有ディスクグループ内のボリュームのリカバリ

vxrecover ユーティリティは、ディスク交換後のプレックスおよびボリュームのリカバリに使われます。ノードがクラスタから切断される際に、矛盾のある状態でいくつかのミラーが残されることがあります。vxrecover ユーティリティは、このようなボリュームのリカバリに使うことができます。vxrecover に -c オプションを指定すると、共有ディスクグループ内のすべてのボリュームのリカバリが行われます。vxconfigd デーモンは、必要に応じて自動的に、-c オプションを付けて vxrecover ユーティリティを呼び出します。

警告: vxrecover ユーティリティの実行中は、システム処理効率が多少低下することがあります。

クラスタパフォーマンスの統計の取得

vxstat ユーティリティは、指定されたオブジェクトについて統計を返します。クラスタ環境では、vxstat はクラスタ内のノードすべてから統計を収集します。統計は、要求されたオブジェクトに対する、すべてのノードからの総使用量を示します。ローカルオブジェクトが指定された場合は、ローカルな使用量が返されます。

次の形式のコマンドを使って、選択的に、ノードのサブセットを指定できます。

```
# vxstat -g diskgroup -n node[,node...]
```

ここで、*node* は CVM（Cluster Volume Manager）ノード ID 番号です。次のコマンドを使って、CVM ノード ID を確認できます。

```
# vxclustadm nidmap
```

複数のノードをカンマで区切りリストとして指定すると、vxstat ユーティリティはそのリストに含まれるノードの統計量の合計を表示します。

たとえば、ノード 2、ボリューム vol1 に対する統計を取得するには、次のコマンドを使います。

```
# vxstat -g diskgroup -n 2 vol1
```

このコマンドの出力は、次のようになります。

```
                OPERATIONS      BLOCKS            AVG TIME(ms)
TYP   NAME    READ    WRITE   READ      WRITE   READ    WRITE
vol   vol1    2421    0       600000    0       99.0    0.0
```

クラスタ全体の統計量を取得し表示するには、次のコマンドを使います。

```
# vxstat -b
```

すべてのノードに対する統計量が合計されます。たとえば、I/O 操作がノード 1 で 100 回実行され、ノード 2 で 200 回実行された場合、vxstat -b コマンドで表示される I/O 操作の回数は合計の 300 回になります。

スレーブノードからの CVM の管理

CVM（Cluster Volume Manager）では、クラスタのマスターノードで、CVM 共有ディスクグループのオブジェクト設定を変更する設定コマンドが実行される必要があります。設定変更の例としては、共有ディスクグループの作成、共有ディスクグループのインポート、共有ディスクグループのデポート、共有ディスクグループ内のボリュームまたはスナップショットの作成などがあります。

SFCFSHA の 5.1 Service Pack 1 リリースからは、共有ディスクグループを操作するほとんどの設定コマンドを、クラスタの任意のノードから実行できます。スレーブノードでコマンドを発行すると、CVM がスレーブノードからマスターノードにコマンドを送信します。その後、CVM はマスターノードでコマンドを実行します。通常の状況では、共有ディスクグループに対する設定変更コマンドは、マスターノードで発行することを推奨します。環境で必要な場合は、スレーブノードからこれらのコマンドを実行できます。

プライベートディスクグループを操作するコマンドは、マスターノードに送信されません。同様に、vxprint や vxdisk list のようなスレーブノードでローカルに動作するコマンドも、CVM は送信しません。

CVM は VCS（Cluster Server）の Group Membership Services and Atomic Broadcast（GAB）転送機構を使ってスレーブノードからマスターノードにコマンドを送信します。

マスターで実行されるコマンドをスレーブで実行すると、（スレーブノードでの）コマンド出力ではマスターノードに対応するオブジェクト名が表示されます。たとえば、コマンドではマスターノードのディスクアクセス名（daname）が表示されます。

vxtask や vxstat などのクエリーコマンドをスレーブノードから実行すると、スレーブノードでのコマンドの状態が表示されます。スレーブノードから最初に発行されてマスターノードで実行しているコマンドの状態は表示されません。

スレーブノードから最初に発行されて、CVM によってマスターで実行されるコマンドに対する、次のエラー処理に注意してください。

- スレーブノードまたはマスターノードの vxconfigd デーモンが失敗すれば、コマンドは終了します。マスター上のコマンドのインスタンスも終了します。コマンドが正常に

実行したかどうかを判定するには、vxprint コマンドを使って VxVM（Veritas Volume Manager）オブジェクトの状態を調べます。

- マスターがコマンドを実行している間に、コマンドを送信したスレーブノードまたはマスターノードがクラスタから切り離された場合、コマンドはマスターノードとスレーブノードの両方で終了します。コマンドが正常に実行したかどうかを判定するには、vxprint コマンドを使って VxVM オブジェクトの状態を調べます。

スレーブノードから CVM コマンドを発行する場合は、次の制限事項に注意してください。

- CVM プロトコルバージョンはクラスタ内のすべてのノードで少なくとも 100 である必要があります。
p.457 の「クラスタプロトコルのバージョンの表示」を参照してください。
- CVM がコマンドを実行するとき、CVM はマスターノードにあるデフォルトファイルの値を使います。あいまいさを避けるため、クラスタ内の各ノードのデフォルトファイルで同じ値を使うことを推奨します。
- CVM では、すべてのコマンドをスレーブノードで実行できるわけではありません。次のコマンドは、マスターノードでのみ発行する必要があります。
  - コントローラの名前を指定するコマンド。次に例を示します。

    ```
    # vxassist -g shareddg make sharedvol 20M ctlr:fscsi0
    ```

  - 共有ディスクグループと専用ディスクグループの両方を指定するコマンド。次に例を示します。

    ```
    # vxdg destroy privatedgshareddg
    ```

  - 引数としてデフォルトのファイルを含んでいるコマンド。次に例を示します。

    ```
    # vxassist -d defaults_file
    ```

  - VVR（Veritas Volume Replicator）コマンド。例: vxibc、vxrlink、vxrsync、vxrvg、vrport、vrstat、vradmin。
  - 共有ディスクグループに対して動作する vxdisk コマンドのオプション。

  p.1195 の「スレーブノードでの実行をサポートされる CVM コマンド」を参照してください。

Flexible Storage Sharing の管理

SFCFSHA を自動的にインストールすると、FSS（Flexible Storage Sharing）機能が有効になります。他のインストール手順は不要です。LLT、GAB、フェンシングは FSS を管理する前に設定する必要があります。フェンシングコーディネーションポイントは、SCSI-3 PR 対応共有ストレージまたは CP サーバーのどちらかに指定できます。

『Storage Foundation Cluster File System High Availability 設定/アップグレードガイド』を参照してください。

FSS 管理タスクには、FSS のディスクのエクスポート、ディスクグループでの FSS オプションの設定、ホストに接続したデバイスに直観的な名前を付けるためにホストの接頭辞を設定すること、エクスポートしたディスクとネットワーク共有ディスクの表示、FSS メモリ消費の調整(省略可能)があります。

p.202 の「Flexible Storage Sharing について」を参照してください。

p.461 の「Flexible Storage Sharing ディスクサポートについて」を参照してください。

Flexible Storage Sharing ディスクサポートについて

CVM では一意のディスクの ID(UDIDS)を使用して、クラスタのさまざまなノードにまたがるディスクを識別します。FSS は一意の ID を生成する機能を持つディスクだけをサポートします。

FSS は次のディスクをサポートします。

- ハードウェア互換性リスト(HCL)に登録済みのディスク:
  https://www.veritas.com/support/en_US/article.000126344
- `vxddladm addjbod` CLIを使用して指定された JBOD 定義があるディスク ご利用の環境での特定のディスクを識別する固有の方法を提供する仕様になっていることを確認してください。
- VPD ページ 0x83 照会データの IEEE 認定済 NAA ID を提供する SCSI-3 ディスク

`vxddladm checkfss` *`diskname`* コマンドを使用して、ディスクが必要な条件を満たすことを確認できます。条件を満たない場合、`vxdisk export` *`diskname`* コマンドを使用した FSS 設定へのディスクの追加は失敗します。

Flexible Storage Sharing ディスクグループのボリュームレイアウトについて

デフォルトでは、FSS 属性設定を持つディスクグループのボリュームがホスト間でミラー化されます。このデフォルトのレイアウトにより、いずれかのホストが利用不可になった場合でもデータを利用できます。関連付けられているインスタントデータ変更オブジェクト(DCO)ログボリュームもデフォルトで作成されます。

次のボリューム属性がデフォルトで想定されます。

- `mirror=host`
- `nmirror=2`
- `logtype=dco`

- ndcomirror=2
- dcoversion=30

ホストディスククラスを使用してボリュームを割り当てるホストを指定することができます。

p.466 の「ホストのディスククラスと割り当てストレージの使用」を参照してください。

従来は FSS 属性設定のないディスクグループには、デフォルトのボリュームレイアウトが連結されていました。ただし、vxassist コマンドの layout=concat オプションを明示的に使用して、FSS 属性設定を持つディスクグループの連結ボリュームを作成することを選択することができます。

デフォルトでは、ミラー化されたボリュームが各ホストに割り当てられます。この基準を満たすためにホスト固有のストレージが利用可能でない場合、ボリュームは従来のディスクグループと同様に、連結されたデフォルトのレイアウトとして外部ストレージに割り当てられます。

dm などの既存のディスククラスは FSS と共に使用することができます。ディスクアクセス名のホスト接頭辞は、ディスクが接続されているホストを示します。

たとえば、1 つのプレックスを持つボリュームをローカルディスク(disk1)に作成し、もう 1 つのプレックスを持つボリュームをリモートディスク(hostA_disk2)に作成することができます。ここで、リモートディスクのホストプレックス(hostA)は、クラスタ内の別のホストを示します。

```
# vxassist -g mydg make vol1 10g layout=mirror dm:disk1 dm:hostA_disk2
```

p.466 の「vxassist を使用したミラー化ボリュームの管理」を参照してください。

vxdisk list *accessname* コマンドを使用して、ディスクに接続されているホストに関する情報など、接続性の情報を表示することもできます。

ホスト接頭辞の設定

ホスト接頭辞により、直観的にディスクのエクスポート元 (ディスクを物理的に接続しているホストなど) がわかります。vxdctl コマンドはホスト接頭辞を設定します。ホスト接頭辞のインスタンスを設定すると、ディスクアクセス名の長さの制約を満たす代替ホスト識別子を設定できます。

メモ: アレイエンクロージャのディスクにはホスト接頭辞は表示されません。

DAS (Direct Attached Storage) ディスクの場合、ホスト接頭辞はディスク命名の競合を避けるためにディスク検出時にディスクに追加されます。

SAN またはファイバーチャネル(FC)を介して接続したアレイベースエンクロージャの場合には、同じタイプ(同じ製造元、同じモデル)の異なるエンクロージャを異なるロードに排他的に接続し、DAS タイプのトポロジーを作成できます。このような場合は、異なるノー

ドの異なるディスクに割り当てられる VxVM デバイス名が同じである可能性があります。このようなディスクをエクスポートして FSS 環境で使う場合は、名前に同じ混乱が起きる可能性があります。ただし、このような場合はエクスポートしたディスクに hostprefix は付きません。命名が競合した場合は、同じ名前のディスクの VxVM デバイス名にシリアル番号を追加すると解決します。ただし、簡単に識別でき、読みやすくするためには、エンクロージャの名前を変更することをお勧めします。

DAS ディスクとエンクロージャベースディスク間の命名動作は、次の理由で異なります。エンクロージャは CVM クラスタの特定のインスタンスにある 1 つのノードのみに接続できます。ただし、ノードの結合で別のインスタンスのエンクロージャに 2 つ(またはそれ以上)のノードを接続している可能性があります。クラスタに動的に参加したり離脱するノードでは、アレイエンクロージャへの接続も動的に変わる可能性があります。そのため、接続情報を使ってトポロジーが SAN または DAS のどちらであるかを判断したり、ホスト接頭辞を追加する必要があるかどうかを判断しても信頼性はありません。その結果、CVM はエンクロージャの接続に基づいて VxVM デバイスに hostprefix を追加しません。命名の競合が起きた場合には、シリアル番号を VxVM デバイス名に追加します。一方で、DAS ディスクは一度に 1 つのノードのみに接続するので、(命名の競合が起きてから対処する必要がなく)デフォルトで hostprefix を追加するので問題ありません。

デフォルトでは、Cluster Volume Manager(CVM)は Cluster Server(VCS)設定ファイルのホスト名をホスト接頭辞として使います。 `/etc/vx/volboot` ファイルのホスト ID が 15文字を超える場合は、短いホスト接頭辞を `vxdctl` を使って設定しないと、クラスタマネージャノード ID(CMID)を接頭辞として使います。

詳しくは、`vxdctl`(1M)マニュアルページを参照してください。

次のコマンドは、エラードメインとしてホストの論理名を設定または修正します。

```
# vxdctl set hostprefix=logicalname
```

エラードメインとして設定したホストの論理名を設定解除するには、次のコマンドを使います。

```
# vxdctl unset hostprefix
```

`vxdctl list` コマンドは、ホスト接頭辞に設定した論理名を表示します。

Flexible Storage Sharing のディスクのエクスポート

クラスタのすべてのノードで物理的に共有していないディスク(クラスタの 1 つ以上のノードに接続したディスク)を使うには、ディスクを最初に共有ネットワークにエクスポートする必要があります。デバイスをエクスポートすると、デバイスをクラスタのすべてのノードで利用できるようになります。`vxdisk` コマンドを使うと、1 つ以上のディスクを共有ネットワークにエクスポートまたは共有ネットワークからアンエクスポートできます。

メモ: 確実にデータを保護するには、FSS で PGR ベースのデータディスクフェンシングを使います。完全にローカルに接続されたディスクの場合は、ディスクのすべての I/O がディスクが直接接続されているノードを経由するため、フェンシングを FSS 内部で暗黙に扱います。したがって、SCSI3 PGR をサポートしないデバイスがフェンシングを使う FSS でサポートされます。複数のホストに接続しているが、SCSI3 PGR をサポートしないディスクの場合は、フェンシングを有効にする方法はないので、この設定はフェンシングが有効な場合はサポートされません。

次のコマンドは共有ネットワークに 1 つ以上のディスクをエクスポートします。

```
# vxdisk export accessname1accessname2
```

accessname1 と *accessname2* は、共有ネットワークにエクスポートするディスクのアクセス名です。

さらに、すべてのローカルディスクとすべての共有ディスク、またはすべてのローカル接続ディスクをエクスポートするには、それぞれ `-o alldisks` オプションと `-o local` オプションを使うことができます。

メモ: ブートディスク、不透過ディスク、インポートまたはデポートディスクグループのディスク部分、VxVM 以外のディスクはエクスポートされません。

反対に、次のコマンドで共有ネットワークから 1 つ以上のディスクをアンエクスポートします。

```
# vxdisk unexport accessname1accessname2
```

accessname1 と *accessname2* は、共有ネットワークからアンエクスポートするディスクのアクセス名です。

さらに、すべてのローカルディスクとすべての共有ディスク、またはすべてのローカル接続ディスクをアンエクスポートするには、それぞれ `-o alldisks` オプションと `-o local` オプションを使うことができます。

`vxdisksetup` と `vxdisk init` のコマンドを使って初期化するときにディスクをエクスポートすると、FSS にディスクを設定することもできます。

次のコマンドのいずれかを使って、共有ネットワークが有効なディスクを初期化します。

```
# vxdisksetup -i disk_address export

# vxdisk [-f] init accessname export
```

disk_address は、作動しているディスクに対応するデバイスです。*accessname* はディスクアドレスに関連するシステム固有の名前です。

ディスクをエクスポートした後で、vxdisk list と vxprint のコマンドを使ってネットワーク共有ディスクの一覧を表示し、コマンドを実行するノードにリモートアクセスするディスクを識別します。

p.468 の「エクスポートしたディスクとネットワーク共有ディスクグループの表示」を参照してください。

ディスクをエクスポートすると、ネットワーク共有ディスクと物理共有ディスクを組み合わせて共有ディスクグループを作成できます。ディスクグループにエクスポートされたディスクを追加する前に、Flexible Storage Sharing 属性を共有ディスクグループで設定する必要があります。

p.465 の「ディスクグループでの Flexible Storage Sharing 属性の設定」を参照してください。

ディスクグループでの Flexible Storage Sharing 属性の設定

エクスポートしたディスクをディスクグループに追加する前に、ディスクグループに FSS 属性を設定する必要があります。設定すると、エクスポートしたディスクをディスクグループに誤って追加するのを防ぎます。さらに、調整可能なパラメータ storage_connectivity を非対称的に設定する必要があります。FSS 属性は共有ディスクグループにのみ設定できます。

vxdg コマンドを実行すると、FSS 属性が有効なディスクグループを作成したり、既存のディスクグループに FSS 属性を設定できます。

メモ: FSS ディスクグループを作成または設定するには、ディスクグループバージョンを 190 以上に設定する必要があります。ディスクグループで FSS 属性を設定するとき、または既存のディスクグループを FSS ディスクグループとしてインポートするとき、ディスクグループバージョンをアップグレードする必要がある場合があります。

次のコマンドは、初期化時にディスクグループに FSS 属性を設定します。

```
# vxdg -s -o fss init diskgroup [medianame=] accessname
```

diskgroup はディスクグループ名、*medianame* は仮想マシンディスクに選択した管理名、*accessname* はデバイス名またはディスクアクセス名です。

次のコマンドは、ディスクグループに FSS 属性を設定します。

```
# vxdg -g diskgroup set fss=on
```

次のコマンドは、ディスクグループのインポート時にディスクグループに FSS 属性を設定します。

```
# vxdg -s -o fss=on import diskgroup
```

ディスクグループに FSS 属性を設定した後で、指定したディスクグループに vxdg list コマンドを実行すると、ローカルディスクをディスクグループに加えているクラスタのすべてのホストと、ディスクグループにディスクを追加した追加元ストレージエンクロージャの一覧を表示します。FSS 属性が有効なディスクグループを作成するか、または既存のディスクグループに FSS 属性を設定すると、ioship ポリシーがオンに設定され、ディスク切断ポリシーがローカル切断ポリシーに設定されます。

p.468 の「エクスポートしたディスクとネットワーク共有ディスクグループの表示」を参照してください。

ホストのディスククラスと割り当てストレージの使用

vxassist コマンドを実行すると、ストレージの割り当てにホストのディスククラスを使うことができます。ホストクラスはストレージの割り当て元ホストを指定します。ホストクラスは FSS に設定されているディスクグループにのみ適用されます。

ホストのディスククラスインスタンスは vxclustadm nidmap コマンドを使って表示できる Cluster Server（VCS）設定ファイルのホスト名と同じです。

ミラー化ボリュームは、次のように mirror=host パラメータを指定してホストのディスククラスを使うとホスト全体にミラーで作成できます。

```
# vxassist -g mydg make vol1 10g host:vm240v5 host:vm240v6

# vxassist -g mydg make vol1 10g mirror=host
```

vxassist を使用したミラー化ボリュームの管理

デフォルトでは、FSS 属性設定を持つディスクグループのボリュームがホスト間でミラー化されます。vxassist コマンドを使用して、混合メディアタイプにまたがって分散した、または内部ディスクと外部共有ディスク（SAN 接続型ディスクなど）の組み合わせを使用した、ミラー化ボリュームセットを作成することができます。

HDD および SSD ディスクでミラー化ボリュームを作成するには

1 HDD または SSD ディスクのボリュームを作成します。

```
# vxassist -g diskgroup make vol1 maxsize layout=concat init=none
```

ここで、*diskgroup* はディスクグループの名前です。HDD はデフォルトで選択されます。

2 SSD に基づいてプレックスを追加します。

```
# vxassist -g diskgroup mirror vol1 mediatype:ssd
```

3 ボリュームをアクティブ化します。

```
# vxvol -g diskgroup init active vol1
```

メモ: ミラー化ボリュームでは割り当ての制約セットが保持されないことがあります。

`init=none` オプションを使って SSD ディスクにミラー化ボリュームを作成した場合は、次のコマンドを使って DCO ボリュームに新しいミラーを手動で追加できます。

```
# vxassist -g diskgroup mirror volume-dcl diskname
```

diskgroup はディスクグループの名前、*diskname* は新しいデータミラーが追加されたホスト内のディスクの名前です。ベリタスはデータミラーが作成されたディスクの名前を指定することをお勧めします。

内部ディスクおよび外部物理共有ディスクを使用してミラーボリュームを作成するには

1 内部ディスクまたは外部物理共有ディスクのボリュームを作成します。

```
# vxassist -g diskgroup make vol1 maxsize layout=concat init=none

host:host1
```

ここで、*diskgroup* はディスクグループの名前です。

2 外部共有ディスクに基づいてプレックスを追加します。

```
# vxassist -g diskgroup mirror vol1 enclr:emc0
```

3 ボリュームをアクティブ化します。

```
# vxvol -g diskgroup init active vol1
```

メモ: ホストディスククラスを指定すると、指定されたホストに接続されている内部ストレージのボリュームが割り当てられます。

p.466 の「ホストのディスククラスと割り当てストレージの使用」を参照してください。

エクスポートしたディスクとネットワーク共有ディスクグループの表示

vxdisk list と vxprint のコマンドは、コマンドを実行するホストにリモート接続されたディスクを識別するためにネットワーク共有ディスクを一覧表示します。vxdisk list コマンドには、すべてのリモートディスクをフィルタで除外している場合にディスクを表示するオプションもあります。

エクスポートしたディスクを表示するには、次のように vxdisk list コマンドを使います。

```
# vxdisk list
DEVICE          TYPE          DISK     GROUP  STATUS
disk_01         auto:cdsdisk  -        -      online exported
disk_02         auto:cdsdisk  -        -      online exported
vm240v6_disk_01 auto:cdsdisk  -        -      online remote
vm240v6_disk_02 auto:cdsdisk  -        -      online remote
```

ディスク名にはディスクの接続先ホストを示す接頭辞が含まれます。たとえば、vm240v6_disk_01 ディスクの場合は vm240v6 がホスト接頭辞です。状態の exported フラグは、FSS にエクスポートしたディスクを示します。remote フラグはコマンドを実行するホストにローカル接続していないディスクを示します。

accessname 引数を指定した場合は、ディスクの接続情報は長い形式の出力で表示されます。この情報はコマンドを実行するノードが CVM クラスタに含まれる場合にのみ利用可能です。

vxdisk list コマンドの -o local オプションはすべてのリモートディスクをフィルタで除外します。

次に例を示します。

```
# vxdisk -o local list
DEVICE          TYPE          DISK     GROUP  STATUS
disk_01         auto:cdsdisk  -        -      online exported
disk_02         auto:cdsdisk  -        -      online
```

-o fullshared オプションは、すべてのアクティブノードにわたって共有されるすべてのディスクを表示します。

-o partialshared オプションは、部分的に共有されているすべてのディスクを表示します。部分的に共有されたディスクは複数のノードに接続されていますが、クラスタのすべてのアクティブノードには接続されていません。

ディスクグループのリモートディスクを表示するには、vxprint コマンドを使います。

```
# vxprint
Disk group: sdg

TY NAME                 ASSOC           KSTATE   LENGTH     PLOFFS  STATE     TUTIL0 PUTIL0
dg sdg                   sdg              -        -          -       -         -       -
dm disk_1                vm240v6_disk_1   -        2027264    -       REMOTE    -       -
dm disk_4                vm240v6_disk_4   -        2027264    -       REMOTE    -       -
dm disk_5                disk5            -        2027264    -       -         -       -
```

vxdg list コマンドは、ローカルディスクがディスクグループに属するクラスタのホストと、ディスクをディスクグループに追加した追加元ストレージエンクロージャを表示します。ローカルディスクがディスクグループに属するホストと、ディスクをディスクグループに追加した追加元ストレージエンクロージャが storage-sources フィールドに一覧表示されます。

このコマンドの出力例は、次のとおりです。

```
Group:     mydg
dgid:      1343697721.24.vm240v5
import-id: 33792.24
flags:     shared cds
version:   190
alignment: 8192 (bytes)
detach-policy:local
ioship: on
fss: on
local-activation: shared-write
storage-sources: vm240v5 vm240v6 emc0
```

FSS 環境でのメモリとパフォーマンスについての LLT のチューニング

RDMA（Remote Direct Memory Access）環境で、LLT 設定ファイルのバッファプールメモリを割り当てるとネットワーク上で I/O を行うメモリの消費を制限できます。LLT_BUFPOOL_MAXMEM チューニングパラメータを使うと、事前割当てが可能な最小メモリ量と、LLT バッファプールに割り当て可能な最大メモリ量を指定できます。このバッファプールを使って、LLT クライアントに配信される RDMA 操作とパケット割り当てにメモリを割り当てます。デフォルト値は 4 GB、最小値は 1 GB、最大値は 10 GB です。GB で値を指定する必要があります。

チューニングパラメータと LLT 設定ファイルについて詳しくは、『Cluster Server 設定およびアップグレードガイド』の付録「FSS 環境でのメモリとパフォーマンスについての LLT のチューニング」を参照してください。

ODM の管理

この項では、次の ODM（Oracle Disk Manager）管理タスクの手順について説明します。

- 「ODM ポートの確認」
- 「ODM の起動」

ODM の管理中に問題が発生した場合について詳しくは、トラブルシューティングの項を参照してください。

ODM ポートの確認

ODM（Oracle Disk Manager）を SFCFSHA で有効にすることをお勧めします。次のコマンドを実行し、ODM が実行されていることを確認します。

```
# gabconfig -a | grep "Port d"
```

ODM の起動

次の手順では ODM (Oracle Disk Manager) を起動する方法について説明します。

ODM を起動するには

◆ 次のコマンドを実行します。

RHEL 7、RHEL 8、SLES 12、SLES 15、およびサポート対象の RHEL 互換配布の場合:

```
# systemctl start vxodm
```

以前のバージョンの RHEL、SLES、およびサポート対象の RHEL 互換配布の場合:

```
# /etc/init.d/vxodm start
```

I/O フェンシングの管理について

I/O フェンシング機能には、`VRTSvxfen` RPMを通じて利用できる次のユーティリティが用意されています。

vxfentsthdw	I/O フェンシング用のディスクの SCSI-3 機能をテストします p.471 の「vxfentsthdw ユーティリティについて」を参照してください。

vxfenconfig	I/O フェンシングの設定と設定解除を行います vxfen ドライバが使うコーディネーションポイントを一覧表示します。
vxfenadm	I/O フェンシング操作に関する情報を表示し、I/O フェンシング用の SCSI-3 ディスクの登録と予約を管理します p.480 の「vxfenadm ユーティリティについて」を参照してください。
vxfenclearpre	SCSI-3 の登録と予約をディスクから削除します p.485 の「vxfenclearpre ユーティリティについて」を参照してください。
vxfenswap	I/O フェンシングを停止せずにコーディネーションポイントを交換します p.488 の「vxfenswap ユーティリティについて」を参照してください。
vxfendisk	ディスクグループ内のディスクのパスの一覧を生成します。このユーティリティを使うには、VxVM（Veritas Volume Manager）がインストールおよび設定されている必要があります。

I/O フェンシングのコマンドは /opt/VRTS/bin フォルダに存在します。PATH 環境変数にこのフォルダのパスを追加したことを確かめてください。

コマンドについて詳しくは、対応するマニュアルページを参照してください。

vxfentsthdw ユーティリティについて

共有ストレージアレイで、データをサポートする SCSI-3 Persistent Reservation と I/O フェンシングが使われているかどうかを確認するには、vxfentsthdw ユーティリティを使います。I/O フェンシングの設定時に、テストユーティリティを使って 1 つのディスクをテストします。このユーティリティには、ストレージデバイスを他の設定でテストするのに適している可能性の高い他のオプションが備えられています。また、コーディネータディスクグループもテストする必要があります。

I/O フェンシングを設定するには、『Storage Foundation Cluster File System High Availability 設定およびアップグレードガイド』を参照してください。

このユーティリティは、クラスタ内の 1 つのシステムから実行できます。このユーティリティを使って、指定したディスクへの SCSI-3 登録の設定と確認、ディスクへの Persistent Reservation の設定と確認、ディスクへのデータの書き込みと読み取り、ディスクからの登録の削除などを行って、データ用として使用するストレージをテストします。

vxfentsthdw（1M）マニュアルページも参照してください。

vxfentsthdw ユーティリティを使うための一般的なガイドライン

vxfentsthdw ユーティリティの使用に関する以下のガイドラインを確認します。

- このユーティリティを使うには、2 つのシステムが共有ストレージに接続されている必要があります。

  注意: `-r` オプションが使われていない場合、ディスク上のデータはテスト時に上書きされて破棄されます。

- 2 つのノードには SSH 通信（デフォルト）または rsh 通信が確立されている必要があります。rsh を使う場合は、`vxfentsthdw` ユーティリティに `-n` オプションを指定して起動します。
  プロセスのテストが完了したら、通信用の権限を削除し、パブリックネットワーク接続をリストアできます。
- テスト時に 2 つのシステムが同じディスクに接続されていることを確認するには、`vxfenadm -i diskpath` コマンドを使って、ディスクのシリアル番号を確認します。p.484 の「ノードが同じディスクを参照することを確認」を参照してください。
- 複数のディスクを持つディスクアレイの場合、ディスクグループを作成する前に `-m` オプションを使っていくつかのディスクのサンプルを取得したり、`-g` オプションを使ってすべてのディスクをテストしたりします。
- ユーティリティは、ディスクが I/O フェンシングに使用できることを、次のようなメッセージで示します。

  ```
  The disk /dev/sdx is ready to be configured for
  I/O Fencing on node sys1
  ```

  ユーティリティが、ディスクが準備が整っていることを表すメッセージを表示しない場合、検証は失敗しました。
- `-o` オプションはディスクサイズに関連するエラーと他のテストでの設定処理を上書きしますが、サイズがサポート対象のサイズより小さい場合、ディスクが正しく設定されないことがあります。データディスクのサポート対象のディスクサイズは 256 MB で、コーディネータディスクは 128 MB です。
- テストを行うディスクに SCSI-3 登録キーがすでに存在する場合、テストが続行される前に警告が出力されます。

vxfentsthdw コマンドのオプションについて

表 15-3 には、ストレージデバイスのテストのためにユーティリティが提供する方法を示しています。

表 15-3 vxfentsthdw のオプション

vxfentsthdw のオプション	説明	用途
-n	ユーティリティは rsh を通信に使います。	通信に rsh を使うときに使います。
-r	非破壊テスト。ディスクの SCSI-3 Persistent Reservation テストが非破壊の方法で行われます。つまり、読み取りのみがテストされ、書き込みはテストされません。-m、-f、-g オプションとともに使えます。	非破壊テスト時に使います。 p.476 の「-r オプションを使った、ディスク上での非破壊テストの実行」を参照してください。
-t	SCSI-3 reservation における SCSI TEST UNIT (TUR) コマンドの戻り値をテストします。TUR テストがエラーになると、警告が出力されます。	TUR テストを実行する場合。
-d	DMP (Dynamic Multi-Pathing) デバイスを使います。 -c または -g のオプションとともに使えます。	デフォルトでは、vxfentsthdw スクリプトはディスクグループのディスク用の DMP パスを選択します。スクリプトがディスクグループに RAW パスを使うようにするには、-w オプションを使います。
-w	RAW デバイスを使います。 -c または -g のオプションとともに使えます。	-w オプションを使うと、vxfentsthdw スクリプトはディスクグループ内のディスクのオペレーティングシステムを選択します。デフォルトでは、スクリプトは -d オプションを使ってディスクグループ内のディスクの DMP パスを選択します。
-c	ユーティリティはコーディネータディスクグループをテストします。システムおよびデバイスの指定を求めるプロンプトを表示し、成功または失敗を報告します。	コーディネータディスクグループのディスクのテスト。 p.474 の「vxfentsthdw の -c オプションを使ったコーディネータディスクグループのテスト」を参照してください。

vxfentsthdw のオプション	説明	用途
-m	ユーティリティを手動により対話式で実行します。システムおよびデバイスの指定を求めるプロンプトを表示し、成功または失敗を報告します。 -r または -t のオプションとともに使えます。-m はデフォルトオプションです。	いくつかのディスクのテスト、または比較的大きいアレイのディスクのサンプリング。 p.476 の「vxfentsthdw -m オプションを使った、共有ディスクのテスト」を参照してください。
-f *filename*	ユーティリティは、テキストファイルに指定されているシステムとデバイスの組み合わせをテストします。 -r または -t のオプションとともに使えます。	複数のディスクのテスト。 p.478 の「vxfentsthdw -f オプションを使った、ファイルにリストされた共有ディスクのテスト」を参照してください。
-g *disk_group*	ユーティリティは、特定のディスクグループ内のディスクデバイスをすべてテストします。 -r または -t のオプションとともに使えます。	多数のディスクおよびディスクアレイのテスト。テスト用のディスクグループを一時的に作成し、テスト後に破壊(グループ解除)することができます。 p.479 の「vxfentsthdw -g オプションを使った、ディスクグループ内の全ディスクのテスト」を参照してください。
-o	ユーティリティはディスクサイズに関連するエラーを上書きします。	ディスクの SCSI-3 Reservation 準拠に関するテストのために、ディスクサイズに関連するエラーを上書きします。

vxfentsthdw の -c オプションを使ったコーディネータディスクグループのテスト

vxfentsthdw ユーティリティを使って、ディスクが I/O フェンシングをサポートするように設定されていることを確認します。この手順では、vxfentsthdw ユーティリティを使って各ノードから一度に 1 つずつ、3 つのディスクをテストします。

たとえば、この項の手順では、次のディスクを使います。

- ノード sys1 では、ディスクが /dev/sdg、/dev/sdh、/dev/sdi として認識されます。
- ノード sys2 では、同じディスクが /dev/sdx、/dev/sdy、/dev/sdz として認識されます。

メモ: vxfentsthdw ユーティリティを使ってコーディネータディスクグループをテストするには、このユーティリティでコーディネータディスクグループの vxfencoorddg が 2 つのノードからアクセスできるようにする必要があります。

vxfentsthdw -c を使ってコーディネータディスクグループをテストするには

1 vxfentsthdw コマンドに -c オプションを指定して実行します。次に例を示します。

```
# vxfentsthdw -c vxfencoorddg
```

2 コーディネータディスクをテストするために使うノードを入力します。

```
Enter the first node of the cluster: sys1
Enter the second node of the cluster: sys2
```

3 両ノードのコーディネータディスクグループの全ディスクについてテストプロセスの出力を確認します。各ディスクについて次のような出力が表示される必要があります。

```
ALL tests on the disk /dev/sdg have PASSED.
The disk is now ready to be configured for I/O Fencing on node
sys1 as a COORDINATOR DISK.

ALL tests on the disk /dev/sdx have PASSED.
The disk is now ready to be configured for I/O Fencing on node
sys2 as a COORDINATOR DISK.
```

4 ディスクグループ内のすべてのディスクをテストすると、vxfencoorddg ディスクグループを使えるようになります。

障害のあるディスクの削除と交換

検証に失敗したディスクがコーディネータディスクグループ内にある場合、障害の発生したディスクまたは LUN を vxfencoorddg ディスクグループから削除し、他のディスクに置き換えて、ディスクグループを再テストします。

障害の発生したディスクを削除して置き換えるには

1 vxdiskadm ユーティリティを使って、障害のあるディスクをディスクグループから削除します。

詳しくは、『 Storage Foundation 管理者ガイド』を参照してください。

2 新しいディスクをノードに追加し、初期化して、コーディネータディスクグループに追加します。

I/O フェンシング用ディスクを初期化して、コーディネータディスクグループを設定する手順については、『Storage Foundation Cluster File System High Availability 設定およびアップグレードガイド』を参照してください。

必要に応じて、ディスクグループを起動します。

ディスクグループの起動方法について詳しくは、『 Storage Foundation Cluster File System High Availability 管理者ガイド』を参照してください。

3 ディスクグループを再度テストします。

p.474 の「vxfentsthdw の -c オプションを使ったコーディネータディスクグループのテスト」を参照してください。

-r オプションを使った、ディスク上での非破壊テストの実行

データを保持したい場合、ディスクデバイス上で非破壊テストを実行できます。

ディスクで非破壊テストを実行するには

◆ 保存しておきたいデータが含まれたディスクデバイスをテストするには、-m、-f、-g オプションのいずれかとともに -r オプションを使うことができます。

たとえば、-m オプションと -r オプションを使うには、ユーティリティを次のように実行できます。

```
# vxfentsthdw -rm
```

-r オプション付きで呼び出された場合、ユーティリティはディスクの書き込みテストを実行しません。したがってこの場合、ディスクは、通常の使用状況について完全にはテストされません。

vxfentsthdw -m オプションを使った、共有ディスクのテスト

共有ディスクをテストする手順を確認します。デフォルトでは、ユーティリティは -m オプションを使います。

この手順では、/dev/sdx ディスクを使います。

ユーティリティが、ディスクが準備が整っていることを表すメッセージを表示しない場合、検証は失敗しました。検証が失敗している場合、ディスクアレイの設定が不適切な可能性があります。不良ディスクも不適切な設定の原因となる場合があります。

失敗の原因が不良ディスクである場合、そのディスクを取り除いて交換します。
vxfentsthdw ユーティリティは、ディスクが I/O フェンシング用として使用できることを、次のようなメッセージで示します。

```
The disk /dev/sdx is ready to be configured for
I/O Fencing on node sys1
```

メモ: A/P アレイの場合、アクティブな有効パスでのみ vxfentsthdw コマンドを実行してください。

vxfentsthdw スクリプトを使ってディスクをテストするには

1. システム間の通信が適切に機能していることを確認します。
2. 1 つのノードからユーティリティを開始します。

   ```
   # vxfentsthdw [-n]
   ```

3. テストを実行するとディスクのデータが上書きされることを示す説明と警告を確認したら、処理の続行を確認して、ノード名を入力します。

   ```
   ******** WARNING!!!!!!!! ********
   THIS UTILITY WILL DESTROY THE DATA ON THE DISK!!

   Do you still want to continue : [y/n] (default: n) y
   Enter the first node of the cluster: sys1
   Enter the second node of the cluster: sys2
   ```

4 確認するディスク名を入力します。各ノードで、ディスクは同じ名前で認識されている場合があります。

```
Enter the disk name to be checked for SCSI-3 PGR on node
sys1 in the format:
        for dmp: /dev/vx/rdmp/sdx
        for raw: /dev/sdx
Make sure it's the same disk as seen by nodes sys1 and sys2
/dev/sdr

Enter the disk name to be checked for SCSI-3 PGR on node
sys2 in the format:
        for dmp: /dev/vx/rdmp/sdx
        for raw: /dev/sdx
Make sure it's the same disk as seen by nodes sys1 and sys2
/dev/sdr
```

ディスクのシリアル番号が同じでない場合、テストは停止します。

5 ユーティリティが検査を実行してその活動が報告されたら、出力を確認します。

6 各ノードのディスクが I/O フェンシングを実行できる状態であれば、ユーティリティは正常終了を報告します。

```
ALL tests on the disk /dev/sdx have PASSED
The disk is now ready to be configured for I/O Fencing on node
sys1
...
Removing test keys and temporary files, if any ...
.
.
```

7 検証するディスクごとに vxfentsthdw ユーティリティを実行します。

vxfentsthdw -f オプションを使った、ファイルにリストされた共有ディスクのテスト

-f オプションを使って、テキストファイルに記述されたディスクをテストします。次の手順例を確認してください。

ファイルに指定された共有ディスクをテストするには

1 次のように、システム sys1 と sys2 により共有される 2 つのディスクをテストするためのテキストファイル disks_test を作成します。

```
sys1 /dev/sdz sys2 /dev/sdy
sys1 /dev/sdu sys2 /dev/sdw
```

ここで、最初のディスクは 1 行目に記述され、sys1 では /dev/sdz、sys2 では /dev/sdy として参照されます。もう一つのディスクは 2 行目に記述され、sys1 では /dev/sdu、sys2 では /dev/sdw として参照されます。一般には、ディスクのリストはこの例よりも長くなります。

2 ディスクをテストするには、次のコマンドを入力します。

```
# vxfentsthdw -f disks_test
```

ユーティリティはテスト結果を一度に 1 ディスクずつ報告します。-m オプションの場合と同じです。

vxfentsthdw -g オプションを使った、ディスクグループ内の全ディスクのテスト

-g オプションを使って、ディスクグループ内のすべてのディスクをテストします。たとえば、ディスクアレイのすべてのディスクで構成される一時ディスクグループを作成してテストできます。

メモ: テストするディスクグループを共有としてインポートしないでください。つまり、-s オプションは vxdg import コマンドとともに使わないでください。

テストが終わったら、このディスクグループを破棄し、必要に応じてディスクをディスクグループに入れてください。

ディスクグループ内のすべてのディスクをテストするには

1 テストするディスク用のディスクグループを作成します。

2 次のコマンドを実行して、ディスクグループ test_disks_dg をテストします。

```
# vxfentsthdw -g test_disks_dg
```

ユーティリティは、一度に 1 つのテスト結果を報告します。

既存のキーによるディスクのテスト

コーディネータディスクに既存のキーが検出されると、次のようなメッセージが表示されます。

```
There are Veritas I/O fencing keys on the disk. Please make sure
that I/O fencing is shut down on all nodes of the cluster before
continuing.

         ******** WARNING!!!!!!!! ********

THIS SCRIPT CAN ONLY BE USED IF THERE ARE NO OTHER ACTIVE NODES
IN THE CLUSTER!  VERIFY ALL OTHER NODES ARE POWERED OFF OR
INCAPABLE OF ACCESSING SHARED STORAGE.

If this is not the case, data corruption will result.

Do you still want to continue : [y/n] (default: n) y
```

処理を継続する前に、ユーティリティによって警告が表示されます。I/O フェンシングがまだ設定されていない場合は、処理を継続することができます。

vxfenadm ユーティリティについて

管理者は、vxfenadm コマンドを使って、フェンシング設定のトラブルシューティングとテストを行うことができます。

管理者が使うコマンドのオプションは次のとおりです。

-s	ディスクのキーを読み込み、数字、文字、およびノード形式でキーを表示します **メモ:** -g と -G オプションは推奨されていません。-s オプションを使ってください。
-i	SCSI 照会情報をデバイスから読み取ります
-m	ディスクに登録します
-n	ディスクへの予約
-p	他のシステムによって行われた登録を削除します
-r	予約を読み取ります
-x	登録を削除します

コマンドオプションについて詳しくは、vxfenadm(1M)マニュアルページを参照してください。

I/O フェンシング登録キーの形式について

vxfen ドライバがデータディスクとコーディネータディスクで登録するキーは 8 つのバイトから成っています。キーの形式はコーディネータディスクとデータディスクで異なります。

コーディネータディスクのキーの形式は次のとおりです。

バイト	0	1	2	3	4	5	6	7
値	V	F	clD 0x	clD 0x	clD 0x	clD 0x	nlD 0x	nlD 0x

各オプションの説明

- VF はキーの名前空間を取得する一意の識別子です（2 バイトを消費します）
- clD 0x は 16 進の LLT クラスタ ID です（4 バイトを消費します）
- nlD 0x は 16 進の LLT ノード ID です（2 バイトを消費します）

vxfen ドライバは I/O フェンシングの両方の sybase モードでこのキー形式を使います。

VCS でフェールオーバーディスクグループとして設定されるデータディスクのキー形式は次のとおりです。

バイト	0	1	2	3	4	5	6	7
値	A+nID	V	C	S				

ここで、nID は LLT ノード ID です

たとえば、ノード ID が 1 の場合、最初のバイトの値は B（'A' + 1 = B）です。

CVM（Cluster Volume Manager）配下で並列ディスクグループとして設定されるデータディスクのキー形式は次のとおりです。

バイト	0	1	2	3	4	5	6	7
値	A+nID	P	G	R	DGcount	DGcount	DGcount	DGcount

ここで、DGcount は設定内のディスクグループの数です（4 バイトを消費します）。

デフォルトでは、CVM は各ディスクグループに対して一意のフェンシングキーを使います。ただし、一部のアレイでは、登録できる一意のキーの合計数に制限があります。この場合は、`same_key_for_alldgs` チューニングパラメータを使ってデフォルトの動作を変更できます。このパラメータのデフォルト値は `off` です。設定において一意のキーの合計数に関するストレージアレイの制限に達した場合は、`vxdefault` コマンドを次のように使って値を `on` に変更できます。

```
# vxdefault set same_key_for_alldgs on
# vxdefault list
```

```
KEYWORD               CURRENT-VALUE    DEFAULT-VALUE
...
same_key_for_alldgs   on               off
...
```

チューニングパラメータを on に変更すると、ディスクグループのインポート時または作成時に CVM が生成するすべての後続キーは、最後の 4 バイトとして「0000」を持ちます（DGcount は 0）。same_key_for_alldgs チューニングパラメータの変更された値を有効にするには、インポート済みのすべてのディスクグループをデポートして再インポートする必要があります。

I/O フェンシング登録キーの表示

vxfenadm コマンドを使ってディスクに現在割り当てられているキーを表示できます。

次の手順の *disk_7*、*disk_8*、*disk_9* などの変数は設定のディスク名を表します。

I/O フェンシング登録キーを表示するには

1 ディスクのキーを表示するには、次のコマンドを実行します。

```
# vxfenadm -s disk_name
```

次に例を示します。

- ノード ID 1 を含むシステムからコーディネータディスク /dev/sdx のキーを表示するには、次のコマンドを入力します。

```
# vxfenadm -s /dev/sdx
key[1]:
  [Numeric Format]:  86,70,68,69,69,68,48,48
  [Character Format]: VFDEED00
* [Node Format]: Cluster ID: 57069 Node ID: 0 Node Name: sys1
```

vxfenadm の -s オプションでは、8 バイトからなるキー値が 3 つの形式で表示されます。数字形式では、

  - 最初の 2 つのバイトが識別子 VF を表し、ASCII 値 86、70 を含んでいます。
  - 次の 4 つのバイトは 16 進でエンコードされたクラスタ ID 57069（0xDEED）の ASCII 値 68、69、69、68 を含んでいます。
  - 残りのバイトはノード ID 0（0x00）の ASCII 値を 48、48 を含んでいます。ノード ID 1 は 01 で、ノード ID 10 は 0A です。

ノード形式の前のアスタリスクは、LLT が設定されて動作しているクラスタのノードから vxfenadm コマンドが実行されることを示します。

- CVM 並列ディスクグループのキーを表示するには:

```
# vxfenadm -s /dev/vx/rdmp/disk_7

Reading SCSI Registration Keys...

Device Name: /dev/vx/rdmp/disk_7
Total Number Of Keys: 1
key[0]:
  [Numeric Format]:  66,80,71,82,48,48,48,49
  [Character Format]: BPGR0001
  [Node Format]: Cluster ID: unknown  Node ID: 1 Node Name:
sys2
```

- VCS（Cluster Server）フェールオーバーディスクグループのキーを表示するには

```
# vxfenadm -s /dev/vx/rdmp/disk_8

Reading SCSI Registration Keys...

Device Name: /dev/vx/rdmp/disk_8
Total Number Of Keys: 1
key[0]:
  [Numeric Format]:  65,86,67,83,0,0,0,0
  [Character Format]: AVCS
  [Node Format]: Cluster ID: unknown  Node ID: 0 Node Name:
sys1
```

2 ディスクファイルで指定されたすべてのディスクで登録されているキーを表示するには:

```
# vxfenadm -s all -f disk_filename
```

次に例を示します。

コーディネータディスクのすべてのキーを表示するには:

```
# vxfenadm -s all -f /etc/vxfentab

Device Name: /dev/vx/rdmp/disk_9
Total Number Of Keys: 2
key[0]:
[Numeric Format]:  86,70,70,68,57,52,48,49
[Character Format]: VFFD9401
* [Node Format]: Cluster ID: 64916 Node ID: 1   Node Name: sys2
key[1]:
```

```
[Numeric Format]:  86,70,70,68,57,52,48,48
[Character Format]: VFFD9400
* [Node Format]: Cluster ID: 64916 Node ID: 0   Node Name: sys1
```

クラスタ ID は `lltstat -C` コマンド、ノード ID は `lltstat -N` コマンドを使って確認できます。次に例を示します。

```
# lltstat -C
57069
```

ディスクに特定のクラスタに属さないキーがある場合、vxfenadm コマンドはノード ID のノード名を参照できないため、ノード名を不明として出力します。次に例を示します。

```
Device Name: /dev/vx/rdmp/disk_7
Total Number Of Keys: 1
key[0]:
  [Numeric Format]:  86,70,45,45,45,45,48,49
  [Character Format]: VF----01
  [Node Format]: Cluster ID: unknown  Node ID: 1  Node Name: sys2
```

任意形式のキーを持つディスクについては、vxfenadm コマンドはすべてのフィールドを不明として出力します。次に例を示します。

```
  [Numeric Format]:  65,66,67,68,49,50,51,45
  [Character Format]: ABCD123-
  [Node Format]: Cluster ID: unknown Node ID: unknown
         Node Name: unknown
```

ノードが同じディスクを参照することを確認

ディスク(LUN)が SCSI-3 Persistent Reservation をサポートするどうかを確認するには、2 つのノードが同じディスクに同時アクセスできる必要があります。共有ディスクの名前は各ノードで異なる可能性があるため、シリアル番号をチェックしてディスクの ID を確認します。vxfenadm コマンドと `-i` オプションを使って、LUN へのパスすべてで、LUN に対して同じシリアル番号が返されることを確認します。

たとえば、ある EMC ディスクにノード A の `/dev/sdr` パスとノード B の `/dev/sdt` パスがアクセス可能であるとします。

ノードが同じディスクを参照することを確認するには

1 データ用の共有ストレージが、Storage Foundation Cluster File System High Availability がインストールされている 2 つのノードと接続されていることを確認します。

2 ノード A で、次のコマンドを実行します。

```
# vxfenadm -i /dev/sdr

Vendor id       : EMC
Product id      : SYMMETRIX
Revision        : 5567
Serial Number   : 42031000a
```

/dev/sdt パスを使ってノード B で同じコマンドを実行したときに、同じシリアル番号情報が表示される必要があります。

Hitachi Data Systems など別の製造元のディスクでは、次のように、出力が異なる場合があります。

```
# vxfenadm -i /dev/sdt

Vendor id       : HITACHI
Product id      : OPEN-3
Revision        : 0117
Serial Number   : 0401EB6F0002
```

詳しくは、vxfenadm(1M)のマニュアルページを参照してください。

vxfenclearpre ユーティリティについて

vxfenclearpre ユーティリティを使って、ディスク上およびコーディネーションポイントサーバー上の SCSI-3 の登録と予約を削除できます。

p.486 の「すでに存在するキーの削除」を参照してください。

このユーティリティはサーバー型のフェンシングもサポートするようになりました。vxfenclearpre ユーティリティを使用して現在のクラスタの CP サーバー(コーディネーションポイントサーバー)から登録を削除できます。ユーティリティを実行するローカルノードでは、/etc/vx/.uuids ディレクトリの clusuuid ファイルに現在のクラスタの UUID が含まれている必要があります。対象クラスタの UUID ファイルがローカルノードにない場合、ユーティリティはコーディネーションポイントサーバーから登録を削除しません。

メモ: ユーティリティを使用してコマンドで指定した任意のクラスタの一連のコーディネータディスクから登録キーおよび登録(予約)を削除することができますが、CP サーバーからは現在のクラスタの登録しか消去できません。また、ユーティリティがコーディネーションポイントサーバーによって使用されている IP アドレスとネットワーク接続を確立しようとするため、コーディネーションポイントサーバーの登録を消去するのに時間がかかる場合があります。この遅延は、IP アドレスに到達できないか、IP アドレスが正しくない場合に、ネットワークの問題により発生する可能性があります。

vxfenclearpre ユーティリティの使用上発生した問題については、ログファイル /var/VRTSvcs/log/vxfen/vxfen.log を参照することができます。

すでに存在するキーの削除

スプリットブレイン状態が起きた場合は、vxfenclearpre ユーティリティを使って CP サーバー、SCSI-3 登録、予約をコーディネータディスク、コーディネーションポイントサーバーからだけでなくすべての共有ディスクグループのデータディスクから削除します。

この手順を使って、別のノードの登録キーと予約キー、共有ディスクの他のノード、CP サーバーを削除することもできます。

スプリットブレイン後にキーをクリアするには

1. すべてのノード上で VCS を停止します。

```
# hastop -all
```

2. すべてのノードでポート h が閉じていることを確認します。各ノードで次のコマンドを実行してポート h が閉じることを検証します。

```
# gabconfig -a
```

ポート h が出力に表示されていてはなりません。

3. すべてのノードで I/O フェンシングを停止します。ノードごとに次のコマンドを入力します。

```
# systemctl stop vxfen
```

4. 共有ストレージにアクセス可能な VCS 制御の範囲外で実行されるアプリケーションがある場合、共有ストレージにアクセス可能なクラスタ内の他のすべてのノードをシャットダウンします。これによりデータの破損を防止します。

5. vxfenclearpre スクリプトを開始します。

```
# /opt/VRTSvcs/vxfen/bin/vxfenclearpre
```

6 スクリプトの開始メッセージと警告を確認します。このとき、スクリプトの実行を選択できます。

```
Do you still want to continue: [y/n] (default : n) y
```

ノードがディスク/LUN から削除された場合、クラスタ内の 1 つのノードのコンソールに次のような情報メッセージが表示される場合があります。これらの情報メッセージを無視できます。

```
<date> <system name> scsi: WARNING: /sbus@3,0/lpfs@0,0/
sd@0,1(sd91):
<date> <system name> Error for Command: <undecoded
cmd 0x5f> Error Level: Informational
<date> <system name> scsi: Requested Block: 0 Error Block 0
<date> <system name> scsi: Vendor: <vendor> Serial Number:
0400759B006E
<date> <system name> scsi: Sense Key: Unit Attention
<date> <system name> scsi: ASC: 0x2a (<vendor unique code
0x2a>), ASCQ: 0x4, FRU: 0x0
```

スクリプトはディスクをクリーンアップし、次のステータスメッセージを表示します。

```
Cleaning up the coordinator disks...

Cleared keys from n out of n disks,
where n is the total number of disks.

Successfully removed SCSI-3 persistent registrations
from the coordinator disks.

Cleaning up the Coordination Point Servers...

...................
[10.209.80.194]:50001: Cleared all registrations
[10.209.75.118]:443: Cleared all registrations

Successfully removed registrations from the Coordination Point
Servers.

Cleaning up the data disks for all shared disk groups ...

Successfully removed SCSI-3 persistent registration and
reservations from the shared data disks.
```

```
See the log file /var/VRTSvcs/log/vxfen/vxfen.log

You can retry starting fencing module. In order to restart the
whole
product, you might want to reboot the system.
```

7 すべてのノードでフェンシングモジュールを開始します。

```
# systemctl start vxfen
```

8 すべてのノードで VCS を起動します。

```
# hastart
```

vxfenswap ユーティリティについて

vxfenswap ユーティリティを使うと、オンライン状態のクラスタ内にあるコーディネータポイントを追加、削除、交換できます。このユーティリティは、新しいディスクのシリアル番号がすべてのノードで同一になっていることや、新しいディスクが I/O フェンシングをサポートできることを確認します。

このユーティリティは、ディスクベースとサーバーベースのフェンシングをサポートします。

ユーティリティはクラスタ内のノード間の通信に SSH、RSH または hacli を使用します。ユーティリティを実行する前に、ノード間の通信が次のいずれかのプロトコルでセットアップされていることを確認してください。

vxfenswap(1M)マニュアルページを参照してください。

I/O フェンシングの条件について詳しくは、『Storage Foundation Cluster File System High Availability 設定およびアップグレードガイド』を参照してください。

次の場合は、I/O フェンシングを停止せずにコーディネータディスクを置き換えることができます。

- ディスクに不具合が発生したか、またはディスクが操作不能になったため、新しいディスクグループに切り換える。
  p.489 の「クラスタがオンラインのときの I/O フェンシングコーディネータディスクの置き換え」を参照してください。
  p.493 の「オンラインになっているクラスタ内のコーディネータディスクグループの置き換え」を参照してください。
  クラスタがオフラインのときにコーディネータディスクを置き換える場合、vxfenswap ユーティリティを使うことはできません。ユーティリティが実行するコーディネータディスクの置き換え手順を手動で実行する必要があります。
- コーディネータディスクに登録されているキーが失われている。

このようなケースでは、クラスタはネットワーク分割が発生したときにパニックすることがあります。vxfenswap コマンドを使って、コーディネータディスクを同じディスクに置き換えることができます。データ置き換え中に、失われたキーが再度登録されます。データ破損の危険性はありません。

p.499 の「コーディネータディスク上の紛失キーのリフレッシュ」を参照してください。

サーバー型のフェンシングの設定では、次のタスクを実行するために vxfenswap ユーティリティを使うことができます。

- カスタマイズされたコーディネーションポイントの計画された置き換えを実行する(CP サーバーまたは SCSI-3 ディスク)。

  p.508 の「オンラインクラスタでサーバーベースのフェンシングに使うコーディネーションポイントの置き換え」を参照してください。

- コーディネーションポイントで登録される I/O フェンシングのキーを更新する。

  p.510 の「サーバー型のフェンシングのコーディネーションポイントの登録キーの更新」を参照してください。

vxfenswap ユーティリティを使用して、SFCFSHA クラスタで、アプリケーションを停止させることなく、ディスクベースのフェンシングとサーバーベースのフェンシング間の移行を行うこともできます。

p.519 の「オンラインクラスタでのディスクベースのフェンシングからサーバーベースのフェンシングへの移行」を参照してください。

p.520 の「オンラインクラスタでのサーバーベースのフェンシングからディスクベースのフェンシングへの移行」を参照してください。

vxfenswap 操作が失敗した場合、`vxfenswap` コマンドの `-a cancel` を使って、vxfenswap ユーティリティによって加えられた変更を手動でロールバックできます。

- ディスク型のフェンシングの場合は、`vxfenswap -g diskgroup -a cancel` コマンドを使って vxfenswap 操作を取り消します。
  ディスクの置き換え処理中にノードが失敗した場合や、ディスクの置き換えを中止した場合は、このコマンドを実行する必要があります。
- サーバー型のフェンシングの場合は、`vxfenswap -a cancel` コマンドを使って vxfenswap 操作を取り消します。

クラスタがオンラインのときの I/O フェンシングコーディネータディスクの置き換え

動作しているクラスタで 1 つ以上のコーディネータディスクを追加、削除、置換する各手順を確認します。

警告: スクリプトが一連のコーディネータディスクを置き換える前にいずれかのノードがクラスタメンバーシップから除外されている場合、クラスタがパニックを起こす可能性があります。

クラスタがオンライン状態のとき、コーディネータディスクグループのディスクを置き換えるには

1. システム間の通信が適切に機能していることを確認します。
2. FaultTolerance 属性の値を決定します。

   ```
   # hares -display coordpoint -attribute FaultTolerance -localclus
   ```

3. フェンシングの設定の一部としての使用を計画するコーディネーションポイントの数を推定します。
4. FaultTolerance 属性の値を 0 に設定します。

   メモ: 値を 0 に設定する必要があるのは、この後の手順で、この属性の値をコーディネーションポイントの数未満の数に再設定する必要があるためです。これにより、Coordpoint エージェントにエラーが発生しないようにします。

5. LevelTwoMonitorFreq 属性の既存の値を確認します。

   ```
   #hares -display coordpoint -attribute LevelTwoMonitorFreq
   -localclus
   ```

   メモ: 次のステップに進む前に属性値をメモします。移行後、属性を再び有効化するときに同じ値に設定する必要があります。

   また、`hares -display coordpoint` を実行し、LevelTwoMonitorFreq 値が設定されているかどうかを確認できます。

6. Coordpoint エージェントのレベル 2 の監視の無効化します。

   ```
   # hares -modify coordpoint LevelTwoMonitorFreq 0
   ```

7 クラスタがオンラインになっていることを確認します。

```
# vxfenadm -d

I/O Fencing Cluster Information:
================================
Fencing Protocol Version: 201
Fencing Mode: SCSI3
Fencing SCSI3 Disk Policy: dmp
Cluster Members:
  * 0 (sys1)
  1 (sys2)
RFSM State Information:
  node 0 in state 8 (running)
  node 1 in state 8 (running)
```

8 コーディネータディスクグループをインポートします。

ファイル `/etc/vxfendg` には、コーディネータディスクを含むディスクグループの名前(通常は、vxfencoorddg)が含まれます。したがって次のコマンドを実行します。

```
# vxdg -tfC import `cat /etc/vxfendg`
```

ここで、

-t は、ノードが再起動するまでの間のみ、ディスクグループをインポートするよう指定します。

-f は、強制的にインポートすることを指定します。これは、1 つ以上のディスクがアクセス不能の場合に必要です。

-C は、インポートしたすべてのロックを削除するよう指定します。

9 セットアップで `VRTSvxvm`***version*** を使う場合、手順 10 に進みます。ディスクを追加または削除するために `coordinator=off` を設定する必要はありません。その他の VxVM バージョンでは、次の手順を実行します。

version **は特定のリリースバージョンです。**

コーディネータディスクグループの coordinator 属性の値を off にします。

```
# vxdg -g vxfencoorddg set coordinator=off
```

10 ディスクをコーディネータディスクグループから削除するには、VxVM ディスク管理者ユーティリティの `vxdiskadm` を使います。

11 コーディネータディスクグループに新しいディスクを追加するには、次の手順を実行します。

- ノードに新しいディスクを追加します。
- VxVM ディスクとして新しいディスクを初期化します。
- I/O フェンシングに準拠しているかどうかディスクを確認します。
- 新しいディスクをコーディネータディスクグループに追加し、コーディネータ属性値をコーディネータディスクグループに対して「オン」と設定します。

手順について詳しくは、『Storage Foundation Cluster File System High Availability 設定およびアップグレードガイド』を参照してください。

ディスクグループの内容は変わっていますが、I/O フェンシングは同じ状態を保っています。

12 1 つのノードから vxfenswap ユーティリティを開始します。ディスクグループをユーティリティに指定する必要があります。

このユーティリティのタスクは次のとおりです。

- 既存の /etc/vxfentab ファイルのバックアップを作成します。
- 各ノードで、変更するディスクグループ用のテストファイル /etc/vxfentab.test を作成します。
- vxfenswap コマンドで指定したディスクグループを読み取り、そのディスクグループを各ノードの /etc/vxfentab.test ファイルに追加します。
- すべてのノードで新しいディスクのシリアル番号が同じになっていることを確認します。チェックが失敗すると、スクリプトは終了します。
- 各ノードで新しいディスクが I/O フェンシングをサポートできることを確認します。

13 正常なディスクであることが確認されると、ユーティリティは、成功したことを報告し、新しいコーディネータディスクセットをコミットするかどうかを確認します。

14 コーディネーションポイント上のキーをクリアして vxfenswap 操作を続行するかどうかを指定します。

```
Do you want to clear the keys on the coordination points
and proceed with the vxfenswap operation? [y/n] (default: n) y
```

15 ユーティリティが表示するメッセージを確認し、コーディネータディスクの新しいセットをコミットすることを確認します。置き換えない場合は、手順 16 にスキップします。

```
Do you wish to commit this change? [y/n] (default: n) y
```

ユーティリティがコミットを正常に処理すると、ユーティリティは /etc/vxfentab.test ファイルを /etc/vxfentab ファイルに移動します。

16 コーディネータディスクの新しいセットをコミットしない場合は、n と答えます。

vxfenswap ユーティリティはディスク置き換え操作をロールバックします。

17 コーディネータフラグがステップ 9 で off に設定されている場合は、on に設定します。

```
# vxdg -g vxfencoorddg set coordinator=on
```

18 ディスクグループをデポートします。

```
# vxdg deport vxfencoorddg
```

19 CoordPoint エージェントの LevelTwoMonitorFreq 属性を再び有効化します。属性を無効にする前に設定されていた値を使うこともできます。

```
# hares -modify coordpoint LevelTwoMonitorFreq Frequencyvalue
```

Frequencyvalue は属性値です。

20 FaultTolerance 属性の値を、コーディネーションポイントの合計数の 50% よりも低い値に設定します。

たとえば、設定に 4 つのコーディネーションポイントがある場合、属性値は 2 未満の値にする必要があります。2 を超える値を設定した場合、CoordPoint エージェントにエラーが発生します。

オンラインになっているクラスタ内のコーディネータディスクグループの置き換え

vxfenswap ユーティリティを使って、コーディネータディスクグループを置き換えることもできます。次の例は、コーディネータディスクグループ vxfencoorddg を新しいディスクグループ vxfendg に置き換えます。

コーディネータディスクグループを置き換えるには

1 システム間の通信が適切に機能していることを確認します。

2 FaultTolerance 属性の値を決定します。

```
# hares -display coordpoint -attribute FaultTolerance -localclus
```

3 フェンシングの設定の一部としての使用を計画するコーディネーションポイントの数を推定します。

4 FaultTolerance 属性の値を 0 に設定します。

メモ: 値を 0 に設定する必要があるのは、この後の手順で、この属性の値をコーディネーションポイントの数未満の数に再設定する必要があるためです。これにより、Coordpoint エージェントにエラーが発生しないようにします。

---

5 LevelTwoMonitorFreq 属性の既存の値を確認します。

```
# hares -display coordpoint -attribute LevelTwoMonitorFreq
-localclus
```

メモ: 次のステップに進む前に属性値をメモします。移行後、属性を再び有効化するときに同じ値に設定する必要があります。

6 Coordpoint エージェントのレベル 2 の監視の無効化します。

```
# haconf -makerw

# hares -modify coordpoint LevelTwoMonitorFreq 0

# haconf -dump -makero
```

7 クラスタがオンラインになっていることを確認します。

```
# vxfenadm -d


I/O Fencing Cluster Information:
================================
Fencing Protocol Version: 201
Fencing Mode: SCSI3
Fencing SCSI3 Disk Policy: dmp
Cluster Members:
  * 0 (sys1)
  1 (sys2)
RFSM State Information:
  node 0 in state 8 (running)
  node 1 in state 8 (running)
```

8 /etc/vxfendg ファイル内にある現在のコーディネータディスクグループの名前(通常は vxfencoorddg)を見つけます。

```
# cat /etc/vxfendg
vxfencoorddg
```

9 現在のコーディネータディスクグループを置き換えることができる代替ディスクグループを見つけます。

```
# vxdisk -o alldgs list

DEVICE    TYPE    DISK  GROUP    STATUS
sda    auto:cdsdisk    -  (vxfendg)    online
sdb    auto:cdsdisk    -  (vxfendg)    online
sdc    auto:cdsdisk    -  (vxfendg)    online
sdx    auto:cdsdisk    -  (vxfencoorddg)    online
sdy    auto:cdsdisk    -  (vxfencoorddg)    online
sdz    auto:cdsdisk    -  (vxfencoorddg)    online
```

10 I/O フェンシングに準拠しているかどうか新しいディスクグループを検証します。次のコマンドを実行します。

```
# vxfentsthdw -c vxfendg
```

p.474 の「vxfentsthdw の -c オプションを使ったコーディネータディスクグループのテスト」を参照してください。

11 新しいディスクグループがまだデポートされていない場合は、次のコマンドを実行してディスクグループをデポートします。

```
# vxdg deport vxfendg
```

12 次のいずれかを実行します。

- 新しいフェンシングモードとディスクポリシー情報を含む /etc/vxfenmode.test ファイルを作成します。
- 新しいフェンシングモードとディスクポリシー情報を使って既存の /etc/vxfenmode を編集し、既存の /etc/vxfenmode.test ファイルがあれば削除します。

/etc/vxfenmode.test ファイルと /etc/vxfenmode ファイルの形式が同じであることに注意してください。

詳しくは、『Storage Foundation Cluster File System High Availability 設定およびアップグレードガイド』を参照してください。

13 任意のノードで、vxfenswap ユーティリティを起動します。たとえば、vxfendg がコーディネータディスクグループとして使う新しいディスクグループの場合、次のコマンドを実行します。

```
# vxfenswap -g vxfendg [-n]
```

このユーティリティのタスクは次のとおりです。

- 既存の /etc/vxfentab ファイルのバックアップを作成します。
- 各ノードで、変更するディスクグループ用のテストファイル /etc/vxfentab.test を作成します。
- vxfenswap コマンドで指定したディスクグループを読み取り、そのディスクグループを各ノードの /etc/vxfentab.test ファイルに追加します。
- すべてのノードで新しいディスクのシリアル番号が同じになっていることを確認します。チェックが失敗すると、スクリプトは終了します。
- 各ノードで新しいディスクグループが I/O フェンシングをサポートできることを確認します。

14 正常なディスクであることが確認されると、ユーティリティは、成功したことを報告し、コーディネータディスクグループを置き換えるかどうかを確認します。

15 コーディネーションポイント上のキーをクリアして vxfenswap 操作を続行するかどうかを指定します。

```
Do you want to clear the keys on the coordination points
and proceed with the vxfenswap operation? [y/n] (default: n) y
```

16 ユーティリティから表示されたメッセージを確認し、コーディネータディスクグループを置き換えることを確認します。置き換えない場合は、手順 21 にスキップします。

```
Do you wish to commit this change? [y/n] (default: n) y
```

ユーティリティがコミットを正常に処理すると、ユーティリティは /etc/vxfentab.test ファイルを /etc/vxfentab ファイルに移動します。

ユーティリティはこの新しいディスクグループで /etc/vxfendg ファイルも更新します。

17 コーディネータフラグを「on」に設定する前にインポートされていない場合は、新しいディスクグループをインポートします。

```
# vxdg -t import vxfendg
```

18 新しいコーディネータディスクグループに対して、coordinator 属性の値を「on」に設定します。

```
# vxdg -g vxfendg set  coordinator=on
```

古いディスクグループに対して、coordinator 属性の値を「off」に設定します。

```
# vxdg -g vxfencoorddg set  coordinator=off
```

19 新しいディスクグループをデポートします。

```
#  vxdg deport vxfendg
```

20 コーディネータディスクグループが変更されたことを確認します。

```
# cat /etc/vxfendg
vxfendg
```

コーディネータディスクグループの交換操作が完了しました。

21 コーディネータディスクグループを置き換えない場合は、プロンプトで n と答えます。

vxfenswap ユーティリティは、コーディネータディスクグループへのすべての変更をロールバックします。

22 CoordPoint エージェントの LevelTwoMonitorFreq 属性を再び有効にします。属性を無効化する前に設定した値を使用する必要がある場合があります。

```
# haconf -makerw

# hares -modify coordpoint LevelTwoMonitorFreq Frequencyvalue

# haconf -dump -makero
```

Frequencyvalue は属性値です。

23 FaultTolerance 属性の値を、コーディネーションポイントの合計数の 50% よりも低い値に設定します。

たとえば、設定に 4 つのコーディネーションポイントがある場合、属性値は 2 未満の値にする必要があります。2 を超える値を設定した場合、CoordPoint エージェントにエラーが発生します。

回復したサイトからコーディネータディスクグループへのディスクの追加

キャンパスクラスタ環境で、プライマリサイトが停止し、限定されたディスクセットを持つセカンダリサイトがオンラインになった場合を考えます。プライマリサイトが復元されると、プライマリサイトのディスクも利用可能になり、コーディネータディスクとしての機能を果たすことができます。vxfenswap ユーティリティを使って、このディスクをコーディネータディスクグループに追加できます。

回復したサイトからコーディネータディスクグループに新しいディスクを追加するには

1 システム間の通信が適切に機能していることを確認します。

2 クラスタがオンラインになっていることを確認します。

```
# vxfenadm -d


I/O Fencing Cluster Information:
================================
Fencing Protocol Version: 201
Fencing Mode: SCSI3
Fencing SCSI3 Disk Policy: dmp
Cluster Members:
  * 0 (sys1)
  1 (sys2)
RFSM State Information:
  node 0 in state 8 (running)
  node 1 in state 8 (running)
```

3 コーディネータディスクグループの名前を確認します。

```
# cat /etc/vxfendg
vxfencoorddg
```

4 次のコマンドを実行します。

```
# vxdisk -o alldgs list

DEVICE    TYPE    DISK  GROUP    STATUS
sdx    auto:cdsdisk    -  (vxfencoorddg)    online
sdy    auto    -  -    offline
sdz    auto    -  -    offline
```

5 コーディネータディスクグループで使われるディスク数を確認します。

```
# vxfenconfig -l
I/O Fencing Configuration Information:
======================================
Count                : 1
Disk List
Disk Name        Major  Minor  Serial Number     Policy

/dev/vx/rdmp/sdx       32  48  R450 00013154 0312     dmp
```

6 プライマリサイトがオンラインになったら、クラスタ内の任意のノードで vxfenswap ユーティリティを開始します。

```
# vxfenswap -g vxfencoorddg [-n]
```

7 コーディネータディスクの数を確認します。

```
# vxfenconfig -l
I/O Fencing Configuration Information:
======================================
Single Disk Flag      : 0
Count                 : 3
Disk List
Disk Name         Major  Minor  Serial Number     Policy

/dev/vx/rdmp/sdx       32  48  R450 00013154 0312     dmp
/dev/vx/rdmp/sdy       32  32  R450 00013154 0313     dmp
/dev/vx/rdmp/sdz       32  16  R450 00013154 0314     dmp
```

コーディネータディスク上の紛失キーのリフレッシュ

登録されているキーがコーディネータディスクから失われた場合に、ネットワーク分割が発生が発生すると、クラスタはパニックを起こす場合があります。

vxfenswap ユーティリティを使って、コーディネータディスクを同じディスクに置き換えることができます。ディスクの置き換え中に、vxfenswap ユーティリティは失われたキーを登録します。

コーディネータディスクで失われたキーを更新するには

1. システム間の通信が適切に機能していることを確認します。
2. クラスタがオンラインになっていることを確認します。

```
# vxfenadm -d


I/O Fencing Cluster Information:
================================
Fencing Protocol Version: 201
Fencing Mode: SCSI3
Fencing SCSI3 Disk Policy: dmp
Cluster Members:
  * 0 (sys1)
  1 (sys2)
RFSM State Information:
  node 0 in state 8 (running)
  node 1 in state 8 (running)
```

3. 次のコマンドを実行して、キーのないコーディネータディスクを表示します。

```
# vxfenadm -s all -f /etc/vxfentab


Device Name: /dev/vx/rdmp/sdx
Total Number of Keys: 0
No keys...
...
```

4. /etc/vxfenmode ファイルを /etc/vxfenmode.test ファイルにコピーします。
   これにより、両方のファイルの設定の詳細が確実に同じになります。
5. 任意のノードで、次のコマンドを実行して vxfenswap ユーティリティを開始します。

```
# vxfenswap -g vxfencoorddg [-n]
```

6. キーがコーディネータディスクに自動的に配置されていることを確認します。

```
# vxfenadm -s all -f /etc/vxfentab


Device Name: /dev/vx/rdmp/sdx
Total Number of Keys: 4
...
```

コーディネーションポイントサーバーの管理について

この項では、コーディネーションポイントサーバー（CP サーバー）上で管理と保守のタスクを実行する方法について説明します。

cpsadm コマンドと関連するコマンドオプションについて詳しくは、cpsadm（1M）マニュアルページを参照してください。

CP サーバーの操作（cpsadm）

表 15-4 は、コーディネーションポイントサーバー（CP サーバー）の操作と必要な権限のリストです。

表 15-4 CP サーバーの操作に対するユーザーの権限

CP サーバーの操作	CP サーバーのオペレータ	CP サーバーの管理者
add_cluster	–	✓
rm_clus	–	✓
add_node	✓	✓
rm_node	✓	✓
add_user	–	✓
rm_user	–	✓
add_clus_to_user	–	✓
rm_clus_from_user	–	✓
reg_node	✓	✓
unreg_node	✓	✓
preempt_node	✓	✓
list_membership	✓	✓
list_nodes	✓	✓
list_users	✓	✓
halt_cps	–	✓
db_snapshot	–	✓
ping_cps	✓	✓

CP サーバーの操作	CP サーバーのオペレータ	CP サーバーの管理者
client_preupgrade	✓	✓
server_preupgrade	✓	✓
list_protocols	✓	✓
list_version	✓	✓
list_ports	–	✓
add_port	–	✓
rm_port	–	✓

CP サーバーデータベースからの SFCFSHA クラスタエントリの追加と削除

- SFCFSHA クラスタを CP サーバーデータベースに追加するには
  次のようにコマンドを入力します。

  ```
  # cpsadm -s cp_server -a add_clus -c cluster_name -u uuid
  ```

- SFCFSHA クラスタを CP サーバーデータベースから削除するには
  次のようにコマンドを入力します。

  ```
  # cpsadm -s cp_server -a rm_clus -u uuid
  ```

cp_server CP サーバーの仮想 IP アドレスまたは仮想ホスト名。

cluster_name SFCFSHA クラスタ名。

uuid SFCFSHA クラスタの UUID(全世界で一意の ID)。

CP サーバーデータベースに対する SFCFSHA クラスタノードの追加と削除

- SFCFSHA クラスタノードを CP サーバーデータベースに追加するには
  次のようにコマンドを入力します。

  ```
  # cpsadm -s cp_server -a add_node -u uuid -n nodeid
  -h host
  ```

- SFCFSHA クラスタノードを CP サーバーデータベースから削除するには

次のようにコマンドを入力します。

```
# cpsadm -s cp_server -a rm_node -u uuid -n nodeid
```

cp_server	CP サーバーの仮想 IP アドレスまたは仮想ホスト名。
uuid	SFCFSHA クラスタの UUID(全世界で一意の ID)。
nodeid	SFCFSHA クラスタノードのノード ID。
host	ホスト名

CP サーバーユーザーの追加または削除

- ユーザーを追加するには
  次のようにコマンドを入力します。

  ```
  # cpsadm -s cp_server -a add_user -e user_name -f user_role
  -g domain_type -u uuid
  ```

- ユーザーを削除するには
  次のようにコマンドを入力します。

  ```
  # cpsadm -s cp_server -a rm_user -e user_name -g domain_type
  ```

cp_server	CP サーバーの仮想 IP アドレスまたは仮想ホスト名。
user_name	CP サーバー設定に追加するユーザー。
user_role	ユーザーの役割。cps_admin または cps_operator。
domain_type	ドメインタイプ(例: vx、unixpwd、nis)。
uuid	SFCFSHA クラスタの UUID(全世界で一意の ID)。

CP サーバーユーザーのリスト表示

CP サーバーユーザーのリストを表示するには

次のようにコマンドを入力します。

```
# cpsadm -s cp_server -a list_users
```

すべての SFCFSHA クラスタのノードのリスト表示

すべての SFCFSHA クラスタノードのノードをリスト表示するには

次のようにコマンドを入力します。

```
# cpsadm -s cp_server -a list_nodes
```

SFCFSHA クラスタのノードのメンバーシップのリスト表示

SFCFSHA クラスタのノードのメンバーシップをリスト表示するには
次のようにコマンドを入力します。

```
# cpsadm -s cp_server -a list_membership -c cluster_name
```

cp_server CP サーバーの仮想 IP アドレスまたは仮想ホスト名。

cluster_name SFCFSHA クラスタ名。

ノードの獲得

ノードを獲得するには次のコマンドを使います。

ノードを獲得するには

◆ 次のようにコマンドを入力します。

```
# cpsadm -s cp_server -a preempt_node -u uuid -n nodeid
-v victim_node id
```

cp_server CP サーバーの仮想 IP アドレスまたは仮想ホスト名。

uuid SFCFSHA クラスタの UUID(全世界で一意の ID)。

nodeid SFCFSHA クラスタノードのノード ID。

victim_node id 1 つ以上の被害側ノードのノード ID。

ノードの登録と登録解除

■ ノードを登録するには
次のようにコマンドを入力します。

```
# cpsadm -s cp_server -a reg_node -u uuid -n nodeid
```

■ ノードを登録解除するには
次のようにコマンドを入力します。

```
# cpsadm -s cp_server -a unreg_node -u uuid -n nodeid
```

cp_server	CP サーバーの仮想 IP アドレスまたは仮想ホスト名。
uuid	SFCFSHA クラスタの UUID(全世界で一意の ID)。
nodeid	SFCFSHA クラスタノードのノード ID。

SFCFSHA クラスタへのユーザーのアクセスの有効化と無効化

- SFCFSHA クラスタへのユーザーのアクセスを有効にするには
  次のようにコマンドを入力します。

```
# cpsadm -s cp_server -a add_clus_to_user -e user
-f user_role -g domain_type -u uuid
```

- SFCFSHA クラスタへのユーザーのアクセスを無効にするには
  次のようにコマンドを入力します。

```
# cpsadm -s cp_server -a rm_clus_from_user -e user_name
-f user_role -g domain_type -u uuid
```

cp_server	CP サーバーの仮想 IP アドレスまたは仮想ホスト名。
user_name	CP サーバーに追加するユーザー名。
user_role	ユーザーの役割。cps_admin または cps_operator。
domain_type	ドメインタイプ(例: vx、unixpwd、nis)。
uuid	SFCFSHA クラスタの UUID(全世界で一意の ID)。

VCS 制御外での CP サーバーの起動と停止

VCS 制御外の CP(コーディネーションポイント)サーバーの起動と停止を行うことができます。

VCS 制御外の CP サーバーを起動するには

1. `vxcpserv` のバイナリを直接実行します。

   ```
   # /opt/VRTScps/bin/vxcpserv
   ```

   コマンドが成功した場合、コマンドはメッセージなしですぐに返されます。

2. CP サーバーの状態を確認するには、ログファイル
   `/var/VRTScps/log/cpserver_A.log` を検証します。

VCS 制御外の CP サーバーを停止するには

1 次のコマンドを実行します。

```
# cpsadm -s cp_server -a halt_cps
```

変数 *cp_server* は CP サーバーの仮想 IP アドレスまたは仮想ホスト名を表し、*port_number* は CP サーバーが応答準備しているポート番号を表します。

2 CP サーバーが停止メッセージを受信して停止されたことを確認するには、ログファイル `/var/VRTScps/log/cpserver_A.log` を検証します。

CP サーバーの接続性の確認

CP サーバーの接続性を確認するには

次のようにコマンドを入力します。

```
# cpsadm -s cp_server -a ping_cps
```

実行時における CP サーバーの仮想 IP アドレスとポートの追加と削除

実行時に CP サーバーの仮想 IP アドレスとポートを追加および削除する手順は、Veritas Product Authentication Services (AT) を介した通信と、非セキュア通信にのみ適用されます。HTTPS での通信には適用されません。

コーディネーションポイントサーバー(CP サーバー)通信に対しては、複数の仮想 IP アドレスを使用できます。各仮想 IP アドレスにはポート番号を割り当てることができます。

CP サーバーの初期設定を行った後に仮想 IP アドレスを追加または削除する場合は、`cpsadm` コマンドを実行できます。追加または削除するこれらの仮想 IP アドレスとポートによって `vxcps.conf` ファイルは変更されません。そのため、これらの変更は CP サーバーを再起動すると維持されません。

詳しくは、`cpsadm`(1M)マニュアルページを参照してください。

実行時に CP サーバーに対する仮想 IP アドレスとポートを追加または削除するには

1 応答準備するように CP サーバーが設定されているすべてのポートを一覧表示するには、次のコマンドを実行します。

```
# cpsadm -s cp_server -a list_ports
```

CP サーバーが特定のポートで少なくとも 1 回正常に応答準備できなかった場合は、出力の Connect History に never と表示されます。vxcpserv プロセスの開始時に IP アドレスがダウンしていた場合、IP アドレスが後にアップすると、vxcpserv はそれらの IP アドレスにバインドします。次に例を示します。

```
# cpsadm -s 127.0.0.1 -a list_ports

IP Address                      Connect History
[10.209.79.60]:14250            once
[10.209.79.61]:56789            once
[10.209.78.252]:14250           never
[192.10.10.32]:14250            once
```

CP サーバーは、ポートの健全性をアクティブに監視します。IP アドレスとポートのいずれかの組み合わせで CP サーバーが少なくとも 1 回正常に応答準備した場合は、CP サーバーの有効期間中において後にポートがダウンしても、その IP アドレスとポートの Connect History に once と表示されます。IP アドレスの最新の状態は、VCS で設定されている該当の IP リソース状態から取得できます。

2 CP サーバーを再起動せずに CP サーバーに対して新しいポート(IP アドレスとポート)を追加するには、次のコマンドを実行します。

```
# cpsadm -s cp_server -a add_port
-i ip_address -r port_number
```

次に例を示します。

```
# cpsadm -s 127.0.0.1 -a add_port -i 10.209.78.52 -r 14250
Port [10.209.78.52]:14250 successfully added.
```

3 CP サーバーを再起動せずに CP サーバーがポート(IP アドレスとポート)で応答準備するのを停止するには、次のコマンドを実行します。

```
# cpsadm -s cp_server -a rm_port
-i ip_address -r port_number
```

次に例を示します。

```
# cpsadm -s 10.209.78.52 -a rm_port -i 10.209.78.252
No port specified. Assuming default port i.e 14250
Port [10.209.78.252]:14250 successfully removed.
```

CP サーバーデータベースのスナップショットの取得

CP サーバーデータベースのスナップショットを取得するには

次のようにコマンドを入力します。

```
# cpsadm -s cp_server -a db_snapshot
```

CP サーバーデータベースのスナップショットは次の場所に格納されます。`/etc/VRTScps/db/cpsdbsnap. DATE.TIME`

ここで、*DATE* はスナップショットの作成日、*TIME* はスナップショットの作成時間です。

オンラインクラスタでサーバーベースのフェンシングに使うコーディネーションポイントの置き換え

オンラインの SFCFSHA クラスタで、アプリケーションを停止させることなく、カスタマイズされたコーディネーションポイント(CP サーバーまたは SCSI-3 ディスク)の計画的な置き換えを実行するには、次の手順を使います。

メモ: 複数のクラスタが同じ CP サーバーを共有する場合、各クラスタでこの置き換え手順を実行する必要があります。

オンラインクラスタで、vxfen_mechanism=cps 設定を使ってカスタマイズモードでフェンシングを実行しているときは、vxfenswap ユーティリティを使ってコーディネーションポイントを置き換えることができます。オンラインクラスタで、ユーティリティはサーバーベースのフェンシング(vxfen_mode=customized)からディスクベースのフェンシング(vxfen_mode=scsi3)への移行、またはその逆の移行もサポートします。

ただし、SFCFSHA クラスタでフェンシングが無効になっている場合 (vxfen_mode=disabled) は、クラスタをオフラインにして、ディスクベースのフェンシングまたはサーバーベースのフェンシングを設定する必要があります。

p.512 の「CP サーバーの配備と移行のシナリオ」を参照してください。

コーディネーションポイントの置き換え操作は vxfenswap -a cancel コマンドを使っていつでも取り消すことができます。

p.488 の「vxfenswap ユーティリティについて」を参照してください。

オンラインクラスタのコーディネーションポイントを交換するには

1 SFCFSHA クラスタノードとユーザーが新しい CP サーバーに追加されたことを確認します。次のコマンドを実行します。

```
# cpsadm -s cpserver -a list_nodes
# cpsadm -s cpserver -a list_users
```

ここで SFCFSHA クラスタノードが存在しない場合は、SFCFSHA クラスタが使用する新しい CP サーバーを準備します。

詳しくは、『Storage Foundation Cluster File System High Availability 設定およびアップグレードガイド』を参照してください。

2 フェンシングが、コーディネーションポイントの古いセットを使用し、カスタマイズされたモードのクラスタで実行していることを確認します。

たとえば、次のコマンドを入力します。

```
# vxfenadm -d
```

コマンドの出力は次のようになります。

```
I/O Fencing Cluster Information:
================================
Fencing Protocol Version: <version>
Fencing Mode: Customized
Cluster Members:
* 0 (sys1)
1 (sys2)
RFSM State Information:
node 0 in state 8 (running)
node 1 in state 8 (running)
```

3 各 SFCFSHA クラスタ ノードで、CP サーバー情報などのフェンシング設定の変更を含む新しい /etc/vxfenmode.test ファイルを作成します。

確認し、必要ならば、セキュリティ、コーディネーションポイント、および設定に該当する場合は vxfendg に関する vxfenmode パラメータを更新します。

これらのパラメータと新しく設定可能な値について詳しくは、vxfenmode ファイル内のテキスト情報を参照してください。

4 クラスタのノードの 1 つから、vxfenswap ユーティリティを実行します。

vxfenswap ユーティリティはすべてのクラスタノードへの安全な ssh の接続を必要とします。デフォルトの ssh の代わりに rsh を使うには -n を使用します。–p *<protocol>* を使います。*<protocol>* は ssh、rsh、または hacli です。

```
#  vxfenswap [-n | -p <protocol>]
```

5 ユーティリティから表示されたメッセージを確認し、変更をコミットするかどうかを確認します。

- 新しいフェンシング設定の変更をコミットしない場合は、Enter キーを押すか、プロンプトで n と答えます。

  ```
  Do you wish to commit this change? [y/n] (default: n) n
  ```

  vxfenswap ユーティリティは移行操作をロールバックします。

- 新しいフェンシング設定の変更をコミットする場合は、プロンプトで y と答えます。

  ```
  Do you wish to commit this change? [y/n] (default: n) y
  ```

  ユーティリティは、操作を正常に完了すると、/etc/vxfenmode.test ファイルを /etc/vxfenmode ファイルに移動します。

6 vxfen ドライバによって現在使用されているコーディネーションポイントを確認することで、vxfenswap ユーティリティの正常な実行を確認します。

たとえば、次のコマンドを実行します。

```
# vxfenconfig -l
```

サーバー型のフェンシングのコーディネーションポイントの登録キーの更新

SFCFSHA クラスタがオンラインの場合のコーディネーションポイント（CP サーバー）のキーの交換には、そのコーディネーションポイントの登録の更新が含まれます。SFCFSHA クラスタでアプリケーション停止時間を発生させずに CP サーバーで登録の計画的な更新を実行できます。CP サーバーエージェントが CP サーバーデータベースのこのような登録の損失についてのアラートを発行した場合は、CP サーバーで登録を更新する必要があります。

次の手順ではコーディネーションポイントの登録を更新する方法を説明します。

サーバーベースのフェンシングで使うコーディネーションポイント上の登録キーを更新するには

1 SFCFSHA クラスタノードとユーザーが新しい CP サーバーに追加されたことを確認します。次のコマンドを実行します。

```
# cpsadm -s cp_server -a list_nodes

# cpsadm -s cp_server -a list_users
```

ここで SFCFSHA クラスタノードが存在しない場合は、SFCFSHA クラスタが使用する新しい CP サーバーを準備します。

詳しくは、『Storage Foundation Cluster File System High Availability 設定およびアップグレードガイド』を参照してください。

2 フェンシングが /etc/vxfenmode ファイルで指定されているコーディネーションポイントを使用し、カスタマイズされたモードでクラスタ上で実行していることを確認します。

/etc/vxfenmode.test ファイルがある場合、そのファイル内の情報と /etc/vxfenmode ファイルを同じにしてください。そうしないと、vxfenswap ユーティリティは /etc/vxfenmode.test ファイルに記載された情報を使います。

たとえば、次のコマンドを入力します。

```
# vxfenadm -d

================================
Fencing Protocol Version: 201
Fencing Mode: CUSTOMIZED
Cluster Members:
* 0 (sys1)
1 (sys2)
RFSM State Information:
node 0 in state 8 (running)
node 1 in state 8 (running)
```

3 I/O フェンシングで現在使っているコーディネーションポイントを一覧表示します。

```
# vxfenconfig -l
```

4 /etc/vxfenmode ファイルを /etc/vxfenmode.test ファイルにコピーします。

これにより、両方のファイルの設定の詳細が確実に同じになります。

5 クラスタの 1 つのノードから vxfenswap ユーティリティを実行します。

vxfenswap ユーティリティはすべてのクラスタノードへの安全な ssh の接続を必要とします。デフォルトの ssh の代わりに rsh を使うには、-n を使用します。

次に例を示します。

```
# vxfenswap [-n]
```

コマンドの出力は次のようになります。

```
VERITAS vxfenswap version <version> <platform>
The logfile generated for vxfenswap is
/var/VRTSvcs/log/vxfen/vxfenswap.log.
19156
Please Wait...
VXFEN vxfenconfig NOTICE Driver will use customized fencing
- mechanism cps
Validation of coordination points change has succeeded on
all nodes.
You may commit the changes now.
WARNING: This may cause the whole cluster to panic
if a node leaves membership before the change is complete.
```

6 変更をコミットするかどうかを確認するメッセージを表示されます。yを入力してコミットします。

コマンドはコーディネーションポイント置き換えの成功確認を返します。

7 vxfenswap ユーティリティの実行が成功したことを確認します。CP エージェントが設定されている場合、コーディネーションポイントの登録が見つかると ONLINE が報告されます。CP サーバー上の登録は cpsadm ユーティリティを、コーディネータディスク上の登録は vxfenadm ユーティリティをそれぞれ使って表示できます。

実行中のオンラインコーディネーションポイントの更新操作は以下のコマンドを使っていつでも取り消すことができます。

```
# vxfenswap -a cancel
```

CP サーバーの配備と移行のシナリオ

表 15-5 に、サポート対象の配備と移行のシナリオ、および SFCFSHA クラスタと CP サーバーで実行する必要がある手順を示します。

表 15-5　CP サーバーの配備と移行のシナリオ

シナリオ	CP サーバー	SFCFSHA クラスタ	必要な処理
SFCFSHA クラスタの CP サーバーを初めて設定する	新しい CP サーバー	コーディネーションポイントとして CP サーバーを使用する新規 SFCFSHA クラスタ	**指定された CP サーバーで、次のタスクを実行します。** **1** 新しい CP サーバーを設定する準備をします。 **2** CP サーバーを設定します。 **3** SFCFSHA クラスタで使用する新しい CP サーバーを準備します。 SFCFSHA クラスタノードで、サーバー型の I/O フェンシングを設定します。 手順については、『Cluster Server 設定およびアップグレードガイド』を参照してください。
既存の運用 CP サーバーに新しい SFCFSHA クラスタを追加する	既存の運用 CP サーバー	新しいSFCFSHA クラスタ	SFCFSHA クラスタノードで、サーバー型の I/O フェンシングを設定します。 手順については、『Cluster Server 設定およびアップグレードガイド』を参照してください。 **メモ:** 既存の CP サーバーが IPv4 のみをサポートし、CP サーバーとの通信に IPv6 アドレスを使用するように設定されたノードが追加する新しいクラスタに存在する場合、既存の CP サーバーを IPv6 またはデュアルスタック構成に移行する必要があります。
既存の CP サーバーから新しい CP サーバーへコーディネーションポイントを交換する	新しい CP サーバー	コーディネーションポイントとして CP サーバーを使用する既存 SFCFSHA クラスタ	**指定された CP サーバーで、次のタスクを実行します。** **1** 新しい CP サーバーを設定する準備をします。 **2** CP サーバーを設定します。 **3** SFCFSHA クラスタで使用する新しい CP サーバーを準備します。 手順については、『Cluster Server 設定およびアップグレードガイド』を参照してください。 SFCFSHA クラスタのノードで、`vxfenswap` コマンドを実行して CP サーバーの交換へと進みます。 p.508 の「オンラインクラスタでサーバーベースのフェンシングに使うコーディネーションポイントの置き換え」を参照してください。

シナリオ	CP サーバー	SFCFSHA クラスタ	必要な処理
既存の CP サーバーから運用 CP サーバーコーディネーションポイントへコーディネーションポイントを交換する	運用 CP サーバー	コーディネーションポイントとして CP サーバーを使用する既存 SFCFSHA クラスタ	SFCFSHA クラスタのノードで、vxfenswap コマンドを実行して CP サーバーの交換へと進みます。 p.508 の「オンラインクラスタでサーバーベースのフェンシングに使うコーディネーションポイントの置き換え」を参照してください。

新しい CP サーバーコーディネーションポイントを持つ SFCFSHA クラスタでフェンシングを有効にする	新しい CP サーバー	フェンシングが無効モードで設定されている既存の SFCFSHA クラスタ	**メモ:** 無効モードのフェンシングからカスタマイズされたモードに移行すると、SFCFSHA クラスタでアプリケーションのダウンタイムが発生します。
			指定された CP サーバーで、次のタスクを実行します。
			1 新しい CP サーバーを設定する準備をします。
			2 新しい CP サーバーを設定します。
			3 SFCFSHA クラスタで使用する新しい CP サーバーを準備します。
			手順については、『Cluster Server 設定およびアップグレードガイド』を参照してください。
			SFCFSHA クラスタノードで、以下の手順を実行します。
			1 SFCFSHA クラスタですべてのアプリケーション、VCS、およびフェンシングを停止します。
			2 VCS を停止するには、(すべての SFCFSHA クラスタノードで実行されるように) 次のコマンドを使用します。
			`# hastop -local`
			3 次のコマンドを使ってフェンシングを停止します。
			RHEL 7、SLES 12、およびサポート対象の RHEL 配布の場合:
			`# systemctl stop vxfen`
			以前のバージョンの RHEL、SLES、およびサポート対象の RHEL 配布の場合:
			`# /etc/init.d/vxfen stop`
			4 SFCFSHA クラスタで I/O フェンシングを再設定します。
			手順については、『Cluster Server 設定およびアップグレードガイド』を参照してください。

シナリオ	CP サーバー	SFCFSHA クラスタ	必要な処理

運用 CP サーバーコーディネーションポイントを持つ SFCFSHA クラスタでフェンシングを有効にする	運用 CP サーバー	フェンシングが無効モードで設定されている既存の SFCFSHA クラスタ	**メモ:** 無効モードのフェンシングからカスタマイズされたモードに移行すると、アプリケーションのダウンタイムが発生します。
			
			指定された CP サーバーで、新しい CP サーバーを設定する準備をします。
			
			この手順については、『Cluster Server 設定およびアップグレードガイド』を参照してください。
			
			SFCFSHA クラスタノードで、以下のタスクを実行します。
			
			1 SFCFSHA クラスタですべてのアプリケーション、VCS、およびフェンシングを停止します。
			
			2 VCS を停止するには、(すべての SFCFSHA クラスタノードで実行されるように) 次のコマンドを使用します。
			
			`#hastop -local`
			
			3 次のコマンドを使ってフェンシングを停止します。
			
			RHEL 7、SLES 12、およびサポート対象の RHEL 配布の場合:
			
			`# systemctl stop vxfen`
			
			以前のバージョンの RHEL、SLES、およびサポート対象の RHEL 配布の場合:
			
			`# /etc/init.d/vxfen stop`
			
			4 SFCFSHA クラスタでフェンシングを再設定します。
			
			手順については、『Cluster Server 設定およびアップグレードガイド』を参照してください。

シナリオ	CP サーバー	SFCFSHA クラスタ	必要な処理

新しい CP サーバーコーディネーションポイントを持つ SFCFSHA クラスタでフェンシングを有効にする	新しい CP サーバー	フェンシングが scsi3 モードで設定されている既存の SFCFSHA クラスタ	**指定された CP サーバーで、次のタスクを実行します。**
			1 新しい CP サーバーを設定する準備をします。
			2 新しい CP サーバーを設定します。
			3 SFCFSHA クラスタで使用する新しい CP サーバーを準備します。
			手順については、『Cluster Server 設定およびアップグレードガイド』を参照してください。
			クラスタがオンラインかオフラインかに応じて、次の手順を実行します。
			オンラインであるクラスタの場合は、SFCFSHA クラスタで次のタスクを実行します。
			◆ `vxfenswap` コマンドを実行して、ディスクベースのフェンシングからサーバーベースのフェンシングに移行します。
			p.519 の「オンラインクラスタでのディスクベースのフェンシングからサーバーベースのフェンシングへの移行」を参照してください。
			オフラインであるクラスタの場合は、SFCFSHA クラスタで次のタスクを実行します。
			1 SFCFSHA クラスタですべてのアプリケーション、VCS、およびフェンシングを停止します。
			2 VCS を停止するには、(すべての SFCFSHA クラスタノードで実行されるように) 次のコマンドを使用します。
			`# hastop -local`
			3 次のコマンドを使ってフェンシングを停止します。
			RHEL 7、SLES 12、およびサポート対象の RHEL 配布の場合:
			`# systemctl stop vxfen`
			以前のバージョンの RHEL、SLES、およびサポート対象の RHEL 配布の場合:
			`# /etc/init.d/vxfen stop`
			4 SFCFSHA クラスタで I/O フェンシングを再設定します。
			手順については、『Cluster Server 設定およびアップグレードガイド』を参照してください。

シナリオ	CP サーバー	SFCFSHA クラスタ	必要な処理
運用 CP サーバーコーディネーションポイントを持つ SFCFSHA クラスタでフェンシングを有効にする	運用 CP サーバー	フェンシングが無効モードで設定されている既存の SFCFSHA クラスタ	クラスタがオンラインかオフラインかに応じて、次の手順を実行します。 **オンラインであるクラスタの場合は、SFCFSHA クラスタで次のタスクを実行します。** ◆ vxfenswap コマンドを実行して、ディスクベースのフェンシングからサーバーベースのフェンシングに移行します。 p.519 の「オンラインクラスタでのディスクベースのフェンシングからサーバーベースのフェンシングへの移行」を参照してください。 **オフラインであるクラスタの場合は、SFCFSHA クラスタで次のタスクを実行します。** **1** SFCFSHA クラスタですべてのアプリケーション、VCS、およびフェンシングを停止します。 **2** VCS を停止するには、(すべての SFCFSHA クラスタノードで実行されるように) 次のコマンドを使用します。 #hastop -local **3** 次のコマンドを使ってフェンシングを停止します。 RHEL 7、SLES 12、およびサポート対象の RHEL 配布の場合: # systemctl stop vxfen 以前のバージョンの RHEL、SLES、およびサポート対象の RHEL 配布の場合: # /etc/init.d/vxfen stop **4** SFCFSHA クラスタでフェンシングを再設定します。 手順については、『Cluster Server 設定およびアップグレードガイド』を参照してください。
アプリケーションダウンタイムを発生させずにコーディネーションポイント (CP サーバー/コーディネータディスク) の SFCFSHA クラスタノードの登録を更新する	運用 CP サーバー	コーディネーションポイントとして CP サーバーを使用する既存 SFCFSHA クラスタ	SFCFSHA クラスタで、vxfenswap コマンドを実行して CP サーバーのキーを更新します。 p.510 の「サーバー型のフェンシングのコーディネーションポイントの登録キーの更新」を参照してください。

CP サーバーと SFCFSHA クラスタの通信に対する設定の非セキュアからセキュアへの移行

次の手順では、コーディネーションポイントサーバー (CP サーバー) と SFCFSHA クラスタの設定を非セキュアからセキュアに移行する方法を説明します。この手順は、CP サーバーと SFCFSHA クラスタ間の Veritas Product Authentication Services (AT) ベースの通信にのみ適用されます。

CP サーバーと SFCFSHA クラスタの設定を非セキュアからセキュアに移行するには

1. CP サーバーを使うすべてのクラスタノードで VCS を停止します。

   ```
   # hastop -all
   ```

2. すべてのクラスタのすべての SFCFSHA クラスタノードでフェンシングを停止します。

   ```
   # systemctl stop vxfen
   ```

3. 各 CP サーバーで次のコマンドを使ってすべての CP のサーバーを停止します。

   ```
   # hagrp -offline CPSSG -any
   ```

4. CP サーバーとすべてのクライアントでセキュリティが通信に対して設定されていることを確認します。

   詳しくは、『Cluster Server 設定およびアップグレードガイド』を参照してください。

5. - CP サーバーが SFHA クラスタでホストされている場合は、各 CP サーバーでこの手順を実行します。
     CPSSG サービスグループ内のマウントリソースをオンラインにします。

     ```
     # hares -online cpsmount -sys local_system_name
     ```

     残りの手順を完了します。
   - CP サーバーがシングルノード VCS クラスタでホストされている場合は、手順 8 にスキップして、残りの手順を完了します。

6. マウントリソースがオンラインになったら、デフォルトの場所から共有ストレージに `credentials` ディレクトリを移動します。

   ```
   # mv /var/VRTSvcs/vcsauth/data/CPSERVER /etc/VRTSvcs/db/
   ```

7. CP サーバーのすべてのノードで softlink を作成します。

   ```
   # ln -s /etc/VRTScps/db/CPSERVER ¥
   /var/VRTSvcs/vcsauth/data/CPSERVER
   ```

8. security=1 に設定するように各 CP サーバーの `/etc/vxcps.conf` を編集します。

9 次のコマンドを使って CP サーバーを起動します。

```
# hagrp -online CPSSG -any
```

10 クラスタの最初のノードで /etc/VRTSvcs/conf/config/main.cf を編集し、UseFence=SCSI3 属性を削除します。

クラスタの最初のノードで VCS を起動した後、クラスタのその他すべてのノードで VCS を起動します。

11 インストーラを使って各クラスタでフェンシングを再設定します。

```
# /opt/VRTS/install/インストーラ -fencing
```

ディスクベースとサーバーベースのフェンシング設定間の移行について

SFCFSHA クラスタのアプリケーションにダウンタイムを発生させずにフェンシング設定間を移行できます。

次の場合に、ディスクベースのフェンシングからサーバーベースのフェンシングに移行できます。

- サーバーベースのフェンシングの利点を活用する場合。
- 障害が発生したコーディネータディスクをコーディネーションポイントサーバー（CP サーバー）に置き換える場合。

p.519 の「オンラインクラスタでのディスクベースのフェンシングからサーバーベースのフェンシングへの移行」を参照してください。

同様に、CP サーバーシステムで保守タスクを実行するときに、サーバーベースのフェンシングからディスクベースのフェンシングに移行できます。

p.520 の「オンラインクラスタでのサーバーベースのフェンシングからディスクベースのフェンシングへの移行」を参照してください。

オンラインクラスタでのディスクベースのフェンシングからサーバーベースのフェンシングへの移行

インストーラを使うかまたは手動のいずれかで、SFCFSHA クラスタ内のアプリケーションダウンタイムを発生させずに、ディスクベースのフェンシングからサーバーベースのフェンシングに移行できます。

p.519 の「ディスクベースとサーバーベースのフェンシング設定間の移行について」を参照してください。

また、応答ファイルを使用してフェンシング間の設定を移行できます。

p.520 の「応答ファイルを使用したフェンシング設定間の移行」を参照してください。

警告: コーディネーションポイントの移行操作が完了する前にいずれかのノードがクラスタメンバーシップから除外された場合、クラスタはパニックを起こす可能性があります。

ここでは、次の内容について説明します。

スクリプトベースのインストーラを使った移行

手動による移行

オンラインクラスタでのサーバーベースのフェンシングからディスクベースのフェンシングへの移行

インストーラを使うかまたは手動のどちらかで、SFCFSHA クラスタ内のアプリケーションダウンタイムを発生させずに、サーバーベースのフェンシングからディスクベースのフェンシングに移行できます。

p.519 の「ディスクベースとサーバーベースのフェンシング設定間の移行について」を参照してください。

また、応答ファイルを使用してフェンシング間の設定を移行できます。

p.520 の「応答ファイルを使用したフェンシング設定間の移行」を参照してください。

警告: コーディネーションポイントの移行操作が完了する前にいずれかのノードがクラスタメンバーシップから除外された場合、クラスタはパニックを起こす可能性があります。

ここでは、次の内容について説明します。

スクリプトベースのインストーラを使った移行

手動による移行

応答ファイルを使用したフェンシング設定間の移行

通常、I/O フェンシング設定間の移行後にインストーラが生成する応答ファイルを使用できます。これらの応答ファイルを編集して、SFCFSHA クラスタの自動フェンシングの再設定を実行します。

応答ファイルを使って I/O フェンシングを設定するには

1. SFCFSHA が設定されていることを確認します。
2. システム間の通信が適切に機能していることを確認します。

3 SFCFSHA クラスタがオンライン状態であり、ディスクベースまたはサーバーベースのフェンシングが使われていることを確認します。

```
# vxfenadm -d
```

SFCFSHAクラスタでディスクベースのフェンシングが使われている場合の例:

```
I/O Fencing Cluster Information:
================================
Fencing Protocol Version: 201
Fencing Mode: SCSI3
Fencing SCSI3 Disk Policy: dmp
Cluster Members:
  * 0 (sys1)
  1 (sys2)
RFSM State Information:
  node 0 in state 8 (running)
  node 1 in state 8 (running)
```

SFCFSHA クラスタでサーバーベースのフェンシングが使われている場合の例:

```
I/O Fencing Cluster Information:
================================
Fencing Protocol Version: 201
Fencing Mode: Customized
Fencing Mechanism: cps
Cluster Members:
  * 0 (sys1)
  1 (sys2)
RFSM State Information:
  node 0 in state 8 (running)
  node 1 in state 8 (running)
```

4 I/O フェンシングを設定するクラスタシステムのいずれかに応答ファイルをコピーします。

サンプルファイルを確認して、I/O フェンシングを再設定します。

p.522 の「ディスクベースのフェンシングからサーバーベースのフェンシングに移行するための応答ファイルのサンプル」を参照してください。

p.523 の「サーバーベースのフェンシングからディスクベースのフェンシングに移行するための応答ファイルのサンプル」を参照してください。

p.523 の「単一 CP サーバーベースのフェンシングからサーバーベースのフェンシングに移行するためのサンプル応答ファイル」を参照してください。

5 必要に応じて、応答ファイルの変数の値を編集します。

p.523 の「フェンシング設定間を移行するための応答ファイル変数」を参照してください。

6 応答ファイルをコピーしたシステムで I/O フェンシングの再設定を開始します。次に例を示します。

```
# /opt/VRTS/install/インストーラ<version> -responsefile /tmp/
¥ response_file
```

<バージョン> は特定のリリースバージョンです。/tmp/*response_file* は応答ファイルの絶対パス名です。

ディスクベースのフェンシングからサーバーベースのフェンシングに移行するための応答ファイルのサンプル

3 枚のコーディネータディスクを使用するディスクベースのフェンシングから、1 台の CP サーバーと 2 枚のコーディネータディスクを使用するサーバーベースのフェンシングに移行するための応答ファイルのサンプルを次に示します。

```
$CFG{disks_to_remove}=[ qw(emc_clariion0_62) ];
$CFG{fencing_cps}=[ qw(10.198.89.251)];
$CFG{fencing_cps_ports}{"10.198.89.204"}=14250;
$CFG{fencing_cps_ports}{"10.198.89.251"}=14250;
$CFG{fencing_cps_vips}{"10.198.89.251"}=[ qw(10.198.89.251
10.198.89.204) ];
$CFG{fencing_ncp}=1;
$CFG{fencing_option}=4;
$CFG{opt}{configure}=1;
$CFG{opt}{fencing}=1;
$CFG{prod}="SFCFSHA60";
$CFG{systems}=[ qw(sys1 sys2) ];
$CFG{vcs_clusterid}=22462;
$CFG{vcs_clustername}="clus1";
```

サーバーベースのフェンシングからディスクベースのフェンシングに移行するための応答ファイルのサンプル

1 台の CP サーバーと 2 枚のコーディネータディスクを使用するサーバーベースのフェンシングから、3 枚のコーディネータディスクを使用するディスクベースのフェンシングに移行するための応答ファイルのサンプルを次に示します。

```
$CFG{fencing_disks}=[ qw(emc_clariion0_66) ];
$CFG{fencing_mode}="scsi3";
$CFG{fencing_ncp}=1;
$CFG{fencing_ndisks}=1;
$CFG{fencing_option}=4;
$CFG{opt}{configure}=1;
$CFG{opt}{fencing}=1;
$CFG{prod}="SFCFS60";
$CFG{servers_to_remove}=[ qw([10.198.89.251]:14250) ];
$CFG{systems}=[ qw(sys1 sys2) ];
$CFG{vcs_clusterid}=42076;
$CFG{vcs_clustername}="clus1";
```

単一 CP サーバーベースのフェンシングからサーバーベースのフェンシングに移行するためのサンプル応答ファイル

単一 CP サーバーベースのフェンシングからサーバーベースのフェンシング（1 つの CP サーバーと 2 つのコーディネータディスク）に移行するためのサンプル応答ファイルを次に示します。

```
$CFG{fencing_disks}=[ qw(emc_clariion0_62 emc_clariion0_65) ];
$CFG{fencing_dgname}="fendg";
$CFG{fencing_scsi3_disk_policy}="dmp";
$CFG{fencing_ncp}=2;
$CFG{fencing_ndisks}=2;
$CFG{fencing_option}=4;
$CFG{opt}{configure}=1;
$CFG{opt}{fencing}=1;
$CFG{prod}="SFCFSHA60";
$CFG{systems}=[ qw(sys1 sys2) ];
$CFG{vcs_clusterid}=42076;
$CFG{vcs_clustername}="clus1";
```

フェンシング設定間を移行するための応答ファイル変数

表 15-6 SFCFSHA に、フェンシング設定間を移行するために必要な情報を指定する応答ファイル変数の一覧を示します。

表 15-6 フェンシング設定間を移行するための特定の応答ファイル変数

変数	リスト/スカラー	説明
CFG {fencing_option}	スカラー	I/O フェンシングの設定モードを指定します。 ■ 1 — Coordination Point Server ベースの I/O フェンシング ■ 2 —コーディネータディスクベースの I/O フェンシング ■ 3 — 無効モード ■ 4 — クラスタがオンラインである場合のフェンシングの移行 (必須)
CFG {fencing_reusedisk}	スカラー	ディスクベースのフェンシング、またはコーディネータディスクを使うサーバーベースのフェンシングに移行する場合に、空きディスク、またはディスクグループに所属済みのディスクを使うかどうかを指定します。 ■ 0 — 空きディスクをコーディネータディスクとして使う ■ 1 — ディスクグループに所属済みのディスクをコーディネータディスクとして使う(コーディネータディスクとしてこれらを設定する前に、インストーラによって所属先ディスクグループからディスクが削除される) (フェンシング設定がコーディネータディスクを使用している場合は、必須)
CFG {fencing_ncp}	スカラー	追加する新しいコーディネーションポイント数を指定します。 (必須)
CFG {fencing_ndisks}	スカラー	追加するコーディネーションポイントのディスク数を指定します。 (フェンシング設定がコーディネータディスクを使用している場合は、必須)

変数	リスト/スカラー	説明
CFG {fencing_disks}	リスト	追加するコーディネーションポイントのディスクを指定します。 (フェンシング設定がコーディネータディスクを使用している場合は、必須)
CFG {fencing_dgname}	スカラー	コーディネータディスクが属しているディスクグループを指定します。 (フェンシング設定がコーディネータディスクを使用している場合は、必須)
CFG {fencing_scsi3_disk_policy}	スカラー	ディスクが使用する必要があるディスクポリシーを指定します。 (フェンシング設定がコーディネータディスクを使用している場合は、必須)
CFG {fencing_cps}	リスト	追加するコーディネーションポイントの CP サーバーを指定します。 (サーバーベースフェンシングの場合は、必須)
CFG {fencing_cps_vips}{$vip1}	リスト	新しい CP サーバーの仮想 IP アドレスまたは完全修飾ホスト名を指定します。 (サーバーベースフェンシングの場合は、必須)
CFG {fencing_cps_ports}{$vip}	スカラー	新しい CP サーバーの仮想 IP がリスニングする必要があるポートを指定します。指定しない場合、デフォルト値は 14250 です。 (オプション)
CFG {servers_to_remove}	リスト	削除するコーディネーションポイントの CP サーバーを指定します。
CFG {disks_to_remove}	リスト	削除するコーディネーションポイントのディスクを指定します。
CFG{donotreconfigurevcs}	スカラー	VCS を再設定する必要があるかどうかを定義します。 (オプション)

変数	リスト/スカラー	説明
CFG{donotreconfigurefencing}	スカラー	フェンシングを再設定する必要があるかどうかを定義します。 (オプション)

優先フェンシングポリシーの有効化と無効化

I/O フェンシング設定で優先フェンシング機能を有効化または無効化できます。

優先フェンシングを有効にして、システムベース制御権獲得ポリシー、グループベース制御権獲得ポリシーまたはサイトベースポリシーを使うことができます。優先フェンシングを無効にした場合、I/O フェンシング設定ではデフォルトの数ベースの制御権獲得ポリシーが使われます。

マジョリティベース I/O フェンシングでは優先フェンシングは適用されません。

p.150 の「優先フェンシングについて」を参照してください。

p.150 の「優先フェンシングのしくみ」を参照してください。

I/O フェンシング設定で優先フェンシングを有効にするには

1 クラスタが I/O フェンシング設定を使って動作していることを確認します。

```
# vxfenadm -d
```

2 クラスタレベル属性 UseFence の値が SCSI3 に設定されていることを確認します。

```
# haclus -value UseFence
```

3 システムベース制御権獲得ポリシーを有効にするには、次の手順を実行します。

- VCS 設定を書き込み可能にします。

```
# haconf -makerw
```

- クラスタレベル属性 PreferredFencingPolicy の値を System に設定します。

```
# haclus -modify PreferredFencingPolicy System
```

- クラスタ内の各ノードにシステムレベル属性 FencingWeight の値を設定します。たとえば、2 ノードクラスタの sys1 に sys2 の 5 倍の重みを割り当てる場合は、次のコマンドを実行します。

```
# hasys -modify sys1 FencingWeight 50
# hasys -modify sys2 FencingWeight 10
```

- VCS 設定を保存します。

```
# haconf -dump -makero
```

- フェンシングノードの重みを確認します。

```
# vxfenconfig -a
```

4 グループベース制御権獲得ポリシーを有効にするには、次の手順を実行します。

- VCS 設定を書き込み可能にします。

```
# haconf -makerw
```

- クラスタレベル属性 PreferredFencingPolicy の値を Group に設定します。

```
# haclus -modify PreferredFencingPolicy Group
```

- 各サービスグループのグループレベル属性 Priority の値を設定します。
たとえば、次のコマンドを実行します。

```
# hagrp -modify service_group Priority 1
```

親サービスグループには、子サービスグループと同等以下の優先度を割り当てるようにしてください。親サービスグループと子サービスグループが異なるサブクラスタでホストされている場合は、子サービスグループをホストするサブクラスタが優先されます。

- VCS 設定を保存します。

```
# haconf -dump -makero
```

5 サイトベース制御権獲得ポリシーを有効にするには、次の手順を実行します。

- VCS 設定を書き込み可能にします。

```
# haconf -makerw
```

- クラスタレベル属性 PreferredFencingPolicy の値を Site に設定します。

```
# haclus -modify PreferredFencingPolicy Site
```

- 各サイトのサイトレベル属性 Preference の値を設定します。

```
For example,
# hasite -modify Pune Preference 2
```

- VCS 設定を保存します。

```
# haconf -dump -makero
```

6 フェンシングドライバに現在設定されているノードフェンシングの重みを表示するには、次のコマンドを実行します。

```
# vxfenconfig -a
```

I/O フェンシング設定で優先フェンシングを無効にするには

1 クラスタが I/O フェンシング設定を使って動作していることを確認します。

```
# vxfenadm -d
```

2 クラスタレベル属性 UseFence の値が SCSI3 に設定されていることを確認します。

```
# haclus -value UseFence
```

3 優先フェンシングを無効にしてデフォルトの制御権獲得ポリシーを使うには、クラスタレベル属性 PreferredFencingPolicy の値を Disabled に設定します。

```
# haconf -makerw
# haclus -modify PreferredFencingPolicy Disabled
# haconf -dump -makero
```

I/O フェンシングのログファイルについて

特定のユーティリティに関連した情報については、適切なログファイルを参照してください。各ユーティリティまたはコマンドのログファイルの場所は次のとおりです。

- vxfen 開始および停止ログ: `/var/VRTSvcs/log/vxfen/vxfen.log`
- vxfenclearpre ユーティリティ: `/var/VRTSvcs/log/vxfen/vxfen.log`
- vxfenctl ユーティリティ: `/var/VRTSvcs/log/vxfen/vxfenctl.log*`
- vxfenswap ユーティリティ: `/var/VRTSvcs/log/vxfen/vxfenswap.log*`
- vxfentsthdw ユーティリティ: `/var/VRTSvcs/log/vxfen/vxfentsthdw.log*`

アスタリスク(*)は、コマンドの呼び出しごとに異なる数を表します。

SFCFSHA のグローバルクラスタの管理

ここでは、次のグローバルクラスタ管理タスクの手順について説明します。

- ファイアドリルの設定について
- ウィザードを使用するファイアドリルサービスグループの設定
  p.529 の「ファイアドリル設定ウィザードを使用するファイアドリルサービスグループの設定について」を参照してください。
- 正常なファイアドリルの確認
  p.530 の「正常なファイアドリルの確認」を参照してください。
- ファイアドリルスケジュールの作成
  p.531 の「ファイアドリルスケジュールの作成」を参照してください。

ファイアドリル設定ウィザードを使用するファイアドリルサービスグループの設定について

ファイアドリル設定ウィザードを使って、ファイアドリルを設定します。

このウィザードの具体的なタスクは次のとおりです。

- ファイアドリルの実行中に変更されたブロックを保存するキャッシュオブジェクトを作成します。これにより、ファイアドリルに必要なディスク領域とディスクスピンドルを最小限に抑えます。
- 実際のアプリケーショングループを模倣する VCS サービスグループを設定します。

このウィザードは、単一のディスクグループを含むアプリケーショングループのみを対象として動作します。ウィザードでは、アプリケーションの最初の RVG を設定します。アプリケーションに 1 つ以上の RVG がある場合、領域最適化スナップショットを作成し、参照として最初の RVG を使い、VCS を手動で設定する必要があります。

`fdsched` スクリプトを使って、サービスグループを対象とするファイアドリルのスケジュールを設定することもできます。

p.531 の「ファイアドリルスケジュールの作成」を参照してください。

ファイアドリル設定ウィザードの実行

ウィザードを実行するには

1. Volume Replicator セカンダリサイトで[RVG セカンダリファイアドリルウィザード(RVG Secondary Fire Drill Wizard)]を起動します。このサイトでは、アプリケーションサービスグループをオフライン化し、サービスグループをセカンダリとしてオンライン化しておきます。

   ```
   # /opt/VRTSvcs/bin/fdsetup
   ```

2. [ようこそ(Welcome)]画面を読み、Enter キーを押します。

3 ウィザードがグローバルサービスグループを識別します。ファイアドリル用のサービスグループの名前を入力します。

4 領域最適化スナップショットに使えるディスクグループのボリュームの一覧を確認します。スナップショット用に選択するボリュームを入力します。通常、複製するかどうかにかかわらず、アプリケーションで使われるすべてのボリュームを準備する必要があります。ボリュームがすべてそろっていないと、スナップショットが失敗する場合があります。

プロンプトが表示されたら Enter キーを押します。

5 スナップショットが存在する場合は、書き込みを保存するためのキャッシュサイズを入力します。キャッシュサイズは、ファイアドリルの実行中に変更が予想されるブロック数を十分保存できる容量が必要です。ただし、キャッシュは、一杯になると自動的に拡張されるよう設定されています。キャッシュを作成するディスクを入力します。

プロンプトが表示されたら Enter キーを押します。

6 コマンドが実行され、ファイアドリルが設定されます。

プロンプトが表示されたら Enter キーを押します。

アプリケーショングループとその関連リソースが作成されます。ファイアドリルグループとアプリケーション（Oracle など）用リソース、CFSMount および RVGSnapshot タイプのリソースも作成されます。

両サービスグループのアプリケーションリソースは、同じアプリケーション（この例では同じデータベース）を定義しています。ウィザードは、アプリケーションリソースの FireDrill 属性を 1 に設定し、実際のアプリケーションインスタンスおよびファイアドリルサービスグループが同時にオンラインの場合、エージェントが同時性違反を報告することのないようにします。

ファイアドリルサービスグループ内のローカル属性の設定について

ファイアドリル設定ウィザードは、リソースのローカル化された属性値を認識しません。アプリケーションサービスグループにローカル（システム単位の）属性値を持つリソースがある場合、ウィザードを実行した後にその値を手動で設定する必要があります。

正常なファイアドリルの確認

アプリケーションを実行していないノードで、ファイアドリルサービスグループをオンラインにします。ファイアドリルサービスグループがオンラインになることを確認します。この作業により、ディザスタリカバリソリューションが正しく設定され、プライマリサイトで障害（災害）が発生した場合は、運用サービスグループがセカンダリサイトにフェールオーバーすることが検証されます。

ファイアドリルサービスグループがオンラインにならない場合は、VCS エンジンログを確認して、問題のトラブルシューティングを行います。これにより、運用サービスグループで必要な対応措置を取ることができます。

/tmp/fd-servicegroup.pid にあるファイアドリルのログも表示できます。

ファイアドリルサービスグループの機能を検証したら、このグループを必ずオフラインにしてください。ファイアドリルサービスグループをオフラインにしなかった場合、運用環境での障害の原因になる可能性があります。たとえば、ファイアドリルサービスグループがオンラインになっているノードにアプリケーションサービスグループがフェールオーバーした場合、リソースが競合し、両方のサービスに障害が発生します。

ファイアドリルスケジュールの作成

fdsched スクリプトを使って、サービスグループを対象とするファイアドリルのスケジュールを設定することもできます。fdsched スクリプトは、クラスタ内で動作しているノードのうち、番号が最小値であるノードでのみ動作するように設計されています。スケジューラは、一定の間隔で hagrp -online *firedrill_group* -any コマンドを実行します。

ファイアドリルスケジュールを作成するには

1. crontab に /opt/VRTSvcs/bin/fdsched ファイルを追加します。
2. ファイアドリルの可用性を高めるには、クラスタの各ノードに fdsched ファイルを追加します。

16

クラスタ化された NFS の使用

この章では以下の項目について説明しています。

クラスタ化された NFS のしくみ

このクラスタ化された NFS 機能は、CFS を使用して複数のノードでマウントされている同じファイルシステムを、フェールオーバー時に機能を失うことなくこれらのノードの任意の組み合わせから NFS を介して共有できるようにします。NFS ロックサーバーのフェールオーバーは、古いノードによって解放され猶予期間中に新しいノードと通信するクライアントによって再生利用されているすべてのロックを含みます。

基本設計

基本設計は、VCS でノード間でフェールオーバーできる仮想 IP（VIP）リソースを管理し、これらのリソースの処理に使用する手順にコードを追加して NFS レベルの操作を適切に処理することです。 他の関係するリソースはすべて、参加しているすべてのノードでアクティブです。共有の領域に保存されるロックデータは仮想 IP リソースとのロック手順で管

理および使用され、すべてのロックがクライアントによって適切に再生利用されると同時に、不適切なロックが間違ったタイミングで行われないようにします。このような干渉ロックは、サービスの停止とVxFS カーネルコード内のロック処理という新しい機能を組み合わせることで防止されます。

カーネル内の新しいコードと通信するために、fsclustadm のコマンドはプライベートの ioctl 呼び出しにコマンドラインインターフェースを追加するように変更されました。

メモ: CNFS サーバーごとに少なくとも 1 つの VIP を設定する必要があります。

内部のクラスタ化された NFS の機能

この項では、クラスタ化された NFS ソリューションの一部であるトリガとアクションスクリプトの内部機能を説明します。

ローカル状態追跡ディレクトリには、NFS サーバーとのトランザクションがある各 NFS クライアント用のファイルが含まれます。CNFS のインストール後、cfsshare は、共有ストレージ内のリカバリフォルダからローカル状態追跡ディレクトリへのシンボリックリンクを作成します。これによって、ノードが停止している場合でも、クラスタノード用のロック状態ファイルにクラスタ内の他のノードからアクセスできます。

次の表は、ローカル状態追跡ディレクトリと、対応する共有ストレージ内のリカバリフォルダを示しています。

プラットフォーム	NFS バージョン	共有ストレージの Recovery_folder（シンボリックリンクのリンク元）	ローカル状態追跡ディレクトリ（シンボリックリンクのリンク先）
RHEL	NFSv3	/locks/sm/nodename/sm	/var/lib/nfs/statd/sm
	NFSv4 Linux カーネル 3.10 より下位の場合	/locks/sm/nodename/v4recovery	/var/lib/nfs/v4recovery
	NFSv4 Linux カーネル 3.10 より上位の場合	/locks/sm/nodename/nfsdcltrack	/var/lib/nfs/nfsdcltrack

プラットフォーム	NFS バージョン	共有ストレージの Recovery_folder（シンボリックリンクのリンク元）	ローカル状態追跡ディレクトリ（シンボリックリンクのリンク先）
SUSE	NFSv3	/locks/sm/nodename/sm	/var/lib/nfs/sm
	NFSv4 Linux カーネル 3.10 より下位の場合	/locks/sm/nodename/v4recovery	/var/lib/nfs/v4recovery
	NFSv4 Linux カーネル 3.10 より上位の場合	/locks/sm/nodename/nfsdcltrack	/var/lib/nfs/nfsdcltrack

cfsshare config -n オプションは、cfsshare が recovery_folders へのシンボリックリンクを作成されるのをユーザーが望まない場合に使用できます。このオプションが使用された場合、ユーザーは手動でシンボリックリンクを作成する必要があります。設定オプションは VCS 設定にこの共有ファイルシステムを追加し、cfsnfssg と呼ばれる並列サービスグループに、対応する CFSMount リソースを作成します。cfsnfssg サービスグループに NFS リソースも作成します。これに加えて、新しい ApplicationNone 型の別のリソースが lockd および statd デーモンを監視するために作成されます。

preonline トリガ

preonline スクリプトは、nfs サーバーが作成したロック状態のファイルを共有ディスクにコピーします。

preonline スクリプトは IP フェールオーバーまたはノードの障害発生時に以下を行います。

- IP およびそれが最後にオンラインだったノードを見つけます。
- IP が次にオンラインになるノードを見つけます。
- 共有ストレージのリカバリフォルダが空かどうかを確認します。空の場合、ロックはないため終了します。nfslocking サービスを再起動する必要はありません。
- /opt/VRTS/bin/fsclustadm frlpause_enable と /opt/VRTS/bin/fsclustadm frlock_pause を呼び出して、ファイルシステムがフェールオーバー中に新しいロックを割り当てないようにします。
- すべてのノードでロックおよびステータスサービスを停止してロックの認可を防ぎます。
- すべてのファイルを */locks/sm/lastonline/recovery_folder/* から */locks/sm/nextonline/recovery_folder* / ディレクトリにコピーします。
  ここで、*locks* は、ロック情報を格納するために作成されるファイルシステムです。

lastonline は、VIP リソースが以前にオンラインだったノードです。
nextonline は、VIP リソースが次にオンラインになるノードです。
recovery_folder は、共有ストレージ上のフォルダです。

- `/opt/VRTS/bin/fsclustadm frlock_resume` を呼び出してロックの割り当てを再開します。

メモ: preonline トリガの最後に、すべてのロックサービスがすべてのノードで停止し、再起動されるまで NFS ロックを要求することはできません。

postonline トリガ

各 VIP の `postonline` スクリプトは IP フェールオーバーまたはノードの障害発生時に以下を行います。

- リカバリフォルダが空かどうかを確認します。空の場合、ロックサービスを再起動する必要はありません。
- すべてのノードでロックサービス、トリガの再生利用、および猶予モードを開始します。
- ステータスモニタの再起動によって状態ディレクトリのすべてのロックステータスファイルをスキャンし、すべてのノードと通信してロックを再生利用します。状態ファイルは処理された後で削除され、再生利用メッセージが適宜送信されます。
- ロックサーバーは猶予モードに入り、クライアントに猶予期間中のロックの回復のみを許可します。猶予期間中に新しいロックを割り当てることはありません。

postoffline トリガ

`postoffline` スクリプトは IP フェールオーバーで以下を行います

- `/opt/VRTS/bin/fsclustadm frlpause_disable` を呼び出して内部の使用カウンタを減らします。
- カーネルは内部カウンタを保持してシステムでアクティブな IP アドレスの数を追跡するので、`/opt/VRTS/bin/fsclustadm frlpause_enable` への各呼び出しは `/opt/VRTS/bin/fsclustadm frlpause_disable` への呼び出しと対応付ける必要があります。システムにアクティブな IP がない場合は、無効モードになります。

メモ: このトリガは管理上開始されたフェールオーバーの場合のみ呼び出されます。実際のシステム障害と再ブートはこの段階で処理されているローカル状態を破棄します。これは以前に VIP をホストしていたノードで呼び出される 1 つのトリガで、他はテイクオーバーするサーバーで呼び出されます。

アクション

- 各ノードに、/opt/VRTSvcs/bin/IP/actions/nfscfs ファイルがインストールされます。このファイルは指定のノードで NFS ロックデーモンを起動および停止するために使われます。action スクリプトは、トリガからのリモートコマンド実行用の rsh、ssh、または hacli を使用する代わりに使用されます。
- 各ノードに、/opt/VRTSvcs/bin/ApplicationNone/actions/nfscfsapp ファイルがインストールされます。このファイルは、cfsshare config と cfsshare unconfig を使って、クラスタ化された CNFS ソリューションを設定および設定解除するときに使われます。

使用例

この項では、2 つの使用例のシナリオを説明します。

- これはアクティブ/パッシブ設定であるため、NFS クライアントは 2 つの CNFS ノードをまたがって負荷分散されます。このために DNS のラウンドロビンを使用できます。
- NFS クライアントは CNFS ノードの 1 つに接続され、CNFS VIP フェールオーバー（サーバーエラーによる）は NFS クライアントに対して合理的かつ透過的です。これは、NFS/NFSRestart エージェントを使ったアクティブ/パッシブ NFS のクラスタ化よりも適切です。

cfsshare のマニュアルページ

このクラスタ化された NFS 機能は、cfsshare と呼ばれる新しい設定ユーティリティを VRTScavf パッケージに追加し、さらに、並列 NFS サーバーリソースを管理するための VCS 設定に追加される複数のスクリプトを追加します。cfsshare コマンドは cfsmntadm のような他のユーティリティによって作成された VCS のリソースを変更します。

cfsshare（1M）のマニュアルページを参照してください。

クラスタ化された NFS の設定および設定解除

この項では、クラスタ化された NFS を設定および設定解除する方法を説明します。

クラスタ化された NFS の設定

CNFS ソリューションは、すべてのクラスタノードでマウントされる、/*locks* などの共有ファイルシステムを必要とします。このファイルシステムはデータファイルシステムではなく、CNFS サーバー上で locks を保持している NFS クライアントに対応するロック状態ファイルが含まれます。

```
cfsshare config -p nfs [-n] shared_disk_groupshared_volumemount_point
```

メモ: 指定された `shared_volume` が VCS にすでに登録されている場合、***cfsshare config*** コマンドは失敗します。次のコマンドの出力を調べて、***shared_volume*** が VCS に登録されていないことを確認してください。

`/opt/VRTS/bin/cfsmntadm display`

CIFS がクラスタですでに設定されている場合は、クラスタ化された NFS の設定に同じ ***shared_volume*** および ***mount_point*** を指定します。

---

cfsshare config -n オプションを実行する場合は、次の手順を実行する必要があります。

1 各ノードで、locks ディレクトリ内に次のディレクトリを作成します（存在しない場合）。

NFSv3	# mkdir -p /locks/sm/<nodename>/sm
NFSv4	# mkdir -p /locks/sm/<nodename>/v4recovery
（Linux カーネル 3.10 より前のバージョンの場合）	
NFSv4	# mkdir -p /locks/sm/<nodename>/nfsdcltrack
（Linux カーネル 3.10 より後のバージョンの場合）	

2 各クラスタノードで、最初に古いディレクトリを移動してから、シンボリックリンクを作成します。

RHEL およびサポート対象の RHEL 互換配布の場合:

NFSv3	# mv /var/lib/nfs/statd/sm.bak /var/lib/nfs/statd/OLD.sm
	# ln -sf /locks/sm/nodename/sm /var/lib/nfs/statd/
NFSv4	# mv /var/lib/nfs/v4recovery /var/lib/nfs/OLD.v4recovery
（Linux カーネル 3.10 より前のバージョンの場合）	# ln -sf /locks/sm/nodename/v4recovery /var/lib/nfs/
NFSv4	# mv /var/lib/nfs/nfsdcltrack /var/lib/nfs/OLD.nfsdcltrack
（Linux カーネル 3.10 より後のバージョンの場合）	# ln -sf /locks/sm/nodename/nfsdcltrack /var/lib/nfs/

SUSE の場合:

NFSv3	# mv /var/lib/nfs/sm.bak /var/lib/nfs/OLD.sm
	# ln -sf /locks/sm/nodename/sm /var/lib/nfs/
NFSv4	# mv /var/lib/nfs/v4recovery /var/lib/nfs/OLD.v4recovery
（Linux カーネル 3.10 より前のバージョンの場合）	# ln -sf /locks/sm/nodename/v4recovery /var/lib/nfs/
NFSv4	# mv /var/lib/nfs/nfsdcltrack /var/lib/nfs/OLD.nfsdcltrack
（Linux カーネル 3.10 より後のバージョンの場合）	# ln -sf /locks/sm/nodename/nfsdcltrack /var/lib/nfs/

3 いずれかのクラスタノードで次のコマンドを実行して、/*locks*/sm の所有者、グループ、および権限を設定します。

```
# chown -R root:root /locks/sm
# chmod -R 755 /locks/sm
```

4 各ノードで次のコマンドを実行し、ユーザー権限を設定します。

RHEL およびサポート対象の RHEL 互換配布の場合:

chown -h rpcuser:rpcuser /var/lib/nfs/statd/sm

SUSE の場合:

chown -h rpcuser:rpcuser /var/lib/nfs/sm

`cfsshare config -p all` コマンドを実行して、CNFS と CIFS の両方を同時に設定できます。

```
# cfsshare config -p all -m user -l /var/run ¥
-c /etc/samba/smb.conf -t /usr shared_diskgroup_nameshared_volume /
mount_point
```

各種の CIFS 関連のオプションに関する説明については、Common Internet File System の章を参照してください。

サービスグループ cfsnfssg_dummy

CNFS 設定の一部として、`cfsnfssg_dummy` と呼ばれるサービスグループが作成されます。このサービスグループは通常オフラインです。

VCS（Cluster Server）で作成できるサービスグループの数には制限があります。この制限に達した場合、`cfsnfssg_dummy` は、リソースが `cfsshare unshare` および `cfsshare delete` 操作中に作成されるサービスグループとして機能します。

GroupLimit 属性について詳しくは、『Cluster Server 管理者ガイド』を参照してください。

クラスタ化された NFS の設定解除

このコマンドは設定段階のすべての手順を元に戻すために使われます。

```
cfsshare unconfig -p nfs
```

メモ: 共有中の CFS ファイルシステムがある場合、または仮想 IP が追加された場合は、`cfsshare unconfig` コマンドは失敗します。

クラスタ化された NFS の管理

この項では、クラスタ化された NFS のシナリオを説明します。

p.544 の「クラスタ化された NFS の設定例」を参照してください。

p.548 の「main.cf ファイル例」を参照してください。

NFS 共有 CFS ファイルシステムの表示

このコマンドは現在クラスタノードによって NFS 共有されている CFS ファイルシステムを表示します。

```
cfsshare display
```

VCS に以前に追加された CFS ファイルシステムの共有

メモ: `-N` オプションを使って、NFS 共有オプションを指定できます。

`-p nfs` を使って、使用するプロトコルが NFS であることを指定することもできます。

詳しくは、`cfsshare`(1M)マニュアルページを参照してください。

このコマンドを実行する前に、ユーザーは `cfsmntadm` コマンドを実行して共有ファイルシステムを VCS 設定に追加し、`cfsmount` コマンドを実行して共有ファイルシステムを *mount_point* でマウントする必要があります。これらのコマンドが実行されると、*mount_point* に対応する CFSMount リソースがデフォルトのサービスグループ(`vrts_vea_cfs_int_cfsmount`*number* のような名前)またはユーザーが指定した別のサービスグループで作成されます。

`cfsshare share` コマンドは、*mount_point* に対応する CFSMount リソースおよび関連付けられた CVMVolDg リソースを(`config` オプションを使用して作成された)cfsnfssg サービスグループに移動します。さらに、このコマンドは同じ cfsnfssg サービスグループ内の CFSMount リソースの上に共有リソースを作成します。

```
cfsshare share mount_point [share_options]
```

```
cfsshare share -p nfs [-N nfs_share_options ] mount_point
```

メモ: VCS にはサービスグループ間でリソースを移動する機能がありません。`cfsshare` コマンドは、cfsnfssg サービスグループに新しい CFSMount および CVMVolDg リソースを作成し、元のサービスグループから対応するリソースを削除します。

新しく作成されるリソースの名前は元のリソースの名前とは異なります。

以前の共有 CFS ファイルシステムの共有解除

このコマンドを実行する前に、ユーザーは cfsshare share コマンドを実行する必要があります。

cfsshare unshare コマンドを使って、ユーザーは *mount_point* でマウントされているファイルシステムの共有を停止できます。このコマンドは、*mount_point* に対応する Share、CFSMount、および CVMVolDg リソースを cfsnfssg サービスグループから新しく作成されるサービスグループに移動します。Share リソースはオフラインになり、その後削除されます。

```
cfsshare unshare [-p nfs] <mount_point>
```

メモ: VCS にはサービスグループ間でリソースを移動する機能がありません。cfsshare コマンドは、新しく作成されるサービスグループに新しい CFSMount および CVMVolDg リソースを作成し、元のサービスグループから対応するリソースを削除します。

新しく作成されるリソースの名前は元のリソースの名前とは異なります。

cfsmntadm delete コマンドを実行しても ActivationMode 属性は削除されません。ディスクグループ内のボリュームまたはボリュームセットが VCS 設定にない場合は、cfsdgadm delete を使用してこの ActivationMode 属性を削除する必要があります。

NFS 共有 CFS ファイルシステムの VCS への追加

このコマンドは、cfsnfssg のサービスグループ内の VCS 設定に CFS ファイルシステムを追加し、*mount_point* でファイルシステムをマウントします。NFS は CFS ファイルシステムを共有します。

```
cfsshare add [-D] shared_disk_group shared_volume mount_point ¥
[share_options] node_name=[mount_options]...

cfsshare add [-D] shared_disk_group shared_volume mount_point ¥
[share_options] all=[mount_options]

cfsshare add -p nfs [-D] [-N nfs_share_options] shared_disk_group ¥
shared_volume mount_point <node_name=[mount_options]>

cfsshare add -p nfs [-D] [-N nfs_share_options] shared_disk_group ¥
shared_volume mount_point all=[mount_options]
```

VCS からの NFS 共有 CFS ファイルシステムの削除

このコマンドを実行する前に、ユーザーは cfsnfssg サービスグループ内の必須リソース（Share、CFSMount、および必要な場合は CVMVolDg）を作成するための cfsshare add コマンドを実行する必要があります。

このコマンドは *mount_point* でマウントされている CFS ファイルシステムの共有を解除し、CFS ファイルシステムのマウントを解除し、VCS 設定から CFS ファイルシステムを削除します。

```
cfsshare delete [-p nfs] <mount_point>
```

VCS への仮想 IP アドレスの追加

このコマンドは、指定したネットワークデバイス用の NIC リソースと仮想 IP アドレス用の IP リソースを含む新しい非並列/フェールオーバーサービスグループを作成するために使われます。

```
cfsshare addvip [ -a nodename ] network_interfaceaddress netmask
```

`cfsshare addvip` コマンドを使用して指定できるのは、すべてのクラスタノード上に存在すると仮定される 1 つのネットワークインターフェースのみです。異なるクラスタノードに異なるネットワークインターフェースを指定する場合は、特定の VCS コマンドを実行する必要があります。次に例を示します。

```
# haconf -makerw
# hares -local vip1 Device
# hares -modify vip1 Device eth1 -sys sys1
# hares -modify vip1 Device eth2 -sys sys2
# hares -local nic1 Device
# hares -modify nic1 Device eth1 -sys sys1
# hares -modify nic1 Device eth2 -sys sys2
# haconf -dump -makero
```

ここで、*vip1* は、`cfsshare addvip` コマンドによって作成される仮想 IP リソースです。

ここで、*nic1* は、`cfsshare addvip` コマンドによってそれぞれ作成される NIC リソースです。

sys1 および *sys2* はクラスタノードです。

VCS からの仮想 IP アドレスの削除

このコマンドは仮想 IP アドレスに対応する非並列/フェールオーバーサービスグループを削除するために使われます。

```
cfsshare deletevip address
```

ピュア IPv6 構成での VCS への IPv6 仮想 IP アドレスの追加

このコマンドを使用して、指定したネットワークデバイス用の NIC リソースと IPv6 仮想 IP アドレス用の IP リソースを含む新しい非並列/フェールオーバーサービスグループを作成します。

```
cfsshare addvipv6 [ -a nodename ] network_interfaceipv6_address ¥
prefixlen [networkhosts]
```

ピュア IPv6 構成での VCS からの IPv6 仮想 IP アドレスの削除

このコマンドを使用して、IPv6 仮想 IP アドレスに対応する非並列/フェールオーバーサービスグループを削除します。

```
cfsshare deletevipv6 ipv6_address
```

デュアルスタック構成での VCS への仮想 IP アドレスの追加

クライアントが IPv4 または IPv6 ネットワーク経由で CNFS 共有にアクセスできるように、IPv4 と IPv6 仮想 IP を追加するには、次のコマンドを使用します。

IPv6 の場合:

```
cfsshare addvipv6 [ -a nodename ] network_interface ipv6_address ¥
prefixlen [networkhosts]
```

IPv4 の場合:

```
cfsshare addvip [ -a nodename ] network_interface ipv4_address ¥
netmask [networkhosts]
```

デュアルスタック構成での VCS からの仮想 IP アドレスの削除

IPv4 と IPv6 仮想 IP アドレスに対応する非並列/フェールオーバーサービスグループを削除するには、このコマンドを使用します。

```
cfsshare deletevip ipv4_address
```

```
cfsshare deletevipv6 ipv6_address
```

NFS 共有と関連付けられている共有オプションの変更

この項では、NFS 共有と関連付けられている共有オプションを変更する方法を説明します。

NFS 共有と関連付けられている共有オプションを変更するには

1 クラスタ内のいずれかのノードで、cfsshare unshare を実行してファイルシステムの共有を解除します。

```
# cfsshare unshare mount_point
```

2 クラスタ内のいずれかのノードで、cfsshare share を実行して、ファイルシステムを目的の共有オプションを使って共有します。

```
# cfsshare share -p nfs mount_pointshare_options
```

メモ: cfsshare unshare 操作は、*mount_point* ファイルシステムをマウントした可能性がある NFS クライアントに影響を及ぼすことがあります。

ファイルシステムチェックポイントの共有

この項では、ファイルシステムチェックポイントを共有する方法を説明します。

ファイルシステムチェックポイントを共有するには

1 VCS の設定にチェックポイントを追加するには、次のように入力してください。

```
# cfsmntadm add ckptckptnamemntpt_of_fsmntpt_of_checkpoint ¥
all=cluster,rw
```

ここで、*cktpname* チェックポイント名です。

mntpt_of_fs はファイルシステムのマウントポイント名です。

mntpt_of_checkpoint はチェックポイントのマウントポイントです。

2 チェックポイントをマウントするには、次のように入力してください。

```
# cfsmount mntpt_of_checkpoint
```

3 cfsshare share コマンドを実行して、このチェックポイントを共有します。

```
# cfsshare share -p nfs mntpt_of_checkpoint
```

クラスタ化された NFS の設定例

以下にクラスタ化された NFS の設定例を 2 つ示します。

メモ: 共有ボリュームとVxFSファイルシステムを持つ共有ディスクグループが設定されていることを確認してください。

cfsshare コマンドの使用例 1: CFS ファイルシステムを VCS 設定に追加

この例は、cfsshare コマンドを使用して CFS ファイルシステムを VCS 設定に追加し、マウントします。次に、NFS を通して共有し、共有解除し、マウント解除し、VCS 設定から CFSのファイルシステムを削除します。

クラスタ化された NFS を設定するには（例 1）

1 CFS/CVM の VCS オプションを設定します。次のように入力してください。

```
# cfscluster config
```

2 CNFS コンポーネントを設定します。次のように入力してください。

```
# cfsshare config -p nfs shared_disk_groupshared_volumemount_point
```

次に例を示します。

```
# cfsshare config -p nfs cfsdg vollocks /locks
```

3 NFS 共有 CFS ファイルシステムを VCS 設定に追加してマウントします。次のように入力してください。

```
# cfsshare add [-D] shared_disk_groupshared_volumemount_point ¥
[share_options] all=[mount_options]
```

次に例を示します。

```
# cfsshare add cfsdg vol1 /mnt1 all=rw
```

4 ユーザーが共有 CFS ファイルシステムにアクセスするための 仮想 IP アドレスを追加します。次のように入力してください。

```
# cfsshare addvip [ -a nodename ] ¥
network_interfaceaddressnetmasknetworkhosts
```

次に例を示します。

```
# cfsshare addvip eth0 ¥
10.182.111.161 255.255.240.0 10.182.111.1
```

5 以前に追加された 仮想 IP アドレスを設定から削除します。次のように入力してください。

```
# cfsshare deletevip address
```

次に例を示します。

```
# cfsshare deletevip 10.182.111.161
```

6 CFS ファイルシステムを共有解除し、マウント解除し、VCS 設定から削除します。次のように入力してください。

```
# cfsshare delete mount_point
```

次に例を示します。

```
# cfsshare delete /mnt1
```

7 CNFS コンポーネントを設定解除します。次のように入力してください。

```
# cfsshare unconfig -p nfs
```

cfsshare コマンドの使用例 2: 設定

この例は、cfsshare コマンドを使用して、この機能を設定および制御します。

クラスタ化された NFS を設定するには（例 2）

1 CFS/CVM の VCS オプションを設定します。次のように入力してください。

```
# cfscluster config
```

2 CNFS コンポーネントを設定します。次のように入力してください。

```
# cfsshare config -p nfs shared_disk_groupshared_volume ¥
mount_point
```

次に例を示します。

```
# cfsshare config -p nfs cfsdg vollocks /locks
```

3 CFS ファイルシステムを VCS 設定に追加してマウントします。次のように入力してください。

```
# cfsmntadm add [-D] shared_disk_groupshared_volumemount_point ¥
[service_group] all=[mount_options]
# cfsmount mount_point
```

次に例を示します。

```
# cfsmntadm add cfsdg vol1 /mnt1 all=delaylog,largefiles
# cfsmount /mnt1
```

4 CFS ファイルシステムを共有します。次のように入力してください。

```
# cfsshare share mount_point [share_options]
```

次に例を示します。

```
# cfsshare share /mnt1 rw, no_root_squash
```

5 現時点でエクスポート済みのファイルシステムを表示するには、次のコマンドを実行します。

```
# cfsshare display
CNFS metadata filesystem : /locks
Protocols Configured : NFS
#RESOURCE     MOUNTPOINT      PROTOCOL   OPTIONS
share1        /mnt1           NFS        rw,no_root_squash
```

6 ユーザーが共有 CFS ファイルシステムにアクセスするための 仮想 IP アドレスを追加します。次のように入力してください。

```
# cfsshare addvip [ -a nodename ] ¥
network_interfaceaddressnetmasknetworkhosts
```

次に例を示します。

```
# cfsshare addvip eth0 ¥
10.182.111.161 255.255.240.0 10.182.111.1
```

7 以前に追加された 仮想 IP アドレスを設定から削除します。次のように入力してください。

```
# cfsshare deletevip address
```

次に例を示します。

```
# cfsshare deletevip 10.182.111.161
```

8 CFS ファイルシステムを共有解除します。次のように入力してください。

```
# cfsshare unshare mount_point
```

次に例を示します。

```
# cfsshare unshare /mnt1
```

9 CFS ファイルシステムをマウント解除し、VCS 設定から削除します。次のように入力してください。

```
# cfsumount mount_point
# cfsmntadm delete mount_point
```

次に例を示します。

```
# cfsumount /mnt1
# cfsmntadm delete /mnt1
```

10 NFS 共有 CFS ファイルシステムを設定解除します。次のように入力してください。

```
# cfsshare unconfig -p nfs
```

main.cf ファイル例

main.cf ファイルの例を以下に示します。

```
include "OracleASMTypes.cf"
include "types.cf"
include "ApplicationNone.cf"
include "CFSTypes.cf"
include "CVMTypes.cf"
include "Db2udbTypes.cf"
include "OracleTypes.cf"
include "SybaseTypes.cf"

cluster cfs782 (
```

```
        UserNames = { admin = ghiAhcHeiDiiGqiChf }
        Administrators = { admin }
        HacliUserLevel = COMMANDROOT
        )

system sys1 (
         )

system sys2 (
         )

system sys3 (
         )

system sys4(
        )


group cfsnfssg (
        SystemList = { sys1 = 0, sys2 = 1, sys3 = 2,
              sys4 = 3 }
        AutoFailOver = 0
        Parallel = 1
        AutoStartList = { sys1, sys2, sys3,
              sys4 }
        )

        Application Samba_winbind (
              StartProgram = "/opt/VRTSvcs/bin/ApplicationNone/
              winbindmonitor.sh start"
              StopProgram = "/opt/VRTSvcs/bin/ApplicationNone/
              winbindmonitor.sh stop"
              PidFiles = { "/var/run/winbindmonitor.pid" }
              )

        ApplicationNone app (
              )

        CFSMount cfsmount3 (
              Critical = 0
              MountPoint = "/mnt2"
              BlockDevice = "/dev/vx/dsk/fsp701-704-v03/vol2"
              NodeList = { sys1, sys2, sys3,
```

```
            sys4 }
      )

CFSMount cfsnfs_locks (
      Critical = 0
      MountPoint = "/lock"
      BlockDevice = "/dev/vx/dsk/fsp701-704-v03/lock"
      NodeList = { sys1, sys2, sys3,
            sys4 }
      )

CVMVolDg cvmvoldg3 (
      Critical = 0
      CVMDiskGroup = fsp701-704-v03
      CVMVolume = { lock, vol2 }
      CVMActivation @sys1 = sw
      CVMActivation @sys2 = sw
      CVMActivation @sys3 = sw
      CVMActivation @sys4 = sw
      )

NFS nfs (
      NFSv4Support = 1
      )

NetBios Samba_netbios (
      SambaServerRes = SambaServerResource
      NetBiosName = cfs782
      )

SambaServer SambaServerResource (
      ConfFile = "/opt/pware/lib/smb.conf"
      SambaTopDir = "/opt/pware"
      LockDir = "/opt/pware/var/locks"
      )

SambaShare sambashare1 (
      Critical = 0
      SambaServerRes = SambaServerResource
      ShareName = cifs1
      ShareOptions = "path=/mnt2;msdfs root=yes;msdfs
      'proxy=¥¥10.209.116.87¥¥cifs1_dfs¥¥"
      )
```

```
        requires group cvm online local firm
        Samba_winbind requires Samba_netbios
        cfsmount3 requires cfsnfs_locks
        cfsmount3 requires cvmvoldg3
        cfsnfs_locks requires cvmvoldg3
        sambashare1 requires SambaServerResource
        sambashare1 requires cfsmount3


        // resource dependency tree
        //
        //    group cfsnfssg
        //    {
        //    Application Samba_winbind
        //        {
        //        NetBios Samba_netbios
        //        }
        //    ApplicationNone app
        //    NFS nfs
        //    SambaShare sambashare1
        //        {
        //        SambaServer SambaServerResource
        //        CFSMount cfsmount3
        //            {
        //            CFSMount cfsnfs_locks
        //                {
        //                CVMVolDg cvmvoldg3
        //                }
        //            CVMVolDg cvmvoldg3
        //            }
        //        }
        //    }


    group cfsnfssg_dummy (
        SystemList = { sys1 = 0, sys2 = 1, sys3 = 2,
              sys4 = 3 }
        AutoFailOver = 0
        Parallel = 1
        AutoStartList = { sys1, sys2, sys3,
              sys4 }
        )
```

```
        requires group cvm online local firm


        // resource dependency tree
        //
        //    group cfsnfssg_dummy
        //    {
        //    }


group cvm (
        SystemList = { sys1 = 0, sys2 = 1, sys3 = 2,
              sys4 = 3 }
        AutoFailOver = 0
        Parallel = 1
        AutoStartList = { sys1, sys2, sys3,
              sys4 }
        )

        CFSfsckd vxfsckd (
              ActivationMode @sys1 = { fsp701-704-v03 = sw }
              ActivationMode @sys2 = { fsp701-704-v03 = sw }
              ActivationMode @sys3 = { fsp701-704-v03 = sw }
              ActivationMode @sys4 = { fsp701-704-v03 = sw }
              )

        CVMCluster cvm_clus (
              CVMClustName = cfs782
              CVMNodeId = { sys1 = 0, sys2 = 1,
                    sys3 = 2,
                    sys4 = 3 }
              CVMTransport = gab
              CVMTimeout = 200
              )

        CVMVxconfigd cvm_vxconfigd (
              Critical = 0
              CVMVxconfigdArgs = { syslog }
              )

        cvm_clus requires cvm_vxconfigd
        vxfsckd requires cvm_clus
```

```
        // resource dependency tree
        //
        //    group cvm
        //    {
        //    CFSfsckd vxfsckd
        //        {
        //        CVMCluster cvm_clus
        //            {
        //            CVMVxconfigd cvm_vxconfigd
        //            }
        //        }
        //    }


group vip1 (
        SystemList = { sys1 = 0, sys2 = 1, sys3 = 2,
              sys4 = 3 }
        AutoStartList = { sys1, sys2, sys3,
              sys4 }
        TriggerPath = "bin/cavftriggers/vip"
        TriggersEnabled @sys1 = { PREONLINE, POSTONLINE, POSTOFFLINE
}
        TriggersEnabled @sys2 = { PREONLINE, POSTONLINE, POSTOFFLINE
}
        TriggersEnabled @sys3 = { PREONLINE, POSTONLINE, POSTOFFLINE
}
        TriggersEnabled @sys4 = { PREONLINE, POSTONLINE, POSTOFFLINE
}
        PreOnline @sys1 = 1
        PreOnline @sys2 = 1
        PreOnline @sys3 = 1
        PreOnline @sys4 = 1
        )

        IP vip1 (
              Device = eth0
              Address = "10.209.116.87"
              NetMask = "255.255.252.0"
              )

        NIC nic1 (
```

```
        Device = eth0
        NetworkHosts = { "10.209.113.1" }
        )

SambaShare sambashare1_dfs (
        Critical = 0
        SambaServerRes = SambaServerResource
        ShareName = cifs1_dfs
        ShareOptions = "path=/mnt2;readonly=no"
        )

requires group cfsnfssg online local firm
sambashare1_dfs requires vip1
vip1 requires nic1


// resource dependency tree
//
//    group vip1
//    {
//    SambaShare sambashare1_dfs
//        {
//        IP vip1
//            {
//            NIC nic1
//            }
//        }
//    }
```

NFS クライアントで NFS エクスポートされたファイルシステムをマウントする方法

この項では、NFS クライアントで NFS エクスポートされたファイルシステムをマウントする方法を説明します。

NFS クライアントで NFS エクスポートされたファイルシステムをマウントするには

- 次のコマンドを実行します。

```
# mount -t nfs -o vers=NFS_version
VIP_address:remote_filesystem mount_point
```

クラスタ化された NFS のデバッグ

cfsshare コマンドはエラーメッセージを /var/VRTSvcs/log ディレクトリ内の VCS ログに記録します。fsclustadm frlpause_query コマンドはグローバルな状態の現在のローカルコピーを表示するために使用できます。これはクラスタ化された NFS 機能の問題のデバッグに役立つことがあります。

17

Common Internet File System の使用

この章では以下の項目について説明しています。

CIFS について

CIFS（Common Internet File System）機能により、Windows クライアントがアクセスできる CIFS プロトコルを使って CFS ファイルシステムを共有できます。ノード障害の発生時またはサービスグループのフェールオーバー時でも、CIFS 共有は他のクラスタノードによって引き続き処理されます。

CIFS 共有を設定するには、`cfsshare` コマンドを使います。

p.557 の「cfsshare のマニュアルページ」を参照してください。

CIFS の必要条件

- Common Internet File System（CIFS）では、Samba バージョン 3.2 以降が必要です。
- Samba の予備知識があることが前提条件です。

Samba のしくみ

Samba は、UNIX サーバーが Windows ネットワークに参加できるようにするネットワーキングツールです。Samba には、2 つの役割があります。1 つは、他の PC が使用できるようにファイルおよびプリンタを分配するサーバーとしての役割であり、もう 1 つは、UNIX システムが他の Windows サーバーまたは Samba サーバー上のファイルおよびプリンタにアクセスできるようにするクライアントユーティリティとしての役割です。

CFS のクラスタ化された NFS と CIFS の設定

CNFS と CIFS を同じクラスタで設定できます。ただし、所定の CFS ファイルシステムを共有できるのは、任意の時点で 1 つのプロトコル (NFS または CIFS) を使用している場合のみです。

cfsshare のマニュアルページ

この CIFS 機能は、VRTScavf RPM の `cfsshare` ユーティリティに新しい機能を追加し、さらに、並列 NFS および CIFS サーバーリソースを管理するための VCS 設定に追加される複数のスクリプトを追加します。`cfsshare` コマンドは `cfsmntadm` のような他のユーティリティによって作成された VCS のリソースを変更します。

メモ: `-p` オプションを使用しない場合、`cfsshare` コマンドはデフォルトで nfs プロトコルを採用します。

`cfsshare` (1M) マニュアルページを参照してください。

```
man -M /opt/VRTS/man cfsshare
```

user モードでの CIFS の設定

この項では、user モードで CIFS を設定する方法について説明します。

このモードでは、ユーザー認証がクラスタノード自体で実行されます。

NIS またはその他のメカニズムをクラスタノードで設定し、すべてのクラスタノードで同じユーザー/グループが同じユーザー/グループ ID を持つようにする必要があります。

設定時に共有ファイルシステムを指定する必要があります。このファイルシステムは、ユーザーの暗号化されたパスワードが含まれる smbpasswd ファイルを格納するために使われます。こうすることで、あるクラスタノードでパスワードが作成されたユーザーが、他のクラスタノードに対しても自身を認証できます。

aa-status コマンドを使用して apparmor サービスの状態を確認します。

smbd サービスプロファイルがロードされている場合は、次のコマンドを使用してプロファイルを無効にします。

```
aa-disable /usr/sbin/smbd
```

cfsshare config コマンドを実行する前に、既存の smb.conf ファイルをバックアップし、/var/log/samba ファイルがすべてのクラスタノードに存在していることを確認する必要があります。

メモ: CNFS がクラスタですでに設定されている場合は、CIFS の設定に同じ *shared_volume* および *mount_point* を指定します。

```
cfsshare config [-n] -p cifs -l samba_lockdir -c ¥
samba_config_file -t samba_topdir -m user ¥
shared_disk_groupshared_volumemount_point
```

次に例を示します。

```
# cfsshare config -p cifs -m user -l /var/run ¥
-c /etc/samba/smb.conf -t /usr lockdg vollocks /locks
```

メモ: smbd -b コマンドを使用して、Samba ロックディレクトリと PID ディレクトリのパスを確認します。PID ディレクトリのパスを -l *samba_lockdir* オプションに使います。

メモ: 指定された *shared_volume* が VCS にすでに登録されている場合、cfsshare config コマンドは失敗します。次のコマンドの出力を調べて、*shared_volume* が VCS に登録されていないことを確認してください。

/opt/VRTS/bin/cfsmntadm display

---

cfsshare config コマンドを使うときに -n オプションを指定する場合は、次の手順に従って CIFS 設定を完了します。

-n オプションを使うときに CIFS 設定を完了するには

1 次の行を smb.conf ファイルにコピーします。

```
security = user
passwd backend = smbpasswd
smbpasswd file = pvtdir/smbpasswd
```

pvtdir は、Samba インストールのプライベートディレクトリです。

2 次のコマンドを実行して既存の smbpasswd ファイルをバックアップします。

```
# cp -f pvtdir/smbpasswd pvtdir/smbpasswd.OLD
```

3 *pvtdir* に、locks ファイルシステムに作成した smbpasswd ファイルへのシンボリックリンクを作成します。

```
# ln -sf mntpt/cifs/smbpasswd pvtdir/smbpasswd
```

mntpt はマウントポイントです。

CIFS の設定を解除するには:

```
# cfsshare unconfig -p cifs
```

メモ: ファイルシステムまたは VIP が CIFS プロトコルを介して共有されている場合、設定解除操作は失敗します。CIFS を介したファイルシステムの共有を停止するには、cfsshare delete コマンドを使います。cfsshare deletevip コマンドを実行して、このファイルを削除します。

---

cfsshare config -p all コマンドを実行して、CNFS と CIFS の両方を同時に設定できます。

```
# cfsshare config -p all -m user -l /var/run ¥
-c /etc/samba/smb.conf -t /usr lockdg vollocks /locks
```

domain モードでの CIFS の設定

この項では、domain モードで CIFS を設定する方法について説明します。

このモードでは、ユーザー認証が NT4 形式のドメインコントローラで実行されます。

クラスタノードはドメイン内でメンバーサーバーとして動作します。winbind の作業を介してユーザーおよびグループをマッピングするには、追加の手順が実行済みである必要があります。

設定時に共有ファイルシステムを指定する必要があります。このファイルシステムは、すべてのクラスタノードに secrets.tdb ファイル (コンピュータパスワードファイル) をレプリケートするために使われます。1 つのクラスタノードだけがクラスタ名を使ってドメインに参加します。このファイルをすべてのクラスタノードにコピーすると、ドメインコントローラは、すべてのクラスタノードを 1 つのメンバーサーバーとして認識します。

共有ファイルシステムを使用して、すべてのクラスタノードで共有される必要がある tdb ファイルを格納することもできます。すべてのクラスタノードで適切なシンボリックリンクを作成する必要があります。

aa-status コマンドを使用して apparmor サービスの状態を確認します。

smbd サービスプロファイルがロードされている場合は、次のコマンドを使用してプロファイルを無効にします。

```
aa-disable /usr/sbin/smbd
```

cfsshare config コマンドを実行する前に、既存の smb.conf ファイルをバックアップし、/var/log/samba ファイルがすべてのクラスタノードに存在していることを確認する必要があります。

メモ: CNFS がクラスタですでに設定されている場合は、CIFS の設定に同じ *shared_volume* および *mount_point* を指定します。

```
cfsshare config [-n] -p cifs -l samba lockdir -c ¥
samba_config_file -t samba_topdir -m domain -d ¥
domain -s password_server -u domain_user ¥
shared_disk_groupshared_volumemount_point
```

次に例を示します。

```
# cfsshare config -p cifs -m domain -l ¥
/var/run -c /etc/samba/smb.conf -t /usr -s sfstest-ad ¥
-d SFSTEST-AD2 -u Administrator lockdg vollocks /locks
```

メモ: smbd -b コマンドを使用して、Samba ロックディレクトリと PID ディレクトリのパスを確認します。PID ディレクトリのパスを -l *samba_lockdir* オプションに使います。

メモ: 指定された *shared_volume* が VCS にすでに登録されている場合、cfsshare config コマンドは失敗します。次のコマンドの出力を調べて、*shared_volume* が VCS に登録されていないことを確認してください。

/opt/VRTS/bin/cfsmntadm display

---

cfsshare config コマンドを使うときに -n オプションを指定する場合は、次の手順に従って CIFS 設定を完了します。

-n オプションを使うときに CIFS 設定を完了するには

1 次の行を smb.conf ファイルにコピーします。

```
security = domain
workgroup = domainname
password server = Domain_Controller_of_the_domain
```

2 次のコマンドを実行して既存の secrets.tdb ファイルをバックアップします。

```
# mv -f pvtdir/secrets.tdb pvtdir/secrets.tdb.OLD
```

pvtdir は、Samba インストールのプライベートディレクトリです。

3 locks ファイルシステムに作成した secrets.tdb ファイルを Samba インストールのプライベートディレクトリにコピーします。

```
# cp -f mntpt/cifs/secrets.tdb pvtdir/secrets.tdb
```

mntpt はマウントポイントです。

CIFS の設定を解除するには:

◆ CIFS の設定解除:

```
# cfsshare unconfig -p cifs
```

メモ: ファイルシステムが CIFS プロトコルを介して共有されている場合、設定解除操作は失敗します。

CNFS と CIFS の両方を設定するには

◆ cfsshare config -p all コマンドを実行して、CNFS と CIFS の両方を同時に設定できます。

```
# cfsshare config -p all -m domain -l ¥
/var/run -c /etc/samba/smb.conf -t /usr -s sfstest-ad ¥
-d SFSTEST-AD2 -u Administrator lockdg vollocks /locks
```

ads モードでの CIFS の設定

この項では、ads モードで CIFS を設定する方法について説明します。

このモードでは、ユーザー認証が Kerberos を使って Active Directory で実行されます。クラスタノードはドメイン内でメンバーサーバーとして動作します。winbind の作業を介してユーザーおよびグループをマッピングするには、追加の手順が実行済みである必要があります。

設定時に共有ファイルシステムを指定する必要があります。このファイルシステムは、すべてのクラスタノードに secrets.tdb ファイル (コンピュータパスワードファイル) をレプリケートするために使われます。1 つのクラスタノードだけがクラスタ名を使ってドメインに参加します。このファイルをすべてのクラスタノードにコピーすると、ドメインコントローラは、すべてのクラスタノードを 1 つのメンバーサーバーとして認識します。

すべてのクラスタノードで Kerberos が設定済みである必要があります。すべてのクラスタノード上の時刻が AD サーバー/KDC と同期されている必要があります。

共有ファイルシステムを使用して、すべてのクラスタノードで共有される必要がある tdb ファイルを格納することもできます。すべてのクラスタノードで適切なシンボリックリンクを作成する必要があります。

aa-status コマンドを使用して apparmor サービスの状態を確認します。

smbd サービスプロファイルがロードされている場合は、次のコマンドを使用してプロファイルを無効にします。

```
aa-disable /usr/sbin/smbd
```

cfsshare config コマンドを実行する前に、既存の smb.conf ファイルをバックアップし、/var/log/samba ファイルがすべてのクラスタノードに存在していることを確認する必要があります。

メモ: CNFS がクラスタですでに設定されている場合は、CIFS の設定に同じ *shared_volume* および *mount_point* を指定します。

```
cfsshare config [-n] -p cifs -l samba _lockdir -c ¥
samba_config_file -t samba_topdir -m ads ¥
-d domain -r realm -s ¥
password_server -u domain_user ¥
shared_disk_groupshared_volumemount point
```

次に例を示します。

```
# cfsshare config -p cifs -m ads -l /var/run ¥
-c /etc/samba/smb.conf -t /usr -s sfstest-ad -d ¥
SFSTEST-AD2 -r SFSTEST-AD2.LOCAL -u Administrator ¥
lockdg lockvol /locks
```

メモ: smbd -b コマンドを使用して、Samba ロックディレクトリと PID ディレクトリのパスを確認します。PID ディレクトリのパスを -l *samba_lockdir* オプションに使います。

メモ: 指定された ***shared_volume*** が VCS にすでに登録されている場合、cfsshare config コマンドは失敗します。次のコマンドの出力を調べて、***shared_volume*** が VCS に登録されていないことを確認してください。

/opt/VRTS/bin/cfsmntadm display

cfsshare config コマンドを使うときに -n オプションを指定する場合は、次の手順に従って CIFS 設定を完了します。

-n オプションを使うときに CIFS 設定を完了するには

1 次の行を smb.conf ファイルにコピーします。

```
security = ads
workgroup = domainname
password server = AD_server_of_the_domain
realm = realm_name
```

2 次のコマンドを実行して既存の secrets.tdb ファイルをバックアップします。

```
# mv -f pvtdir/secrets.tdb pvtdir/secrets.tdb.OLD
```

pvtdir は、Samba インストールのプライベートディレクトリです。

3 locks ファイルシステムに作成した secrets.tdb ファイルを Samba インストールのプライベートディレクトリにコピーします。

```
# cp -f mntpt/cifs/secrets.tdb pvtdir/secrets.tdb
```

mntpt はマウントポイントです。

CIFS の設定を解除するには:

```
# cfsshare unconfig -p cifs
```

メモ: ファイルシステムが CIFS プロトコルを介して共有されている場合、設定解除操作は失敗します。

cfsshare config -p all コマンドを実行して、CNFS と CIFS の両方を同時に設定できます。

```
# cfsshare config -p all -m ads -l /var/run ¥
-c /etc/samba/smb.conf -t /usr -s sfstest-ad -d ¥
```

```
SFSTEST-AD2 -r SFSTEST-AD2.LOCAL -u Administrator ¥
lockdg lockvol /locks
```

CIFS の管理

Windows クライアントから CIFS エクスポートされたファイルシステムにアクセスできるようにするには、最初に仮想 IP を追加する必要があります。CIFS を介してファイルシステムを共有している間にこの仮想 IP を指定する必要があります。

仮想 IP の追加:

```
cfsshare addvip [ -a nodename ] deviceaddressnetmask [networkhosts]
```

次に例を示します。

```
# cfsshare addvip eth0 10.182.79.216 ¥
255.255.240.0 10.182.79.215
```

cfsshare addvip コマンドを使用して指定できるのは、すべてのクラスタノード上に存在すると仮定される 1 つのネットワークインターフェースのみです。異なるクラスタノードに異なるネットワークインターフェースを指定する場合は、特定の VCS コマンドを実行する必要があります。次に例を示します。

```
# haconf -makerw
# hares -local vip1 Device
# hares -modify vip1 Device eth1 -sys sys1
# hares -modify vip1 Device eth2 -sys sys2
# hares -local nic1 Device
# hares -modify nic1 Device eth1 -sys sys1
# hares -modify nic1 Device eth2 -sys sys2
# haconf -dump -makero
```

ここで、*vip1* は、cfsshare addvip コマンドによって作成される仮想 IP リソースです。

ここで、*nic1* は、cfsshare addvip コマンドによってそれぞれ作成される NIC リソースです。

sys1 および *sys2* はクラスタノードです。

CIFS を介したファイルシステムの追加と共有:

```
cfsshare add -p cifs [-D] -v address -n cifs_share_name ¥
shared_disk_groupshared_volumemount_point ¥
share_options all=[ mount_options ]
```

次に例を示します。

```
# cfsshare add -p cifs -v 10.182.79.216 ¥
-n sh1 sharedg vol1 /mnt1 "readonly=no" all=
```

メモ: -C オプションを使って、CIFS 共有オプションを指定できます。

詳しくは、cfsshare（1M）マニュアルページを参照してください。

現時点でエクスポート済みのファイルシステムを表示するには、次のコマンドを実行します。

```
# cfsshare display

CNFS metadata filesystem : /locks
Protocols Configured : CIFS
#RESOURCE     MOUNTPOINT      PROTOCOL   OPTIONS
sambashare1   /mnt1           CIFS       path=/mnt1;readonly=no
```

以前の共有ファイルシステムの削除:

```
cfsshare delete -p cifs mount_point
```

次に例を示します。

```
# cfsshare delete -p cifs /mnt1
```

以前追加した VIP の削除:

```
cfsshare deletevip address
```

次に例を示します。

```
# cfsshare deletevip 10.182.79.216
```

VCS に以前に追加された CFS ファイルシステムの共有

次のコマンドのいずれかを使います。

```
cfsshare share -p cifs -v address -n cifs share name ¥
[-C cifs_share_options] mount_point
```

次に例を示します。

```
# cfsshare share -p cifs -v 10.182.79.216 -n sh1 -C readonly=no /mnt1
```

または

```
cfsshare share -p cifs -v address -n cifs share name ¥
mount_point [share_options]
```

次に例を示します。

```
# cfsshare share -p cifs -v 10.182.79.216 -n sh1 /mnt1 readonly=no
```

メモ: *cfsshare share* コマンドを実行する前に、`cfsshare addvip` コマンドを使って `address` を追加する必要があります。

詳しくは、`cfsshare`(1M)マニュアルページを参照してください。

このコマンドを実行する前に、`cfsmntadm` コマンドを実行して共有ファイルシステムを VCS 設定に追加し、`cfsmount` コマンドを実行して共有ファイルシステムを *mount_point* でマウントする必要があります。これらのコマンドが実行されると、*mount_point* に対応する CFSMount リソースがデフォルトのサービスグループ (`vrts_vea_cfs_int_cfsmount`*number* のような名前)またはユーザーが指定した別のサービスグループで作成されます。

`cfsshare share` コマンドは、*mount_point* に対応する CFSMount リソースおよび関連付けられた CVMVolDg リソースを(`config` オプションを使用して作成された) cfsnfssg サービスグループに移動します。さらに、同じ cfsnfssg サービスグループ内の CFSMount リソースの上に共有リソースを作成します。

メモ: VCS にはサービスグループ間でリソースを移動する機能がありません。`cfsshare` コマンドは、cfsnfssg サービスグループに新しい CFSMount および CVMVolDg リソースを作成し、元のサービスグループから対応するリソースを削除します。

新しく作成されるリソースの名前は元のリソースの名前とは異なります。

VCS に以前に追加された CFS ファイルシステムの IPv4 から IPv6 への移行

メモ: ピュア IPv6 スタックを移行するには、少なくとも 1 つのプライベート IPv4 インターフェースアドレスが各ノードにあることを確認します。

1 以前の共有ファイルシステムを削除します。

```
# cfsshare delete -p cifs <mount_point>
```

2 以前に追加された IPv4 仮想 IP リソースを削除します。

```
# cfsshare deletevip <ipv4_ip_address>
```

3 新しい IPv6 仮想 IP アドレスを追加します。

```
# cfsshare addvipv6 [-a <nodename>]
<network_interface><ipv6_address><prefixlen> [networkhosts]
```

4 新しい IP アドレスを shambashare に追加します。

```
# cfsshare add -p cifs [-D] -v address -n cifs_share_name ¥
shared_disk_group shared_volume mount_point ¥
share_options all=[ mount_options ]
```

次に例を示します。

```
# cfsshare add -p cifs -v 2620:128:f0a2:9002:5231:1ddf:70e:5609
64 ¥
                    -n sh1 sharedg vol1 /mnt1 "readonly=no" all=
```

既存の共有へのデュアルスタックサポートの追加

1 既存の共有にデュアルスタックサポートを追加するには、次のいずれかのコマンドを使用します。

- IPv4 を使用して既存の共有をエクスポートする場合:

```
# cfsshare addvipdual
<ip_grp><network_interface><ipv6_address><prefixlen>
[networkhosts]
```

- IPv6 を使用して既存の共有をエクスポートする場合:

```
# cfsshare addvipdual
<ip_grp><network_interface><ipv4_address><netmask>
[networkhosts]
```

ここで、*ip_grp* は、共有が有効になっている既存の VIP リソースで、*network_interface* は、共有が有効になっている既存の VIP リソースインターフェースです。

2 dual_ip リソースを確認するには、次のコマンドを実行します。

```
# hares -list Type=IP Address=<ip_by_which_resource_is_shared>
```

dual_ip リソースを追加すると、*<{ip_grp}_dual>* としてリソーステーブルに表示されます。

既存の共有からのデュアルスタックサポートの削除

1 dual_ip リソースを表示するには、次のコマンドを実行します。

```
# hares -list Type=IP Address=<dual_ip_address>
```

ここで、*<dual_ip_address>* は、リソースを共有する IP です。

2 dual_ip リソースを削除します。

- Ipv4 を使用して既存の共有がエクスポートされており、次に IPv6 を使用して共有する場合:

```
# cfsshare deletevipv6 <ipv6_dualip_address>
```

- Ipv6 を使用して既存の共有がエクスポートされており、次に IPv4 を使用して共有する場合:

```
# cfsshare deletevip <ipv4_dualip_address>
```

デュアルスタックサポートを削除した後に、ピュア IPv6 に CFS ファイルシステムを移行したい場合があります。p.566 の「VCS に以前に追加された CFS ファイルシステムの IPv4 から IPv6 への移行」を参照してください。

以前の共有 CFS ファイルシステムの共有解除

このコマンドを実行する前に `cfsshare share` コマンドを実行しておく必要があります。

`cfsshare unshare` コマンドを使って、ユーザーは *mount_point* でマウントされているファイルシステムの共有を停止できます。このコマンドは、*mount_point* に対応する Share、CFSMount、および CVMVolDg リソースを cfsnfssg サービスグループから新しく作成されるサービスグループに移動します。SambaShare リソースはオフラインになり、その後削除されます。

```
cfsshare unshare mount_point
```

メモ: VCS にはサービスグループ間でリソースを移動する機能がありません。`cfsshare` コマンドは、新しく作成されるサービスグループに新しい CFSMount および CVMVolDg リソースを作成し、元のサービスグループから対応するリソースを削除します。

新しく作成されるリソースの名前は元のリソースの名前とは異なります。

`cfsmntadm delete` コマンドを実行しても `ActivationMode` 属性は削除されません。ディスクグループ内のボリュームまたはボリュームセットが VCS 設定にない場合は、`cfsdgadm delete` を使用してこの `ActivationMode` 属性を削除する必要があります。

CIFS 用 main.cf ファイルのサンプル

main.cf ファイルの例を以下に示します。

```
include "OracleASMTypes.cf"
include "types.cf"
include "ApplicationNone.cf"
include "CFSTypes.cf"
include "CVMTypes.cf"
include "Db2udbTypes.cf"
include "OracleTypes.cf"
include "SybaseTypes.cf"

cluster cfs782 (
      UserNames = { admin = ghiAhcHeiDiiGqiChf }
      Administrators = { admin }
      HacliUserLevel = COMMANDROOT
      UseFence = SCSI3
      )

system sys1 (
        )

system sys2 (
        )

system sys3 (
        )

system sys4(
      )


group cfsnfssg (
      SystemList = { sys1 = 0, sys2 = 1, sys3 = 2,
            sys4 = 3 }
      AutoFailOver = 0
      Parallel = 1
      AutoStartList = { sys1, sys2, sys3,
            sys4 }
      )

      Application Samba_winbind (
            StartProgram = "/opt/VRTSvcs/bin/ApplicationNone/
```

```
        winbindmonitor.sh start"
        StopProgram = "/opt/VRTSvcs/bin/ApplicationNone/
        winbindmonitor.sh stop"
        PidFiles = { "/var/run/winbindmonitor.pid" }
        )

ApplicationNone app (
        )

CFSMount cfsmount3 (
        Critical = 0
        MountPoint = "/mnt2"
        BlockDevice = "/dev/vx/dsk/fsp701-704-v03/vol2"
        NodeList = { sys1, sys2, sys3,
              sys4 }
        )

CFSMount cfsnfs_locks (
        Critical = 0
        MountPoint = "/lock"
        BlockDevice = "/dev/vx/dsk/fsp701-704-v03/lock"
        NodeList = { sys1, sys2, sys3,
              sys4 }
        )

CVMVolDg cvmvoldg3 (
        Critical = 0
        CVMDiskGroup = fsp701-704-v03
        CVMVolume = { lock, vol2 }
        CVMActivation @sys1 = sw
        CVMActivation @sys2 = sw
        CVMActivation @sys3 = sw
        CVMActivation @sys4 = sw
        )

NFS nfs (
        )

NetBios Samba_netbios (
        SambaServerRes = SambaServerResource
        NetBiosName = cfs782
        )
```

```
SambaServer SambaServerResource (
      ConfFile = "/opt/pware/lib/smb.conf"
      SambaTopDir = "/opt/pware"
      LockDir = "/opt/pware/var/locks"
      )

SambaShare sambashare1 (
      Critical = 0
      SambaServerRes = SambaServerResource
      ShareName = cifs1
      ShareOptions = "path=/mnt2;msdfs root=yes;msdfs
      proxy=¥¥10.209.116.87¥¥cifs1_dfs¥¥"
      )

requires group cvm online local firm
Samba_winbind requires Samba_netbios
cfsmount3 requires cfsnfs_locks
cfsmount3 requires cvmvoldg3
cfsnfs_locks requires cvmvoldg3
sambashare1 requires SambaServerResource
sambashare1 requires cfsmount3


// resource dependency tree
//
//    group cfsnfssg
//    {
//    Application Samba_winbind
//        {
//        NetBios Samba_netbios
//        }
//    ApplicationNone app
//    NFS nfs
//    SambaShare sambashare1
//        {
//        SambaServer SambaServerResource
//        CFSMount cfsmount3
//            {
//            CFSMount cfsnfs_locks
//                {
//                CVMVolDg cvmvoldg3
//                }
//            CVMVolDg cvmvoldg3
```

```
        //             }
        //         }
        //     }


group cfsnfssg_dummy (
        SystemList = { sys1 = 0, sys2 = 1, sys3 = 2,
              sys4 = 3 }
        AutoFailOver = 0
        Parallel = 1
        AutoStartList = { sys1, sys2, sys3,
              sys4 }
        )

        requires group cvm online local firm


        // resource dependency tree
        //
        //     group cfsnfssg_dummy
        //     {
        //     }


group cvm (
        SystemList = { sys1 = 0, sys2 = 1, sys3 = 2,
              sys4 = 3 }
        AutoFailOver = 0
        Parallel = 1
        AutoStartList = { sys1, sys2, sys3,
              sys4 }
        )

        CFSfsckd vxfsckd (
              ActivationMode @sys1 = { fsp701-704-v03 = sw }
              ActivationMode @sys2 = { fsp701-704-v03 = sw }
              ActivationMode @sys3 = { fsp701-704-v03 = sw }
              ActivationMode @sys4 = { fsp701-704-v03 = sw }
              )

        CVMCluster cvm_clus (
              CVMClustName = cfs782
              CVMNodeId = { sys1 = 0, sys2 = 1,
```

```
                sys3 = 2,
                sys4 = 3 }
        CVMTransport = gab
        CVMTimeout = 200
        )

    CVMVxconfigd cvm_vxconfigd (
        Critical = 0
        CVMVxconfigdArgs = { syslog }
        )

    cvm_clus requires cvm_vxconfigd
    vxfsckd requires cvm_clus


    // resource dependency tree
    //
    //    group cvm
    //    {
    //    CFSfsckd vxfsckd
    //        {
    //        CVMCluster cvm_clus
    //            {
    //            CVMVxconfigd cvm_vxconfigd
    //            }
    //        }
    //    }


group vip1 (
    SystemList = { sys1 = 0, sys2 = 1, sys3 = 2,
        sys4 = 3 }
    AutoStartList = { sys1, sys2, sys3,
        sys4 }
    TriggerPath = "bin/cavftriggers/vip"
    TriggersEnabled @sys1 = { PREONLINE, POSTONLINE, POSTOFFLINE
}
    TriggersEnabled @sys2 = { PREONLINE, POSTONLINE, POSTOFFLINE
}
    TriggersEnabled @sys3 = { PREONLINE, POSTONLINE, POSTOFFLINE
}
    TriggersEnabled @sys4 = { PREONLINE, POSTONLINE, POSTOFFLINE
}
```

```
PreOnline @sys1 = 1
PreOnline @sys2 = 1
PreOnline @sys3 = 1
PreOnline @sys4 = 1
)

IP vip1 (
      Device = eth0
      Address = "10.209.116.87"
      NetMask = "255.255.252.0"
      )

NIC nic1 (
      Device = eth0
      NetworkHosts = { "10.209.113.1" }
      )

SambaShare sambashare1_dfs (
      Critical = 0
      SambaServerRes = SambaServerResource
      ShareName = cifs1_dfs
      ShareOptions = "path=/mnt2;readonly=no"
      )

requires group cfsnfssg online local firm
sambashare1_dfs requires vip1
vip1 requires nic1


// resource dependency tree
//
//    group vip1
//    {
//    SambaShare sambashare1_dfs
//        {
//        IP vip1
//            {
//            NIC nic1
//            }
//        }
//    }
```

CIFS のデバッグ

cfsshare コマンドはエラーメッセージを /var/VRTSvcs/log ディレクトリ内の VCS エンジンログに記録します。

18

クラスタ化された NFS を使用した Oracle の展開

この章では以下の項目について説明しています。

CNFS を使用して Oracle を展開するタスク

CNFS を使用して Oracle データベースを設定するために SFDB（Storage Foundation Database）ツールを使用する場合は、タスクを次の順序で実行します。

Oracle の CNFS サーバーを設定します。	p.577 の「CNFS を使用した Oracle の配備について」を参照してください。 p.578 の「Oracle の CNFS サーバーの設定」を参照してください。
Direct NFS の Oracle を設定します。	p.582 の「Direct NFS の Oracle の設定」を参照してください。 p.583 の「NFS の推奨されるマウントオプション」を参照してください。 p.584 の「oranfstab について」を参照してください。

Oracle Direct NFS の使用状況を確認します。 p.582 の「Direct NFS の Oracle の設定」を参照してください。

CNFS を使用した Oracle の配備について

CNFS (Clustered Network File System) は、下位クラスタファイルシステムに対して機能するアクティブ/アクティブ NFS を配信するソリューションです。CNFS クラスタの各ノードは、完全な CVM-CFS-VCS (Cluster Volume Manager-Cluster File System-Veritas Cluster Server) スタックを実行し、さらに CNFS サーバーの並列アプリケーションコンポーネントも実行します。CNFS サーバーは、クライアントから POSIX ファイルシステム要求への NFS 要求を変換し、それを下位 CFS インスタンスに発行します。CFS と CVM インスタンスは、すべてのクラスタノードから 1 つ以上のファイルシステムへの同時アクセスを提供するために調整されます。クラスタ全体のボリュームとファイルシステム設定によって、管理の単純化が可能になります。さらに、統合されたクラスタボリュームマネージャは、クラスタ内のすべてのノードに共有デバイス設定の同一の論理ビューを示します。

SFCFSHA (Storage Foundation Cluster File System High Availability) は、アクティブ/アクティブ NFS 機能を、ハイエンド NAS (ネットワーク接続ストレージ) にかかるコストの一部で提供するための効率的なソリューションを提供します。CFS は既存の SAN インフラストラクチャを利用し、クライアント接続層とバックエンドストレージ層の両方で拡張性を向上させます。CFS は、大容量ファイルへのアクセスから、複数の小さいサイズのファイルにアクセスする多くのクライアントまで、複数のタイプに渡る作業負荷を処理するために調整されます。

Oracle データベースの Direct NFS クライアントは、Oracle ソフトウェアに直接 NFS クライアントの機能を統合します。この統合により、Oracle と NFS サーバーの間の I/O パスは最適化され、パフォーマンスを大幅に向上させます。さらに、Oracle Direct NFS クライアントは、データベースの作業負荷に対する NFS クライアント構成のパフォーマンスの最適化を簡略化し、多くの場合、自動化します。

データベースストレージとして CNFS を使うと、Storage Foundation のすべての拡張機能を使用できます。

CNFS 環境の VCS サービスグループ

1 つの仮想 IP を持つ CNFS（Clustered Network File System）が設定されている CFS（Cluster File System）クラスタでは、次の VCS（ Cluster Server）サービスグループがあります。

- cvm: このサービスグループは、CVM（Cluster Volume Manager）と CFS 共有リソースを制御します。このグループは、CFS のインストール時の設定フェーズで自動的に作成されます。このサービスグループでは、CVM と、vxfsckd を通して提供される CFS の基本機能を管理します。

- cfsnfssg: このサービスグループは、NFS 共有のための CFS マウントリソース、およびロック管理に必要な共有 CFS マウントリソースを含んでいます。このサービスグループは、NFS リソースと、CVMVoldg および CFSMount リソースとは別の共有リソースで構成されています。
- vip1: このサービスグループは、NFS クライアントが接続するために必要とする仮想 IP と NIC リソースを含んでいます。仮想 IPサービスグループは、システムフェールオーバー中に 1 つのノードから別のノードにフェールオーバーします。通常は、CNFS クラスタごとに複数の仮想 IP が割り当てられます。

cvm と cfsnfssg は並列サービスグループとして設定され、すべてのノードでオンライン化されています。vip1 サービスグループはフェールオーバーサービスグループとして設定されます。

サービスグループとサービスグループ依存関係について詳しくは、『Veritas Cluster Server 管理者ガイド』を参照してください。

Oracle の CNFS サーバーの設定

次の手順を実行して、Oracle データベースに CNFS（Clustered Network File System）サーバーを設定できます。次の例では、ホスト名が cnfs-1 と cnfs-2 である、2 つのノードを持つ CFS（Cluster File System）クラスタを使用する手順を想定しています。

Oracle データベースに CNFS サーバーを設定するには

1 Oracle に共有ディスクグループを設定します。

```
[cnfs-1]#  vxdg -s init oradg disk1 disk2 disk3 disk4
```

2 データファイル、アーカイブログ、CNFS ロックにボリュームを作成します。

```
[cnfs-1]# vxassist -g oradg make oranfsdata 100g  ¥
layout=stripe ncolumn=4 st_width=1m disk1 disk2 disk3 disk4
[cnfs-1]# vxassist -g oradg make oranfsarch 10g
[cnfs-1]# vxassist -g oradg make cnfs_locks 2g
```

メモ: Oracle データファイルには、ストライプ幅が 1 MB のストライプボリュームをお勧めします。

---

3 CNFS ロック、データファイル、アーカイブログにファイルシステムを作成します。

```
[cnfs-1]# mkfs -t vxfs /dev/vx/dsk/oradg/oranfsarch
[cnfs-1]# mkfs -t vxfs /dev/vx/rdsk/oradg/oranfsarch
[cnfs-1]# mkfs -t vxfs /dev/vx/rdsk/oradg/cnfs_locks
```

メモ: Oracle データファイルには、8 KB のファイルシステムブロックサイズをお勧めします。

メモ: FileSnap 機能を使うためには、ファイルシステムはディスクレイアウトバージョン 8 以降である必要があります。

4 クラスタ化された NFS を設定します。

```
[cnfs-1]# cfsshare config -p nfs oranfsdg /cnfs_locks
```

5 NFS 共有のために /oranfadata と /oranfsarch を設定します。

```
[cnfs-1]# cfsshare add -p nfs -N "rw,no_wdelay,no_root_squash" ¥
oradg oranfsdata /oranfsdata all=
[cnfs-1]# cfsshare add -p nfs -N "rw,no_wdelay,no_root_squash" ¥
oradg oranfsarch /oranfssarch all=
```

6 VIP（仮想 IP）を追加します。

```
[cnfs-1]# cfsshare addvip eth2 virtual_IPsubnet_mask
```

メモ: クラスタ内のすべてのノードの I/O 負荷を分散するために、CNFS クラスタの各ノードに VIP を 1 つずつ追加することを推奨します。

7 CNFS 設定の詳細を表示します。

```
[cnfs-1]# cfsshare display

  SHARE RESOURCE        MOUNTPOINT      SHARE OPTIONS
  share1                /oranfsarch     rw,wdelay,no_root_squash

  share2                /oranfsdata     rw,wdelay,no_root_squash
```

8 VCS リソースの詳細を表示します。

```
[cnfs-1]# hastatus

group           resource             system             message

                                     cnfs-2             RUNNING

                                     cnfs-1             RUNNING
cfsnfssg                              cnfs-1             ONLINE
cfsnfssg                              cnfs-2             ONLINE
cfsnfssg_dummy                       cnfs-1             OFFLINE
cfsnfssg_dummy                       cnfs-2             OFFLINE
cvm                                   cnfs-1             ONLINE
cvm                                   cnfs-2             ONLINE
vip1                                 cnfs-1             OFFLINE
vip1                                  cnfs-2             ONLINE
vip2                                  cnfs-1             ONLINE
vip2                                 cnfs-2             OFFLINE


                app                   cnfs-1             ONLINE

                app                   cnfs-2             ONLINE


                cfsmount1             cnfs-1             ONLINE

                cfsmount1             cnfs-2             ONLINE


                cfsmount2             cnfs-1             ONLINE

                cfsmount2             cnfs-2             ONLINE


                cfsnfs_locks          cnfs-1             ONLINE

                cfsnfs_locks          cnfs-2             ONLINE



                cvmvoldg1             cnfs-1             ONLINE
```

```
cvmvoldg1            cnfs-2              ONLINE

 nfs                 cnfs-2              ONLINE

 nfs                 cnfs-2              ONLINE

share1               cnfs-1              ONLINE

share1               cnfs-2              ONLINE


share2               cnfs-1              ONLINE

share2               cnfs-2              ONLINE


vxfsckd              cnfs-1              ONLINE

vxfsckd              cnfs-2              ONLINE


cvm_clus             cnfs-1              ONLINE

cvm_clus             cnfs-2              ONLINE


cvm_vxconfigd        cnfs-1              ONLINE

cvm_vxconfigd        cnfs-2              ONLINE


vip1                 cnfs-1              OFFLINE

vip1                 cnfs-2              ONLINE


nic1                 cnfs-1              ONLINE

nic1                 cnfs-2              ONLINE


vip2                 cnfs-1              ONLINE
```

```
vip2                    cnfs-2              OFFLINE


nic2                    cnfs-1              ONLINE

nic2                    cnfs-2              ONLINE
```

9 すべてのノードで NFS サービスが設定および実行されていることを確認します。

RHEL 7、SLES 12、およびサポート対象の RHEL 互換配布の場合:

```
# systemctl list-unit-files --type=service nfs
[cnfs-1]# systemctl status nfs
```

以前のバージョンの RHEL、SLES、およびサポート対象の RHEL 互換配布の場合:

```
[cnfs-1]# chkconfig --list  nfs
nfs    0:off   1:off  2:on  3:on  4:on  5:on  6:off
[cnfs-1]# service nfs  status
rpc.mountd (pid 4530) is running...
nfsd (pid 4527 4526 4525 4524 .........) is running...
rpc.rquotad (pid 4474) is running...
```

Direct NFS の Oracle の設定

次の例で示す手順を実行して、Oracle を Direct NFS (Network File System) クライアントとして設定できます。このサンプル手順では、Oracle データベースの (シングルインスタンス) が Linux ボックスにインストールされています。

Direct NFS に Oracle を設定するには

1 NFS ファイルシステムをマウントします。

p.583 の「NFS の推奨されるマウントオプション」を参照してください。

```
[orahost1]# mount -t nfs -o  ¥
rw,bg,hard,nointr,rsize=1048576,wsize=1048576,noac,forcedirectio,¥
vers=3,suid virtual_IP:/oranfsdata  /oranfsdata
```

メモ: `mount` コマンドの VIP (仮想 IP) を使用します。

2 NFS の Direct NFS Client Oracle Disk Manager 制御を有効にします。

ディレクトリを `$ORACLE _HOME/rdbms/lib` に変更します。

次のコマンドを実行します。

```
[orahost1]# make -f ins_rdbms.mk dnfs_on
```

3 `oranfstab` を設定します。

p.584 の「oranfstab について」を参照してください。

NFS の推奨されるマウントオプション

「表 18-1」に、Solaris、HP-UX、AIX、Linux の各オペレーティングシステムの NFS（ネットワークファイルシステム）のマウントオプションを一覧表示します。お勧めするオプションは、Oracle データファイルのものです。

表 18-1 NFS のマウントオプション

オペレーティングシステム	Oracle データファイルのマウントオプション
Solaris	`rw,bg,hard,nointr,rsize=1048576 wsize=1048576,proto=tcp,noac, forcedirectio, vers=3,suid`
AIX（5L）	`cio,rw,bg,hard,nointr,rsize=1048576, wsize=1048576,proto=tcp,noac, vers=3,timeo=600`
HP-UX 11i v3	`rw,bg,vers=3,proto=tcp,noac, forcedirectio,hard,nointr,timeo=600, rsize=1048576,wsize=1048576,suid`

オペレーティングシステム	Oracle データファイルのマウントオプション
Linux x86	rw,bg,hard,nointr,rsize=1048576, wsize=1048576,tcp,actimeo=0, vers=3,timeo=600
Linux x86-64	rw,bg,hard,nointr,rsize=1048576, wsize=1048576,tcp,actimeo=0, vers=3,timeo=600

oranfstab について

デフォルトでは、Direct NFS は /etc/fstab にあるマウントエントリとして機能するために使用されます。Direct NFS に追加の Oracle 固有のオプションを指定するには、oranfstab を使用します。たとえば、oranfstab を使用して、マウントポイントへの追加のパスを指定できます。さらに、新しい Oracle 固有のファイル oranfstab を /etc または $ORACLE_HOME/dbs のいずれかに追加できます。oranfstab が $ORACLE_HOME/dbs に置かれると、エントリは 1 つのデータベース固有になります。ただし、oranfstab が /etc に配置されると、すべての Oracle データベースでグローバルに使用されるようになり、それによってすべての Oracle データベースのマウントポイントが含まれるようになります。Direct NFS は、/etc/mtab の設定に基づいて、NFS ストレージデバイスへのマウントポイント設定を決定します。

Direct NFS は、次の順序でマウントポイントエントリを探します。

- $ORACLE_HOME/dbs/oranfstab
- /etc/oranfstab
- /etc/mtab

Direct NFS は、最初に一致したエントリをマウントポイントとして使用します。いずれの場合でも Oracle では、マウントポイントが Direct NFS を介して機能する場合でも、マウントポイントはカーネル NFS システムによってマウントされる必要があります。Oracle は、オペレーティングシステムの NFS マウントポイントがある oranfstab のエントリをクロスチェックすることで、カーネル NFS マウントを検証します。不一致がある場合、Direct NFS は情報メッセージをログに記録し、NFS サーバーとして機能しません。

oranfstab の例を次に示します。

```
[orahost1]# cat $ORACLE_HOME/dbs/oranfstab
server: cnfs-1.engba.symantec.com
path: virtual_IP
export:/oranfsdata  mount:/oranfsdata
export:/oranfsarch  mount:/oranfsarch
```

メモ: パスで使用される IP アドレスは、VIP（仮想 IP）アドレスです。

Oracle Direct NFS の使用状況の確認

Oracle では、Direct NFS の使用状況が `alert.log` に、また内部カタログ `v$dnfs` テーブルにも記録されます。次の表は Oracle 側からの DIRECT NFS の状態と正常性の検出に利用できる `v$tables` の一覧です。

表 18-2 Direct NFS 情報の v$tables

テーブル名	説明
v$dnfs_servers	Direct NFS を使用してアクセスするサーバーのテーブルを表示します。
v$dnfs_channels	Direct NFS からのファイル提供対象のサーバーに対して開いているネットワークパス (またはチャネル) のテーブルを示します。
v$dnfs_files	現在 Direct NFS で開かれているファイルのテーブルを示します。
v$dnfs_stats	Direct NFS のパフォーマンス統計情報のテーブルを示します。

Oracle Direct NFS の使用状況を確認するには

1 DNFS メッセージの `alert.log` を確認します。

以下は、ODM で実行されている Oracle データベースインスタンスの `alert.log` 行の例です。

```
Oracle Direct NFS ODM Library Version 6.0

ALTER DATABASE  MOUNT
Direct NFS: channel id [0] path [10.182.110.126] to
filer [cnfs-1.engba.symantec.com] via local [] is UP
Direct NFS: channel id [1] path [10.182.110.126] to
filer [cnfs-1.engba.symantec.com] via local [] is UP
```

2 `v$dnfs_servers` から DNFS サーバー情報を確認します。

```
SQL> select * from v$dnfs_servers;

ID SVRNAME                    DIRNAME      MNTPORT NFSPORT WTMAX    RTMAX
-- ---------------------------------------------------------------------------
1  cnfs-1.engba.symantec.com  /oranfsdata1 33553   2049    1048576  1048576
2  cnfs-1.engba.symantec.com  /oranfsdata1 33553   2049    1048576  1048576
```

3 `v$dnfs_channels` から DNFS チャネル情報を確認します。

```
SQL> select CH_ID, SVR_ID, SENDS, RECVS, PINGS from v$dnfs_channels;

CH_ID        SVR_ID     SENDS      RECVS      PINGS
------       ------ ---------------------------------
0              1        0          0          0
0              1        65         130        0
1              1        44         88         0
1              1        47         94         0
```

4 v$dnfs_files から DNFS ファイル情報を確認します。

```
SQL> select * from v$dnfs_files;
FILENAME                                 FILESIZE       PNUM
SVR_ID
---------------------------------------- ---------- ----------
---------
/oranfsdata1/rw_clone/control01.ctl  16072704       15
1
/oranfsdata1/rw_clone/control02.ctl  16072704       15
1
/oranfsdata1/rw_clone/control03.ctl  16072704       15
1
/oranfsdata1/rw_clone/bench.dbf      838877184      10
1
/oranfsdata1/rw_clone/sysaux.dbf     838877184      10
1
/oranfsdata1/rw_clone/undo1.dbf      838877184      10
1
/oranfsdata1/rw_clone/item_1000      1996505088     10
1
```

5 v$dnfs_stats から DNFS 統計情報を確認します。

```
SQL> select PNUM, NFS_READ, NFS_WRITE, NFS_COMMIT, NFS_MOUNT from v$dnfs_stats;

 PNUM   NFS_READ  NFS_WRITE   NFS_COMMIT    NFS_MOUNT
 -----  --------  ----------------------------------
 10         135      201        0             0
 11           0      201        0             0
 12           0      191        0             0
 13           0      198        0             0
 14          86      813        0             0
 15         426      1293       1             1
```

19

サイトとリモートミラーの管理

この章では以下の項目について説明しています。

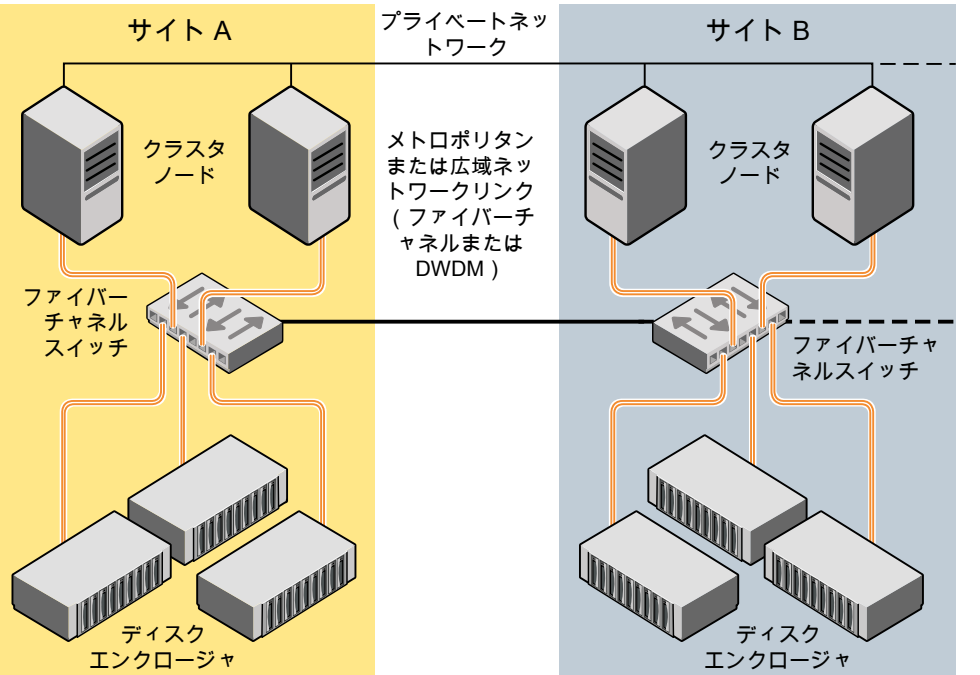
サイトとリモートミラーについて

リモートミラー設定（キャンパスクラスタまたはストレッチクラスタとも呼ぶ）では、通常 1 カ所に配置されるホストとストレージが 2 つ以上のサイトに分割されます。

通常、これらのサイトには、ストレージへのアクセスとクラスタノード間でのプライベートリンク通信を提供する、冗長高キャパシティネットワークを介して接続します。

図 19-1 に、標準的な 2 サイトリモートミラー設定を示します。

図 19-1 2 サイトリモートミラー設定の例



ディスクグループをサイトにあるストレージにわたって設定している場合にサイト間通信が中断されると、それぞれのサイトがローカルディスクグループの設定コピーを更新し続けた結果、シリアルスプリットブレイン条件が発生する可能性があります。

p.1008 の「競合する設定コピーの扱い方」を参照してください。

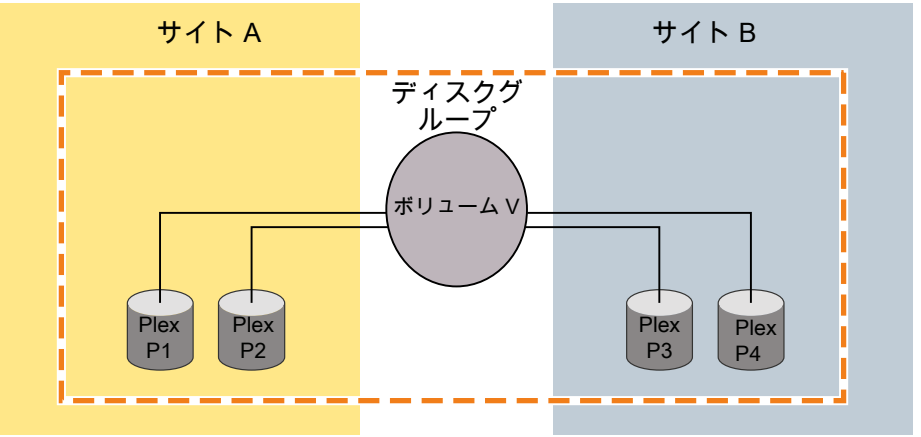
VxVM は、シリアルスプリットブレイン条件の処理、リモートミラーの健全性の監視、各種の障害に対するクラスタの堅ろう性のテスト（ファイアドリルとも呼ぶ）のためのしくみを提供します。

他のサイトにアクセスできなくなった場合にサイトでアプリケーションとサービスを正しく機能させるには、各サイトで各ボリュームの完全なプレックスを少なくとも 1 つ設定し（サイトベースの割り当て）、各サイトのプレックス内にあるデータの一貫性を保証する（サイトの一貫性）必要があります。

ディスクにサイト名のタグを設定することで、ボリュームの作成、サイズ変更、再配置時やボリュームのレイアウト変更時にストレージを正しい場所から割り当てられます。

図 19-2 に 2 つのサイトそれぞれに 2 つのプレックスを設定した、サイトの一貫性があるボリュームの例を示します。

図 19-2 2 つのサイトそれぞれに 2 つのプレックスを設定した、サイトの一貫性があるボリューム



プレックス P1 と P2 の割り当てられたストレージは、サイト A にタグが設定され、プレックス P3 と P4 の割り当てられたストレージは、サイト B にタグが設定されます。

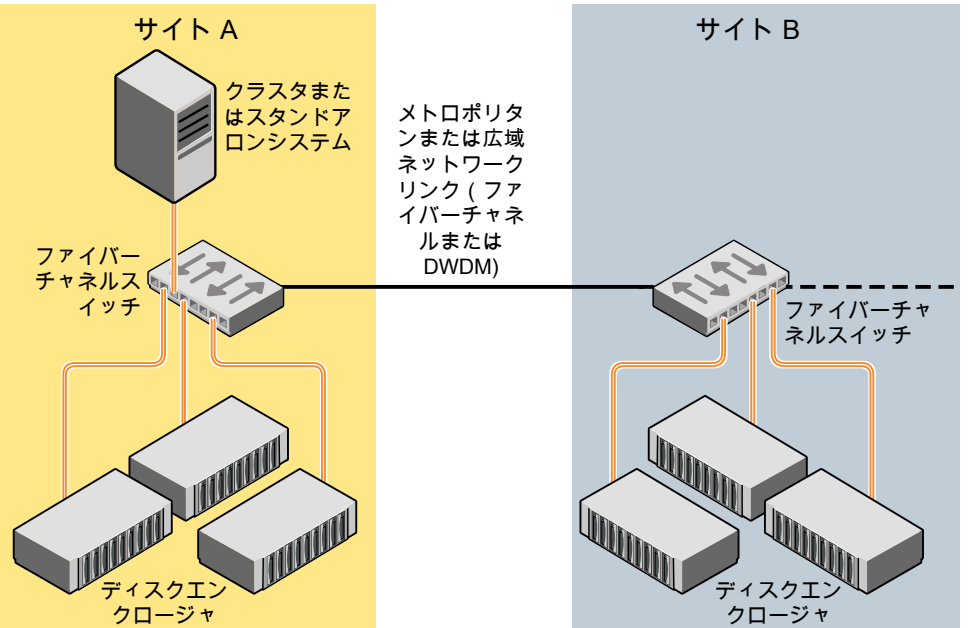
この図には表示されていませんが、DCO ログボリュームもこのサイトでミラー化され、ディスクグループ設定のコピーがサイトで配布されます。

サイトの一貫性とは、ボリュームのプレックス内のデータは各サイトで一貫性がなければならないことを意味します。ボリュームのサイト一貫性は、サイトで最後の完全なプレックスに障害が発生したときにそのサイトを切断することによって保証されます。サイトに障害が発生すると、そのサイトのすべてのプレックスが切断され、サイトは切断されたと見なされます。サイトの一貫性が有効になっていない場合、障害が発生したプレックスのみが切断されます。そのサイトの残りのボリュームとそのボリュームのプレックスは切断されません。

`siteread` 読み取りポリシーをボリュームに設定している場合は読み込みのパフォーマンスを向上するために、VxVM はアプリケーションを実行しているローカルサイトのプレックスから読み込みを行います。書き込みは、すべてのサイトのプレックスに書き込まれます。

図 19-3 に、(同様にサポート対象の)リモートストレージのみを使用した設定を示します。

図 19-3 リモートストレージのみの 2 サイト設定の例



サイトベースの割り当てについて

サイトベースの割り当てポリシーは、サイトで設定されているディスクグループにはデフォルトで強制的に適用されます。サイトベースの割り当てでは、各ボリュームはディスクグループで設定されている各サイトで 1 つ以上のプレックスを保有している必要があります。サイトで設定されているディスクグループ内に新しいボリュームが作成されると、`allsites` 属性はデフォルトで `on` に設定されます。`allsites` 属性は、ボリュームはディスクグループで設定されている各サイトで 1 つ以上のプレックスを保有する必要があることを示します。新しいボリュームでは、読み取りポリシーはデフォルトで `siteread` に設定されます。

サイトにわたるミラーが必要ない場合、または不可能な場合(RAID 5 ボリュームの場合など)、`allsites=off` 属性を `vxassist` コマンドに指定します。サイトがディスクグループで設定されている場合、プレックスは常にサイトに限定され、複数のサイトにわたることはありません。この設定は無効にできません。

ディスクグループに新しいサイトを追加する前に、次の必要条件が満たされていることを確認します。

- 追加しようとしているサイト(タグが付けられたサイト)にディスクが存在するか、ディスクグループにディスクが追加される必要があります。

- allsites が設定されているディスクグループ内の既存の各ボリュームは、追加されるサイトで 1 つ以上のプレックスを保有している必要があります。この条件が満たされない場合、ディスクグループにサイトを追加するコマンドは失敗します。-f オプションを指定すると、コマンドはエラーになりませんが、ボリュームの allsites 属性が off に設定されます。

サイトの一貫性について

サイトの一貫性とは、特定のボリュームセットについての各サイトのデータが、いつの時点でもアプリケーションとの間で一貫性を持つことを意味します。サイトの一貫性が設定されたボリュームでは、ディスクグループ内で設定されている各サイトに 1 つ以上のプレックス(ミラー)が存在する必要があります。サイト一貫性は、サイトの一貫性が設定されたサイトのボリュームが最後の完全なプレックスを失ったときにそのサイトを切断することによって保証されます。サイトの切断により、そのサイトのすべてのプレックスが切断され、同時にそれ以降、そのサイトの設定コピーに対する設定の更新が許可されなくなります。目的のボリュームで siteconsistent 属性を on に設定して、この動作を有効にします。

siteconsistent 属性を off に設定すると、失敗したプレックスのみを切断します。そのサイトにあるその他のボリュームのプレックスは切断されません。

siteconsistent 属性はディスクグループレベルにも存在し、ディスクグループの境界でサイト一貫性機能を有効または無効にするために使用できます。また、ディスクグループの siteconsistent 属性を有効にすると、そのディスクグループ内に作成される新しいボリュームはそれぞれ、ディスクグループの siteconsistent 属性をデフォルトで継承します。ディスクグループに siteconsistent 属性を設定しても、既存のボリュームの siteconsistent 属性に影響はありません。サイトの一貫性は、個々のボリュームで制御することもできます。

デフォルトでは、ボリュームは所属するディスクグループに設定されている値を継承します。

デフォルトでは、サイトの一貫性が設定されたボリュームを作成すると、関連付けされたバージョン 20 の DCO ボリュームも作成され、ボリュームの永続 FastResync が有効になります。これにより、サイトの再接続時にボリュームのリカバリがより高速になります。

p.602 の「ボリュームに対するサイトの一貫性の設定」を参照してください。

ディスクグループに対してサイトの一貫性を設定する前に、次の必要条件が満たされていることを確認します。

- Site Awareness 機能を有効にするライセンスがリモートミラー設定内のすべてのホストでインストールされている必要があります。
- サイトの一貫性を有効にする前に、ディスクグループ内に 2 つ以上のサイトが設定されている必要があります。
  p.596 の「既存のディスクグループに対するサイトの一貫性の設定」を参照してください。

- ディスクグループに siteconsistent 属性を設定する前に、ディスクグループ内のすべてのディスクをいずれかのサイトに登録する必要があります。

サイトタグについて

リモートミラー設定では、ディスクグループの各ストレージデバイスにはサイト情報のタグを付ける必要があります。サイトタグは、デバイスが関連付けられるサイトを示します。VxVM は VxVM で初期化されたディスクに、任意の名前と値の組み合わせを使ってタグを付ける機能を備えています。タグ名 site は VxVM によって予約されており、タグが付けられたディスクのサイト情報の識別に使われます。コマンド vxdisk settag を使うと、1 つのエンクロージャの複数またはすべてのディスクにタグを付けるか、複数のエンクロージャのディスクにタグを付けることができます。コマンドでディスクグループを指定すると、タグを付ける範囲をディスクグループ内のディスクに限定できます。

自動サイトタグ付けを使い、ディスクグループにディスクを追加するときにサイトタグをディスクに割り当てることができます。自動サイトタグ付けが有効な場合、新しく追加したディスクや LUN はディスクグループに保存されているサイトとエンクロージャ間のマッピングのサイトタグを継承します。自動サイトタグ付けを使うには、ディスクグループの自動サイトタグ付けを有効にしてから、ディスクグループのエンクロージャにサイト名を割り当てます。そのディスクグループ内のディスクや LUN は、それらが属するエンクロージャからタグを継承します。

サイトの読み取りポリシーについて

siteread 読み取りポリシーをボリュームに設定している場合は読み取りのパフォーマンスを向上するために、VxVM はアプリケーションを実行しているローカルサイトのプレックスから読み込みを行います。siteread は設定済みのサイトがあるボリュームのデフォルトの読み取りポリシーです。書き込みは、すべてのサイトのプレックスに書き込まれます。サイト情報を使ってホストにタグ付けすることで、VxVM はホストが属するサイトを識別します。あるサイトからホストが読み込みを開始すると、同じサイトのタグが付いているディスクによってその要求が満たされます。ホストとディスクに正しいサイト情報でタグを付けると、siteread 読み取りポリシーの使用時に読み取りのパフォーマンスが最大になります。

Site Awareness 機能を有効にするライセンスをリモートミラー設定のすべてのホストにインストールし、複数のサイトでサイトの一貫性が有効になるようにディスクグループを設定し、allsites=on 属性をボリュームに指定している場合、デフォルトの読み取りポリシーは siteread です。

siteread ポリシーを設定していない場合は、次のコマンドを使ってボリュームの読み取りポリシーを siteread ポリシーに設定します。

```
# vxvol [-g diskgroup] rdpol siteread volume
```

ホストにサイト名を設定していない場合、このコマンドは無効です。

p.256 の「ミラーボリュームの読み取りポリシーの変更」を参照してください。

キャンパスクラスタのディスク切断ポリシーについて

キャンパスクラスタでは、切断ポリシーは他の CVM クラスタの場合と同様にローカルまたはグローバルになります。これらのポリシーの動作は、少数の例外を除き、このセクションで説明されているように他の CVM クラスタと同じです。

ローカルストレージの接続エラーの場合、ローカルの切断ポリシーはボリュームに対して I/O をローカルに失敗します。プレックスは切断されません。I/O 転送を有効にすると、CVM は I/O を別のノードにリダイレクトします。キャンパスクラスタでは、I/O 転送は同じサイトのノードが優先されます。

ローカルストレージの接続エラーの場合、グローバルの切断ポリシーがプレックスを切断します。クラスタの 1 つ以上のノードでサイトのすべてのプレックスが破損し、I/O 転送が有効でない場合、CVM はサイトを切断します。サイトの切断により、そのサイトのすべてのプレックスが切断され、同時にそれ以降、そのサイトの設定コピーの設定の更新が許可されなくなります。I/O 転送を有効にすると、CVM は I/O を別のノードにリダイレクトします。I/O 転送は同じサイトのノードが優先されます。CVM は、各サイトで少なくとも 1 つのプレックスを利用可能に保とうとします。したがって、グローバルの切断ポリシーにより、そのサイトの他のすべてのプレックスが切断された後、サイトの最新のプレックスの I/O エラーが I/O 転送をトリガします。

1 つ以上のノードのボリュームのすべてのプレックスに影響するストレージ接続エラーでは、I/O はボリュームで失敗します。I/O 転送を有効にすると、I/O は別のノードにリダイレクトされます。すべてのサイトのすべてのノードが影響されると、すべてのサイトは最新のサイトを除いて切断され、I/O は失敗します。

表 19-1 に、キャンパスクラスタの切断ポリシーの概略を示します。

表 19-1　キャンパスクラスタのポリシーの切断

影響されるサイト	影響されるプレックス	ローカルの切断ポリシーと I/O 転送 =off	ローカルの切断ポリシーと I/O 転送 =on	グローバルの切断ポリシーと I/O 転送 =off	グローバルの切断ポリシーと I/O 転送 =on
1 つのサイト。	サイトの 1 つ以上のノードで、サイトの一部のプレックスに I/O エラーがあります。 たとえば、サイトにある 3 つのプレックスのうち 2 つにエラーが発生しました。	ボリュームでローカルに I/O エラーが発生します。	別のノードに接続して I/O を転送します。できれば同じサイト内で転送します。	プレックスを切断します。	プレックスを切断します。

影響されるサイト	影響されるプレックス	ローカルの切断ポリシーと I/O 転送 =off	ローカルの切断ポリシーと I/O 転送 =on	グローバルの切断ポリシーと I/O 転送 =off	グローバルの切断ポリシーと I/O 転送 =on
	サイトの 1 つ以上のノードで、サイトにあるすべてのプレックスに I/O エラーがあります たとえば、サイトにある 3 つのプレックスのすべてにエラーが発生しました。	ボリュームでローカルに I/O エラーが発生します。	別のノードに接続して I/O を転送します。できれば同じサイト内で転送します。	プレックスを切断します。	別のノードに接続して I/O を転送します。できれば同じサイト内で転送します。
	サイトの 1 つ以上のノードのボリュームのすべてのプレックス。	ボリュームでローカルに I/O エラーが発生します。	別のノードに接続して I/O を転送します。できれば同じサイト内で転送します。	ボリュームで I/O エラーが発生します。	別のノードに接続して I/O を転送します。できれば同じサイト内で転送します。
すべてのサイト	クラスタのすべてのノードで、プレックスに I/O エラーがあります	プレックスを切断します。	プレックスを切断します。	プレックスを切断します。	プレックスを切断します。
	すべてのサイトのすべてのノードのボリュームのすべてのプレックス。	1 つのサイトを除いてすべて切断します。残りのサイトで I/O エラーが発生します。	1 つのサイトを除いてすべて切断します。残りのサイトで I/O エラーが発生します。	1 つのサイトを除いてすべて切断します。残りのサイトで I/O エラーが発生します。	1 つのサイトを除いてすべて切断します。残りのサイトで I/O エラーが発生します。
すべてのサイト - ストレージの分割	すべてのサイトが他のサイトのストレージに対して接続を失います。	非優先のサイトを切断します。	I/O を転送します。	非優先のサイトを切断します。	I/O を転送します。

p.184 の「ディスク切断ポリシーについて」を参照してください。

既存のディスクグループに対するサイトの一貫性の設定

サイト一貫性機能を使うには、設定に関連するすべてのホストとサイトに Site Awareness 機能を有効にするライセンスがインストールされている必要があります。

既存のディスクグループにサイトの一貫性を設定するには、次の手順を実行します。

1. ディスクグループに対して vxdg upgrade コマンドを実行して、ディスクグループが少なくともバージョン 140 に更新されていることを確認します。

   ```
   # vxdg upgrade diskgroup
   ```

2. ディスクグループにアクセスできる各ホストでサイト名を定義します。

   ```
   # vxdctl set site=sitename
   ```

3. ディスクグループ内のすべてのディスクに該当するサイト名のタグを設定します。

   ```
   # vxdisk [-g diskgroup] settag site=sitename disk1 disk2
   ```

   また、指定したエンクロージャ内のすべてのディスクにタグを設定するには、次のコマンドを使います。

   ```
   # vxdisk [-g diskgroup] settag site=sitename encl:encl_name
   ```

4. サポートされていない RAID-5 ボリュームを別のディスクグループへ移動するには、vxdg move コマンドを使います。または、vxassist convert コマンドを使って、ボリュームを mirror、mirror-stripe などのサポートされているレイアウトに変換します。site や mirror=site ストレージ割り当て属性を使って、正しいストレージにプレックスが作成されるようにできます。

5. 各サイトでボリュームのプレックス数が少なくとも 1 つになるようにするには、vxevac コマンドを使います。site や mirror=site ストレージ割り当て属性を使って、正しいストレージにプレックスが作成されるようにできます。

6. 各サイトのサイトレコードをディスクグループに登録します。

   ```
   # vxdg -g diskgroup addsite sitename
   ```

7. ディスクグループに対するサイトの一貫性をオンにします。

   ```
   # vxdg -g diskgroup set siteconsistent=on
   ```

8 各サイトへのデータレプリケーションが必要なボリュームに対して、allsites フラグを有効にします。

```
# vxvol [-g diskgroup] set allsites=on volume
```

9 ディスクグループ内の、サイトの一貫性が必要な既存の各ボリュームに対して、サイトの一貫性を有効にします。DCOv20 をまだ接続していなければ、これを接続する必要もあります。DCOv20 はサイトの切断と再接続が瞬時に行われるようにするために必要です。

```
# vxvol [-g diskgroup] set siteconsistent=on volume ...
```

リモートミラー設定としての新しいディスクグループの設定

メモ: リモートミラー機能を使うには、設定に関連するすべてのホストとサイトに Site Awareness 機能を有効にするライセンスがインストールされている必要があります。

この項では新しいディスクグループの設定について説明します。リモートミラーの設定として既存のディスクグループを設定するには、追加手順が必要になることがあります。

p.596 の「既存のディスクグループに対するサイトの一貫性の設定」を参照してください。

リモートミラー設定用の新しいディスクグループの設定

1 ディスクグループにアクセスできる各ホストのサイト名を定義します。

```
# vxdctl set site=sitename
```

ホストに割り当てられたサイト名を確認するには、次のコマンドを使います。

```
# vxdctl list
```

2 各サイトのストレージを含むディスクグループを作成します。

3 サイトごとに、サイトレコードをディスクグループに登録します。

```
# vxdg -g diskgroup [-f] addsite sitename
```

4 次のいずれかを実行します。

- ディスクグループに関係なくすべてのディスクにタグを付ける場合は、次の手順を実行します。

ディスクかエンクロージャにサイト名を割り当てます。サイトタグはディスクレベルまたはエンクロージャレベルで設定できます。1 つ以上のエンクロージャを指定する場合、サイトタグはディスクディスクグループ内のそのエンクロージャ内のディスクに適用されます。次のコマンドを入力します。

```
# vxdisk [-g diskgroup] settag site=sitename ¥
  disk disk1... |encl:encl_name encl:encl_name1...
```

ここで、ディスクはディスクアクセス名またはディスクメディア名で指定できます。

- ディスクグループに追加される新しいディスクに、それらが属するエンクロージャに基づいてタグを自動的に付ける場合は、次の手順をこの順番で実行します。これらの手順を使えるのは単一のグループのディスクに限られます。
  - 必要に応じて、ディスクグループに対する autotagging ポリシーを on に設定します。自動タグ付けはデフォルト設定です。したがってこの手順が必要なのは、自動タグ付けポリシーが以前に無効にされた場合のみです。自動タグ付けをオンにするには、次のコマンドを入力します。

    ```
    # vxdg [-g diskgroup] set autotagging=on
    ```

  - サイトとエンクロージャの組み合わせごとに、サイトとエンクロージャ間のマッピング情報をディスクグループに追加します。次のコマンドを入力します。

    ```
    # vxdg [-g diskgroup] settag encl:encl_name1 site=sitename1
    ```

    このコマンドの結果として、指定したディスクグループ内のエンクロージャ encl_name1 のすべてのディスクに、サイト情報を使ってタグが付けられます。

5 ディスクグループに対するサイトの一貫性の必要条件を有効にします。

```
# vxdg -g diskgroup set siteconsistent=on
```

ファイアドリル - 設定のテスト

警告: サービスまたはデータの潜在的な損失を回避するには、次の手順を稼働中のシステムで実行しないことをお勧めします。

サイトのボリュームとディスクグループの一貫性を有効にしてから、考えられる各種の障害が発生した場合に実行する手順の有効性を確認してください。ファイアドリルを使うと、サイト障害などの災害シナリオからのリカバリ中にサイトを正常に起動できることをテストできます。

サイト障害のシミュレート

サイトの障害をシミュレートするには、次のコマンドを使って指定したサイトのすべてのデバイスを切断します。

```
# vxdg -g diskgroup [-f] detachsite sitename
```

サイトのストレージに設定されたプレックスのいずれかが現在オンラインである場合、-f オプションを指定する必要があります。

サイトが切断された後、利用可能なサイトでアプリケーションが正しく動作する必要があります。この手順はプライマリサイトが正常であることを確認します。セカンダリサイトを確認することで、ファイアドリルを続行します。

セカンダリサイトの確認

プライマリサイトからサイトを切断したら、アプリケーションがセカンダリサイトで正しく起動するかどうかを確認します。ファイアドリルは、プライマリサイトで災害が発生したときにアプリケーションをセカンダリで確実に動作できるようにします。この手順は、ファイアドリル操作が開始される前にアプリケーションが正しく動作していることを前提とします。

セカンダリサイトを確認するには、次のコマンドを使って、切断されたサイトを異なるホストにインポートします。

```
# vxdg -o site=sitename import dgname
```

次に、アプリケーションを開始します。アプリケーションがセカンダリサイトで正しく動作すれば、セカンダリのサイトの整合性が確認されます。

シミュレート用のサイト障害からのリカバリ

シミュレート用のサイト障害についてセカンダリのデータを検証したら、セタンダリサイトからディスクグループをデポートします。次に、サイトをプライマリホストに再接続します。

次のコマンドを使って、サイトを再接続し、ディスクグループをリカバリします。

```
# vxdg -g diskgroup [-o overridessb] reattachsite sitename
# vxrecover -g diskgroup
```

シリアルスプリットブレイン条件を指示した場合、-o overridessb オプションの指定が必要なことがあります。

サイト名の変更

リモートミラー設定内の各サイトの識別に使われるサイト名（タグ）は変更できます。サイト名を変更すると、ディスクグループ内のサイトレコードが変更されます。また、既存のサイト名でタグ付されているすべてのディスクとエンクロージャのサイト名も変更されます。

サイト名を変更した後、そのサイトに属する各ホストのサイト名を明示的に変更する必要があります。

p.600 の「ホストのサイト名のリセット」を参照してください。

サイト名を変更するには

- 次のように新しいサイト名を指定します。

```
# vxdg [-g diskgroup] renamesite old_sitenamenew_sitename
```

ホストのサイト名のリセット

サイトの名前を変更する場合は、新しいサイト名を参照するように各ホストを明示的に設定する必要があります。

ホストのサイト名をリセットするには

1. ホストからサイト名を削除します。

```
# vxdctl [-F] unset site
```

インポートしたディスクグループをサイトに登録する場合は、`-F` オプションが必須です。

2. ホストの新しいサイト名を設定します。

```
# vxdctl set site=sitename
```

サイトに割り当てられている名前は `/etc/vx/volboot` ファイルに格納されます。

リモートミラー設定の管理

リモートミラーサイトを設定したら、設定管理のための追加タスクに関する以降の項を参照してください。

ディスクまたはエンクロージャのサイトタグの設定

リモートミラー設定をセットアップするには、ディスクグループ内の各ストレージデバイスがどのサイトに属するかを指定します。1 つ以上のディスクまたはエンクロージャにサイトタ

グを割り当てます。ディスクまたはエンクロージャがディスクグループに属さなければ、この方法を使ってサイトタグを割り当てる必要があります。

サイト名を使ってディスクまたはエンクロージャにタグを付けるには

- 次のコマンドを使って 1 つ以上のディスクまたはエンクロージャにサイト名を割り当てます。

```
# vxdisk [-g diskgroup] settag site=sitename ¥
disk disk1...|encl:encl_name encl:encl_name1...
```

ここで、ディスクはディスクアクセス名またはディスクメディア名で指定できます。

サイトに登録されたディスクまたはエンクロージャを表示するには

- サイトに登録されているディスクまたはエンクロージャを調べるには、次のコマンドを使います。

```
# vxdisk [-g diskgroup] listtag
```

ディスクまたはエンクロージャからサイトタグを削除するには

- ディスクまたはエンクロージャからサイトタグを削除するには、次のコマンドを使います。

```
# vxdisk rmtag site=sitename ¥
disk disk1...|encl:encl_name encl:encl_name1...
```

ディスクグループに対する自動サイトタグ付けの設定

ディスクまたは LUN のタグをエンクロージャから継承する場合、自動サイトタグ付けを設定します。ディスクグループに対して自動サイトタグ付けを有効にしたら、ディスクグループ内のエンクロージャにサイト名を割り当てます。そのディスクグループに追加されるディスクまたは LUN は、それらが属するエンクロージャからタグを継承します。

ディスクグループに対して自動サイトタグ付けを設定するには

1. ディスクグループに対する自動タグ付けポリシーをオンに設定します。自動タグ付けはデフォルト設定です。したがってこの手順が必要なのは、自動タグ付けポリシーが以前に無効にされた場合のみです。

   自動タグ付けをオンにするには、次のコマンドを使います。

   ```
   # vxdg [-g diskgroup] set autotagging=on
   ```

2. 次のコマンドを使って、ディスクグループ内のエンクロージャにサイト名を割り当てます。

   ```
   # vxdg [-g diskgroup] settag encl:encl_name site=sitename
   ```

ディスクグループのサイトタグを一覧表示するには

- ディスクグループのサイトタグを一覧表示するには、次のコマンドを使います。

```
# vxdg  [-q] [-o tag=name|~name[=value|~value] ¥
listtag [diskgroup ...]
```

エンクロージャまたはディスクグループからサイトタグを削除するには

- ディスクグループからサイトタグを削除するには、次のコマンドを使います。

```
# vxdg [-g diskgroup] rmtag [encl:encl_name] site=sitename
```

ボリュームに対するサイトの一貫性の設定

ボリュームの作成時にサイトの一貫性必要条件を設定するには、次に示すように vxassist make コマンドに siteconsistent 属性を指定します。

```
# vxassist [-g diskgroup] make volume size ¥
nmirror=4 siteconsistent={on|off}
```

デフォルトでは、ボリュームは所属するディスクグループに設定されている値を継承します。

デフォルトでは、サイトの一貫性が設定されたボリュームを作成すると、関連付けされたバージョン 20 の DCO ボリュームも作成され、ボリュームの永続 FastResync が有効になります。これにより、サイトの再接続時にボリュームのリカバリがより高速になります。

既存のボリュームに対してサイトの一貫性必要条件をオンにするには、次の形式の vxvol コマンドを使います。

```
# vxvol [-g diskgroup] set siteconsistent=on volume
```

ボリュームに対してサイトの一貫性必要条件をオフにするには、次のコマンドを使います。

```
# vxvol [-g diskgroup] set siteconsistent=off volume
```

サイトの一貫性が設定されたディスクグループ内の RAID 5 ボリュームでは、siteconsistent 属性と allsites 属性を off に設定する必要があります。

サイトを指定したストレージ割り当ての例

表 19-2 に、vxassist コマンドでサイトを使ってストレージを割り当てる方法の例を示します。この例では、ディスクグループ ccdg はサイトの一貫性が有効にされており、site1 と site2 の 2 つのサイトでディスクが構成されているものとします。また、ccdg01、ccdg02、ccdg03 はサイト site1 によってタグ付けされたディスクのディスクメディア名

で、ccdg09、ccdg10、ccdg11 はサイト site2 でタグ付けされたディスクのディスクメディア名です

表 19-2　　サイトを指定したストレージ割り当ての例

コマンド	説明
`# vxassist -g ccdg make vol 2g ¥ nmirror=2`	各サイトにミラーを 1 つ持つボリュームを作成します。nmirror キーワードはオプションです。nmirror キーワードを指定する場合、その値はサイト数と同じ値にする必要があります。
`# vxassist -g ccdg -o ordered ¥ make vol 2g ¥ layout=mirror-stripe ncol=3 ¥ ccdg01 ccdg02 ccdg03 ccdg09 ¥ ccdg10 ccdg11`	複数のサイトにわたる冗長性を有効にするために、割り当て順を指定してミラー化ストライプボリュームを作成します。名前付きのディスクには該当するサイト名のタグが設定されており、各サイトにはボリュームを作成するための十分なディスクがある必要があります。
`# vxassist -g ccdg make vol 2g ¥ nmirror=2 ccdg01 ccdg09`	名前付きの各ディスクにミラーを 1 つ持つボリュームを作成します。名前付きのディスクには該当するサイト名のタグが設定されており、各サイトにはボリュームを作成するための十分なディスクがある必要があります。
`# vxassist -g ccdg make vol 2g ¥ nmirror=2 siteconsistent=off ¥ allsites=off`	サイトの一貫性が設定されていないミラーボリュームを作成します。どちらのミラーもディスクグループの利用できる任意のストレージから割り当て可能ですが、各ミラーのストレージは単一のサイトに限定されます。
`# vxassist -g ccdg make vol 2g ¥ nmirror=2 site:site2 ¥ siteconsistent=off ¥ allsites=off`	サイトの一貫性が設定されていないミラーボリュームを作成します。両方のミラーは、site2 に所属するタグが設定されたディスクグループ内の利用可能ないずれかのストレージから割り当てられます。

コマンド	説明
`# vxassist -g ccdg make vol 2g ¥ nmirror=2 ¥!site:site1 ¥ siteconsistent=off ¥ allsites=off`	サイトの一貫性が設定されていないミラーボリュームを作成します。両方のミラーは、site1 に所属するタグが設定されたディスクグループ内の利用可能ないずれかのストレージから割り当てられます。 **メモ:** ! 文字は一部のシェルでの特殊文字です。次の bash シェルの例では、この文字をエスケープしています。
`# vxassist -g ccdg mirror vol ¥ site:site1`	指定したサイトにミラーを追加します。このコマンドは、サイトに利用可能なストレージが十分ないとエラーになります。このコマンドはボリュームの allsites や siteconsistent の属性に影響を与えません。
`# vxassist -g ccdg remove ¥ mirror vol site:site1`	指定したサイトのボリュームからミラーを削除します。ボリュームの allsites 属性が on に設定されている場合、サイトの最後のプレックスを削除しようとするとコマンドはエラーになります。
`# vxassist -g ccdg growto vol ¥ 4g`	ボリュームを拡張します。ボリュームの各ミラーは、それが属するサイトと同じストレージを使って拡張されます。各サイトに、ミラーを拡張する十分なストレージがない場合、コマンドは失敗します。

サイト情報の表示

ホストのサイト名を表示するには

- ホストの所属先サイトを確認するには、ホスト上で次のコマンドを使います。

```
# vxdctl list | grep siteid
siteid: building1
```

サイトに登録されたディスクまたはエンクロージャを表示するには

- サイトに登録されているディスクまたはエンクロージャを調べるには、次のコマンドを使います。

```
# vxdisk [-g diskgroup] listtag
```

ディスクグループに対する自動サイトタグ付け設定を表示するには

- ディスクグループで自動サイトタグ付けがオンになっているかどうかを確認するには、次のコマンドを使います。

```
# vxprint -g diskgroup -F"%autotagging" diskgroup
```

ディスクグループに対してサイトの一貫性が有効になっているかどうかを確認するには

- ディスクグループに対してサイトの一貫性が有効になっているかどうかを確認するには、次のコマンドを使います。

```
# vxdg list diskgroup | grep siteconsistent
flags: siteconsistent
```

ボリュームに対してサイトの一貫性が有効になっているかどうかを確認するには

- ボリュームに対してサイトの一貫性が有効になっているかどうかを確認するには、次のコマンドを使います。

```
# vxprint -g diskgroup -F"%siteconsistent" vol
```

プレックスまたはミラーの割り当て元サイトを識別するには

- プレックスまたはミラーの割り当て元サイトを識別するには、次のコマンドを使います。

```
# vxprint -g diskgroup -F"%site" plex
```

ディスクグループのサイトタグを一覧表示するには

- ディスクグループのサイトタグを一覧表示するには、次のコマンドを使います。

```
# vxdg  [-q] [-o tag=name|~name[=value|~value] ¥
listtag [diskgroup ...]
```

障害とリカバリのシナリオ

表 19-3 に、考えられる障害のシナリオと、リモートミラー機能のリカバリ手順を示します。

表 19-3　　障害シナリオとリカバリ手順

障害シナリオ	リカバリ手順
サイト間のネットワークリンクの中断	p.606 の「サイト接続性の損失からのリカバリ」を参照してください。
サイトにあるホストの障害	p.607 の「ホスト障害からのリカバリ」を参照してください。

障害シナリオ	リカバリ手順
サイトにあるストレージの障害	p.607 の「ストレージ障害からのリカバリ」を参照してください。
サイトにあるホストとストレージ両方の障害	p.607 の「サイト障害からのリカバリ」を参照してください。
全サイトのホストからリモートサイトのストレージに対する接続の中断	p.608 の「全サイトのホストからリモート サイトのストレージに対する接続のリカバリ」を参照してください。
あるサイトのホストから全サイトのストレージに対する接続の中断	p.608 の「あるサイトのホストから全サイトのストレージに対する接続のリカバリ」を参照してください。

サイト接続性の損失からのリカバリ

警告: データの潜在的な損失を回避するため、ネットワークのスプリットブレインを処理するように Cluster Server（VCS）を設定することをお勧めします。

サイト間のネットワークリンクが中断された場合、アプリケーション環境が引き続き並行して実行され、そのために各サイトのディスクグループの設定コピー間に不整合が発生することがあります。アプリケーションのパラレルインスタンスがボリュームに対する書き込みを発行すると、回復不能のデータ損失が起きる可能性があり、その場合には手動での介入が必要になります。データ損失を回避するため、ネットワークのスプリットブレインの状況を処理するように VCS フェンシング機構を設定することをお勧めします。

VCS フェンシングを使わないと、シリアルスプリットブレイン状態が発生することがあります。サイト間の接続性が復元されたときには、サイト間でシリアルスプリットブレイン条件が検出されます。データとディスクグループの優先されるバージョンの設定コピーを持つサイトを 1 つ選択する必要があります。選択したサイトのデータが他のサイトに再同期されます。ネットワークのスプリットブレイン発生後にボリュームに対して新しい書き込みが発行された場合、それらは選択したサイトのデータで上書きされます。他のサイトの設定コピーは、選択したサイトにあるコピーから更新されます。

選択したサイトで次のコマンドを使って、サイトを再接続し、ディスクグループをリカバリします。

```
# vxdg -g diskgroup -o overridessb reattachsite sitename
# vxrecover -g diskgroup
```

リモートサイトの唯一のストレージがある単一のサイトでホストシステムを設定した場合は、ストレージがオンラインに復帰したときに VxVM の通常の再同期機構を使ってリモートプレックスを回復します。

p.1008 の「競合する設定コピーの扱い方」を参照してください。

ホスト障害からのリカバリ

サイトの 1 つ以上のクラスタノードで障害が発生し、ストレージは引き続きオンラインのとき、Storage Foundation HA 製品の場合は VCS のフェールオーバーによって、ノードが Storage Foundation Cluster File System ソフトウェアによってサポートされる共有ディスクグループのマスターであった場合はノードのマスターフェールオーバーによって処理されます。

ストレージ障害からのリカバリ

サイトでストレージに障害が発生した場合、サイトの一貫性が設定されたボリュームにそのサイトで利用可能な他のミラーがあるときは、そのストレージに設定されたプレックスはローカルで切断されます。VxVM のホットリロケーション機能は、ディスクグループ内の利用可能な他のストレージに障害の発生したプレックスを再作成しようとします。サイトの一貫性が設定されたボリュームの稼働中プレックスがサイトに残っておらず、ホットリロケーションがそのサイトにプレックスを再作成できない場合、そのサイトは切断されます。サイトの接続性は失われていないため、そのサイトのホストで実行中のアプリケーションは、引き続き他のサイトのデータにアクセスできます。

ストレージがオンラインに復帰すると、vxattachd はサイトを自動的に再接続します。

p.608 の「サイトの自動再接続」を参照してください。

vxattachd が動作しない場合、次のコマンドを使ってサイトを再接続し、ディスクグループをリカバリします。

```
# vxdg -g diskgroup reattachsite sitename
# vxrecover -g diskgroup
```

ディスクグループのリカバリについて詳しくは、『Veritas InfoScale トラブルシューティングガイド』を参照してください。

サイト障害からのリカバリ

サイトのすべてのホストとストレージに障害が発生した場合、サイトがオンラインに復帰してから次のコマンドを使ってサイトを再接続し、ディスクグループをリカバリします。

```
# vxdg -g diskgroup [-o overridessb] reattachsite sitename
# vxrecover -g diskgroup
```

-o overridessb オプションは、シリアルスプリットブレイン状態が示された場合のみ必要です。シリアルスプリットブレイン状態は、プライベートネットワークリンクが動作不能のときにサイトが起動された場合に発生することがあります。このオプションは、再接続されたサイトの設定データベースを他のサイトの一貫性のあるコピーで更新します。

p.1008 の「競合する設定コピーの扱い方」を参照してください。

ディスクグループのリカバリについて詳しくは、『Veritas InfoScale トラブルシューティングガイド』を参照してください。

全サイトのホストからリモート サイトのストレージに対する接続のリカバリ

このシナリオでは、サイトのホストは相互のサイトにあるストレージへの接続を失います。たとえば、サイト A のホストはサイト B のストレージへの接続を失い、サイト B のホストはサイト A のストレージへの接続を失います。このような状況は、ファイバーチャネル（FC）スイッチ間のリンクが停止したために発生することがあります。

この場合、いずれかのサイトが切断されます。 切断されたサイトにあるホストでは、I/O エラーが発生します。切断されたサイトにあるホストのアプリケーションは、他のサイトにあるホストにフェールオーバーする必要があります。

リカバリするには、FC リンクを再接続します。vxattachd デーモンが実行中であると、切断されたサイトは自動的に再接続され、リカバリが開始されます。それ以外の場合は、手動で vxreattach コマンドと vxrecover コマンドを実行し、サイトを ACTIVE 状態に戻します。

p.607 の「ストレージ障害からのリカバリ」を参照してください。

あるサイトのホストから全サイトのストレージに対する接続のリカバリ

このシナリオでは、あるサイトのホストが、全サイトのストレージに対する接続を失います。たとえば、サイト A のホストが、サイト A とサイト B の両方で、ストレージに対する接続を失います。

この場合は、サイトは切断されません。全サイトのストレージに対する接続を失ったホストでの I/O は失敗します。アプリケーションは、他のサイトにあるホストにフェールオーバーする必要があります。

ストレージに対する接続を復元し、アプリケーションをオンラインにします。

サイトの自動再接続

サイトの自動再接続デーモン vxattachd は、サイトの自動再接続を実現します。vxattachd デーモンは、vxnotify 機構を使って、サイトで障害が発生した後にオンラインに復帰するストレージを監視し、サイト間でミラーの冗長性を復元します。

ホットリロケーションデーモン vxrelocd が動作している場合、vxattachd はサイトの再接続を試みて、vxrelocd がディスクグループ内の利用可能なディスクを使って障害の発生したサブディスクを再配置できるようにします。vxrelocd は、障害の発生したサブディスクの再配置に成功すると、サイトでプレックスのリカバリを開始します。すべてのプレックスをリカバリすると、プレックスは ACTIVE 状態になり、サイトの状態は ACTIVE に設定されます。

vxrelocd が動作していない場合、vxattachd は、そのサイトのすべてのディスクがアクセス可能になった場合にのみサイトを再接続します。再接続が成功すると、vxattachd はサイトの状態を ACTIVE に設定し、プレックスのリカバリを開始します。すべてのプレックスをリカバリすると、プレックスは ACTIVE 状態になります。

メモ: vxattachd は、ユーザーが vxdg detachsite コマンドを使って明示的に切断したサイトの再接続を試みません。

サイトの自動再接続機能はデフォルトで有効になっています。vxattachd デーモンは、サイトの再接続が試みられたり、それらのサイトでプレックスのリカバリが試みられると、電子メールを使って root に通知します。

他のユーザーにメールを送信するには、次の起動スクリプト内にある vxattachd を開始する行にユーザー名を追加して、システムを再ブートします。

RHEL 7、SLES 12、およびサポート対象の RHEL 互換配布の場合:
/etc/vx/vxvm-recover

以前のバージョンの RHEL、SLES、およびサポート対象の RHEL 互換配布の場合:

/etc/init.d/vxvm-recover

サイトの自動リカバリを行わない場合は、vxattachd デーモンを強制終了し、そのデーモンが再起動されないようにします。vxattachd を停止すると、プレックスの自動再接続も停止します。デーモンを強制終了するには、コマンドラインから次のコマンドを実行します。

```
# ps -afe
```

vxattachd のプロセステーブルエントリを見つけ、そのプロセス ID を指定して強制終了します。

```
# kill -9 PID
```

プロセステーブルに vxattachd エントリがなければ、サイトの自動再接続機能は無効になります。

サイトの自動再接続機能を再起動しないようにするには、起動スクリプト vxvm-recover 内の vxattachd を開始する行をコメントアウトします。

20

SFCFSHA を使った iSCSI の管理

この章では以下の項目について説明しています。

SFCFSHA 機能付きの iSCSI について

Storage Foundation Cluster File System High Availability（SFCFSHA）機能付きの iSCSI は、クラスタファイルシステムに存在するファイルによってバッキングされる iSCSI LUN のエクスポートの管理を簡素化する機構を提供します。

前提条件

- ディスクレイアウトバージョン 7 以降であることを確認します。
- iSCSI ターゲットがインストールされ、デーモンが実行状態になっていることを確認します。

svsiscsiadm マニュアルページ

`svsiscsiadm` コマンドは、iSCSI エクスポートの追加、削除、監視を行います。これによって、オペレーティングシステムとともに出荷される iSCSI ターゲットドライバの実装を活用できます。

svsiscsiadm(1M)マニュアルページを参照してください。

SFCFSHA を使った iSCSI の管理

ここでは、このリリースの Storage Foundation Cluster File System High Availability (SFCFSHA)を使用した iSCSI の管理方法について説明します。

iSCSI のクラスタの設定

iSCSI のクラスタを設定するには

- iSCSI のクラスタを設定します。

  ```
  # svsiscsiadm config iqn_prefix
  ```

  次に例を示します。

  ```
  # svsiscsiadm config iqn.2007:07:com.symantec.storage
  ```

ターゲットの作成

ターゲットを作成するには

1. ターゲットを作成します。

   ```
   # svsiscsiadm create target -a ACL CFSMountResource
   ```

   次に例を示します。

   ```
   # svsiscsiadm create target -a 10.172.139.31 1.2.3.4 cfsmount2
   ```

2. 現在の iSCSI エクスポートを表示します。

   ```
   # svsiscsiadm list
   ```

ターゲットへの LUN の追加

デフォルトのオプションを使って LUN とターゲットを作成するには

1 デフォルトのオプションを使って LUN とターゲットを作成します。

```
# svsiscsiadm create lun Path_Of_LUNBackingFile Size_Of_LUN
```

次に例を示します。

```
# svsiscsiadm create lun /mnt0/target1/lun1 1G
```

2 現在の iSCSI エクスポートを表示します。

```
# svsiscsiadm list
```

2 番目の LUN を同じターゲット下に作成するには

1 2 番目のターゲットを同じターゲットに対して作成します。

```
# svsiscsiadm create lun [-t TargetID] LUNBackingFileSize
```

次に例を示します。

```
# svsiscsiadm create lun -t 1 /mnt0/target1/lun2 1G
```

2 現在の iSCSI エクスポートを表示します。

```
# svsiscsiadm list
```

LUN の削除

LUN を削除するには

◆ LUN を削除します。

```
# svsiscsiadm remove lun -f TargetID LUN_ID
```

次に例を示します。

```
# svsiscsiadm remove lun -f 2 1
```

ターゲットの削除

ターゲットを削除するには

1 ターゲットを削除します。

```
# svsiscsiadm remove target TargetID
```

次に例を示します。

```
# svsiscsiadm remove target 1
```

複数のターゲットがある場合は、この手順を繰り返します。

2 現在の iSCSI エクスポートを表示します。

```
# svsiscsiadm list
```

iSCSI のクラスタの設定解除

iSCSI のクラスタを設定解除するには

- iSCSI のクラスタを設定解除します。

```
# svsiscsiadm unconfig
```

FileSnap を使ったクローンの作成

FileSnap を使ってクローンを作成するには

- FileSnap を使ってクローンを作成します。

```
# svsiscsiadm create lun -s PATH_Of_LUNBackingFile ¥
PATH_Of_LUNBackingFileSNAP
```

次に例を示します。

```
# svsiscsiadm create lun -s /mnt1/target1/lun1
/mnt1/target1/lun1_snap
```

iSCSI 対応の SFCFSHA ストレージ共有の vCenter と ESX への追加

iSCSI 対応の SFCFSHA ストレージ共有を vCenter と ESX に追加するには

- iSCSI イニシエータと iSCSI ストレージの設定方法や、iSCSI ストレージの追加について詳しくは、『VMware iSCSI SAN Configuration Guide』 http://www.vmware.com/pdf/vsphere4/r40/vsp_40_iscsi_san_cfg.pdf を参照してください。

ターゲットのオンライン化

ターゲットをオンライン化するには

- ターゲットをオンライン化します。

```
# svsiscsiadm online TargetID
```

次に例を示します。

```
# svsiscsiadm online 1
```

ターゲットのオフライン化

ターゲットをオフライン化するには

- ターゲットをオフライン化します。

```
# svsiscsiadm offline TargetID
```

次に例を示します。

```
# svsiscsiadm offline 1
```

強制的にターゲットをオフライン化するには

- 強制的にターゲットをオフライン化します。

```
# svsiscsiadm offline -f TargetID
```

次に例を示します。

```
# svsiscsiadm offline -f 1
```

LUN の状態の表示

LUN の状態を表示するには

- LUN の状態を表示します。

```
# svsiscsiadm list
```

サンプル出力:

```
Target 1: iqn.2011-07.com.veritas:svst1 /vxfsshare
         1: /vxfsshare/lun23               23G *
         2: /vxfsshare/lun22               22G *
```

クラスタ全体の使用状態を取得するには

- クラスタ全体の使用状態を取得します。

```
# svsisciadm list -s
```

サンプル出力:

```
Target 1: iqn.2011-07.com.veritas:svst1 /vxfsshare
         1: /vxfsshare/lun23         23G *
                fssolspr13                          <Online>
                fssolspr14                          <Online>
         2: /vxfsshare/lun22         22G *
                fssolspr13                          <Online>
                fssolspr14                          <Online>
```

21

SFCFSHA を使ったデータストアの管理

この章では以下の項目について説明しています。

SFCFSHA を使ったデータストアの管理について

Storage Foundation Cluster File System High Availability（SFCFSHA）とともに出荷される svsdatastore ユーティリティは、NFS データストアのエンドツーエンドのプロビジョニングを提供します。ディスク群と仮想 IP 情報を取得し、NFS から VMware ESX に簡単に追加できるデータストアを設定します。また、このユーティリティは、設定された NFS データストアの拡張、縮小、削除などの基本操作を実行できるようにします。このユーティリティは、SFCFSHA スタックのすべての内部詳細を非表示にし、簡素化された概要をユーザーに提供します。

svsdatastore ユーティリティについて

NFS データストアのエンドツーエンドのプロビジョニングを提供します。svsdatastore コマンドは、拡張、縮小、削除などの基本操作を NFS データストアで実行できるようにします。

svsdatastore コマンドでは、次の処理が実行されます。

- マウントポイントを取得し、それに関連付けられている NFS データストアをサイズ変更（拡張と縮小）します。
- ディスク群を取得し、ESX に追加できる NFS データストアを作成します。

- マウントポイントを取得し、それに関連付けられている NFS データストアを削除します。
- IP アドレス、ネットマスク、デバイスを取得し、仮想 IP（VIP）として設定します。
- 完全な CNFS 設定（マウントポイントと仮想 IP アドレス）を表示します。

Cluster Manager ソフトウェアは起動している必要があります。また、`svsdatastore` コマンドを実行する前に `cfscluster config` コマンドを実行してください。権限のあるユーザーのみが、このコマンドを実行できます。

`svsdatastore`（1M）、`cfscluster`（1M）、`cfsshare`（1M）のマニュアルページを参照してください。

NFS データストアの管理

disk_0 と disk_1 のディスクを持つ新しいデータストアを追加するには

◆ disk_0 と disk_1 のディスクを持つ新しいデータストアを追加します。

```
# svsdatastore add disk_0 disk_1
```

サイズが 10 G の disk_0 と disk_1 のディスクを持つ新しいデータストアを追加するには

◆ サイズが 10 G の disk_0 と disk_1 のディスクを持つ新しいデータストアを追加します。

```
# svsdatastore add -s 10g disk_0 disk_1
```

サイズが 10 G の disk_0 と disk_1 のディスクを持つ新しいデータストアをマウントポイントで追加するには

◆ サイズが 10 G の disk_0 と disk_1 のディスクを持つ新しいデータストアをマウントポイントに追加します。

```
# svsdatastore add -s 10g -m /mntpt disk_0 disk_1
```

マウントポイントに関連付けられているデータストアを削除するには

◆ マウントポイントに関連付けられているデータストアを削除します。

```
# svsdatastore delete -m /mntpt
```

マウントポイントに関連付けられているデータストアを 15 G にサイズ変更するには

◆ マウントポイントに関連付けられているデータストアを 15 G にサイズ変更します。

```
# svsdatastore resize -m /mntpt -s 15g
```

仮想 IP「10.192.111.222」とネットマスク「255.255.240.0」をネットワークインターフェース「NIC」で追加するには

- 仮想 IP「10.192.111.222」とネットマスク「255.255.240.0」をネットワークインターフェース「NIC」で追加します。

```
# svsdatastore addvip -i 10.192.111.222 -n 255.255.240.0 -e eth0
```

仮想 IP「10.192.111.222」を設定から削除するには

- 仮想 IP「10.192.111.222」を設定から削除します。

```
# svsdatastore deletevip -i 10.192.111.222
```

データストア設定を表示するには

- データストア設定を表示します。

```
# svsdatastore display
CNFS metadata filesystem : /locks

# MOUNTPOINT          SIZE    SHARE OPTIONS
/defragvol            250G    rw,no_root_squash

# Virtual IP          STATE
10.209.87.147         ONLINE on swlx65
```

5

I/O パフォーマンスの最適化

22

Veritas File System I/O

この章では以下の項目について説明しています。

Veritas File System I/O について

VxFS は次の 2 タイプの基本的なファイルシステム I/O を処理します。

- 順次
- ランダム I/O または非順次 I/O

順次 I/O では、アプリケーションがデータを読み取る時に、VxFS はデフォルトで先読みポリシーを採用します。書き込みでは、可能であれば連続するブロックが割り当てられます。ほとんどの場合、VxFS はバッファ I/O を使って順次 I/O を処理します。VxFS はバッファのないダイレクト I/O を使ってランダム I/O または非順次 I/O を処理します。

VxFS は、ファイルアクセス時に使う I/O キャッシュアドバイザリセットを提供します。

『Veritas File System プログラマリファレンスガイド』を参照してください。

`vxfsio`(7)のマニュアルページを参照してください。

バッファ I/O とダイレクト I/O

VxFS は順次読み取り I/O の先読みに対応します。これは、バッファ I/O になります。アプリケーション用にデータはバッファに事前にフェッチされ保持されます。通常、このデータバッファは VxFS バッファキャッシュと呼ばれます。これは VxFS のデフォルト動作です。

一方のダイレクト I/O は、基本デバイスへの I/O が完了した時にデータをバッファに保持しません。これにより、メモリや CPU などのシステムリソースの使用を節約できます。アラインメントやサイズ基準が満たされている場合にのみ、ダイレクト I/O は可能です。

p.621 の「ダイレクト I/O の必要条件」を参照してください。

サポート対象のすべてのプラットフォームは VxFS バッファキャッシュを備えています。また、各プラットフォームはページキャッシュまたは独自のバッファキャッシュを備えています。通常、これらのキャッシュはファイルシステムキャッシュとして知られています。

ダイレクト I/O ではこのようなキャッシュは使われません。ダイレクト I/O では、I/O が完了すると使ったメモリは破棄されるので、バッファされることはありません。

ダイレクト I/O

ダイレクト I/O は非バッファ形式の I/O です。`VX_DIRECT` キャッシュアドバイザリを設定している場合、ユーザーは、ディスクとユーザーバッファ間でデータを直接転送し、読み書きするように要求できます。この機能により、データはカーネルにバッファリングされず、カーネルバッファとユーザーバッファ間のデータコピーが省略されるため、I/O に関連付けられている CPU オーバーヘッドを削減できます。また、メモリは、バッファキャッシュ分の領域が使われないため、その分有効に活用できます。アプリケーションによっては、ダイレクト I/O 機能によってパフォーマンスが大幅に向上します。

ダイレクト I/O および `VX_DIRECT` キャッシュアドバイザリはファイル記述子単位で保持されます。

ダイレクト I/O の必要条件

ダイレクト I/O として実行される I/O 操作は、整列に関する基準を満たす必要があります。通常、ディスクドライバ、ディスクコントローラ、システムメモリ管理ハードウェアおよびソフトウェアによりアラインメント制約が決まります。

ダイレクト I/O の必要条件は次のとおりです。

- 開始ファイルオフセットは、512 バイトの境界に揃えられている必要があります。
- 終了ファイルオフセットは、512 バイトの境界に揃えられているか、または長さが 512 バイトの倍数である必要があります。
- メモリバッファが 8 バイトの境界上で開始する必要があります。

ダイレクト I/O と同期 I/O

ダイレクト I/O は、同期 I/O と同等のデータ整合性を保持するため、現在、同期 I/O を使っている多くのアプリケーションで使えます。ダイレクト I/O で、領域の割り当てやファイルの拡張を実行しない場合、i ノードへの書き込みはすぐには実行されません。

ダイレクト I/O の CPU オーバーヘッド

ダイレクト I/O の CPU コストは、RAW ディスク転送とほぼ同じです。大容量ファイルへの順次 I/O を、大容量転送サイズを使ったダイレクト I/O に置き換えると、バッファ付き I/O と同等の速度でデータが転送できるため、CPU オーバーヘッドを削減できます。

ファイルの拡張中や、領域の割り当て中には、ダイレクト I/O は i ノード更新の書き込みが、アプリケーションに制御を戻す前に実行される必要があります。このため、ダイレクト I/O の一部のパフォーマンスの優位性が低下します。

ディスカバードダイレクト I/O

ディスカバードダイレクト I/O は、vxtunefs コマンドを使って設定するファイルシステムのチューニングパラメータです。ファイルシステムはカーネルパラメータである discovered_direct_iosz の設定値を超える大容量 I/O 要求を受け取ると、ダイレクト I/O を試みます。大容量 I/O の場合、ディスカバードダイレクト I/O はバッファ付き I/O よりも高性能を発揮します。

ディスカバードダイレクト I/O はダイレクト I/O と類似しており、データブロックの整列条件に対して同じ制約があります。ただし、格納領域を割り当てる操作、ファイルのサイズを拡張する書き込み操作を実行する際に、i ノード更新の書き込みがアプリケーションに制御を戻す前に実行される必要がありません。

非バッファ I/O

VX_UNBUFFERED キャッシュアドバイザリを設定すると、VX_DIRECT キャッシュアドバイザリが設定されているダイレクト I/O と同じ I/O 動作になるため、ダイレクト I/O に適用されるアラインメント制約は非バッファ I/O にも適用されます。ただし、非バッファ I/O では、ファイルの拡張中や領域の割り当て中に際して、i ノード更新の同期が制御をアプリケーションに戻す前に実行されることはありません。VX_UNBUFFERED キャッシュアドバイザリはファイル記述子単位で保持されます。

データ同期 I/O

VX_DSYNC キャッシュアドバイザリが設定されている場合、ユーザーはデータ同期 I/O を要求します。データ同期 I/O では、データが書き込まれると、更新時刻および拡張したファイルサイズ（必要な場合）とともに i ノードへの書き込みも行われます。データ同期 I/O では、データ制御をアプリケーションに戻す前に、ディスクへ同期的に転送されます。書き込みでファイルが拡張されない場合は、データがメモリ上で更新された段階で、制御を

アプリケーションに戻します。書き込み操作でファイルが拡張される場合は、制御をアプリケーションに戻す前に、i ノード更新の書き込みが実行されます。

ダイレクト I/O および VX_DSYNC キャッシュアドバイザリはファイル記述子単位で保持されます。

データ同期 I/O と同期 I/O

ダイレクト I/O と同様に、データ同期 I/O 機能でも、アプリケーションのパフォーマンスが大幅に向上します。データ同期 I/O は、同期 I/O と同等のデータ整合性を保持するため、現在、同期 I/O を使っている多くのアプリケーションで使えます。データ同期 I/O で、領域の割り当てやファイルの拡張を実行しない場合、i ノードへの書き込みはすぐには実行されません。データ同期 I/O にはアラインメント制約がないため、ダイレクト I/O のアラインメント制約に準拠するのが難しいアプリケーションには、データ同期 I/O の使用をお勧めします。

データ同期 I/O ではファイルの拡張中や領域の割り当て中には、制御をアプリケーションに戻す前に、i ノード更新の書き込みが実行される必要があります。このため、データ同期 I/O のパフォーマンスが低下します。

同時 I/O

同時 I/O (VX_CONCURRENT)によって、複数のプロセスは他の read(2) 呼び出しまたは write(2)呼び出しをブロックすることなく、同じファイルに読み書きを実行できます。POSIX セマンティクスでは、read 呼び出しと write 呼び出しをファイル上で他の read 呼び出しと write 呼び出しとシリアル化する必要があります。POSIX のセマンティクスでは、read 呼び出しによるデータの読み取りは、write 呼び出しの前か後になります。VX_CONCURRENT キャッシュアドバイザリを設定すれば、キャラクタデバイスの場合と同じく、read 操作と write 操作はシリアル化されません。このキャッシュアドバイザリは通常、同じファイルに対する重複書き込みを実行しないアプリケーションでデータアクセスのパフォーマンスを高くしなければならない場合に使います。同時 I/O を使うときに、同じファイルに対する write 操作を調整するのは、アプリケーションまたは実行中のスレッドの責任です。

同時 I/O は、以下の方法で有効にできます。

- VX_SETCACHE ioctl コマンドのファイル記述子として VX_CONCURRENT アドバイザリフラグを指定します。そのファイル記述子で実行する read(2)呼び出しと write(2)呼び出しだけが同時 I/O を使います。同じファイルでも、他のファイル記述子で実行する read 操作と write 操作は、POSIX のセマンティクスに準拠することになります。vxfsio(7)のマニュアルページを参照してください。
- cio mount オプションを使います。そのファイルシステムでは、すべてのファイルに対する read(2)操作と write(2)操作が同時 I/O を使います。
p.273 の「cio マウントオプション」を参照してください。

mount_vxfs(1M)のマニュアルページを参照してください。

キャッシュアドバイザリ

VxFS を使うと、アプリケーションがファイルにアクセスするときに使うキャッシュアドバイザリを設定できます。VxFS キャッシュアドバイザリを使うと、アプリケーションでバッファキャッシュを監視し、パフォーマンス向上のためのバッファキャッシュの最適な調整方法を提供できるようになります。

キャッシュアドバイザリは、バッファキャッシュを大きくすればブロック X の後の再読み取りを防げたかどうかを報告します。また逆に、キャッシュアドバイザリは、ブロック X を危険にさらすことなくバッファキャッシュサイズを安全に小さくすることができたかどうかも報告します。

キャッシュアドバイザリはメモリ内にのみ存在し、再起動すると消失します。一部のキャッシュアドバイザリは、現在、ファイル記述子単位ではなくファイル単位で保持されています。1 つのファイルへのすべてのアクセスに有効なアドバイザリは 1 セットのみになります。2 つの競合するアプリケーションに異なるキャッシュアドバイザリを設定すると、両方のアプリケーションは最後に設定したキャッシュアドバイザリを使う必要があります。

キャッシュアドバイザリはすべて、VX_SETCACHE ioctl システムコールを使って設定されます。現在のキャッシュアドバイザリの設定は、VX_GETCACHE ioctl システムコールを使って取得できます。

vxfsio(7)のマニュアルページを参照してください。

ファイルシステムのフリーズとアンフリーズ

ファイルシステムのフリーズ処理は、ファイルシステムの、ボリュームレベルで一貫性のある安定したイメージを取得するために必要な手順です。ボリュームレベルで一貫性のあるファイルシステムイメージは、ファイルシステムスナップショットツールを使って取得し、使うことができます。フリーズ処理によって、ダーティなメタデータやユーザーデータが格納されたファイルシステムキャッシュ内のすべてのバッファおよびページがフラッシュされます。さらに、この操作によって、ファイルシステムがアンフリーズされるまで、ファイルシステムに対するすべての新しい操作が保留されます。

VX_FREEZE ioctl コマンドを使うと、ファイルシステムがフリーズします。ファイルシステムをフリーズすると、ファイルシステムへのすべての I/O 処理が一時的にブロックされ、ファイルシステム上で sync が実行されます。VX_FREEZE ioctl システムコールが発行されると、システムコールレベルでファイルシステムへのアクセスがすべて中断されます。現在の操作が完了して、ファイルシステムはディスクに同期されます。

ファイルシステムがフリーズすると、VX_THAW ioctl コマンドを使う場合を除いて、プロセスで VX_THAW ioctl コマンドを実行するか、フリーズ制限時間が経過するまで、フリーズしたファイルシステムは使えなくなります。

I/O サイズの取得

VxFS には、VX_GET_IOPARAMETERS ioctl システムコールが用意されており、ファイルシステムで使うために推奨される I/O サイズを取得できます。この ioctl システムコールをアプリケーションで使うと、VxFS に発行されるファイルやファイルデバイスの I/O サイズを決定できます。

vxtunefs(1M)および vxfsio(7)の各マニュアルページを参照してください。

Veritas InfoScale 製品コンポーネントのデータベースアクセラレータについて

どのような環境でも、相応のパフォーマンスを維持することや、パフォーマンス SLA(Service Level Agreements)を満たすことが主要な懸案事項です。Veritas InfoScale 製品コンポーネントは、様々な方法によってデータベース環境全体のパフォーマンスを向上させます。

表 22-1 Veritas InfoScale 製品コンポーネントのデータベースアクセラレータ

Veritas InfoScale データベースアクセラレータ	サポート対象のデータベース	使用例と注意事項
Oracle Disk Manager (ODM)	Oracle	■ ファイル I/O に対する最先端のカーネルサポートを含む改善されたアプリケーションプログラミングインターフェース(API)を利用して、Oracle のパフォーマンスを向上させ、システム帯域幅を管理します。 ■ Oracle Resilvering を使い、Veritas Volume Manager DRL(Dirty Region Logging)をオフにしてパフォーマンスを向上させるには、ODM を使います。 ■ ミッションクリティカルなアプリケーションの I/O 帯域幅をより多く利用できるようにして、データベースの一貫性を復元するために必要な時間を短縮するには、SmartSync Recovery Accelerator を使います。
Cached Oracle Disk Manager(Cached ODM)	Oracle	選択した I/O でキャッシュを使って ODM I/O のパフォーマンスを向上させるには、Cached ODM を使います。

Veritas InfoScale データベースアクセラレータ	サポート対象のデータベース	使用例と注意事項
同時 I/O	DB2 Sybase	CIO（Concurrent I/O）は、DB2 環境および Sybase 環境向けに最適化されています。 ファイルサイズの拡張に関する制限なしで、VxFS ファイルシステム上で実行されるデータベースのパフォーマンスを向上させるには、Veritas InfoScale Concurrent I/O を使います。

これらのデータベースアクセラレータ技術によって、データベースのパフォーマンスは RAW ディスクパーティションと同等になりますが、さらにファイルシステムの管理性が得られるという利点があります。Storage Foundation の DMP（Dynamic Multi-Pathing）機能により、サーバーからアレイまでのすべての使用可能なパスで I/O アクティビティの負荷を分散することで、パフォーマンスが最大限に引き上げられます。DMP は主要なハードウェア RAID ベンダーをすべてサポートしているので、サードパーティ製のマルチパスソフトウェアが不要となり、TCO を削減できます。

Veritas InfoScale データベースアクセラレータを使うと、データベースのパフォーマンスをより高精度で管理できるようになります。

ODM および Cached ODM for Oracle の使用について詳しくは、『Veritas InfoScale Oracle データベース用ストレージと可用性管理』を参照してください。

DB2 での同時 I/O の使用について詳しくは、『Veritas InfoScale DB2 データベース用ストレージと可用性管理』を参照してください。

23

Veritas Volume Manager I/O

この章では以下の項目について説明しています。

Veritas Volume Manager の管理 I/O の調整

Veritas Volume Manager（VxVM）は管理 I/O の調整機能を提供します。重い I/O 負荷がかかっている間、VxVM は管理処理を行うために作成する I/O を調整します。この動作によって、管理上の I/O がアプリケーションの I/O パフォーマンスに影響しない状態を確保できます。アプリケーションの I/O 負荷が軽いときは、VxVM では管理上の I/O 操作のための帯域幅の使用量を増やします。

VxVM によって、ストレージで認識された負荷に基づき、管理タスクの I/O 調整が自動的に管理されます。現在 I/O 調整は、ATOMIC_COPY を使い、1 つの対象のミラーが関係するコピー操作でサポートされます。I/O 調整は透過的に行われ、コマンドの使用方法や出力が変更されることはありません。次のコマンドがサポートされます。

- `vxassist mirror`
- `vxassist snapcreate`
- `vxevac`
- `vxplex att`
- `vxplex cp`
- `vxplex mv`
- `vxsnap addmir`

- vxsnap reattach
- vxsd mv

管理 I/O 操作は、I/O のメモリを独立したメモリプールから割り当てます。このプールの最大サイズは、vol_max_adminio_poolsz チューニングパラメータを使って調整できます。

たとえば、vol_max_adminio_poolsz パラメータを 256 MB に変更するには、次のコマンドを使います。

```
# vxtune vol_max_adminio_poolsz 256M
```

詳しくは vxtune (1M) コマンドを参照してください。

最大 IOPS 設定を使ったアプリケーション I/O 負荷の管理

複数のアプリケーションで共通のストレージサブシステムが使用される場合、複数のアプリケーションが共有環境で共存できるようにアプリケーション I/O 要求のバランスを取ることが重要です。このニーズに対応するには、アプリケーションのボリュームに対し IOPS (I/O Per Second) で最大しきい値を設定します。アプリケーションのボリュームはグループ化され、アプリケーションボリュームグループが形成されます。最大 IOPS 制限によって、アプリケーションボリュームグループ内のすべてのボリュームによって一括して処理される IOPS の最大数が決定します。

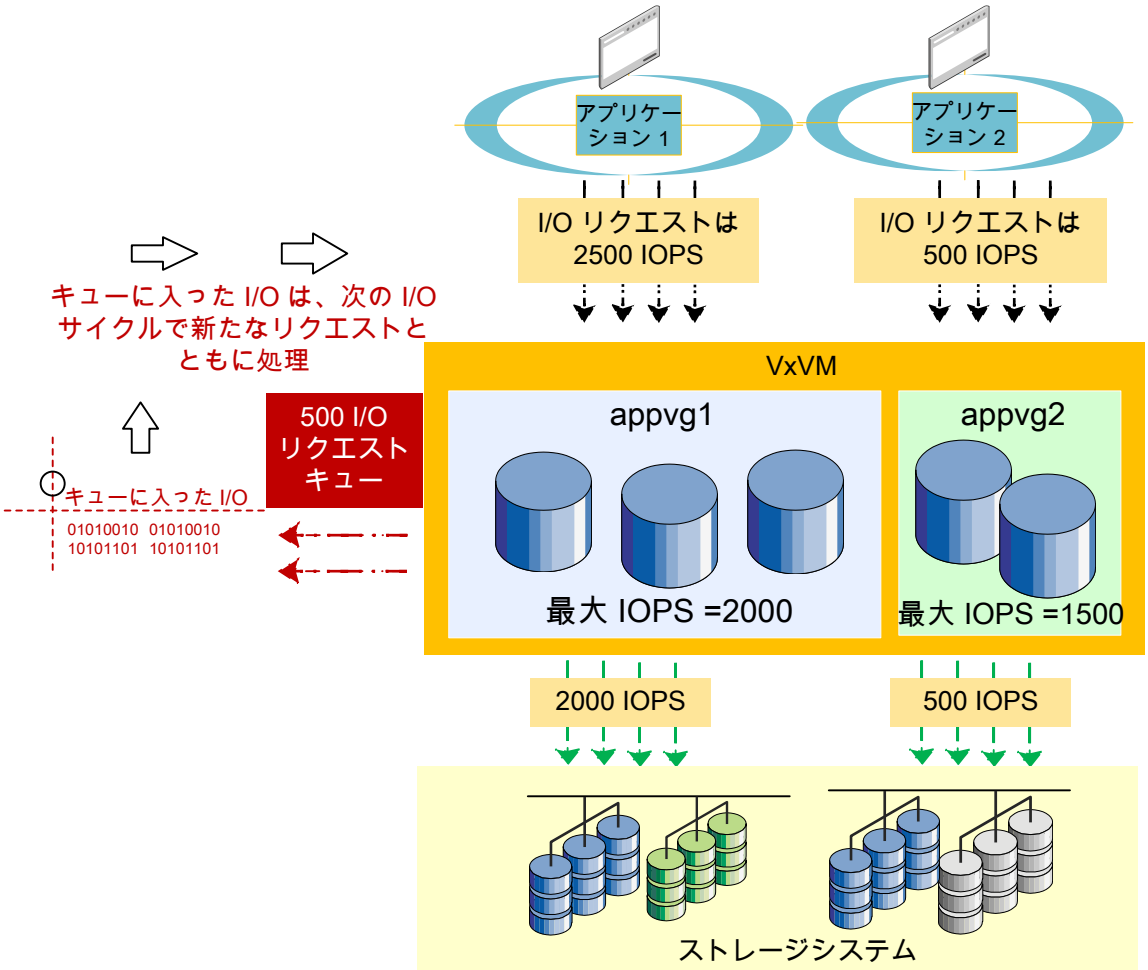
I/O 要求はアプリケーションから送信されると、アプリケーションボリュームグループが IOPS 制限に到達するまでグループ内のボリュームによって処理されます。特定の時間間隔にグループがこの制限を超えると、グループ上の以降の I/O 要求がキューイングされます。キューイングされた I/O は次の時間間隔に、アプリケーションからの新しい I/O 要求とともに優先されます。

最大 IOPS しきい値の設定時に、次の要因を考慮します。

- 共有サブシステムのストレージ容量
- アクティブなアプリケーションの数
- 個別のアプリケーションの I/O 要件
- VxVM 管理 I/O
  たとえば、ボリュームに VxVM インスタントまたは領域最適化スナップショットが関連付けられている場合、ソースボリュームに設定するしきい値を小さくします。

図 23-1 はプロセスを示しています。

図 23-1 アプリケーション I/O 負荷の管理



アプリケーションボリュームグループについて

アプリケーションボリュームグループは、アプリケーションと関連付けられているボリュームの論理グループです。グループには 1 つ以上のボリュームが含まれていることがあります。アプリケーションボリュームグループ内のすべてのボリュームは、同じディスクグループから選択する必要があります。ボリュームはプライベートディスクグループまたは共有ディスクグループに属している場合があります。アプリケーションボリュームグループに最大 IOPS しきい値を設定して、複数のアプリケーション I/O 負荷のバランスを取ります。IOPS 値は、アプリケーションボリュームグループ内のすべてのボリュームに対して組み合わせたしきい値として設定されます。

そのようなアプリケーションボリュームグループを複数構築して、似たような I/O 負荷特性を持つボリュームを結合させることができます。たとえば、データベースアプリケーションの REDO ログボリュームの I/O 負荷特性はアプリケーションのアーカイブボリュームのものと異なるため、異なる IOPS しきい値が必要になります。そのようなシナリオで、別のアプリケーションボリュームグループを作成し、各 IOPS しきい値を設定します。ボリュームは最大 IOPS 制限を設定するために複数のアプリケーションボリュームグループ間で共有することはできません。

クラスタ化された環境で、アプリケーションボリュームグループを作成し、そのしきい値をクラスタ内のノードから設定できます。アプリケーションボリュームグループはすべてのノードで表示され、同じしきい値の値がクラスタ内のすべてのノードに伝播されます。ただし、しきい値は各ノードに個別に適用されます。

vxvolgrp コマンドを使用して、アプリケーションボリュームグループとその設定を管理できます。

vxvolgrp（1M）のマニュアルページを参照してください。

vxstat コマンドにより、各アプリケーションボリュームグループの IOPS 統計の詳細ビューが提供されます。

vxstat（1M）マニュアルページを参照してください。

アプリケーションボリュームグループの作成

既存のアプリケーションボリュームグループの一部ではないボリュームを識別します。選択するボリュームが同じディスクグループに属していることを確認します。

vxvolgrp コマンドを使用して、アプリケーションボリュームグループを作成できます。

vxvolgrp（1M）のマニュアルページを参照してください。

アプリケーションボリュームグループを作成するには

1 アプリケーションボリュームグループを作成します。

ボリューム *datavol1*、*datavol2*、*datavol3* から成るアプリケーションボリュームグループ *datavol_grp* が作成されます。*oradg* は、ボリュームが選択されるディスクグループの名前です。

```
# vxvolgrp -g oradg make datavol_grp ¥
datavol1 datavol2 datavol3
```

2 アプリケーションボリュームグループが正常に作成されていることを確認します。

次に例を示します。

```
# vxvolgrp -g oradg list datavol_grp
Volume Group:                   datavol_grp
volume(s):                      datavol1 datavol2 datavol3
```

アプリケーションボリュームグループの一覧の表示

vxvolgrp コマンドを使用して、以下のことが行われます。

- 既存のアプリケーションボリュームグループの一覧の表示
- 特定のアプリケーションボリュームグループに関する詳細情報の表示

vxvolgrp (1M) マニュアルページを参照してください。

既存のアプリケーションボリュームグループの一覧を表示するには、次の操作を行います。

```
# vxvolgrp -g dgname list
```

oradg ディスクグループのボリュームから作成されたアプリケーションボリュームグループの一覧を表示するには、次の操作を行います。

```
# vxvolgrp -g oradg list
Volume Group:                   datavol_grp
volume(s):                      datavol1 datavol2 datavol3

Volume Group:                   logvol_grp
volume(s):                      logvol1 logvol2 logvol3 logvol4
```

たとえば、アプリケーションボリュームグループ *datavol_grp* に関する情報を表示するには:

```
# vxvolgrp -g oradg list datavol_grp
Volume Group:                   datavol_grp
volume(s):                      datavol1 datavol2 datavol3
```

アプリケーションボリュームグループでの最大 IOPS しきい値の設定

最大 IOPS しきい値は、アプリケーションと関連付けられているアプリケーションボリュームグループに設定されます。これは、グループ内のすべてのボリュームに適用するグループしきい値です。IOPS 設定はアプリケーションの実行中を含めいつでも更新できます。

vxvolgrp コマンドを使用して、グループに対し最大 IOPS 値を設定できます。

vxvolgrp（1M）のマニュアルページを参照してください。

メモ: グループ内の個別のボリュームに対ししきい値を設定することはできません。

アプリケーションボリュームグループに最大 IOPS しきい値を設定するには

1 アプリケーションボリュームグループに対し最大 IOPS 値を設定します。

たとえば、次のコマンドは、アプリケーションボリュームグループ *datavol_grp* に最大 IOPS しきい値 *1000* を設定します。

```
# vxvolgrp -g oradg set datavol_grp maxiops=1000
```

2 最大 IOPS しきい値が正常に設定されていることを確認します。

次に例を示します。

```
# vxvolgrp -g oradg list datavol_grp
Volume Group:                   datavol_grp
volume(s):                      datavol1 datavol2 datavol3
volume group attributes:        maxiops=1000
```

アプリケーションボリュームグループの IOPS 統計の表示

vxstat コマンドの -G オプションにより、アプリケーションボリュームグループの IOPS 統計の詳細ビューが提供されます。1 秒あたりの統計、または指定した時間間隔に蓄積された 1 秒あたりの統計の平均の詳細ビューを取得できます。

クラスタ化された環境の場合、クラスタ内のノードごとに詳細な統計が表示されます。

vxstat(1M)マニュアルページを参照してください。

次の情報を表示できます。

MaxIOPS	1 つのアプリケーションボリュームグループに対して設定される最大 IOPS しきい値
ServicedIOPS	1 つのアプリケーションボリュームグループに対して処理される 1 秒あたりの I/O の数
QueuedIOPS	1 つのアプリケーションボリュームグループに対して調整される 1 秒あたりの I/O の数

たとえば、既存のアプリケーションボリュームグループの 1 秒あたりの IOPS 統計を表示するには:

```
# vxstat -g oradg -G  <<<< average per second output
                                    AVG PER SECOND VOLUMEGROUP STATISTICS
TYP NAME          MaxIOPS     IncomingIOPS    ServicedIOPS       QueuedIOPS(Transient)
grp datavol_grp    1000          897             893               56
grp logvol_grp     600           599             598               404
```

アプリケーションボリュームグループ *datavol_grp* の 1 秒あたりの IOPS 統計を表示するには:

```
# vxstat -g oradg -G datavol_grp
                                    AVG PER SECOND VOLUMEGROUP STATISTICS
TYP NAME         MaxIOPS     ServicedIOPS       QueuedIOPS(Transient)
grp datavol_grp  1000             980              360
```

アプリケーションボリュームグループ *datavol_grp* の 3 秒間隔の平均 IOPS 統計を表示するには:

```
# vxstat -g oradg -G datavol_grp -i 3
                                    AVG PER SECOND VOLUMEGROUP STATISTICS
TYP NAME                            MaxIOPS   ServicedIOPS  QueuedIOPS(Transient)

vmr720-23 Tue 12 Jan 2016 02:42:52 PM UTC
grp datavol_grp                        1000   1000           386

vmr720-23 Tue 12 Jan 2016 02:42:55 PM UTC
grp datavol_grp                        1000   1000           393

vmr720-23 Tue 12 Jan 2016 02:42:58 PM UTC
grp datavol_grp                        1000   1000           393

vmr720-23 Tue 12 Jan 2016 02:43:01 PM UTC
grp datavol_grp                        1000   1000           391

vmr720-23 Tue 12 Jan 2016 02:43:04 PM UTC
grp datavol_grp                        1000    589           129

vmr720-23 Tue 12 Jan 2016 02:43:07 PM UTC
grp datavol_grp                        1000      0           0
```

アプリケーションボリュームグループからの最大 IOPS 設定の削除

vxvolgrp コマンドを使用して、最大 IOPS 設定を削除できます。

vxvolgrp(1M)のマニュアルページを参照してください。

たとえば、アプリケーションボリュームグループ *datavol_grp* から最大 IOPS 設定を削除するには:

```
# vxvolgrp -g oradg clear datavol_grp maxiops
# vxvolgrp -g oradg list datavol_grp
```

```
Volume Group:                   datavol_grp
volume(s):                      datavol1 datavol2 datavol3
```

アプリケーションボリュームグループへのボリュームの追加

ボリュームを既存のアプリケーションボリュームグループに追加できます。ディスクグループ内からボリュームを選択してください。ボリュームはその他のアプリケーションボリュームグループに属していない必要があります。VxVM は新しいボリュームをアプリケーションボリュームグループの名前および対応する最大 IOPS 値とタグ付けします。

vxvolgrp コマンドを使用して、ボリュームをアプリケーションボリュームグループに追加できます。

vxvolgrp(1M)のマニュアルページを参照してください。

たとえば、ボリュームをアプリケーションボリュームグループ *logvol_grp* に追加するには:

```
# vxvolgrp -g oradg addvol logvol_grp ¥
logvol3 logvol4
# vxvolgrp -g oradg list logvol_grp
Volume Group:                   logvol_grp
volume(s):                      logvol1 logvol2 logvol3 logvol4
volume group attributes:        maxiops=1200
```

アプリケーションボリュームグループからのボリュームの削除

アプリケーションボリュームグループからボリュームを削除すると、VxVM はアプリケーションボリュームグループ名と IOPS しきい値をボリュームから消去します。グループ内に残っている唯一のボリュームを削除すると、アプリケーションボリュームグループが削除されます。

vxvolgrp コマンドを使用して、アプリケーションボリュームグループからボリュームを削除できます。

vxvolgrp(1M)のマニュアルページを参照してください。

たとえば、アプリケーションボリュームグループ *datavol_grp* からボリュームを削除するには:

```
# vxvolgrp -g oradg rmvol datavol_grp datavol3
# vxvolgrp -g oradg list datavol_grp
Volume Group:                   datavol_grp
volume(s):                      datavol1 datavol2
volume group attributes:        maxiops=1000
```

アプリケーションボリュームグループの削除

アプリケーションボリュームグループを削除すると、ボリュームが解放され、他のアプリケーションボリュームグループで使用できるようになります。IOPS 設定およびその関連付けられているタグがボリュームから削除されます。

アプリケーションボリュームグループを削除するには、vxvolgrp コマンドを使用します。

vxvolgrp（1M）のマニュアルページを参照してください。

たとえば、アプリケーションボリュームグループ *logvol_grp* を削除するには:

```
# vxvolgrp -g oradg remove logvol_grp
```

6

Veritas Extension for Oracle Disk Manager

24

Veritas Extension for Oracle Disk Manager の使用

この章では以下の項目について説明しています。

Oracle Disk Manager について

Veritas Extension for ODM (Oracle Disk Manager) は、特に Oracle10g 以降でファイル管理とディスク I/O のスループットを拡張するために設計されました。Oracle Disk Manager の機能は、Veritas File System のファイルシステム上に構成されたデータベースの最適化機能の 1 つです。Oracle Disk Manager を使うと、Oracle10g 以降のユー

ザーは、特殊な I/O 最適化により、I/O の多い作業負荷を持つデータベースの性能を改善できます。

Veritas Extension for Oracle Disk Manager は Oracle Resilvering をサポートします。Oracle Resilvering では、ストレージ層が Oracle データベースから、ミラー化データファイルのどの領域またはブロックをシステムクラッシュ後に再同期するのかという情報を受信します。Oracle Resilvering は VxVM DRL からのオーバーヘッドを避けてパフォーマンスを高めます。

Oracle Disk Manager では Oracle Managed Files の拡張サポートが用意されており、管理オーバーヘッドが減少します。Veritas Extension for Oracle Disk Manager はユーザーに対して透過的です。Veritas Extension for Oracle Disk Manager を使って管理するファイルでは、特別なファイル命名規則が必要ありません。Oracle Disk Manager インターフェースは通常のデータベースファイルを使います。

データベース管理者は、Oracle 製品で使うデータファイルのタイプを選択できます。従来、ファイルシステムに配置したファイルと RAW デバイスのどちらを選択するかは、管理性およびパフォーマンスに基づいて行われました。ただし、Oracle Parallel Server での使用を目的とするデータベースは例外です。この場合、多くのプラットフォームで RAW デバイスが必要です。パフォーマンスよりも管理性が重要な場合、一般に、ファイル形式としてファイルシステムのファイルが推奨されます。ただし、アプリケーションによっては、最初に実装されたときに十分な I/O 必要条件が満たせないような場合、I/O 必要条件が変更されることがあります。アプリケーションが I/O スループットに依存するようになると、ファイルシステムから RAW デバイスへのデータファイルの変換が必要になる場合があります。

Oracle Disk Manager は、パフォーマンスと管理性の両方を実現するように設計されています。Oracle Disk Manager は、VxFS ファイルシステム上のデータベースストレージでの Oracle のファイル管理と I/O 呼び出しをサポートします。この機能は、動的にロードされる共有ライブラリとして提供され、この共有ライブラリはロードされるときに Oracle にバインドされます。Oracle Disk Manager ライブラリは、起動時にカーネルにロードされて、Oracle Disk Manager ドライバと連動します。

Oracle Disk Manager を使うメリットは次のとおりです。

- ファイルに対する本格的なカーネル非同期 I/O
- システムコールに対するオーバーヘッドの軽減
- VxFS ファイルシステムに隣接するようにファイルに領域を事前に割り当てることによるファイルシステムレイアウトの改善
- ファイルシステム上のファイルでの RAW デバイスと同じパフォーマンス
- ユーザーに対する透過性
- 連続するデータファイルの割り当て

Oracle Disk Manager によるデータベースパフォーマンスの改善方法

Oracle Disk Manager では、次により、VxFS ファイルシステムに対するデータベースの I/O パフォーマンスを向上させます。

- カーネルの非同期 I/O のサポート
  p.639 の「カーネルの非同期 I/O サポートについて」を参照してください。
- ダイレクト I/O のサポートと二重バッファリングの回避
  p.639 の「ダイレクト I/O サポートと二重バッファリングの回避について」を参照してください。
- データベースファイルに対するカーネル書き込みロックの回避
  p.640 の「データベースファイルのカーネル書き込みロックの回避について」を参照してください。
- 1 回のシステムコールでの複数の同時 I/O のサポート
  p.640 の「1 回のシステムコールでの複数の同時 I/O のサポートについて」を参照してください。
- 複数の Oracle インスタンスによるファイルの二重オープンの回避
  p.640 の「ファイルのオープンコールの重複の回避について」を参照してください。
- 連続するデータファイルの割り当て
  p.640 の「連続するデータファイルの割り当てについて」を参照してください。

カーネルの非同期 I/O サポートについて

非同期 I/O では、ブロッキングされていないシステムレベルの読み取りおよび書き込みが実行されます。これにより、システムは複数の I/O 要求を同時に実行できます。カーネル非同期 I/O は、コンテキストの切り替えを最小限にして作業を完了させるために、I/O がカーネル内のディスクデバイスドライバにキューイングされるため、ライブラリ非同期 I/O より優れています。

ダイレクト I/O サポートと二重バッファリングの回避について

read() および write() システムコールを使うファイルシステムに対する I/O は、通常、データを 2 回コピーします。1 回目はユーザーとカーネル領域の間で、2 回目はカーネル領域とディスクの間で実行されます。対照的に、RAW デバイスの I/O は、ユーザー領域とディスク間で直接コピーされるため、1 回分のコピー操作が短縮されます。RAW デバイスの I/O と同様に、Oracle Disk Manager の I/O でも余分のコピーが回避されます。Oracle Disk Manager は、システムキャッシュをバイパスし、RAW デバイスと同じ効率でファイルにアクセスします。二重バッファリングの回避により、システム上のメモリオーバーヘッドを軽減できます。カーネルからユーザーのアドレス空間へのコピーが回避されるため、カーネルモードでのプロセッサの使用率が大幅に軽減され、解放されたプロセッササイクルによって今までよりも多くのアプリケーションコードを実行できます。

データベースファイルのカーネル書き込みロックの回避について

write() システムコールを使ってデータベース I/O を実行すると、システムコールごとに、ファイルに対するカーネルの書き込みロックの取得および解放が行われます。このロックにより、同一ファイルで同時に書き込み操作が実行されるのを防ぎます。通常、データベースシステムには、ファイルに対する同時アクセスを管理する独自のロック機能が実装されているため、ファイルごとの書き込みロックは不要な I/O 操作です。Oracle Disk Manager は、ファイルシステムによるロックをバイパスし、データベースサーバーがデータアクセスを制御できるようにします。

1 回のシステムコールでの複数の同時 I/O のサポートについて

非同期 I/O を実行すると、Oracle プロセスは I/O の完了を待つ間は他に実行できる処理が存在しないため、完了を待ちながら追加 I/O 要求を発行したり、特定の I/O 要求を待機する場合があります。また、Oracle プロセスは別のファイルに対して要求を発行する場合もあります。このすべての動作は、Oracle で Oracle Disk Manager I/O インターフェースが使われていれば、1 回のシステムコールで実現できます。このインターフェースによって、同じ作業を完了するために実行されるシステムコール数が削減されるため、プロセス内のユーザー空間とカーネル空間のコンテキスト切り替え回数が減ります。

ファイルのオープンコールの重複の回避について

Oracle Disk Manager では、「ファイル識別子」が用意されており、ファイルを 1 回のみ開くことが許されます。これはファイルの「識別」と呼ばれます。同じファイル識別子を、Oracle インスタンス内の他のプロセスで使うことができます。ファイルの状態は、カーネル内の Oracle Disk Manager ドライバによって保持されます。ファイルのオープンコール数が減少することで、プロセスの初期化時と終了時の処理オーバーヘッドが減少し、また、カーネル内で必要なファイル状態の構造数が減少します。

連続するデータファイルの割り当てについて

Oracle Disk Manager は、ソートクエリーやパラレルクエリーなどの一時表領域を使うクエリーのパフォーマンスを向上させることができます。Oracle Disk Manager を使わない場合、Oracle は一時表領域用のデータファイルを初期化しません。したがって、データファイルはスパースファイルになり、通常は断片化されます。スパースファイルまたは断片化されたファイルでは、クエリーパフォーマンスが低下します。Oracle Disk Manager を使うと、データファイルは一時表領域用に初期化されて連続的に割り当てられるため、スパースファイルにはなりません。

Oracle Disk Manager と Storage Foundation Cluster File System High Availability について

ODM（Oracle Disk Manager）は、SFCFSHA 環境内にあるクラスタファイルへのアクセスをサポートします。Veritas InfoScale™ Storage ライセンスがある場合、ODM はシリアル排他モードで SFCFSHA ファイルをサポートします。このモードでは、一度に 1 つのノードから各 SFCFSHA ファイルにアクセスできますが、複数ノードから同時アクセスすることはできません。

クラスタサポートモードについて詳しくは `mount.vxodmfs`（8）マニュアルページを参照してください。

Oracle Disk Manager と Oracle Managed Files について

OMF (Oracle Managed Files) では、ファイル名、ファイルの場所、ストレージ属性、ファイルがデータベースで使用されているかどうかなどのデータファイル属性が管理されます。OMF は、ファイルシステム内に存在するデータベースに対してのみサポートされます。OMF の機能は、Oracle Disk Manager によって大幅に拡張されています。

OMF はファイル管理機能の 1 つであり、次の機能を持っています。

- 一意なファイル名の指定が不要
- 表領域自動拡張機能により、動的な領域管理を提供

OMF の主な必要条件は、データベースがファイルシステムのファイルに配置されていることです。また、ファイルシステム自体に必要な前提条件は、他にもあります。

OMF は、動的なファイルシステムの拡張をサポートしているストライプ化論理ボリューム内に構成したファイルシステム上でのみ使用してください。また、OMF の使用を対象としたファイルシステムでは、表領域の自動拡張を容易にするため、大容量かつ拡張可能なファイルをサポートする必要があります。このため、RAW パーティションは、OMF には使えません。

デフォルトでは、自動拡張性を伴う OMF データファイルが作成されます。この属性は、既存のデータベースの保持および新しいアプリケーションの実装と関連するキャパシティ計画を軽減します。時間の経過とともに拡張される表領域に伴い発生するディスクの断片化により、データベース管理者には、自動拡張可能な表領域を考慮する際に注意が必要でした。Oracle Disk Manager はこの点を改善しました。

Oracle Disk Manager を OMF と組み合わせて使うと、自動拡張時に表領域に割り当てられる領域も含めて、データファイルには必ず隣接したディスク容量が割り当てられるように、Veritas Extension for Disk Manager で考慮されます。表および索引スキャンのスループットは、表領域の拡張に伴って低下することはありません。

Oracle Disk Manager と Oracle Managed Files の連携

次の例は、Oracle Disk Manager と Oracle Managed Files (OMF) との関係を示しています。例では、init.ora の内容とデータベースインスタンスを起動するコマンドを示します。Oracle の元に戻す管理を簡略化するために、init.ora パラメータ UNDO_MANAGEMENT はAUTO に設定されます。これは、システム管理 UNDO と呼ばれます。

メモ: OMF データベースを構築する前に、init.ora の適切なデフォルト値が必要です。これらの値により、CREATE DATABASE 文の実行後に、SYSTEM 表領域、オンライン REDO ログ、制御ファイルの場所を制御できます。

```
$ cat initPROD.ora
UNDO_MANAGEMENT = AUTO
DB_CREATE_FILE_DEST = '/PROD'
DB_CREATE_ONLINE_LOG_DEST_1 = '/PROD'
db_block_size = 4096
db_name = PROD
$ sqlplus /nolog
SQL> connect / as sysdba
SQL> startup nomount pfile= initPROD.ora
```

Oracle インスタンスが起動します。

```
Total System Global Area 93094616 bytes
Fixed Size 279256 bytes
Variable Size 41943040 bytes
Database Buffers 50331648 bytes
Redo Buffers 540672 bytes
```

EMP_TABLE 表領域に関連付けられたファイルを、EMP_INDEX 表領域とは別のディレクトリに配置するレイアウトを実装するには、ALTER SYSTEM 文を使います。この例では、OMF によるファイル名、storage 句とパスの取り扱い方法を示します。このレイアウトにより、表領域をデータファイルの集合ではなくファイルシステム内のオブジェクトと見なすことができます。OMF では Oracle Disk Manager のファイルサイズ変更機能が使われるため、表領域ファイルは最初にデフォルトサイズ 100 MB で作成されますが、必要に応じてサイズが拡張されます。拡張を制限するには MAXSIZE 属性を使います。

次の例に、OMF データベースを作成するコマンドと、それぞれの場所に EMP_TABLE と EMP_INDEX 表領域を作成するコマンドを示します。

```
SQL> create database PROD;
```

メモ: OMF が機能する上でディレクトリが存在する必要があるため、SQL*Plus HOST コマンドを使ってディレクトリを作成します。

データベースが作成されます。

```
SQL> HOST mkdir /PROD/EMP_TABLE;
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/PROD/EMP_TABLE';
```

システムが変更されました。

```
SQL> create tablespace EMP_TABLE DATAFILE AUTOEXTEND ON MAXSIZE ¥
500M;
```

表領域が作成されました。

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/PROD/EMP_INDEX';
```

システムが変更されました。

```
SQL> create tablespace EMP_INDEX DATAFILE AUTOEXTEND ON MAXSIZE ¥
100M;
```

表領域が作成されました。

ls コマンドを使って、新しく作成されたデータベースを表示します。

```
$ ls -lFR
total 638062
drwxr-xr-x 2 oracle10g dba 96 May  3 15:43 EMP_INDEX/
drwxr-xr-x 2 oracle10g dba 96 May  3 15:43 EMP_TABLE/
-rw-r--r-- 1 oracle10g dba 104858112 May 3 17:28 ora_1_BEhYgc0m.log

-rw-r--r-- 1 oracle10g dba 104858112 May 3 17:27 ora_2_BEhYu4NA.log
-rw-r--r-- 1 oracle10g dba 806912 May 3 15:43 ora_BEahlfUX.ctl
-rw-r--r-- 1 oracle10g dba 10489856 May 3 15:43
ora_sys_undo_BEajPSVq.dbf
-rw-r--r-- 1 oracle10g dba 104861696 May 3 15:4
ora_system_BEaiFE8v.dbf
-rw-r--r-- 1 oracle10g dba 186 May 3 15:03 PROD.ora

./EMP_INDEX:
total 204808
-rw-r--r-- 1 oracle10g dba 104861696 May 3 15:43
ora_emp_inde_BEakGfun.dbf

./EMP_TABLE:
```

```
total 204808
-rw-r--r-- 1 oracle10g dba 104861696 May 3 15:43
ora_emp_tabl_BEak1LqK.dbf
```

Veritas Extension for Oracle Disk Manager の設定

Veritas Extension for Oracle Disk Manager は Storage Foundation Cluster File System High Availability の一部です。Veritas Extension for Oracle Disk Manager は、Storage Foundation Cluster File System High Availability と Oracle10g 以降がインストールされると有効になります。Veritas Extension for Oracle Disk Manager ライブラリは {ORACLE_HOME}/lib ディレクトリ内のライブラリにリンクされます。

クラスタファイルシステム上ではなく、ローカルで Oracle インストールを実行する場合、クラスタ内のすべてのノードで ODM リンクを実行する必要があります。

Veritas Extension for Oracle Disk Manager を設定するには、次の条件を満たす必要があります。

前提条件

- Veritas InfoScale Enterprise 製品がシステムにインストールされている必要があります。
  Storage Foundation Cluster File System High Availability システムにインストールされている必要があります。
- Oracle10g 以降がシステムにインストールされている必要があります。

使用に関する注意事項

- Oracle10g 以降の Storage Foundation Cluster File System High Availability がインストールされていない場合 Oracle はデフォルトのファイルアクセス方式を使用します。

Veritas Extension for Oracle Disk Manager の設定

ORACLE_HOME が共有ファイルシステムにある場合は、任意のノードから次のコマンドを実行します。その他の場合では各ノードでコマンドを実行します。

ORACLE_HOME は Oracle データベースのバイナリがインストールされる場所です。

Veritas Extension for Oracle Disk Manager を設定するには

1. oracle としてログインします。
2. Oracle データベースが動作している場合は、Oracle データベースをシャットダウンします。

   Oracle RAC の場合、すべてのインスタンスをシャットダウンします。
3. /opt/VRTSodm/lib64/libodm.so が存在することを確認します。

4 ORACLE_HOME にある Oracle の ODM ライブラリに Veritas Extension for Oracle Disk Manager ライブラリをリンクします。

https://www.veritas.com/content/support/en_US/article.100014296

5 Oracle データベースを起動します。

6 Oracle データベースを Veritas Extension for ODM から起動していることを調べるには、Oracle 警告ログに次のテキストが含まれていることを確認します。

```
Oracle instance running with ODM: Veritas 7.4.2.0.0 ODM Library,
Version 2.0
```

Oracle Disk Manager 用の既存のデータベースストレージの準備

VxFS ファイルシステム内のファイルは、Oracle Disk Manager と連携するために一切の変更を必要としません。ファイルは、デフォルトで Oracle Disk Manager I/O として検出され識別されます。Oracle Disk Manager データファイルを最大限に活用するには、ファイルを断片化しないでください。

Oracle Disk Manager を使用するには、Oracle10g 以上を実行している必要があります。

Oracle Disk Manager が設定されていることの検証

Oracle Disk Manager (ODM) が設定されていることを検証する前に、次の条件を満たしていることを確認します。

前提条件

- /opt/VRTSodm/lib64/libodm.so が存在する必要があります。
- $ORACLE_HOME/rdbms/lib/odm/libodm*<version>*.so が /opt/VRTSodm/lib64/libodm.so にリンクされています。
- VRTSdbed ライセンスが有効であることが必要です。
- VRTSodm RPM がインストールされている必要があります。

Oracle Disk Manager が設定されていることを検証するには

1 ODM 機能がライセンスに含まれていることを検証します。

```
# /opt/VRTS/bin/vxlicrep | grep ODM
```

出力は ODM が有効になっていることを検証します。

メモ: ODM 機能を含んでいるライセンスキーが期限切れになっていないことを検証します。ライセンスキーが期限切れになっている場合、ODM 機能は使えません。

---

2 VRTSodm RPM をインストールしていることを確認します。

```
# rpm -qa | grep VRTSodm
VRTSodm-7.4.2.0000-RHEL7.x86_64
```

3 libodm.so が存在することを確認します。

```
# ls -lL /opt/VRTSodm/lib64/libodm.so
-rwxr-xr-x 1 bin bin 49808 Sep 1 18:42
/opt/VRTSodm/lib64/libodm.so
```

Oracle Disk Manager が実行されていることを検証するには

1 Oracle データベースを起動します。

2 インスタンスが Oracle Disk Manager 機能を使っていることを確認します。

```
# cat /dev/odm/stats
# echo $?
0
```

3 Oracle Disk Manager が読み込まれていることを検証します。

```
# lsmod | grep odm
vxodm   164480  1
fdd 78976 1 vxodm
```

4 Oracle 警告ログで、Oracle インスタンスが実行されていることを検証します。ログに次のような出力が含まれている必要があります。

```
Oracle instance running with ODM: Veritas 7.4.2.0.0 ODM Library,
Version 2.0
```

Oracle Disk Manager 機能の無効化

Oracle Disk Manager 機能は通常のファイルを使うため、この機能を無効にすることで、すぐに通常の VxFS ファイルとしてファイルにアクセスできます。

メモ: Oracle Disk Manager 機能を無効にする前に、ファイルのバックアップを作成しておくと安全です。

Oracle インスタンス内の Oracle Disk Manager 機能を無効にするには

1. データベースインスタンスを停止します。
2. `rm` コマンドと `ln` コマンドを使って、Oracle Disk Manager ライブラリへのリンクを削除します。

   次の記事を参照してください。

   https://www.veritas.com/content/support/en_US/article.100014296
3. データベースインスタンスを再起動します。

Cached ODM の使用

ODM I/O は通常ファイルシステムキャッシュをバイパスし、ディスクから直接読み込み、ディスクに直接書き込みます。Cached ODM によって一部の I/O はキャッシュを使用して先読みを行うことができ、ODM I/O パフォーマンスを改善できます。Cached ODM は、Oracle からの I/O ごとのヒントを基に条件形式のキャッシュを実行します。ヒントは Oracle がデータに対して何を行うかを示します。ODM はこれらのヒントを使用してキャッシュを実行し、一部の読み取りで先読みを行いますが、ODM は同じファイルであってもその他の読み込みではキャッシュを使用しません。

Cached ODM はローカルマウントファイルとクラスタマウントファイルで有効にできます。

p.648 の「ファイルシステムの Cached ODM の有効化」を参照してください。

Cached ODM は 2 つの方法で設定できます。1 つ目の設定方法は、ファイルごとにすべての I/O のキャッシュをオンまたはオフにします。2 つ目の設定方法は ODM の cachemap を調整します。cachemap はファイルのタイプと I/O のタイプの組み合わせをキャッシュアドバイザリへとマップします。

p.648 の「個々のファイルの Cached ODM 設定の変更」を参照してください。

p.649 の「cachemap を使った Cached ODM 設定の追加」を参照してください。

ファイルシステムの Cached ODM の有効化

ファイルシステムに対する Cached ODM は最初は無効になっています。ファイルシステムのマウント後に、vxtunefs コマンドの odm_cache_enable オプションを設定して、ファイルシステムに対する Cached ODM を有効にします。

vxtunefs(1M)のマニュアルページを参照してください。

メモ: この vxtunefs コマンドを使うと、このファイルシステム上のすべての ODM ファイルのキャッシュが有効になります。

ファイルシステムの Cached ODM を有効にするには

1 VxFS ファイルシステム /database01 で Cached ODM を有効にします。

```
# vxtunefs -s -o odm_cache_enable=1 /database01
```

2 /etc/vx/tunefstab ファイルにファイルシステムエントリを追加すると、以後のマウントでもこの設定を有効にできます。

```
/dev/vx/dsk/datadg/database01 odm_cache_enable=1
```

tunefstab(4)マニュアルページを参照してください。

個々のファイルの Cached ODM 設定の変更

odmadm setcachefile コマンドを使用すると、特定のファイルの cachemap を無視して、ODM がファイルへの I/O をすべてキャッシュに保存する、またはいずれもキャッシュに保存しないようにできます。キャッシュ状態は、ON、OFF、DEF(デフォルト)のいずれかにできます。キャッシュ状態 DEF は条件キャッシュで、I/O ごとに、ODM は cachemap を調べ、指定されたファイルのタイプと I/O のタイプの組み合わせをキャッシュに保存する必要があるかどうかを判断します。キャッシュ状態 ON は、指定されたファイルを常にキャッシュに保存し、キャッシュ状態 OFF は指定されたファイルをキャッシュに保存しません。

odmadm(1M)のマニュアルページを参照してください。

メモ: ファイルシステムに対する Cached ODM を有効にしている場合にかぎり、キャッシュアドバイザリが機能します。odm_cache_enable フラグが 0 に設定されている場合は、特定のファイルのキャッシュアドバイザリを有効(ON)に設定しても、そのファイルシステムのすべてのファイルに対する Cached ODM が無効(OFF)になります。

特定のファイルで無条件キャッシュを有効にする方法

- /mnt1/file1 ファイルで無条件キャッシュを有効にします。

```
# odmadm setcachefile /mnt1/file1=on
```

このコマンドにより、ODM は file1 からのすべての読み取りをキャッシュに保存します。

特定のファイルでキャッシュを無効にするには

- /mnt1/file1 ファイルでキャッシュを無効にします。

```
# odmadm setcachefile /mnt1/file1=off
```

このコマンドにより、ODM は file1 からの読み取りをキャッシュに保存しません。

ファイルの現在のキャッシュアドバイザリの設定を確認するには

- /mnt1/file1 および /mnt2/file2 ファイルの現在のキャッシュアドバイザリ設定を確認します。

```
# odmadm getcachefile /mnt1/file1 /mnt2/file2
/mnt1/file1,ON
/mnt2/file2,OFF
```

すべてのファイルをデフォルトのキャッシュアドバイザリにリセットする方法

- すべてのファイルをデフォルトのキャッシュアドバイザリにリセットします。

```
# odmadm resetcachefiles /mnt1
```

cachemap を使った Cached ODM 設定の追加

odmadm setcachemap コマンドを使って、cachemap を設定できます。cachemap は、ファイルタイプと I/O タイプの組み合わせをキャッシュアドバイザリにマップします。ODM は、デフォルトの条件付きキャッシュ設定を持つすべてのファイルに対して cachemap を使います。これらのファイルは、odmadm setcachefile コマンドによってキャッシュ処理がオンまたはオフにされていないファイルです。

odmadm(1M)のマニュアルページを参照してください。

デフォルトでは、cachemap は空です。ただし、odmadm setcachemap コマンドを使ってキャッシュアドバイザリを追加できます。

キャッシュアドバイザリを cachemap に追加するには

- キャッシュアドバイザリを cachemap に追加するには、次のコマンドを実行します。

```
# odmadm setcachemap data/data_read_seq=cache,readahead
```

このコマンド例では、ODM は I/O にキャッシュ処理と readahead を使って、data_read_seq I/O タイプを持つオンラインログファイル（data）をオンラインにします。odmadm getcachemap コマンドの出力で、有効なファイルタイプと I/O タイプの値を確認できます。

odmadm（1M）のマニュアルページを参照してください。

マウント全体を通したキャッシュ設定の永続化

デフォルトでは、Cached ODM 設定はマウント全体をとおして永続的ではありません。/etc/vx/odmadm ファイルを作成し、そのファイル内にキャッシュアドバイザリ設定を列挙することで、設定を永続化できます。

マウント全体をとおしてキャッシュ設定を永続化にするには

- /etc/vx/odmadm ファイルを作成して、ファイルとそのキャッシュアドバイザリを列挙します。次の /etc/vx/odmadm ファイルの例では、/dev/vx/dsk/rootdg/vol1 デバイスを /mnt1 にマウントすると、odmadm は /mnt1/oradata/file1 のキャッシュ処理をオフにします。

```
setcachemap data/read_data_header=cache
setcachemap all/datapump=cache,readahead
device /dev/vx/dsk/rootdg/vol1
setcachefile oradata/file1=off
```

7

PITC (Point-In-Time Copy) の使用

25

PITC 方法の理解

この章では以下の項目について説明しています。

PITC（Point-In-Time Copy）の概要

Storage Foundation は、業務上の重要なデータを管理するための柔軟で効率的な手段です。Storage Foundation を使うと、頻繁に更新されるデータベースのある瞬間におけるオンラインイメージ、すなわちポイントインタイムコピーを取得できます。

トランザクション処理、意思決定、知的財産の作成などに継続的に（週 7 日 24 時間）利用しなければならないデータ量がますます増えることが予想されています。消失や破壊からデータを保護することもますます重要になっています。以前は、データのバックアップが発生している間、データが変更されないように、データを停止状態にしていました。ただし、このオプションはダウンタイム最小化のニーズを満たしません。

PITC（ポイントインタイムコピー）はデータのオンライン可用性を最大化させます。PITC を使うと、システムのバックアップやアップグレード、その他の保守作業を行うことができます。PITC は、アクティブデータと同じホストまたは異なるホスト上で処理できます。必要な場合は、PITC の処理を別のホストにオフロードし、実稼動サーバーでのシステムリソースの競合を防止できます。この方法はオフホスト処理と呼ばれます。正しく実装されれ

ば、オフホスト処理ソリューションが実稼動中のプライマリシステムの処理効率に影響を及ぼすことはほとんどありません。

PITC の特定の使用例について詳しくは、『Veritas InfoScale ソリューションガイド』を参照してください。

PITC を使う状況

Veritas InfoScale FlashSnap による PITC ソリューションを用いるのに適した状況の例を次に挙げます。

- データバックアップ - 多くの企業では、365 日 24 時間データが使用可能でなくてはなりません。企業は、クリティカルなデータをオフラインでバックアップするときのダウンタイムを受け入れる余裕がありません。データのスナップショットを作成し、そのスナップショットからバックアップすれば、ダウンタイムを最小に抑え、処理効率に影響を与えることなく業務上の重要なアプリケーションを実行し続けることができます。
- データ継続性の提供 - プライマリストレージに障害が発生した場合に、サービスが継続するように、アプリケーションデータをリカバリするために PITC ソリューションを使用できます。サーバーにエラーが発生している場合、SFCFSHA または SFHA の高可用性クラスタ機能と共に、PITC（point-in-time copy）ソリューションを使うことができます。
- 意思決定支援システムの分析およびレポート作成 - 意思決定支援システムの分析や業務レポート作成などには、必ずしもリアルタイムな情報が必要というわけではありません。このような処理に対しては、プライマリデータベースへのアクセス競合が発生しないように、スナップショットから作成した複製データベースを使うよう設定できます。必要な場合は、複製データベースとプライマリデータベースを短時間で再同期することができます。
- テストとトレーニング - 開発グループまたはサービスグループは、スナップショットを新しいアプリケーションのテストデータとして使えます。開発者、テスト担当者、品質管理グループなどは、スナップショットデータを実際的な基準として、新しいアプリケーションの堅牢性、統合性および処理効率をテストできます。
- データベースエラーのリカバリ - 管理者やアプリケーションプログラムによって引き起こされる論理エラーによって、データベースの統合性が損なわれることがあります。Storage Checkpoint やスナップショットコピーを使ってデータベースファイルをリストアすると、テープなどのバックアップメディアからすべて修復するよりも短時間でデータベースを復旧させることができます。

  Storage Checkpoint を使って、データベースのインスタンスを過去のある時点の状態にすばやくロールバックすることができます。
- データのクローン - ファイルシステムまたはアプリケーションデータのクローンを作成できます。この機能を使用して、仮想デスクトップのプロビジョニングを迅速かつ効率的に行うことができます。

ここまでで説明したスナップショットソリューションはすべて、Volume Replicator と組み合わせてディザスタリカバリのサイトでも利用可能です。

レプリケーションを使用するスナップショットについて詳しくは、『Veritas InfoScale 7.4.2 レプリケーション管理者ガイド』を参照してください。

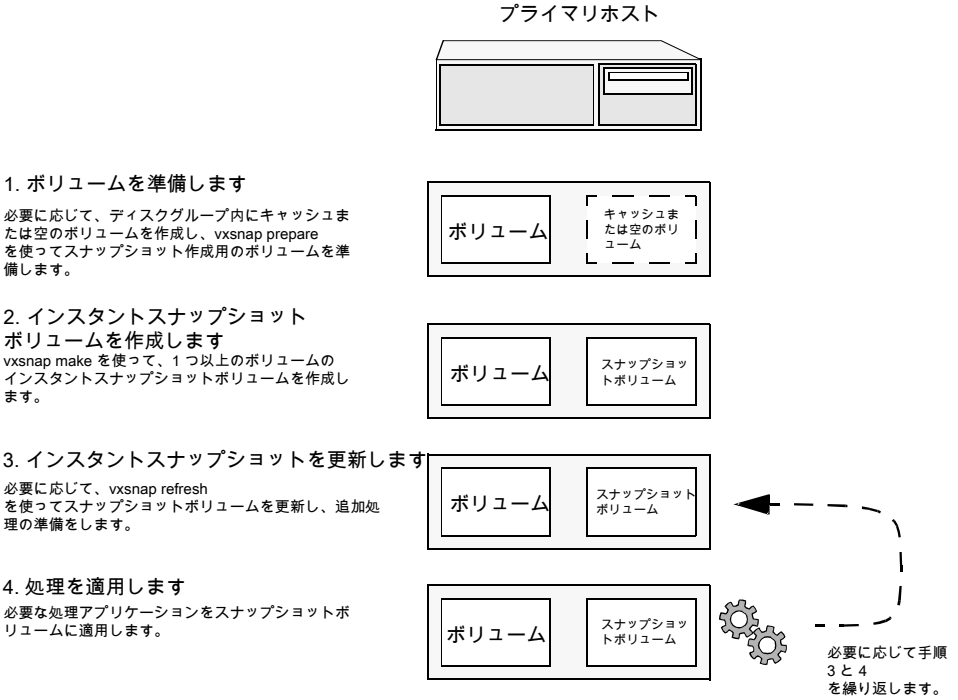
Storage Foundation は、次の使用例を含むいくつかの PITC ソリューションを、必要条件に応じて提供します。

- 意志決定支援システムの複製データベースの作成。
- スナップショットを使用したデータベースのバックアップとリカバリ。
- オフホストのクラスタファイルシステムのバックアップとリカバリ。
- オンラインデータベースのバックアップとリカバリ。

プライマリホストに対する PITC ソリューションの実装

図 25-1 は、プライマリホストでアプリケーションを設定するために必要な手順を示します。

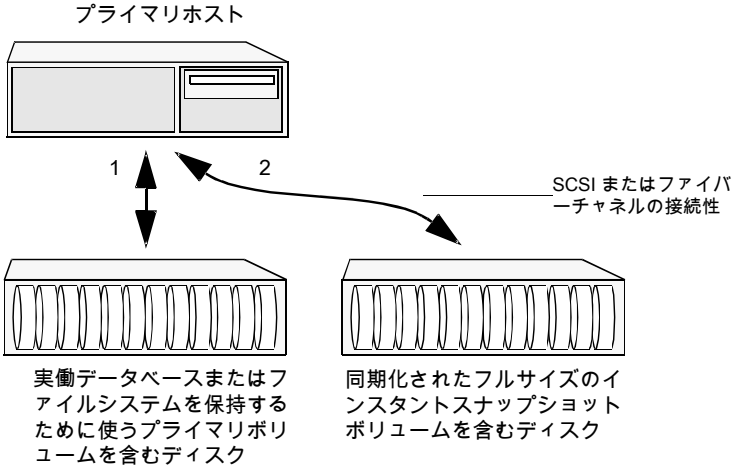
図 25-1 プライマリホストに対する PITC ソリューション実装のためのスナップショットおよび FastResync の使用



メモ: ディスクグループの分割および結合機能は使いません。処理はすべて同じディスクグループ内で実行されるため、ディスクの競合を回避する場合を除き、スナップショットの内容と元のボリュームの同期は必要ありません。スナップショットの作成と更新は、ほぼ即時に完了します。

図 25-2 は、プライマリホストにソリューションを導入したうえでディスクの競合を回避するための推奨事例です。

図 25-2 プライマリホストに対する PITC ソリューションの実装



この設定では、別々のコントローラからの個別のパス(図中の 1 および 2)をそれぞれプライマリボリューム用ディスクとスナップショットボリューム用ディスクに設定することをお勧めします。この設定によりディスクアクセスの競合は回避されますが、処理アプリケーション実行中はプライマリホストの CPU、メモリおよび I/O リソースにより大きな負荷がかかります。

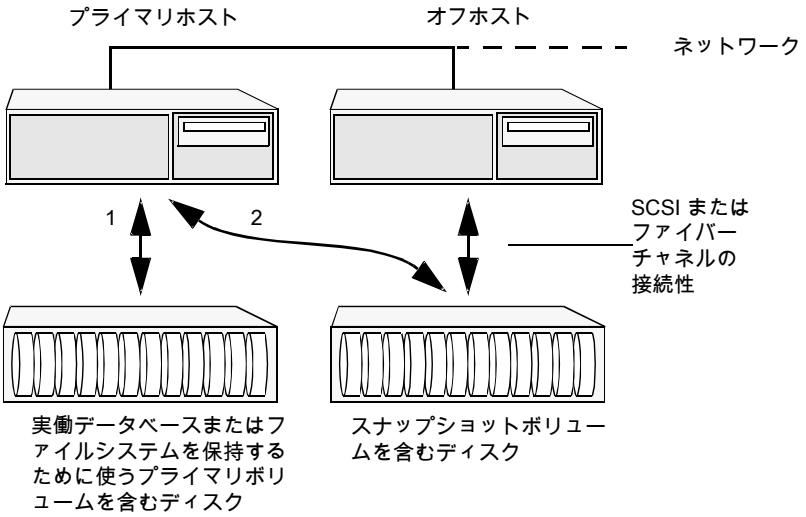
メモ: 領域最適化インスタントスナップショットまたは同期のとれていないフルサイズインスタントスナップショットの場合は、このように I/O 経路を分離することはできません。これらのスナップショットに格納されている内容は元のボリュームの変更部分だけであり、不変部分に存在するデータにアプリケーションがアクセスした場合、そのデータは元のボリュームから読み取られるためです。

オフホストに対する PITC ソリューションの実装

図 25-3 に示すように、スナップショットボリュームを運用系以外のオフホストで使うと、プライマリホストにおけるデータベース実行などの主要業務が、CPU と I/O を集中的に使

う処理（オンラインバックアップや意思決定支援システム等）によって処理効率を低下させられることはありません。

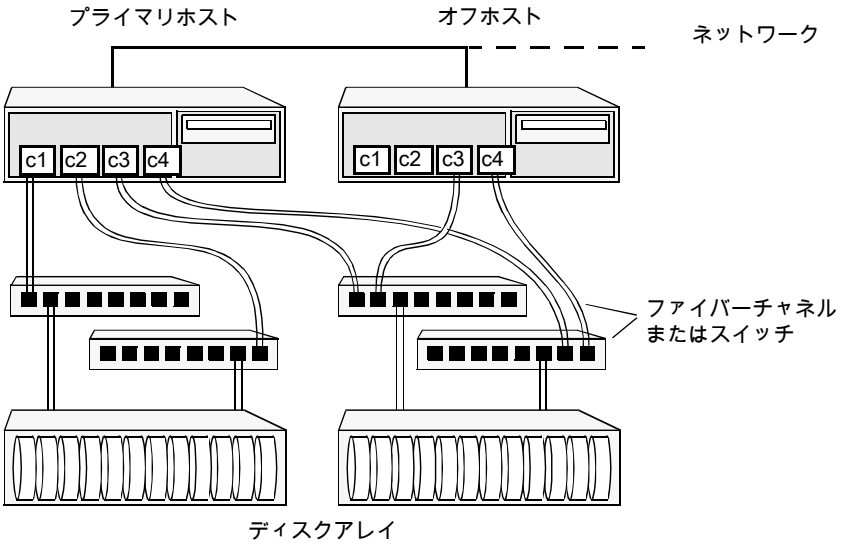
図 25-3　　オフホスト PITC ソリューションの実装例



また、プライマリボリュームのディスクとは別のホストコントローラに接続されているディスクにスナップショットボリュームを配置すると、プライマリホストでの I/O リソースの競合を回避できます。このソリューションを導入するには、図 25-3 に示すパス 1 およびパス 2 を個別のコントローラに接続する必要があります。

図 25-4 には、プライマリホストが 4 つのファイバーチャネルコントローラを使うテクノロジで接続を実現する例が示されています。

図 **25-4** 冗長ループアクセスを使ったオフホスト処理の接続例



この構成では冗長ループアクセスを使って、システムとディスクアレイ間のコンポーネントで発生する可能性のある障害に対処します。

メモ: オペレーティングシステムによっては、コントローラ名が次に示す例とは異なる場合があります。

図 25-5 は、クラスタノードの 1 つを代替ノードとして設定し、クラスタ内でオフホスト処理を実現する方法を示します。

図 25-5 クラスタノードを使用したオフホスト PITC ソリューションの実装例

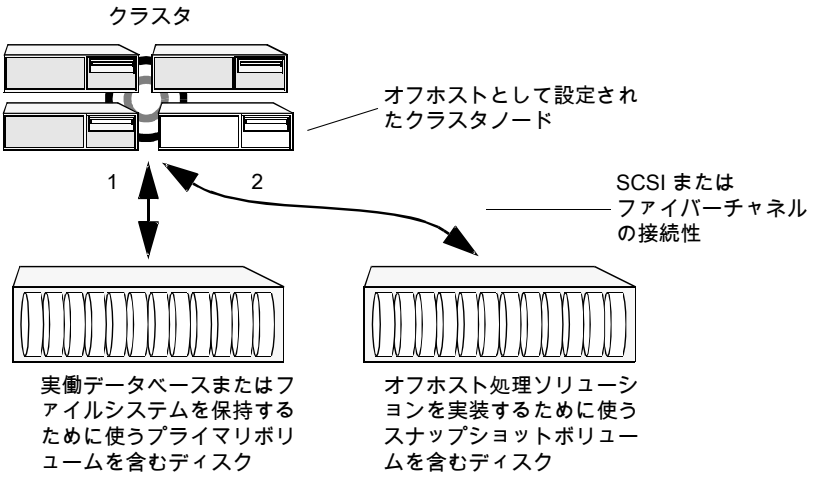
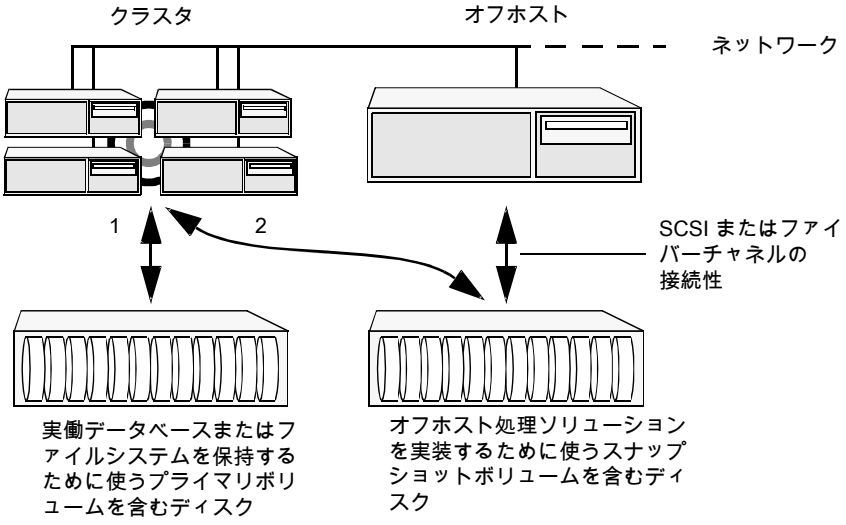


図 25-6 に代替設定を示します。代替ノードをクラスタにネットワーク接続されている別システム上に構築することもできます。ただしその場合、代替ノードはクラスタノードではなく、クラスタのプライベートネットワークには接続されません。

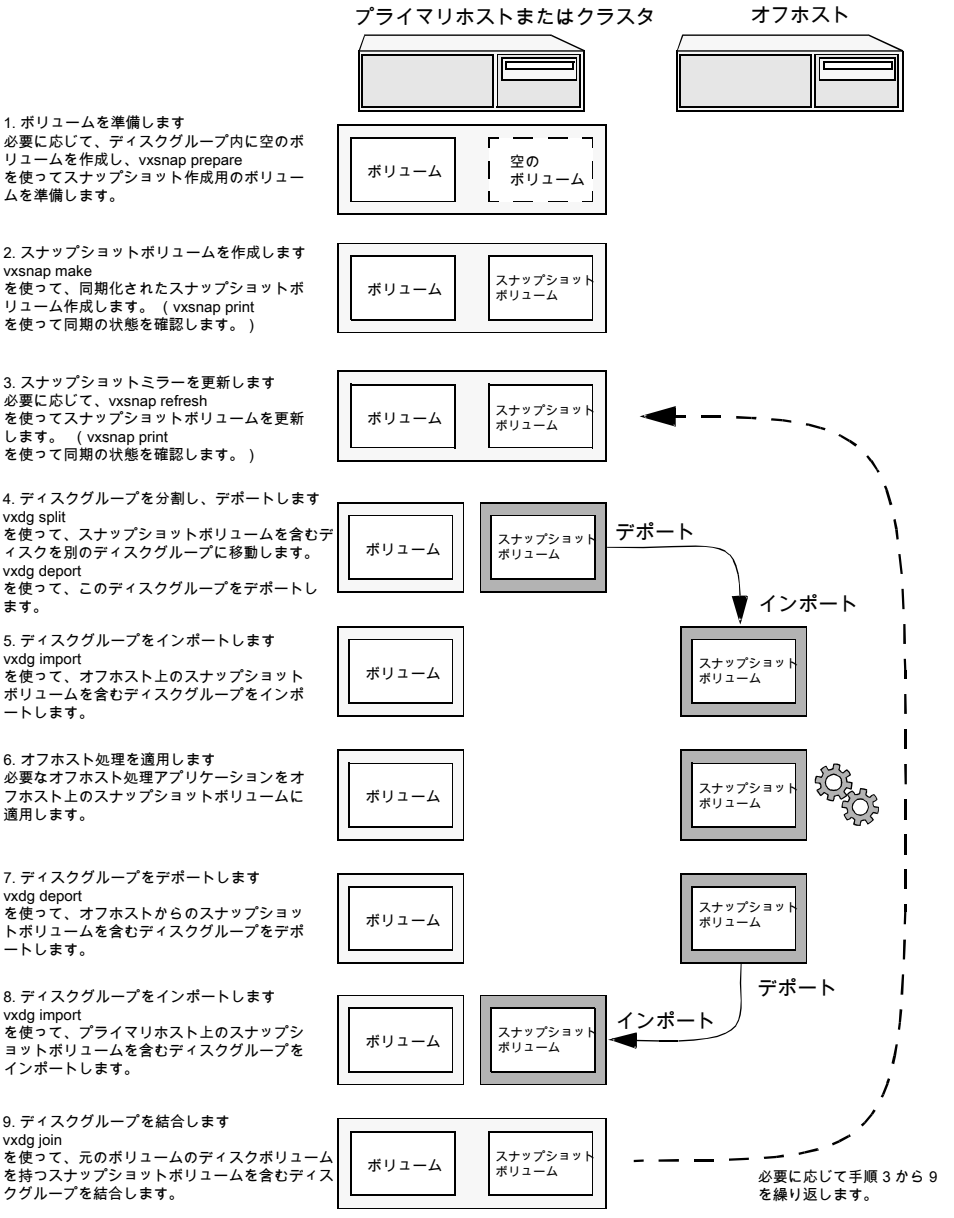
図 25-6 クラスタ外に配置したオフホストを使用したオフホスト PITC ソリューションの実装例



メモ: このマニュアルで取り上げるオフホスト処理の導入例では、クラスタ外に配置したオフホストがバックアップや意思決定支援システム専用に使われていることを前提にしています。クラスタについては、クラスタのメンバーではないオフホストのほうが設定しやすく効率的である場合があります。

図 25-7 は、プライマリホストでアプリケーションを設定するために必要な手順を示します。

図 25-7 オフホスト処理ソリューションの実装



スナップショットボリュームを分割して、オフホストにインポートされた別のディスクグループに結合するには、ディスクグループの分割および結合を使います。

メモ: スナップショットボリュームは別のディスクグループに移動してから別のホストにインポートするため、スナップショットボリュームの内容をあらかじめ親ボリュームと同期させておく必要があります。スナップショットボリュームを再インポートする場合、元のボリュームに基づくスナップショットボリュームの内容の更新は FastResync により高速化されます。

Storage Foundation PITC テクノロジについて

このトピックでは、Veritas FlashSnap™ 技術を活用した PITC（point-in-time copy）ソリューションの導入について説明します。Veritas FlashSnap テクノロジには Veritas InfoScale Enterprise ライセンスまたは Storage ライセンスが必要です。

Veritas InfoScale FlashSnap には、業務上の重要なデータを管理するためのフレキシブルで効率的な機能が用意されています。Veritas FlashSnap を使うと、変動するデータのオンラインイメージをある一時点で捉えて記録することができます。このイメージを「PITC」といいます。PITC を使うと、重要なデータの可用性を維持したまま、システムのバックアップ、アップグレードおよび他の保守作業を実行できます。必要な場合は、PITC の処理を別のホストにオフロードし、実稼動サーバーでのシステムリソースの競合を防止できます。

FlashSnap は、次の各種 PITC ソリューションをサポートしています。

- ボリュームレベルのソリューション。複数のタイプのボリュームレベルスナップショットがあります。これらの機能は、スナップショットを作成するときに別々のストレージが好ましいソリューションに適しています。たとえば、低層のストレージがあります。これらの技術のいくつかには、例外的なオフホスト機能が提供されています。
- ファイルシステムレベルのソリューションには、Veritas File System の Storage Checkpoint 機能が使われます。Storage Checkpoint は、次のようなストレージへのソリューションとして適しています。
  - ファイル数は少ないが、そのほとんどのファイルサイズが大きいという構成のファイルシステム
  - アプリケーションによるデータブロックの変更割合が少ないファイルシステム（たとえば、Web サーバーのコンテンツやデータベースなど）
  - アプリケーションのテストまたはバージョン管理のために書き込み可能なコピーを複数必要とするファイルシステム

  p.667 の「Storage Checkpoint」を参照してください。
- ファイルレベルのスナップショット。
  FileSnap 機能は、個々のファイルレベルでスナップショットを提供します。

PITC ソリューションの比較

次の表に、Storage Foundation の PITC ソリューションを並べて比較したものを示します。

表 25-1 PITC ソリューションの比較

ソリューション	詳細度	スナップされたデータの場所	スナップショット技術	内部の内容	エクスポートされた内容	オフホストで移動可能	可用性
フルサイズインスタントスナップショット	Volume	別個のボリューム	コピーオンライト / 完全な複製	変更された領域/完全ボリューム	読み取り/書き込みボリューム	同期後可	即時
インスタント領域最適化スナップショット	Volume	キャッシュオブジェクト(個別のキャッシュボリューム)	コピーオンライト	変更された領域	読み取り/書き込みボリューム	いいえ	即座
リンクされたプレックスブレークオフ	Volume	別個のボリューム	コピーオンライト / 完全な複製	変更された領域/完全ボリューム	読み取り/書き込みボリューム	同期後可	即時
vxsnap を使用したプレックスのブレークオフ	Volume	別個のボリューム	コピーオンライト / 完全な複製	変更された領域/完全ボリューム	読み取り/書き込みボリューム	同期後可	即時
vxassist を使用した従来のプレックスのブレークオフ	Volume	別個のボリューム	完全コピー	完全ボリューム	読み取り/書き込みボリューム	はい(同期後)	完全な同期後
Storage Checkpoint	File system	ファイルシステム内の領域	コピーオンライト	変更されたファイルシステムブロック	読み取り/書き込みファイルシステム	いいえ	即座
ファイルシステムのスナップショット	File system	独立したボリューム	コピーオンライト	変更されたファイルシステムブロック	読み取り専用ファイルシステム	いいえ	即座
FileSnap	ファイル	ファイルシステム内の領域	コピーオンライト/遅延コピーオンライト	変更されたファイルシステムブロック	読み取り/書き込みファイルシステム	いいえ	即座

ボリュームレベルのスナップショット

ボリュームスナップショットは、特定の時点での VxVM（Veritas Volume Manager）ボリュームのイメージです。ボリュームセットのスナップショットを作成することもできます。

スナップショットを使うと、ユーザーの作業中断を最小に抑えて、ボリュームのバックアップコピーをオンラインで作成できます。作成したバックアップコピーを使って、ディスク障害、ソフトウェア障害または人為的なミスが原因で失われたデータを復元したり、レポートの生成、アプリケーションの開発またはテストを行うための複製ボリュームを作成することができます。

ボリュームスナップショットを使って、オフホストオフラインバックアップも実装できます。

物理的には、データセットの完全なコピー（すべてのビットのコピー）の場合や、スナップショットが作成されてから更新されたデータセット要素だけが含まれる場合があります。後者の場合は、最初に書き込むときに割り当てる（allocate-on-first-write）スナップショットと呼ばれることもあります。もとのデータセット内の要素がはじめて更新（上書き）されるときにのみ、データ要素の領域がスナップショットイメージに追加されるためです。Storage Foundation の最初に書き込むときに割り当てる（allocate-on-first-write）スナップショットは、領域最適化スナップショットと呼ばれます。

ボリュームスナップショットの永続 FastResync

永続 FastResync がボリュームに対して有効にされると、VxVM は FastResync マップを使って、ボリュームやスナップショット内で更新されたブロックの履歴を残します。

スナップショットボリュームがもとのボリュームに再接続されると、永続 FastResync はスナップショットデータをただちに更新し再利用可能な状態にします。永続 FastResync はディスク領域上に FastResync マップを割り当てるため、システムやクラスタがクラッシュしても消失しません。専用ディスクグループ内のボリュームに対して永続 FastResync が有効にされると、ホストの再ブート後も更新分の再同期が実行されます。

永続 FastResync は、ボリュームとそのスナップショットボリュームの関係を、他のディスクグループへの移動後も追跡することができます。ディスクグループの再結合後は、永続 FastResync によってスナップショットプレックスを短時間で再同期できます。

ボリュームスナップショットのデータ整合性

ボリュームスナップショットは、特定時点でボリューム内に存在しているデータを取得します。したがって、上位のファイルシステムや、ファイルシステム内のファイルを開いているアプリケーション（データベースなど）によってメモリ上にキャッシュされているデータは、VxVM では認識されません。スナップショットの一貫性はクラッシュがあっても常に保持されます。つまり、アプリケーションにリカバリを実行させることで、スナップショットを使用可能にできます。これはサーバーのクラッシュ後にアプリケーションのリカバリが行われる方法と似ています。マウントされた Veritas File System（VxFS）を含むボリュームに対して、ボリュームの usetype 属性 `fsgen` が設定されている場合、VxVM は VxFS と連携して、

キャッシュ内にあるデータをボリュームにフラッシュします。したがって、これらのスナップショットは常に VxFS に対して整合性があり、マウントするときに VxFS リカバリファイルが必要ありません。

データベースでは、適切な機構を追加的に使って、ボリュームスナップショットの作成時に表領域データの一貫性を確保する必要があります。最近のデータベースソフトウェアの多くは、ファイルシステムの I/O を一時的に停止する機能を備えています。この操作の実行方法は、このマニュアルに記載している例の中で説明しています。また、ファイルシステム内の通常のファイルは、さまざまなアプリケーションで開かれることがありますが、そのファイルデータの完全な整合性を確保するには、アプリケーションを停止し、ファイルシステムを一時的にマウント解除する以外に方法はありません。通常、整合性の確保が重要になるのは、スナップショットの作成時に使われていなかったファイルデータのみです。ただし、アプリケーションが調整されるすべての例では、スナップショットはクラッシュから回復可能です。

サードミラーブレークオフスナップショット

プレックスブレークオフスナップショットでは、スナップショットの作成に追加のミラーを使用します。1 つのプレックスボリュームにプレックスブレークオフスナップショットを作成できますが、通常はミラーボリュームのスナップショットを作成します。1 つのミラーボリュームには 1 つ以上のプレックス(ミラー)が存在し、各プレックスはデータのコピーです。スナップショット操作により、スナップショットボリュームになるプレックスの「ブレークオフ」、つまり切り離しが行われます。既存のプレックスを切り離したり、スナップショットミラーとして専用に機能する新しいプレックスを追加したりすることができます。通常は、元のボリュームに対して冗長性を維持します。元のボリュームが 2 つのプレックスを持つミラーボリュームの場合は、スナップショットのサードミラーを追加します。そのため、このタイプのスナップショップはサードミラースナップショットとも呼ばれます。

スナップショットプレックスは、同じディスクグループ内にあるボリュームの既存のプレックスとは異なるディスク上に存在する必要があります。ディスクには、既存のボリュームのコンテンツを含むのに十分なディスク領域がなければなりません。ボリュームが 1 TB の場合は、さらに 1 TB のディスク領域が必要になります。

スナップショットを作成すると、プレックスは 2 つのボリュームに分けられます。元のボリュームでは、元のプレックスが維持されます。スナップショットボリュームには、スナップショットプレックスが含まれます。元のボリュームでは、引き続き I/O を受け入れます。スナップショットボリュームでは、そのボリュームで処理を実行することを選択するまで、スナップショットが作成された時点でのデータを保持します。

スナップショットは複数作成できます。つまり、元のデータのコピーを複数持つことが可能です。

サードミラーブレークオフスナップショットは、領域最適化インスタントスナップショットやフルサイズインスタントスナップショットのコピーオンライト機構を使うと処理効率が低下する、書き込みを集中的に行うボリューム(データベース REDO ログなど)に適しています。

領域最適化インスタントスナップショット

領域最適化スナップショットには、元のデータオブジェクトが表現する完全な物理イメージは含まれないので、オフホスト代替処理用として選択することはできません。領域最適化インスタントスナップショットは、元のボリュームの変更部分のみをストレージキャッシュに記録します。 元のボリュームに対する書き込みが発生すると、VxVM は書き込みをコミットする前にそのデータをキャッシュに保存します。ストレージキャッシュに必要なストレージのサイズは通常、元のボリュームよりもはるかに小さくて済むため、このスナップショットを領域最適化されていると呼びます。 領域最適化スナップショットでは、スナップショットの存在期間中に元のボリューム上で更新されたデータ量に比例して、ストレージと I/O 帯域幅が消費されます。

領域最適化インスタントスナップショットの利点としては、即時使用や高速更新が可能であること、設定と管理が容易であることなどが挙げられます。 完全コピースナップショットに比べて消費するストレージと I/O 帯域幅が少ないため、領域最適化スナップショットはより頻繁に作成できます。つまり、データ破損からのリカバリには適しています。

領域最適化スナップショットは、時間の経過とともに元のオブジェクトで変更されるデータが増えるため、自然に大きくなる傾向があります。したがって本質的に、有効期間が短い場合に適しています。

領域最適化スナップショットは、オフホスト代替処理用として選択することはできません。

領域最適化インスタントスナップショットの動作

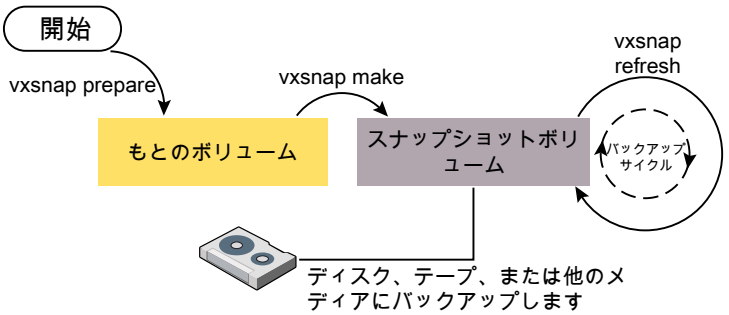
領域最適化スナップショットでは、コピーオンライト機構を使って、最初の作成時またはデータの更新時にスナップショットをすぐに使用可能にします。

1 つのディスクグループに 1 つのストレージキャッシュを設定し、そのディスクグループ内のすべてのボリュームで共有できるようにすることが可能です。この場合、宣言するキャッシュの名前は、各ボリュームの領域最適化スナップショットで同一にする必要があります。キャッシュはディスク上に格納され、永続的です。

キャッシュがいっぱいに近づいた場合は、ディスクグループ内の任意の利用可能な空き領域を使ってキャッシュが自動的に拡大されるように VxVM を設定できます。

図 25-8 に領域最適化インスタントスナップショットのモデルを示します。

図 25-8　領域最適化インスタントスナップショットの作成とバックアップサイクルでの使用法



p.700 の「領域最適化インスタントスナップショットの作成と管理」を参照してください。

スナップショットの再同期についてのオプション

共有ディスクグループのもとのボリュームにスナップショットボリュームを再結合する場合、ボリューム内のデータの再同期について 2 つの選択肢があります。

- もとのボリュームに基づいてスナップショットを再同期する - スナップショットの作成後に変更されたプライマリボリュームのデータを使ってスナップショットを更新します。再同期後は、再びバックアップや意思決定支援システムのためのスナップショットを作成できる状態になります。このタイプの再同期はスナップショットの更新ともいいます。
- スナップショットに基づいてボリュームを再同期する - スナップショットの作成後に変更されたスナップショットボリュームのデータを使ってもとのボリュームを更新します。この方法は、壊れたデータベースやファイルシステムのリストア、実稼動ソフトウェアをアップグレードする場合などに有効です。バックアップメディアからの完全リストアなどの他の方法よりは一般的に短時間で済みます。このタイプの再同期は、コピー（複製）からのスナップショットのリストアともいいます。

ディスクグループの分割および結合

1 つ以上のボリューム、スナップショットボリュームなどを別個のディスクグループに分割し、デポートすることを可能にする機能です。この処理を実行すると、オフホスト処理専用の別のホストにインポートすることができます。このホストはクラスタのメンバーである必要はありませんが、ボリュームが設定されているディスクにアクセスできる必要があります。このディスクグループは、後でデポートして再インポートすることにより、もとのディスクグループや別のディスクグループに結合することができます。

メモ: 領域最適化インスタントスナップショットにはもとのボリュームの変更部分に関する情報のみが記録されるため、別のディスクグループに移動することはできません。したがって、このマニュアルで説明するオフホスト処理アプリケーションには適していません。

フルサイズインスタントスナップショットを別のディスクグループに移動し、ホストからデポートする場合は、スナップショットの内容をもとのボリュームの不変部分と完全に同期させておく必要があります。

Storage Checkpoint

Storage Checkpoint は、特定時点におけるファイルシステムの永続イメージです。Storage Checkpoint は、コピーオンライト技術を使って、前回の Storage Checkpoint の作成後に変更されたファイルシステムブロックのみを特定し記録することにより、I/O オーバーヘッドを抑えます。Storage Checkpoint の重要な機能は次のとおりです。

- Storage Checkpoint は、システムに再ブートやクラッシュが発生しても、消失することはありません。
- Storage Checkpoint が作成されたときに対象ファイルシステム上にユーザーデータが存在している場合、Storage Checkpoint はファイルシステムのメタデータおよびディレクトリ階層に加えユーザーデータも保存します。
- マウントされたファイルシステムの Storage Checkpoint 作成後も、Storage Checkpoint のイメージに影響を与えずに、ファイルシステムのファイルを作成、削除および更新できます。
- ファイルシステムスナップショットとは異なり、Storage Checkpoint は書き込み可能としてマウントすることができます。
- ディスク領域の使用量を最小限に抑えるため、Storage Checkpoint はファイルシステム内の空き領域を使います。

各種データベース向けの Storage Foundation で提供されている Storage Checkpoint および Storage Rollback 機能は、データベースの破壊、ファイルの消失、表領域の削除などの論理エラーからデータベースを高速にリカバリします。データベースの連続的な Storage Checkpoint をマウントしてエラーを検出し、問題が発生する前の Storage Checkpoint にデータベースをロールバックすることができます。

Storage Checkpoint とスナップショットの違い

Storage Checkpoint は、次の点で Veritas File System スナップショットと異なります。

- Storage Checkpoint 自体に書き込み操作を実行することができる。
- システムの再ブートやシステム障害後も永続的に存在する。
- ファイルシステムと同じ空き領域プールを共有する。

- 最新の Storage Checkpoint の作成後に変更したファイルブロックのみを認識して、他の Storage Checkpoint との関連付けを管理する。
- 最新の Storage Checkpoint のみがプライマリファイルシステムから更新を蓄積するため、複数の読み取り専用 Storage Checkpoint を使うことで、I/O 操作と必要な格納領域を抑える。
- Storage Checkpoint 作成時点の状態にファイルシステムを復元できる。

Storage Checkpoint は、各種のバックアップおよびレプリケーションソリューションにおいて有効に活用できます。Storage Checkpoint には最後の Storage Checkpoint の後に変更されたファイルシステムブロックを追跡するための機能があるため、変更されたデータのみを取得すれば済むようなバックアップおよびレプリケーションアプリケーションの適用を容易にしています。Storage Checkpoint は、バックアップおよびレプリケーションソリューションの使用頻度を増やすことにより、データの移動を最小限に抑え、より高度な可用性およびデータの一貫性を実現します。

Storage Checkpoint は、多くのファイルを処理する環境（数百万のファイルを扱うファイルサーバーなど）でパフォーマンスにほとんど影響を与えることなく利用できます。ファイルシステムは Storage Checkpoint の作成時にフリーズされたままではないため、Storage Checkpoint が使われている場合でも、アプリケーションはファイルシステムにアクセスできます。ただし、Storage Checkpoint の作成は、ファイルシステム内のファイル数によっては時間がかかる場合があります。

Storage Checkpoint の動作

Storage Checkpoint 機能は、マウントされたファイルシステム（プライマリファイルセット）をフリーズし、Storage Checkpoint を初期化してから、ファイルシステムをアンフリーズします。この場合、まずファイルシステムを静的な状態にしてから、すべてのデータをディスクに書き込みます。フリーズ処理により、ファイルシステムへの I/O 操作がすべて遮断されます。Storage Checkpoint は実際のデータを含まずに作成されます。すなわち、Storage Checkpoint は、データではなく、プライマリファイルセットのブロックマップを示します。次に実行されるアンフリーズプロセスで、ファイルシステムへの I/O 操作が再開されます。

単一のファイルシステムまたは複数のファイルシステムに対して、Storage Checkpoint を作成できます。複数のファイルシステムの Storage Checkpoint の場合は、ファイルシステムを同時にフリーズし、ファイルシステムすべてに対して Storage Checkpoint を作成した後、ファイルシステムをアンフリーズします。その結果、複数のファイルシステムの Storage Checkpoint は同じ作成タイムスタンプを持ちます。Storage Checkpoint 機能は、操作の進行中にシステムクラッシュが発生しない限り、複数のファイルシステムの Storage Checkpoint は、指定したすべてのファイルシステムに作成されるか、または、どのファイルシステムにも作成されないか、のどちらかであることを保証します。

メモ: システムクラッシュが発生すると、アプリケーションの呼び出し時に Storage Checkpoint をクリーンアップします。

プライマリファイルセットの Storage Checkpoint には、最初はプライマリファイルセット内の既存のデータブロックへのポインタだけが含まれ、独自のデータブロックは割り当てられていません。

図 25-9 は、ファイルシステム `/database` とその Storage Checkpoint を示しています。Storage Checkpoint は、作成時にはプライマリファイルセットと論理的に同一ですが、実際のデータブロックは含まれていません。

図 25-9 プライマリファイルセットと Storage Checkpoint

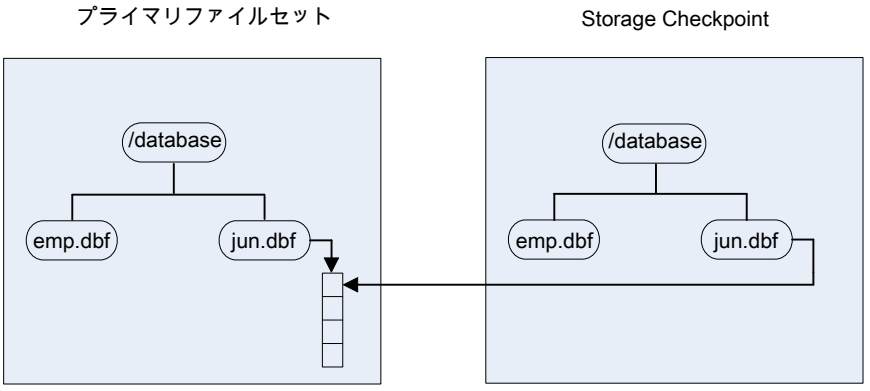
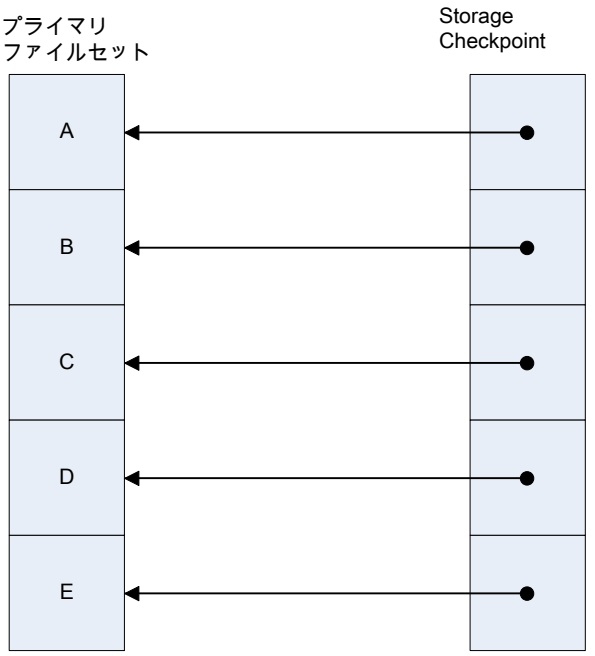


図 25-10で、四角はファイルシステムの各ブロックを表しています。この図の Storage Checkpoint には、図 25-9 で示されているように、Storage Checkpoint が作成された時点のプライマリファイルセットへのポインタが含まれています。

図 25-10 Storage Checkpoint の初期化



Storage Checkpoint は、プライマリファイルセットからデータを検索することで、ファイルシステムと完全に一致するイメージを提供します。Storage Checkpoint は、VxFS のコピーオンライト技術を使って更新されます。

p.670 の「コピーオンライト」を参照してください。

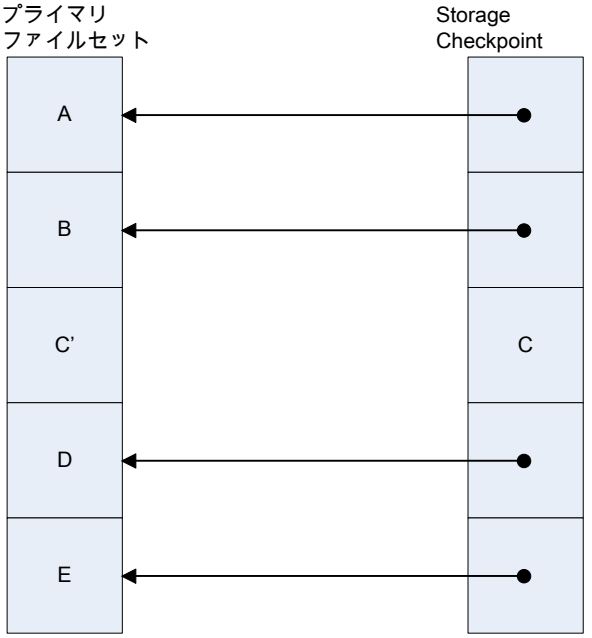
コピーオンライト

図 25-11 のプライマリファイルセットでは、最初は C が含まれていた 3 番目のデータブロックが更新されています。

新しいデータでデータブロックが更新される前に、もとのデータが Storage Checkpoint にコピーされます。この技術は、コピーオンライトと呼ばれるものです。これにより、Storage Checkpoint では Storage Checkpoint 作成時のプライマリファイルセットのイメージが維持されます。

更新や書き込み操作が行われるたびに、Storage Checkpoint にデータをコピーする処理が常に行われるわけではありません。古いデータを保存するのは 1 回だけでいいためです。プライマリファイルセットのブロックの更新が続くと、Storage Checkpoint にもとのデータブロックが蓄積されます。この例では、現在 C を含んでいる 3 番目のデータブロックがその後更新されても、Storage Checkpoint へのコピーは実行されません。これは、C を含むブロックのもとのイメージがすでに保存されているためです。

図 25-11 プライマリファイルセットの更新



Storage Checkpoint の可視性

ckptautomnt マウントオプションによって、すべての Storage Checkpoint は、ディレクトリのリストに表示されない .checkpoint という特殊な名前を持つファイルシステムのルートディレクトリ内のディレクトリを介して、自動的にアクセス可能になります。このディレクトリの中には、ファイルシステムの各 Storage Checkpoint のディレクトリがあります。これらの各ディレクトリは、対応する Storage Checkpoint のマウントとして動作します。ただし、次の例外は除きます。

- NFS など外部アプリケーションは、ファイルをもとのマウントポイントの一部として参照します。したがって、追加の NFS エクスポートは不要です。
- アプリケーションに開示される i ノード番号は、マウントオプションに応じて一意にできます。

Storage Checkpoint は内部で自動マウントされますが、その自動マウントはオペレーティングシステムでは認識されません。これは、Storage Checkpoint を手動でマウントできず、マウントされたファイルシステムのリストに表示されないことを意味します。Storage Checkpoint が作成または削除されると、Storage Checkpoint ディレクトリのエントリは自動的に更新されます。Storage Checkpoint のファイルがまだ使用中の場合でも、Storage Checkpoint が -f オプションで削除されると、Storage Checkpoint は強制的にマウント解除され、ファイルでの操作はすべて EIO エラーで失敗します。

VxFS（Veritas File System）の古いバージョンで作成されたディレクトリが存在する場合や Storage Checkpoint への可視機能が削除された場合など、ファイルシステムのルートディレクトリに .checkpoint という名前のファイルやディレクトリがすでに存在する場合、Storage Checkpoint へのアクセスを提供する擬似ディレクトリにはアクセスできません。この機能が有効の場合、ルートディレクトリに .checkpoint という名前のファイルまたはディレクトリを作成しようとすると、EEXIST エラーで失敗します。

メモ: 自動マウントされた Storage Checkpoint が NFS マウントによって使われている場合、Storage Checkpoint の削除は、強制（-f）オプションを指定しなくても成功することがあります。

Storage Checkpoint と 64 ビットの i ノード番号

ファイルの i ノード番号は、Storage Checkpoint 全体で同じです。たとえば、ファイル file1 がファイルシステムにあり、Storage Checkpoint がそのファイルシステムを取る場合、元のファイルシステムと Storage Checkpoint の file1 で stat コマンドを実行すると、st_ino で同じ値が返されます。st_ino と st_dev の組み合わせは、システム内のすべてのファイルを一意に識別する必要があります。これは、Storage Checkpoint は別々にマウントされ、st_dev が異なるため、通常は問題ありません。Storage Checkpoint のファイルに Storage Checkpoint の可視性拡張子を介してアクセスする場合、st_dev は元のファイルシステムと同様に、すべての Storage Checkpoint で同一です。つまり、st_ino と st_dev を使用してもファイルを一意に識別できなくなったことを意味します。

通常は、システムのすべてのファイルを一意に識別する必要はありません。ただし、正しく機能するためには一意に識別する必要がなるアプリケーションもあります。たとえば、あるバックアップアプリケーションは、ファイルが別のファイルにハードリンクされているかどうか確認するために、両方のファイルで stat を呼び出し、st_ino と st_dev が同一であるかどうか調べる場合があります。Storage Checkpoint の可視性拡張子を介して 2 つのクローンを同時にバックアップするようにバックアップアプリケーションに指示があった場合、それらのファイルに含まれているデータが異なる場合でも、アプリケーションは誤って 2 つのファイルが同一であると推測します。

デフォルトでは、SF（Storage Foundation）は i ノード番号を一意にしません。ただし、一意の 64 ビットの i ノード番号の使用を有効にするために uniqueino マウントオプションを指定できます。このオプションは再マウント中には変更できません。

Storage Checkpoint の種類

次の種類の Storage Checkpoint を作成することができます

- 「Data Storage Checkpoint」
- 「Nodata Storage Checkpoint」
- 「Removable Storage Checkpoint」

- 「Non-mountable Storage Checkpoint」

Data Storage Checkpoint

Data Storage Checkpoint は、Storage Checkpoint の作成時のファイルシステムの完全なイメージです。この Storage Checkpoint には、ファイルシステムのメタデータおよびファイルデータブロックが含まれます。ファイルシステムの場合と同様に、Data Storage Checkpoint では、マウント、アクセスおよび書き込みを実行できます。Data Storage Checkpoint は、アクティブなファイルシステムの永続的で安定したイメージが必要なバックアップアプリケーションに有効です。Data Storage Checkpoint により、書き込み操作を実行するシステムやアプリケーションにオーバーヘッドが発生します。Data Storage Checkpoint の有効期間を制限することにより、システムリソースへの影響を最小限に抑えることができます。

p.743 の「data Storage Checkpoint と nodata Storage Checkpoint の相違点の表示」を参照してください。

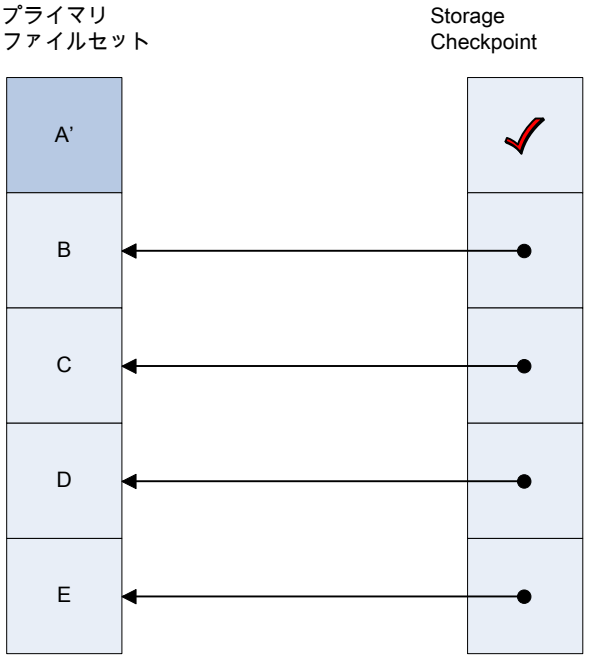
Nodata Storage Checkpoint

Nodata Storage Checkpoint はファイルシステムのメタデータのみを含み、ファイルデータブロックは含みません。もとのファイルシステムが変更されると、Nodata Storage Checkpoint は変更されたすべてのブロックの位置を記録します。 Nodata Storage Checkpoint では、データ自体のコピーを実行しないことで、システムリソースの使用とファイルシステムのパフォーマンスへの影響を最小限に抑えています。

図 25-12 では、もとのデータの A を含む 1 番目のブロックが更新されています。

もとのデータは Storage Checkpoint にコピーされませんが、変更されたブロックには、Storage Checkpoint でマークが付けられます。マーカーは変更されたデータを示します。

図 25-12 Nodata クローンの更新



p.743 の「data Storage Checkpoint と nodata Storage Checkpoint の相違点の表示」を参照してください。

Removable Storage Checkpoint

Removable Storage Checkpoint は、ファイルシステムの容量が不足する特定の状況下で、自動的に削除されるようにすることができます。

p.751 の「Storage Checkpoint の領域管理に関する注意事項」を参照してください。

`create` や `mkdir` などのユーザー操作中に、ファイルシステムの領域が不足すると、Storage Checkpoint がマウントされていても Removable Storage Checkpoint は削除されます。これにより、ディスクスペースの領域不足による中断なしに、アプリケーションを続行できます。Non-removable Storage Checkpoint は `ENOSPC` のような状況下では自動的に削除されません。Removable Storage Checkpoint のみを作成することをお勧めします。ただし、`fsadm` などの特定の管理操作中は、ファイルシステムの領域が不足しても Removable Storage Checkpoint は削除されません。

Storage Checkpoints はデフォルトで non-removable として作成されます。デフォルトの動作は、`vxtunefs -D ckpt_removable=1` コマンドを使用することで、VxFS が Removable Storage Checkpoint を作成するように変更できます。デフォルトで Removable Storage Checkpoint を作成するように設定した場合、Non-removable

Storage Checkpoint は `fsckptadm -R create ckpt_name mount_point` コマンドを使用することで作成できます。

`vxtunefs`(1M) と `fsckptadm`(1M) のマニュアルページを参照してください。

Non-mountable Storage Checkpoint

`fsckptadm set nomount` コマンドを使って、マウントできない Storage Checkpoint を作成できます。`nomount` オプションは `fsckptadm clear nomount` コマンドを使って消去できます。

Non-mountable Storage Checkpoint をセキュリティ機能として使用できます。これにより、他のアプリケーションによる Storage Checkpoint へのアクセスや変更を防ぐことができます。

FileSnap について

FileSnap は、同じファイルシステムに格納された、同じ名前空間内のファイルの原子的に領域が最適化されたコピーです。VxFS（Veritas File System）はファイルシステムのディスクレイアウトバージョン 8 以降でスナップショットをサポートします。

FileSnap は、ファイルシステムまたはボリュームより細かい詳細度のオブジェクトのスナップショットを作成する機能を備えています。ファイルシステムの名前空間の一部のスナップショットを作成する機能では、ファイルシステムに格納されるデータのアプリケーションベースまたはユーザーベースの管理が必要になります。これは、ファイルシステムが一連のユーザーまたはアプリケーションによって共有されるか、あるいはデータが同じファイルシステム内でさまざまな重要度に分類されるときに便利です。

すべての通常のファイル操作は FileSnap でサポートされ、VxFS は FileSnap を区別することはありません。

FileSnap のプロパティ

FileSnap は、管理者権限の必要なしに、所有するデータのスナップショットを作成する機能を非 root ユーザーに提供します。これにより、アプリケーションサイクルの適切な時点にスナップショットをスケジュール化することで、ユーザーやアプリケーションがデータのバージョン管理、バックアップ、および復元を行えるようになります。FileSnap からの復元は、スナップショットをソースファイルとして指定し、元のファイルを `vxfilesnap` コマンドの引数としての書き込み先ファイルとして指定することで簡単に行えます。

FileSnap の作成では、ソースファイルが読み取り専用としてロックされ、操作の間書き込み先ファイルが排他的にロックされて、スナップショットが原子的に作成されます。ファイルシステム内の残りのファイルは、FileSnap の作成が進行中に I/O を一時停止せずにアクセスできます。スナップショットの作成が進行中に、ソースファイルへの読み取りアク

セスも中断されません。これにより、複数のユーザーやアプリケーションによる、強制的ではない方法によるファイルシステムの真の共有が可能になります。

ソースファイルと書き込み先ファイル間の名前空間の関係は、書き込み先のファイルパスを指定することでユーザーが実行する vxfilesnap コマンドによって定義されます。Veritas File System（VxFS）は、ソースファイルと書き込み先ファイルを区別することなく、これらの 2 つのファイル間の内部関係を維持することもありません。スナップショットが完了すると、ソースファイルと書き込み先ファイル間の唯一の共有プロパティは共有されるデータブロックとブロックマップになります。

ファイルの FileSnap の数は事実上無制限です。技術的な限度は、VxFS ファイルシステムによってサポートされるファイルの最大数、つまりファイルセットあたり 1,000,000,000 ファイルです。同じファイルから何千もの FileSnap が作成され、これらのスナップショットファイルのそれぞれが同時に何千ものスレッドによって読み取りおよび書き込みが行われた場合、FileSnap は上書きによって共有解除が発生した場合に共有ブロックの競合を防止する設計によって適切に拡大縮小されます。FileSnap の上書きによる共有ブロックの共有解除が発生した場合に見られるパフォーマンスは、従来のコピーオンライトのパフォーマンスより書き込みを割り当てる場合のパフォーマンスに近くなります。

ディスクレイアウトバージョン 8 以降では、ファイル間で共有するブロックまたはエクステントをサポートするために、参照カウントは各共有エクステントで追跡されます。VxFS は、遅延型のエクステントの共有および共有解除によって参照カウントの更新を処理します。また、いったん共有としてマーク付けされたエクステントは、すべての参照が消失するまで共有が解除されることはありません。これは FileSnap の作成パフォーマンスとデータエクステント共有解除のパフォーマンスを改善するためです。ただし、これにより、遅延再生の処理時点でのみ正確であるファイルシステムの共有ブロック統計が事実上生成されます。つまり、ファイルシステムおよびファイル上の共有エクステント統計は、ファイルシステムの状態によって無効である可能性があります。

FileSnap に対する同時 I/O

FileSnap の設計と実装により、同一のファイルの異なるスナップショットへの同時読み取りまたは書き込みが、あたかも独立したファイルであるかのように実行できるようになります。エクステントが同一のファイルのスナップショット間で共有されている場合でも、共有が同時 I/O に悪影響を及ぼすことはありません。

コピーオンライトと FileSnap

Veritas File System（VxFS）は、共有エクステントによって参照されるファイルの領域を上書きするときに【★遅延コピーオンライト★】を実行するオプションをサポートします。通常のコピーオンライトの実装には、古いデータの読み取り、新しいブロックの割り当て、古いデータの新しいブロックへの同期コピーまたは書き込み、および新しいデータの新しいブロックへの書き込みが含まれます。この結果、最悪の場合 1 つ以上のトランザクションの割り当てが行われ、続いて読み取り、同期書き込み、および上書きのために必要な I/O 動作に適合する別の書き込みが行われる可能性.があります。このシーケンスにより、通

常のコピーオンライト操作がコストのかかるものになります。VxFS の遅延コピーオンライトの実装の場合、新しく割り当てられたブロックに古いデータがコピーされず、したがって新しいデータがブロック全体をカバーする限り、古いデータを読み取る必要がありません。この動作と共有エクステントアカウント処理の遅延処理を組み合わせることにより、将来遅延コピーオンライトは書き込みの割り当て処理に匹敵する完全なものになります。ただし、サーバークラッシュの発生時に、サーバーが新しく割り当てられたブロックに新しいデータをフラッシュしなかった場合、上書きされた領域内のデータは、データをフラッシュする前にサーバーがクラッシュした際に書き込みを割り当てた場合のデータに似たものになる場合があります。これはデフォルトの動作ではありません。デフォルトの動作では、上書きされた領域内のデータは新しいデータまたは古いデータのいずれかになります。

FileSnap からの読み取り

通常の読み取り要求の場合、Veritas File System（VxFS）は、共有データブロックが任意の FileSnap またはソースファイルからアクセスできる場合であっても、特定の共有データブロックのページキャッシュ内のデータページの 1 つのコピーのみキャッシュに保存します。共有データページがキャッシュに保存されると、FileSnap またはソースファイルのいずれかによるその後の要求はページキャッシュから読み込まれます。これにより、ディスクへの重複読み取り要求が不要になり、アレイの I/O 負荷が軽減されます。また、ページキャッシュの重複が減少し、その結果何千もの FileSnap がアクセスされたときにキャッシュのチャーニングがほとんどなくなり、システムページキャッシュを効率的に使用できるようになります。

ブロックマップの断片化と FileSnap

ソースファイルのブロックマップはスナップショットファイルによって共有されます。以前に共有された領域でデータが上書きされると、書き込みが行われるファイルのブロックマップが変更されます。ソースファイルの共有データエクステントが同じ領域への上書き要求のサイズより大きい場合、書き込まれるファイルのブロックマップは一層断片化された状態になります。

バックアップと FileSnap

共有ブロックを含む VxFS ファイルシステムの完全バックアップでは、ソースファイルシステム内の物理ブロックへの論理参照の合計数と同じスペースがターゲットにも必要になる場合があります。たとえば、1000 個の FileSnap が作成された 20 GB ファイルを所有している場合、論理ブロック参照の合計数は約 20 TB になります。VxFS ファイルシステムは、ファイルとファイル内の 1000 個のスナップショットを格納するために 20 GB を少し超える物理ブロックのみ必要になる一方で、このファイルシステムは、バックアップターゲットが重複排除をサポートしていないと仮定した場合、ファイルシステムのバックアップを作成するために、バックアップターゲット上に 20 TB を超える領域が必要になります。

スナップショットファイルシステムについて

スナップショットファイルシステムは、バックアップを作成する機能が用意された VxFS ファイルシステム（スナップファイルシステム）と完全に一致するイメージ（スナップショット）を格納するファイルシステムです。スナップショットには、スナップショットが作成されたポイントインタイム特定時点でのファイルシステムと一致するイメージが表示されます。ファイルを選択して、スナップショットからのバックアップ（cpio、cp などの標準ユーティリティを使用）や、ファイルシステム全体のイメージのバックアップ（vxdump ユーティリティまたは fscat ユーティリティを使用）を実行できます。

mount コマンドを使ってスナップショットファイルシステムを作成します。mkfs コマンドは必要ありません。スナップショットファイルシステムは常に読み取り専用です。スナップショットファイルシステムは、スナップしたファイルシステムがマウントされている間のみ存在し、マウントが解除されると消失します。スナップされたファイルシステムは、そのスナップショットがすべてマウント解除されるまではマウント解除できません。1 つのファイルシステムの複数のスナップショットを作成および保持することはできますが、スナップショットのスナップショットは作成できません。

メモ: マウントを解除すると、スナップショットは消失します。再マウントすると、スナップファイルシステムには新規のスナップショットが使われます。スナップショットファイルシステムは、関係するスナップファイルシステムのマウントを解除する前に、マウントを解除する必要があります。fuser コマンドと mount コマンドのどちらを使っても、スナップショットが存在するためにスナップファイルシステムのマウントを解除できないことを判定することはできません。

クラスタファイルシステムでは、スナップショットをクラスタのいずれかのノードで作成し、そのノードからバックアップ操作を実行できます。クラスタファイルシステムのスナップショットは、それが作成されるノードでのみアクセスできます。すなわち、スナップショットファイルシステム自体はクラスタとしてマウントできません。

『Storage Foundation Cluster File System High Availability 管理者ガイド』を参照してください。

スナップショットファイルシステムの動作

スナップショットファイルシステムは、現在マウントされているファイルシステムのスナップショットとなる空のディスクスライスをマウントすることによってを作成されます。ビットマップ、ブロックマップ、スーパーブロックを初期化すると、現在マウントされているファイルシステムがフリーズします。スナップするファイルシステムがフリーズされると、スナップショットが有効になり、マウントされ、スナップファイルシステムがアンフリーズされます。スナップショットは、スナップショットを作成した時点のスナップファイルシステムと完全に一致するイメージとして表示されます。

p.624 の「ファイルシステムのフリーズとアンフリーズ」を参照してください。

スナップショットを作成した直後では、スナップショットファイルシステムはスナップファイルシステムのデータを検索し、要求プロセスに検出されたデータを戻して、読み取り要求に応答します。i ノードの更新または書き込み時において、スナップファイルシステムのデータブロック n のデータが変更された場合は、まず更新前データを読み取り、スナップショットにコピーした後、スナップファイルシステムのデータブロックを更新します。スナップショットファイルシステム上のデータブロック n に対応したビットマップエントリは、0 から 1 に変わります。これは、データブロック n のデータに対する更新がスナップファイルシステム上で検出されたことを示します。その後、データブロック n のブロックマップエントリが、0 から更新前データを含むスナップショットファイルシステムのブロック番号に変わり、更新前データを保持します。

スナップショットファイルシステムのデータブロック n に対するその後の読み取り要求に対しては、データブロック n のビットマップエントリを調べ、スナップファイルシステムのデータブロック n からではなく、スナップショットファイルシステムの指示されたデータブロックからデータを読み取ります。このテクノロジは、コピーオンライトと呼ばれます。更新前データは一度保存するだけでよいため、スナップファイルシステムのデータブロック n に対するその後の書き込みでは、スナップショットファイルシステムへのコピーは実行されません。

i ノード、ディレクトリ、ファイルのデータ、エクステントマップなどに関するスナップファイルシステムへのすべての更新がこの方法で処理されるため、スナップショットには、スナップショットが作成された時点のスナップファイルシステムのすべてのファイルシステム構造が整合性を保ちながら表示されます。スナップファイルシステムでデータブロックが変更されると、スナップショットにはスナップファイルシステムからコピーされたデータが順番に保存されます。

スナップショットに必要なディスク領域は、スナップファイルシステムの変更率とスナップショットの保守時間によって異なります。最悪のシナリオは、スナップファイルシステムが飽和し、すべてのファイルが削除された後、書き替えられた場合が考えられます。スナップショットファイルシステムには、スナップファイルシステムの各データブロックのコピーを保持するのに十分なデータブロックおよびスナップショットファイルシステムを構成するデータ構造用のデータブロックが必要です。これは、理論値ではスナップファイルシステムのサイズの約 101 % になります。通常、多くのファイルシステムではこれほど極端な率で変更が実行されることはありません。使用量が少ない間、スナップショットが必要とするデータブロックは、通常、スナップファイルシステムのデータブロックのわずか 2 から 6 % です。使用量が増えると、スナップファイルシステムのデータブロックの 15 % 程度を必要とします。この割合は大容量ファイルシステムほど低く、小容量ファイルシステムほど高い傾向があります。

警告: 変更後のデータブロックを保持する領域が不足すると、スナップショットファイルシステムは無効になり、アクセスできなくなります。これにより、スナップファイルシステムが影響を受けることはありません。

26

ボリュームスナップショットの管理

この章では以下の項目について説明しています。

ボリュームスナップショットについて

VxVM は特定時点のボリュームのイメージを作成できます。このイメージはボリュームスナップショットと呼ばれます。

p.663 の「ボリュームレベルのスナップショット」を参照してください。

ボリュームセットのスナップショットを作成することもできます。

VxVM でポイントインタイムコピーソリューションを実装するときは、vxsnap コマンドを使ってスナップショットを作成することをお勧めします。vxassist コマンドを使った従来のサードミラースナップショットのサポートは、今後のリリースで中止される予定です。

インスタントスナップショットコマンドの障害からのリカバリについては、『Veritas InfoScale トラブルシューティングガイド』を参照してください。

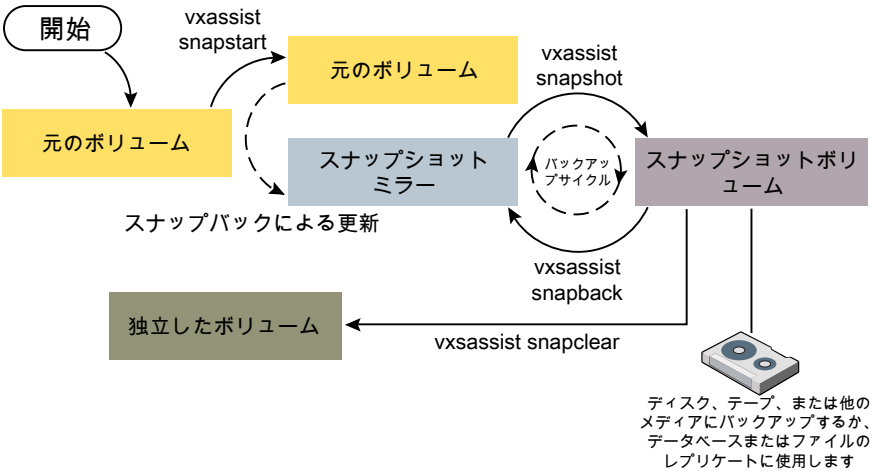
従来のサードミラーブレークオフスナップショット

ボリュームのバックアップをコマンドラインやスクリプトから実行する際の方法として、vxsnap コマンドを使うことをお勧めします。vxassist snapstart, snapwait、snapshot コマンドは、後方互換性のためにサポートされています。

インスタントスナップショットの作成用に準備したボリュームについては、vxassist コマンドを使った、従来の (サードミラーブレークオフ) スナップショットの管理はサポートされません。代わりに、vxsnap コマンドを使います。

「図 26-1」は、vxassist コマンドでサポートされる従来のサードミラーブレークオフボリュームスナップショットのモデルを示しています。

図 26-1　サードミラースナップショットの作成と使用法



vxassist snapstart コマンドでスナップショットに使うミラーを作成し、作成したミラーをスナップショットミラーとしてボリュームに接続します。ミラーを作成する場合のように、ボリュームの内容を新しいスナップショットプレックスにコピーするには時間がかかることがあります (vxassist snapabort コマンドはこの操作を取り消し、スナップショットミラーを削除します)。

接続が完了したら、vxassist snapshot コマンドを使って、1 つ以上のスナップショットミラーをそのデータプレックスとして使って、新しいスナップショットボリュームを作成します。スナップショットボリュームには、スナップショットを作成したときのもとのボリュームのデータのコピーが保存されます。複数のスナップショットミラーを使う場合は、スナップショットボリューム自体がミラー化されます。

vxassist snapback コマンドを使って、スナップショットプレックスを元のボリュームに戻すことができます。また、元のボリュームのデータからスナップショットミラー内のデータを再同期することもできます。これにより、スナップショットを使ってバックアップした後、スナッ

プショット内のデータを更新できます。同様のコマンドを使って、以前に作成したスナップショットからもとのボリュームの内容を復元できます。

FastResync 機能によって、スナップショットのデータを再同期するために必要な時間と I/O が最小限に抑えられます。FastResync が有効でない場合は、データ全体の再同期が必要です。

最後に、vxassist snapclear コマンドを使って、元のボリュームとスナップショットボリューム間の関連付けを解除できます。スナップショットの関係が壊れているため、変更の記録は実行されません。 新しい PIT を作成するためにスナップショットボリュームを再利用する必要がない場合は、このコマンドを使います。

従来のサードミラーブレークオフスナップショットの作成

VxVM では、vxassist やその他のコマンドを使って、ボリュームデバイスのサードミラーブレークオフスナップショットイメージが得られます。

ボリュームスナップショットの効率および有用性を高めるには、FastResync を有効にします。

永続 FastResync が必要な場合は、バージョン 0 の DCO をボリュームに関連付ける必要があります。

p.732 の「バージョン 0 の DCO および DCO ボリュームの追加」を参照してください。

ボリューム内容をすべて格納できる大きさのプレックスが必要です。または、領域最適化インスタントスナップショットを使うこともできます。

ボリュームのバックアップをコマンドラインやスクリプトから実行する際の方法として、vxsnap コマンドを使うことをお勧めします。vxassist snapstart、snapwait、snapshot コマンドは、下位互換性のためにサポートされています。

vxassist snapshot 手順は、次の 2 つの手順で構成されます。

- vxassist snapstart を実行してスナップショットミラーを作成します。
- vxassist snapshot を実行してスナップショットボリュームを作成します。

vxassist snapstart 処理により、ボリュームに接続し同期を行う、書き込み専用バックアッププレックスを作成します。

ボリュームがパスワードまたはパスフレーズで暗号化されている場合、パスワードまたはパスフレーズの入力を求めるメッセージが表示されます。

ボリュームと同期すると、バックアッププレックスはスナップショットミラーとして使う準備ができたことになります。この手順は、新しいスナップショットミラーが SNAPDONE 状態になるまでかかります。この状態遷移は、vxassist snapwait タスクで追跡することができます。このタスクは少なくとも 1 つのミラーが SNAPDONE 状態に変更されるまで待機します。 接続プロセスが失敗すると、スナップショットミラーは削除され、領域が解放されます。

メモ: スナップスタート手順が中断された場合、スナップショットミラーはボリューム起動時に自動的に削除されます。

スナップショットミラーはいったん同期されると、切断するまで更新され続けます。任意の時間に、既存ボリュームのイメージとしてスナップショットボリュームを作成することができます。また、スナップショットの実行に必要な間、ユーザーにシステムの使用を控えてもらうこともできます（通常は 1 分未満です）。スナップショットミラーの作成に必要な時間は、スナップショットボリュームが作成されるわずかな時間に比べると、一般的に長くなります。

オンラインバックアップ手順を完了するには、`vxassist snapshot` コマンドを SNAPDONE 状態のスナップショットを持つボリューム上で実行します。このタスクでは、SNAPDONE 状態のスナップショットを切り離し（ミラーにするために）通常の新規ボリュームを作成し、スナップショットミラーとしてスナップショットボリュームに接続します。これで、スナップショットは通常の機能を持つボリュームになり、スナップショットは ACTIVE 状態に設定されます。

vxassist コマンドを使ってボリュームのバックアップを作成するには、次の手順を実行します。

1 次のコマンドを使って、ボリュームにスナップショットミラーを作成します。

```
# vxassist [-b] [-g diskgroup] snapstart [nmirror=N] volume
```

たとえば、ボリューム voldef のスナップショットミラーを作成するには、次のコマンドを使います。

```
# vxassist [-g diskgroup] snapstart voldef
```

vxassist snapstart コマンドによって書き込み専用ミラーが作成され、バックアップ対象のボリュームに追加されて同期されます。

デフォルトでは、VxVM はデータボリュームのプレックスをすでに保持しているディスクを避けてスナップショットミラーを配置しようとします。ただし、ディスクグループ内に十分な領域が存在しない場合は、配置先に選択の余地がありません。この場合は、VxVM は、ディスクグループ内の他のディスクで使える領域を使います。他のボリュームのプレックスを保持しているディスク上にスナップショットプレックスが配置されると、後にスナップショットボリュームを別のディスクグループに移動しようとする際に、問題が発生する可能性があります。

p.955 の「ディスクグループ間の DCO ボリュームの移動」を参照してください。

ストレージ属性を使って、スナップショットプレックスに使うディスクを明示的に指定すると、デフォルトのストレージ割り当てポリシーを上書きすることができます。

p.248 の「指定したディスクにおけるボリュームの作成」を参照してください。

-b オプションを指定して vxassist snapstart コマンドをバックグラウンドで起動する場合、次のように vxassist snapwait コマンドを使ってミラーの作成完了まで待機できます。

```
# vxassist [-g diskgroup] snapwait volume
```

vxassist snapstart コマンドがバックグラウンドで実行されていない場合は、ミラーがボリュームと同期されるまでは終了できません。同期化が完了すると、そのミラーをスナップショットボリュームのプレックスとして使えるようになります。元のボリュームに追加されたミラーの内容は、スナップショットを作成するまで更新され続けます。

nmirror 属性を使って、スナップショットボリュームに必要な数だけスナップショットミラーを作成します。バックアップには、通常 1 つ(デフォルト)で十分です。

ボリューム内の既存のプレックスからスナップショットプレックスを作成することもできます。

p.686 の「プレックスからスナップショットプレックスへの変換」を参照してください。

2 スナップショットの作成に適した時間を選択します。可能ならば、ユーザーのボリュームに対するアクセスができるだけ少ないときにスナップショットを取得するよう計画します。

3 次のコマンドを使ってスナップショットボリュームを作成します。

```
# vxassist [-g diskgroup] snapshot [nmirror=N] volume snapshot
```

必要に応じて、nmirror 属性を使ってスナップショットボリューム内のミラー数を指定します。

たとえば、voldef ボリュームのスナップショットを作成するには、次のコマンドを使います。

```
# vxassist -g mydg snapshot voldef snapvoldef
```

vxassist snapshot タスクでは終了したスナップショットミラーを切断し、新規ボリュームを作成してスナップショットミラーを接続します。この手順は数分で完了します。スナップショットボリュームは、スナップショット時の元のボリュームを反映します。元のボリュームをアプリケーションやユーザーが使えるようにしたままで、スナップショットボリュームをバックアップに使えます。

必要に応じて、ディスクグループ内の複数のボリュームに対して同時にスナップショットボリュームを作成できます。

p.687 の「vxassist コマンドによる複数のスナップショットの作成」を参照してください。

4 VxVM 以外のファイルシステムやデータベースのログ再生の場合は、fsck などの適切なユーティリティを使って、一時使用ボリュームの内容をクリーンにします。VxVM はスナップショットを作成する直前に VxFS を呼び出し、VxFS ファイルシステムを定常状態にするため、通常、一時使用ボリュームの VxFS ファイルシステムに対して fsck を実行する必要はありません。VxFS ファイルシステムにデータベースが含まれている場合は、データベースのログ再生を引き続き実行する必要があります。

5 スナップショットのデータのバックアップが必要な場合は、適切なユーティリティやオペレーティングシステムのコマンドを使って、スナップショットの内容をテープまたは他のバックアップメディアにコピーします。

6 バックアップが完了すると、スナップショットボリュームの処理方法を次の 3 つの中から選択できます。

- スナップショットボリュームのプレックスの一部または全体を元のボリュームに再接続します。
  p.688 の「スナップショットボリュームの再接続」を参照してください。
- スナップショットの作成前にボリューム上で FastResync が有効になっていた場合は、これによりスナップショットプレックスの再同期が高速化され、バックアップサイクルが手順 3 から再開されるまでの時間が短縮されます。

- スナップショットボリュームと元のボリュームの関連付けを完全に解除します。p.690 の「スナップショットボリュームの関連付けの解除」を参照してください。
- この方法は、テストやレポート生成など、他の用途にコピーを使う場合に便利です。
- 不要であれば、ディスク領域の節約のため、次のコマンドを使ってスナップショットボリュームを削除します。

```
# vxedit [-g diskgroup] -rf rm snapshot
```

FastResync が有効になっていた場合、関連付けられているスナップショットボリュームを解除または削除すると、高速再同期の利点が失われます。使用可能なスナップショットプレックスがなくなった場合、それ以降に作成するスナップショットには、元のボリュームの完全なコピーが必要です。

プレックスからスナップショットプレックスへの変換

従来のサードミラーブレークオフスナップショットでは、ボリュームに既存のプレックスをスナップショットプレックスに変換できます。Veritas はプレックスからスナップショットプレックスに変換するのではなく、インスタントスナップショット機能を使用することをお勧めします。

メモ: 階層化ボリュームまたは関連インスタントスナップの DCO ボリュームを持つボリュームのスナップショットプレックスにプレックスを変換することはできません。

状況によっては、`vxassist snapstart` を実行するよりも、ボリューム内の既存のプレックスをスナップショットプレックスに変換した方が都合がよい場合があります。たとえば、スナップショットプレックスを作成するためのディスク領域が不足しており、スナップショットを作成するボリュームに 2 つ以上のプレックスが含まれている場合には、この操作が必要となることがあります。

また、ミラーボリュームが 3 つ以上のプレックスで作成され、`init=active` が指定されている場合は、この手順を使うとスナップショットボリュームの作成を高速化できます。

データの冗長性を確保するため、1 つのボリューム内に少なくとも 2 つのプレックスを保持することをお勧めします。

既存のプレックスを、永続 FastResync が有効であるボリュームのスナップショットプレックスに変換するには、次のコマンドを使います。

```
# vxplex [-g diskgroup] -o dcoplex=dcologplex convert ¥
  state=SNAPDONE plex
```

dcologplex は、新しいスナップショットプレックスと関連付けられる既存の DCO プレックス名です。DCO ボリューム名を確認するには、`vxprint` コマンドを使います。

p.732 の「バージョン 0 の DCO および DCO ボリュームの追加」を参照してください。

たとえば、3 つのプレックスを持つボリューム trivol 内のプレックス trivol-03 から、スナップショットプレックスを作成するには、次のコマンドを使います。

```
# vxplex -o dcoplex=trivol_dco-03 convert state=SNAPDONE ¥
  trivol-03
```

ここで、DCO プレックス trivol_dco_03 は、新しいスナップショットプレックスと関連付けられた DCO プレックス名として指定されています。

既存のプレックスを、非永続 FastResync が有効であるボリュームの、SNAPDONE 状態のスナップショットプレックスに変換するには、次のコマンドを使います。

```
# vxplex [-g diskgroup] convert state=SNAPDONE plex
```

変換されたプレックスは SNAPDONE 状態であり、このプレックスを使ってすぐにスナップショットボリュームを作成できます。

メモ: ボリューム内で唯一の完全な通常のプレックスや、完全でないスパースプレックス、および DRL（dirty region logging）ログプレックスは、スナップショットプレックスに変換できません。

---

p.664 の「サードミラーブレークオフスナップショット」を参照してください。

vxassist コマンドによる複数のスナップショットの作成

複数のボリュームにおけるスナップショットの同時作成を簡単に行うため、スナップショットオプションでは、次のように、引数として複数のボリューム名を使えます。

```
# vxassist [-g diskgroup] snapshot volume1
    volume2 ...
```

デフォルトでは、最初のスナップショットボリュームは *SNAP-volume* と名付けられ、これ以降のスナップショットボリュームは *SNAPnumber-volume* と名付けられます。ここで、*number* は、一意のシリアル番号であり、*volume* はスナップショットが作成されるボリューム名です。このデフォルトのパターンは、vxassist（1M）マニュアルページの説明に従い、オプション -o name=*pattern* を使って上書きできます。たとえば、パターン SNAP%v-%d では、名前の中の *number* と *volume* のコンポーネントの順番が逆になります。

1 つのディスクグループ内のボリュームすべてからスナップショットを取得するには、vxassist にオプション -o allvols を指定します。

```
# vxassist -g diskgroup -o allvols snapshot
```

この操作では、すべての snapstart 操作がボリューム上で完了している必要があります。ディスクグループ内のボリュームのいずれにも、SNAPDONE 状態のスナップショットプレックスがない場合、この操作は失敗します。

メモ: vxsnap コマンドは、複数のスナップショットを作成するための同様の機能を提供します。

スナップショットボリュームの再接続

スナップバック操作は、ボリュームのスナップショットコピーともとのボリュームを結合します。1 つ以上のスナップショットプレックスがスナップショットボリュームから切断され、もとのボリュームに再接続されます。スナップショットプレックスが残らずスナップバックされると、スナップショットボリュームは削除されます。このタスクでは、プレックスが一致するようにボリュームのデータを再同期します。

スナップバック操作は、DCO と DCO ボリュームの追加により RAID 5 ボリュームが特別な階層化ボリュームのレイアウトに変更されている場合を除いて、RAID 5 ボリュームには適用できません。

p.732 の「バージョン 0 の DCO および DCO ボリュームの追加」を参照してください。

スナップバック操作の効率を高めるために、スナップショットを取得する前にボリューム上の FastResync を有効にします。

1 つのスナップショットプレックスをもとのボリュームと再結合するには、次のコマンドを使います。

```
# vxassist [-g diskgroup] snapback snapshot
```

ここで、*snapshot* はボリュームのスナップショットコピーです。

スナップショットボリューム内のすべてのスナップショットプレックスをもとのボリュームと再結合するには、次のコマンドを使います。

```
# vxassist [-g diskgroup] -o allplexes snapback snapshot
```

スナップショットボリューム内の特定数のプレックスをもとのボリュームと再結合するには、次のコマンドを使います。

```
# vxassist [-g diskgroup] snapback nmirror=number snapshot
```

再接続するスナップショットボリュームのミラー数は、nmirror 属性で指定します。

再接続とデータの再同期が完了したスナップショットプレックスは、別の snapshot 操作で使えるようになります。

デフォルトでは、再接続したスナップショットプレックスがもとのボリュームのデータで更新されます。逆にスナップショットボリュームからデータをコピーするには、次のコマンドを使います。

```
# vxassist [-g diskgroup] -o resyncfromreplica snapback snapshot
```

警告: スナップショットボリュームがマウントされている場合は、スナップバックを実行する前にマウントを解除する必要があります。また、resyncfromreplica オプションを使う前に、プライマリボリュームに対応するファイルシステムのマウントを解除する必要があります。

スナップショットボリュームへのプレックスの追加

スナップバック操作後にスナップショットボリューム内の既存のプレックスを保持する場合は、スナップバックに使うスナップショットプレックスを追加作成できます。

スナップショットボリュームにプレックスを追加するには

1 次の vxprint コマンドを使って、スナップショットボリュームのデータ変更オブジェクト(DCO)および DCO ボリュームの名前を確認します。

```
# DCONAME=`vxprint [-g diskgroup] -F%dco_name snapshot`
# DCOVOL=`vxprint [-g diskgroup] -F%log_vol $DCONAME`
```

2 vxassist mirror コマンドを使って、既存のスナップショットボリュームおよびその DCO ボリュームのミラーを作成します。

```
# vxassist -g diskgroup mirror snapshot
# vxassist -g diskgroup mirror $DCOVOL
```

DCO ボリューム内の新しいプレックスは、スナップショット内の新しいデータプレックスで使うために必要です。

3 vxprint コマンドを使って、追加のスナップショットプレックスの名前を確認します。

```
# vxprint -g diskgroup snapshot
```

4 vxprint コマンドを使って、追加の DCO プレックスのレコード ID を確認します。

```
# vxprint -g diskgroup -F%rid $DCOVOL
```

5 vxedit コマンドを使って、新しいデータプレックスの dco_plex_rid フィールドを新しい DCO プレックスの名前に設定します。

```
# vxedit -g diskgroup set dco_plex_rid=dco_plex_rid new_plex
```

これで、新しいデータプレックスを、スナップバック操作の実行に使う準備が整いました。

スナップショットボリュームの関連付けの解除

スナップショットボリュームが独立したボリュームになるよう、スナップショットともとのボリュームとの間のリンクを完全に解除します。次のコマンドを使ってスナップショットボリューム *snapshot* の関連付けを解除します。

```
# vxassist snapclear snapshot
```

スナップショット情報の表示

vxassist snapprint コマンドは、もとのボリュームとそれぞれの複製(スナップショットコピー)との関連性を表示します。

```
# vxassist snapprint [volume]
```

このコマンドによる出力例は次のようになります。

```
# vxassist -g mydg snapprint v1

V  NAME         USETYPE      LENGTH
SS SNAPOBJ      NAME         LENGTH     %DIRTY
DP NAME         VOLUME       LENGTH     %DIRTY

v  v1           fsgen        20480
ss SNAP-v1_snp  SNAP-v1      20480      4
dp v1-01        v1           20480      0
dp v1-02        v1           20480      0

v  SNAP-v1      fsgen        20480
ss v1_snp       v1           20480      0

# vxassist -g mydg snapprint v2

V  NAME         USETYPE      LENGTH
```

```
SS SNAPOBJ       NAME          LENGTH      %DIRTY
DP NAME          VOLUME        LENGTH      %DIRTY

v  v2            fsgen         20480
ss --            SNAP-v2       20480       0
dp v2-01         v2            20480       0

v  SNAP-v2       fsgen         20480
ss --            v2            20480       0
```

この例では、永続 FastResync はボリューム v1 で有効であり、非永続 FastResync はボリューム v2 で有効です。v、dp および ss で始まる行は、ボリューム、切断されたプレックス、スナップショットプレックスをそれぞれ示しています。%DIRTY フィールドは、スナップショットプレックスや切断したプレックスのうち、もとのボリュームに関してダーティであるものの割合を示しています。ボリューム v2 またはスナップショットボリューム SNAP-v2 と関連付けられているスナップオブジェクトがないことに注意してください。

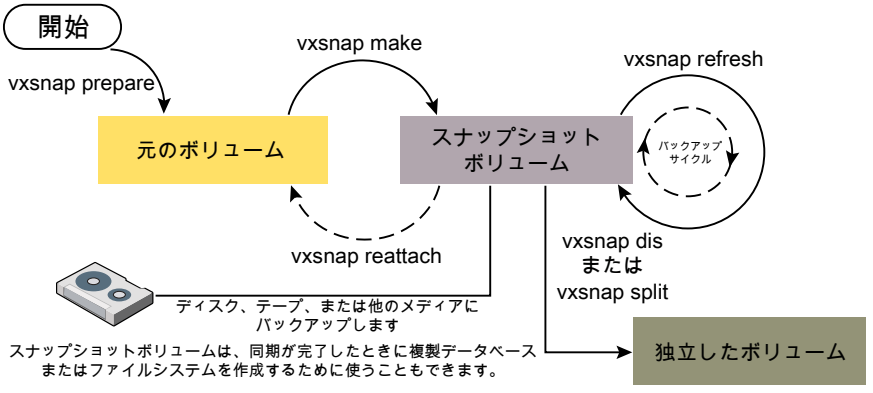
ボリュームを指定した場合、FastResync マップがそのボリュームで有効になっていないと、snapprint コマンドではエラーメッセージを表示します。

フルサイズインスタントスナップショット

フルサイズインスタントスナップショットは、サードミラーボリュームスナップショットモデルの変形です。このスナップショットを使うと、スナップショットプレックスの作成後すぐに、スナップショットボリュームの I/O アクセスが可能になります。

図 26-2 にフルサイズインスタントボリュームスナップショットのモデルを示します。

図 26-2　フルサイズインスタントスナップショットの作成とバックアップサイクルでの使用法



インスタントスナップショットを作成するには、vxsnap make コマンドを使います。このコマンドは、スナップショットボリューム用に適切に準備した空のボリュームに適用するために使えます。また、もとのボリュームから 1 つ以上の同期化されたプレックスを切り離すためにも使えます。

フルサイズインスタントスナップショットでは、もとのボリュームからスナップショットプレックスを完全に同期化しないで、バックアップを作成したり、もとのボリュームからその内容をすぐに更新できます。また、そのプレックスをもとのボリュームに接続することもできます。

VxVM では、コピーオンライト機構を使って、スナップショットの作成時に、スナップショットボリュームが、もとのボリュームの内容を保持するようにします。ボリュームのもとの内容が上書きされそうになると、データが書き替えられる直前にボリューム内のもとのデータがスナップショットボリューム上に移動されます。時間の経過とともにボリュームの内容が更新されると、もとの内容は段階的にスナップショットボリュームに再配置されます。

読み取り要求がスナップショットボリュームに到着したのにデータが（まだ変更されていないため）元のボリュームに置かれている場合、VxVM は自動的に元のボリュームから透過的にデータを読み込みます。

必要に応じて、バックグラウンド（非ブロック）またはフォアグラウンド（ブロック）でのスナップショットボリュームの同期化を実行できます。これは、スナップショットボリュームを別のディスクグループに移動してオフホストで処理する場合や、スナップショットボリュームを独立したボリュームに変更する場合に便利です。

vxsnap refresh コマンドを使うと、たとえばバックアップを作成する前にスナップショット内のデータを更新できます。

vxsnap reattach コマンドを使って、スナップショットプレックスを元のボリュームに接続し、元のボリュームからプレックス内のデータを再同期できます。または、vxsnap restore コマンドを使って、過去の時点で作成したスナップショットから元のボリュームの内容を復元できます。さらに、もとのボリュームのリストアが完了した後でもスナップショットボリュームを存続させるかどうかを選択することもできます。

デフォルトでは、スナップショットミラーのデータを再同期するのに必要な時間と I/O を最小限にするために VxVM の FastResync 機能を使います。インスタントスナップショットを作成するには、FastResync を有効にする必要があります。

p.703 の「フルサイズインスタントスナップショットの作成と管理」を参照してください。

フルサイズインスタントスナップショットとリンクブレークオフスナップショットで使うための空のボリュームを準備する必要があります。

p.698 の「フルサイズインスタントスナップショットまたはリンクブレークオフスナップショットに使うボリュームの作成」を参照してください。

インスタントスナップショットの作成

メモ: この機能を使うには、Veritas InfoScale Storage または Veritas InfoScale Enterprise のライセンスが必要です。

VxVM では、vxsnap コマンドを使ってインスタントスナップショットを作成できます。

DCO と DCO ボリュームの追加により特別な階層化ボリュームのレイアウトに変換されている RAID 5 ボリュームのインスタントスナップショットも作成できます。

フルサイズインスタントスナップショットのプレックスには、元のボリュームと同じ量の領域が必要です。ボリュームの領域最適化インスタントスナップショットを作成する場合には、スナップショットの存在期間中に変更された親ボリュームの元の内容を記録する領域だけあれば十分です。

ボリュームのバックアップをコマンドラインやスクリプトから実行する際の方法として、vxsnap コマンドを使うことをお勧めします。vxsnap prepare および make の各タスクにより、ユーザーの作業中断を最小に抑えてボリュームのオンラインバックアップを実行できます。

vxsnap prepare は DCO と DCO ボリュームを作成し、それを元のボリュームに関連付けます。また、永続 FastResync も有効にします。

vxsnap は、バックアップの作成にすぐに使えるインスタントスナップショットを作成します。スナップショットが作成されると、インスタントスナップショットボリュームのデータの読み取り要求は、元のボリュームの更新されていない領域、またはスナップショットによって記録された更新済み領域の元の内容のコピーから読み取りを行うことで処理されます。

メモ: 元のボリュームからのフルサイズインスタントスナップショットの同期は、デフォルトで有効になっています。vxsnap make に syncing=no 属性を指定すると同期が無効になるため、インスタントスナップショットの内容が、スナップショットが作成された時点におけるもとのボリュームの内容と完全に同期していない可能性があります。このような場合、そのスナップショットはオフホスト処理に使うことも、独立したボリュームになることもできません。

vxsnap refresh コマンドを使うと、フルサイズインスタントスナップショットまたは領域最適化インスタントスナップショットをいつでもすぐに更新できます。完全に同期化されたインスタントスナップショットが必要な場合は、まず新しい再同期処理を完了する必要があります。

ボリュームセットのインスタントスナップショットを作成するには、vxsnap コマンドでボリューム名の代わりにボリュームセット名を使います。

p.711 の「ボリュームセットのインスタントスナップショットの作成」を参照してください。

vxsnap prepare コマンドまたは vxassist make コマンドを使ってインスタントスナップショット操作用にボリュームを準備する場合、チューニングパラメータ voliomem_maxpool_sz の値の半分より大きい値を指定すると、操作は正常に実行されますが次のような警告が表示されます(voliomem_maxpool_sz が 12 MB に設定されているシステムの場合)。

```
VxVM vxassist WARNING V-5-1-0 Specified regionsize is
larger than the limit on the system
(voliomem_maxpool_sz/2=6144k).
```

このメッセージが表示された場合、このようなボリュームに対して、vxsnap make、refresh、restore 操作を実行しようとすると、システムが異常終了する可能性があるため操作は失敗します。このボリュームで使える操作は、ブレークオフスナップショット操作の reattach 操作と make 操作のみです。

このようなボリュームでインスタントスナップショット操作を使えるようにするには、vxsnap unprepare を実行してから再度 vxsnap prepare を実行し、領域のサイズが voliomem_maxpool_sz の半分より小さくなる(この例では 1 MB と指定)ようにボリュームを準備し直す必要があります。

```
# vxsnap -g mydg -f unprepare vol1
# vxsnap -g mydg prepare vol1 regionsize=1M
```

p.711 の「ボリュームセットのインスタントスナップショットの作成」を参照してください。

p.700 の「領域最適化インスタントスナップショットの作成と管理」を参照してください。

p.703 の「フルサイズインスタントスナップショットの作成と管理」を参照してください。

p.705 の「サードミラーブレークオフスナップショットの作成と管理」を参照してください。

p.708 の「リンクされたブレークオフスナップショットボリュームの作成と管理」を参照してください。

インスタントスナップの DCO と DCO ボリュームの追加

インスタントスナップショット操作を実行するためのボリュームの準備として、まず、インスタントスナップのデータ変更オブジェクト(DCO)および DCO ボリュームをそのボリュームに関連付ける必要があります。この手順により、ボリュームの永続 FastResync も有効になります。

次の手順を実行する必要があるのは、ボリュームにインスタントスナップ DCO ボリュームがない場合だけです。

デフォルトでは、シンプロビジョニング LUN のボリュームはインスタントスナップ DCO ボリュームとともに作成されます。

インスタントスナップ DCO と DCO ボリュームを追加するには

1 ボリュームにインスタントスナップデータ変更オブジェクト(DCO)と DCO ボリュームがあることと、ボリュームで FastResync が有効になっていることを確認します。

```
# vxprint -g volumedg -F%instant volume
# vxprint -g volumedg -F%fastresync volume
```

2 つのコマンドがどちらも on を返した場合は、手順 3 にスキップします。それ以外の場合は、引き続き手順 2 を実行してください。

2 インスタントスナップショット用のボリュームを準備するには、次のコマンドを使います。

```
# vxsnap [-g diskgroup] prepare volume [regionsize=size] ¥
  [ndcomirs=number] [alloc=storage_attributes]
```

インスタントスナップ DCO と DCO ボリュームが存在しない場合にのみ、ボリュームで vxsnap prepare コマンドを実行します。

たとえば、ディスクグループ mydg にボリューム myvol を準備するには、次のコマンドを使います。

```
# vxsnap -g mydg prepare myvol regionsize=128k ndcomirs=2 ¥
  alloc=mydg10,mydg11
```

この例では、DCO オブジェクトと、ディスク mydg10 および mydg11 上に 2 つのプレックスが配置された冗長性のある DCO ボリュームを作成し、さらに myvol に関連付けています。領域のサイズもデフォルトの 64 KB から 128 KB に拡張されます。領域のサイズは 16 KB 以上の 2 の累乗の値にする必要があります。値を小さくすると変更マップに必要なディスク領域が増えますが、細分化すればするほど再同期が速くなります。

3 ディスクグループ内のボリュームについて複数の領域最適化インスタントスナップショットが必要な場合は、スナップショットごとに別個のキャッシュオブジェクトを作成するよりも、1 つの共有キャッシュオブジェクトをディスクグループ内に作成したほうが便利です。

p.696 の「共有キャッシュオブジェクトの作成」を参照してください。

フルサイズインスタントスナップショットとリンクされたブレークオフスナップショットの場合は、スナップショットボリュームとして使うボリュームを準備する必要があります。このボリュームは、スナップショットを作成するデータボリュームと同じサイズにし、領域のサイズも同じにする必要があります。

p.698 の「フルサイズインスタントスナップショットまたはリンクブレークオフスナップショットに使うボリュームの作成」を参照してください。

共有キャッシュオブジェクトの作成

ディスクグループ内のボリュームについて複数の領域最適化インスタントスナップショットが必要な場合は、スナップショットごとに別個のキャッシュオブジェクトを作成するのではなく、1 つの共有キャッシュオブジェクトをディスクグループ内に作成できます。

共有キャッシュオブジェクトを作成するには

1 キャッシュオブジェクトの基盤となるキャッシュボリュームに適用する特性を決定します。

- キャッシュボリュームにはスナップショットの更新の合間に発生した親ボリュームの変更が記録されるため、それに見合ったサイズが必要です。更新間隔が 24 時間の場合の推奨値は、親ボリュームの合計サイズの 10 % です。
- 冗長性を確保するためにキャッシュボリュームをミラー化できます。
- キャッシュボリュームをミラー化する場合は、ミラーと同数以上のディスク上に領域が必要になります。親ボリュームと同じディスクは使えません。重要なボリュームの I/O 処理効率に影響を与えたり、ディスクグループの分割および結合の操作を妨げることがないようにするために、これらのディスクを重要なボリュームで使われているディスクと共有しないでください。

2 特性を決定したら、vxassist コマンドを使ってキャッシュボリュームを作成します。次の例は、ディスク mydg16 と mydg17 を使ってディスクグループ mydg 内に 1 GB のキャッシュボリューム cachevol を作成し、そのキャッシュボリュームをミラー化する方法を示しています。

```
# vxassist -g mydg make cachevol 1g layout=mirror ¥
  init=active mydg16 mydg17
```

属性 init=active は、キャッシュボリュームをすぐに使えるようにします。

3 vxmake cache コマンドを使って、前の手順で作成したキャッシュボリュームの上にキャッシュオブジェクトを作成します。

```
# vxmake [-g diskgroup] cache cache_object ¥
  cachevolname=volume [regionsize=size] [autogrow=on] ¥
  [highwatermark=hwmk] [autogrowby=agbvalue] ¥
  [maxautogrow=maxagbvalue]]
```

領域のサイズ regionsize を指定する場合は、16 KB(16k)以上の 2 の累乗の値にする必要があります。この値を指定しなかった場合、キャッシュの領域のサイズは 64 KB に設定されます。

キャッシュを共有する領域最適化スナップショットの領域のサイズは、キャッシュに設定した領域のサイズと同サイズかまたはその整数倍にする必要があります。また、もとのボリュームの領域のサイズがキャッシュの領域のサイズより小さいと、スナップショットの作成は失敗します。

領域最適化スナップショットの領域のサイズがキャッシュの領域のサイズとは異なる場合、領域のサイズが同じ場合に比べシステムのパフォーマンスが低下する可能性があります。

キャッシュが自動的に拡張されないようにするには、autogrow=off を指定します。デフォルトでは、autogrow=on です。

次の例では、キャッシュボリューム cachevol の上にキャッシュオブジェクト cobjmydg を作成し、キャッシュの領域のサイズを 32 KB に設定して、自動拡張機能を有効にしています。

```
# vxmake -g mydg cache cobjmydg cachevolname=cachevol ¥
  regionsize=32k autogrow=on
```

4 次のコマンドを使用してキャッシュオブジェクトを有効にします。

```
# vxcache [-g diskgroup] start cache_object
```

たとえば、キャッシュオブジェクト cobjmydg を起動するには、次のコマンドを実行します。

```
# vxcache -g mydg start cobjmydg
```

p.725 の「キャッシュの削除」を参照してください。

フルサイズインスタントスナップショットまたはリンクブレークオフスナップショットに使うボリュームの作成

フルサイズインスタントスナップショットまたはリンクされたブレークオフスナップショットに使う空のボリュームを作成するには、次の手順を実行します。

1 元のボリュームに対して vxprint コマンドを実行し、スナップショットボリュームに必要なサイズを検索します。

```
# LEN=`vxprint [-g diskgroup] -F%len volume`
```

ここで示すコマンドは、sh、ksh、bash などの Bourne シェルを前提にしています。その他のシェル（csh や tcsh など）を使う場合は、シェルに合わせてコマンドを修正する必要があります。

2 元のボリュームに対して vxprint コマンドを実行し、DCO ボリュームの名前を確認します。

```
# DCONAME=`vxprint [-g diskgroup] -F%dco_name volume`
```

3 DCO ボリュームに対して vxprint コマンドを実行し、その領域のサイズ(ブロック数)を確認します。

```
# RSZ=`vxprint [-g diskgroup] -F%regionsz $DCONAME`
```

4 vxassist コマンドを使って、必要なサイズと冗長性を備えたボリューム *snapvol* と、適切な領域サイズを持つインスタントスナップの DCO ボリュームを作成します。

```
# vxassist [-g diskgroup] make snapvol $LEN ¥
  [layout=mirror nmirror=number] logtype=dco drl=off ¥
  dcoversion=20 [ndcomirror=number] regionsz=$RSZ ¥
  init=active [storage_attributes]
```

ストレージ属性を指定することにより、vxassist がボリュームの設定に使うディスクやコントローラなどのデバイスを制御できます。

p.248 の「指定したディスクにおけるボリュームの作成」を参照してください。

DCO ミラーの数(ndcomirror)は、ボリューム内のミラー数(nmirror)と同数にします。init=active 属性を指定すると、ボリュームがすぐに使用可能になります。ストレージ属性は、ボリュームを配置するディスクを指定する場合に使います。

スナップショットボリュームと DCO ボリュームを 1 つの手順で作成するのではなく、まずボリュームを作成し、そのボリュームを対象としてインスタントスナップショット操作のための準備を行う場合は、次のコマンドを実行します。

```
# vxassist [-g diskgroup] make snapvol $LEN ¥
  [layout=mirror nmirror=number] init=active ¥
  [storage_attributes]
# vxsnap [-g diskgroup] prepare snapvol [ndcomirs=number] ¥
  regionsize=$RSZ [storage_attributes]
```

VxVM ボリュームのインスタントスナップデータ変更オブジェクト(DCO)と DCO ボリュームのアップグレード

インスタントスナップ DCO(旧バージョン 20 DCO)は、VxVM ボリュームのインスタントスナップショットの作成をサポートします。インスタントスナップ DCO と DCO ボリュームをアップグレードし、VxVM の最新版との互換性を確保します。アップグレード操作はボリュームがオンラインの間、実行できます。

アップグレード操作は、バージョン 0 DCO からのアップグレードをサポートしません。

ディスクグループのすべてのボリュームのインスタントスナップ DCO をアップグレードするには

1 ディスクグループがバージョン 180 以上であることを確認します。ディスクグループをアップグレードするには

```
# vxdg upgrade diskgroup
```

2 次のコマンドを使って、ディスクグループのすべてのボリュームのインスタントスナップ DCO をアップグレードします。

```
# vxsnap -g diskgroup upgradeall
```

diskgroup はアップグレードされるボリュームを含むディスクグループです。

upgradeall 操作の追加オプションについては、vxsnap(1M)マニュアルページを参照してください。

指定したボリュームのインスタントスナップ DCO をアップグレードするには

1 ディスクグループがバージョン 180 以上であることを確認します。ディスクグループをアップグレードするには

```
# vxdg upgrade diskgroup
```

2 DCO をアップグレードするには、次のコマンドに 1 つ以上のボリュームまたはボリュームセットを指定します。

```
# vxsnap [-g diskgroup] upgrade
[volume1|volset1][volume2|volset2...]
```

diskgroup はアップグレードされるボリュームを含むディスクグループです。

アップグレード操作の追加オプションについては、vxsnap(1M)マニュアルページを参照してください。

領域最適化インスタントスナップショットの作成と管理

領域最適化インスタントスナップショットは、コピーオンライト機構を使うと処理効率が低下することがあるので、書き込みを集中的に行うボリューム(データベース REDO ログなど)には適していません。

ボリュームとスナップショットを個別のディスクグループに分割する場合は(オフホスト処理を実行する場合など)、キャッシュオブジェクトを必要としない、完全に同期されたフルサイズスナップショット、サードミラーブレークオフスナップショット、リンクされたブレークオフスナップショットのいずれかを使う必要があります。領域最適化インスタントスナップショットを使うことはできません。

ボリュームに対して指定した領域のサイズが、キャッシュに設定した領域のサイズよりも小さい場合、共有キャッシュを使う領域最適化スナップショットの作成は失敗します。

領域最適化スナップショットの領域のサイズがキャッシュの領域のサイズとは異なる場合、領域のサイズが同じ場合に比べシステムのパフォーマンスが低下する可能性があります。

p.696 の「共有キャッシュオブジェクトの作成」を参照してください。

スナップショットの属性は、一組にまとめて vxsnap make コマンドに指定します。このコマンドでは複数の組を指定できます。作成するスナップショットごとに組を 1 つ指定する必要があります。組の各要素は、スラッシュ文字(/)で区切ります。複数の組はスペースで区切ります。

領域最適化インスタントスナップショットを作成して管理するには、次の手順を実行します。

1 vxsnap make コマンドを実行して、領域最適化インスタントスナップショットを作成します。ディスクグループ内の既存のキャッシュオブジェクトを使ってこのスナップショットを作成するか、または新しいキャッシュオブジェクトを作成できます。

- 特定の共有キャッシュオブジェクトを使う領域最適化インスタントスナップショット *snapvol* を作成するには、次のコマンドを実行します。

  ```
  # vxsnap [-g diskgroup] make source=vol/newvol=snapvol¥
  /cache=cacheobject [alloc=storage_attributes]
  ```

  たとえば、ディスクグループ mydg のディスク mydg14 上のボリューム myvol において、共有キャッシュオブジェクト cobjmydg を使う領域最適化インスタントスナップショット snap3myvol を作成するには、次のコマンドを使います。

  ```
  # vxsnap -g mydg make source=myvol/newvol=snap3myvol¥
  /cache=cobjmydg alloc=mydg14
  ```

  DCO は指定の割り当てで作成されます。

- 領域最適化インスタントスナップショット *snapvol* を作成し、そのスナップショットで使うキャッシュオブジェクトも作成するには、次のコマンドを使います。

  ```
  # vxsnap [-g diskgroup] make source=vol/newvol=snapvol¥
  [/cachesize=size][/autogrow=yes][/ncachemirror=number]¥
    [alloc=storage_attributes]
  ```

  cachesize 属性には、ボリュームのサイズに応じたキャッシュのサイズを指定します。autogrow 属性は、オーバーフローの危険性がある場合に VxVM によるキャッシュの自動拡張を行うかどうかを指定します。デフォルトでは、autogrow=on であり、キャッシュは自動的に拡張されます。

autogrow 属性が有効になっていても、キャッシュを拡張できない場合があります。この場合は、同じキャッシュを使っている最も古い最大のスナップショットが無効にされ、そのキャッシュ領域が使えるように解放されます。

ncachemirror 属性には、キャッシュボリュームに作成するミラーの数を指定します。バックアップ用には、デフォルト値の 1 で十分です。

たとえば、ディスクグループ mydg のディスク mydg15 上のボリューム myvol において、新しく割り当てられた 1 GB のキャッシュオブジェクトを使い、かつサイズを自動的に拡張できる領域最適化インスタントスナップショット snap4myvol を作成するには、次のコマンドを使います。

```
# vxsnap -g mydg make source=myvol/new=snap4myvol¥
/cachesize=1g/autogrow=yes alloc=mydg15
```

cachesize を指定することによって暗黙的にキャッシュを作成し、ncachemirror に 1 より大きい値を指定した場合は、DRL（dirty region logging）を実現するために DCO がキャッシュボリュームに関連付けられます。DRL を使うと、システムクラッシュ後のキャッシュバッキングストアのリカバリが高速化されます。この DCO は、ソースボリュームの DCO が格納されているディスクと同じディスクに割り当てられます。これは、ディスクグループの移動、分割および結合の操作を実行しても、キャッシュとソースボリュームが同じディスクグループに残るようにするためです。

2 VxVM 以外のファイルシステムやデータベースのログ再生の場合は、fsck などの適切なユーティリティを使って、一時使用ボリュームの内容をクリーンにします。VxVM はスナップショットを作成する直前に VxFS を呼び出し、VxFS ファイルシステムを定常状態にするため、通常、一時使用ボリュームの VxFS ファイルシステムに対して fsck を実行する必要はありません。VxFS ファイルシステムにデータベースが含まれている場合は、データベースのログ再生を引き続き実行する必要があります。

3 スナップショットのデータをバックアップするには、適切なユーティリティまたはオペレーティングシステムのコマンドを使って、スナップショットの内容をテープまたはその他のバックアップメディアにコピーします。

4 ここで、次の方法を選択できます。

- スナップショットの内容を更新します。この方法を使うと、元のボリュームの新しいポイントインタイムイメージを作成し、別のバックアップに使うことができます。スナップショットで同期化がすでに進行中の場合は、この操作によって、スナップショットの大部分の再同期化が必要になることがあります。
  p.715 の「インスタント領域最適化スナップショットの更新」を参照してください。
- スナップショットボリュームから元のボリュームの内容を復元します。領域最適化インスタントスナップショットは、この操作によって変更されることはありません。
  p.717 の「領域最適化インスタントスナップショットからのボリュームのリストア」を参照してください。

- スナップショットを破棄します。
  p.718 の「インスタントスナップショットの削除」を参照してください。

フルサイズインスタントスナップショットの作成と管理

フルサイズインスタントスナップショットは、コピーオンライト機構を使うと処理効率が低下することあがるので、書き込みを集中的に行うボリューム(データベース REDO ログなど)には適していません。

フルサイズインスタントスナップショットの場合は、スナップショットボリュームとして使うボリュームを準備する必要があります。このボリュームは、スナップショットを作成するボリュームと同じサイズにし、領域のサイズも同じにする必要があります。

p.698 の「フルサイズインスタントスナップショットまたはリンクブレークオフスナップショットに使うボリュームの作成」を参照してください。

スナップショットの属性は、一組にまとめて `vxsnap make` コマンドに指定します。このコマンドでは複数の組を指定できます。作成するスナップショットごとに組を 1 つ指定する必要があります。組の各要素は、スラッシュ文字(/)で区切ります。複数の組はスペースで区切ります。

フルサイズインスタントスナップショットを作成して管理するには、次の手順を実行します。

1 フルサイズインスタントスナップショットを作成するには、次の形式の `vxsnap make` コマンドを使います。

```
# vxsnap [-g diskgroup] make source=volume/snapvol=snapvol¥
[/snapdg=snapdiskgroup] [/syncing=off]
```

このコマンドでは、以前準備したボリューム ***snapvol*** を指定します。

たとえば、事前に準備したボリューム `snap1myvol` を、ボリューム `myvol` のスナップショットとしてディスクグループ `mydg` で使うには、次のコマンドを使います。

```
# vxsnap -g mydg make source=myvol/snapvol=snap1myvol
```

空のボリュームから作成されたフルサイズインスタントスナップショットの場合、バックグラウンドの同期がデフォルトで有効になっています(`syncing=on` 属性を指定した場合と同じです)。スナップショットを別のディスクグループに移動したり、独立したボリュームに変更する場合は、スナップショットのコンテンツが親ボリュームのコンテンツと同期されるまで待機する必要があります。

次に示すように、`vxsnap syncwait` コマンドを使うと、スナップショットボリュームの同期の完了を待機できます。

```
# vxsnap [-g diskgroup] syncwait snapvol
```

たとえば、スナップショットボリューム snap2myvol での同期の完了を待機する場合は、次のコマンドを使います。

```
# vxsnap -g mydg syncwait snap2myvol
```

このコマンドは、スナップショットボリュームの同期が完了すると終了します(リターンコードは 0)。スナップショットボリュームは別のディスクグループに移動されるか、独立したボリュームになります。

p.721 の「インスタントスナップショットの同期の制御」を参照してください。

必要に応じて、次のコマンドを使って、ボリュームの同期が完了しているかどうかをテストできます。

```
# vxprint [-g diskgroup] -F%incomplete snapvol
```

このコマンドは、ボリューム snapvol の同期が完了している場合は ***off*** の値を返し、完了していない場合は on の値を返します。

同期の進行状況は、vxsnap print コマンドでも確認できます。

p.690 の「スナップショット情報の表示」を参照してください。

スナップショットを別のディスクグループに移動したり、独立したボリュームに変更する予定がない場合は、syncing=off 属性を指定します。これにより、システムにおける不要なオーバーヘッドをなくすことができます。たとえば、ボリューム myvol のスナップショットを作成する場合に同期を無効にするには、次の形式で vxsnap make コマンドを使います。

```
# vxsnap -g mydg make source=myvol/snapvol=snap1myvol¥
/syncing=off
```

2 VxVM 以外のファイルシステムやデータベースのログ再生の場合は、fsck などの適切なユーティリティを使って、一時使用ボリュームの内容をクリーンにします。VxVM はスナップショットを作成する直前に VxFS を呼び出し、VxFS ファイルシステムを定常状態にするため、通常、一時使用ボリュームの VxFS ファイルシステムに対して fsck を実行する必要はありません。VxFS ファイルシステムにデータベースが含まれている場合は、データベースのログ再生を引き続き実行する必要があります。

3 スナップショットのデータをバックアップするには、適切なユーティリティまたはオペレーティングシステムのコマンドを使って、スナップショットの内容をテープまたはその他のバックアップメディアにコピーします。

4 ここで、次の方法を選択できます。

- スナップショットの内容を更新します。この方法を使うと、元のボリュームの新しいポイントインタイムイメージを作成し、別のバックアップに使うことができます。スナップショットで同期化がすでに進行中の場合は、この操作によって、スナップショットの大部分の再同期化が必要になることがあります。

p.715 の「インスタント領域最適化スナップショットの更新」を参照してください。

- スナップショットボリュームのプレックスの一部または全体を元のボリュームに再接続します。
  p.715 の「フルサイズインスタントスナップショットまたはプレックスのブレークオフスナップショットの再接続」を参照してください。
- スナップショットボリュームから元のボリュームの内容を復元します。結果の動作として、元のボリュームにスナップショットボリュームのプレックスを戻さない、サブセットのみ戻す、すべてを戻すのいずれかを選択できます。
  p.717 の「領域最適化インスタントスナップショットからのボリュームのリストア」を参照してください。
- スナップショットボリュームと元のボリュームの関連付けを完全に解除します。この方法は、テストやレポート生成など、他の用途にコピーを使う場合に便利です。必要に応じて、関連付けを解除したボリュームを削除することができます。
  p.717 の「インスタントスナップショットの関連付けの解除」を参照してください。
- スナップショットがスナップショット階層の一部である場合は、この階層と親ボリュームの関連付けを解除することができます。
  p.718 の「インスタントスナップショット階層の分割」を参照してください。

サードミラーブレークオフスナップショットの作成と管理

ブレークオフスナップショットは、データベースREDOログなどの書き込みを集中的に行うボリュームに適しています。

ボリューム内の 1 以上の既存のプレックスを、サードミラーブレークオフインスタントスナップショットボリュームに変換するには、ボリュームがミラーまたはミラー化ストライプのレイアウトを持つ非階層化ボリュームであるか、特別な階層化ボリュームに変換された RAID 5 ボリュームである必要があります。ストライプ化ミラーレイアウトのボリューム内のプレックスは、サブボリュームレベルでミラー化されるため切り離すことができません。

スナップショットの属性は、一組にまとめて `vxsnap make` コマンドに指定します。このコマンドでは複数の組を指定できます。作成するスナップショットごとに組を 1 つ指定する必要があります。組の各要素は、スラッシュ文字(/)で区切ります。複数の組はスペースで区切ります。

サードミラーブレークオフスナップショットを作成して管理するには、次の手順を実行します。

1 このスナップショットの作成には、ボリュームの既存の ACTIVE プレックスの一部を利用するか、または次のコマンドを実行してボリュームに新しいスナップショットのミラーを追加できます。

```
# vxsnap [-b] [-g diskgroup] addmir volume [nmirror=N] ¥
  [alloc=storage_attributes]
```

nmirror 属性を使ってミラーの数の変更を指定しないかぎり、vxsnap addmir コマンドはデフォルトで 1 つのスナップショットミラーをボリュームに追加します。ミラーは完全に同期されるまで SNAPATT 状態のままです。-b オプションを使うと、同期をバックグラウンドで実行できます。同期が完了すると、ミラーは SNAPDONE 状態になります。

たとえば、ディスク mydg10 および mydg11 上のボリューム vol1 に 2 つのミラーを追加するには、次のコマンドを実行します。

```
# vxsnap -g mydg addmir vol1 nmirror=2 alloc=mydg10,mydg11
```

vxsnap addmir コマンドに -b オプションを指定する場合、次の例のように vxsnap snapwait コマンドを使って、スナップショットプレックスの同期が完了するまで待機できます。

```
# vxsnap -g mydg snapwait vol1 nmirror=2
```

2 サードミラーブレークオフスナップショットを作成するには、次の形式の vxsnap make コマンドを使います。

```
# vxsnap [-g diskgroup] make source=volume[/newvol=snapvol]¥
{/plex=plex1[,plex2,...]|/nmirror=number}
```

次のいずれかの属性を指定し、元のボリューム内の 1 つ以上の既存のプレックスを切り離すことにより新しいスナップショットボリューム *snapvol* を作成します。

plex	ブレークオフする既存ボリューム内のプレックスを指定します。
nmirror	ブレークオフするプレックスの数を指定します。この属性は SNAPDONE 状態のプレックスにのみ使えます。(このようなプレックスは vxsnap addmir コマンドを使ってボリュームに追加された可能性があります) に対してのみ有効です。

ボリュームにある 1 つ以上の ACTIVE または SNAPDONE プレックスから作成されるスナップショットは定義上、すでに同期されています。

バックアップを目的とする場合には、1 つのプレックスのスナップショットボリュームで十分です。

たとえば、ボリューム内の 1 つの既存のプレックスから、ディスクグループ mydg 内のボリューム myvol のインスタントスナップショットボリューム snap2myvol を作成するには、次のコマンドを使います。

```
# vxsnap -g mydg make source=myvol/newvol=snap2myvol/nmirror=1
```

次の例は、ボリューム内の 2 つの既存のプレックスからミラースナップショットを作成する方法を示しています。

```
# vxsnap -g mydg make source=myvol/newvol=snap2myvol/plex=myvol-03,myvol-04
```

3 VxVM 以外のファイルシステムやデータベースのログ再生の場合は、fsck などの適切なユーティリティを使って、一時使用ボリュームの内容をクリーンにします。VxVM はスナップショットを作成する直前に VxFS を呼び出し、VxFS ファイルシステムを定常状態にするため、通常、一時使用ボリュームの VxFS ファイルシステムに対して fsck を実行する必要はありません。VxFS ファイルシステムにデータベースが含まれている場合は、データベースのログ再生を引き続き実行する必要があります。

4 スナップショットのデータをバックアップするには、適切なユーティリティまたはオペレーティングシステムのコマンドを使って、スナップショットの内容をテープまたはその他のバックアップメディアにコピーします。

5 ここで、次の方法を選択できます。

- スナップショットの内容を更新します。この方法を使うと、元のボリュームの新しいポイントインタイムイメージを作成し、別のバックアップに使うことができます。ス

ナップショットで同期化がすでに進行中の場合は、この操作によって、スナップショットの大部分の再同期化が必要になることがあります。
p.715 の「インスタント領域最適化スナップショットの更新」を参照してください。

- スナップショットボリュームのプレックスの一部または全体を元のボリュームに再接続します。
p.715 の「フルサイズインスタントスナップショットまたはプレックスのブレークオフスナップショットの再接続」を参照してください。
- スナップショットボリュームから元のボリュームの内容を復元します。結果の動作として、元のボリュームにスナップショットボリュームのプレックスを戻さない、サブセットのみ戻す、すべてを戻すのいずれかを選択できます。
p.717 の「領域最適化インスタントスナップショットからのボリュームのリストア」を参照してください。
- スナップショットボリュームと元のボリュームの関連付けを完全に解除します。この方法は、テストやレポート生成など、他の用途にコピーを使う場合に便利です。必要に応じて、関連付けを解除したボリュームを削除することができます。
p.717 の「インスタントスナップショットの関連付けの解除」を参照してください。
- スナップショットがスナップショット階層の一部である場合は、この階層と親ボリュームの関連付けを解除することができます。
p.718 の「インスタントスナップショット階層の分割」を参照してください。

リンクされたブレークオフスナップショットボリュームの作成と管理

リンクされたブレークオフスナップショットは、書き込みを集中的に行うボリュームに適しています。特に、スナップショットを別のディスクグループから開始することで、ディスクグループの分割および結合の操作を回避できる可能性があるため、ブレークオフスナップショットはオフホスト処理に使われます。

リンクされたブレークオフスナップショットの場合は、スナップショットボリュームとして使うボリュームを準備する必要があります。このボリュームは、スナップショットを作成するボリュームと同じサイズにし、領域のサイズも同じにする必要があります。

p.698 の「フルサイズインスタントスナップショットまたはリンクブレークオフスナップショットに使うボリュームの作成」を参照してください。

スナップショットの属性は、一組にまとめて `vxsnap make` コマンドに指定します。このコマンドでは複数の組を指定できます。作成するスナップショットごとに組を 1 つ指定する必要があります。組の各要素は、スラッシュ文字(/)で区切ります。複数の組はスペースで区切ります。

リンクされたブレークオフスナップショットを作成して管理するには、次の手順を実行します。

1 次のコマンドを使って、準備したスナップショットボリューム *snapvol* をデータボリュームにリンクします。

```
# vxsnap [-g diskgroup] [-b] addmir volume mirvol=snapvol ¥
  [mirdg=snapdg]
```

省略可能な mirdg 属性を使って、スナップショットボリュームの現在のディスクグループ *snapdg* を指定できます。-b オプションを使うと、同期をバックグラウンドで実行できます。-b オプションを指定しない場合、このコマンドはリンクが ACTIVE になるまで戻りません。

たとえば、次のコマンドは、ディスクグループ mysnapdg 内の準備したボリューム prepsnap をディスクグループ mydg 内のボリューム vol1 にリンクします。

```
# vxsnap -g mydg -b addmir vol1 mirvol=prepsnap mirdg=mysnapdg
```

-b オプションを指定する場合は、次の例のように vxsnap snapwait コマンドを使って、リンクされたスナップショットボリュームの同期が完了するのを待機できます。

```
# vxsnap -g mydg snapwait vol1 mirvol=prepsnap mirdg=mysnapvoldg
```

2 リンクされたブレークオフスナップショットを作成するには、次の形式の vxsnap make コマンドを使います。

```
# vxsnap [-g diskgroup] make   source=volume/snapvol=snapvol¥
[/snapdg=snapdiskgroup]
```

スナップショットボリュームのディスクグループがデータボリュームのディスクグループと異なる場合は、snapdg 属性を使用してスナップショットボリュームのディスクグループを指定する必要があります。

たとえば、準備したボリューム prepsnap を、ボリューム vol1 のスナップショットとしてディスクグループ mydg で使うには、次のコマンドを使います。

```
# vxsnap -g mydg make source=vol1/snapvol=prepsnap/snapdg=mysnapdg
```

3 VxVM 以外のファイルシステムやデータベースのログ再生の場合は、fsck などの適切なユーティリティを使って、一時使用ボリュームの内容をクリーンにします。VxVM はスナップショットを作成する直前に VxFS を呼び出し、VxFS ファイルシステムを定常状態にするため、通常、一時使用ボリュームの VxFS ファイルシステムに対して fsck を実行する必要はありません。VxFS ファイルシステムにデータベースが含まれている場合は、データベースのログ再生を引き続き実行する必要があります。

4 スナップショットのデータをバックアップするには、適切なユーティリティまたはオペレーティングシステムのコマンドを使って、スナップショットの内容をテープまたはその他のバックアップメディアにコピーします。

5 ここで、次の方法を選択できます。

- スナップショットの内容を更新します。この方法を使うと、元のボリュームの新しいポイントインタイムイメージを作成し、別のバックアップに使うことができます。スナップショットで同期化がすでに進行中の場合は、この操作によって、スナップショットの大部分の再同期化が必要になることがあります。
  p.715 の「インスタント領域最適化スナップショットの更新」を参照してください。
- スナップショットボリュームを元のボリュームに再接続します。
  p.716 の「リンクされたブレークオフスナップショットボリュームの再接続」を参照してください。
- スナップショットボリュームと元のボリュームの関連付けを完全に解除します。この方法は、テストやレポート生成など、他の用途にコピーを使う場合に便利です。必要に応じて、関連付けを解除したボリュームを削除することができます。
  p.717 の「インスタントスナップショットの関連付けの解除」を参照してください。
- スナップショットがスナップショット階層の一部である場合は、この階層と親ボリュームの関連付けを解除することができます。
  p.718 の「インスタントスナップショット階層の分割」を参照してください。

複数のインスタントスナップショットの作成

一貫性のあるグループを形成するすべてのボリュームに対して、複数のインスタントスナップショットを作成できます。vxsnap make コマンドでは、複数のスナップショットおよびボリューム名を引数として指定できます。たとえば、指定したストレージから、それぞれ同様の冗長性を備えた 3 つのインスタントスナップショットを作成するには、次のコマンド形式を使います。

```
# vxsnap [-g diskgroup] make source=vol1/snapvol=snapvol1 ¥
source=vol2/snapvol=snapvol2 source=vol3/snapvol=snapvol3
```

スナップショットボリューム（*snapvol1*、*snapvol2* など）は、事前に準備しておく必要があります。

p.698 の「フルサイズインスタントスナップショットまたはリンクブレークオフスナップショットに使うボリュームの作成」を参照してください。

指定するソースボリューム（*vol1*、*vol2* など）は、同じボリュームでも異なるボリュームでもかまいません。

すべてのスナップショットを領域最適化スナップショットとし、さらに同じキャッシュを共有する場合は、次のコマンド形式を使います。

```
# vxsnap [-g diskgroup] make ¥
source=vol1/newvol=snapvol1/cache=cacheobj ¥
source=vol2/newvol=snapvol2/cache=cacheobj ¥
source=vol3/newvol=snapvol3/cache=cacheobj ¥
  [alloc=storage_attributes]
```

次に示すように、vxsnap make コマンドを使うと、タイプや冗長性が異なり、かつ異なったストレージから設定されたスナップショットを作成できます。

```
# vxsnap [-g diskgroup] make source=vol1/snapvol=snapvol1 ¥
source=vol2[/newvol=snapvol2]/cache=cacheobj¥
[/alloc=storage_attributes2][/nmirror=number2]
source=vol3[/newvol=snapvol3][/alloc=storage_attributes3]¥
/nmirror=number3
```

この例では、*snapvol1* は事前に準備したボリュームを使うフルサイズのスナップショット、*snapvol2* は事前に準備したキャッシュを使う領域最適化スナップショット、*snapvol3* はもとのボリュームのプレックスから形成されたフルサイズのブレークオフスナップショットです。

同時に複数のタイプのスナップショットを作成する例として、データベース REDO ログとデータベーステーブルを含むボリュームのスナップショットを作成する例を次に示します。

```
# vxsnap -g mydg make ¥
source=logv1/newvol=snplogv1/drl=sequential/nmirror=1 ¥
source=logv2/newvol=snplogv2/drl=sequential/nmirror=1 ¥
source=datav1/newvol=snpdatav1/cache=mydgcobj/drl=on ¥
source=datav2/newvol=snpdatav2/cache=mydgcobj/drl=on
```

この例では、REDO ログボリュームのスナップショットに対してはシーケンシャル DRL を有効にし、データベーステーブルを含むボリュームのスナップショットに対しては通常の DRL を適用しています。2 つの領域最適化スナップショットは、ディスクグループ内の同一のキャッシュオブジェクトを共有するように設定されています。また、書き込みが集中的に行われる REDO ログのボリュームに対してブレークオフスナップショットを使っています。

ボリュームセットのインスタントスナップショットの作成

ボリュームセット名は、インスタントスナップショットでの addmir、dis、make、prepare、reattach、refresh、restore、rmmir、split、syncpause、syncresume、syncstart、syncstop、syncwait、unprepare などの vxsnap 操作で、ボリューム名の代わりに使えます。

ボリュームセットのインスタントスナップショットの作成手順は、スタンドアロンボリュームの場合と同じです。ただし、フルサイズインスタントスナップショットを事前に準備したボリュームセットから作成する場合は、一定の制限があります。ボリュームセットのフルサイズイン

スタントスナップショットは、それ自体が、親と同じ数のボリューム、同じサイズのボリューム、および同じボリュームインデックス番号を持つボリュームセットである必要があります。たとえば、ボリュームセットに 3 つのボリュームが存在し、それぞれ、サイズが 1 GB、2 GB および 3 GB、インデックスが 0、1 および 2 である場合は、スナップショットボリュームセットに、同じインデックス番号セットと一致する、同じサイズの 3 つのボリュームが存在する必要があります。親ボリュームセットおよびスナップショットボリュームセット内の対応するボリュームも、スタンドアロンボリュームとそのスナップショットとの間に適用される制限と同じ制限を受けます。

vxvset list コマンドを使うと、ボリュームセットが、次の例に示すように、同じ特性を持つことを確認できます。

```
# vxvset -g mydg list vset1

VOLUME        INDEX      LENGTH      KSTATE      CONTEXT
vol_0         0          204800      ENABLED     -
vol_1         1          409600      ENABLED     -
vol_2         2          614400      ENABLED     -

# vxvset -g mydg list snapvset1

VOLUME        INDEX      LENGTH      KSTATE      CONTEXT
svol_0        0          204800      ENABLED     -
svol_1        1          409600      ENABLED     -
svol_2        2          614400      ENABLED     -
```

ボリュームセットのフルサイズインスタントスナップショットは、事前に準備したボリュームセットを使って作成できます。準備したボリュームセットの各ボリュームは、親ボリュームセット内の対応するボリュームと同じサイズになります。また、nmirrors 属性を使うと、ボリュームセット内の各ボリュームにプレックスが十分に存在する場合に切り離すプレックスの数を指定できます。

次の例に、ソースボリュームセット vset1 と、後でスナップショットの作成に使う、同一のボリュームセット snapvset1 を準備する方法を示します。

```
# vxsnap -g mydg prepare vset1
# vxsnap -g mydg prepare snapvset1
# vxsnap -g mydg make source=vset1/snapvol=snapvset1
```

フルサイズサードミラーブレークオフスナップショットを作成するには、ソースボリュームセット内の各ボリュームにプレックスが十分に存在することを確認する必要があります。次の例に、vxsnap コマンドを使って、スナップショットを切り離す前に必要な数のプレックスを追加することにより、十分な数のプレックスを確保する方法を示します。

```
# vxsnap -g mydg prepare vset2
# vxsnap -g mydg addmir vset2 nmirror=1
# vxsnap -g mydg make source=vset2/newvol=snapvset2/nmirror=1
```

p.713 の「ボリュームへのスナップショットミラーの追加」を参照してください。

ボリュームセットの領域最適化インスタントスナップショットを作成するときに使うコマンドも、次の例に示すように、スタンドアロンボリュームの場合と同じです。

```
# vxsnap -g mydg prepare vset3
# vxsnap -g mydg make source=vset3/newvol=snapvset3/cachesize=20m

# vxsnap -g mydg prepare vset4
# vxsnap -g mydg make source=vset4/newvol=snapvset4/cache=mycobj
```

ここでは、ボリュームセット vset3 の新しいキャッシュオブジェクトが作成され、既存のキャッシュオブジェクト mycobj が vset4 に使われます。

ボリュームへのスナップショットミラーの追加

フルサイズブレークオフスナップショットボリュームを作成する場合は、次のコマンドを使ってボリュームに新しいスナップショットミラーを追加します。

```
# vxsnap [-b] [-g diskgroup] addmir volume|volume_set ¥
[nmirror=N] [alloc=storage_attributes]
```

ボリュームは、vxsnap prepare コマンドを使って準備しておく必要があります。

ボリュームの代わりにボリュームセット名が指定された場合は、指定した数のプレックスがボリュームセット内の各ボリュームに追加されます。

nmirror 属性を使ってミラーの数の変更を指定しない限り、vxsnap addmir コマンドはデフォルトで 1 つのスナップショットミラーをボリュームに追加します。ミラーは完全に同期されるまで SNAPATT 状態のままです。-b オプションを使うと、同期をバックグラウンドで実行できます。同期が完了すると、ミラーは SNAPDONE 状態になります。

たとえば、ディスク mydg10 および mydg11 上のボリューム vol1 に 2 つのミラーを追加するには、次のコマンドを実行します。

```
# vxsnap -g mydg addmir vol1 nmirror=2 alloc=mydg10,mydg11
```

このコマンドは、vxassist snapstart コマンドと使用方法が似ており、従来のサードミラーブレークオフスナップショットモデルをサポートしています。したがって、インスタントスナップショット機能は提供されません。

いったん 1 つ以上のスナップショットミラーをボリュームに追加すると、nmirror 属性または plex 属性のいずれかを指定して vxsnap make コマンドを使い、スナップショットボリュームを作成できます。

スナップショットミラーの削除

ボリュームから単一のスナップショットミラーを削除するには、次のコマンドを使います。

```
# vxsnap [-g diskgroup] rmmir volume|volume_set
```

たとえば、ボリューム vol1 からスナップショットミラーを削除するには、次のコマンドを実行します。

```
# vxsnap -g mydg rmmir vol1
```

このコマンドは、vxassist snapabort コマンドと使用方法が似ています。

ボリュームの代わりにボリュームセット名が指定された場合は、ボリュームセット内の各ボリュームからミラーが削除されます。

リンクされたブレークオフスナップショットボリュームの削除

ボリュームからリンクされたブレークオフスナップショットボリュームを削除するには、次のコマンドを使います。

```
# vxsnap [-g diskgroup] rmmir volume|volume_set mirvol=snapvol ¥
[mirdg=snapdiskgroup]
```

mirvol と省略可能な mirdg 属性は、スナップショットボリューム *snapvol* とそのディスクグループ *snapdiskgroup* を指定しています。たとえば、次のコマンドは、リンクされたスナップショットボリューム prepsnap をボリューム vol1 から削除します。

```
# vxsnap -g mydg rmmir vol1 mirvol=prepsnap mirdg=mysnapdg
```

カスケードスナップショット階層へのスナップショットの追加

スナップショットを作成し、それをもとのボリュームと既存のスナップショットボリュームで形成されているスナップショット階層内に配置するには、既存のスナップショットボリューム名を vxsnap make コマンドの infrontof 属性の値として指定します。次の例は、ボリューム dbvol の領域最適化スナップショット thurs_bu を既存のスナップショット wed_bu フロントボリュームとして配置する方法を示しています。

```
# vxsnap -g dbdg make source=dbvol/newvol=thurs_bu/¥
infrontof=wed_bu/cache=dbdgcache
```

同様に、次に作成されるスナップショット fri_bu も、thurs_bu の前に配置されます。

```
# vxsnap -g dbdg make source=dbvol/newvol=fri_bu/¥
infrontof=thurs_bu/cache=dbdgcache
```

p.721 の「インスタントスナップショットの同期の制御」を参照してください。

インスタント領域最適化スナップショットの更新

インスタントスナップショットを更新すると、そのスナップショットは親ボリュームのポイントインタイムコピーに置き換えられます。1 つ以上のスナップショットを更新し、すぐに使えるようにするには、次のコマンドを使います。

```
# vxsnap [-g diskgroup] refresh snapvolume|snapvolume_set ¥
  [source=volume|volume_set] [snapvol2 [source=vol2]...] ¥
```

ソースボリュームが指定されていない場合は、スナップショットの直属の親が使われます。

警告: 更新処理中のスナップショットを別のアプリケーションで開くことはできません。たとえば、ボリューム上に設定されているファイルシステムはあらかじめマウント解除する必要があります。

フルサイズインスタントスナップショットまたはプレックスのブレークオフスナップショットの再接続

インスタントスナップショットのプレックスの一部または全体を、指定したもとのボリュームまたはスナップショット階層内のそのスナップショットボリュームの上位にあるソースボリュームに再接続するには、次のコマンドを使います。

```
# vxsnap [-g diskgroup] reattach snapvolume|snapvolume_set ¥
  source=volume|volume_set [nmirror=number]
```

デフォルトでは、すべてのプレックスが再接続され、その結果スナップショットが削除されます。再接続するプレックスの数は、必要に応じて、nmirror 属性の値として指定できます。

警告: 再接続処理中のスナップショットを別のアプリケーションで開くことはできません。たとえば、スナップショットボリューム上に設定されているファイルシステムはあらかじめマウント解除する必要があります。

ボリュームは、ボリュームサイズと領域のサイズに互換性がある場合は無関係なボリュームにも再接続できます。

たとえば、スナップショットボリューム snapmyvol の 1 つのプレックスをボリューム myvol に再接続するには、次のコマンドを使います。

```
# vxsnap -g mydg reattach snapmyvol source=myvol nmirror=1
```

再接続されたプレックスは、親ボリュームのデータから同期されている間、SNAPTMP 状態のままとなります。再同期が完了すると、プレックスは SNAPDONE 状態になります。次に示

すように、vxsnap snapwait コマンド(vxsnap syncwait コマンドではない)を使うと、再接続されたプレックスの再同期の完了まで待機できます。

```
# vxsnap -g mydg snapwait myvol nmirror=1
```

ボリュームとそのスナップショットを両方とも同じだけサイズ変更(拡張または縮小)した後に再接続した場合、引き続き高速再同期を実行できます。ボリューム全体を再同期する必要はありません。インスタントスナップの DCO ボリュームの場合、対応するデータボリュームがサイズ変更されると、それに比例してサイズ変更されます。バージョン 0 の DCO ボリュームの場合、FastResync マップは同じサイズのままですが、領域のサイズが再計算され、既存マップ内のダーティビットの位置が調整されます。どちらのバージョンの場合も、新しい領域はマップ内でダーティとしてマークされます。

リンクされたブレークオフスナップショットボリュームの再接続

他のタイプのスナップショットとは異なり、リンクされたブレークオフスナップショットボリュームの再接続操作は、スナップショットボリュームのプレックスを親ボリュームに返しません。スナップショットボリュームを親ボリュームのミラーにするリンク関係が再確立され、これにより、スナップショットデータを再同期できます。

リンクされたブレークオフスナップショットを再接続するには、次の形式の vxsnap reattach コマンドを使います。

```
# vxsnap [-g snapdiskgroup] reattach snapvolume|snapvolume_set ¥
source=volume|volume_set [sourcedg=diskgroup]
```

データボリュームのディスクグループが、スナップショットボリュームのディスクグループである snapdiskgroup と異なる場合、***sourcedg*** 属性を使ってデータボリュームのディスクグループを指定する必要があります。

警告: 再接続処理中のスナップショットを別のアプリケーションで開くことはできません。たとえば、スナップショットボリューム上に設定されているファイルシステムはあらかじめマウント解除する必要があります。

ボリュームは、サイズと領域のサイズに互換性がある場合は無関係なボリュームにも再接続できます。

たとえば、次のコマンドは、ディスクグループ snapdg 内のスナップショットボリューム prepsnap をディスクグループ mydg 内のボリューム myvol に再接続します。

```
# vxsnap -g snapdg reattach prepsnap source=myvol sourcedg=mydg
```

スナップショットボリュームの再同期が完了すると、リンクは ACTIVE 状態になります。次に示すように、vxsnap snapwait コマンド(vxsnap syncwait コマンドではない)を使うと、再接続されたボリュームの再同期の完了まで待機できます。

```
# vxsnap -g snapdg snapwait myvol mirvol=prepsnap
```

領域最適化インスタントスナップショットからのボリュームのリストア

場合によっては、バックアップからまたはスナップショットボリューム内の複製から変更されたボリュームの内容を復元する必要があります。指定したスナップショットから 1 つまたは複数のボリュームを復元するには、次のコマンドを使います。

```
# vxsnap [-g diskgroup] restore volume|volume_set ¥
  source=snapvolume|snapvolume_set ¥
  [[volume2|volume_set2 ¥
  source=snapvolume2|snapvolume_set2]...]¥
 [syncing=yes|no]
```

領域最適化インスタントスナップショットの場合は、キャッシュデータを使って、指定したボリュームの内容が再作成されます。`restore` 操作では、領域最適化インスタントスナップショットを変更することはできません。

警告: 復元するボリュームとスナップショットボリュームが別のアプリケーションで開かれている場合は、この操作を正常に実行することはできません。たとえば、いずれかのボリューム上に設定されているファイルシステムは前もってマウント解除する必要があります。

ボリュームは、無関係なボリュームからは復元できません。

次の例は、領域最適化スナップショット `snap3myvol` からボリューム `myvol` を復元する方法を示しています。

```
# vxsnap -g mydg restore myvol source=snap3myvol
```

インスタントスナップショットの関連付けの解除

次のコマンドを実行すると、フルサイズインスタントスナップショットボリューム *snapvol* とその親ボリュームの関連付けが解除され、スナップショットを独立したボリュームとして使えるようになります。

```
# vxsnap [-f] [-g diskgroup] dis snapvolume|snapvolume_set
```

スナップショット *snapvol* の下位に同期していないスナップショットが存在する場合、この操作は失敗します。その場合、依存している下位のスナップショットを、*snapvol* と完全に同期する必要があります。依存するスナップショットがなくなると、*snapvol* の関連付けを解除できます。スナップショット階層は *snapvol* の親ボリュームに取り込まれます。

p.721 の「インスタントスナップショットの同期の制御」を参照してください。

p.718 の「インスタントスナップショットの削除」を参照してください。

次のコマンドを実行すると、スナップショット snap2myvol とその親ボリュームの関連付けが解除されます。

```
# vxsnap -g mydg dis snap2myvol
```

警告: この操作をボリュームセット、またはボリュームセットのコンポーネントボリュームに適用すると、システムクラッシュやハードウェア障害が発生した場合に、スナップショット内に不整合が生じる可能性があります。この操作をボリュームセットに適用する場合は、-f (force) オプションを指定する必要があります。

インスタントスナップショットの削除

フルサイズインスタントスナップショットの関連付けを解除すると、次の例に示すように vxedit コマンドを使ってこのインスタントスナップショットを完全に削除できます。

```
# vxedit -g mydg -r rm snap2myvol
```

このコマンドでは、領域最適化インスタントスナップショットをそのキャッシュから削除することもできます。

p.725 の「キャッシュの削除」を参照してください。

インスタントスナップショット階層の分割

メモ: この操作は、領域最適化インスタントスナップショットではサポートされていません。

次のコマンドを実行すると、スナップショットボリューム *snapvol* 以下のスナップショット階層とその親ボリュームとの関連付けが解除され、スナップショット階層を親ボリュームから独立して使えるようになります。

```
# vxsnap [-f] [-g diskgroup] split snapvolume|snapvolume_set
```

このコマンドを正常に実行するためには、関連付けを解除するスナップショット階層内の最上位スナップショットボリュームが完全に同期している必要があります。その階層内の下位にあるスナップショットは、完全に再同期化されていなくてもかまいません。

p.721 の「インスタントスナップショットの同期の制御」を参照してください。

次のコマンドを実行すると、snap2myvol の下位にある階層とその親ボリュームの関連付けが解除されます。

```
# vxsnap -g mydg split snap2myvol
```

警告: この操作をボリュームセット、またはボリュームセットのコンポーネントボリュームに適用すると、システムクラッシュやハードウェア障害が発生した場合に、スナップショット内に不整合が生じる可能性があります。この操作をボリュームセットに適用する場合は、-f (force) オプションを指定する必要があります。

インスタントスナップショット情報の表示

vxsnap print コマンドを使うと、ボリュームに関連付けられているスナップショットに関する情報を表示できます。

```
# vxsnap [-g diskgroup] print [vol]
```

このコマンドを実行すると、スナップショットまたはボリュームの同期の進行状況が割合で表示されます。ボリュームを指定しなかった場合は、ディスクグループ内のすべてのボリュームのスナップショットに関する情報が表示されます。次の例のボリューム vol1 にはフルサイズのスナップショット snapvol1 が格納されていますが、このスナップショットの内容はボリューム vol1 とは同期されていません。

```
# vxsnap -g mydg print
NAME      SNAPOBJECT      TYPE    PARENT   SNAPSHOT   %DIRTY   %VALID

vol1      --              volume  --       --         --       100
          snapvol1_snp1   volume  --       snapvol1   1.30     --
snapvol1  vol1_snp1       volume  vol1     --         1.30     1.30
```

snapvol1 の %DIRTY の値は、このボリュームと vol1 の内容の差異が 1.30 % であることを示しています。snapvol1 は vol1 と同期していないため、%VALID の値と %DIRTY の値は同じになります。スナップショットの一部が同期している場合は、%VALID の値は %DIRTY の値と 100 % の間の値になります。スナップショットが完全に同期している場合は、%VALID の値は 100 % になります。このスナップショットは、独立したスナップショットにしたり、別のディスクグループに移動することができます。

vxsnap print コマンドで -n オプションを使うと、ボリュームやボリュームセットのスナップショットに関する追加情報を取得できます。

```
# vxsnap [-g diskgroup] -n [-l] [-v] [-x] print [vol]
```

また、vxsnap -n print コマンドのエイリアスである vxsnap list コマンドを使うこともできます。

```
# vxsnap [-g diskgroup] [-l] [-v] [-x] list [vol]
```

次の出力は、ディスクグループ dg1 でこのコマンドを使用した例です。

```
# vxsnap -g dg -vx list
```

```
NAME    DG    OBJTYPE SNAPTYPE   PARENT   PARENTDG   SNAPDATE         CHANGE_DATA
SYNCED_DATA
vol     dg1   vol     -          -        -          -         -                    10G
(100%)
svol1   dg2   vol     fullinst   vol      dg1        2006/2/1 12:29   20M (0.2%)     60M
(0.6%)
svol2   dg1   vol     mirbrk     vol      dg1        2006/2/1 12:29   120M (1.2%)    10G
(100%)
svol3   dg2   vol     volbrk     vol      dg1        2006/2/1 12:29   105M (1.1%)    10G
(100%)
svol21  dg1   vol     spaceopt   svol2    dg1        2006/2/1 12:29   52M (0.5%)     52M
(0.5%)
vol-02  dg1   plex    snapmir    vol      dg1        -                -              56M
(0.6%)
mvol    dg2   vol     mirvol     vol      dg1        -                -              58M
(0.6%)
vset1   dg1   vset    -      -   -        -          -                               2G (100%)
v1      dg1   compvol -      -   -        -          -                               1G (100%)
v2      dg1   compvol -      -   -        -          -                               1G (100%)
svset1  dg1   vset    mirbrk     vset     dg1        2006/2/1 12:29   1G (50%)       2G (100%)
sv1     dg1   compvol mirbrk     v1       dg1        2006/2/1 12:29   512M (50%)     1G (100%)
sv2     dg1   compvol mirbrk     v2       dg1        2006/2/1 12:29   512M (50%)     1G (100%)
vol-03  dg1   plex    detmir     vol      dg1        -                20M (0.2%)     -
mvol2   dg2   vol     detvol     vol      dg1        -                20M (0.2%)     -
```

これは、ボリューム vol に 3 つのフルサイズ スナップショット(svol1、svol2 、svol3)があり、これらのタイプがフルサイズインスタント(fullinst)、ミラーブレークオフ(mirbrk)、リンクされたブレークオフ(volbrk)であることを示しています。さらに、このボリュームには 1 つのスナップショットプレックス(snapmir)、vol-02 と、1 つのリンクされたミラーボリューム(mirvol)、mvol があります。スナップショット svol2 自体には領域最適化インスタントスナップショット(spaceopt)、svol21 があります。さらに、コンポーネントボリューム v1 と v2 を含むボリュームセット vset1 もあります。このボリュームセットには、コンポーネントボリューム sv1 と sv2 を含むミラーブレークオフスナップショット svset1 があります。最後の 2 つのエントリは、親ボリュームとして vol を持つ、切断されたプレックス vol-03 と切断されたミラーボリューム mvol2 を示しています。これらのスナップショット オブジェクトは、I/O エラーのために、またはプレックスの場合は vxplex det コマンドの実行のために切断された可能性があります。

CHANGE_DATA 列には、スナップショットの現在の内容とその親ボリュームのおおよその違いが表示されます。これは、コンテンツを再び同一にするために再同期する必要があるデータの量に対応します。

SYNCED_DATA 列には、スナップショットの作成後の同期のおおよその進行状況が表示されます。

-l オプションを使って、表形式ではなくより長い形式の出力を一覧表示できます。

-x オプションは、ボリュームセットのコンポーネントボリュームも含まれるように出力を展開します。

vxsnap print コマンドと vxsnap list コマンドの使用について詳しくは、vxsnap(1M)マニュアルページを参照してください。

インスタントスナップショットの同期の制御

領域最適化インスタントスナップショットでは、スナップショットの内容ともとのボリュームとの同期は実行できません。

デフォルトでは、インスタントスナップショットでの vxsnap reattach、refresh、restore 操作に対する同期が有効になっています。それ以外の場合は、vxsnap コマンドに syncing=yes 属性を指定しない限り、同期は行われません。

「表 26-1」に、同期を手動で制御するためのコマンドを示します。

表 26-1　インスタントスナップショットの同期を制御するコマンド

コマンド	説明
vxsnap [-g *diskgroup*] syncpause ¥ *vol*|*vol_set*	ボリュームの同期を一時停止します。
vxsnap [-g *diskgroup*] syncresume ¥ *vol*|*vol_set*	ボリュームの同期を再開します。
vxsnap [-b] [-g *diskgroup*] syncstart ¥ *vol*|*vol_set*	ボリュームの同期を開始します。-b オプションを使うと、この操作がバックグラウンドで実行されます。
vxsnap [-g *diskgroup*] syncstop ¥ *vol*|*vol_set*	ボリュームの同期を停止します。
vxsnap [-g *diskgroup*] syncwait ¥ *vol*|*vol_set*	ボリュームの同期が完了すると終了します。*vol* や *vol_set* が無効の場合(たとえば、領域最適化スナップショットなど)、または *vol* や *vol_set* が同期化されていない場合、エラーが返されます。 **メモ:** 再接続されたプレックスの場合、このコマンドを使って同期の完了を待機することはできません。

「表 26-1」で示すコマンドを使っても、リンクされたブレークオフスナップショットの同期を制御できません。

再同期の後に新しいリンクされたブレークオフスナップショットの間のリンクが ACTIVE になるか、再接続されたスナップショットプレックスが SNAPDONE 状態になるまで待機する vxsnap snapwait コマンドが用意されています。

p.708 の「リンクされたブレークオフスナップショットボリュームの作成と管理」を参照してください。

p.715 の「フルサイズインスタントスナップショットまたはプレックスのブレークオフスナップショットの再接続」を参照してください。

p.716 の「リンクされたブレークオフスナップショットボリュームの再接続」を参照してください。

スナップショットの同期処理のパフォーマンスの向上

-o オプションには、make、refresh、restore、syncstart の各操作の使用時に同期処理のパフォーマンスを最適化できるように、次のオプションの引数が用意されています。

iosize=*size*	ボリュームの領域を同期する際に使う、各 I/O 要求のサイズを指定します。大きいサイズを指定するほど同期処理は速くなりますが、ボリュームにアクセスする他のプロセスの処理効率への影響が大きくなります。パフォーマンスの高いアレイおよびコントローラハードウェアに対しては、size の値をデフォルトの 1m(1 MB)以上の値にすることをお勧めします。指定した値は、ボリュームの領域のサイズの倍数に丸められます。
slow=*iodelay*	iosize の値で指定された領域を連続して同期するときの各同期処理間の遅延時間(ミリ秒)を指定します。この引数を使うと、同期によるシステム処理効率への影響を調整できます。*iodelay* のデフォルト値は、0 ミリ秒(遅延なし)です。この値を大きくすると同期処理の速度が低下し、ボリュームにアクセスする可能性がある他のプロセスとの I/O 帯域幅の競合が減少します。

オプションは、次の例のように組み合わせることができます。

```
# vxsnap -g mydg -o iosize=2m,slow=100 make ¥
  source=myvol/snapvol=snap2myvol/syncing=on

# vxsnap -g mydg -o iosize=10m,slow=250 syncstart snap2myvol
```

メモ: iosize パラメータと slow パラメータは、領域最適化インスタントスナップショットではサポートされていません。

キャッシュ上で作成したスナップショットの一覧表示

キャッシュオブジェクト上で作成した領域最適化インスタントスナップショットの一覧を表示するには、次のコマンドを使います。

```
# vxcache [-g diskgroup] listvol cache_object
```

スナップショットの名前が、タイムスタンプ順にカンマで区切って出力されます。タイムスタンプが同じスナップショットが複数存在する場合は、サイズでソートされ降順で表示されます。

キャッシュの autogrow 属性のチューニング

autogrow 機能が有効で、VxVM キャッシュデーモン(vxcached)が動作している場合、vxcached によるキャッシュ管理の動作は、highwatermark、autogrowby、maxautogrow の 3 つの属性によって決まります。

- キャッシュ使用率が高水準値 highwatermark(デフォルト値は 90 %)に達した場合、vxcached は autogrowby の値(デフォルトはキャッシュボリュームサイズの 20 % のブロック数)の分だけキャッシュボリュームのサイズを拡張します。ただし、新たに必要となるキャッシュサイズが maxautogrow の値(デフォルト値はキャッシュボリュームサイズの 2 倍のブロック数)を超える場合は拡張されません。
- キャッシュ使用率が高水準値に達し、新たに必要となるキャッシュサイズが maxautogrow の値を超える場合、vxcached はキャッシュ内の最も古いスナップショットを削除します。経過日数の同じスナップショットが複数存在する場合は、サイズが最大のものが削除されます。

autogrow 機能が無効な場合

- キャッシュ使用率が高水準値に達すると、vxcached はキャッシュ内の最も古いスナップショットを削除します。経過日数の同じスナップショットが複数存在する場合は、サイズが最大のものが削除されます。スナップショットが 1 つしかない場合は、そのスナップショットが切断されて無効に設定されます。

メモ: vxcached デーモンでは、現在起動していないスナップショットや、キャッシュ内の最後(唯一)のスナップショットは削除されません。

キャッシュ領域が完全に消費されると、スナップショットは切断されて無効に設定されます。この場合、切断されたスナップショットはリカバリ不能であり、削除する必要があります。autogrow 機能をキャッシュで有効にすると、この問題の発生を回避できます。ただし、キャッシュが数 MB 程度で非常に小さい場合には、システムによる自動拡張が間に合わず、キャッシュが完全に消費されてしまう可能性があります。このような場合、キャッシュのサイズを手動で減らすことができます。

または、この例にあるように、vxcache set コマンドを使って highwatermark の値を減らすことができます。

```
# vxcache -g mydg set highwatermark=60 cobjmydg
```

キャッシュ拡張後のサイズの上限は maxautogrow 属性で制限できます。このサイズを見積もるには、スナップショットの更新によってそれぞれのソースボリュームの内容がどの程度変化するかを考慮した上で、緊急に備えて予備の領域を持たせるようにします。

キャッシュに対応する他の autogrow 属性の値は、必要に応じて vxcache set コマンドで変更できます。

vxcache(1M)マニュアルページを参照してください。

キャッシュ使用率の監視と表示

vxcache stat コマンドを使ってキャッシュ使用率を表示できます。たとえば、どのくらいの領域が使われているかや、ディスクグループ mydg 内のすべてのキャッシュオブジェクトで利用できる領域がどのくらい残っているかを確認するには、次のように入力します。

```
# vxcache -g mydg stat
```

キャッシュの拡張と縮小

vxcache コマンドを使うと、キャッシュオブジェクトに関連付けられているキャッシュボリュームのサイズを拡張できます。

```
# vxcache [-g diskgroup] growcacheto cache_object
    size
```

たとえば、キャッシュオブジェクト mycache に関連付けられているキャッシュボリュームのサイズを 2 GB に拡張するには、次のコマンドを使います。

```
# vxcache -g mydg growcacheto mycache 2g
```

指定した分量だけキャッシュを拡張するには、次のコマンド形式を使います。

```
# vxcache [-g diskgroup] growcacheby cache_object
    size
```

mycache のサイズを 1 GB だけ拡張するには、次のコマンドを使います。

```
# vxcache -g mydg growcacheby mycache 1g
```

同様に、shrinkcacheby および shrinkcacheto の操作によってキャッシュのサイズを縮小できます。

vxcache(1M)マニュアルページを参照してください。

キャッシュの削除

キャッシュオブジェクトとそのキャッシュボリューム、さらに領域最適化スナップショットもすべて含めてキャッシュを完全に削除するには、次の手順を実行します。

1 次のコマンドを実行して、キャッシュオブジェクト上に設定されている最上位のスナップショットボリュームの名前を確認します。

```
# vxprint -g diskgroup -vne ¥
  "v_plex.pl_subdisk.sd_dm_name ~ /cache_object/"
```

ここで、*cache_object* はキャッシュオブジェクトの名前です。

2 最上位のスナップショットとその下位に属するスナップショットをすべて削除します(この処理は 1 つのコマンドで実行できます)。

```
# vxedit -g diskgroup -r rm snapvol ...
```

ここで、*snapvol* は最上位のスナップショットボリュームの名前です。

3 キャッシュオブジェクトを停止します。

```
# vxcache -g diskgroup stop cache_object
```

4 最後に、キャッシュオブジェクトとそのキャッシュボリュームを削除します。

```
# vxedit -g diskgroup -r rm cache_object
```

リンクされたブレークオフスナップショット

サードミラーブレークオフスナップショットの一種に、vxsnap addmir コマンドを使って、データボリュームを含む特別に準備されたボリュームをリンクするリンクブレークオフスナップショットがあります。スナップショットに使われるボリュームは、フルサイズインスタントスナップショットと同じ方法で準備されます。ただし、フルサイズインスタントスナップショットとは違い、このボリュームは、データボリュームとは異なるディスクボリューム内に設定できます。このため、リンクされたブレークオフスナップショットは、ディスクグループの分割や結合の管理手順が回避されることから、特に繰り返しのオフホスト処理アプリケーションに適しています。サードミラーブレークオフスナップショットと同様に、スナップショットボリュームの内容がデータボリュームと同期されるまで待ってから vxsnap make コマンドを使ってスナップショットを作成する必要があります。

ボリュームとスナップショットになるミラーの間にリンクが作成された場合、スナップオブジェクトに似た別々のリンクオブジェクトがボリュームとそのミラーに関連付けられます。もとのボリュームのリンクオブジェクトはミラーボリュームを指し、ミラーボリュームのリンクオブジェクトはもとのボリュームを指します。すべての I/O は、もとのボリュームとミラーの両方に対して実行され、もとのボリューム内のデータからミラーへの同期が開始されます。

vxprint コマンドを使ってリンクオブジェクトの状態を表示できます。これはタイプ ln として表示されます。リンクオブジェクトには次のような状態があります。

ACTIVE	ミラーボリュームがもとのボリュームから完全に同期化されています。vxsnap make コマンドを実行してスナップショットを作成できます。
ATTACHING	ミラーボリュームの同期が進行中です。状態が ACTIVE に変わるまでは、vxsnap make コマンドを使ってスナップショットを作成できません。vxsnap snapwait コマンドを使って同期の完了を待機できます。
BROKEN	I/O エラーまたはミラーボリュームの拡大の失敗のために、ミラーボリュームがもとのボリュームから切断されています。vxrecover コマンドを使って、DISABLED ボリュームと同じ方法でミラーボリュームを回復できます。

ボリュームのサイズを変更 (拡大または縮小) すると、そのボリュームのすべての ACTIVE なリンクミラーボリュームのサイズも同時に変更されます。ボリュームとそのミラーは、同じディスクグループ内にも異なるディスクグループ内にも置けます。操作が正常に終了した場合は、ボリュームとそのミラーが同じサイズになります。

ボリュームを拡大した場合、リンクミラーボリューム内の拡大された領域の再同期が開始され、再同期が完了するまでリンクは ATTACHING 状態のままになります。vxsnap snapwait コマンドを使って、状態が ACTIVE. になるのを待機できます。

vxsnap make コマンドを使ってスナップショットボリュームを作成すると、リンクが削除され、スナップショットボリュームと元のボリュームの間にスナップショット関係が確立されます。

vxsnap reattach 操作は、2 つのボリュームの間にリンク関係を再び確立し、ミラーボリュームの再同期を開始します。

p.708 の「リンクされたブレークオフスナップショットボリュームの作成と管理」を参照してください。

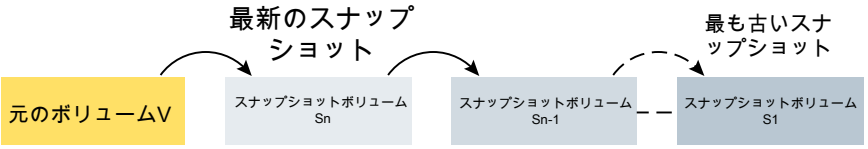
リンクブレークオフスナップショットで使う空のボリュームを準備する必要があります。

p.698 の「フルサイズインスタントスナップショットまたはリンクブレークオフスナップショットに使うボリュームの作成」を参照してください。

カスケードスナップショット

図 26-3 は、スナップショットカスケードと呼ばれるスナップショット階層を示しています。一部のアプリケーションでは、このスナップショット階層によって書き込みパフォーマンスを向上できます。

図 26-3 スナップショットカスケード



ボリュームから複数の独立したスナップショットを作成するよりも、古いスナップショットを最新のスナップショットの子オブジェクトにするほうが効率的です。

スナップショットカスケードは、ボリュームの定期的なオンラインバックアップに多く使われます。この場合、領域最適化スナップショットは、テープではなくディスクに書き込まれます。

スナップショットカスケードを使うと、複数の独立したスナップショットを作成するよりも書き込みの処理効率が向上します。また、スナップショットを領域最適化すれば、必要なディスク領域も少なくて済みます。もとのボリュームが変更されたときに更新する必要があるのは、最新のスナップショットのみです。必要に応じて、最新のスナップショットから古いスナップショットに変更内容を取り込むことができます。

スナップショットカスケードに新規スナップショットを追加する場合は、2 番目のスナップショット以降では `infrontof` 属性を直前のスナップショットに指定して `vxsnap make` コマンドを実行します。もとのボリュームのブロックに対する変更は、カスケード内で作成された最新のスナップショットボリュームにのみ書き込まれます。古いスナップショットからデータの読み取りを実行して、該当するデータがこのスナップショットにない場合には、1 つ前の新しいスナップショットから階層を上に向かって再帰的に検索し、該当データを取得します。

次の点を参考に、スナップショットカスケードを使うことが適切かどうかを決定します。

- カスケード内のスナップショットを削除する際には、スナップショットのデータをカスケード内の次のスナップショットにコピーする必要があります。このためスナップショットの削除には時間がかかります。
- カスケード内のスナップショットの信頼性は、そのスナップショットの前にある一連の新しいスナップショットによって決まります。したがって、カスケード内で最も信頼性が低いのは、最も古いスナップショットです。
- カスケードのスナップショットからデータを読み取るには、カスケード内の他の 1 つ以上のスナップショットからデータをフェッチする必要があります。

このため、カスケード内からスナップショットを削除または分割する必要があるアプリケーションには、スナップショットカスケードを使わないことをお勧めします。このようなアプリケーションには、次の項の説明に従って、スナップショットのスナップショットを作成する方が適しています。

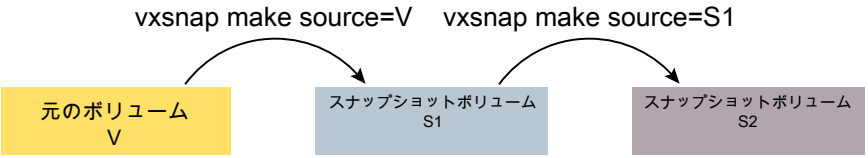
p.714 の「カスケードスナップショット階層へのスナップショットの追加」を参照してください。

メモ: 通常、カスケードの対象となるのは、同期していないフルサイズインスタントスナップショットや領域最適化インスタントスナップショットのみです。infrontof スナップショットボリュームが完全に同期している場合（切り離すタイプのスナップショットの場合など）は、カスケードスナップショットを作成しても効用はほとんどありません。

スナップショットのスナップショット作成

図 26-4 は、既存スナップショットのスナップショット作成を表しています。

図 26-4 スナップショットのスナップショット作成



この図のスナップショットの配置は、スナップショットカスケードと同じように見えますが、スナップショット間の関係が再帰的でない点が異なります。スナップショット S2 からデータの読み取りを実行して、該当するデータが S1 にない場合には、もとのボリューム、V から直接データを取得します。

p.727 の 図 26-3 を参照してください。

このような配置は、スナップショットボリューム S1 が処理に不可欠な場合に便利です。たとえば、S1 は、もとのボリューム V の静的なコピーとして使うことができるとします。この場合、もとのボリュームが破損した場合には、もう 1 つのスナップショットボリューム S2 を使ってボリュームをリストアすることができます。データベースでは、S2 から V をリストアする前に S2 に対して REDO ログの再生が必要になることがあります。

図 26-5 に、データベースのリストアに必要な手順を示します。

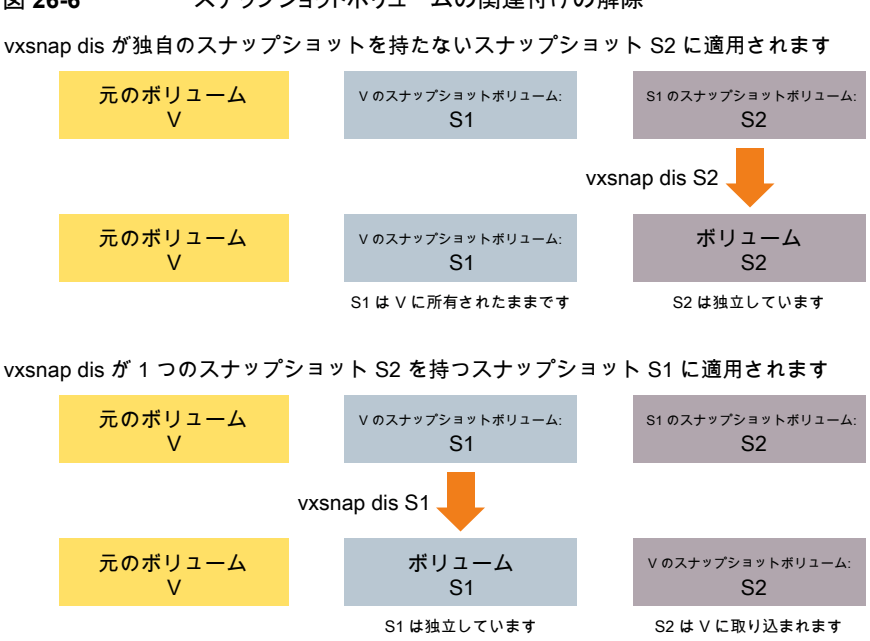
図 26-5 スナップショットのスナップショットを使ったデータベースのリストア

この方法でスナップショットを設定した場合には、1 つ以上のスナップショットを独立したボリュームに変更できます。次の 2 つの `vxsnap` コマンドを使って、この操作を行うことができます。

- `vxsnap dis` コマンドは、スナップショットの関連付けを解除して、独立したボリュームに変更します。関連付けを解除するスナップショットは、その親から完全に同期化されている必要があります。スナップショットボリュームに子スナップショットボリュームがある場合、その子スナップショットも含めて完全に同期化する必要があります。コマンドが正常に実行されると、子スナップショットはもとのボリュームのスナップショットになります。

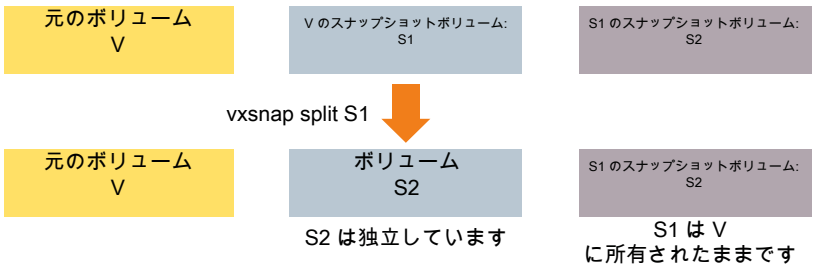
  図 26-6 では、依存するスナップショットを持つスナップショットと、依存するスナップショットを持たないスナップショットに `vxsnap dis` コマンドを適用した結果を示します。

図 26-6 スナップショットボリュームの関連付けの解除



- `vxsnap split` コマンドは、スナップショットと依存するスナップショットを、親ボリュームの関連付けから解除します。分割するスナップショットは、その親から完全に同期化されている必要があります。
  図 26-7 に、`vxsnap split` コマンドの操作を示します。

図 26-7 スナップショットの分割



複数のスナップショットの作成

複数ボリュームのスナップショットの同時作成を容易にするために、`vxsnap make` コマンドと `vxassist snapshot` コマンドにはどちらも複数のボリューム名を引数として指定できるようになっています。

従来のスナップショットでは、vxassist snapshot コマンドにオプション -o allvols を指定し、1 つのディスクグループ内のすべてのボリュームを対象としてスナップショットを作成することもできます。

デフォルトでは、各複製ボリュームは *SNAPnumber-volume* と名付けられます。ここで number は一意のシリアル番号であり、*volume* はスナップショットが作成されるボリューム名です。このデフォルトは、オプション *-o name=pattern* を使って上書きできます。

vxassist(1M)マニュアルページを参照してください。

vxsnap(1M)マニュアルページを参照してください。

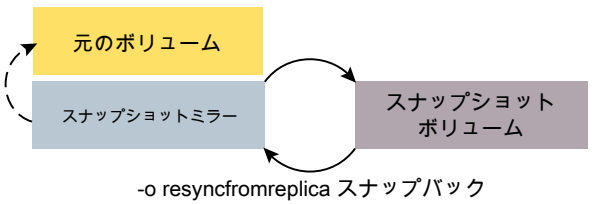
論理グループを形成するすべてのボリューム(たとえば、データベースインスタンスに適合するすべてのボリューム)のスナップショットを作成できます。

スナップショットからの元のボリュームのリストア

従来型のスナップショットの場合、vxassist snapback コマンドの実行中にスナップショットプレックスは元のボリュームのデータから再同期されます。

図 26-8 は、スナップショットで元のボリュームを上書きするための別の方法を示しています。

図 26-8　　元のボリュームのスナップショットからの再同期



オプション -o resyncfromreplica を vxassist に指定すると、スナップショットのデータから元のボリュームが再同期されます。

警告: -o resyncfromreplica オプションを指定した snapback 処理でスナップショットからボリュームを再同期している間は、元のボリュームを使わないでください。データベースなどのアプリケーションをすべて停止し、ボリュームを使うよう設定されているファイルシステムのマウントをすべて解除します。

インスタントスナップショットの場合、vxsnap restore コマンドを使って、インスタントスナップショットまたはインスタントスナップショットから派生したボリュームから、元のボリュームの内容を復元できます。元のボリュームの復元に使うボリュームは、特定時点での元のボリュームの内容の完全なバックアップか、何らかの方法(たとえば、データベースログの

再生を適用するか、fsck などのファイルシステムの検査ユーティリティを実行する）で変更されているボリュームのいずれかになります。元のボリュームを復元する前に、このバックアップボリュームの内容が完全に同期化されている必要があります。内容を復元中でも、元のボリュームにはすぐに使えます。

p.717 の「領域最適化インスタントスナップショットからのボリュームのリストア」を参照してください。

バージョン 0 の DCO および DCO ボリュームの追加

永続 FastResync をあるボリュームに対して有効にする場合、データ変更オブジェクト（DCO）および DCO ボリュームをそのボリュームに関連付ける必要があります。DCO オブジェクトや DCO ボリュームをボリュームに追加すると、ボリューム上で永続 FastResync を有効にできるようになります。

メモ: FastResync 機能を使うには、FastResync ライセンスキーが必要です。ライセンスがない場合でも、DCO オブジェクトと DCO ボリュームの設定によって、スナップオブジェクトを元のボリュームやスナップショットボリュームに関連付けることができます。

この項の手順では、バージョン 0 のレイアウト DCO を追加することについて説明します。バージョン 0 の DCO レイアウトは、vxassist コマンドで管理される従来の（サードミラーブレークオフの）スナップショットをサポートします。バージョン 0 の DCO レイアウトは、フルサイズインスタントスナップショットや領域最適化インスタントスナップショットはサポートしません。

DCO オブジェクトおよび DCO ボリュームを既存のボリュームに追加するには

1 対象となる既存のボリュームを含むディスクグループが、最低でもバージョン 90 にアップグレードされていることを確認します。ディスクグループのバージョンを確認するには、次のコマンドを使います。

```
# vxdg list diskgroup
```

必要に応じて、ディスクグループを最新バージョンにアップグレードします。

```
# vxdg upgrade diskgroup
```

2 現在、元のボリュームで非永続 FastResync が有効になっている場合は、次のコマンドを使って無効にします。

```
# vxvol [-g diskgroup] set fastresync=off volume
```

どのボリュームで非永続 FastResync が有効になっているかが不明な場合は、次のコマンドを使ってそのような状態にあるボリュームのリストを取得します。

メモ: ! 文字は一部のシェルでの特殊文字です。次の例では、bash シェルでこの文字をエスケープ処理する方法を示します。

```
# vxprint [-g diskgroup] -F "%name" ¥
  -e "v_fastresync=on && ¥!v_hasdcolog"
```

3 DCO と DCO ボリュームを(DRL(dirty region logging)がすでに有効になっている可能性のある)既存のボリュームに追加します。

```
# vxassist [-g diskgroup] addlog volume logtype=dco ¥
  [ndcomirror=number] [dcolen=size] [storage_attributes]
```

非階層化ボリュームの場合、ミラー化された DCO ボリュームのデフォルトのプレックス数は、データボリューム内のプレックス数または 2 のうち、いずれか少ない方に等しくなります。階層化ボリュームの場合、デフォルトの DCO プレックス数は常に 2 です。必要であれば、ndcomirror 属性を使って、別の数を指定することもできます。ボリューム内の既存のデータとスナップショットプレックスと同じ数の DCO プレックスを設定することをお勧めします。たとえば、3 面ミラーボリュームに DCO を追加する場合は、ndcomirror=3 と指定します。

各プレックスのデフォルトサイズは 132 ブロックです。dcolen 属性を使って、別のサイズを指定することができます。プレックスのサイズは、2112 ブロックを上限として 33 ブロックの整数倍で指定する必要があります。

vxassist 形式のストレージ属性を指定して、DCO ボリュームのプレックスを配置できるディスクと配置できないディスクを定義できます。

p.734 の「バージョン 0 の DCO プレックスのストレージの指定」を参照してください。

バージョン 0 の DCO プレックスのストレージの指定

ボリュームとそのスナップショットを格納しているディスクを別のディスクグループに移動または分割する場合は、それぞれの DCO プレックスを格納しているディスクも一緒に移動または分割できる状態にある必要があります。デフォルトでは、VxVM はバージョン 0 の DCO プレックスを親ボリュームのデータプレックスと同じディスクに配置しようとします。ただし、該当するディスクに十分な領域が存在しない場合は配置できません。この場合は、VxVM は、ディスクグループ内の他のディスクで使える領域を使います。他のボリュームのプレックスを保持しているディスク上に DCO プレックスが配置されると、そのボリュームを後に別のディスクグループへ移動しようとする際に問題が発生する可能性があります。

ストレージ属性を使うと、DCO プレックスに使うディスクを明示的に指定できます。 可能な場合は、ボリュームが作成されたのと同じディスクを指定します。

たとえば、ディスクグループ mydg 内のボリューム myvol に DCO オブジェクトおよび DCO ボリュームを追加し、そのプレックスを mydg05 および mydg06 上の 264 ブロックで構成するには、次のコマンドを使います。

```
# vxassist -g mydg addlog myvol logtype=dco dcolen=264 mydg05 mydg06
```

ボリュームに関連付けられた DCO オブジェクトと DCO ボリュームの詳細を表示するには、vxprint コマンドを使います。ボリューム vol1 についての vxprint の出力例を次に示します(簡潔にするため、TUTIL0 カラムや PUTIL0 カラムは省略しています)。

```
TY  NAME         ASSOC        KSTATE    LENGTH   PLOFFS   STATE  ...
v   vol1         fsgen        ENABLED   1024     -        ACTIVE
pl  vol1-01      vol1         ENABLED   1024     -        ACTIVE
sd  disk01-01    vol1-01      ENABLED   1024     0        -
pl  vol1-02      vol1         ENABLED   1024     -        ACTIVE
sd  disk02-01    vol1-02      ENABLED   1024     0        -
dc  vol1_dco     vol1         -         -        -        -
v   vol1_dcl     gen          ENABLED   132      -        ACTIVE
pl  vol1_dcl-01  vol1_dcl     ENABLED   132      -        ACTIVE
sd  disk03-01    vol1_dcl-01  ENABLED   132      0        -
pl  vol1_dcl-02  vol1_dcl     ENABLED   132      -        ACTIVE
sd  disk04-01    vol1_dcl-02  ENABLED   132        0        -
```

この出力では、DCO オブジェクトは vol1_dco と表示されています。DCO ボリューム vol1_dcl は 2 つのプレックス vol1_dcl-01 および vol1_dcl-02 と表示されています。

必要に応じて、vxassist move コマンドを使って、DCO プレックスを別のディスクに再配置できます。次のコマンドを使うと、ボリューム vol1 の DCO ボリューム vol1_dcl のプレックスを disk03 と disk04 から disk07 と disk08 に移動できます。

メモ: ! 文字は一部のシェルでの特殊文字です。次の例では、bash シェルでこの文字をエスケープ処理する方法を示します。

```
# vxassist -g mydg move vol1_dcl ¥!disk03 ¥!disk04 disk07 disk08
```

p.955 の「ディスクグループ間の DCO ボリュームの移動」を参照してください。

vxassist(1M)マニュアルページを参照してください。

バージョン 0 の DCO および DCO ボリュームの削除

ボリュームからバージョン 0 の DCO オブジェクト、DCO ボリュームおよび任意のスナップオブジェクトの関連付けを解除するには、次のコマンドを使います。

```
# vxassist [-g diskgroup] remove log volume logtype=dco
```

これによって、DCO オブジェクト、DCO ボリュームおよびスナップオブジェクトは完全に削除されます。さらに、このボリュームの FastResync は無効になります。

また、vxdco コマンドを使っても同じ結果が得られます。

```
# vxdco [-g diskgroup] [-o rm] dis dco_obj
```

ボリュームの DCO オブジェクトのデフォルト名 ***dco_obj*** は、通常、親ボリュームの名前に文字列 `_dco` を追加して作成されます。関連付けられた DCO オブジェクトの名前を確認するには、ボリューム上で `vxprint` コマンドを使います。

ディスクグループ `mydg` 内のボリューム `myvol` から DCO オブジェクト、DCO ボリュームおよびスナップオブジェクトを削除しないで関連付けを解除するには、次のコマンドを使います。

```
# vxdco -g mydg dis myvol_dco
```

この形式のコマンドによって、ボリュームから DCO オブジェクトの関連付けが解除されますが、DCO オブジェクトや DCO ボリュームは破棄されません。`-o rm` オプションを指定すると、DCO オブジェクト、DCO ボリュームとそのプレックス、およびスナップオブジェクトも削除されます。

警告: ボリュームから DCO と DCO ボリュームの関連付けを解除すると、そのボリュームでは永続 FastResync が無効になります。残りのスナップショットをスナップバックする場合は、全体を再同期する必要があります。

`vxassist`(1M)マニュアルページを参照してください。

`vxdco`(1M)マニュアルページを参照してください。

バージョン 0 の DCO および DCO ボリュームの再接続

`vxdco` に `-o rm` オプションを指定しても、バージョン 0 の DCO オブジェクトと DCO ボリュームが削除されない場合は、次のコマンドを使って、親ボリュームに再接続できます。

```
# vxdco [-g diskgroup] att volume dco_obj
```

たとえば、DCO オブジェクト `myvol_dco` をボリューム `myvol` に再接続するには、次のコマンドを使います。

```
# vxdco -g mydg att myvol myvol_dco
```

`vxdco`(1M)マニュアルページを参照してください。

27

Storage Checkpoint の管理

この章では以下の項目について説明しています。

Storage Checkpoint について

Veritas File System（VxFS）には、Storage Checkpoint 機能が用意されており、特定時刻のファイルシステムの永続的なイメージを瞬時に作成できます。Storage Checkpoint は、コピーオンライト技術を使って最後の Storage Checkpoint またはバックアップの実行後に変更されたファイルシステムブロックのみを認識および保持することにより、I/O オーバーヘッドを大幅に低減します。

p.670 の「コピーオンライト」を参照してください。

Storage Checkpoint は次の機能を提供します。

- 再ブートやクラッシュを通した持続性
- ファイルシステムのメタデータ、ディレクトリ階層、ユーザーデータの保存による、データの即時書き込み機能

Storage Checkpoint は実際はデータオブジェクトで、ファイルシステムにより管理、制御されます。Storage Checkpoint は名前を持つデータオブジェクトであり、作成、削除および名前の変更が可能です。

p.668 の「Storage Checkpoint の動作」を参照してください。

独立した格納領域を必要とするディスクベースのミラー化技術とは異なり、Storage Checkpoint では、ファイルシステムの同じ空き領域内で Storage Checkpoint を使うことにより、使うディスク領域が最小限に抑えられます。

マウントされたファイルシステムの Storage Checkpoint の作成後でも、Storage Checkpoint の論理イメージに影響を与えることなく、ファイルシステムのファイルを引き続き作成、削除および更新できます。Storage Checkpoint では、ファイルシステムの名前空間（ディレクトリ階層）のみならず、ファイルシステムのイメージが取得された時に存在するユーザーデータも保存されます。

Storage Checkpoint には、様々な使い方があります。たとえば、次のような使い方があります。

- テープにバックアップできるファイルシステムの安定したイメージを作成します。
- エンドユーザー独自のファイルが誤って削除されてもそのファイルをリストアできるように、ファイルシステムのマウント済みオンディスクバックアップを作成します。これは、特にホームディレクトリ環境、エンジニアリング環境、電子メール環境で便利な機能です。
- パッチをインストールする前に、問題が発生した時にロールバックできるようにアプリケーションのバイナリーコピーを作成します。
- 従来のテープバックアップに加えて、ファイルシステムのオンディスクバックアップの作成により、高速なバックアップとリストアが行えるようになります。
- Storage Checkpoint を書き込み可能としてマウントすることにより、現在のプライマリファイルセット内の稼動中のデータを危険にさらすことなくプライマリファイルセットのポイントインタイムイメージで新しいソフトウェアをテストします。

Storage Checkpoint の管理

Storage Checkpoint の管理操作には `fsckptadm` ユーティリティが必要です。

`fsckptadm`（1M）のマニュアルページを参照してください。

`fsckptadm` ユーティリティを使うと、Storage Checkpoint の作成と削除、属性の変更および統計データの確認ができます。各 Storage Checkpoint には名前が付いています。この名前を使うと、Storage Checkpoint を管理できます。名前は最長 127 文字に制限されており、コロン（:）は使えません。

p.739 の「Storage Checkpoint の作成」を参照してください。

p.740 の「Storage Checkpoint の削除」を参照してください。

Storage Checkpoint には、ファイルシステムの割り当てポリシーまたは Storage Checkpoint の割り当てポリシーによって指定されたボリュームまたはボリュームセット上に、メタデータのための一定の領域が必要です。ボリュームまたはボリュームセットにメタ

データを保存するための十分な空き領域がない場合、fsckptadm ユーティリティはエラーを表示します。メタデータに必要な領域の量は、ファイルシステムのディスクレイアウトバージョンに基づいて概算できます。

ディスクレイアウトバージョン 7 以降の場合は、i ノードの数に 1 バイトを乗算し、1 または 2 メガバイトを加算すると、必要な領域の概数値が得られます。i ノードの数は、fsckptadm ユーティリティで特定できます。

ボリュームセットに十分な空き領域があるかどうかを判断するには、fsvoladm コマンドを使います。

fsvoladm(1M)のマニュアルページを参照してください。

次の例では、ボリュームセットを一覧表示し、ヒューマンフレンドリな単位でストレージサイズを示します。

```
# fsvoladm -H list /mnt0
devid   size     used    avail   name
0       20 GB    10 GB   10 GB   vol1
1       30 TB    10 TB   20 TB   vol2
```

Storage Checkpoint の作成

次の例では、/mnt0 ファイルシステム上に thu_7pm という名前の Nodata Storage Checkpoint を 1 つ作成し、/mnt0 ファイルシステムのすべての Storage Checkpoint を一覧表示します。

```
# fsckptadm -n create thu_7pm /mnt0
# fsckptadm list /mnt0
/mnt0
thu_7pm:
 ctime   = Thu 3 Mar 2005 7:00:17 PM PST
 mtime   = Thu 3 Mar 2005 7:00:17 PM PST
 flags   = nodata, largefiles
```

次の例では、/mnt0 ファイルシステム上に thu_8pm という名前の Removable Storage Checkpoint を 1 つ作成し、/mnt0 ファイルシステムのすべての Storage Checkpoint を一覧表示します。

```
# fsckptadm -r create thu_8pm /mnt0
# fsckptadm list /mnt0
/mnt0
thu_8pm:
 ctime   = Thu 3 Mar 2005 8:00:19 PM PST
 mtime   = Thu 3 Mar 2005 8:00:19 PM PST
 flags   = largefiles, removable
```

```
thu_7pm:
 ctime   = Thu 3 Mar 2005 7:00:17 PM PST
 mtime   = Thu 3 Mar 2005 7:00:17 PM PST
 flags   = nodata, largefiles
```

Storage Checkpoint の削除

Storage Checkpoint を削除するには、fsckptadm コマンドの remove キーワードを指定します。具体的には、同期または非同期のいずれかの方法を使って、Storage Checkpoint を削除します。デフォルトでは、非同期方法が指定されています。同期方法の場合は、Storage Checkpoint 全体を削除し、すべてのブロックをファイルシステムの空き領域に戻した後、fsckptadm 操作を完了します。非同期方法の場合は、削除する Storage Checkpoint にマークを付けたら、ただちに fsckptadm が戻ります。その後、独立したカーネルスレッドにより、削除を実行し、Storage Checkpoint をファイルシステムの空き領域に戻します。

この例では、/mnt0 は、マウント済みの VxFS ファイルシステムです。この例は、thu_8pm という名前の Storage Checkpoint の非同期削除と thu_7pm という名前の Storage Checkpoint の同期削除を示しています。ここでは、指定した Storage Checkpoint を削除した後、/mnt0 ファイルシステム上に存在するすべての Storage Checkpoint を一覧表示します。

```
# fsckptadm remove thu_8pm /mnt0
# fsckptadm list /mnt0
/mnt0
thu_7pm:
 ctime   = Thu 3 Mar 2005 7:00:17 PM PST
 mtime   = Thu 3 Mar 2005 7:00:17 PM PST
 flags   = nodata, largefiles
# fsckptadm -s remove thu_7pm /mnt0
# fsckptadm list /mnt0
/mnt0
```

Storage Checkpoint へのアクセス

Storage Checkpoint をマウントするには、mount コマンドの -o ckpt=*ckpt_name* マウントオプションを使います。

mount_vxfs（1M）のマニュアルページを参照してください。

Storage Checkpoint をマウントする場合は、次のルールに従います。

- デフォルトでは、Storage Checkpoint は読み書き Storage Checkpoint としてマウントされます。

- Storage Checkpoint が現時点で読み取り専用 Storage Checkpoint としてマウントされている場合は、-o remount オプションを使って、書き込み可能な Storage Checkpoint として再マウントできます。
- ファイルシステムの Storage Checkpoint をマウントするには、まずファイルシステム自体をマウントする必要があります。
- ファイルシステムのマウントを解除する前に、ファイルシステムのすべての Storage Checkpoint のマウントを解除する必要があります。

警告: バックアップ目的で Storage Checkpoint を作成する場合は、書き込み可能な Storage Checkpoint としてマウントしないでください。Storage Checkpoint に誤って書き込むと、その時点でのイメージが失われます。

既存の Storage Checkpoint がある場合は、書き込み可能な Storage Checkpoint への書き込みアクティビティでコピー操作が発生し、既存の Storage Checkpoint で使われる領域が増えることがあります。

Storage Checkpoint は、特殊な擬似デバイスにマウントされます。この擬似デバイスはシステムの名前空間には存在せず、Storage Checkpoint がマウントされるときにシステムによってシステム内部に作成され、使われます。この擬似デバイスは、Storage Checkpoint のマウントを解除すると削除されます。擬似デバイス名は、ファイルシステムデバイス名に Storage Checkpoint 名を追加して作成します。区切り文字にはコロン(:)を使います。

たとえば、特殊なデバイス /dev/vx/dsk/fsvol/vol1 に常駐するファイルシステムに Storage Checkpoint として may_23 がある場合、Storage Checkpoint の擬似デバイス名は次のようになります。

```
/dev/vx/dsk/fsvol/vol1:may_23
```

- ディレクトリ /fsvol_may_23 に読み取り専用 Storage Checkpoint として may_23 という名前の Storage Checkpoint をマウントするには、次のように入力します。

```
# mount -t vxfs -o ckpt=may_23 /dev/vx/dsk/fsvol/vol1:may_23 ¥
/fsvol_may_23
```

メモ: Storage Checkpoint をマウントするには、vol1 ファイルシステムがすでにマウントされている必要があります。

- may_23 という名前の Storage Checkpoint を書き込み可能な Storage Checkpoint として再マウントするには、次のように入力します。

```
# mount -t vxfs -o ckpt=may_23,remount,rw ¥
/dev/vx/dsk/fsvol/vol1:may_23 /fsvol_may_23
```

- システムの起動時に Storage Checkpoint を自動的にマウントするには、/etc/fstab ファイルに次のように記述します。

```
Device-Special-File  Mount-Point    fstype  options     backup-    pass-
                                                         frequency  number
/dev/vx/dsk/fsvol/   /fsvol         vxfs    defaults    0          0
  vol1
/dev/vx/dsk/fsvol/   /fsvol_may_23  vxfs    ckpt=may_23 0          0
  vol1:may_23
```

- クラスタファイルシステムの Storage Checkpoint をマウントするには、-o cluster オプションも指定する必要があります。

```
# mount -t vxfs -o cluster,ckpt=may_23 ¥
/dev/vx/dsk/fsvol/vol1:may_23 /fsvol_may_23
```

Storage Checkpoint をクラスタレベルでマウントできるのは、Storage Checkpoint が属するファイルシステムもクラスタレベルでマウントされている場合のみです。同様に、Storage Checkpoint をローカルにマウントできるのは、Storage Checkpoint が属するファイルシステムもローカルにマウントされている場合のみです。

umount コマンドを使って Storage Checkpoint のマウントを解除できます。

umount（1M）マニュアルページを参照してください。

次のようにマウントポイントまたは擬似デバイスの名前を使って、Storage Checkpoint のマウントを解除することができます。

```
# umount /fsvol_may_23
# umount /dev/vx/dsk/fsvol/vol1:may_23
```

メモ: 擬似デバイスは実際のファイルシステムの一部であるため、fsck ユーティリティを、Storage Checkpoint の擬似デバイスに対して実行する必要はありません。

Nodata Storage Checkpoint への Data Storage Checkpoint の変換

Nodata Storage Checkpoint には、実際のファイルデータが含まれません。Nodata Storage Checkpoint には、Storage Checkpoint の作成後に変更されたすべてのブロックの位置を示すマーカーの集合が含まれます。

p.672 の「Storage Checkpoint の種類」を参照してください。

Data Storage Checkpoint の Nodata Storage Checkpoint への変換には、同期変換または非同期変換の 2 つの方法があります。デフォルトの方法は非同期変換です。同期変換の場合、fsckptadm は、すべてのファイルが変換プロセスで nodata 状態になるまで待機し、操作を完了します。非同期変換の場合は、Storage Checkpoint のデータ

ブロックがファイルシステムの空きブロックプールにすぐに戻されないときでも、すぐに fsckptadm が制御を戻し、Storage Checkpoint に Nodata Storage Checkpoint のマークを付けます。Storage Checkpoint により、バックグラウンドで確保されたすべてのファイルデータブロックの領域が解放され、最後にファイルシステムの空きブロックプールに戻されます。

ファイルシステムの過去の Storage Checkpoint がすべて Nodata Storage Checkpoint である場合は、同期方法を使って、Data Storage Checkpoint を Nodata Storage Checkpoint に変換します。ファイルシステムに過去の Data Storage Checkpoint が存在する場合は、非同期方法を使って、後から変換する必要がある Storage Checkpoint にマークを付けます。この場合、Storage Checkpoint がファイルシステムで最も古い Storage Checkpoint になるまで、または過去の Storage Checkpoint すべてが Nodata Storage Checkpoint に変換されるまで、実際の変換操作は遅延されます。

メモ: Nodata Storage Checkpoint では、ブロックの変更位置が追跡されるだけで、ファイルデータブロックの内容が保存されないため、Nodata Storage Checkpoint を Data Storage Checkpoint に変換することはできません。

---

data Storage Checkpoint と nodata Storage Checkpoint の相違点の表示

次の例で、data Storage Checkpoint と nodata Storage Checkpoint の相違点について示します。

メモ: nodata Storage Checkpoint には、実際のファイルデータが含まれません。

Storage Checkpoint 間の相違点を表示するには

1 次の例のように、ファイルシステムを作成し、/mnt0 にマウントします。

```
# mkfs -t vxfs /dev/vx/rdsk/dg1/test0

version 16 layout
62914560 sectors, 31457280 blocks of size 1024, log size 65536
blocks
rcq size    2048 blocks
largefiles  supported
maxlink     supported
WORM not supported

size  65536 blocks, largefiles supported
# mount -t vxfs /dev/vx/dsk/dg1/test0 /mnt0
```

2 次の例のように、既知の内容を確認するための小さなファイルを作成します。

```
# echo "hello, world" > /mnt0/file
```

3 次の例のように、Storage Checkpoint を作成し、/mnt0@5_30pm にマウントします。

```
# fsckptadm create ckpt@5_30pm /mnt0
# mkdir /mnt0@5_30pm
# mount -t vxfs -o ckpt=ckpt@5_30pm ¥
  /dev/vx/dsk/dg1/test0:ckpt@5_30pm  /mnt0@5_30pm
```

4 元のファイルと Storage Checkpoint ファイルの内容を確認します。

```
# cat /mnt0/file
hello, world
# cat /mnt0@5_30pm/file
hello, world
```

5 次のように、元のファイルの内容を変更します。

```
# echo "goodbye" > /mnt0/file
```

6 元のファイルと Storage Checkpoint ファイルの内容を確認します。次のように、元のファイルには、最新のデータが含まれているのに対して、Storage Checkpoint ファイルには、Storage Checkpoint 作成時のデータが含まれています。

```
# cat /mnt0/file
goodbye
# cat /mnt0@5_30pm/file
hello, world
```

7 Storage Checkpoint のマウントを解除し、nodata Storage Checkpoint に変換してから再マウントします。

```
# umount /mnt0@5_30pm
# fsckptadm -s set nodata ckpt@5_30pm /mnt0
# mount -t vxfs -o ckpt=ckpt@5_30pm ¥
/dev/vx/dsk/dg1/test0:ckpt@5_30pm /mnt0@5_30pm
```

8 両方のファイルの内容を確認します。次のように、元のファイルには、最新のデータが含まれています。

```
# cat /mnt0/file
goodbye
```

次のように、ディレクトリを探して nodata Storage Checkpoint のディレクトリを読み取ります。ただし、ファイルにはデータが含まれておらず、Storage Checkpoint の作成後に変更されたファイルブロックを示すマーカーのみが含まれています。

```
# ls -l /mnt0@5_30pm/file
-rw-r--r--   1 root     other 13 Jul 13 17:13 ¥
# cat /mnt0@5_30pm/file
cat: /mnt0@5_30pm/file: Input/output error
```

複数の Storage Checkpoint の変換

同じファイルシステム上にある古い Storage Checkpoint を扱う際に Storage Checkpoint を Nodata Storage Checkpoint に変換することができます。

複数の Storage Checkpoint の変換手順

1 次のように、ファイルシステムを作成して /mnt0 にマウントします。

```
# mkfs -t vxfs /dev/vx/rdsk/dg1/test0
version 16 layout
134217728 sectors, 67108864 blocks of size 1024, log size 65536
blocks
rcq size 4096 blocks
largefiles supported
maxlink    supported
# mount -t vxfs /dev/vx/dsk/dg1/test0 /mnt0
```

2 次のように、このファイルシステムに Data Storage Checkpoint を 4 つ作成して一覧表示します。作成する順序に注意してください。

```
# fsckptadm create oldest /mnt0
# fsckptadm create older /mnt0
# fsckptadm create old /mnt0
# fsckptadm create latest /mnt0
# fsckptadm list /mnt0
  /mnt0
  latest:
    ctime                =  Mon 26 Jul 11:56:55 2004
    mtime                =  Mon 26 Jul 11:56:55 2004
    flags                =  largefiles
  old:
    ctime                =  Mon 26 Jul 11:56:51 2004
    mtime                =  Mon 26 Jul 11:56:51 2004
    flags                =  largefiles
  older:
    ctime                =   Mon 26 Jul 11:56:46 2004
    mtime                =  Mon 26 Jul 11:56:46 2004
    flags                =  largefiles
  oldest:
    ctime                =  Mon 26 Jul 11:56:41 2004
    mtime                =  Mon 26 Jul 11:56:41 2004
    flags                =  largefiles
```

3 Nodata Storage Checkpoint への latest Storage Checkpoint の同期変換の実行を試みます。この試みは次のように失敗します。理由は、latest Storage Checkpoint より古い Storage Checkpoint が old 、older、oldest という Data Storage Checkpoint であるためです。

```
# fsckptadm -s set nodata latest /mnt0
UX:vxfs fsckptadm: ERROR: V-3-24632: Storage Checkpoint
set failed on latest.  File exists (17)
```

4 この場合は、遅延または非同期変換を使って、latest Storage Checkpoint を Nodata Storage Checkpoint に変換できます。

```
# fsckptadm set nodata latest /mnt0
```

5 次の例のように、Storage Checkpoint を一覧表示します。Storage Checkpoint の latest には遅延変換するためのマークが付いています。

```
# fsckptadm list /mnt0
  /mnt0
  latest:
    ctime                = Mon 26 Jul 11:56:55 2004
    mtime                = Mon 26 Jul 11:56:55
    flags                = nodata, largefiles, delayed
  old:
    ctime                = Mon 26 Jul 11:56:51 2004
    mtime                = Mon 26 Jul 11:56:51 2004
    flags                = largefiles
  older:
    ctime                = Mon 26 Jul 11:56:46 2004
    mtime                = Mon 26 Jul 11:56:46 2004
    flags                = largefiles
  oldest:
    ctime                = Mon 26 Jul 11:56:41 2004
    mtime                = Mon 26 Jul 11:56:41 2004
    flags                = largefiles
```

遅延 Nodata Storage Checkpoint の作成

Storage Checkpoint を遅延 Nodata Storage Checkpoint として作成できます。作成プロセスでは、古い Data Storage Checkpoint を検出し、Storage Checkpoint の「」を遅延 Nodata Storage Checkpoint として作成します。次の例では、latest という名前の Storage Checkpoint を削除し、遅延 Nodata Storage Checkpoint として再作成します。

遅延 Nodata Storage Checkpoint を作成するには

1 Storage Checkpoint の「latest」を削除します。

```
# fsckptadm remove latest /mnt0
# fsckptadm list /mnt0
/mnt0
old:
  ctime                = Mon 26 Jul 11:56:51 2004
  mtime                = Mon 26 Jul 11:56:51 2004
  flags                = largefiles
older:
  ctime                = Mon 26 Jul 11:56:46 2004
  mtime                = Mon 26 Jul 11:56:46 2004
  flags                = largefiles
oldest:
  ctime                = Mon 26 Jul 11:56:41 2004
  mtime                = Mon 26 Jul 11:56:41 2004
  flags                = largefiles
```

2 latest の Storage Checkpoint を Nodata Storage Checkpoint として再作成します。

```
# fsckptadm -n create latest /mnt0
# fsckptadm list /mnt0
/mnt0
latest:
  ctime                = Mon 26 Jul 12:06:42 2004
  mtime                = Mon 26 Jul 12:06:42 2004
  flags                = nodata, largefiles, delayed
old:
  ctime                = Mon 26 Jul 11:56:51 2004
  mtime                = Mon 26 Jul 11:56:51 2004
  flags                = largefiles
older:
  ctime                = Mon 26 Jul 11:56:46 2004
  mtime                = Mon 26 Jul 11:56:46 2004
  flags                = largefiles
oldest:
  ctime                = Mon 26 Jul 11:56:41 2004
  mtime                = Mon 26 Jul 11:56:41 2004
  flags                = largefiles
```

3 ファイルシステムに、Storage Checkpoint の「oldest」より古い Storage Checkpoint でデータを含んでいるものはないため、これを Nodata Storage Checkpoint に変換します。

メモ: この手順は同期実行できます。

```
# fsckptadm -s set nodata oldest /mnt0
# fsckptadm list /mnt0
/mnt0
latest:
  ctime                = Mon 26 Jul 12:06:42 2004
  mtime                = Mon 26 Jul 12:06:42 2004
  flags                = nodata, largefiles, delayed
old:
  ctime                = Mon 26 Jul 11:56:51 2004
  mtime                = Mon 26 Jul 11:56:51 2004
  flags                = largefiles
older:
  ctime                = Mon 26 Jul 11:56:46 2004
  mtime                = Mon 26 Jul 11:56:46 2004
  flags                = largefiles
oldest:
  ctime                = Mon 26 Jul 11:56:41 2004
  mtime                = Mon 26 Jul 11:56:41 2004
  flags                = nodata, largefiles
```

4 older の Storage Checkpoint と old の Storage Checkpoint を削除します。

```
# fsckptadm remove older /mnt0
# fsckptadm remove old /mnt0
# fsckptadm list /mnt0
/mnt0
latest:
  ctime                = Mon 26 Jul 12:06:42 2004
  mtime                = Mon 26 Jul 12:06:42 2004
  flags                = nodata, largefiles
oldest:
  ctime                = Mon 26 Jul 11:56:41 2004
  mtime                = Mon 26 Jul 11:56:41 2004
  flags                = nodata, largefiles
```

メモ: 次のように、older の Storage Checkpoint と old の Storage Checkpoint を削除すると、latest の Storage Checkpoint が自動的に Nodata Storage Checkpoint に変換されます。これは、唯一残っている oldest の Storage Checkpoint がすでに Nodata Storage Checkpoint であるためです。

Storage Checkpoint の可視性を有効または無効にする

Storage Checkpoint の可視性は、ckptautomnt マウントオプションを off、ro、rw のいずれかの値に設定することで有効にできます。Storage Checkpoint の可視性を有効にするとクローンの手動マウントができなくなるため、デフォルト値は off です。オプションを ro に設定すると、すべてのクローンが読み取り専用として自動マウントされ、rw に設定するとすべてのクローンは読み取り/書き込み用として自動マウントされます。

プライマリファイルセットの代わりに既存の Storage Checkpoint の Storage Checkpoint を選択する場合、.checkpoint 関数のソース Storage Checkpoint のディレクトリはマウントポイントとして機能します。たとえば、/mnt にマウントされているファイルシステムの Storage Checkpoint clone1 の Storage Checkpoint を選択する場合は、次のコマンドを実行します。

```
# fsckptadm create clone2 /mnt/.checkpoint/clone1
```

デフォルトでは、SF（Storage Foundation）は i ノード番号を一意にしません。ただし、一意の 64 ビットの i ノード番号の使用を有効にするために uniqueino マウントオプションを指定できます。このオプションは再マウント中には変更できません。

次の例では、すべてのクローンを読み取り/書き込み用として自動マウントすることによって、Storage Checkpoint の可視性を有効にします。

```
# mount -t vxfs -o ckptautomnt=rw /dev/vx/dsk/dg1/vol1 /mnt1
```

Storage Checkpoint の領域管理に関する注意事項

Storage Checkpoint を含むファイルシステムで領域が不足すると、ファイルの削除や既存ファイルの上書きなど、一部の操作が失敗することがあります。システムで十分な領域を確保できない場合は、操作に失敗します。

通常、データベースではファイル用に格納領域が事前に割り当てられており、書き込み操作が失敗することはありません。`create` や `mkdir` などのユーザー操作中に、ファイルシステムの領域が不足すると、Removable Storage Checkpoint は削除されます。これにより、ディスクスペースの領域不足による中断なしに、アプリケーションを続行できます。Non-removable Storage Checkpoint は `ENOSPC` のような状況下では自動的に削除されません。Removable Storage Checkpoint のみを作成することをお勧めします。ただし、特定の管理操作(`fsadm`コマンドの使用、`qiomkfile`コマンドの使用、`fsckptadm` コマンドによる Storage Checkpoint の作成など)の実行中は、ファイルシステムの領域が不足しても Removable Storage Checkpoint は削除されません。

カーネルが Storage Checkpoint を自動的に削除する場合は、次のポリシーが適用されます。

- 削除する Storage Checkpoint 数をできる限り抑えようとします。
- Non-removable Storage Checkpoint を削除候補から除外します。
- Data Storage Checkpoint が無くなった場合に限り、Nodata Storage Checkpoint を選択します。
- 最も古い Storage Checkpoint から削除します。
- Storage Checkpoint を削除します(マウントされている場合でも)。このように削除された Storage Checkpoint では、新しい操作が失敗し、該当するエラーコードが表示されます。
- 最も古い Storage Checkpoint が Non-removable Storage Checkpoint の場合は、最も古い Removable Storage Checkpoint が削除用に選択されます。このような場合、データを Non-removable Storage Checkpoint に強制的に保存しなければならない可能性があり、これに失敗した場合は FULLFSCK としてファイルシステムがマーク付けされます。これを避けるには、Removable Storage Checkpoint のみを作成することをお勧めします。

Storage Checkpoint からのリストア

バックアップアプリケーションやリストアアプリケーションでは、整合性があり損傷を受けていないファイルシステム上にあるマウント可能な Data Storage Checkpoint を使って、個別のファイルまたはファイルシステム全体をリストアできます。Storage Checkpoint からリ

ストアすることで、不正に変更されたファイルを修復することもできます。ただし、通常は、ハードウェアの障害などによるファイルシステムの整合性の問題を修復することはできません。

メモ: ハードウェアなどによる整合性の問題が発生した場合は、Storage Checkpoint からではなく、他のメディアからバックアップする必要があります。

ファイルをリストアするには、マウントされた Storage Checkpoint からプライマリファイルセットにファイル全体をコピーします。ファイルシステム全体をリストアするには、`fsckpt_restore` コマンドを使って、マウント可能な Data Storage Checkpoint をプライマリファイルセットとして指定します。

`fsckpt_restore`(1M)のマニュアルページを参照してください。

`fsckpt_restore` コマンドを使って Storage Checkpoint からファイルシステムをリストアする場合、その Storage Checkpoint の作成後に発生したファイルシステムへの変更は、完全に失われます。保存される Storage Checkpoint とデータは、選択した Storage Checkpoint の作成と同時か、それ以前に作成されたものだけです。`fsckpt_restore` が呼び出されるときにファイルシステムをマウントすることはできません。

メモ: 個々のファイルをリストアするときは、ライブラリ関数 `fsckpt_fbmap`(3)を使ってファイルデータの変更部分のみをリストアすることで、効率的にリストアできます。

Storage Checkpoint からのリストアは、ディスクレイアウトバージョンが 6 以降であるファイルシステムに対してのみ実行できます。

次の例では、ホームディレクトリに格納されているファイル `file1.txt` を、`CKPT1` という Storage Checkpoint からデバイス `/dev/vx/dsk/dg1/vol-01` にリストアします。このデバイスのマウントポイントは `/home` です。

Storage Checkpoint からのファイルのリストア方法

1 `/home` の Storage Checkpoint `CKPT1` を作成します。

```
$ fsckptadm create CKPT1 /home
```

2 Storage Checkpoint `CKPT1` をディレクトリ `/home/checkpoints/mar_4` にマウントします。

```
$ /opt/VRTS/bin/mount -o ckpt=CKPT1 /dev/vx/dsk/dg1/vol- ¥
01:CKPT1 /home/checkpoints/mar_4
```

3 ホームディレクトリから file1.txt ファイルを削除します。

```
$ cd /home/users/me
$ rm file1.txt
```

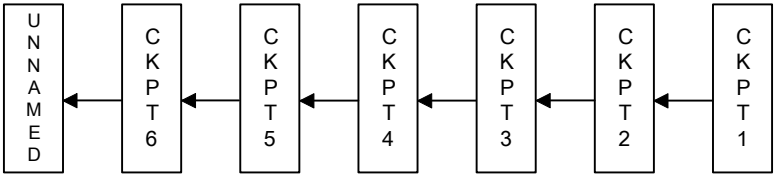
4 ホームディレクトリのイメージが含まれている /home/checkpoints/mar_4/users/me ディレクトリに移動します。

```
$ cd /home/checkpoints/mar_4/users/me
$ ls -l
-rw-r--r--  1  me  staff  14910  Mar 4  17:09  file1.txt
```

5 ファイル file1.txt をホームディレクトリにコピーします。

```
$ cp file1.txt /home/users/me
$ cd /home/users/me
$ ls -l
-rw-r--r--   1 me  staff  14910  Mar 4  18:21  file1.txt
```

次の例では、CKPT3 という Storage Checkpoint からファイルシステムをリストアします。リストア前のファイルセットの一覧には、名前がない 1 つのルートファイルセットと 6 つの Storage Checkpoint があります。



Storage Checkpoint からファイルシステムをリストアするには

1 fsckpt_restore コマンドを実行します。

```
# fsckpt_restore -l /dev/vx/dsk/dg1/vol2
/dev/vx/dsk/dg1/vol2:
UNNAMED:
  ctime        = Thu 08 May 2004 06:28:26 PM PST
  mtime        = Thu 08 May 2004 06:28:26 PM PST
  flags        = largefiles, file system root
CKPT6:
   ctime        = Thu 08 May 2004 06:28:35 PM PST
  mtime        = Thu 08 May 2004 06:28:35 PM PST
  flags        = largefiles
CKPT5:
  ctime        = Thu 08 May 2004 06:28:34 PM PST
  mtime        = Thu 08 May 2004 06:28:34 PM PST
  flags        = largefiles, nomount
CKPT4:
  ctime        = Thu 08 May 2004 06:28:33 PM PST
  mtime        = Thu 08 May 2004 06:28:33 PM PST
  flags        = largefiles
CKPT3:
  ctime        = Thu 08 May 2004 06:28:36 PM PST
  mtime        = Thu 08 May 2004 06:28:36 PM PST
  flags        = largefiles
CKPT2:
  ctime        = Thu 08 May 2004 06:28:30 PM PST
  mtime        = Thu 08 May 2004 06:28:30 PM PST
  flags        = largefiles
CKPT1:
  ctime        = Thu 08 May 2004 06:28:29 PM PST
  mtime        = Thu 08 May 2004 06:28:29 PM PST
  flags        = nodata, largefiles
```

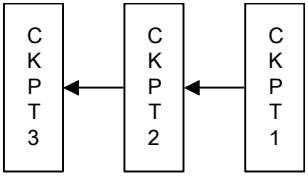
2 次の例では、Storage Checkpoint[CKPT3]を新しいルートファイルセットとして選択します。

```
Select Storage Checkpoint for restore operation
 or <Control/D> (EOF) to exit
 or <Return> to list Storage Checkpoints: CKPT3
CKPT3:
  ctime        = Thu 08 May 2004 06:28:31 PM PST
  mtime        = Thu 08 May 2004 06:28:36 PM PST
  flags        = largefiles
UX:vxfs fsckpt_restore: WARNING: V-3-24640: Any file system
changes or Storage Checkpoints made after
Thu 08  May 2004 06:28:31 PM PST will be lost.
```

3 y を入力して、CKPT3 からファイルシステムをリストアします。

```
Restore the file system from Storage Checkpoint CKPT3 ?
(ynq) y
(Yes)
UX:vxfs fsckpt_restore: INFO: V-3-23760: File system
restored from CKPT3
```

この時点でファイルセットの一覧を表示すると、以前に表示されていた UNNAMED（名前がない）ルートファイルセット、CKPT6、CKPT5 および CKPT4 が削除され、CKPT3 がプライマリファイルセットとして表示されます。これで、CKPT3 がデフォルトでマウントされるファイルセットになります。



4 fsckpt_restore コマンドを実行します。

```
# fsckpt_restore -l /dev/vx/dsk/dg1/vol2
/dev/vx/dsk/dg1/vol2:
CKPT3:
  ctime        = Thu 08 May 2004 06:28:31 PM PST
  mtime        = Thu 08 May 2004 06:28:36 PM PST
  flags        = largefiles, file system root
CKPT2:
  ctime        = Thu 08 May 2004 06:28:30 PM PST
  mtime        = Thu 08 May 2004 06:28:30 PM PST
  flags        = largefiles
CKPT1:
  ctime        = Thu 08 May 2004 06:28:29 PM PST
  mtime        = Thu 08 May 2004 06:28:29 PM PST
  flags        = nodata, largefiles
Select Storage Checkpoint for restore operation
 or <Control/D> (EOF) to exit
 or <Return> to list Storage Checkpoints:
```

Storage Checkpoint クォータ

VxFS の fsckptadm コマンドインターフェースには、Storage Checkpoint クォータを管理するオプションがあります。Storage Checkpoint クォータでは、プライマリファイルセットのすべての Storage Checkpoint で使われる領域量を次のように制限します。

ハード制限　超過できない絶対的な制限です。ハード制限を超えた場合は、以後の任意の Storage Checkpoint の割り当てはすべて失敗します。ただし、既存の Storage Checkpoint は保持されます。

ソフト制限　ハード制限の範囲内で設定する必要があります。ソフト制限を超えた場合は、新しい Storage Checkpoint を作成できません。さらに多くの Storage Checkpoint を作成するには、使用するブロック数を、ソフト制限を超えない値まで抑える必要があります。アラートとコンソールメッセージが生成されます。

ハード制限を超えた場合は、fsckptadm ユーティリティで -f オプションを指定する、または指定しないという各種対処法があります。

fsckptadm(1M)のマニュアルページを参照してください。

次は、-f オプションの指定の有無による効果の違いです。

- -f オプションが指定されていない場合、削除可能な Storage Checkpoint が 1 つ以上削除され、操作を実行するために必要な領域が確保されます。これはデフォルトの対処法です。
- -f オプションが指定されている場合、以後の任意の Storage Checkpoint の割り当てはすべて失敗します。ただし、既存の Storage Checkpoint は保持されます。

メモ: 別のプロセスで使用中のファイルを削除する場合、そのプロセスが終了するまでファイルは削除されません。ファイルを削除することで、データがダウンストリーム Storage Checkpoint(次に古い Storage Checkpoint)に強制的に保存される場合があるため、ファイルセットのハード制限クォータを超えてしまうことがあります。この場合は、ハード制限がゆるくなり、i ノードは不良とマークされません。これは、一部の非同期 i ノード操作にも当てはまります。

28

FileSnaps の管理

この章では以下の項目について説明しています。

FileSnap の作成

同一のファイルの FileSnap を作成する単一のスレッドは 1 分あたり 10,000 以上のスナップショットを作成できます。FileSnaps は、仮想マシンのゴールデンイメージのクローンを作成することで、新しい仮想マシンの高速プロビジョニング用に使うことができます。ゴールデンイメージは、仮想環境のデータストアとして使われる VxFS ファイルシステムまたは Storage Foundation Cluster File System High Availability（SFCFSHA）ファイルシステムにファイルとして格納されます。

Network File System での FileSnap の作成

既存のファイルから拡張子「::snap:vxfs:」を持つ新しいファイルへのハードリンクを作成することによって、NFS（ネットワークファイルシステム）に FileSnap を作成できます。たとえば、次のコマンドでは `file1` という名前の新しいファイルが作成されますが、`file1` を `file2` のハードリンクとして作成する代わりに、`file2` のリンク数が変更されないように `file1` が FileSnap になります。

```
# ln file1 file2::snap:vxfs:
```

これは次のコマンドを使用する場合と同様です。

```
# vxfilesnap -p file1 file2
```

新しいファイルは古いファイルと同じ属性を持ち、古いファイルのエクステントがすべて共有されます。

この名前空間の拡張子を使用するアプリケーションでは、`file1` ではなく `file1::snap:vxfs:` など、作成されたファイルに名前空間の拡張子があるかどうかを確認する必要があります。これは、NFS 経由でエクスポートされたファイルシステムが VxFS でない、ファイルシステムが VxFS の古いバージョンである、ファイルシステムに FileSnap のライセンスがないなどの理由で、名前空間の拡張子がサポートされていないことを示します。

`vxfilesnap` コマンドと同様に、FileSnap は 1 つのファイルセット内に作成する必要があります。

FileSnap の使用

表 28-1 は、FileSnap の管理を可能にする VxFS (Veritas File System) コマンドの一覧を提供します。

表 28-1

コマンド	機能
`fiostat`	`fiostat` コマンドには、時間間隔ごとの統計を表示する `-S shared` オプションがあります。このオプションを使用しない場合、コマンドは時間間隔全体の累積統計を表示します。
`fsadm`	`fsadm` コマンドには、ファイルシステム内の共有ブロックの使用率をレポートする `-S` オプションがあります。このオプションを使って、FileSnap によって達成されたストレージの節約量とすべてのファイルを完全にコピーした場合に実際に必要になるストレージ容量を確認することができます。 `fsadm_vxfs`(1M) のマニュアルページを参照してください。
`fsmap`	`fsmap` コマンドには、1 つのファイルが消費する物理ブロックの合計数とこれらのブロックの何個が特定のファイル専用になっていないかをレポートする `-c` オプションがあります。 `fsmap`(1) のマニュアルページを参照してください。
`mkfs`	`mkfs` コマンドで `-o version=`*`supportedVersion`*を指定して、特定のサポート対象バージョンのファイルシステムでディスクレイアウトを作成します。デフォルトでは、サポートされている最新のディスクレイアウトバージョンが使用されます。 VxFS は、共有エクステント参照での遅延操作のリストを内部的に保持し、このリストのサイズ (`rcqsize`) のデフォルト値をファイルシステムサイズの関数の値に設定しますが、ファイルシステムを作成する際に変更できます。 `mkfs_vxfs`(1M) のマニュアルページを参照してください。

コマンド	機能
vxfilesnap	vxfilesnapコマンドを使って、ファイル、ファイルのセット、またはディレクトリ内のファイルのスナップショットを作成します。vxfilesnap コマンドを使って、古いバージョンのファイルを現在のファイルに復元することもできます。 vxfilesnap(1) のマニュアルページを参照してください。
vxtunefs	vxtunefsコマンドは、パフォーマンスを改善するために、ファイルシステム上で遅延コピーオンライトチューニングパラメータ lazy_copyonwrite を有効にするオプションをサポートします。 vxtunefs(1M) のマニュアルページを参照してください。

FilsSnap を使用した PITC（ポイントインタイムコピー）ファイルの作成

FileSnap のパフォーマンスを最大化するための秘訣は、コピーオンライトのオーバーヘッドを最小化することです。遅延コピーオンライトを有効化することで、これを実現できます。遅延コピーオンライトは簡単に有効化することができ、通常パフォーマンスが大幅に向上します。遅延コピーオンライトが検討中の使用事例で有効なオプションでない場合、ソースファイルを効率的に割り当てることで、コピーオンライトの必要性を軽減することができます。

仮想デスクトップをプロビジョニングするための FileSnap の使用

VDI（仮想デスクトップインフラ）オペレーティングシステムのブートイメージは FileSnap の良い使用事例です。変更される可能性があるブートイメージの部分は、ユーザープロファイル、ページファイル（または UNIX/Linux のスワップ）、およびアプリケーションデータです。これらのデータをブートイメージから分離して、共有解除を最小化する必要があります。マスターブートイメージファイルに単一エクステントを割り当てる必要があります。

次の例では、すべてのスナップショットによって共有される単一エクステントを含む 4 GB のマスターブートイメージを使います。

```
# touch /vdi_images/master_image
# /opt/VRTS/bin/setext -r 4g -f chgsize /vdi_images/master_image
```

master_image ファイルは、オペレーティングシステムをインストールするために仮想マシンのディスクデバイスとして表すことができます。いったんオペレーティングシステムがインストールされ、設定されると、ファイルはスナップショットの準備が完了しています。

FileSnap を使用した仮想マシンに対する書き込みを集中的に行うアプリケーション の最適化

仮想マシンの作成で書き込みに集中するタスクが実行される場合、かなりの量の共有解除が発生する可能性があります。Veritas は遅延コピーオンライトを有効にすることによって、パフォーマンスを最適化することをお勧めします。遅延コピーオンライトを有効化できない使用事例では、慎重に計画することで、共有解除の発生を減少させることができます。共有解除を減少させる最も簡単な方法は、アプリケーションデータをブートイメージ以外のファイルに分離することです。アプリケーションの性質によって分離できない場合は、次の例に類似した処理を行うことができます。

ブートイメージとアプリケーションデータに必要なディスク容量が 20 GB であると仮定します。これから、オペレーティングシステムが使うのは 4 GB のみで、残りの 16 GB はアプリケーションが書き込む領域です。仮想マシンの各インスタンスに必要なデータまたはバイナリには、共有エクステントの最初の 4 GB の部分を使えます。ほとんどが書き込みの 16 GB の部分で行われると想定されるので、次のコマンドに示すように、領域の 16 GB が共有されないようにマスターイメージを割り当てる必要があります。

```
  # touch /vdi_images/master_image
  # /opt/VRTS/bin/setext -r 4g -f chgsize /vdi_images/master_image
  # dd if=/dev/zero of=/vdi_images/master_image seek=20971520 ¥
bs=1024 count=1
```

最後のコマンドはファイルの末尾に 20 GB の空白部分を作成します。空白部分にはエクステントが割り当てられていないので、空白部分への書き込みは共有が解除されている必要はありません。

FileSnaps を使用してデータの複数のコピーを瞬時に作成する

レポートの生成、マイニング、およびテストのために運用データの 1 つ以上のコピーを作成することは一般的に行われます。この場合、データのコピーは最新データによって頻繁に更新され、1 つ以上のデータのコピーが常に存在します。FileSnap は、複数のコピーを即座に作成するために使うことができます。元のデータを使うアプリケーションでは、更新時に発生するデータの共有解除によってパフォーマンスヒットがわずかに生じる場合があります。

FileSnap の実行例

FileSnap の実行例は次のとおりです。

```
# vxfilesnap tfile1 stfile1
# ls -ltr
total 1108
drwxr-xr-x   2 root     root           96  Jul  6 00:41 lost+found
-rw-r--r--   1 root     root       282686  Jul  6 00:43 tfile1
```

```
-rw-r--r--   1 root     root      282686  Jul  6 00:44 stfile1
# ls -ltri
total 1108
3 drwxr-xr-x   2 root     root          96 Jul  6 00:41 lost+found
4 -rw-r--r--   1 root     root      282686 Jul  6 00:43 tfile1
5 -rw-r--r--   1 root     root      282686 Jul  6 00:44 stfile1
```

fsadm -S shared、du、および df コマンドの論理サイズ出力の比較

fsadm -S shared、du、および df コマンドは、FileSnap のサイズをそれぞれ異なる値で報告します。fsadm -S shared コマンドは、このサイズが「論理サイズ」(消費されている論理領域 (KB)) として表示し、さらに排他的ブロックと共有ブロックの両方の内訳が表示されます。この値は、ファイルシステムに共有ブロックがない場合に必要な実際のディスク容量を表します。fsadm -S shared コマンドから返される値は、du -sk コマンドの出力とは異なります。du コマンドは、VxFS 構造化ファイルが消費するブロックを追跡しないためです。その結果、du -sk コマンドの出力は、fsadm -S shared コマンドから報告される論理サイズ出力より小さくなります。

次の例は、fsadm -S shared、du、および df コマンドの出力を示します。

```
# mkfs -t vxfs /dev/vx/rdsk/dg/vol3
version 16 layout
134217728 sectors, 67108864 blocks of size 1024, log size 65536 blocks
rcq size   4096 blocks
largefiles supported
maxlink    supported

# mount -t vxfs /dev/vx/dsk/dg/vol3 /mnt

# df -k /mnt
Filesystem           1K-blocks    Used   Available Use% Mounted on
/dev/vx/dsk/dg1/vol3 52428800     83590  49073642  1%   /mnt

# /opt/VRTS/bin/fsadm -S shared /mnt
Mountpoint    Size(KB)   Available(KB)   Used(KB)   Logical_Size(KB) Space_Saved(KB)
/mnt          52428800   49073642        83590      83590            0

# du -sk /mnt
0       /mnt

# dd if=/dev/zero of=/mnt/foo bs=1024 count=10
```

```
10+0 records in
10+0 records out
10240 bytes (10 kB) copied, 0.018901 seconds, 542 kB/s

# vxfilesnap /mnt/foo /mnt/foo.snap

# df -k /mnt
Filesystem           1K-blocks      Used   Available Use% Mounted on
/dev/vx/dsk/dg1/vol3 52428800       83600  49073632  1%   /mnt

# /opt/VRTS/bin/fsadm -S shared /mnt
Mountpoint    Size(KB)   Available(KB)   Used(KB)   Logical_Size(KB) Space_Saved(KB)
/mnt          52428800   49073632        83600      83610            10

# du -sk /mnt
20      /mnt
```

29

スナップショットファイルシステムの管理

この章では以下の項目について説明しています。

スナップショットファイルシステムのバックアップ

スナップショットファイルシステムの作成後、スナップショットによって、データの一貫性がとれたバックアップがスナップファイルシステム内に維持されます。

標準ファイルシステムツリーのバックアップを作成するバックアッププログラム（cpio など）は、変更しなくてもスナップショット上で使えます。これは、スナップショットによってスナップファイルシステムと同じデータが使われるからです。一方、ファイルシステムのディスク構造にアクセスするバックアッププログラム（vxdump など）はある程度の変更を加えることによって、スナップショットファイルシステムを処理することができます。

VxFS ユーティリティは、スナップショットファイルシステムを認識し、ユーティリティの動作を変更して、通常のファイルシステムに対して操作を実行するように、スナップショットファイルシステムに対して操作を実行します。通常、RAW ディスクイメージを読み取る他のバックアッププログラムは、バックアップの手順を変更しないとスナップショットを処理できません。

このような他のバックアッププログラムを使うには、fscat コマンドを使って、ファイルシステム全体の RAW イメージを取得します。このイメージは、スナップショット作成時のスナッ

プファイルシステムを含むディスクデバイスの dd コマンドが取得したイメージと同一です。snapread ioctl システムコールは、read システムコールの引数と同様の引数をとり、スナップショット作成時のスナップファイルシステムを含むディスクデバイスで read を実行して取得できるものと同じ結果を返します。ただし、どちらの場合も、スナップショットには、すべてのトランザクションが完了している状態のスナップファイルシステムと一致するイメージが保存されます。これは、スナップショットはスナップファイルシステム全体を瞬時に読み取ったことに等しいことを意味しています。これは、使用中のファイルシステムのディスクデバイスで dd コマンドまたは read コマンドが取得する結果と著しく異なります。

スナップショットファイルシステムのパフォーマンス

スナップショットファイルシステムは、スナップファイルシステムへの書き込みを禁止にして、スナップショットのパフォーマンスを最適化します。スナップショットファイルシステムからの読み取りは、通常、標準 VxFS ファイルシステムの読み取りスループットに近い速度で実行されます。

スナップショットの使用は、一般にスナップファイルシステムからの読み取りパフォーマンスには影響を与えません。ただし、スナップファイルシステムへの書き込みには、スナップショットを使わない場合に比べて、平均 2、3 倍の時間がかかります。これは、データブロックへの最初の書き込み時において、もとのデータの読み取り、スナップショットへの更新前データの書き込み、さらにスナップファイルシステムへの新しいデータの書き込みが必要になるためです。同一のスナップファイルシステムに複数のスナップショットが存在する場合は、書き込み速度がさらに低下します。ただし、初回の書き込み後のインテントログへの書き込みや i ノードの更新などの操作は通常の速度で処理されます。書き込み速度が低下するのはデータブロックへの初回の書き込み時のみです。

スナップファイルシステムが使用中である場合は、スナップファイルシステムに関連付けられたすべてのディスク I/O でスナップショットの読み取り速度が低下するため、スナップショットファイルシステムからの読み取りに影響があります。

スナップショット全体への影響は、アプリケーションの読み書き率および I/O 操作回数によって異なります。たとえば、スナップファイルシステム上でオンライントランザクション処理（OLTP）のような作業負荷がかかるディスク I/O を実行するデータベースアプリケーションでは、スナップショットを作成していないファイルシステムと比べて、約 15 から 20% の速度の低下が測定されています。

スナップショットファイルシステムのディスク構造について

スナップショットファイルシステムは次のもので構成されています。

- スーパーブロック
- ビットマップ

- ブロックマップ
- スナップファイルシステムからコピーされたデータブロック

次の図に、スナップショットファイルシステムのディスク構造を示します。

図 29-1 スナップショットディスク構造

スーパーブロックは、標準 VxFS ファイルシステムのスーパーブロックに類似しています。ただし、マジックナンバーが異なっており、多くのフィールドは適用できません。

ビットマップには、スナップファイルシステムのデータブロックごとに 1 ビットが用意されています。初期設定では、ビットマップエントリはすべて 0 です。1 は、適切なデータブロックがスナップファイルシステムからスナップショットにコピーされたことを示します。この場合、データブロックマップの適切な位置でコピーされたデータブロックが参照されます。

ブロックマップには、スナップファイルシステムのデータブロックごとにエントリが 1 つ用意されています。初期設定では、エントリはすべて 0 です。データブロックがスナップファイルシステムからスナップショットにコピーされると、スナップファイルシステムからコピーされたデータを保持するスナップショットファイルシステム上のブロック番号を保存するため、ブロックマップ上の対応するエントリが変更されます。

データブロックには、スナップファイルシステムからコピーされたデータがデータブロック領域の先頭から保存されます。

スナップショットと Storage Checkpoint の相違点

スナップショットと Storage Checkpoint はともに特定時点でのファイルシステムのイメージを作成し、変更されたデータブロックのみを更新します。ただし、この 2 つのテクノロジには、次のような明確な相違が存在します。

表 29-1 スナップショットと Storage Checkpoint の相違点

スナップショット	Storage Checkpoint
ストレージ用に別のデバイスを必要とする。	元のファイルシステムと同じデバイス上に存在する。
読み取り専用である。	読み取り専用、または読み書き可能である。
一時的なものである。	永続的なものである。
マウント解除後に消失する。	Storage Checkpoint 自体にマウントできる。
変更されたデータブロックをファイルシステムレベルで追跡する。	変更されたデータブロックをファイルシステムの各ファイルレベルで追跡する。

Storage Checkpoint は、Veritas の BLI バックアップと Storage Rollback を有効にするテクノロジです。この 2 つの機能はデータベースのバックアップを作成するために広く使われています。

p.737 の「Storage Checkpoint について」を参照してください。

スナップショットファイルシステムの作成

スナップショットファイルシステムを作成するには、mount コマンドの -o snapof= オプションを使います。マウントされるデバイスのディスクラベルからデバイスサイズを識別できない場合、またはデバイス全体のサイズよりも小さいサイズが必要な場合は、-o snapsize= オプションが必要となることもあります。

スナップショットファイルシステムが存在する間は、スナップファイルシステムへの書き込みがある可能性があるため、スナップファイルシステム上のすべてのデータブロックを保存できるサイズのスナップショットファイルシステムを作成する必要があります。コピーされたデータを保存するための領域が不足すると、スナップショットファイルシステムは無効になり、その後のスナップショットファイルシステムへのすべてのアクセスが失敗します。

夜間や週末など使用量が少ない間、スナップショットが必要とするデータブロックは、通常、スナップファイルシステムのデータブロックのわずか 2 から 6% です。使用量が増えると、通常のファイルシステムのスナップショットは、スナップファイルシステムのデータブロックの 15% を必要とします。通常のシステムは、1 日でデータブロックの 15% を使いません。この割合は大容量ファイルシステムほど低く、小容量ファイルシステムほど高い傾向があります。ファイルシステムの使用率とバックアップの所要時間に基づいて、スナップショットにデータブロックを割り当てることができます。

警告: スナップショットに使うデバイスに存在するデータは上書きされます。

スナップショットファイルシステムを作成するには

◆ -o snapof= オプションを指定してファイルシステムをマウントします。

```
# mount -t vxfs -o ro,snapof=/ ¥
snapped_mount_point_mnt, snapsize=snapshot_size ¥
/dev/vx/dsk/diskgrp/volume snapshot_mount_point
```

次の例では、vxdump ユーティリティにより /dev/rdsk/fsvol/vol1 が /backup/home としてマウントされたスナップショットであることを認識し、適切な処理を実行してマウントポイントからスナップショットデータを取得します。

次に、スナップショットマウントポイント /backup/home のボリューム管理ツールボリューム上に存在するスナップショットを使って、/home という 30 万個のデータブロックが存在するファイルシステムの標準的なバックアップを作成する例を示します。

スナップショットファイルシステムを使ってバックアップを作成する例

1 cpio を使って、先週変更されたファイルに対してバックアップを実行するには、次を実行します。

```
# mount -t vxfs -o snapof=/home,snapsize=100000 ¥
/dev/vx/dsk/fsvol/vol1 /backup/home
# cd /backup
# find home -ctime -7 -depth -print | cpio -oc > /dev/st1
# umount /backup/home
```

2 /dev/vx/rdsk/fsvol/vol1 のレベル 3 のバックアップを実行し、現在のディレクトリで変更されたファイルを収集するには、次を実行します。

```
# vxdump 3f - /dev/vx/rdsk/fsvol/vol1 | vxrestore -xf -
```

3 ディスク /dev/vx/rdsk/fsvol/vol1 に存在する /home の完全バックアップを実行し、dd を使って、vxdump を使うテープデバイスへの出力のブロッキングを制御するには、次を実行します。

```
# mount -t vxfs -o snapof=/home,snapsize=100000 ¥
/dev/vx/dsk/fsvol/vol1 /backup/home
# vxdump f - /dev/vx/rdsk/fsvol/vol1 | dd bs=128k > /dev/st1
```

8

Storage Foundation Cluster File System High Availability を使用したストレージの最適化

30

Storage Foundation Cluster File System High Availability のストレージ最適化ソリューションについて

この章では以下の項目について説明しています。

シンプロビジョニングについて

シンプロビジョニングは、ストレージをオンデマンドで割り当て、再生することによって、ストレージの使用を最適化するストレージアレイの機能です。シンプロビジョニングを使用して、ストレージが必要となったときにだけ、アレイは空きストレージのプールからストレージをアプリケーションに割り当てます。シンプロビジョニングは、利用可能なアレイ容量に対して低い使用率の問題を解決するものです。管理者は、アプリケーションが必要とする

ストレージ容量を見積もる必要がありません。代わりにシンプロビジョニングを使うと、大規模のシン LUN または再生可能なシン LUN をホストにプロビジョニングできます。アプリケーションでデータに書き込むときに、物理ストレージはアレイの空きプールからシンプロビジョニングの LUN に割り当てられます。

2 つのタイプのシンプロビジョニング LUN とは、シン対応 LUN、シン再生可能 LUN です。両方のタイプの LUN では、空きプールからストレージを必要に応じて割り当てる機能を提供します。たとえば、ファイルシステムがファイルを作成または変更するときに、ストレージが割り当てられます。ただし、ファイルが削除されるときには、このストレージは空きプールに解放されません。したがって、データが削除された未使用の空き領域がファイルシステムに含まれるにつれて、シンプロビジョニングの LUN は時間の経過と共に「ファット」になることができます。シン再生可能な LUN は、使用されたことがあるストレージを空きストレージのプールに解放する機能を使用して、この問題に対応します。この操作はシンストレージ再生と呼ばれます。

シン再生可能な LUN は、自動的に再生を実行しません。LUN を使用するサーバーは再生を開始する必要があります。管理者は手動再生または、スケジュール設定された再生操作を開始できます。

Storage Foundation Cluster File System High Availability は、シンプロビジョニングとシン再生をサポートする機能と、シンプロビジョニングアレイのストレージの使用を最適化する機能をいくつか提供します。

p.772 の「SmartMove について」を参照してください。

Storage Foundation Cluster File System High Availability のシン最適化ソリューションについて

シンストレージやシンプロビジョニングなどのようなアレイベースのオプションは、ストレージ管理者がストレージ管理の課題に対応するために役立ちます。これらの課題には、ストレージのプロビジョニング、ストレージ利用率を最大化するためのデータ移行、最適化されたストレージ利用率の維持などがあります。Storage Foundation Cluster File System High Availability の複数の機能は、これらの課題を解決するためにアレイ機能と連携して動作します。

表 30-1 に、シンストレージに関連する Storage Foundation Cluster File System High Availability の機能と利点を示します。

表 30-1 Storage Foundation Cluster File System High Availability のシンストレージソリューション

機能	説明	メリット
SmartMove	SmartMove 機能では、Veritas File System で使用されているブロックのみ移動またはコピーします。	シンストレージの使用を最大化します。 p.772 の「SmartMove について」を参照してください。 コピー操作のパフォーマンスを向上させます。 ファット LUN からシンプロビジョニング LUN への移行を有効化します。 p.777 の「シンプロビジョニングへの移行」を参照してください。
シンディスクの検出	Storage Foundation Cluster File System High Availability ではシンストレージデバイスの検出を行うことができます。	シンディスクのシン属性を認識し、表示します。
シン再生	シン再生コマンドを使用して、ファイルシステム、ディスク、ディスクグループ、エンクロージャレベルの領域を再生できます。	ストレージ使用と保存の機能を向上させます。 p.773 の「シン再生機能について」を参照してください。

SmartMove について

Storage Foundation Cluster File System High Availability は、移動操作とコピー操作を最適化するための SmartMove ユーティリティを提供します。SmartMove ユーティリティは、VxFS（Veritas File System）での VxVM（Veritas Volume Manager）ストレージに関する知識を活用します。VxFS は VxVM に、どのブロックにデータがあるか通知します。VxVM がデータのコピー操作または移動操作を実行するときに、SmartMove はファイルシステムによって使用されたブロックをコピーのみまたは移動のみする操作を有効にします。これによってコピーされるブロック数が減るため、この機能は同期、ミラー化、コピー操作の処理効率が向上させます。SmartMove は、VxVM ボリュームにマウントされる VxFS ファイルシステムでのみ連携します。ファイルシステムがマウントされない場合、ユーティリティからファイルシステムの使用状況を見ることはできません。

SmartMove はインスタントスナップショットがあるボリュームには使用しません。

また、ファットストレージからシンプロビジョニングのストレージにデータを移行するには、SmartMove 操作も使用できます。SmartMove はファイルシステムで使用中のブロックのみコピーするため、結果として移行プロセスはシンプロビジョニング LUN になります。

シンプロビジョニングの SmartMove

Storage Foundation Cluster File System High Availability ではシンプロビジョニングの SmartMove 機能を使用できます。SmartMove によって、ファットストレージからシンストレージにストレージを移行できるようになります。また、シンプロビジョニングのインテントを維持する機能を提供します。

SmartMove を使用しない場合、ディスク間の同期では、VxFS（Veritas File System）と VxVM（Veritas Volume Manager）に割り当てられているストレージ全体がコピーされます。ボリューム、プレックス、サブディスクを同期したり再同期したりすると、シンディスクに未使用領域が割り当てられる場合があります。そのうちに、正常な操作によってストレージが厚くなってしまいます。SmartMove を使用すると、ディスクを同期するときにファイルシステムレベルで実際に使用中であるブロックのみがコピーされます。この動作によって、ディスクが同期または再同期されても未使用領域が割り当てられないようにします。ディスクはシン状態のままです。

SmartMove 機能は、デフォルトですべてのディスクで有効になっています。シンプロビジョニングを利用するには、SmartMove を少なくともシンディスクで有効にする必要があります。

シン再生機能について

Storage Foundation Cluster File System High Availability はシン再生対応アレイの未使用ストレージの再生をサポートします。Storage Foundation Cluster File System High Availability はシン再生をサポートする LUN を自動的に検出します。

VxFS（Veritas File System）ファイルシステムは、シン対応アレイによってサポートされる VxVM（Veritas Volume Manager）ボリュームにマウントできます。VxVM ボリュームのサイズは、空きストレージプールによってサポートされる仮想サイズです。ファイルが作成または変更されるときに、ストレージはアレイからファイルシステムに物理的に割り当てられます。ファイルシステムのファイルが削除された場合、またはファイルサイズが縮小された場合、領域はファイルシステムでの使用から解放されます。ただし、この領域は物理割り当てからは削除されません。時間の経過につれて、ファイルシステムに割り当てられた物理領域は、ファイルシステムによって使われる実際の領域より大きくなります。最終的にシン LUN は「ファット」LUN になり、割り当てられた物理領域は LUN のサイズに近づきます。

シン再生機能はこの未使用領域を空きプールに解放する機能を提供します。Storage Foundation Cluster File System High Availability は VxFS の割り当てテーブルを使って未使用のブロックを識別します。VxVM は、未使用ブロックについてのこの情報をディスクレベルにマップし、空きプールにそれらのブロックを戻すために VxVM を有効にしま

す。VxFS ファイルシステムがマウントされない場合、VxVM からファイルシステムの使用状況を見ることはできません。したがって、再生を実行するときにはファイルシステムがマウントされている状態であることが重要です。再生の操作は、ディスクグループ、LUN、エンクロージャ、ファイルシステムで行うことができます。

VxVM は、ボリュームやプレックスを削除するときに、領域を自動的に再生します。領域がアレイのレベルですぐに再生されないように、自動再生は非同期です。ディスクは保留中の再生としてマークされます。再生が完了するまで、VxVM からディスクを削除できません。自動再生のタイミングと頻度を制御できます。

TRIM 操作によるソリッドステートデバイス（SSD）の領域の再生について

ファイルの作成および削除を行うファイルシステムは、ストレージブロックを新しい内容で上書きして頻繁に再利用します。ソリッドステートドライブ（SSD）デバイスがストレージのブロックを上書きする際は、先にブロックを消去します。この動作により、未使用または消去されたブロックへの書き込みと比較した場合に、以前使用したブロックへの書き込みでパフォーマンスコストが発生します。このコストを避けるため、TRIM 操作はブロック使用されておらず、消去可能な SSD を通知します。今後の SSD への I/O の書き込みのパフォーマンスを向上するためにブロックの再利用が必要になる前に、SSD は未使用のブロックを消去します。また、未使用のブロックが消去されるため、TRIM 操作はウェアレベリングおよび断片化を低減します。ガーベジコレクションまたはクリーニングサイクルの間、未使用データは移動しません。

SFCFSHA はサポート対象デバイスにのみ TRIM 操作を行います。詳しくは、Veritas のハードウェア互換性リスト（HCL）を参照してください。

https://www.veritas.com/support/en_US/article.000126344

SFCFSHA コンポーネント、VxFS（Veritas File System）および VxVM（Veritas Volume Manager）は、有効なデータを含んでいないブロックの解放に TRIM 操作を使用します。TRIM 機能はシン再生に類似していて、同じコマンドで実行されます。デフォルトの SFCFSHA 再生コマンドは、SSD の TRIM とThin Reclaimable LUN のシン再生を実行します。SSD と Thin Reclaimable LUN の両方を使用するボリュームおよびファイルシステムでは、SFCFSHA が TRIM 操作のみ、シン再生のみ、または両方のいずれかを実行するように選択できます。

p.790 の「ディスク、ディスクグループ、エンクロージャの領域の再生」を参照してください。

p.788 の「ファイルシステムの領域の再生」を参照してください。

SSD についての情報を表示するには、`vxdisk -o ssd list` コマンドを使います。SFCFSHA は、SSD の VxFS（Veritas File System）ファイルシステムのディスク領域の使用状況も検出および表示できます。VxFS ファイルシステムは、VxVM（Veritas Volume

Manager) 上にマウントする必要があります。`vxdisk -o ssd -o fssize list` コマンドを使います。

`vxdisk` (1M) マニュアルページを参照してください。

シン再生 LUN の領域を再生する状況の確認

シン LUN が Veritas Volume Manager のディスクとして使われると、領域はアプリケーション書き込みでのみ割り当てられます。ストレージ領域は、ファイルが作成されてファイルシステムに書き込まれると、空きプールから割り当てられます。ただし、データがファイルシステムから削除されても、このストレージは自動的に空きプールには解放されません。その結果、すべてのシン LUN は、無駄なストレージ (割り当てられているがストレージはアプリケーションデータをサポートしていないストレージ) の量が増えるにつれて、時間の経過とともに厚くなりがちです。

ストレージの管理者は、シン再生をいつトリガするか判断する必要があります。シン再生のプロセスは、ファイルシステムのサイズと断片化などさまざまな要因によっては、時間がかかる場合があります。再生する領域のサイズと再生操作にかかる時間のバランスを考えて決定する必要があります。

次の注意事項が適用されます。

- VxVM ボリュームにマウントされる VxFS ファイルシステムでは、再生が適切かどうかを判断するために、ファイルシステムの使用状況と実際の物理割り当てサイズを比較します。ファイルシステムの使用状況が物理割り当てサイズより大幅に小さい場合、再生できる領域が多くあることを示します。その場合、ファイルシステムの再生をトリガできます。ファイルシステムの使用状況が物理割り当てサイズに近い場合、物理割り当てがほとんど使用されていることを示します。その場合、再生をトリガしない方がよいかもしれません。
  p.786 の「シン再生 LUN での VxFS ファイルシステムの使用状況の表示」を参照してください。
- アレイでは、ストレージプールの使用状況が一定のしきい値に到達したら通知を表示する場合があります。ストレージプールでより多くの領域を解放するために Storage Foundation Cluster File System High Availability で領域を再生するかどうかを判断できます。
- 削除されたボリュームは自動的に再生されます。自動再生のスケジュールをカスタマイズできます。
  p.794 の「自動再生の設定」を参照してください。

自動再生の動作

シン再生可能なアレイでは、使用されなくなったストレージはアレイによって再生する必要があります。Storage Foundation Cluster File System High Availability は、特定の管理操作に対して、次のようにアレイの領域を自動的に再生します。

- ボリュームの削除。
- ミラーの削除
- ボリュームの縮小。
- ログの削除。
- `init=zero` オプションによるボリュームの作成または拡張。

アレイ上でストレージを再生する処理には、アレイに負荷がかかる可能性があります。アレイへの通常の I/O に影響を与えないように、Storage Foundation Cluster File System High Availability では再生操作を非同期で行います。ディスクには保留中の再生フラグが設定されます。`vxrelocd`(またはリカバリ)デーモンは、今後、再生のマークが付けられているディスクを非同期で再生します。デフォルトでは、`vxrelocd` デーモンは毎日 22:10 に実行され、削除してから 1 日経過したボリュームまたはプレックス上のストレージが再生されます。

再生を保留しているディスクを表示するには、次のコマンドを入力します。

```
# vxprint -z
```

自動再生を設定することで、すぐに再生、または非同期再生をスケジュール設定できます。

p.794 の「自動再生の設定」を参照してください。

ディスク、ディスクグループ、エンクロージャの再生も手動でトリガできます。この操作では、再生が保留としてフラグ設定されているディスクも再生できます。

p.790 の「ディスク、ディスクグループ、エンクロージャの領域の再生」を参照してください。

31

ファットストレージからシンストレージへのデータの移行

この章では以下の項目について説明しています。

- シンストレージへの移行のための SmartMove の使用について
- シンプロビジョニングへの移行

シンストレージへの移行のための SmartMove の使用について

ファット LUN に既存のデータがある場合は、SmartMove 機能を使用してデータをシン LUN に移行できます。移行プロセスでは、VxFS（Veritas File System）が使用中のブロックのみシン LUN にコピーします。SmartMove 機能は、Veritas Volume Manager（VxVM）ボリューム内のデータが含まれるブロックに関する Veritas File System（VxFS）の情報を利用します。このため、移行機能は VxVM ボリュームが VxFS ファイルシステムにマウントされている場合にのみ利用可能です。

シン LUN にデータを移行するには、お勧めする手順に従ってください。

p.777 の「シンプロビジョニングへの移行」を参照してください。

シンプロビジョニングへの移行

SmartMove™ 機能は、従来の LUN からシンプロビジョニングされた LUN への移行や、プロセス内での未使用領域の削除を可能にします。

シンプロビジョニングに移行するには

1 SmartMove 機能が有効になっているかどうかを確認します。

```
# vxdefault list
KEYWORD                         CURRENT-VALUE   DEFAULT-VALUE
usefssmartmove                  all             all
...
```

出力に現在の値が none と表示される場合、すべてのディスクまたはシンディスクの SmartMove を設定します。

p.974 の「SmartMove の設定」を参照してください。

2 新しいシン LUN を既存のディスクグループに追加します。次のコマンドを入力します。

```
# vxdisksetup -i da_name
# vxdg -g datadg adddisk da_name
```

da_name は VxVM 上でのディスクアクセス名です。

3 thinonly または thinrclm 属性を持つ LUN を識別するには、次のように入力します。

```
# vxdisk -o thin list
```

4 新しいシン LUN を新しいプレックスとしてボリュームに追加します。シン LUN で、ミラー化されたボリュームを作成するか既存の LUN にミラーを追加するとき、VxVM はデフォルトでデータ変更オブジェクト(DCO)を作成します。DCO はミラーの完全な再同期化を不要にすることにより、シン LUN が「ファット」になることを防ぐのに役立ちます。

メモ: SmartMove 機能の利点を得るには、VxFS ファイルシステムがマウントされている必要があります。

LUN を追加するには次の方法があります。

- vxassist コマンドのデフォルト設定を使用します。

  ```
  # vxassist -g datadg mirror datavol da_name
  ```

- 完了を高速化するには、vxassist コマンドオプションを指定します。-b オプションを指定すると、ブロックがバックグラウンドでコピーされます。次のコマンドは I/O スループットを向上させます。

  ```
  # vxassist -b -oiosize=1m -t thinmig -g datadg mirror ¥
    datavol da_name
  ```

  コマンドの状態を表示するには、vxtask コマンドを使用します。

```
# vxtask list
TASKID PTID TYPE/STATE    PCT   PROGRESS
211         ATCOPY/R 10.64% 0/20971520/2232320 PLXATT vol1 vol1-02 xivdg smartmove
212         ATCOPY/R 09.88% 0/20971520/2072576 PLXATT vol1 vol1-03 xivdg smartmove
219         ATCOPY/R 00.27% 0/20971520/57344 PLXATT vol1 vol1-04 xivdg smartmove

# vxtask monitor 211
TASKID PTID TYPE/STATE    PCT   PROGRESS
211         ATCOPY/R 50.00% 0/20971520/10485760 PLXATT vol1 vol1-02 xivdg smartmove
211         ATCOPY/R 50.02% 0/20971520/10489856 PLXATT vol1 vol1-02 xivdg smartmove
211         ATCOPY/R 50.04% 0/20971520/10493952 PLXATT vol1 vol1-02 xivdg smartmove
211         ATCOPY/R 50.06% 0/20971520/10498048 PLXATT vol1 vol1-02 xivdg smartmove
211         ATCOPY/R 50.08% 0/20971520/10502144 PLXATT vol1 vol1-02 xivdg smartmove
211         ATCOPY/R 50.10% 0/20971520/10506240 PLXATT vol1 vol1-02 xivdg smartmove
```

- システムパフォーマンスへの影響を軽減するには、vxassist コマンドオプションを指定します。次のコマンドは完了までに時間がかかります。

  ```
  # vxassist -oslow -g datadg mirror datavol da_name
  ```

5 任意で、古い LUN を削除する前に新しい LUN のパフォーマンスをテストすることもできます。

パフォーマンスをテストするには、次の手順を使います。

- どのプレックスがシン LUN に対応するかを確認します。

  ```
  # vxprint -g datadg
  ```

```
TY NAME              ASSOC          KSTATE   LENGTH     PLOFFS  STATE     TUTIL0 PUTIL0
dg datadg            datadg           -        -          -       -         -       -
dm THINARRAY0_02     THINARRAY0_02    -        83886080   -       -         -       -
dm STDARRAY1_01      STDARRAY1_01     -        41943040   -       -OHOTUSE  -       -

v  datavol           fsgen            ENABLED  41943040   -       ACTIVE    -       -
pl datavol-01        datavol          ENABLED  41943040   -       ACTIVE    -       -
sd STDARRAY1_01-01   datavol-01       ENABLED  41943040   0       -         -       -
pl datavol-02        datavol          ENABLED  41943040   -       ACTIVE    -       -
sd THINARRAY0_02-01  datavol-02       ENABLED  41943040   0       -         -       -
```

この出力例は、シン LUN がプレックス datavol-02 に対応することを示しています。

- すべての読み取りがこれらの LUN から来るように指示します。

  ```
  # vxvol -g datadg rdpol prefer datavol datavol-02
  ```

6 元のシンでない LUN を削除します。

メモ: ! 文字は一部のシェルでの特殊文字です。次の bash シェルの例では、この文字をエスケープしています。

```
# vxassist -g datadg remove mirror datavol ¥!STDARRAY1_01
# vxdg -g datadg rmdisk STDARRAY1_01
# vxdisk rm STDARRAY1_01
```

7 ファイルシステムとボリュームを拡大して、より大きなシン LUN をすべて使うようにします。

```
# vxresize -g datadg -x datavol 40g da_name
```

32

シン再生機能によるシンストレージの保守

この章では以下の項目について説明しています。

シン再生アレイでのストレージの再生

Storage Foundation Cluster File System High Availability は、シン再生可能なアレイおよび LUN の未使用ストレージの再生をサポートします。Storage Foundation Cluster File System High Availability は、VxVM（Veritas Volume Manager）ボリュームにマウントされている VxFS（Veritas File System）ファイルシステムのブロックを再生できます。

シン再生機能は、`thinrclm` 属性がある LUN のみでサポートされます。VxVM はシン対応ストレージアレイからシン再生をサポートする LUN を自動的に検出します。`thin` または `thinrclm` 属性がホストにあることが分かっているデバイスを一覧表示できます。

p.783 の「シン LUN およびシン再生 LUN の識別 」を参照してください。

シン再生をサポートするストレージアレイのリストについて詳しくは、ハードウェア互換性リスト (HCL) を参照してください。

https://www.veritas.com/support/en_US/article.000126344

シン再生はブートデバイスではサポートされません。

シン再生機能は、次のように使うことができます。

- ボリュームが削除されると、領域は自動的に再生されます。この処理は非同期であるため、再生された領域をすぐに確認できない場合があります。
- `vxdisk` コマンドを使用して、ディスクグループ、LUN、エンクロージャの再生操作を実行します。
  p.790 の「ディスク、ディスクグループ、エンクロージャの領域の再生」を参照してください。
- `fsadm` コマンドを使用して、VxFS（Veritas File System）ファイルシステムの再生操作を実行します。
  p.788 の「ファイルシステムの領域の再生」を参照してください。

ディスク、ディスクグループ、またはエンクロージャのシン再生について

Storage Foundation Cluster File System High Availability を使用すると、アプリケーションの I/O を停止せずにシンプロビジョニングアレイ上の未使用領域を再生することができます。Veritas File System（VxFS）のファイルシステムがマウントされている必要があります。

1 つ以上のディスク、ディスクグループ、またはエンクロージャ上でシン再生をトリガできます。再生のプロセスでは、VxFS ファイルシステムにマウントされた VxVM ボリュームの指定されたストレージをスキャンします。各ボリュームで、以前に割り当てられて VxFS ファイルシステムで使用されなくなった領域が分析されます。使用されていない領域が、シンアレイ上のストレージの空きプールに解放されます。VxFS ファイルシステムにマウントされていないボリュームはスキップされます。また、再生プロセスでは、再生保留中とマーク付けされたボリュームやプレックスの領域も解放します。

デフォルトでは、指定したストレージが Solid State Devices（SSD）に存在する場合、再生コマンドは TRIM 操作も実行します。

p.774 の「TRIM 操作によるソリッドステートデバイス（SSD）の領域の再生について」を参照してください。

完全な再生プロセスでは、VxVM ボリュームの外部にある指定されたストレージの空き領域もスキャンします。

シン再生はインスタントスナップショット階層の一部であるボリュームの領域を再生しません。

大量の LUN、エンクロージャ、またはディスクグループに対してシンストレージを再生するときは、シン再生にかなりの時間がかかります。他の時間がかかる操作と同様に、VxVM

は再生操作用のタスクを作成します。vxtaskコマンドを使用すると再生操作を監視できます。

p.793 の「vxtask コマンドを使ったシン再生の監視」を参照してください。

ファイルシステムのシン再生について

Veritas File System（VxFS）はシンストレージ LUN 上の空きストレージの再生をサポートします。空きストレージは fsadm コマンドを使って再生されます。デフォルトの再生または積極的な再生を実行できます。ファイルシステムを長期間使っていて、ファイルシステムで再生を実行する必要がある場合は、積極的な再生を実行することをお勧めします。積極的な再生により、割り当て済みのブロックが縮小され、再生可能な空きブロックが増加します。

fsadm_vxfs（1M）のマニュアルページを参照してください。

シン再生は VxVM ボリュームにマウントされたファイルシステム上でのみサポートされます。

シン再生は RAID5 ボリュームにマウントされたファイルシステムでサポートされません。

Veritas File System は、vxfs_ts_reclaim() API を使って、ファイルシステムの一部の再生もサポートします。

vxfs_ts_reclaim（3）マニュアルページと『Veritas File System プログラマーズリファレンスガイド』を参照してください。

メモ: シン再生は遅いプロセスで、ファイルシステムのサイズによっては完了までに数時間を要することがあります。シン再生によって 100% の空き領域の再生が保証されるわけではありません。

Veritas Volume Manager（VxVM）コマンド vxdisk reclaim を使うときに、vxtask list コマンドを使って、シン再生プロセスの進行状況を追跡できます。

vxtask（1M）と vxdisk（1M）のマニュアルページを参照してください。

VxVM コマンドを使ってシン再生を管理できます。

シン LUN およびシン再生 LUN の識別

DMP（Dynamic Multi-Pathing）を使用して、Storage Foundation Cluster File System High Availability は、ホストで thin または thinrclm として認識されるシンデバイスを自動的に検出します。DMP は、ベンダ固有のシン属性を認識するため、および thin または thinrclm に従ってデバイスを要求するために、Veritas ASL（Array Support Library）を使用します。

thin として分類されたデバイスでは、シンプロビジョニングが可能です。Veritas のシン再生は、thinrclm 属性が設定されているデバイスのみで稼働します。シン再生を実行する前に、システムが LUN を thinrclm LUN として認識するかどうかを確認します。

thin または thinrclm 属性を持っていると認識されているホスト上のデバイスを識別するには、vxdisk -o thin list コマンドを使用します。vxdisk -o thin list コマンドは、ディスクのサイズ、アレイで割り当てられる物理領域を報告します。

thin LUN と thinrclm LUN を識別するには

◆ ホストにローカルに認識されているすべての thin または thinrclm LUN を識別するには、次のコマンドを使用します。

```
# vxdisk -o thin list

DEVICE          SIZE(MB) PHYS_ALLOC(MB) GROUP   TYPE
RECLAIM_CMD
xiv0_6695       16384    30             dg1     thinrclm WRITE_SAME
xiv0_6696       16384    30             dg1     thinrclm WRITE_SAME
xiv0_6697       16384    30             dg1     thinrclm WRITE_SAME
xiv0_6698       16384    30             dg1     thinrclm WRITE_SAME
xiv0_6699       16384    30             dg1     thinrclm WRITE_SAME
3pardata0_5074  2048     2043           vvrdg   thinrclm WRITE_SAME
3pardata0_5075  2048     2043           vvrdg   thinrclm WRITE_SAME
3pardata0_5076  2048     1166           vvrdg   thinrclm WRITE_SAME
3pardata0_5077  2048     2043           vvrdg   thinrclm WRITE_SAME
3pardata0_5081  2048     1092           vvrdg   thinrclm WRITE_SAME
```

出力では、SIZE 列にディスクのサイズが表示されます。PHYS_ALLOC 列には、アレイ側の物理割り当てが表示されています。TYPE には、アレイが thin または thinrclm のどちらであるか示します。RECLAIM_CMD カラムには、DMP が使う再生方法が表示されます。

vxdisk(1M)マニュアルページを参照してください。

再生コマンドに関する詳細情報の表示

Dynamic Multi-Pathing(DMP)では、複数のアレイレベルの再生コマンド、UNMAP、WRITE_SAME、TRIM、PTRIM がサポートされています。各アレイに対する ASL(Array Support Library)には、アレイに対しサポートされているものから最適な方法が使用されます。DMP では、ベンダーがパフォーマンス分析に基づいて再生方法を提案または選択する優先再生方法が使用されます。

DMP で使用される再生方法を変更することはできません。ただし、DMP で選択された再生コマンドに関する情報を表示することができます。

DMP で再生要求の作成に使用されるその他の再生属性に関する情報を表示できます。再生属性はベンダーによって異なります。

シン再利用方法に関する情報を表示するには

◆ デバイスに対するシン再利用方法に関する詳細情報を表示するには、次のコマンドを使用します。

```
# vxdisk -p list xiv0_6699

DISK                : xiv0_6699
VID                 : IBM
UDID                : IBM%5F2810XIV%5F0E95%5F1A2B
TP_PREF_RCLMCMD     : write_same
TP_RECLM_CMDS       : write_same, unmap
TP_ALLOC_UNIT       : 1048576
TP_MAX_REC_SIZE     : 268435456
TP_LUN_SHIFT_OF     : 0
SCSI_VERSION        : 5
SCSI3_VPD_ID        : 001738000E951A2B
REVISION            : 10.2
.
.
.
LUN_SIZE            : 33554432
NUM_PATHS           : 4
STATE               : online
```

次のフィールドには、再生属性に関する情報が示されます。

TP_PREF_RCLMCMD	このアレイに対する優先再生方法。
TP_RECLM_CMDS	下位デバイスでサポートされる再生方法。
TP_ALLOC_UNIT	デバイスのシンプロビジョニングのためのアロケーションユニットのサイズ(バイト単位)。
TP_MAX_REC_SIZE	デバイスの再生 I/O の最大サイズ(バイト単位)。
TP_LUN_SHIFT_OF	ベンダーが LUN に対する初期オフセットを移動する値(バイト単位)。この値を設定して、再生要求を TP_ALLOC_UNIT に合わせます。

シン再生 LUN での VxFS ファイルシステムの使用状況の表示

Storage Foundation Cluster File System High Availability は、thin デバイスまたは thinrclm デバイス上の VxFS（Veritas File System）ファイルシステムのディスク領域の使用状況を検出および表示できます。VxFS ファイルシステムは、VxVM（Veritas Volume Manager）上にマウントする必要があります。使用状況の情報は、ファイルシステムのシン再生をいつ実行するか決定する場合に役立ちます。

p.775 の「シン再生 LUN の領域を再生する状況の確認」を参照してください。

VxVM ボリュームに現在マウントされている VxFS ファイルシステムの LUN ディスク領域ごとの使用状況を報告するには、vxdisk -o thin -o fssize list コマンドを使用します。このコマンドでは、現在ファイルを含み、VxFS ファイルシステムで現在使用中であるディスク領域の量を表示します。使用状況には、ファイルシステムに割り当てられているがファイル削除によって解放された領域は含まれません。複数のマウントされた VxFS ファイルシステムによってデバイスが使用される場合、ファイルシステムの使用状況の列には、統合された領域の使用状況が表示されます。-o fssize オプションは、ファイルシステム領域の使用量を基盤となる LUN にマップします。ディスク領域の使用量の統計値は、計算に使ったベースユニットが理由で、ほかのユーティリティによって報告された使用量と若干異なる場合があります。

次の制限事項は、ファイルシステムの使用状況を表示するコマンドに適用されます。

- -o fssize オプションでは、キャッシュオブジェクトまたはインスタントスナップショットによって使用される領域は表示しません。
- RAID 5 形式はサポートされていません。
- VxFS ファイルシステムがマウントされていない、またはデバイスにマウントされた VxFS ファイルシステムとマウント解除された VxFS ファイルシステムが両方ともある場合は、情報は表示されません。ファイルシステムの使用状況（FS_SIZE）の列にはハイフン（-）が表示されます。

すべての thin LUN または thinrclm LUN のサイズと使用状況を表示できます。または、エンクロージャ名またはデバイス名を指定できます。1 つ以上のデバイスまたはエンクロージャを指定すると、コマンドには指定されたデバイス領域の使用状況のみが表示されます。指定されたデバイスが thin デバイスまたは thinrclm デバイスでない場合、デバイスはリスト表示されますが、FS_SIZE 列にはハイフン（-）が表示されません。

VxFS ファイルシステムが複数のデバイスに渡る場合、ファイルシステム全体の使用状況を表示するには、すべてのデバイスを指定する必要があります。一部のデバイスのみ指定する場合、ファイルシステムの使用状況は不完全な状態です。コマンドは、指定されていないデバイスのファイルシステムの使用状況は無視します。

メモ: コマンドは、ファイルシステムのサイズ、断片化のレベル、他の要因によって、完了に長時間かかる場合があります。コマンドは、vxtaskコマンドで監視できるタスクを作成します。

コマンド出力には次の情報が表示されます。

DEVICE	エンクロージャに基づく命名規則(EBM)または OS に基づく命名規則(OSN)での VxVM ディスクの名前。
SIZE	ディスクのサイズ。ファイルシステムに示されるサイズです。このサイズは、デバイスで使用される実際の物理領域ではなく、仮想サイズを表します。
PHYS_ALLOC	アレイ側の物理的な割り当て。このサイズは、ファイルシステムにアプリケーション書き込みとして割り当てられる物理領域を表します。ファイルが削除または変更されると、物理領域は再生が実行されるまで割り当てられた状態のままになります。この場合、物理サイズには未使用の領域が含まれます。
FS_SIZE	VxFS(Veritas File System)ファイルシステムが使用する物理領域。VxFS ファイルシステムは VxVM ボリュームにマウントする必要があります。この情報は、シンプロビジョニング対応(thin)LUN またはシン再生対応(thinrclm)LUN のみに表示されます。
GROUP	ディスクを含むディスクグループ。
TYPE	シンデバイスのタイプ - シンプロビジョニング対応(thin)またはシン再生対応(thinrclm)。vxdisk -o thin listコマンドは、コマンドラインで明示的にディスク名を指定した場合のみファットディスクを表示します。
RECLAIM_CMD	DMP が使う再生方法。

すべてのシン LUN でファイルシステムの使用状況を表示するには

- システムでローカルに認識されているすべての thin LUN または thinrclm LUN のファイルシステムの使用状況を表示するには、次のコマンドを使用します。

```
$ vxdisk -o thin,fssize [-u unit] list
```

unit は表示用のサイズ単位です。次に例を示します。

```
$ vxdisk -o thin,fssize -u m list
```

```
DEVICE    SIZE       PHYS_ALLOC FS_SIZE GROUP  TYPE     RECLAIM_CMD
emc0_428a 16384.00m 6335.00m   610.00m  mydg  thinrclm  WRITE_SAME
emc0_428b 16384.00m 3200.00m    22.00m  mydg  thinrclm  WRITE_SAME
emc0_4287 16384.00m 6233.00m   617.00m  mydg  thinrclm  WRITE_SAME
emc0_4288 16384.00m 1584.00m  1417.00m  mydg  thinrclm  WRITE_SAME
emc0_4289 16384.00m 2844.00m  1187.00m  mydg  thinrclm  WRITE_SAME
xiv0_030f 16384.00m 2839.00m  1223.00m  xivdg thinrclm  WRITE_SAME
xiv0_0307 16384.00m  666.00m   146.00m  xivdg thinrclm  WRITE_SAME
xiv0_0308 16384.00m  667.00m   147.00m  xivdg thinrclm  WRITE_SAME
xiv0_0309 16384.00m    3.00m   -        -     thinrclm  WRITE_SAME
xiv0_0310 16384.00m   30.00m   -        -     thinrclm  WRITE_SAME
```

特定の LUN またはエンクロージャのファイルシステムの使用状況を表示するには、次の形式のコマンドを使用します。

```
$ vxdisk -o thin,fssize list [-u unit] disk|enclosure
```

次に例を示します。

```
$ vxdisk -o thin,fssize list emc0
```

```
DEVICE    SIZE(MB) PHYS_ALLOC(MB) FS_SIZE(MB) GROUP TYPE     RECLAIM_CMD
emc0_428a 16384    6335           610          mydg  thinrclm WRITE_SAME
emc0_428b 16384    6335           624          mydg  thinrclm WRITE_SAME
emc0_4287 16384    6335           617          mydg  thinrclm WRITE_SAME
emc0_4288 16384    1584           617          mydg  thinrclm WRITE_SAME
emc0_4289 16384    2844           1187         mydg  thinrclm WRITE_SAME
```

ファイルシステムの領域の再生

表 32-1に、シン再生を管理する fsadm コマンドのオプションを示します。

表 32-1 シン再生を管理する fsadm のオプション

オプション	説明
-o aggressive | -A	シンストレージの積極的な再生を開始します。積極的な再生は SSD デバイスではサポートされません。
-o analyse|analyze	分析再生オプションを開始します。
-o auto	自動再生オプションを開始します。
-o ssd	TRIM 対応の下位 SSD デバイスで TRIM コマンドを開始します。
-o thin	Thin Reclaim 対応の下位デバイスでシン再生を開始します。
-P	マルチスレッドの Thin Storage Reclamation を実行します。デフォルトでは、fsadm コマンドは単一スレッドの Thin Storage Reclamation を実行します。マルチスレッドの Thin Storage Reclamation を使用するには、アレイで複数の再生操作の同時実行がサポートされている必要があります。
-R	VxFS ファイルシステムのシンストレージ LUN に対して、空きストレージの再生を実行します。

fsadm_vxfs(1M)のマニュアルページを参照してください。

積極的な領域再生を実行するには

1 VxFS ファイルシステムがマウントされていることを確認します。

mount(1M) のマニュアルページを参照してください。

VxFS ファイルシステムをマウントする必要がある場合は、mount_vxfs(1M) のマニュアルページを参照してください。

2 /mnt1 にマウントされた VxFS ファイルシステム上のシンストレージ LUN に対し、空きストレージの積極的な再生を実行します。

```
# /opt/VRTS/bin/fsadm -R -o aggressive /mnt1
```

領域再生を実行するには

1 VxFS ファイルシステムがマウントされていることを確認します。

`mount`(1M) のマニュアルページを参照してください。

VxFS ファイルシステムをマウントする必要がある場合は、`mount_vxfs`(1M) のマニュアルページを参照してください。

2 `/mnt1` にマウントされた VxFS ファイルシステムの領域再生を実行します。

```
# /opt/VRTS/bin/fsadm -R /mnt1
```

ディスク、ディスクグループ、エンクロージャの領域の再生

1 つ以上のディスク、ディスクグループ、またはエンクロージャ上でオンラインシン再生をトリガするときは、`vxdisk reclaim` コマンドを使います。デフォルトでは、`vxdisk reclaim` コマンドによって、VxVM ボリュームに VxFS ファイルシステムが「マウントされている」ディスクでシン再生が実行されます。VxFS ファイルシステムがマウントされていないディスクはスキップされます。シン再生は RAID 5 ボリュームまたはインスタントスナップショットではサポートされていません。

Storage Foundation Cluster File System High Availability は、`/etc/vx/log/reclaim_log` ファイルに再生利用イベントの統計のログを記録します。

p.792 の「再生ログファイルについて」を参照してください。

デフォルトでは、次のコマンドは指定されたディスクがサポート対象のソリッドステートデバイス(SSD)である場合も TRIM 再生を実行します。

ディスクの領域の再生

◆ 再生をトリガするには、次のコマンドを実行します。

```
# vxdisk reclaim [disk...]
```

たとえば、hitachi_usp0_065a と hitachi_usp0_065b という 2 つの LUN の再生をトリガする場合は、次を実行します。

```
# vxdisk reclaim hitachi_usp0_065a hitachi_usp0_065b
```

前の例では、hitachi_usp0_065a に VxFS ファイルシステムがマウントされた VxVM ボリューム vol1 が含まれていることを想定しています。VxFS ファイルシステムがマウントされていない場合、hitachi_usp0_065a の再生がスキップされます。hitachi_usp0_065b をスキャンし、未使用の領域があれば再生します。

ディスクでの積極的な領域再生の実行

- 再生をトリガするには、次のコマンドを実行します。

  ```
  # vxdisk -o full reclaim [disk...]
  ```

  たとえば、hitachi_usp0_065a という LUN の再生をトリガする場合は、次を実行します。

  ```
  # vxdisk -o full reclaim hitachi_usp0_065a
  ```

  前の例では、hitachi_usp0_065a に VxFS ファイルシステムがマウントされた VxVM ボリューム vol1 が含まれていることを想定しています。-o full オプションを指定したこのコマンドで vol1 外の未使用領域に対して hitachi_usp0_065a をスキャンし、未使用の領域があれば再生します。たとえば、サブディスク間に領域があれば、それが再生されます。

SSD のディスク上の領域の再生

- TRIM 操作をトリガするには、次のコマンドを実行します。

  ```
  # vxdisk [-o ssd] reclaim [disk...]
  ```

  たとえば、fiodrive0_0 と fiodrive0_1 の TRIM 操作をトリガする場合、次の手順を実行します。

  ```
  # vxdisk reclaim fiodrive0_0 fiodrive0_1
  ```

ディスクグループの領域の再生

- 再生をトリガするには、次のコマンドを実行します。

  ```
  # vxdisk [-o ssd | -o thin] reclaim diskgroup
  ```

  たとえば、ディスクグループ oradg の再生をトリガするには、次を実行します。

  ```
  # vxdisk reclaim oradg
  ```

  ディスクグループに SSD とシン再生 LUN の両方が含まれる場合、-o ssd オプションを使って TRIM 操作のみを実行できます。シン再生のみを実行するには、-o thin オプションを使います。

エンクロージャの領域の再生

- 再生をトリガするには、次のコマンドを実行します。

  ```
  # vxdisk reclaim enclosure
  ```

  たとえば、enclosure=EMC_CLARiiON0 の再生をトリガするには、次を実行します。

  ```
  # vxdisk reclaim EMC_CLARiiON0
  ```

次のコマンドを使用して、特定のデバイスの TRIM 機能またはシン再生をオフにできます。

```
# vxdisk set reclaim=off disk
```

vxdisk(1M)マニュアルページを参照してください。

再生ログファイルについて

Storage Foundation Cluster File System High Availability は、/etc/vx/log/reclaim_log ファイルでの再生イベントの統計情報をロギングします。表 32-2 では、再生ログファイルのフィールドを説明しています。

VVR(Veritas Volume Replicator)では、再生のロギングはローカルノードの場合にのみ発生します。

表 32-2　再生ログファイルフィールド

LOG フィールド	説明
START_TIME	再生タスクの開始時間。
DURATION	再生タスクの完了にかかる時間。
DISKGROUP	サブディスクに関連付けられるディスクグループ名。TYPE=GAP では、ディスクグループ値は NULL 値である場合があります。
VOLUME	サブディスクに関連付けられるボリューム。ボリュームがサブディスクに関連付けられない場合、値は NULL です。
DISK	サブディスクに関連付けられるディスク。
SUBDISK	再生操作が実行されるサブディスクの名前。
OFFSET	サブディスクの開始のオフセット。
LEN	サブディスクの合計長。
PA_BEFORE	再生タスクの前の物理的な割り当て。
PA_AFTER	再生タスクの後の物理的な割り当て。
TYPE	再生操作のタイプ。変数は次のいずれかになります。 ■ GAP: サブディスク間のギャップを再生します ■ SD: サブディスクを再生します ■ FULL: DG の存在しないディスクで完全な LUN を再生します ■ VXFS: マウントされた VxFS ファイルシステムを再生します。

LOG フィールド	説明
STATUS	再生操作が成功したかどうか。 エラーの場合は、STATUS にもエラーコードが表示されます。 ボリュームまたはプレックスなどのオブジェクトが削除されると、状態は「Pending」としてロギングされます。

vxtask コマンドを使ったシン再生の監視

シン再生は、ディスクのサイズと再生される領域のサイズによっては、長時間におよぶ可能性がある集中的な操作です。他の実行時間の長いタスクと同様に、vxtask コマンドを使用する操作を監視できます。

シン再生を監視するには

1 ディスク、ディスクグループ、エンクロージャのシン再生を、通常とおり起動します。

```
# vxdisk reclaim diskgroup| disk| enclosure
```

次に例を示します。

```
# vxdisk reclaim dg100
```

2 再生の状態を監視するには、別のセッションで次のコマンドを実行します。

```
# vxtask monitor
```

```
TASKID PTID TYPE/STATE    PCT   PROGRESS
1258   -    RECLAIM/R 17.28% 65792/33447328/5834752 RECLAIM vol4 dg100
1259   -    RECLAIM/R 25.98% 0/20971520/5447680     RECLAIM vol2 dg100
1263   -    RECLAIM/R 25.21% 0/20971520/5287936     RECLAIM vol3 dg100
1258   -    RECLAIM/R 25.49% 0/20971520/3248128     RECLAIM vol4 dg100
1258   -    RECLAIM/R 27.51% 0/20971520/3252224     RECLAIM vol4 dg100
1263   -    RECLAIM/R 25.23% 0/20971520/5292032     RECLAIM vol3 dg100
1259   -    RECLAIM/R 26.00% 0/20971520/5451776     RECLAIM vol2 dg100
```

3 複数のタスクがある場合、タスクを表示するには次のコマンドを使用できます。

```
# vxtask list

TASKID PTID TYPE/STATE    PCT   PROGRESS
1258   -    RECLAIM/R 17.28% 65792/33447328/5834752 RECLAIM vol4 dg100
1259   -    RECLAIM/R 25.98% 0/20971520/5447680     RECLAIM vol2 dg100
1263   -    RECLAIM/R 25.21% 0/20971520/5287936     RECLAIM vol3 dg100
```

4 タスクを監視するには、前の出力のタスク ID を使用します。

```
# vxtask monitor 1258

TASKID  PTID TYPE/STATE    PCT   PROGRESS
1258     -   RECLAIM/R 17.28% 65792/33447328/5834752 RECLAIM vol4 dg100
1258     -   RECLAIM/R 32.99% 65792/33447328/11077632 RECLAIM vol4 dg100
1258     -   RECLAIM/R 45.55% 65792/33447328/15271936 RECLAIM vol4 dg100
1258     -   RECLAIM/R 50.00% 0/20971520/10485760     RECLAIM vol4 dg100
.
.
.
```

vxtask list コマンドを実行中、別のセッションで vxdisk reclaim コマンドが実行されます。

vxtask(1m) のマニュアルページを参照してください。

自動再生の設定

vxrelocd デーモンは、再生が必要なディスクを追跡します。デフォルトでは、vxrelocd デーモンは毎日 22:10 に実行され、削除してから 1 日経過したボリューム上のストレージが再生されます。

再生のスケジュールを制御するには、次のチューニングパラメータを使用します。

reclaim_on_delete_wait_period	VxVM がストレージ領域を再生したときにボリュームまたはプレックスが削除された後の日数を指定します。値は、-1 と 367 の間の整数です。 デフォルト値は 1 です。この場合、領域は翌日に再生されます。 値が -1 の場合、ストレージはすぐに再生されます。 値が 367 の場合、ストレージ領域は自動的に再生されません。ストレージ領域は、vxdisk reclaim コマンドを使って手動でのみ再生できます。
reclaim_on_delete_start_time	削除されたボリュームの再生を VxVM が開始した時刻。値は、24 時間形式で指定します。(hh:mm) デフォルトでは 22:10 に設定されます。

チューニングパラメータは vxdefault コマンドを使って変更できます。vxdefault(1M) マニュアルページを参照してください。

33

Veritas InfoScale 4 k セクタのデバイスサポートのソリューション

この章では以下の項目について説明しています。

4 K セクタサイズの技術について

従来、ハードディスクドライブ (HDD) とソリッドステートデバイス (SSD) などのストレージデバイスに格納されているデータは、セクタと呼ばれる小さな論理ブロックにフォーマットされてきました。長年にわたるストレージ密度の増加に対し、ストレージデバイスのセクタサイズは一貫して 512 バイトのままでした。ただし、このデバイスのセクタサイズはソリッドステートデバイス (SSD) には不十分です。

512 バイトから 4096 バイトまたは 4 K セクタに移行する利点

4 K セクタのディスクは、最先端の次世代型デバイスです。このディスクは、セクタのヘッダーとエラー訂正コードに割り当てられている領域が削減されているのでストレージの表面上の領域を最適に使用できます。小さいファイルに比べ、大きいファイルのほうが効率的であるとされています。

4 K セクタサイズの先進型デバイスは次の理由により、512 バイトセクタサイズを超える利点があると考えられます。

1. フォーマットの効率が向上する

2. 強力なエラー訂正を提供する

メリットを考慮して、Hitachi 社、NEC 社、Fujitsu 社など、多くのストレージデバイスメーカーは 4 K セクタのデバイスの出荷を開始しています。

ただし、現在のコンピューティングのさまざまな側面からまだ相変わらずセクタは 512 バイトが前提になっています。代替手段として、512 バイトセクタのエミュレーション方式を用いて 4 K セクタに移行します。512 バイトセクタのエミュレーション方式の欠点は、デバイスの効率が低下することです。

Veritas InfoScale は、Veritas Volume Manager と Veritas File System ストレージコンポーネントを使用してストレージ環境で 4 K セクタのデバイス (4 KB でフォーマット) をサポートするソリューションを提供します。以前は、512 バイトで 4 K のデバイスをフォーマットする必要がありました。現在、追加フォーマットを行わずに Veritas InfoScale で 4 K セクタのデバイスを直接使用できます。

サポート対象のオペレーティングシステム

Linux (RHEL、SLES、サポート対象の RHEL 互換配布) と Solaris 11 オペレーティングシステムでのみ、Veritas InfoScale 7.2 以降のバージョンで 4 K セクタのデバイスを使用できます。

p.797 の「Veritas InfoScale のサポート外の構成」を参照してください。

p.798 の「512 バイトセクタ サイズのデバイスから 4 K セクタサイズのデバイスへの VxFS ファイルシステムの移行」を参照してください。

Veritas InfoScale のサポート外の構成

このセクションでは、4 K セクタのデバイスでサポートされていないさまざまな Veritas InfoScale の機能を示します。

- ボリュームレイアウト: RAID 5 はサポートされません。その他のすべてのボリューム レイアウトはサポートされます。
- VxVM ディスクグループのサポート: クロスプラットフォームデータ共有 (CDS) ディスクグループ形式はサポートされます。512 バイトセクタディスクと 4 K セクタディスクを組み合わせた (混合した) ディスクグループはサポートされません。2 つの異なるディスクグループ (1 つは 4 K ディスク、もう 1 つは 512 バイトディスク) が共存できます
- VxVM SmartIO の設定サポート: アプリケーションボリュームをホストするディスクとキャッシュをホストするディスクのセクタサイズが異なる場合、そのアプリケーションボリュームではキャッシュは有効化されません。
- ストレージエリアネットワーク (SAN) のブート
- ルートディスクのカプセル化
- 別のセクタサイズディスクを含むディスクグループのスナップショット

- VVR (Veritas Volume Replicator) などのボリュームレベルのレプリケーション
- VxFS ファイルシステムのサポート: ファイルシステムの 4 KB 未満のブロックサイズと logiosize は、4 K セクタデバイスでサポートされていません

512 バイトセクタ サイズのデバイスから 4 K セクタサイズのデバイスへの VxFS ファイルシステムの移行

このセクションでは、512 バイトから 4 K セクタサイズのデバイスに VxFS ファイルシステムを移行する手順について説明します。

既存の 512 バイトセクタデバイスの VxFS ファイルシステムは、4 K セクタデバイスでサポートされていない 1 KB または 2 KB のファイルシステムのブロックサイズで作成されていることがあります。そのため、アレイレベルまたはボリュームレベルの移行や複製など、従来のストレージの移行ソリューションが正しく動作しない可能性があります。

Veritas InfoScale 7.2 以降のバージョンでは、標準ファイルコピー機構を使用して 512 バイトセクタサイズのデバイスから 4 K セクタサイズのデバイスに VxFS ファイルシステムを移行できます。

メモ: 標準ファイルコピー機構では、特定のファイルレベル属性と割り当てジオメトリが保持されないことがあります。

メモ: 512 バイトセクタサイズから 4 K セクタサイズへの VxFS ファイルシステムの移行は、Linux (RHEL、SLES、サポート対象の RHEL 互換配布) と Solaris 11 オペレーティングシステムでのみサポートされます。

---

512 バイトセクタサイズのデバイスから 4 K セクタサイズのデバイスに VxFS ファイルシステムを移行するには、次の操作をします。

1 512 バイトと 4 K の VxFS ファイルシステムをマウントする

```
# mount -t vxfs /dev/vx/dsk/diskgroup/volume_512B /mnt1

# mount -t vxfs /dev/vx/dsk/diskgroup/volume_4K /mnt2
```

2 /mnt1 のすべてのファイルを手動で /mnt2 にコピーする

```
# cp -r /mnt1 /mnt2
```

3 512 バイトと 4 K の両方の VxFS ファイルシステムをマウント解除する

```
# umount  /mnt1

# umount  /mnt2
```

p.796 の「4 K セクタサイズの技術について」を参照してください。

p.797 の「Veritas InfoScale のサポート外の構成」を参照してください。

9

ストレージ利用率の最大化

34

SmartTier によるストレージの階層化について

この章では以下の項目について説明しています。

SmartTier について

SmartTier は、データ使用上の必要条件に基づいてデータストレージの一致処理を行います。データの一致処理が終了すると、データの使用上の必要条件と、ストレージ管理者またはデータベース管理者（DBA）により定義された他の必要条件に応じてこのデータが再配置されます。

時間の経過とともに保持されるデータが増えるにつれ、一部のデータはしだいに使用頻度が低くなっていきます。使用頻度が低くても、データを保持するには多くのディスク領域が必要です。SmartTier を使うと、データベース管理者が、使用頻度の低いデータをより低速で安価なディスクに移動してデータを管理できるようになります。また、頻繁に利用されるデータをより高速のディスクに保持してすばやく検索できるようにすることも可能です。

ストレージの階層化は、複数のタイプのデータをさまざまなストレージに割り当てることで、パフォーマンスを高速化し、経費を削減する方法です。SmartTier では、ストレージクラスを使って、特定の層を構成するディスクを指定します。ストレージクラスを定義するには、一般的に 2 つの方法があります。

- 処理効率またはストレージが要求されるクラス: 最も使用頻度が高いクラスは、高速で高価なディスクです。定期的には必要とされなくなったデータは、低速で安価なディスクで構成される別のクラスに移動できます。

- 耐障害性を備えたクラス: 各クラスは、非ミラーボリューム、ミラーボリューム、n-way 型ミラーボリュームで構成されます。

  たとえば、通常、データベースはデータ、インデックス、ログで構成されます。データは重要であるため、3 方向のミラーで設定できます。インデックスは重要ですが再作成が可能なため、2 方向のミラーで設定できます。REDO ログおよびアーカイブログを毎日採取する必要はありませんが、これらはデータベースリカバリに必須です。ミラー化することもお勧めします。

SmartTier は、作成したルールに従って異なるストレージ階層からファイルストレージ領域を割り当てることができる VxFS の機能です。SmartTier では、現在より柔軟な方法で階層化されたストレージの管理を行うことができます。ストレージの静的な階層化では、ストレージクラスへのアプリケーションファイルの割り当てを手動で 1 回のみ行いますが、これは長期的に見ると柔軟性に欠けます。通常、階層型ストレージ管理のソリューションでは、アプリケーションのアクセス要求に対する処理を実行する前に、ファイルをファイルシステムの名前空間に戻す必要があり、その結果として遅延とランタイムのオーバーヘッドが発生します。一方、SmartTier では、以下のことが可能です。

- 時間とともにファイルの価値が変わると、最適なストレージ階層にファイルを動的に移動することで、ストレージ資産を最適化します。
- ユーザーまたはアプリケーションのファイルアクセス方法を変更せずに、ストレージ階層間のデータの移動を自動化します。
- 管理者が設定したポリシーに基づいてデータを自動的に移行します。これにより、階層化されたストレージに対する操作要件や、データの移動に関連するダウンタイムが不要になります。

メモ: SmartTier は、以前に Dynamic Storage Tiering（DST）と呼ばれていた機能を拡張し、名前を変更したものです。

SmartTier のポリシーでは、初期のファイルの場所と、既存ファイルが再配置される環境を制御します。これらのポリシーをファイルに適用すると、ファイルシステムのボリュームセットの特定のサブセット（配置クラスと呼ばれる）上でそのファイルを作成し、拡張できます。ファイルは、指定した名前、タイミング、アクセス率、ストレージ容量に関連した条件に一致すると、他の配置クラス内のボリュームに再配置されます。

事前設定ポリシーに加え、必要に応じて、SmartTier で高速または低速のストレージにファイルを手動で移動できます。また、有効なポリシーの一覧表示、ファイル状態の表示、ボリュームの使用状況の表示、ファイルの統計情報の表示などを行うレポートを実行できます。

SmartTier では、Storage Foundation Cluster File System High Availability に搭載される 2 つの主要なテクノロジを利用しています。1 つは MVS ファイルシステムです。もう 1 つはファイルシステムで管理されるストレージ内でのポリシーに基づくファイルの自動配置です。MVS ファイルシステムは、2 つ以上の仮想ストレージボリュームを占有するので、単一のファイルシステムを異機種混在の可能性もある複数の物理ストレージデバイス

にまたがるようにすることが可能になります。たとえば、1 つ目のボリュームが EMC Symmetrix DMX のスピンドルに存在し、2 つ目のボリュームが EMC CLARiiON のスピンドルに存在するということも可能です。単一の名前空間を提供することで、ユーザーとアプリケーションは複数のボリュームを透過的に認識します。この MVS ファイルシステムでは各ボリュームが識別されるので、個々のファイルが格納されている場所を制御することが可能です。MVS ファイルシステムをポリシーに基づくファイルの自動配置と組み合わせることで、アプリケーションにもユーザーにもダウンタイムの影響を与えずにデータを自動的に移動する理想的なストレージ階層化機能を実現できます。

データベース環境では、アクセス期間のルールを一部のファイルに適用できます。ただし、たとえばデータファイルによっては、アクセスされるたびに更新されて、アクセス期間のルールを使用できない場合もあります。SmartTier には、ファイル全体だけでなくファイルの一部をセカンダリ階層に再配置する機構も用意されています。

SmartTier を使うには、次の機能でストレージを管理している必要があります。

- VxFS MVS ファイルシステム
- VxVM ボリュームセット
- ボリュームタグ
- ファイルレベルの SmartTier 管理
- サブファイルレベルの SmartTier 管理

VxFS MVS ファイルシステムについて

MVS ファイルシステムは、2 つ以上の仮想ボリュームを占有するファイルシステムです。ボリュームの集合はボリュームセットと呼ばれ、1 つの Veritas Volume Manager（VxVM）ディスクグループに属するディスクまたはディスクアレイ LUN で構成されます。MVS ファイルシステムは、単一の名前空間を示し、これによって複数のボリュームの存在がユーザーとアプリケーションに透過的になります。各ボリュームは管理者用に個別の ID 情報を維持し、これによって各ファイルが配置される場所を制御できます。

p.815 の「MVS ファイルシステムについて」を参照してください。

この機能は、次の必要条件を満たすファイルシステムでのみ利用できます。

- ディスクグループのバージョンが 140 以上。
  p.976 の「ディスクグループバージョン」を参照してください。
- ファイルレベルの SmartTier に対するファイルシステムレイアウトのバージョンが 7 以上。
- サブファイルレベルの SmartTier に対するファイルシステムレイアウトのバージョンが 8 以上。

既存の VxFS システムを VxFS MVS ファイルシステムに変換するには、単一ボリュームをボリュームセットに変換する必要があります。

p.820 の「MVS ファイルシステムへの単一ボリュームファイルシステムの変換」を参照してください。

VxFS ボリュームを管理するには、VxFS ボリューム管理ユーティリティ(fsvoladm ユーティリティ)が使えます。fsvoladm ユーティリティは、指定の Veritas File System へのボリュームの追加、削除、サイズ変更、カプセル化、フラグ設定、フラグ消去、フラグ問い合わせといったさまざまな管理作業を行います。

このユーティティについて詳しくは、`fsvoladm` (1M) のマニュアルページを参照してください。

VxVM ボリュームセットについて

ボリュームセットでは、複数のボリュームを 1 つの論理オブジェクトで表すことができます。ボリュームセットを空にすることはできません。基盤となるボリュームとの間のすべての I/O 処理は、ボリュームセットの I/O インターフェースを経由して実行されます。ボリュームセット機能は、Veritas File System(VxFS)のマルチボリュームの機能を強化します。この機能により、ファイルシステムで、基盤となるボリュームの処理効率や可用性などの様々な特性を最大限に活用できます。たとえば、ファイルシステムメタデータを冗長性の高いボリューム上に保存し、ユーザーデータを処理効率の高いボリュームに保存できます。

ボリュームタグについて

ボリュームタグをボリュームに関連付けることによって、配置クラスの VxVM ボリューム部分を作成します。ファイルを配置する目的で、VxFS は配置クラスのボリュームをすべて同等に扱い、ボリューム間に領域割り当てを分散します。ボリュームには、2 つ以上のタグを関連付けることができます。ボリュームに複数のタグを関連付けた場合、そのボリュームは複数の配置クラスに属し、いずれかの配置クラスに関連する割り当てポリシーと再配置ポリシーの影響を受けます。

警告: 複数のタグ付けは慎重に行ってください。

配置クラスは、MVS ファイルシステムのボリュームセットに存在する特定のボリュームの SmartTier 属性です。この属性は文字列で、ボリュームタグと呼ばれます。

SmartTier ファイルの管理

SmartTier では、VxFS MVS ファイルシステムの管理者は、初期ファイルの場所と既存ファイルが再配置される環境を制御する配置ポリシーを定義することによって、ボリュームセットの個々のボリューム上でファイルの配置を管理できます。これらの配置ポリシーをファイルに適用すると、ファイルシステムのボリュームセットの特定のサブセット(配置クラスと呼ばれる)上でそのファイルを作成し、拡張できます。ファイルは、指定した名前、タイ

ミング、アクセス率、ストレージ容量に関連した条件に一致すると、他の配置クラス内のボリュームに再配置されます。

ファイルベースの移動

- 管理者は、ファイルの新規作成前に、ファイル名拡張子に基づくファイル割り当てポリシーを作成し、データベース作成時には適切な階層の上にデータファイルを作成できます。
- また、管理者は、データベースファイルまたは他のタイプのファイルに対してファイル再配置ポリシーを作成し、ファイルが使われる頻度に基づいてファイルを再配置できます。

SmartTier のサブファイルオブジェクトの管理

SmartTier では、MVS VxFS ファイルシステムの管理者は、ファイルオブジェクトや個々のボリューム上のファイル全体の配置を管理することができます。

サブファイルベースの移動によって、次のことができます。

- 指定したマウントセットの指定したファイルセット内の一連の範囲を、要求に応じて目的の階層セットに移動します。
- 次の目的で、ファイルのセグメントを自動的に移動します。
  - 一連のファイルを監視して、I/O 統計情報を収集します。
  - 必要に応じてクラスタ全体で、統計情報を定期的に収集し、保持します。
  - 目的の階層セットに対する相対的なアクセス頻度に基づいて、登録したファイルセットへの範囲制限を定期的に実行します。
  - それらの範囲の移動履歴を追跡します。

SmartTier ポリシーと共有エクステントの連携方法

SmartTier の実施操作では、共有エクステントの移動は無視されます。たとえば、デバイス 1 に所属する共有およびプライベートエクステントを含むファイル A を想定します。ファイル A のすべてのエクステントをデバイス 2 に割り当てる必要があるポリシーを設定した場合、SmartTier の実施操作によってすべての非共有エクステントはデバイス 1 からデバイス 2 に移動されます。ただし、SmartTier の実施操作では、共有エクステントの移動は無視されます。その結果、ファイル A にはデバイス 1 に所属する共有エクステントはまだ含まれます。SmartTier の実施操作が成功した場合でも、この状態は発生します。

ただし、以降のファイル A に代わる新しい割り当ては、事前設定された SmartTier ポリシーに従います。コピーオンライト操作や共有解除操作には新しい割り当てが必要なため、SmartTier の実施操作は事前設定済みのポリシーに従います。ファイル A の書き込み操作によって共有エクステントへの書き込みが行われる場合、コピーオンライト操作

の一部である新しい割り当て操作は、デバイス 2 から行われます。この動作は、事前設定された SmartTier ポリシーに従います。

高可用性（HA）環境での SmartTier

Cluster Server には、ボリュームセット用の付属エージェントが用意されていません。ボリュームまたはボリュームセットに問題が起きた場合、その問題は DiskGroup リソースと Mount リソースレベルでのみ検出できます。

DiskGroup エージェントは、Veritas Volume Manager（VxVM）ディスクグループをオンライン状態またはオフライン状態にしたり、監視したりします。このエージェントでは VxVM コマンドを使用します。StartVolumes 属性と StopVolumes 属性の値が両方とも 1 の場合、DiskGroup エージェントは、ディスクグループのインポート操作とデポート操作中、ボリュームをオンライン状態にしたり、オフライン状態にしたりします。ボリュームセットを使っている場合、そのボリュームセットを含んでいる DiskGroup リソースの StartVolumes 属性と StopVolumes 属性の値を 1 に設定します。そのボリュームセットにファイルシステムを作成する場合、Mount リソースを使ってボリュームセットをマウントします。

Mount エージェントは、ファイルシステムまたは NFS クライアントのマウントポイントをオンライン状態またはオフライン状態にしたり、監視したりします。

高可用性（HA）環境で SmartTier for Oracle コマンドを使う場合は、クラスタ内の各システムの時刻を同期させる必要があります。同期しない場合、サービスグループのフェールオーバー後、スケジュール設定されたタスクが設定時刻に実行されないことがあります。

詳しくは『Cluster Server 付属エージェントリファレンスガイド』を参照してください。

35

ボリュームセットの作成と管理

この章では以下の項目について説明しています。

ボリュームセットについて

Veritas File System（VxFS）はボリュームセットを使い、マルチボリュームサポート（MVS: Multi Volume Support）機能と SmartTier 機能を実装しています。

p.801 の「SmartTier について」を参照してください。

Veritas Volume Manager（VxVM）には、ボリュームセットを作成および管理するための `vxvset` コマンドが用意されています。

vxvset（1M）マニュアルページを参照してください。

ボリュームセットには次の制限があります。

- ボリュームセットでは、最大 2048 のボリュームを設定できます。
- ボリュームセットでは、Veritas File System のみがサポートされます。

- ボリュームセットの最初のボリューム(インデックス 0)は、ボリュームサイズの 4000 分の 1、VxFS インテントログのサイズ、および 1MB の合計より大きいサイズにする必要があります。258 MB 以上のボリュームであれば十分なはずです。
- ボリュームセットに対する raw I/O はサポートされていません。
- ボリュームセットのコンポーネントボリュームに対する raw I/O は、一定の条件の下でサポートされています。
  p.811 の「コンポーネントボリュームでの RAW デバイスノードの管理」を参照してください。
- ボリュームセットをボリュームの代わりに使うには、インスタントスナップショットに対する vxsnap 操作(addmir、dis、make、prepare、reattach、refresh、restore、rmmir、split、syncpause、syncresume、syncstart、syncstop、syncwait、unprepare)を使います。ボリュームセット内の各ボリュームにプレックスが十分に存在している場合、ボリュームセットでは、フルサイズインスタントスナップショットのサードミラーブレークオフモデルがサポートされます。
  スナップショットについて詳しくは、『Veritas InfoScale ソリューションガイド』を参照してください。
- ボリュームセットのフルサイズスナップショットは、それ自体が、親と同じ数のボリュームおよび同じボリュームインデックス番号を持つボリュームセットである必要があります。親ボリュームセットおよびスナップショットボリュームセット内の対応するボリュームも、スタンドアロンボリュームとそのスナップショットとの間に適用される制限と同じ制限を受けます。

ボリュームセットの作成

Veritas File System(VxFS)で使うボリュームセットを作成するには、次のコマンドを使います。

```
# vxvset [-g diskgroup] -t vxfs make volset
    volume
```

ここで、*volset* はボリュームセットの名前であり、*volume* はボリュームセット内の 1 番目のボリューム名です。-t vxfs オプションは、VxFS で使うように設定されたボリュームセットを作成します。このコマンドを実行する前にボリュームを作成する必要があります。vxvset ではボリュームが自動的に作成されません。

たとえば、ボリューム vol1 を保持するボリュームセット myvset をディスクグループ mydg に作成するには、次のコマンドを使います。

```
# vxvset -g mydg -t vxfs make myvset vol1
```

ボリュームセットへのボリュームの追加

ボリュームを 1 つ保持するボリュームセットを作成したら、次のコマンドを使って他のボリュームをボリュームセットに追加できます。

```
# vxvset [-g diskgroup] [-f] addvol volset
    volume
```

たとえば、ボリューム vol2 をボリュームセット myvset に追加するには、次のコマンドを使います。

```
# vxvset -g mydg addvol myvset vol2
```

警告: 追加するボリュームまたは追加先のボリュームセット内のボリュームが、スナップショットまたはスナップショットの親である場合は、-f（強制）オプションを指定する必要があります。操作に関係するボリュームのいずれかがすでに特定のスナップショットチェーンに属している場合は、このオプションの使用によりスナップショット階層内に不整合が発生する可能性があります。

ボリュームセットからのボリュームの削除

ボリュームセットからコンポーネントボリュームを削除するには、次のコマンドを使います。

```
# vxvset [-g diskgroup] [-f] rmvol volset
    volume
```

たとえば、次のコマンドは、ボリュームセット myvset からボリューム vol1 と vol2 を削除します。

```
# vxvset -g mydg -f rmvol myvset vol1
# vxvset -g mydg -f rmvol myvset vol2
```

最後のボリュームを削除すると、ボリュームセットが削除されます。

警告: 削除しようとしているボリューム、またはボリュームセット内の何らかのボリュームがスナップショットまたはスナップショットの親の場合は、-f (force) オプションを指定する必要があります。このオプションを使うと、操作にかかわるボリュームがスナップショットチェーンにすでに含まれている場合は、スナップショット階層に不整合が生じる潜在的な可能性があります。

ボリュームセットの詳細の一覧表示

ボリュームセットのコンポーネントボリュームの詳細を一覧表示するには、次のコマンドを使います。

```
# vxvset [-g diskgroup] list [volset]
```

ボリュームセットの名前を指定しないでコマンドを実行した場合は、次の例のように、ディスクグループ内のすべてのボリュームの詳細が一覧表示されます。

```
# vxvset -g mydg list

NAME          GROUP       NVOLS       CONTEXT
set1          mydg            3       -
set2          mydg            2       -
```

ボリュームセット内の各ボリュームの詳細を一覧表示するには、コマンドの引数としてボリュームセットの名前を指定します。

```
# vxvset -g mydg list set1

VOLUME            INDEX           LENGTH         KSTATE         CONTEXT
vol1                  0         12582912         ENABLED          -
vol2                  1         12582912         ENABLED          -
vol3                  2         12582912         ENABLED          -
```

コンテキストのフィールドには、ボリュームまたはボリュームセットの各目的を示すタグとして、アプリケーションによって設定された文字列の詳細が表示されます。

ボリュームセットの停止と起動

ある状況のもとでは、ボリュームセットの停止と再起動が必要になることがあります。たとえば、次に示すように、セット内のボリュームが切断されている場合がこれに該当します。

```
# vxvset -g mydg list set1

VOLUME            INDEX           LENGTH         KSTATE         CONTEXT
vol1                  0         12582912       DETACHED         -
vol2                  1         12582912         ENABLED        -
vol3                  2         12582912         ENABLED        -
```

1 つ以上のボリュームセットを停止して再起動するには、次のコマンドを使います。

```
# vxvset [-g diskgroup] stop volset ...
# vxvset [-g diskgroup] start volset ...
```

前述の例のコンポーネントボリュームに対してこれらのコマンドを実行すると、結果は次のようになります。

```
# vxvset -g mydg stop set1

# vxvset -g mydg list set1

VOLUME          INDEX          LENGTH        KSTATE        CONTEXT
vol1                0        12582912      DISABLED        -
vol2                1        12582912      DISABLED        -
vol3                2        12582912      DISABLED        -

# vxvset -g mydg start set1

# vxvset -g mydg list set1

VOLUME          INDEX          LENGTH        KSTATE        CONTEXT
vol1                0        12582912       ENABLED        -
vol2                1        12582912       ENABLED        -
vol3                2        12582912       ENABLED        -
```

コンポーネントボリュームでの RAW デバイスノードの管理

ファイルシステムとデータを事故破損から守るために、デフォルトでは、コンポーネントボリュームのデバイスノードにある `/dev/vx/rdsk/diskgroup` ディレクトリと `/dev/vx/dsk/diskgroup` ディレクトリに RAW エントリとブロックエントリは設定されません。そのため、アプリケーションはボリュームセットのコンポーネントボリュームに直接読み書きできません。

Veritas NetBackup™ ソフトウェアの RAW ボリュームバックアップ、リストア機能など、一部のアプリケーションにおいて `/dev/vx/rdsk/diskgroup` ディレクトリの RAW デバイスノードにアクセスしてコンポーネントボリュームを読み書きすることが必要な場合、`vxvset` コマンドに追加のコマンドラインオプションを指定することでサポートされます。ボリュームセットのコンポーネントボリュームにあるブロックデバイスノードに対するアクセスはサポートされません。

警告: ボリュームセットのコンポーネントボリュームにある RAW デバイスノードに対して直接書き込みまたは読み取りを行うのは、アクセス中にボリュームのデータが他に変更されない場合のみにしてください。

ボリュームセットのコンポーネントボリュームでは、すべての RAW デバイスノードは単一の操作で作成または削除できます。ボリュームセットに追加されたボリュームの RAW デバイスノードは必要に応じて自動的に作成され、既存のデバイスノードのアクセスモードを継承します。

コンポーネントボリュームの RAW デバイスノードへのアクセスは、読み取り専用または読み書き両用に設定できます。このモードは、ボリュームセットのコンポーネントボリュームにあるすべての RAW デバイスノードによって共有されます。読み取り専用アクセスモードでは RAW デバイスに対するすべての書き込みは失敗しますが、`ioctl` インターフェースを使った書き込みまたは VxFS によるメタデータの更新は許可されます。読み書き両用アクセスモードでは RAW デバイスを介した直接書き込みが許可されます。ボリュームセットの RAW デバイスノードに対するアクセスモードは、必要に応じて変更できます。

RAW デバイスノードの存在とそのアクセスモードは、システムを再起動しても維持されます。

この機能には次の制限があることに注意してください。

- ディスクグループのバージョンは 140 以上である必要があります。
- ボリュームセットのコンポーネントボリュームにある RAW デバイスノードに対するアクセスは、専用ディスクグループのみでサポートされています。クラスタ内の共有ディスクグループではサポートされていません。

ボリュームセット作成時の RAW デバイスアクセスの有効化

ボリュームセットの作成時に RAW デバイスアクセスを有効にするには、`vxvset make` コマンドを次の形式で使います。

```
# vxvset [-g diskgroup] -o makedev=on ¥
[-o compvol_access={read-only|read-write}] ¥
  [-o index] [-c "ch_addopt"] make vset
    vol [index]
```

`-o makedev=on` オプションを使うと、ボリュームセットの作成と同時にコンポーネントボリュームの RAW デバイスノードを作成できます。デフォルト設定は `off` です。

`-o compvol_access=read-write` オプションを指定すると、各コンポーネントボリュームの RAW デバイスに対する直接書き込みが許可されます。値を `read-only` に設定すると、各コンポーネントボリュームの RAW デバイスからの読み取りのみが許可されます。

`-o makedev=on` オプションを指定しても、`-o compvol_access` を指定していない場合、デフォルトのアクセスモードは `read-only` になります。

ボリュームセットにボリュームを追加するために vxvset addvol コマンドを後で使った場合、makedev 属性の値を on に設定していると、新しい RAW デバイスノードが */dev/vx/rdsk/diskgroup* に作成されます。アクセスモードは、compvol_access 属性の現在の設定によって決まります。

次の例は、ボリューム myvol1 を含むボリュームセット myvset1 を、RAW デバイスに対するアクセスを読み書き両用モードで有効にして、ディスクグループ mydg 内に作成しています。

```
# vxvset -g mydg -o makedev=on -o compvol_access=read-write ¥
  make myvset1 myvol1
```

ボリュームセットの RAW デバイスアクセス設定の表示

ボリュームセットの現在の設定を表示するには vxprint -m コマンドを使います。makedev 属性を on に設定している場合、出力に次のいずれかの文字列が表示されます。

vset_devinfo=on:read-only	RAW デバイスノードは読み取り専用モードです。
vset_devinfo=on:read-write	RAW デバイスノードは読み書き両用モードです。

makedev を off に設定している場合、文字列は表示されません。

vxprint -m コマンドの出力を vxmake コマンドに渡してボリュームセットを再作成する場合、vset_devinfo 属性は off に設定する必要があります。必要なアクセスモードで RAW デバイスへのアクセスを再び有効にするには、vxvset set コマンドを使います。

p.813 の「既存のボリュームセットの RAW デバイスに対するアクセスの制御」を参照してください。

既存のボリュームセットの RAW デバイスに対するアクセスの制御

既存のボリュームセットの RAW デバイスノードに対するアクセスを有効または無効にするには、次のコマンドを使います。

```
# vxvset [-g diskgroup] [-f] set makedev={on|off} vset
```

vxvset set コマンドに makedev 属性を指定すると、ボリュームセットのコンポーネントボリュームに RAW デバイスノードを作成（makedev=on）または削除（makedev=off）できます。起動しているコンポーネントボリュームがある場合、-f（force）オプションを指定して属性を off に設定する必要があります。

makedev=off を指定すると、*/dev/vx/rdsk/diskgroup* ディレクトリから既存の RAW デバイスノードが削除されます。

makedev 属性を off に設定し、mknod コマンドを使って RAW デバイスノードを作成した場合、makedev の値を on に設定しない限り、これらのノードからの読み取りやこれらのノードへの書き込みはできません。

ボリュームセットに compvol_access 属性を設定する構文は次のとおりです。

```
# vxvset [-g diskgroup] [-f] set ¥
  compvol_access={read-only|read-write} vset
```

vxvset set コマンドに compvol_access 属性を指定すると、ボリュームセットのコンポーネントボリュームに対するアクセスモードを変更できます。起動しているコンポーネントボリュームがある場合、-f（force）オプションを指定して属性を read-only に設定する必要があります。

次の例は、ディスクグループ mydg のボリュームセット myvset2 に、makedev=on 属性と compvol_access=read-only 属性を設定しています。

```
# vxvset -g mydg set makedev=on myvset2
```

次の例は、ボリュームセット myvset2 に、compvol_access=read-write 属性を設定しています。

```
# vxvset -g mydg set compvol_access=read-write myvset2
```

最後の例は、ボリュームセット myvset2 の RAW デバイスノードに対するアクセスを削除しています。

```
# vxvset -g mydg set makedev=off myvset2
```

36

MVS ファイルシステム

この章では以下の項目について説明しています。

MVS ファイルシステムについて

Veritas File System（VxFS）は、Veritas Volume Manager と組み合わせて使われた場合に、MVS ファイルシステム（MVFS）をサポートします。MVFS を使うと、1 つのファイルシステムを複数のボリュームにまたがって作成した上、各ボリュームに独自のプロパティを設定できます。たとえば、メタデータはミラー化されたストレージに配置する一方で、ファイルデータは RAID 1+0（ストライプ化ミラー）などのパフォーマンスの高いボリュームに配置できます。このボリュームは、ボリュームセットと同じディスクグループ内にある必要があり、別のボリュームセットのメンバーであってはなりません。

また、MVFS 機能を使うと、異なるクラスのデバイスにファイルシステムを配置できるため、安価なディスクと高価なアレイの両方を使って 1 つのファイルシステムをサポートできま

す。どのデータをどのボリュームタイプに入れるかは、MVFS 管理インターフェースを使って制御できます。

メモ: マルチボリュームファイルシステムのサポートはバージョン 7 以降のディスクレイアウトを使用するファイルシステムでのみ利用できます。ファイルシステムのディスクレイアウトバージョンを判断するには vxupgrade(1M)マニュアルのページを参照してください。

ボリュームの種類について

Veritas File System(VxFS)では 2 種類のボリュームが使われます。一方はデータのみを含むデータ専用ボリューム(dataonly)で、もう一方はメタデータまたはデータを含むことのできるメタデータ対応ボリューム(metadataok)です。

データは、ファイルシステムの正規のファイルや名前付きデータストリームのダイレクトエクステント(ユーザーデータが含まれる)を指します。

メタデータは、正規のファイルまたは名前付きデータストリームのエクステントでないすべてのエクステントを指します。これには、正規のファイルのように見えるがそうではないファイル(FCL ファイルなど)が含まれます。

データ専用ボリューム(dataonly)またはメタデータ対応ボリューム(metadataok)を指定するには、ボリュームの可用性フラグを設定します。ボリュームの可用性フラグは、fsvoladm コマンドを使って、設定、消去、一覧表示ができます。

fsvoladm(1M)のマニュアルページを参照してください。

MVFS(Multi Volume File System)を使って実装されている機能

次の機能は MVFS を使って実装することができます:

- ファイルを格納する場所を複数のレベルから選択します。これにより、特定のファイルやファイル階層を異なるボリュームに保存することができます。この機能は、Veritas File System の SmartTier 機能で使えます。p.831 の「SmartTier について」を参照してください。
- VxFS インテントログを単一ボリューム上に配置します。これにより、ディスクヘッドの移動が最小限になり、パフォーマンスが向上します。インテントログを個別のボリュームに配置してディスクヘッドの移動を最小限にし、パフォーマンスを向上します。
- Storage Checkpoints に割り当てられたデータがファイルシステムのほかのデータと区別されるように、Storage Checkpoints を分離します。
- Storage Checkpoints を分離します。

- ボリュームがファイルシステム上にファイルとして表示されるように、ボリュームをカプセル化します。これは、RAW ボリュームで実行されているデータベースでは特に便利です。
- データ専用ボリュームが使用不可能になったときに、メタデータ対応ボリュームが使用不可能にならないようになっています。
p.817 の「ボリュームの可用性」を参照してください。

MVS ファイルシステムの機能を使うには、Veritas Volume Manager がインストールされており、ボリュームセット機能にアクセスできる必要があります。ボリュームセット機能は別ライセンスです。

ボリュームの可用性

MVFS では、データ専用ボリュームが使用不可能になったときに、メタデータ対応ボリュームが使用不可能にならないようになっています。このため、1 つ以上のコンポーネントを含むデータ専用ボリュームが見つからない場合であっても MVS ファイルシステムをマウントできます。

ボリュームは、そのボリューム上でメタデータが許可されているかどうかにより分類されています。データ専用ボリュームで発生する I/O エラーは、他のいかなるボリュームへのアクセスにも影響を与えません。見つからないデータ専用ボリュームにアクセスしない VxFS 操作はすべて正常に動作します。

見つからないデータ専用ボリュームにアクセスせず、正常に動作する VxFS 操作には次のようなものがあります。

- ファイルシステムが読み取り専用または読み書きのいずれのモードであっても MVS ファイルシステムをマウントする場合
- カーネル操作
- `fsck` の再生を実行する場合。ログに記録された書き込みは、対応するボリュームがデータ専用であると通常の書き込みに変換されます。
- `fsck` の完全実行
- 見つからないボリュームのデータにアクセスすることのない、その他のすべてのコマンドの使用

データ専用ボリュームが見つからない場合に処理が失敗することのある操作は、次のとおりです。

- ファイルのデータエクステントが、見つからないデータ専用ボリュームから割り当てられている場合のファイルデータの読み取りまたは書き込み
- `vxdump` コマンドの使用

ボリュームの可用性がサポートされるのは、ファイルシステムのディスクレイアウトバージョン 7 以上になります。

メモ: ボリュームの可用性が異なる場合は、マウントオプション ioerror=disable または ioerror=wdisable で MVS システムをマウントしないでください。Veritas では、クラスタのマウントとローカルのマウントの両方に ioerror=mdisable のマウントオプションをそれぞれ使うことをお勧めします。

MVS ファイルシステムの作成

MVS ファイルシステムを作成すると、ボリューム 0 以外のすべてのボリュームが dataonly になり、ファイルシステムのメタデータを保存するために使われます。ボリューム 0 のボリュームの可用性フラグは、データ専用 (dataonly) に設定できません。

メタデータは、データ専用ボリューム (metadataok) から割り当てることができません。メタデータ対応ボリューム (dataonly) を使ってメタデータに十分な領域を割り当てる必要があります。df コマンドでファイルシステムに空き領域が存在することが示されても、メタデータで利用可能な領域が不足していると、「ファイルシステムの領域不足」エラーが発生します。fsvoladm コマンドを使って、各ボリュームの空き領域を確認してボリュームの可用性フラグを設定できます。

特に明記されていない場合、VxFS コマンドは、MVS ファイルシステム上で、単一ボリュームファイルシステム上の場合と同じように機能します。

次の手順では、MVS ファイルシステムの作成例を示します。

MVS ファイルシステムの作成例

1 ボリュームセットを作成したら、引数としてボリュームセット名を mkfs に指定して VxFS ファイルシステムを作成します。

p.808 の「ボリュームセットの作成」を参照してください。

```
# mkfs -t vxfs /dev/vx/rdsk/dg1/myvset
  version 16 layout
  12288 sectors, 6144 blocks of size 1024, log size 256 blocks

  rcq size 1024 blocks
  largefiles supported
  maxlink supported
  WORM not supported
```

ファイルシステムを作成すると、VxFS は、ボリュームセット内の複数のボリュームから領域を割り当てます。

2 fsvoladm コマンドを使って、ボリュームセットのコンポーネントボリュームを一覧で表示します。

```
# mount -t vxfs /dev/vx/dsk/dg1/myvset /mnt1
# fsvoladm -H list /mnt1
devid    size     used    avail    name
0        20 GB    10 GB   10 GB    vol1
1        30 TB    10 TB   20 TB    vol2
```

3 新しいボリュームをボリュームセット、ファイルシステムの順に追加します。

```
# vxassist -g dg1 make vol5 50m
# vxvset -g dg1 addvol myvset vol5
# fsvoladm add /mnt1 vol5 50m
# fsvoladm -H list /mnt1
ddevid    size         used       avail     name
 0       10 GB       74.6 MB    9.93 GB    vol1
 1       20 GB        16 KB     20.0 GB    vol2
 2       50 MB        16 KB     50.0 MB    vol5
```

4 fsvoladm コマンドを使って、ボリュームの可用性フラグを一覧で表示します。

```
  # fsvoladm queryflags /mnt1
volname       flags
vol1          metadataok
vol2          dataonly
vol5          dataonly
```

5 fsvoladmコマンドを使って、ファイルシステムにおけるメタデータの領域を増やします。

```
  # fsvoladm clearflags dataonly /mnt1 vol2
  # fsvoladm queryflags /mnt1
volname       flags
vol1          metadataok
vol2          metadataok
vol5          dataonly
```

MVS ファイルシステムへの単一ボリュームファイルシステムの変換

次は、ディスクグループ dg1 にある単一ボリューム vol1 上の従来の単一ボリュームファイルシステム /mnt1 を MVS ファイルシステムに変換する手順を示しています。

MVS ファイルシステムへの単一ボリュームファイルシステムの変換

1 ボリュームのディスクグループのバージョンを確認します。

```
# vxdg list dg1 | grep version: | awk '{ print $2 }'
90
```

2 バージョンが 110 よりも小さい場合は、ディスクグループをアップグレードします。

```
# vxdg upgrade dg1
```

3 ファイルシステムのディスクレイアウトバージョンを確認します。

```
# vxupgrade /mnt1
/mnt1: vxfs file system version 16 layout
```

4 ディスクレイアウトバージョンが 12 より低い場合は、必要条件に従って、バージョン 12 以上のディスクレイアウトにアップグレードしてください。

現在、最大でバージョン 16 のディスクレイアウトにアップグレードできます。

```
# vxupgrade -n 10 /mnt1
# vxupgrade -n 11 /mnt1
# vxupgrade -n 12 /mnt1
# vxupgrade -n 13 /mnt1
# vxupgrade -n 14 /mnt1
# vxupgrade -n 15 /mnt1
# vxupgrade -n 16 /mnt1
```

5 次のように入力して、ファイルシステムをマウント解除します。

```
# umount /mnt1
```

6 ボリュームをボリュームセットに変換します。

```
# vxvset -g dg1 make vset1 vol1
```

7 /etc/fstab ファイルを編集して、ボリュームセット名 vset1 をボリュームデバイス名 vol1 に変更します。

8 次のコマンドを実行して、ファイルシステムをマウントします。

```
# mount -t vxfs /dev/vx/dsk/dg1/vset1 /mnt1
```

9 必要に応じて、ボリュームセットに対してボリュームを作成したり、追加したりする場合は、次を実行します。

```
# vxassist -g dg1 make vol2 256M
# vxvset -g dg1 addvol vset1 vol2
```

10 タグのないすべてのボリュームに対して配置クラスタグを設定する場合は、次を実行します。

```
# vxassist -g dg1 settag vol1 vxfs.placement_class.tier1
# vxassist -g dg1 settag vol2 vxfs.placement_class.tier2
```

MVS ファイルシステムのボリュームの追加と削除

fsvoladm コマンドを使って次の関数を実行します。

- 「MVS ファイルシステムへのボリュームの追加」
- 「MVS ファイルシステムからのボリューム削除」

fsck コマンドを使って次の関数を実行します。

- 「MVS ファイルシステムのボリュームの強制削除」

vxassist コマンドを使って次の関数を実行します。

- 「MVS ファイルシステムのボリューム 0 の移動」

MVS ファイルシステムへのボリュームの追加

MVS ファイルシステムにボリュームを追加するには、fsvoladm add コマンドを使います。

MVS ファイルシステムにボリュームを追加するには

◆ MVS ファイルシステムに新しいボリュームを追加します。

```
# fsvoladm add /mnt1 vol2 256m
```

MVS ファイルシステムからのボリューム削除

fsvoladm remove コマンドを使って、MVS ファイルシステムからボリュームを削除します。削除しようとしているボリュームが、割り当てポリシーで設定されている最初のボリュームである場合、fsvoladm remove コマンドは失敗します。

MVS ファイルシステムからボリュームを削除する方法

◆ MVS ファイルシステムからボリュームを削除します。

```
# fsvoladm remove /mnt1 vol2
```

MVS ファイルシステムのボリュームの強制削除

ファイルシステムからボリュームを強制的に削除する必要がある場合、たとえば、ボリュームを永久的に破棄して、消滅したボリュームの配下にあるポインタを削除する場合は、fsck -o zapvol=volname コマンドを使います。zapvol オプションを指定すると、ファイルシステムを完全に検査して、指定したボリュームを参照する i ノードをすべて消去します。fsck コマンドを実行すると、このコマンドにより破棄されるすべてのファイルの i ノード番号が表示されますが、ファイル名自体は表示されません。zapvol オプションは、データ専用ボリュームで使われない限り通常のファイルには影響を及ぼしません。ただし、このオプションをメタデータ対応ボリューム (metadataok) で指定すると構造化ファイルが破損してしまう恐れがあります。このことは、ファイルシステムのリカバリ不能の原因ともな

り得ます。このことから、この zapvol オプションをメタデータ対応ボリューム(metadataok)で使う場合には注意してください。

MVS ファイルシステムのボリューム 0 の移動

MVS ファイルシステムのボリューム 0 をファイルシステムから削除することはできませんが、vxassist move コマンドを使ってボリューム 0 を別のストレージに移動することはできます。vxassist コマンドの実行により、必要とされる一時ミラーが作成され、処理の終了時にこのミラーが削除されます。

ボリューム 0 を移動するには

- ボリューム 0 を移動します。たとえば、ディスクグループ mydg のボリューム vol1 を disk1 から disk4 に移動するには次のようにします。

```
# vxassist -g mydg move vol1 ¥!disk1 disk4
```

ボリュームのカプセル化

メモ: Linux 7.3.1 以降では、ルートディスクのカプセル化 (RDE) はサポートされません。

MVS(Multi Volume Support)を使うと、既存の RAW ボリュームをカプセル化し、そのボリュームの内容をファイルシステムの 1 つのファイルとして表示できます。

ボリュームをカプセル化するには、次のことを実行します。

- 既存ボリュームセットに対するボリュームの追加
- fsvoladm によるファイルシステムへのボリュームの追加

ボリュームのカプセル化

メモ: ルートディスクのカプセル化 (RDE) は、7.3.1 以降から Linux でサポートされません。

次の例では、ボリュームのカプセル化について説明します。

ボリュームをカプセル化するには

1 ボリュームを一覧で表示します。

```
# vxvset -g dg1 list myvset

VOLUME    INDEX  LENGTH       KSTATE      CONTEXT
vol3       0     104857600    ENABLED       -
vol2       1     104857600    ENABLED       -
```

このボリュームセットには 2 つのボリュームがあります。

2 3 つ目のボリュームを作成し、このボリュームに passwd ファイルをコピーします。

```
# vxassist -g dg1 make dbvol 100m
# dd if=/etc/passwd of=/dev/vx/rdsk/dg1/dbvol count=1
1+0 records in
1+0 records out
```

この 3 つ目のボリュームは、ボリュームに 1 つのファイルとしてアクセスする方法を示すために使います (後述もあり)。

3 ボリュームセットでファイルシステムを作成します。

```
# /opt/VRTS/bin/mkfs /dev/vx/rdsk/dg1/myvset
  version 16 layout
  134217728 sectors, 67108864 blocks of size 1024, log size 65536
 blocks
  rcq size 4096 blocks
  largefiles supported
  maxlink  supported
```

4 ボリュームセットをマウントします。

```
# /opt/VRTS/bin/mount /dev/vx/dsk/dg1/myvset /mnt1
```

5 新しいボリュームをボリュームセットに追加します。

```
# vxvset -g dg1 addvol myvset dbvol
```

6 dbvol をカプセル化します。

```
# fsvoladm encapsulate /mnt1/dbfile dbvol 100m
# ls -l /mnt1/dbfile
-rw------- 1 root other 104857600 May 22 11:30 /mnt1/dbfile
```

7 dbfile の内容を調べて 1 つのファイルとしてアクセス可能であることを確認します。

```
# head -2 /mnt1/dbfile
root:x:0:1:Super-User:/:/sbin/sh
daemon:x:1:1::/:
```

RAW ボリュームに書き込まれた passwd ファイルの内容が、新しいファイルに表示されます。

メモ: カプセル化されたファイルが、割り当てポリシー、サイズ変更ボリュームを使って拡張、切り捨てまたは移動など、いずれかの方法で変更された場合、またはボリュームがバイアスを使ってカプセル化されている場合、ファイルのカプセル化は解除できません。

ボリュームのカプセル化解除

メモ: Linux 7.3.1 以降では、ルートディスクのカプセル化 (RDE) はサポートされません。

次の例では、ボリュームのカプセル化解除について説明します。

ボリュームのカプセル化を解除するには

1 ボリュームを一覧で表示します。

```
# vxvset -g dg1 list myvset
VOLUME    INDEX    LENGTH    KSTATE    CONTEXT
vol1      0        102400    ACTIVE    -
vol2      1        102400    ACTIVE    -
dbvol     2        102400    ACTIVE    -
```

このボリュームセットには 3 つのボリュームがあります。

2 dbvol のカプセル化を解除します。

```
# fsvoladm deencapsulate /mnt1/dbfile
```

ファイルエクステントの出力

MVS 機能では、fsmap コマンドと fsvmap コマンドを使ってファイルとボリューム間の相互マッピングを実現できます。fsmap コマンドを使って、ボリューム名、論理オフセット、データエクステントのサイズを出力するか、MVS ファイルシステムにおける 1 つのファイルに関連付けられた間接エクステントのボリューム名やサイズを出力することができます。fsvmap コマンドは、ボリューム上にエクステントが確保されたファイルにそのボリュームをマッピングします。

fsmap（1M）と fsvmap（1M）のマニュアルページを参照してください。

fsmap コマンドを実行するには、指定するファイルやディレクトリに open() 権限が必要です。特定のボリューム上のエクステントを持つファイルの一覧を表示するには、root 権限が必要です。

次の例では、fsmap コマンドと fsvmap コマンドの一般的な使い方について説明します。

fsmap コマンドの使用例

- find コマンドを使ってディレクトリを再帰的に下り、ファイルの一覧で fsmap を実行します。

```
# find . | fsmap -
Volume  Extent Type    File
vol2    Data           ./file1
vol1    Data           ./file2
```

fsvmap コマンドの使用例

1 複数のボリュームにおけるファイルのエクステントを出力します。

```
# fsvmap /dev/vx/rdsk/fstest/testvset vol1 vol2
vol1  /.
vol1  /ns2
vol1  /ns3
vol1  /file1
vol2  /file1
vol2  /file2
```

2 すべての Storage Checkpoint における単一ボリュームでデータまたはメタデータのいずれかを含むファイルのエクステントを出力し、ボリュームにファイルシステムのメタデータが含まれているかどうかを示します。

```
# fsvmap -mvC /dev/vx/rdsk/fstest/testvset vol1
Meta  Structural   vol1  //volume has filesystem metadata//
Data  UNNAMED      vol1  /.
Data  UNNAMED      vol1  /ns2
Data  UNNAMED      vol1  /ns3
Data  UNNAMED      vol1  /file1
Meta  UNNAMED      vol1  /file1
```

負荷分散

分散の割り当て順序を割り当てポリシーに定義し、そのポリシーを、特定のボリュームセットにランダムに分散して割り当てる必要のあるファイルに適用することができます。これらのファイルに関連付けされる各エクステントは、必須チャンクサイズとして割り当てポリシーに定義された最大サイズに制限されます。通常は、一杯かまたは無効になっているボリュームがない場合にエクステントが分散されます。

負荷分散割り当てポリシーは、個々のファイルまたはファイルシステム内の全ファイルに割り当てることができます。データエクステントを複数のボリュームに分散させる場合でも、負荷分散ポリシーを必要に応じてメタデータポリシーとして割り当てることができます。このとき制限は何もありません。

メモ: ファイルに固定エクステントサイズセットと負荷分散割り当てポリシーの両方がある場合、特定の操作が行われます。割り当てポリシーのチャンクサイズが固定エクステントサイズよりも大きい場合、ファイルのすべてのエクステントはチャンクサイズに制限されます。たとえば、チャンクサイズが 16 MB で、固定エクステントサイズが 3 MB の場合、両方の条件を満たす最大エクステントは 15 MB です。固定エクステントサイズがチャンクサイズよりも大きい場合、すべてのエクステントは固定エクステントサイズに制限されます。たとえば、チャンクサイズが 2 MB で、固定エクステントサイズが 3 MB の場合、ファイルのすべてのエクステントは 3 MB に制限されます。

負荷分散の割り当てポリシーの定義と割り当て

次の例では、負荷分散ポリシーを定義して、そのポリシーを /mnt/file.db ファイルに割り当てています。

負荷分散ポリシーの定義と割り当てを行うには

1 -o balance と -c のオプションを指定して負荷分散ポリシーを定義します。

```
  # fsapadm define -o balance -c 2m /mnt loadbal vol1 vol2 vol3
vol4

  # fsapadm define /mnt meta vol1
```

2 負荷分散ポリシーを割り当てます。

```
  # fsapadm assignfile /mnt/file.db loadbal meta
```

3 ファイルを適用します。

```
# fsapadm enforcefile /mnt/file.db
```

エクステントの再分散

厳密に割り当てポリシーを実施することでエクステントを再分散できます。一般的に、ポリシーからボリュームが追加または削除された場合、またはチャンクサイズが修正された場合に再分散が必要です。ボリュームがボリュームセットから削除される場合、ボリューム上の削除されるすべてのエクステントは、ポリシー内の別のボリュームに自動的に再分散されます。

次は、4 つのボリュームが指定されたポリシーに 2 つのボリュームの追加と、1 つの既存ボリュームの削除を再定義し、次にポリシーを実施して再分散を行う例です。

エクステントを再分散するには

1 -o balance と -c のオプションを指定して負荷分散ポリシーを定義します。

```
# fsapadm define -o balance -c 2m /mnt loadbal vol1 vol2 vol4
¥
vol5 vol6
```

2 負荷分散ポリシーを適用します。

```
# fsapadm enforcefile -f strict /mnt/filedb
```

MVS ファイルシステムの単一ボリュームファイルシステムへの変換

MVS ファイルシステムにおいてボリューム間でデータの再配置が行えるため、単一ボリューム上のファイルシステムデータすべてを移動させて、MVS ファイルシステムを従来の単一ボリュームファイルシステムへと変換させることができます。このような変換は、MVS ファイルシステムまたは SmartTier を使うユーザーにとって有効ですが、MVS ファイルシステムの永続的な使用を保証するものではありません。

p.831 の「SmartTier について」を参照してください。

この操作には、次の 3 つの制限事項が設けられています。

- 単一ボリュームはボリュームセットで最初のボリュームとしなければならない。
- 最初のボリュームには、データとファイルシステムのメタデータをすべて格納するうえで十分な空き領域を必要とする。
- ボリュームは、データの移動を制限するいかなる割り当てポリシーをも所持できない。

次は、ボリュームセット vset1 にある既存の MVS ファイルシステム /mnt1 を、ディスクグループ dg1 のボリューム vol1 にある単一ボリュームファイルシステム /mnt1 に変換する手順を示しています。

メモ: 手順 5、6、7、および 8 はオプションとなっており、ボリュームセットオブジェクトのラッパーを削除する場合に実施します。

単一ボリュームファイルシステムへの変換

1. ボリュームセットにある、デバイス番号 0 で識別される最初のボリュームに、削除する他のボリュームからデータを取得するための容量が確保されていることを確認します。

```
# df /mnt1
/mnt1  (/dev/vx/dsk/dg1/vol1):16777216 blocks  3443528 files
```

2. 最初のボリュームに十分な容量が確保されていない場合は、容量のサイズを大幅に増やします。

```
# fsvoladm resize /mnt1 vol1 150g
```

3. 既存の割り当てポリシーをすべて削除します。

```
# fsppadm unassign /mnt1
```

4. ボリュームセットの最初のボリュームを除き、すべてのボリュームを削除します。

```
# fsvoladm remove /mnt1 vol2
# vxvset -g dg1 rmvol vset1 vol2
# fsvoladm remove /mnt1 vol3
# vxvset -g dg1 rmvol vset1 vol3
```

ファイルシステムでは、ボリュームを削除する前に、そのボリューム上のファイルを再配置します。ファイルの再配置を正常に実行するには、別のボリュームに領域が必要になります。割り当てポリシーでは、ファイルをボリュームに定着させることはできません。コマンドの実行が完了するまでの時間は、再配置するデータ量に比例します。

5. 次のように入力して、ファイルシステムをマウント解除します。

```
# umount /mnt1
```

6. ボリュームセットから対象のボリュームを削除します。

```
# vxvset -g dg1 rmvol vset1 vol1
```

7. `/etc/fstab` ファイルを編集して、ボリュームセット名 `vset1` をボリュームデバイス名 `vol1` に変更します。
8. 次のコマンドを実行して、ファイルシステムをマウントします。

```
# mount -t vxfs /dev/vx/dsk/dg1/vol1 /mnt1
```

37

SmartTier の管理

この章では以下の項目について説明しています。

SmartTier について

Veritas File System（VxFS）の上で機能する SmartTier 機能を通じて、複数階層のオンラインストレージを使います。MVS ファイルシステムは、2 つ以上の仮想ボリュームを占有するファイルシステムです。ボリュームの集まりはボリュームセットと呼ばれます。ボリュームセットは、1 つの VxVM（Veritas Volume Manager）ディスクグループに属するディスクまたはディスクアレイ LUN で構成されます。MVS ファイルシステムは、単一の名前空間を示し、これによって複数のボリュームの存在がユーザーとアプリケーションに透

過的になります。各ボリュームは管理者用に個別の ID 情報を維持し、これによって各ファイルが配置される場所を制御できます。

p.815 の「MVS ファイルシステムについて」を参照してください。

メモ: 配置ポリシーの管理をよりユーザーフレンドリーなものとするため、リリース 4.1 から現在のリリースの間で、変更または削除されたコマンドがあります。削除されたコマンドは `fsrpadm`、`fsmove`、`fssweep` です。`fsapadm` コマンドの `queryfile`、`queryfs`、`list` オプションの出力では、割り当てが番号順ではなく名前順に出力されます。

以前の VxFS 5.x リリースでは、SmartTier は Dynamic Storage Tiering と呼ばれていました。

MVS VxFS ファイルシステムの管理者は、SmartTier で配置ポリシーを定義することで、ボリュームセット内の個々のボリューム上でファイルと、ファイルの各部分をどのように配置するかを管理できます。配置ポリシーでは、初期のファイルの場所と、既存ファイルが再配置される環境を制御します。これらの配置ポリシーをファイルに適用すると、ファイルシステムのボリュームセットの特定のサブセット(配置クラスと呼ばれる)上でそのファイルを作成し、拡張できます。ファイルは、指定した名前、タイミング、アクセス率、ストレージ容量に関連した条件に一致すると、他の配置クラス内のボリュームに再配置されます。

ボリュームタグをボリュームに関連付けることによって、配置クラスの VxVM ボリューム部分を作成します。ファイルを配置する目的で、VxFS は配置クラスのボリュームをすべて同等に扱い、ボリューム間に領域割り当てを分散します。ボリュームには、2 つ以上のタグを関連付けることができます。ボリュームに複数のタグを関連付けた場合、そのボリュームは複数の配置クラスに属し、いずれかの配置クラスに関連する割り当てポリシーと再配置ポリシーの影響を受けます。複数のタグ付けは慎重に行ってください。

p.833 の「配置クラス」を参照してください。

VxFS では、配置クラスに容量、パフォーマンス、可用性などの制約はありません。配置クラスにはあらゆるボリュームを追加できます。そのボリュームの種類や配置クラス内の他のボリュームの種類は関係ありません。ただし、同じ配置クラスには、同様の I/O パフォーマンスや可用性を持つボリュームを配置することが望まれます。

SmartTier によるファイルの圧縮について

SmartTier 機能を使用して、配置ポリシーで定義したルールに基づいて自動的にファイルを圧縮または圧縮解除できます。SmartTier はポリシーで指定した階層から直接選択したファイルの圧縮または圧縮解除エクステントの割り当てを実行します。選択したファイルは、ストレージの指定階層に再配置される間に圧縮または圧縮解除されます。

階層全体のインプレース圧縮を実行し、階層のすべてのファイルの未圧縮エクステントをすべて圧縮できます。この操作は、階層のファイルに書き込みまたは追加が実行された結果、ファイルに未圧縮のエクステントが存在する場合に便利です。

SmartTier はデフォルトで圧縮アルゴリズムとして gzip を使用し、デフォルトの圧縮ブロックサイズは 1 MB になります。これらのデフォルト値を XML のポリシーファイルを通して設定することはできません。

SmartTier は配置ポリシーの指定に従い、次の方法でファイルを圧縮または圧縮解除できます。

- MVS ファイルシステムで階層を渡ってファイルを再配置する間に圧縮します
- MVS ファイルシステムで階層を渡ってファイルを再配置する間に圧縮解除します
- MVS ファイルシステムでインプレース圧縮します
- MVS ファイルシステムでインプレース圧縮解除します
- 単一のファイルシステムでインプレース圧縮します
- 単一のファイルシステムでインプレース圧縮解除します
- MVS ファイルシステムで階層全体を圧縮します
- MVS ファイルシステムで階層全体を圧縮解除します

p.932 の「圧縮ファイルについて」を参照してください。

サポートされる SmartTier 文書型定義

表 37-1 は、Veritas File System（VxFS）リリースがサポートする SmartTier 文書型定義を示しています。

表 37-1　サポートされる SmartTier 文書型定義

VxFS のバージョン	DTD のバージョン	
	1.0	1.1
5.0	サポート対象	サポートなし
5.1	サポート対象	サポート対象
5.1 SP1	サポート対象	サポート対象
6.0	サポート対象	サポート対象
6.0.1	サポート対象	サポート対象

配置クラス

配置クラスは、MVS ファイルシステムのボリュームセットに存在する特定のボリュームの SmartTier 属性です。この属性は文字列で、ボリュームタグと呼ばれます。1 つのボリュー

ムには複数の異なるタグをもつことができ、そのうち 1 つを配置クラスにできます。配置クラスタグが、SmartTier にボリュームを識別させます。

ボリュームタグは階層構造を持つ名前空間となっており、その各階層レベルはそれぞれピリオドで区切られています。慣例では、ボリュームタグ階層の最上位レベルは、タグを 1 つ使う Storage Foundation Cluster File System High Availability のコンポーネントやアプリケーションを意味し、その 1 つ下の階層がタグの目的を示します。SmartTier では、フォーム `vxfs.placement_class.`*`class_name`* のボリュームタグを識別することができます。タグは、接頭辞 `vxfs` によって VxFS に関連付けられると識別されます。`placement_class` 文字列によって、タグは SmartTier が使うファイル配置クラスとして識別されます。*`class_name`* 文字列は、タグが関連付けているボリュームが属しているファイル配置クラスの名前を表現しています。たとえば、vxfs.placement_class.tier1 タグが付いたボリュームは、配置クラス `tier1` に属します。管理者は、`vxassist` コマンドを使ってタグをボリュームに関連付けさせることができます。

`vxassist`(1M)マニュアルページを参照してください。

SmartTier のポリシールールでは、個々のボリュームという観点よりもむしろ配置クラスという観点でファイル配置を規定しています。ファイルの作成や再配置に関しては、特定の配置クラスに属するボリュームであればどのボリュームも同様に機能します。 特定のボリュームというよりも配置クラスの観点でファイル配置を規定すると、多層ストレージの管理が単純化されます。

多層ストレージの管理がどのように単純化されるかを次にまとめます

- ボリュームの追加または削除においては、ファイル配置ポリシーを変更する必要はありません。`vxfs.placement_class.tier2` のタグ値を含むボリュームがファイルシステムのボリュームセットに追加された場合は、`tier2` を参照するすべてのポリシーを、新しく追加されたボリュームに対して、管理アクションを使わず即時に適用できます。同様に、ボリュームが空の状態(つまりデータが削除された状態)となる場合があります。またボリュームがポリシーの変更がない状態でファイルシステムから削除される可能性があります。有効なポリシーは、引き続きファイルシステムの残りすべてのボリュームに適用されます。
- ファイル配置ポリシーは、ファイルシステム固有ではありません。ファイル配置ポリシーは、そのポリシーで名前を定義したタグ値(配置クラス)のボリュームを含んだボリュームセットを持つ、すべてのファイルシステムに適用できます。この特性により、多くのサーバーを使ったデータセンターにおいて標準の配置ポリシーを定義して、1 回の管理アクションですべてのサーバーにそのポリシーを均一に適用させることができます。

配置クラスとしてのボリュームのタグ付け

次の例では、`vxassist settag` を使って、`vsavola`、`vsavolb`、`vsavolc`、`vsavold` に対してそれぞれ配置クラス `tier1`、`tier2`、`tier3`、`tier4` としてタグ付けを行います。

ボリュームにタグを付ける方法

◆ ボリュームを配置クラスとしてタグを付けます。

```
# vxassist -g cfsdg settag vsavola vxfs.placement_class.tier1
# vxassist -g cfsdg settag vsavolb vxfs.placement_class.tier2
# vxassist -g cfsdg settag vsavolc vxfs.placement_class.tier3
# vxassist -g cfsdg settag vsavold vxfs.placement_class.tier4
```

配置クラスのリスト

配置クラスは、vxassist listtag コマンドを使って一覧で表示されます。

vxassist(1M)マニュアルページを参照してください。

次の例は、ディスクグループ cfsdg のボリューム vsavola に設定されている配置クラスを含むボリュームタグをすべて一覧で表示しています。

配置クラスを一覧で表示する方法

◆ 配置クラスを含むボリュームタグを一覧で表示します。

```
# vxassist -g cfsdg listtag vsavola
```

配置ポリシーの管理

VxFS ファイル配置ポリシー文書にはルールが含まれます。VxFS はこのルールによりファイルを作成、再配置、削除しますが、配置ポリシーは特定のファイルシステムまたはボリュームを参照しません。fsppadm コマンドまたは GUI を使って配置ポリシー文書をファイルシステムに割り当てることで、ファイルシステムのアクティブなファイル配置ポリシーを作成することができます。

fsppadm(1M)のマニュアルページを参照してください。

メモ: 異なるターミナルから fsppadm コマンドを同時に実行しないでください。

fsppadmコマンドを使用するには、lost+found が存在している必要があります。

どのような場合でも、多くて 1 つのファイル配置ポリシーしか VxFS ファイルシステムには割り当てられません。ファイルシステムにファイル配置ポリシーを割り当てないことも可能です。この場合、VxFS は独自の内部アルゴリズムに従って新しいファイルの領域を割り当てます。

SFMS(Storage Foundation Management Server)ソフトウェアがインストールされたシステムでは、ファイル配置ポリシーの情報は SFMS データベースに格納されます。SFMS

データベースには、XML ポリシー文書や、その各文書が現在アクティブなポリシーとなっているホストやファイルシステムのリストが含まれています。ポリシー文書が更新される時に、SFMS は、現在その文書を基にポリシーがアクティブになっているすべてのファイルシステムに、更新された文書を割り当てることができます。デフォルトでは、SFMS は、ファイルシステムのアクティブなポリシーがローカル（その配置ポリシーのファイルシステムを制御するホスト）で作成または修正されている場合、そのポリシーを更新しません。SFMS 管理者が配置ポリシーをファイルシステムに強制的に割り当てた場合、そのファイルシステムのアクティブな配置ポリシーは上書きされ、ローカルで配置ポリシーに加えられた変更はすべて失われます。

/opt/VRTSvxfs/etc ディレクトリには配置ポリシーのサンプルがあります。配置ポリシーのサンプルは、VxFS RPM のインストール時にインストールされます。

配置ポリシーの割り当て

次の例では、fsppadm assign コマンドを使い、マウントポイント /mnt1 のファイルシステムに対して、XML ポリシー文書 /tmp/policy1.xml で定義されているファイル配置ポリシーを割り当てます。

配置ポリシーを割り当てる方法

- 配置ポリシーを特定のファイルシステムに割り当てます。

```
# fsppadm assign /mnt1 /tmp/policy1.xml
```

配置ポリシーの割り当て解除

次の例では、fsppadm unassign コマンドを使って、マウントポイント /mnt1 のファイルシステムからアクティブなファイル配置ポリシーの割り当てを解除します。

配置ポリシーの割り当てを解除する方法

- 配置ポリシーを特定のファイルシステムから割り当て解除します。

```
# fsppadm unassign /mnt1
```

配置ポリシーの実施に伴う領域への影響の分析

次の例では、fsppadm analyze コマンドを使って、XML ポリシー文書 /tmp/policy1.xml に表現されているファイル配置ポリシーをマウントポイント /mnt1 に実施した場合の影響を分析します。このコマンドでは、必要に応じて I/O 頻度データベースを構築できます。

配置ポリシーの実施に伴う領域への影響を分析する方法

- XML のポリシー文書 /tmp/policy1.xml に表現されているファイル配置ポリシーをマウントポイント /mnt1 に実施した場合の影響を分析します。

```
# fsppadm analyze -F /tmp/policy1.xml -i /mnt1
```

配置ポリシーの実施により影響を受けるファイルの問い合わせ

次の例では、fsppadm query コマンドを使って、配置ポリシーの実施により影響を受けるファイルの一覧を生成します。またこのコマンドでは、対象となるファイルの現在位置情報(ファイルが再配置される場所)と、ファイルに適用されている配置ポリシーの種類を表示することができます。

配置ポリシーの実施により影響を受けるファイルを問い合わせる方法

- 影響を受けるファイルを問い合わせます。

```
# fsppadm query /mnt1/dir1/dir2 /mnt2 /mnt1/dir3
```

配置ポリシーの実施

ファイルシステムに配置ポリシーを実施するには、そのポリシーがあらかじめファイルシステムに割り当てられている必要があります。配置ポリシーを実施する前に割り当てを行ってください。

p.836 の「配置ポリシーの割り当て」を参照してください。

実施処理の内容は、マウントポイントの lost+found ディレクトリ内の隠しファイル .__fsppadm_enforce.log にログとして記録されます。このログファイルには、ファイルの以前の格納場所、ファイルの新しい格納場所、ファイルを再配置した目的などが記されています。実施処理では、ログファイルが見つからないと .__fsppadm_enforce.log ファイルを生成します。ログファイルがすでに存在している場合は、実施処理の内容を追記します。通常のファイルと同様に、.__fsppadm_enforce.log ファイルは、バックアップを作成することも、削除することもできるようになっています。

-F オプションを使うことで、既存のアクティブな配置ポリシーでない配置ポリシーを指定できます。このオプションを使って、ファイルシステムから LUN を再生するなどの保守作業のために特定の配置ポリシーに含まれるルールを実施することができます。

-p オプションを使うことで、fsppadm 操作を実行するために使う同時スレッド数を指定できます。*io_nice* パラメータは 1 から 100 の整数として指定します。50 がデフォルト値です。指定する値が 1 の場合は、1 回のマウントで 1 つのスレーブと 1 つのマスタースレッドが生成されます。指定する値が 50 の場合は、1 回のマウントで 16 のスレーブと 1 つのマスタースレッドが生成されます。指定する値が 100 の場合は、1 回のマウントで 32 のスレーブと 1 つのマスタースレッドが生成されます。

fsppadm コマンドに -c オプションを指定することで、配置ポリシーに指定されている期間にアクティビティ統計情報が FCL（File Change Log）ログファイルに記録されたスレッドだけを処理できます。-c オプションは、ポリシーの ACCESSTEMP または IOTEMP 要素で Prefer 基準を使う場合にのみ指定できます。

fsppadm コマンドに -t オプションを指定することで、スイープおよび再配置するファイルが含まれる配置クラスを指定できます。-t オプションは、ポリシーで IOTEMP のポリシーの Prefer 基準を使う場合にのみ指定できます。

fsppadm（1M）のマニュアルページを参照してください。

次の例では、fsppadm enforce コマンドを使って、マウントポイント /mnt1 にあるファイルシステムのファイル配置ポリシーを実施します。またこの例では、レポート /tmp/report で指定したパスに関するアクセス時間、修正時間、ファイルサイズが示されています。

配置ポリシーを実施する方法

◆ 特定のファイルシステムに対して配置ポリシーを実施します。

```
# fsppadm enforce -a -r /tmp/report /mnt1
Current Current Relocated Relocated
Class   Volume  Class     Volume    Rule   File
tier3   vole    tier3     vole      a_to_z /mnt1/mds1/d1/file1
tier3   vole    tier3     vole      a_to_z /mnt1/mds1/d1/file2
tier3   vole    tier3     vole     a_to_z /mnt1/mds1/d1/d2/file3
tier3   volf    tier3     volf     a_to_z /mnt1/mds1/d1/d2/file4
.
.
.
Sweep path    : /mnt1
Files moved   : 42
KB moved      : 1267


Tier Name       Size (KB)   Free Before (KB)   Free After (KB)
tier4              524288             524256            524256
tier3              524288             522968            522968
tier2              524288             524256            524256
tier1              524288             502188            501227
```

配置ポリシーの有効性確認

次の例では、fsppadm validate コマンドを使って、マウントされたすべてのファイルシステムに対して配置ポリシー policy.xml の有効性を確認します。

マウントされたすべてのファイルシステムに対して配置ポリシーの有効性を確認する方法

◆ 配置ポリシーの有効性を確認します。

```
# fsppadm validate /tmp/policy.xml
```

ファイル配置ポリシーの文法

VxFS は、MVS ファイルシステム内でファイルの割り当てと再配置を行います。この操作は、ファイルに関連するファイルシステムメタデータのプロパティを元に行われます。配置は、ファイル名、保管ディレクトリ、最終アクセス時間、アクセス頻度、ファイルサイズ、所有権により決められます。個々のファイルシステムの、ファイルの割り当てと再配置に対する基準は、ファイルシステムのファイル配置ポリシーに示されます。

VxFS ファイル配置ポリシーには、VxFS MVS ファイルシステムのボリューム上での、ファイルセットの配置が定義されます。ファイル配置ポリシーには、どこの配置クラスのボリュームにファイルを作成するか、どこの代替配置クラスのボリュームにどのような条件でファイルを再配置するか、またどこの場所のファイルをどのような条件で削除するかを指定します。ファイル配置ポリシー文書は XML テキストファイルで、XML エディタ、テキストエディタ、VOM（Veritas Operations Manager）を使って作成できます。

配置ポリシーの全体構造については、`/opt/VRTSvxfs/etc/placement_policy.dtd` ファイルを参照してください。

ファイル配置ポリシーのルール

VxFS ファイル配置ポリシーは 1 つ以上のルールで構成されます。各ルールを 1 つ以上のファイルに適用します。ルールを適用するファイルは 1 つ以上の `SELECT` 文に指定します。`SELECT` 文では、ファイル名または名前パターン、ファイルが保管されているディレクトリ、ファイル所有者のユーザー名、ファイル所有者のグループ名、この 4 つのプロパティのうち 1 つ以上に従ってファイルを指定します。

1 つのファイルに 2 つ以上のルールが指定されることがあります。たとえば、あるルールをディレクトリ `/dir` 内のファイルに指定して、別のルールを所有者が `user1` のファイルに指定すると、`/dir` の中にある、所有者が `user1` のファイルには 2 つのルールが指定されます。ファイルには、配置ポリシーの中で最初に記述された方のルールのみが適用され、以降のルールは無視されます。

ファイルシステム全体の名前空間を網羅しない配置ポリシーを定義できます。作成されたファイルに、ファイルシステムのアクティブな配置ポリシー内のルールが何も指定されていない場合、VxFS は独自の内部アルゴリズムに従ってそのファイルを配置します。ファイル配置に対してフルコントロールを維持するには、各配置ポリシー文書の最後に、`SELECT` 文のファイル指定に名前パターンの「`*`」を指定した包括ルールを組み込みます。配置ポ

リシー文書に記述されたここまでのルールでは指定されなかったすべてのファイルをこのルールで指定します。

data、ckpt という 2 種類のルールがあります。data ルールでは、通常のデータファイルを再配置できます。ckpt ルールでは、Storage Checkpoint を再配置できます。ルールの種類は、ルールに Flags 属性を設定することで指定します。

SELECT 文

VxFS 配置ポリシールールの SELECT 文は、ルールを適用するファイルの集合を指定します。

次の XML の抜粋は、一般的な SELECT 文の形式を示します。

```
<SELECT>
  <DIRECTORY Flags="directory_flag_value"> value
  </DIRECTORY>
  <PATTERN Flags="pattern_flag_value"> value </PATTERN>
  <USER> value </USER>
  <GROUP> value </GROUP>
</SELECT>
```

SELECT 文では、次の選択基準を使ってファイルを指定できます。

<DIRECTORY>　ファイルシステムのマウントポイントに対する絶対パス名。Flags="directory_flag_value" XML 属性の値を nonrecursive にして指定ディレクトリのファイルのみを指定するか、または recursive にして指定ディレクトリのすべてのサブディレクトリのファイルを指定する必要があります。Flags 属性は必須です。

<DIRECTORY> 基準はオプションです。2 回以上指定できます。

<PATTERN>

ファイルの正確な名前またはワイルドカード文字(*)1 つを使ったパターンのいずれかです。たとえば、パターン「abc*」は、名前が「abc」で始まるすべてのファイルを示します。パターン「abc.*」は、名前が厳密に「abc」で、その後にピリオドと任意の拡張子が続くすべてのファイルを示します。パターン「*abc」は、名前が「abc」で終わるすべてのファイルを示します。その名前が拡張子のすべて、または一部であるかは問いません。パターン「*.abc」は、拡張子(ピリオドの後)の名前が「abc」の任意の名前のファイルを示します。パターン「ab*c」は、名前が「ab」で始まり「c」で終わるすべてのファイルを示します。一番初めの「*」文字はワイルドカードと見なされますが、以降の「*」文字はリテラルテキストと見なされます。パターンに「/」を含めることはできません。

ワイルドカード文字は、シェルで使用する場合とは異なり、「.」、「?」、「[」などのすべての文字に一致します。

Flags="pattern_flag_value" XML 属性はオプションです。これには recursive 値のみを指定できます。パターンがディレクトリの場合にのみ、Flags="recursive" を指定します。Flags を指定しない場合、デフォルトの属性値は非再帰的です。Flags="recursive" を指定する場合、<DIRECTORY> に指定したディレクトリよりも下位で、パターンに一致するコンポーネントディレクトリがあり次の条件のいずれかを満たす場合、囲まれた選択基準はそのコンポーネントディレクトリのすべてのファイルを選択します。

- <DIRECTORY> が指定され再帰的フラグがある。
- <DIRECTORY> の指定がなく、ディレクトリがファイルシステム内にある。

パターンにワイルドカード文字「*」が含まれている場合、ワイルドカード文字の照合が行われます。

<PATTERN> 基準はオプションです。2 回以上指定できます。値は <PATTERN> 要素ごとに 1 つだけ指定できます。

<USER>

ファイル所有者のユーザー名です。ユーザー名の代わりにユーザー番号を指定することはできません。

<USER> 基準はオプションです。2 回以上指定できます。

<GROUP>

ファイル所有者のグループ名です。グループ名の代わりにグループ番号を指定することはできません。

<GROUP> 基準はオプションです。2 回以上指定できます。

1 つの SELECT 文の中に、すべてのファイル選択基準のインスタンスを 1 つ以上指定できます。異なる 2 種類以上の選択基準が 1 つの文に指定された場合、選択されるファイルは各種類の基準を 1 つ満たす必要があります。

次の例が実行されると、ora/db または crash/dump のいずれかに存在し、所有者が user1 または user2 のいずれかのファイルのみが選択されます。

```
<SELECT>
  <DIRECTORY Flags="nonrecursive">ora/db</DIRECTORY>
  <DIRECTORY Flags="nonrecursive">crash/dump</DIRECTORY>
  <USER>user1</USER>
  <USER>user2</USER>
</SELECT>
```

1 つのルールには複数の SELECT 文を含めることができます。ファイルは、1 つの SELECT 文の選択基準を満たせば、処理の対象となります。

次の例では、所有者が user1 または user2 のファイル（保存されたディレクトリは関係なし）と、ディレクトリ ora/db または crash/dump 内のファイル（所有するユーザーは関係なし）が処理の対象となります。

```
<SELECT>
  <DIRECTORY Flags="nonrecursive">ora/db</DIRECTORY>
  <DIRECTORY Flags="nonrecursive">crash/dump</DIRECTORY>
</SELECT>
<SELECT>
  <USER>user1</USER>
  <USER>user2</USER>
</SELECT>
```

VxFS で新しいファイルを作成する場合、VxFS は、アクティブな配置ポリシーの XML ソースファイルに記述された順番に従って、アクティブな配置ポリシールールを適用します。作成するファイルを SELECT 文で指定している 1 番目のルールで、ファイルの配置を決定します。以降のルールは適用しません。同様に、ファイルを再配置するときに、VxFS は各ファイルに代わってアクティブなポリシールールをスキャンします。ファイルを指定した SELECT 文を含むルールが最初に見つかった時点でルールのスキャンは停止します。この適用可能なルールが結果的に何も処理しない場合でも、スキャンは停止します。たとえば、あるポリシールールに 30 日間アクティブになっていない .dat ファイルは再配置することが示されているとします。また以降のルールに、10 MB よりも大きな .dat ファイルは再配置することが示されているとします。10 日間アクティブになっていない 20 MB の .dat ファイルは再配置されません。それは前者のルールが適用されたからです。後者のルールはスキャンされません。

配置ポリシールールの動作文は、ルールの SELECT 文で指定されたすべてのファイルに適用されます。既存ファイルが、ファイルシステムのアクティブな配置ポリシールールの SELECT 文で指定されていない場合、SmartTier はそのファイルを再配置、削除しません。アプリケーションでファイルが作成されたときに、そのファイルがファイルシステムのアクティブなポリシールールの SELECT 文で指定されていない場合、VxFS は独自の内部アルゴリズムに従ってそのファイルを配置します。この方法が適切でない場合は、ファイルシステムのアクティブな配置ポリシーの基準になっているポリシー文書の最後のルールで、SELECT 文に <PATTERN>*</PATTERN> のみの選択基準を指定し、CREATE 文に他のルールで選択されないファイルの配置クラスの名前を指定します。

CREATE 文

配置ポリシールールの CREATE 文で、ボリュームに 1 つ以上の配置クラスを指定します。VxFS はルールが適用されるファイルを新しく作成するときにそのファイル用の領域を指定した配置クラスに割り当てます。CREATE 文には配置クラスのみを指定できます。個々のボリューム名は指定できません。

ファイル配置ポリシールールに、CREATE 文は 1 つしか含めることはできません。ルールに CREATE 文が含まれていない場合、VxFS は、ルールの SELECT 文に指定されたファイルを、その内部アルゴリズムに従って配置します。ただし、ルールに CREATE 文がなくても、ルールの SELECT 文に指定した既存ファイルの再配置または削除に使えます。

次の XML の抜粋は、一般的な CREATE 文の形式を示します。

```
<CREATE>
  <ON Flags="flag_value">
    <DESTINATION>
      <CLASS> placement_class_name </CLASS>
      <BALANCE_SIZE Units="units_specifier"> chunk_size
      </BALANCE_SIZE>
    </DESTINATION>
    <DESTINATION> additional_placement_class_specifications
    </DESTINATION>
  </ON>
</CREATE>
```

CREATE 文には 1 つの <ON> 節が含まれます。その中に 1 つ以上の <DESTINATION> XML 要素を記述し、初期ファイル割り当て用の配置クラスを優先度の高いものから順に指定します。VxFS は、指定された 1 番目のクラスのボリュームに利用可能な領域があれば、そこにルールが適用された新しいファイル用の領域を割り当てます。1 番目のどのボリュームにも領域を割り当てられない場合、VxFS は指定された 2 番目のクラスのボリュームに利用可能な領域があれば領域を割り当てます。以下同様に続けていきます。

指定した配置クラスのどのボリュームにも領域を割り当てることができない場合、ファイルシステムのボリュームセット以外に十分な領域があったとしても、ファイル作成は失敗し、ENOSPC エラーが出力されます。Flags 属性の <ON> 節に「any」値を指定すれば、この状況を避けることができます。CREATE 文に <ON Flags="any"> が指定されている場合、VxFS は最初に、ルールが適用された新しいファイル用の領域を指定の配置クラスに割り当てようとします。これに失敗した場合、VxFS はその内部領域割り当てアルゴリズムを用います。したがってファイルの割り当ては、ファイルシステムのボリュームセットのどの場所にも利用可能な領域がない限り失敗しません。

Flags="any" 属性は包括ルールとは異なり、ルールの SELECT 文で指定したファイルのみに適用します。包括ルールのファイル選択で <PATTERN>*</PATTERN> を指定するよりも包括の度合いを少なくできます。

<DESTINATION> XML 要素には、<CLASS> 副要素に指定した配置クラス名に加え、<BALANCE_SIZE> 副要素を含めることができます。<BALANCE_SIZE> 要素を指定すると、領域の割り当ては指定したサイズのチャンクで配置クラスの全ボリュームに分散されます。たとえば、3 つのボリュームが含まれる配置クラスで分散サイズ 1 MB が指定されている場合、VxFS は新しいファイルまたは拡張ファイルの領域の最初の 1 MB を、そのクラスの 1 番目（最小インデックス）のボリュームに割り当てます。次の 1 MB は 2 番目のボリューム、その次の 1 MB は 3 番目のボリューム、その次の 1 MB は 1 番目のボリューム、以降同様に割り当てます。<BALANCE_SIZE> XML タグで Units を使えば分散サイズを次の単位で指定できます。

bytes	バイト
KB	キロバイト
MB	メガバイト
GB	ギガバイト

<BALANCE_SIZE> 要素を使うと、データベースファイルは配置クラスの全ボリュームに分散して割り当てられます。原則として、各ファイルデータを複数ボリュームに分散すると、I/O 負荷も複数ボリュームに分散されます。

次の例の CREATE 文では、ルールが適用されたファイルは tier1 ボリュームに領域があればそこに作成され、なければ tier2 ボリュームの 1 つに作成されます。tier1 と tier2 ボリュームへの領域の割り当てが不可能な場合、tier3 ボリュームに利用可能な領域があったとしても、ファイル作成は失敗します。

```
<CREATE>
  <ON>
    <DESTINATION>
      <CLASS>tier1</CLASS>
    </DESTINATION>
    <DESTINATION>
      <CLASS>tier2</CLASS>
      <BALANCE_SIZE Units="MB">1</BALANCE_SIZE>
    </DESTINATION>
  </ON>
</CREATE>
```

tier2 ボリュームの割り当てで、<BALANCE_SIZE> 要素に 1 MB の値が指定されています。tier2 ボリュームに割り当てられるファイルでは、最初の 1 MB は 1 番目のボリュームに割り当てられ、次の 1 MB は 2 番目のボリューム、以下同様に割り当てられます。

RELOCATE 文

ファイル配置ポリシールールの RELOCATE 動作文には、ファイルシステムを定期スキャンするときに、指定されたファイルに対して VxFS が実行する動作とその動作の実行環境を指定します。fsppadm enforce コマンドを使って、ファイルシステムのすべてまたは一部をスキャンし、アクティブな配置ポリシーのルールに従って、スキャン時に再配置が必要なファイルを探します。

fsppadm(1M)のマニュアルページを参照してください。

fsppadm enforce コマンドは、ファイルシステムをパス名の順でスキャンします。VxFS は、各ファイルに対し、ルールの SELECT 文を判別して、アクティブな配置ポリシーの中で最初に適用可能なルールを特定します。ルールのいずれかの RELOCATE 文の <FROM> 節に指定されたボリュームに存在するファイルが、文の <WHEN> 節に指定された配置基準を満たす場合、そのファイルは、<TO> 節に記述された配置クラスの中で、ファイルを格納する領域のある最初の配置クラスのボリュームに、再配置されるようにスケジュールされます。fsppadm enforce コマンドの発行により実行されたスキャンが完了してから、ファイルは再配置されます。

次の XML の抜粋は、一般的な RELOCATE 文の形式を示します。

```
<RELOCATE>
  <FROM>
    <SOURCE>
      <CLASS> placement_class_name </CLASS>
    </SOURCE>
    <SOURCE> additional_placement_class_specifications
    </SOURCE>
  </FROM>
  <TO>
    <DESTINATION>
      <CLASS> placement_class_name </CLASS>
      <BALANCE_SIZE Units="units_specifier">
        chunk_size
      </BALANCE_SIZE>
    </DESTINATION>
    <DESTINATION>
      additional_placement_class_specifications
    </DESTINATION>
  </TO>
  <WHEN> relocation_conditions </WHEN>
</RELOCATE>
```

RELOCATE 文には次の節が含まれます。

- <FROM> - 再配置もとの配置クラスのリストを含めるオプションの節です。この配置クラスのボリュームの中の指定されたファイルが <WHEN> 節の条件を満たせば再配置されます。<FROM> 節に記述された配置クラスの順番に優先度はありません。ルールが適用されたファイルが、指定したいずれかの配置クラスのボリュームにある場合、そのファイルは再配置対象と見なされます。

  RELOCATE 文に <FROM> 節がある場合、VxFS は、この節に指定された配置クラスのボリュームに存在するファイルのみを再配置対象と見なします。<FROM> 節がない場合は、ファイルが存在する場所に関係なく、対象のファイルは再配置されます。

- <TO> - 対象ファイルの再配置先の配置クラスです。FROM 節の配置もとの配置クラスリストとは違い、<TO> 節の配置クラスは優先度順に指定します。ファイルの再配置は、可能な場合は 1 番目に指定された配置クラスのボリューム、不可能な場合は 2 番目、以降同様にしてボリュームが決まります。

  RELOCATE 文の <TO> 節には、配置クラスを指定する <DESTINATION> XML 要素のリストが含まれます。VxFS は、指定した配置クラスのボリュームに対象ファイルを再配置します。配置クラスは優先度順に指定します。VxFS は、領域が利用可能であれば、指定した最初の配置クラスのボリュームに対象ファイルを再配置します。

  <DESTINATION> 要素には、オプションの <BALANCE_SIZE> 修飾子副要素を含めることができます。<BALANCE_SIZE> 修飾子を設定すると、再配置されるファイルは、指定したサイズのチャンクで配置先の配置クラスの全ボリュームに分散されます。たとえば、3 つのボリュームが含まれる配置クラスに分散サイズ 1 MB が指定されたとすると、VxFS はファイルの最初の 1 MB を、そのクラスの 1 番目（最小インデックス）のボリュームに再配置します。次の 1 MB は 2 番目のボリューム、その次の 1 MB は 3 番目のボリューム、その次の 1 MB は 1 番目のボリューム、以降同様に再配置します。<BALANCE_SIZE> XML タグの Units 属性を使えば、分散サイズ値に指定するチャンク値を、バイト単位（Units="bytes"）、KB 単位（Units="KB"）、MB 単位（Units="MB"）、GB 単位（Units="GB"）で指定できます。

  <BALANCE_SIZE> 要素を使うと、データベースファイルは配置クラスの全ボリュームに分散して割り当てられます。原則として、各ファイルデータを複数ボリュームに分散すると、I/O 負荷も複数ボリュームに分散されます。

  MVS ファイルシステムシステムのために、compress フラグまたは uncompress フラグを <TO> 句で指定できます。compress フラグにより、<DESTINATION> 要素によって指定された階層にファイルを再配置している間に、SmartTier はファイルのエクステントを圧縮します。SmartTier は、ファイルスパンが複数の階層に渡っている場合にもファイル全体を圧縮し、指定した階層にファイルを再配置します。uncompress フラグにより、<DESTINATION> 要素によって指定された階層にファイルを再配置している間に、SmartTier はファイルのエクステントを圧縮解除します。
  次の XML の抜粋は、compress フラグを指定します:

```
<TO Flags="compress">
   <DESTINATION>
      <CLASS> tier4 </CLASS>
```

```
    </DESTINATION>
</TO>
```

次の XML の抜粋は、uncompress フラグを指定します:

```
<TO Flags="uncompress">
    <DESTINATION>
        <CLASS> tier4 </CLASS>
    </DESTINATION>
</TO>
```

- <WHEN> - ルールが適用されたファイルの再配置条件を指定するオプションの節です。指定した期間アクセスがないかまたは修正がないファイル、特定のサイズに達したファイル、特定の I/O 頻度またはアクセス頻度のレベルに達したファイルなどを再配置できます。RELOCATE 文に <WHEN> 節がない場合、ルールが適用されたファイルは無条件に再配置されます。
  RELOCATE 文に <WHEN> 節を含めると、4 つの基準のいずれかまたはすべてを満たす場合にのみファイルを再配置することができます。1 つ以上の基準を満たすファイルを再配置するように指定できます。

<WHEN> 節では、次の基準が指定できます。

<ACCAGE>	指定した期間にファイルがアクティブにならない場合、つまり fsppadm enforce コマンドの発行から指定した期間にファイルがアクティブにならない場合にこの基準が満たされます。
<MODAGE>	指定した期間にファイルの修正がない場合、つまり fsppadm enforce コマンドの発行から指定した期間にファイルの修正がない場合にこの基準が満たされます。
<SIZE>	ファイルが、指定したサイズを超えるまたは下回る場合、または指定したサイズ範囲内に収まる場合にこの基準が満たされます。
<IOTEMP>	ファイルが、指定した I/O 頻度を超えるまたは下回る場合、または指定した I/O 頻度範囲内に収まる場合にこの基準が満たされます。ファイルの I/O 頻度とは、fsppadm enforce コマンドの発行より前の <PERIOD> 要素に指定した期間における、ファイルの I/O 負荷の測定値です。 p.879 の「I/O 頻度とアクセス頻度の計算」を参照してください。
<ACCESSTEMP>	ファイルが、指定した平均アクセス頻度を超えるまたは下回る場合、または指定したアクセス頻度範囲内に収まる場合にこの基準が満たされます。ファイルのアクセス頻度はファイルの I/O 頻度と同じですが、アクセス頻度はファイルへの転送バイト数ではなく I/O 要求数を使って計算されます。

メモ: VxFS サーバーを NFS サーバーとして使う場合、<IOTEMP> と <ACCESSTEMP> を使ってサーバー上にデータを配置する動作が、NFS キャッシュのためにあまり効果的でない場合があります。NFS クライアント側キャッシュと NFS の動作によって、NFS サーバー側で I/O が発生せずに、NFS クライアントで I/O が発生することがあります。そのような場合、サーバー側で行われる頻度測定は、配置ポリシーに指定されている I/O 動作を正しく反映しません。

サーバーを NFS サーバーとしてのみ使う場合は、頻度しきい値を適切に調整するか下げることによって、この問題を軽減できる場合があります。ただし、しきい値を調整しても望ましい効果が得られないこともあります。また、同じマウントポイントを NFS エクスポートとローカルマウントとして使う場合は、NFS キャッシュスキューのために、頻度に基づいて配置を決定する方法はあまり効果的ではありません。

次の XML の抜粋は、RELOCATE 文の一般的な <WHEN> 節の形式を示します。

```
 <WHEN>
   <ACCAGE Units="units_value">
     <MIN Flags="comparison_operator">
       min_access_age</MIN>
     <MAX Flags="comparison_operator">
       max_access_age</MAX>
   </ACCAGE>
   <MODAGE Units="units_value">
     <MIN Flags="comparison_operator">
       min_modification_age</MIN>
     <MAX Flags="comparison_operator">
       max_modification_age</MAX>
   </MODAGE>
   <SIZE " Units="units_value">
     <MIN Flags="comparison_operator">
       min_size</MIN>
     <MAX Flags="comparison_operator">
       max_size</MAX>
   </SIZE>
   <IOTEMP Type="read_write_preference"
Prefer="temperature_preference">
     <MIN Flags="comparison_operator">
       min_I/O_temperature</MIN>
     <MAX Flags="comparison_operator">
       max_I/O_temperature</MAX>
     <PERIOD Units="days_or_hours"> days_or_hours_of_interest
</PERIOD>
   </IOTEMP>
```

```
   <ACCESSTEMP Type="read_write_preference"
    Prefer="temperature_preference">
     <MIN Flags="comparison_operator">
       min_access_temperature</MIN>
     <MAX Flags="comparison_operator">
       max_access_temperature</MAX>
     <PERIOD Units="days_or_hours"> days_or_hours_of_interest
</PERIOD>
   </ACCESSTEMP>
 </WHEN>
```

アクセス期間(<ACCAGE>)要素は、ファイルが最後にアクセスされてからの時間です。VxFS は、fsppadm enforce コマンドの発行時間からファイルの最終アクセス時間(atime)を引いて、アクセス期間を算出します。<ACCAGE> 節の <MIN> と <MAX> XML 要素は、それぞれ再配置のための最小アクセス期間と最大アクセス期間のしきい値を示します。これらの要素はオプションですが、少なくとも 1 つは含める必要があります。Units XML 属性を指定すれば、<MIN> と <MAX> 要素は次の単位で指定できます。

hours	時間
days	日。1 日は fsppadm enforce コマンドの発行より前の 24 時間として考えられます。

<MIN> と <MAX> 要素は両方とも Flags 属性を指定して操作を指示する必要があります。

<MIN> では、次の Flags 属性値を指定できます。

gt	最終アクセス時間は指定の間隔よりも大きい必要があります。
eq	最終アクセス時間は指定の間隔と等しい必要があります。
gteq	最終アクセス時間は指定の間隔よりも大きいか等しい必要があります。

<MAX> では、次の Flags 属性値を指定できます。

lt	最終アクセス時間は指定の間隔よりも小さい必要があります。
lteq	最終アクセス時間は指定の間隔よりも小さいか等しい必要があります。

<WHEN> 節に <MIN> 要素を含めると、VxFS はルールが適用されたファイルで指定した間隔よりも長い期間アクティブになっていないファイルを再配置します。通常、このようなルールを使って、アクティブになっていないファイルを安価なストレージ階層に再配置します。逆に、<MAX> は指定した間隔にアクセスのあったファイルを再配置します。通常この要素を使って、アクティビティを再開した非アクティブなファイルを、パフォーマンスが速

いかまたは信頼性の高いストレージに移動します。<MIN> と <MAX> 節の 2 つを含めると、VxFS はアクセス期間がこの 2 つの間に収まるファイルを再配置します。

修正期間の再配置基準(<MODAGE>)はアクセス期間と同じですが、ファイルの POSIX mtime 値が計算に使われます。通常、<MODAGE> 基準を指定するのは、最近修正されたファイルを今後反復的にアクセスされることを予想してパフォーマンスの速いまたは信頼性の高いストレージ階層に再配置するためです。

ファイルサイズの再配置基準(<SIZE>)を指定すると、fsppadm enforce コマンドが発行された時に、指定された再配置基準の <MIN> 値よりも大きいか、または <MAX> 値よりも小さいファイルが再配置されます。両方の基準を指定すると、VxFS はサイズがこの 2 つの間に収まるファイルを再配置するようにスケジュールします。Units 属性を使うと、ファイルサイズのしきい値を次の単位で指定できます。

bytes	バイト
KB	キロバイト
MB	メガバイト
GB	ギガバイト

I/O 頻度再配置基準の指定

I/O 頻度再配置基準(<IOTEMP>)を指定すると、fsppadm enforce コマンド発行直前の指定した期間に、指定値を超えるかまたは下回る I/O 頻度のファイルが再配置されます。ファイルの I/O 頻度とは、ファイルに対する読み取り、書き込み、つまり I/O 負荷合計の測定値で、ファイルサイズに対して正規化されます。高い I/O 頻度は、アプリケーションのアクティビティレベルが高いことを示します。逆に I/O 頻度が低いとアクティビティレベルが低いことになります。VxFS は、指定期間中のファイルへの(またはファイルからの)転送バイト数(読み取り、書き込み、またはその両方)を、fsppadm enforce コマンド発行時のファイルサイズで割って、ファイルの I/O 頻度を算出します。

p.879 の「I/O 頻度とアクセス頻度の計算」を参照してください。

他のファイル再配置基準と同様に、<IOTEMP> に <MIN> 要素を使って下位のしきい値を指定でき、<MAX> 要素を使って上位のしきい値、両方を使って範囲を指定できます。ただし、I/O 頻度に寸法はないので単位の指定はありません。

VxFS がファイルの I/O 頻度を算出するときの期間は、fsppadm enforce コマンド発行時から、過去 <PERIOD> 要素に指定した日数または時間数までの期間です。1 日は 24 時間を意味します。デフォルトの単位は日数です。<PERIOD> 要素の Units 属性を hours に設定することで、時間数を単位として指定できます。ソリッドステートディスク(SSD)を使う場合にのみ、時間数を指定することをお勧めします。

p.891 の「ソリッドステートディスクでの SmartTier のスキャン頻度」を参照してください。

たとえば、fsppadm enforce コマンドを水曜日の午後 2 時で発行したけれども、月曜日の午後 2 時から水曜日の午後 2 時まで(2 日間)のファイル I/O アクティビティを計算する場合は、次のように <PERIOD> 要素を指定します。

```
<PERIOD> 2 </PERIOD>
```

代わりに、fsppadm enforce コマンドを実行する 3 時間前からコマンドを実行した時点までのファイル I/O アクティビティを計算する場合は、次のように <PERIOD> 要素を指定します。

```
<PERIOD Units="hours"> 3 </PERIOD>
```

FCL(File Change Log)ファイルで使われるディスク領域のため、<PERIOD> に 1 週間または 2 週間を超える期間を指定しないでください。

p.1101 の「Veritas File System ファイルの変更ログファイルについて」を参照してください。

I/O 頻度は I/O 負荷の測定値でアクセス期間よりも滑らかな値です。アクセス期間では、ファイルへのただ 1 回のアクセスでファイルの atime は現在時刻にリセットされます。これに対して、ファイルの I/O 頻度は、ファイルへのアクセスがない間は徐々に減少し、ファイルが定期的にアクセスされると徐々に増加していきます。たとえば、月曜日に 10 MB の新しいファイルが作成され、5 回完全に読み取られた場合、深夜に fsppadm enforce を実行すると、このファイルの 2 日間の I/O 頻度は 5、アクセス期間は 0 日になります。火曜日にこのファイルが 1 回読み取られた場合、深夜の時点でのファイルのアクセス期間は 0 日になり、ファイルの 2 日間の I/O 頻度は 3 に減少します。水曜日にこのファイルが 1 回読み取られた場合、深夜の時点でのファイルのアクセス期間は依然として 0 日ですが、ファイルの 2 日間の I/O 頻度は月曜日の I/O による影響がなくなるため 1 に減少します。

ファイルアクセスがあるかぎりファイルを所定の場所(上位階層のストレージデバイスなど)に保持することがファイル配置ポリシーの目的である場合、適切な再配置基準はアクセス期間になります。しかし、I/O 負荷の減少したファイルを再配置することが目的である場合、適切な再配置基準は I/O 頻度になります。

上方再配置の場合も同様です。アクセス頻度が少ないために低階層のストレージデバイスに再配置されたファイルで、アプリケーションアクティビティが更新された場合、このファイルを上位階層のデバイスに再配置することが適切な場合があります。ポリシールールに <MAX> 値の低いアクセス期間を再配置基準として指定すると、fsppadm enforce の実行間隔に 1 回でもアクセスのあったファイルが再配置されます。逆に、ポリシーに <MIN> 値付きの I/O 頻度を指定すると、目的の期間に持続的なアクティビティを行ったファイルのみが再配置されます。

Prefer 属性

<IOTEMP> と <ACCESSTEMP> 基準に Prefer 属性の値を指定することで、ファイルを再配置するときの優先度を設定できます。Prefer 属性には low または high 値を指定できます。low を指定した場合は、I/O 頻度の低いファイルが再配置されてから、I/O 頻度の高い?ファイルが再配置されます。high を指定した場合は、I/O 頻度の高いファイルが再配置されてから、I/O 頻度の低いファイルが再配置されます。Prefer 属性値は、ソリッドステートディスク(SSD)を使う場合にのみ指定することをお勧めします。

p.890 の「ソリッドステートディスクの Prefer 機構」を参照してください。

同じポリシー内に複数の RELOCATE 文がある場合、それらの <IOTEMP> と <ACCESSTEMP> 基準で異なる <PERIOD> 要素を使ってもかまいません。

次の配置ポリシーは、Prefer 基準の例です。

```
<RELOCATE>
    ...
    <WHEN>
        <IOTEMP Type="nrbytes" Prefer="high">
            <MIN Flags="gteq"> 3.4 </MIN>
            <PERIOD Units="hours"> 6 </PERIOD>
        </IOTEMP>
    </WHEN>
</RELOCATE>
```

I/O 頻度が最小値より高いファイルがいくつかある場合には、頻度の低いファイルよりも頻度の高いファイルが最初に RELOCATE 操作の対象になります。

Average I/O アクティビティ基準

Average 基準には、<PERIOD> 要素に指定する期間に発生するファイル単位アクティビティと、より長期間にわたって発生するファイルシステム全体のアクティビティの比として、I/O 頻度の値を指定できます。RELOCATE 基準の <PERIOD> 要素には、スキャン直前の時間数または日数を指定します。その期間に収集される I/O 統計が、スキャンされているファイルを処理するために使われます。I/O アクティビティは時間によって変化する可能性があるので、平均 I/O アクティビティは <PERIOD> 値よりも長い期間(デフォルトで 24 時間)にわたって収集してください。そうすることで、ファイルシステム全体の平均頻度を計算できます。Average 属性値は、ソリッドステートディスク(SSD)を使う場合にのみ指定することをお勧めします。

p.890 の「ソリッドステートディスクの Average I/O アクティビティ基準」を参照してください。

次の配置ポリシーは、Average 基準の例です。

```
<RELOCATE>
   ...
   <WHEN>
      <IOTEMP Type="nrbytes" Prefer="high" Average="*">
         <MIN Flags="gteq"> 1.5 </MIN>
         <PERIOD Units="hours"> 6 </PERIOD>
      </IOTEMP>
   </WHEN>
</RELOCATE>
```

この例では、ファイルシステム全体の全アクティブファイルのうち、過去 6 時間にわたる読み取り IOTEMP が過去 24 時間にわたる読み取り IOTEMP の 1.5 倍のファイルが再配置されます。この Average 基準は、絶対値よりも直観的で簡単に指定できます。

次の式によって、指定するファイルの読み取り IOTEMP が計算されます。

```
IOTEMP = (bytes of the file that are read in the PERIOD) /
         (PERIOD in hours * size of the file in bytes)
```

書き込みと読み書き IOTEMP も同じように計算されます。

次の式によって、平均読み取り IOTEMP が計算されます。

```
Average IOTEMP = (bytes read of all active files in the last h hours)
 /
                  (h * size of all the active files in bytes)
```

h はデフォルトでは 24 時間です。平均書き込みと読み書き IOTEMP も同じように計算されます。

前の例での値 1.5 は、ファイルシステム全体（正確にはすべてのアクティブな i ノード、つまりスキャン時でも FCL（File Change Log）ファイル内でアクティビティを収集できるもの）で過去 24 時間の平均読み取り IOTEMP の乗数です。したがって、再配置を決定するときは、これらのファイルの過去 6 時間にわたる読み取り IOTEMP アクティビティが過去 24 時間の平均アクティビティの 1.5 倍と比較されます。この方法を使うことで、<IOTEMP> または <ACCESSTEMP> 基準に特定の値を指定する必要がなくなり、代わりに Average 頻度の倍数を指定できます。指定する <PERIOD> 値よりもこの平均期間を長くすることで、ファイルアクティビティの非常に多いときと少ないときの影響をならすことができます。

Average 基準を <ACCESSTEMP> 基準と一緒に使うこともできます。使用目的と使用方法は同じです。

Average 基準を <IOTEMP> 基準と一緒に指定するか、<ACCESSTEMP> 基準と一緒に指定するかによって、平均の種類が決まります。Average 基準は、使用する基準に応じて次のいずれかの種類になります。

- 読み取り平均 IOTEMP

- 書き込み平均 IOTEMP
- 読み書き平均 IOTEMP
- 読み取り平均 ACCESSTEMP
- 書き込み平均 ACCESSTEMP
- 読み書き平均 ACCESSTEMP

デフォルトの Average は 24 時間平均頻度です。これは、FCL ファイル内で過去 24 時間に収集できるすべての頻度の合計を、それらの I/O 統計がまだ FCL ファイル内にあるときのファイル数で割ったものです。時間数は、<PLACEMENT_POLICY> 要素内に AveragePeriod 属性を指定することで上書きできます。AveragePeriod 属性値は、ソリッドステートディスク(SSD)を使う場合にのみ指定することをお勧めします。

次の例では、平均ファイルシステムアクティビティが、デフォルトの 24 時間ではなく 30 時間にわたって収集されて計算されます。

```
<PLACEMENT_POLICY Name="Policy1" Version="5.1" AveragePeriod="30">
```

RELOCATE 文の例

次の例は条件の指定がない再配置文で、RELOCATE ポリシールール文の最も単純な形式です。

```
 <RELOCATE>
   <FROM>
     <SOURCE>
       <CLASS>tier1</CLASS>
     </SOURCE>
   </FROM>
   <TO>
     <DESTINATION>
       <CLASS>tier2</CLASS>
     </DESTINATION>
   </TO>
 </RELOCATE>
```

<Check Alignment of PHs>fsppadm enforce コマンド発行時に、ルールの SELECT 文で指定したファイルが配置クラス tier1 のボリュームに存在し、配置クラス tier2 に許可される領域があれば、そのファイルは無条件にボリューム tier2 に再配置されます。この種類のルールは、たとえば、実行すると新しいファイルの作成とアクセスは行っても既存ファイルのアクセスはほとんど行わないアプリケーションに使われます。CREATE 文では、高速または高可用性だと思われる tier1 ボリュームへの作成を指定します。fsppadm enforce の各インスタンス化で、前回の実行以後に作成されたファイルを tier2 ボリュームに再配置します。

次の例はより包括的な RELOCATE 文の形式です。この文では、tier1 から tier2 ボリュームへのファイルの再配置基準としてアクセス期間を使っています。このルールは、アクティビティのないファイルを tier2 ボリュームに再配置して、tier1 ボリュームの空き領域を確保するように作られています。

```
<RELOCATE>
  <FROM>
    <SOURCE>
      <CLASS>tier1</CLASS>
    </SOURCE>
  </FROM>
  <TO>
    <DESTINATION>
      <CLASS>tier2</CLASS>
     </DESTINATION>
  </TO>
  <WHEN>
    <SIZE Units="MB">
      <MIN Flags="gt">1</MIN>
      <MAX Flags="lt">1000</MAX>
    </SIZE>
    <ACCAGE Units="days">
      <MIN Flags="gt">30</MIN>
    </ACCAGE>
  </WHEN>
</RELOCATE>
```

SELECT 文で指定され、サイズが 1 MB から 1000 MB で 30 日間アクセスのなかったファイルは、tier1 から tier2 ボリュームに再配置されます。VxFS は、ファイルシステムのディレクトリツリーをスキャンする時に見つけた対象ファイルの順番で再配置を行います。VxFS は、スケジュール設定済みファイルの再配置で tier2 ボリュームが満杯になると計算した時点で、対象ファイルの再配置スケジュールを停止します。

次の例は考えられるもう一方のルールで、ファイルの I/O 頻度をもとに tier2 から tier1 ボリュームに再配置します。このルールを使うと、アクティビティがないため tier2 に再配置されたファイルは、そのファイルに対するアプリケーションアクティビティが増加したときに、tier1 ボリュームに戻されます。アクセス期間ではなく I/O 頻度を再配置基準に使う場合、アプリケーションが実際にはほとんど使わないファイルを再配置する機会は少なくなります。このルールでは、直近の 2 日間でファイルに対する持続的なアクティビティがない限り、ファイルを再配置しません。

```
<RELOCATE>
  <FROM>
    <SOURCE>
```

```
      <CLASS>tier2</CLASS>
    </SOURCE>
  </FROM>
  <TO>
    <DESTINATION>
      <CLASS>tier1</CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <IOTEMP Type="nrbytes">
      <MIN Flags="gt">5</MIN>
      <PERIOD>2</PERIOD>
    </IOTEMP>
  </WHEN>
 </RELOCATE>
```

このルールでは、tier2 ボリュームにあるファイルで、fsppadm enforce コマンド発行直前の 2 日間に I/O 頻度が 5 より大きいものを tier1 ボリュームに再配置します。VxFS は、ファイルシステムのディレクトリツリーをスキャンする時に見つけた対象ファイルの順番で再配置を行います。tier1 ボリュームが満杯になる時点で、VxFS は対象ファイルの再配置スケジュールを停止します。

VxFS のファイル配置ポリシーで、多数の配置クラスにファイルを配置するように制御できます。次の例は、I/O 頻度の低いファイルを tier1 から tier2 ボリュームに再配置し、tier2 ボリュームが満杯の場合は tier3 ボリュームに再配置するルールです。

```
 <RELOCATE>
   <FROM>
     <SOURCE>
       <CLASS>tier1</CLASS>
     </SOURCE>
   </FROM>
   <TO>
     <DESTINATION>
       <CLASS>tier2</CLASS>
     </DESTINATION>
     <DESTINATION>
       <CLASS>tier3</CLASS>
     </DESTINATION>
   </TO>
   <WHEN>
     <IOTEMP Type="nrbytes">
       <MAX Flags="lt">4</MAX>
       <PERIOD>3</PERIOD>
```

```
    </IOTEMP>
  </WHEN>
</RELOCATE>
```

このルールでは、3 日間の I/O 頻度が 4 未満で tier1 ボリュームに存在するファイルが再配置されます。VxFS は、再配置済みのファイルで tier2 ボリュームが満杯になると計算した時点で、対象ファイルを代わりの tier3 ボリュームに再配置します。VxFS は、ファイルシステムのディレクトリツリーをスキャンする時に見つけた対象ファイルを再配置します。

RELOCATE 文の <FROM> 節はオプションです。この節がない場合、VxFS は、fsppadm enforce コマンド発行時にファイルがどのボリュームに存在するかは関係なく、ルールの SELECT 文で指定された再配置対象のファイルかどうかを評価します。次の断片的なポリシールールの例では、ファイルサイズに従ってファイルが再配置されます。fsppadm enforce コマンドが発行される時にファイルがどの場所に存在するかは関係ありません。

```
<RELOCATE>
  <TO>
    <DESTINATION>
      <CLASS>tier1</CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <SIZE Units="MB">
      <MAX Flags="lt">10</MAX>
    </SIZE>
  </WHEN>
</RELOCATE>
<RELOCATE>
  <TO>
    <DESTINATION>
      <CLASS>tier2</CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <SIZE Units="MB">
      <MIN Flags="gteq">10</MIN>
      <MAX Flags="lt">100</MAX>
    </SIZE>
  </WHEN>
</RELOCATE>
<RELOCATE>
  <TO>
    <DESTINATION>
```

```
      <CLASS>tier3</CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <SIZE Units="MB">
      <MIN Flags="gteq">100</MIN>
    </SIZE>
  </WHEN>
</RELOCATE>
```

このルールでは、10 MB 未満のファイルは tier1 ボリュームに再配置され、10 MB 以上 100 MB 以下のファイルは tier2 ボリューム、100 MB より大きいファイルは tier3 ボリュームに再配置されます。VxFS は、fsppadm enforce コマンド発行時に、DESTINATION 配置クラスのボリュームにまだ存在していないすべての対象ファイルを再配置します。

次の例では、tier2 の拡張子 dbf を持つファイルのうち、過去 30 日以内にアクセスのなかったファイルをすべて圧縮して tier4 に再配置します。

```
<SELECT Flags="Data">
  <PATTERN> *.dbf </PATTERN>
</SELECT>

<RELOCATE>
  <FROM>
    <SOURCE>
      <CLASS> tier2 </CLASS>
    </SOURCE>
  </FROM>
  <TO Flags="compress">
    <DESTINATION>
      <CLASS> tier4 </CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <ACCAGE Units="days">
      <MIN Flags="gt">30</MIN>
    </ACCAGE>
  </WHEN>
</RELOCATE>
```

次の例では、tier3 の拡張子 dbf を持つファイルのうち、過去 1 時間以内にアクセスのなかったファイルをすべて圧縮解除して tier1 に再配置します。

```
<SELECT Flags="Data">
  <PATTERN> *.dbf </PATTERN>
</SELECT>

<RELOCATE>
  <FROM>
    <SOURCE>
      <CLASS> tier3 </CLASS>
    </SOURCE>
  </FROM>
  <TO Flags="uncompress">
    <DESTINATION>
      <CLASS> tier1 </CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <ACCAGE Units="hours">
      <MIN Flags="gt">1</MIN>
    </ACCAGE>
  </WHEN>
</RELOCATE>
```

DELETE 文

ファイル配置ポリシールールの DELETE 文は RELOCATE 文と形式および機能において非常に似ています。DELETE 文には <TO> 節がないだけです。ファイル配置ポリシーに基づく削除は、再配置先を固定した再配置と同じようなものだと考えられます。

メモ: DELETE 文は注意して使ってください。

次の XML の抜粋は、一般的な DELETE 文の形式を示します。

```
<DELETE>
  <FROM>
    <SOURCE>
      <CLASS> placement_class_name </CLASS>
    </SOURCE>
    <SOURCE>
      additional_placement_class_specifications
    </SOURCE>
  </FROM>
  <WHEN> relocation_conditions </WHEN>
</DELETE>
```

DELETE 文には次の節が含まれます。

<FROM> 削除元の配置クラスのリストを含めるオプションの節です。この配置クラスのボリュームの中の指定されたファイルが <WHEN> 節の条件を満たせば削除されます。<FROM> 節に記述された配置クラスの順番に優先度はありません。ルールが適用されたファイルが、指定したいずれかの配置クラスのボリュームにある場合、そのファイルは削除されます。DELETE 文に <FROM> 節がない場合、VxFS は、ファイルがどのファイルシステムのボリュームに存在しているかに関係なく、対象ファイルを削除します。

<WHEN> ルールが適用されたファイルの削除条件を指定するオプションの節です。DELETE 文の <WHEN> 節の形式は、RELOCATE 文の <WHEN> 節の形式と同じです。DELETE 文に <WHEN> 節がない場合、ルールの SELECT 文で指定されたファイルと、<FROM> 節（記述されている場合）で指定されたファイルは無条件に削除されます。

DELETE 文の例

次は DELETE 文の使用例です。

```
<DELETE>
  <FROM>
    <SOURCE>
      <CLASS>tier3</CLASS>
    </SOURCE>
  </FROM>
</DELETE>
<DELETE>
  <FROM>
    <SOURCE>
      <CLASS>tier2</CLASS>
    </SOURCE>
  </FROM>
  <WHEN>
    <ACCAGE Units="days">
      <MIN Flags="gt">120</MIN>
    </ACCAGE>
  </WHEN>
</DELETE>
```

最初の DELETE 文では、fsppadm enforce コマンド実行時に、tier3 ボリュームに存在し、ルールの SELECT 文で指定されたファイルは無条件で削除されます。DELETE 文に <WHEN> 節がない場合は、指定されたファイルは無条件に削除されることを示します。

2 番目の DELETE 文では、fsppadm enforce コマンド実行時に、tier2 ボリュームに存在し、ルールが適用されたファイルで過去 120 日間アクセスのなかったものが削除されます。

COMPRESS 文

配置ポリシーのルールの COMPRESS 文で、マルチボリュームまたは単一ボリュームファイルシステムにインプレースファイル圧縮を指定します。配置ポリシーは選択したファイルに割り当てられた状態になり、圧縮されたエクステントの割り当ては <FROM> 節の <SOURCE> 要素で指定された階層から行われます。SmartTier は、ファイルスパンが複数の階層に渡っている場合にもファイル全体のインプレース圧縮を実行します。

メモ: SmartTier は圧縮アクティビティをスケジュールしません。Veritas InfoScale の製品を VOM (Veritas Operations Manager) と統合していない場合は、cron ジョブによるスケジュールなどの方法で圧縮アクティビティを自動化する必要があります。

次の XML の抜粋は、一般的な COMPRESS 文の形式を示します。

```
<COMPRESS>
  <FROM>
    <SOURCE>
      <CLASS> placement_class_name </CLASS>
    </SOURCE>
    <SOURCE> additional_placement_class_specifications
    </SOURCE>
  </FROM>
  <WHEN> compression_conditions </WHEN>
</COMPRESS>
```

COMPRESS 文には次の節が含まれます。

<FROM>
: 圧縮元の配置クラスのリストを含めるオプションの節です。この配置クラスのボリュームの中の指定されたファイルが <WHEN> 節の条件を満たせば圧縮されます。<FROM> 節に記述された配置クラスの順番に優先度はありません。ルールが適用されたファイルが、指定したいずれかの配置クラスのボリュームにある場合、そのファイルは圧縮対象と見なされます。

  COMPRESS 文に <FROM> 節がある場合、VxFS は、この節に指定された配置クラスのボリュームに存在するファイルのみを圧縮対象と見なします。<FROM> 節がない場合は、ファイルが存在する場所に関係なく、対象のファイルは圧縮されます。

<WHEN>　ルールが適用されたファイルの圧縮条件を指定するオプションの節です。指定した期間アクセスがないかまたは修正がないファイル、特定のサイズに達したファイル、特定の I/O 頻度またはアクセス頻度のレベルに達したファイルなどを圧縮できます。COMPRESS 文に <WHEN> 節がない場合、ルールが適用されたファイルは無条件に圧縮されます。

COMPRESS 文に <WHEN> 節を含めると、4 つの基準のいずれかまたはすべてを満たす場合にのみファイルを圧縮することができます。1 つ以上の基準を満たすファイルを圧縮するように指定できます。

<WHEN> 節では、次の基準が指定できます。

<ACCAGE>　指定した期間にファイルがアクティブにならない場合、つまり fsppadm enforce コマンドの発行から指定した期間にファイルがアクティブにならない場合にこの基準が満たされます。

<MODAGE>　指定した期間にファイルの修正がない場合、つまり fsppadm enforce コマンドの発行から指定した期間にファイルの修正がない場合にこの基準が満たされます。

<SIZE>　ファイルが、指定したサイズを超えるまたは下回る場合、または指定したサイズ範囲内に収まる場合にこの基準が満たされます。

<IOTEMP>　ファイルが、指定した I/O 頻度を超えるまたは下回る場合、または指定した I/O 頻度範囲内に収まる場合にこの基準が満たされます。ファイルの I/O 頻度とは、fsppadm enforce コマンドの発行より前の <PERIOD> 要素に指定した期間における、ファイルの I/O 負荷の測定値です。

p.879 の「I/O 頻度とアクセス頻度の計算」を参照してください。

<ACCESSTEMP>　ファイルが、指定した平均アクセス頻度を超えるまたは下回る場合、または指定したアクセス頻度範囲内に収まる場合にこの基準が満たされます。ファイルのアクセス頻度はファイルの I/O 頻度と同じですが、アクセス頻度はファイルへの転送バイト数ではなく I/O 要求数を使って計算されます。

メモ: VxFS サーバーを NFS サーバーとして使う場合、<IOTEMP> と <ACCESSTEMP> を使ってサーバー上にデータを配置する動作が、NFS キャッシュのためにあまり効果的でない場合があります。NFS クライアント側キャッシュと NFS の動作によって、NFS サーバー側で I/O が発生せずに、NFS クライアントで I/O が発生することがあります。そのような場合、サーバー側で行われる頻度測定は、配置ポリシーに指定されている I/O 動作を正しく反映しません。

サーバーを NFS サーバーとしてのみ使う場合は、頻度しきい値を適切に調整するか下げることによって、この問題を軽減できる場合があります。ただし、しきい値を調整しても望ましい効果が得られないこともあります。また、同じマウントポイントを NFS エクスポートとローカルマウントとして使う場合は、NFS キャッシュスキューのために、頻度に基づいて配置を決定する方法はあまり効果的ではありません。

次の XML の抜粋は、COMPRESS 文の一般的な <WHEN> 節の形式を示します。

```
 <WHEN>
   <ACCAGE Units="units_value">
     <MIN Flags="comparison_operator">
       min_access_age</MIN>
     <MAX Flags="comparison_operator">
       max_access_age</MAX>
   </ACCAGE>
   <MODAGE Units="units_value">
     <MIN Flags="comparison_operator">
       min_modification_age</MIN>
     <MAX Flags="comparison_operator">
       max_modification_age</MAX>
   </MODAGE>
   <SIZE " Units="units_value">
     <MIN Flags="comparison_operator">
       min_size</MIN>
     <MAX Flags="comparison_operator">
       max_size</MAX>
   </SIZE>
   <IOTEMP Type="read_write_preference"
Prefer="temperature_preference">
     <MIN Flags="comparison_operator">
       min_I/O_temperature</MIN>
     <MAX Flags="comparison_operator">
       max_I/O_temperature</MAX>
     <PERIOD Units="days_or_hours"> days_or_hours_of_interest
</PERIOD>
   </IOTEMP>
```

```
   <ACCESSTEMP Type="read_write_preference"
    Prefer="temperature_preference">
     <MIN Flags="comparison_operator">
       min_access_temperature</MIN>
     <MAX Flags="comparison_operator">
       max_access_temperature</MAX>
     <PERIOD Units="days_or_hours"> days_or_hours_of_interest
</PERIOD>
   </ACCESSTEMP>
 </WHEN>
```

アクセス期間(<ACCAGE>)要素は、ファイルが最後にアクセスされてからの時間です。VxFS は、fsppadm enforce コマンドの発行時間からファイルの最終アクセス時間(atime)を引いて、アクセス期間を算出します。<ACCAGE> 節の <MIN> と <MAX> XML 要素は、それぞれ圧縮のための最小アクセス期間と最大アクセス期間のしきい値を示します。これらの要素はオプションですが、少なくとも 1 つは含める必要があります。Units XML 属性を指定すれば、<MIN> と <MAX> 要素は次の単位で指定できます。

hours	時間
days	日。1 日は fsppadm enforce コマンドの発行より前の 24 時間として考えられます。

<MIN> と <MAX> 要素は両方とも Flags 属性を指定して操作を指示する必要があります。

<MIN> では、次の Flags 属性値を指定できます。

gt	最終アクセス時間は指定の間隔よりも大きい必要があります。
eq	最終アクセス時間は指定の間隔と等しい必要があります。
gteq	最終アクセス時間は指定の間隔よりも大きいか等しい必要があります。

<MAX> では、次の Flags 属性値を指定できます。

lt	最終アクセス時間は指定の間隔よりも小さい必要があります。
lteq	最終アクセス時間は指定の間隔よりも小さいか等しい必要があります。

<WHEN> 節に <MIN> 要素を含めると、VxFS はルールが適用されたファイルで指定した間隔よりも長い期間アクティブになっていないファイルを圧縮します。通常、このようなルールを使って、アクティブになっていないファイルを安価なストレージ階層に圧縮します。逆に、<MAX> は指定した間隔にアクセスのあったファイルを再配置します。通常この要素を使って、アクティビティを再開した非アクティブなファイルを、パフォーマンスが速

いかまたは信頼性の高いストレージに移動します。<MAX> と <MIN> 節の 2 つを含めると、VxFS はアクセス期間がこの 2 つの間に収まるファイルを圧縮します。

修正期間の圧縮基準(<MODAGE>)はアクセス期間と同じですが、ファイルの POSIX mtime 値が計算に使われます。通常、<MODAGE> 基準を指定するのは、最近修正されたファイルを今後反復的にアクセスされることを予想してパフォーマンスの速いまたは信頼性の高いストレージ階層に圧縮するためです。

ファイルサイズの圧縮基準(<SIZE>)を指定すると、fsppadm enforce コマンドが発行された時に、指定された圧縮基準の <MIN> 値よりも大きいか、または <MAX> 値よりも小さいファイルが圧縮されます。両方の基準を指定すると、VxFS はサイズがこの 2 つの間に収まるファイルを圧縮するようにスケジュールします。Units 属性を使うと、ファイルサイズのしきい値を次の単位で指定できます。

bytes	バイト
KB	キロバイト
MB	メガバイト
GB	ギガバイト

I/O 頻度圧縮基準の指定

I/O 頻度圧縮基準(<IOTEMP>)を指定すると、fsppadm enforce コマンド発行直前の指定した期間に、指定値を超えるかまたは下回る I/O 頻度のファイルが圧縮されます。ファイルの I/O 頻度とは、ファイルに対する読み取り、書き込み、つまり I/O 負荷合計の測定値で、ファイルサイズに対して正規化されます。高い I/O 頻度は、アプリケーションのアクティビティレベルが高いことを示します。逆に I/O 頻度が低いとアクティビティレベルが低いことになります。VxFS は、指定期間中のファイルへの(またはファイルからの)転送バイト数(読み取り、書き込み、またはその両方)を、fsppadm enforce コマンド発行時のファイルサイズで割って、ファイルの I/O 頻度を算出します。

p.879 の「I/O 頻度とアクセス頻度の計算」を参照してください。

他のファイル圧縮基準と同様に、<IOTEMP> に <MIN> 要素を使って下位のしきい値を指定でき、<MAX> 要素を使って上位のしきい値、両方を使って範囲を指定できます。ただし、I/O 頻度に寸法はないので単位の指定はありません。

VxFS がファイルの I/O 頻度を算出するときの期間は、fsppadm enforce コマンド発行時から、過去 <PERIOD> 要素に指定した日数または時間数までの期間です。1 日は 24 時間を意味します。デフォルトの単位は日数です。<PERIOD> 要素の Units 属性を hours に設定することで、時間数を単位として指定できます。ソリッドステートディスク(SSD)を使う場合にのみ、時間数を指定することをお勧めします。

p.891 の「ソリッドステートディスクでの SmartTier のスキャン頻度」を参照してください。

たとえば、fsppadm enforce コマンドを水曜日の午後 2 時で発行したけれども、月曜日の午後 2 時から水曜日の午後 2 時まで(2 日間)のファイル I/O アクティビティを計算する場合は、次のように <PERIOD> 要素を指定します。

```
<PERIOD> 2 </PERIOD>
```

代わりに、fsppadm enforce コマンドを実行する 3 時間前からコマンドを実行した時点までのファイル I/O アクティビティを計算する場合は、次のように <PERIOD> 要素を指定します。

```
<PERIOD Units="hours"> 3 </PERIOD>
```

FCL(File Change Log)ファイルで使われるディスク領域のため、<PERIOD> に 1 週間または 2 週間を超える期間を指定しないでください。

p.1101 の「Veritas File System ファイルの変更ログファイルについて」を参照してください。

I/O 頻度は I/O 負荷の測定値でアクセス期間よりも滑らかな値です。アクセス期間では、ファイルへのただ 1 回のアクセスでファイルの atime は現在時刻にリセットされます。これに対して、ファイルの I/O 頻度は、ファイルへのアクセスがない間は徐々に減少し、ファイルが定期的にアクセスされると徐々に増加していきます。たとえば、月曜日に 10 MB の新しいファイルが作成され、5 回完全に読み取られた場合、深夜に fsppadm enforce を実行すると、このファイルの 2 日間の I/O 頻度は 5、アクセス期間は 0 日になります。火曜日にこのファイルが 1 回読み取られた場合、深夜の時点でのファイルのアクセス期間は 0 日になり、ファイルの 2 日間の I/O 頻度は 3 に減少します。水曜日にこのファイルが 1 回読み取られた場合、深夜の時点でのファイルのアクセス期間は依然として 0 日ですが、ファイルの 2 日間の I/O 頻度は月曜日の I/O による影響がなくなるため 1 に減少します。

ファイルアクセスがあるかぎりファイルを所定の場所(上位階層のストレージデバイスなど)に保持することがファイル配置ポリシーの目的である場合、適切な圧縮基準はアクセス期間になります。しかし、I/O 負荷の減少したファイルを圧縮することが目的である場合、適切な圧縮基準は I/O 頻度になります。

上方圧縮の場合も同様です。アクセス頻度が少ないために低階層のストレージデバイスに圧縮されたファイルで、アプリケーションアクティビティが更新された場合、このファイルを上位階層のデバイスに圧縮することが適切な場合があります。ポリシールールに <MAX> 値の低いアクセス期間を圧縮基準として指定すると、fsppadm enforce の実行間隔に 1 回でもアクセスのあったファイルが圧縮されます。逆に、ポリシーに <MIN> 値付きの I/O 頻度を指定すると、目的の期間に持続的なアクティビティを行ったファイルのみが圧縮されます。

Prefer 属性

<IOTEMP> と <ACCESSTEMP> 基準に Prefer 属性の値を指定することで、ファイルを圧縮するときの優先度を設定できます。Prefer 属性には low または high 値を指定できます。low を指定した場合は、I/O 頻度の低いファイルが圧縮されてから、I/O 頻度の高いファイルが圧縮されます。high を指定した場合は、I/O 頻度の高いファイルが圧縮されてから、I/O 頻度の低いファイルが圧縮されます。Prefer 属性値は、ソリッドステートディスク(SSD)を使う場合にのみ指定することをお勧めします。

p.890 の「ソリッドステートディスクの Prefer 機構」を参照してください。

同じポリシー内に複数の COMPRESS 文がある場合、それらの <IOTEMP> と <ACCESSTEMP> 基準で異なる <PERIOD> 要素を使ってもかまいません。

次の配置ポリシーは、Prefer 基準の例です。

```
<COMPRESS>
   ...
   <WHEN>
      <IOTEMP Type="nrbytes" Prefer="high">
         <MIN Flags="gteq"> 3.4 </MIN>
         <PERIOD Units="hours"> 6 </PERIOD>
      </IOTEMP>
   </WHEN>
</COMPRESS>
```

I/O 頻度が最小値より高いファイルがいくつかある場合には、頻度の低いファイルよりも頻度の高いファイルが最初に COMPRESS 操作の対象になります。

Average I/O アクティビティ基準

Average 基準には、<PERIOD> 要素に指定する期間に発生するファイル単位アクティビティと、より長期間にわたって発生するファイルシステム全体のアクティビティの比として、I/O 頻度の値を指定できます。COMPRESS 基準の <PERIOD> 要素には、スキャン直前の時間数または日数を指定します。その期間に収集される I/O 統計が、スキャンされているファイルを処理するために使われます。I/O アクティビティは時間によって変化する可能性があるので、平均 I/O アクティビティは <PERIOD> 値よりも長い期間(デフォルトで 24 時間)にわたって収集してください。そうすることで、ファイルシステム全体の平均頻度を計算できます。Average 属性値は、ソリッドステートディスク(SSD)を使う場合にのみ指定することをお勧めします。

p.890 の「ソリッドステートディスクの Average I/O アクティビティ基準」を参照してください。

次の配置ポリシーは、Average 基準の例です。

```
<COMPRESS>
   ...
   <WHEN>
      <IOTEMP Type="nrbytes" Prefer="high" Average="*">
         <MIN Flags="gteq"> 1.5 </MIN>
         <PERIOD Units="hours"> 6 </PERIOD>
      </IOTEMP>
   </WHEN>
</COMPRESS>
```

この例では、ファイルシステム全体の全アクティブファイルのうち、過去 6 時間にわたる読み取り IOTEMP が過去 24 時間にわたる読み取り IOTEMP の 1.5 倍のファイルが圧縮されます。この Average 基準は、絶対値よりも直観的で簡単に指定できます。

次の式によって、指定するファイルの読み取り IOTEMP が計算されます。

```
IOTEMP = (bytes of the file that are read in the PERIOD) /
         (PERIOD in hours * size of the file in bytes)
```

書き込みと読み書き IOTEMP も同じように計算されます。

次の式によって、平均読み取り IOTEMP が計算されます。

```
Average IOTEMP = (bytes read of all active files in the last h hours)
 /
                  (h * size of all the active files in bytes)
```

h はデフォルトでは 24 時間です。平均書き込みと読み書き IOTEMP も同じように計算されます。

前の例での値 1.5 は、ファイルシステム全体（正確にはすべてのアクティブな i ノード、つまりスキャン時でも FCL（File Change Log）ファイル内でアクティビティを収集できるもの）で過去 24 時間の平均読み取り IOTEMP の乗数です。したがって、圧縮を決定するときは、これらのファイルの過去 6 時間にわたる読み取り IOTEMP アクティビティが過去 24 時間の平均アクティビティの 1.5 倍と比較されます。この方法を使うことで、<IOTEMP> または <ACCESSTEMP> 基準に特定の値を指定する必要がなくなり、代わりに Average 頻度の倍数を指定できます。指定する <PERIOD> 値よりもこの平均期間を長くすることで、ファイルアクティビティの非常に多いときと少ないときの影響をならすことができます。

Average 基準を <ACCESSTEMP> 基準と一緒に使うこともできます。使用目的と使用方法は同じです。

Average 基準を <IOTEMP> 基準と一緒に指定するか、<ACCESSTEMP> 基準と一緒に指定するかによって、平均の種類が決まります。Average 基準は、使用する基準に応じて次のいずれかの種類になります。

- 読み取り平均 IOTEMP

- 書き込み平均 IOTEMP
- 読み書き平均 IOTEMP
- 読み取り平均 ACCESSTEMP
- 書き込み平均 ACCESSTEMP
- 読み書き平均 ACCESSTEMP

デフォルトの Average は 24 時間平均頻度です。これは、FCL ファイル内で過去 24 時間に収集できるすべての頻度の合計を、それらの I/O 統計がまだ FCL ファイル内にあるときのファイル数で割ったものです。時間数は、<PLACEMENT_POLICY> 要素内に AveragePeriod 属性を指定することで上書きできます。AveragePeriod 属性値は、ソリッドステートディスク(SSD)を使う場合にのみ指定することをお勧めします。

次の例では、平均ファイルシステムアクティビティが、デフォルトの 24 時間ではなく 30 時間にわたって収集されて計算されます。

```
<PLACEMENT_POLICY Name="Policy1" Version="5.1" AveragePeriod="30">
```

COMPRESS 文の例

次の例では、MVS ファイルシステムの tier2 の拡張子 dbf を持つファイルのうち、30 日以上アクセスされなかったものをすべて圧縮します。

```
<SELECT Flags="Data">
  <PATTERN> *.dbf </PATTERN>
</SELECT>

<COMPRESS>
  <FROM>
    <SOURCE>
      <CLASS> tier2 </CLASS>
    </SOURCE>
  </FROM>
  <WHEN>
    <ACCAGE Units="days">
      <MIN Flags="gt">30</MIN>
    </ACCAGE>
  </WHEN>
</COMPRESS>
```

fsppadm enforce コマンド発行時に、ルールの SELECT 文で指定した配置クラス tier2 のボリュームに存在するファイルがインプレース圧縮されます。fsppadm enforce の各インスタンス化で、前回の実行以後に作成されたファイルを tier2 ボリュームに圧縮します。

次の例では、単一ボリュームの拡張子 dbf を持つファイルのうち、1 分以上アクセスされなかったものをすべて圧縮します。

```
<SELECT Flags="Data">
  <PATTERN> *.dbf </PATTERN>
</SELECT>

<COMPRESS>
  <WHEN>
    <ACCAGE Units="minutes">
      <MIN Flags="gt">1</MIN>
    </ACCAGE>
  </WHEN>
</COMPRESS>
```

単一ボリュームでは <FROM> 節は必要ありません。fsppadm enforce コマンド発行時に、ルールの SELECT 文で指定したファイルがインプレース圧縮されます。fsppadm enforce の各インスタンス化で、前回の実行以後に作成されたファイルをボリュームに圧縮します。

次の例では、tier3 のすべてのファイルを圧縮します。

```
<SELECT Flags="Data">
  <PATTERN> * </PATTERN>
</SELECT>

<COMPRESS>
  <FROM>
    <SOURCE>
      <CLASS> tier3 </CLASS>
    </SOURCE>
  </FROM>
</COMPRESS>
```

このルールは、fsppadm enforce コマンド発行時に tier3 にあるすべてのファイルをインプレース圧縮します。

UNCOMPRESS 文

配置ポリシーのルールの UNCOMPRESS 文で、マルチボリュームまたは単一ボリュームファイルシステムにインプレースファイル圧縮解除を指定します。配置ポリシーは選択したファイルに割り当てられた状態になり、圧縮解除されたエクステントの割り当ては <FROM> 節の <SOURCE> 要素で指定された階層から行われます。

ファイルが部分的に圧縮されている場合、ファイルはインプレース圧縮にのみ選択できます 圧縮された後、ファイルは次のポリシーの実施操作で、再配置される前に圧縮解除されます。

メモ: SmartTier は圧縮解除アクティビティをスケジュールしません。Veritas InfoScale の製品を VOM (Veritas Operations Manager)と統合していない場合は、cron ジョブによるスケジュールなどの方法で圧縮解除アクティビティを自動化する必要があります。

次の XML の抜粋は、一般的な UNCOMPRESS 文の形式を示します。

```
<UNCOMPRESS>
  <FROM>
    <SOURCE>
      <CLASS> placement_class_name </CLASS>
    </SOURCE>
    <SOURCE> additional_placement_class_specifications
    </SOURCE>
  </FROM>
  <WHEN> uncompression_conditions </WHEN>
</UNCOMPRESS>
```

UNCOMPRESS 文には次の節が含まれます。

<FROM> 圧縮元の配置クラスのリストを含めるオプションの節です。この配置クラスのボリュームの中の指定されたファイルが <WHEN> 節の条件を満たせば圧縮解除されます。<FROM> 節に記述された配置クラスの順番に優先度はありません。ルールが適用されたファイルが、指定したいずれかの配置クラスのボリュームにある場合、そのファイルは圧縮解除対象と見なされます。

UNCOMPRESS 文に <FROM> 節がある場合、VxFS は、この節に指定された配置クラスのボリュームに存在するファイルのみを圧縮解除対象と見なします。<FROM> 節がない場合は、ファイルが存在する場所に関係なく、対象のファイルは圧縮解除されます。

<WHEN> ルールが適用されたファイルの圧縮解除条件を指定するオプションの節です。指定した期間アクセスがないかまたは修正がないファイル、特定のサイズに達したファイル、特定の I/O 頻度またはアクセス頻度のレベルに達したファイルなどを圧縮解除できます。UNCOMPRESS 文に <WHEN> 節がない場合、ルールが適用されたファイルは無条件に圧縮解除されます。

UNCOMPRESS 文に <WHEN> 節を含めると、4 つの基準のいずれかまたはすべてを満たす場合にのみファイルを圧縮解除することができます。1 つ以上の基準を満たすファイルを圧縮解除するように指定できます。

<WHEN> 節では、次の基準が指定できます。

<ACCAGE> 指定した期間にファイルがアクティブにならない場合、つまり fsppadm enforce コマンドの発行から指定した期間にファイルがアクティブにならない場合にこの基準が満たされます。

<MODAGE> 指定した期間にファイルの修正がない場合、つまり fsppadm enforce コマンドの発行から指定した期間にファイルの修正がない場合にこの基準が満たされます。

<SIZE> ファイルが、指定したサイズを超えるまたは下回る場合、または指定したサイズ範囲内に収まる場合にこの基準が満たされます。

<IOTEMP> ファイルが、指定した I/O 頻度を超えるまたは下回る場合、または指定した I/O 頻度範囲内に収まる場合にこの基準が満たされます。ファイルの I/O 頻度とは、fsppadm enforce コマンドの発行より前の <PERIOD> 要素に指定した期間における、ファイルの I/O 負荷の測定値です。

p.879 の「I/O 頻度とアクセス頻度の計算」を参照してください。

<ACCESSTEMP> ファイルが、指定した平均アクセス頻度を超えるまたは下回る場合、または指定したアクセス頻度範囲内に収まる場合にこの基準が満たされます。ファイルのアクセス頻度はファイルの I/O 頻度と同じですが、アクセス頻度はファイルへの転送バイト数ではなく I/O 要求数を使って計算されます。

メモ: VxFS サーバーを NFS サーバーとして使う場合、<IOTEMP> と <ACCESSTEMP> を使ってサーバー上にデータを配置する動作が、NFS キャッシュのためにあまり効果的でない場合があります。NFS クライアント側キャッシュと NFS の動作によって、NFS サーバー側で I/O が発生せずに、NFS クライアントで I/O が発生することがあります。そのような場合、サーバー側で行われる頻度測定は、配置ポリシーに指定されている I/O 動作を正しく反映しません。

サーバーを NFS サーバーとしてのみ使う場合は、頻度しきい値を適切に調整するか下げることによって、この問題を軽減できる場合があります。ただし、しきい値を調整しても望ましい効果が得られないこともあります。また、同じマウントポイントを NFS エクスポートとローカルマウントとして使う場合は、NFS キャッシュスキューのために、頻度に基づいて配置を決定する方法はあまり効果的ではありません。

次の XML の抜粋は、UNCOMPRESS 文の一般的な <WHEN> 節の形式を示します。

```
<WHEN>
  <ACCAGE Units="units_value">
    <MIN Flags="comparison_operator">
      min_access_age</MIN>
    <MAX Flags="comparison_operator">
      max_access_age</MAX>
```

```
   </ACCAGE>
   <MODAGE Units="units_value">
     <MIN Flags="comparison_operator">
       min_modification_age</MIN>
     <MAX Flags="comparison_operator">
       max_modification_age</MAX>
   </MODAGE>
   <SIZE " Units="units_value">
     <MIN Flags="comparison_operator">
       min_size</MIN>
     <MAX Flags="comparison_operator">
       max_size</MAX>
   </SIZE>
   <IOTEMP Type="read_write_preference"
Prefer="temperature_preference">
     <MIN Flags="comparison_operator">
       min_I/O_temperature</MIN>
     <MAX Flags="comparison_operator">
       max_I/O_temperature</MAX>
     <PERIOD Units="days_or_hours"> days_or_hours_of_interest
</PERIOD>
   </IOTEMP>
   <ACCESSTEMP Type="read_write_preference"
    Prefer="temperature_preference">
     <MIN Flags="comparison_operator">
       min_access_temperature</MIN>
     <MAX Flags="comparison_operator">
       max_access_temperature</MAX>
     <PERIOD Units="days_or_hours"> days_or_hours_of_interest
</PERIOD>
   </ACCESSTEMP>
 </WHEN>
```

アクセス期間（<ACCAGE>）要素は、ファイルが最後にアクセスされてからの時間です。VxFS は、fsppadm enforce コマンドの発行時間からファイルの最終アクセス時間（atime）を引いて、アクセス期間を算出します。<ACCAGE> 節の <MIN> と <MAX> XML 要素は、それぞれ圧縮解除のための最小アクセス期間と最大アクセス期間のしきい値を示します。これらの要素はオプションですが、少なくとも 1 つは含める必要があります。Units XML 属性を指定すれば、<MIN> と <MAX> 要素は次の単位で指定できます。

```
hours           時間
```

days	日。1 日は fsppadm enforce コマンドの発行より前の 24 時間として考えられます。

<MIN> と <MAX> 要素は両方とも Flags 属性を指定して操作を指示する必要があります。

<MIN> では、次の Flags 属性値を指定できます。

gt	最終アクセス時間は指定の間隔よりも大きい必要があります。
eq	最終アクセス時間は指定の間隔と等しい必要があります。
gteq	最終アクセス時間は指定の間隔よりも大きいか等しい必要があります。

<MAX> では、次の Flags 属性値を指定できます。

lt	最終アクセス時間は指定の間隔よりも小さい必要があります。
lteq	最終アクセス時間は指定の間隔よりも小さいか等しい必要があります。

<WHEN> 節に <MIN> 要素を含めると、VxFS はルールが適用されたファイルで指定した間隔よりも長い期間アクティブになっていないファイルを圧縮解除します。通常、このようなルールを使って、アクティブになっていないファイルを安価なストレージ階層に圧縮解除します。逆に、<MAX> は指定した間隔にアクセスのあったファイルを圧縮解除します。通常この要素を使って、アクティビティを再開した非アクティブなファイルを、パフォーマンスが速いかまたは信頼性の高いストレージに移動します。<MIN> と <MAX> 節の 2 つを含めると、VxFS はアクセス期間がこの 2 つの間に収まるファイルを圧縮解除します。

修正期間の圧縮解除基準(<MODAGE>)はアクセス期間と同じですが、ファイルの POSIX mtime 値が計算に使われます。通常、<MODAGE> 基準を指定するのは、最近修正されたファイルを今後反復的にアクセスされることを予想してパフォーマンスの速いまたは信頼性の高いストレージ階層に圧縮解除するためです。

ファイルサイズの圧縮解除基準(<SIZE>)を指定すると、fsppadm enforce コマンドが発行された時に、指定された圧縮解除基準の <MIN> 値よりも大きいか、または <MAX> 値よりも小さいファイルが圧縮解除されます。両方の基準を指定すると、VxFS はサイズがこの 2 つの間に収まるファイルを圧縮解除するようにスケジュールします。Units 属性を使うと、ファイルサイズのしきい値を次の単位で指定できます。

bytes	バイト
KB	キロバイト
MB	メガバイト
GB	ギガバイト

I/O 頻度圧縮解除基準の指定

I/O 頻度圧縮解除基準(<IOTEMP>)を指定すると、fsppadm enforce コマンド発行直前の指定した期間に、指定値を超えるかまたは下回る I/O 頻度のファイルが圧縮解除されます。ファイルの I/O 頻度とは、ファイルに対する読み取り、書き込み、つまり I/O 負荷合計の測定値で、ファイルサイズに対して正規化されます。高い I/O 頻度は、アプリケーションのアクティビティレベルが高いことを示します。逆に I/O 頻度が低いとアクティビティレベルが低いことになります。VxFS は、指定期間中のファイルへの(またはファイルからの)転送バイト数(読み取り、書き込み、またはその両方)を、fsppadm enforce コマンド発行時のファイルサイズで割って、ファイルの I/O 頻度を算出します。

p.879 の「I/O 頻度とアクセス頻度の計算」を参照してください。

他のファイル圧縮解除基準と同様に、<IOTEMP> に <MIN> 要素を使って下位のしきい値を指定でき、<MAX> 要素を使って上位のしきい値、両方を使って範囲を指定できます。ただし、I/O 頻度に寸法はないので単位の指定はありません。

VxFS がファイルの I/O 頻度を算出するときの期間は、fsppadm enforce コマンド発行時から、過去 <PERIOD> 要素に指定した日数または時間数までの期間です。1 日は 24 時間を意味します。デフォルトの単位は日数です。<PERIOD> 要素の Units 属性を hours に設定することで、時間数を単位として指定できます。ソリッドステートディスク(SSD)を使う場合にのみ、時間数を指定することをお勧めします。

p.891 の「ソリッドステートディスクでの SmartTier のスキャン頻度」を参照してください。

たとえば、fsppadm enforce コマンドを水曜日の午後 2 時で発行したけれども、月曜日の午後 2 時から水曜日の午後 2 時まで(2 日間)のファイル I/O アクティビティを計算する場合は、次のように <PERIOD> 要素を指定します。

```
<PERIOD> 2 </PERIOD>
```

代わりに、fsppadm enforce コマンドを実行する 3 時間前からコマンドを実行した時点までのファイル I/O アクティビティを計算する場合は、次のように <PERIOD> 要素を指定します。

```
<PERIOD Units="hours"> 3 </PERIOD>
```

FCL(File Change Log)ファイルで使われるディスク領域のため、<PERIOD> に 1 週間または 2 週間を超える期間を指定しないでください。

p.1101 の「Veritas File System ファイルの変更ログファイルについて」を参照してください。

I/O 頻度は I/O 負荷の測定値でアクセス期間よりも滑らかな値です。アクセス期間では、ファイルへのただ 1 回のアクセスでファイルの atime は現在時刻にリセットされます。これに対して、ファイルの I/O 頻度は、ファイルへのアクセスがない間は徐々に減少し、ファイルが定期的にアクセスされると徐々に増加していきます。たとえば、月曜日に 10 MB の新しいファイルが作成され、5 回完全に読み取られた場合、深夜に fsppadm enforce

を実行すると、このファイルの 2 日間の I/O 頻度は 5、アクセス期間は 0 日になります。火曜日にこのファイルが 1 回読み取られた場合、深夜の時点でのファイルのアクセス期間は 0 日になり、ファイルの 2 日間の I/O 頻度は 3 に減少します。水曜日にこのファイルが 1 回読み取られた場合、深夜の時点でのファイルのアクセス期間は依然として 0 日ですが、ファイルの 2 日間の I/O 頻度は月曜日の I/O による影響がなくなるため 1 に減少します。

ファイルアクセスがあるかぎりファイルを所定の場所（上位階層のストレージデバイスなど）に保持することがファイル配置ポリシーの目的である場合、適切な圧縮解除基準はアクセス期間になります。しかし、I/O 負荷の減少したファイルを圧縮解除することが目的である場合、適切な圧縮解除基準は I/O 頻度になります。

上方圧縮解除の場合も同様です。アクセス頻度が少ないために低階層のストレージデバイスに圧縮解除されたファイルで、アプリケーションアクティビティが更新された場合、このファイルを上位階層のデバイスに圧縮解除することが適切な場合があります。ポリシールールに <MAX> 値の低いアクセス期間を圧縮解除基準として指定すると、fsppadm enforce の実行間隔に 1 回でもアクセスのあったファイルが圧縮解除されます。逆に、ポリシーに <MIN> 値付きの I/O 頻度を指定すると、目的の期間に持続的なアクティビティを行ったファイルのみが圧縮解除されます。

Prefer 属性

<IOTEMP> と <ACCESSTEMP> 基準に Prefer 属性の値を指定することで、ファイルを圧縮解除するときの優先度を設定できます。Prefer 属性には low または high 値を指定できます。low を指定した場合は、I/O 頻度の低いファイルが圧縮解除されてから、I/O 頻度の高いファイルが圧縮解除されます。high を指定した場合は、I/O 頻度の高いファイルが圧縮解除されてから、I/O 頻度の低いファイルが圧縮解除されます。Prefer 属性値は、ソリッドステートディスク（SSD）を使う場合にのみ指定することをお勧めします。

p.890 の「ソリッドステートディスクの Prefer 機構」を参照してください。

同じポリシー内に複数の UNCOMPRESS 文がある場合、それらの <IOTEMP> と <ACCESSTEMP> 基準で異なる <PERIOD> 要素を使ってもかまいません。

次の配置ポリシーは、Prefer 基準の例です。

```
<UNCOMPRESS>
   ...
   <WHEN>
      <IOTEMP Type="nrbytes" Prefer="high">
         <MIN Flags="gteq"> 3.4 </MIN>
         <PERIOD Units="hours"> 6 </PERIOD>
      </IOTEMP>
   </WHEN>
</UNCOMPRESS>
```

I/O 頻度が最小値より高いファイルがいくつかある場合には、頻度の低いファイルよりも頻度の高いファイルが最初に UNCOMPRESS 操作の対象になります。

Average I/O アクティビティ基準

Average 基準には、<PERIOD> 要素に指定する期間に発生するファイル単位アクティビティと、より長期間にわたって発生するファイルシステム全体のアクティビティの比として、I/O 頻度の値を指定できます。UNCOMPRESS 基準の <PERIOD> 要素には、スキャン直前の時間数または日数を指定します。その期間に収集される I/O 統計が、スキャンされているファイルを処理するために使われます。I/O アクティビティは時間によって変化する可能性があるので、平均 I/O アクティビティは <PERIOD> 値よりも長い期間（デフォルトで 24 時間）にわたって収集してください。そうすることで、ファイルシステム全体の平均頻度を計算できます。Average 属性値は、ソリッドステートディスク（SSD）を使う場合にのみ指定することをお勧めします。

p.890 の「ソリッドステートディスクの Average I/O アクティビティ基準」を参照してください。

次の配置ポリシーは、Average 基準の例です。

```
<UNCOMPRESS>
   ...
   <WHEN>
      <IOTEMP Type="nrbytes" Prefer="high" Average="*">
         <MIN Flags="gteq"> 1.5 </MIN>
         <PERIOD Units="hours"> 6 </PERIOD>
      </IOTEMP>
   </WHEN>
</UNCOMPRESS>
```

この例では、ファイルシステム全体の全アクティブファイルのうち、過去 6 時間にわたる読み取り IOTEMP が過去 24 時間にわたる読み取り IOTEMP の 1.5 倍のファイルが圧縮解除されます。この Average 基準は、絶対値よりも直観的で簡単に指定できます。

次の式によって、指定するファイルの読み取り IOTEMP が計算されます。

```
IOTEMP = (bytes of the file that are read in the PERIOD) /
         (PERIOD in hours * size of the file in bytes)
```

書き込みと読み書き IOTEMP も同じように計算されます。

次の式によって、平均読み取り IOTEMP が計算されます。

```
Average IOTEMP = (bytes read of all active files in the last h hours)
 /
                  (h * size of all the active files in bytes)
```

h はデフォルトでは 24 時間です。平均書き込みと読み書き IOTEMP も同じように計算されます。

前の例での値 1.5 は、ファイルシステム全体（正確にはすべてのアクティブな i ノード、つまりスキャン時でも FCL（File Change Log）ファイル内でアクティビティを収集できるもの）で過去 24 時間の平均読み取り IOTEMP の乗数です。したがって、圧縮解除を決定するときは、これらのファイルの過去 6 時間にわたる読み取り IOTEMP アクティビティが過去 24 時間の平均アクティビティの 1.5 倍と比較されます。この方法を使うことで、<IOTEMP> または <ACCESSTEMP> 基準に特定の値を指定する必要がなくなり、代わりに Average 頻度の倍数を指定できます。指定する <PERIOD> 値よりもこの平均期間を長くすることで、ファイルアクティビティの非常に多いときと少ないときの影響をならすことができます。

Average 基準を <ACCESSTEMP> 基準と一緒に使うこともできます。使用目的と使用方法は同じです。

Average 基準を <IOTEMP> 基準と一緒に指定するか、<ACCESSTEMP> 基準と一緒に指定するかによって、平均の種類が決まります。Average 基準は、使用する基準に応じて次のいずれかの種類になります。

- 読み取り平均 IOTEMP
- 書き込み平均 IOTEMP
- 読み書き平均 IOTEMP
- 読み取り平均 ACCESSTEMP
- 書き込み平均 ACCESSTEMP
- 読み書き平均 ACCESSTEMP

デフォルトの Average は 24 時間平均頻度です。これは、FCL ファイル内で過去 24 時間に収集できるすべての頻度の合計を、それらの I/O 統計がまだ FCL ファイル内にあるときのファイル数で割ったものです。時間数は、<PLACEMENT_POLICY> 要素内に AveragePeriod 属性を指定することで上書きできます。AveragePeriod 属性値は、ソリッドステートディスク（SSD）を使う場合にのみ指定することをお勧めします。

次の例では、平均ファイルシステムアクティビティが、デフォルトの 24 時間ではなく 30 時間にわたって収集されて計算されます。

```
<PLACEMENT_POLICY Name="Policy1" Version="5.1" AveragePeriod="30">
```

UNCOMPRESS 文の例

次の例では、MVS ファイルシステムの tier3 の拡張子 dbf を持つファイルのうち、60 日以上アクセスされなかったものをすべて圧縮解除します。

```
<SELECT Flags="Data">
  <PATTERN> *.dbf </PATTERN>
```

```
</SELECT>

<UNCOMPRESS>
  <FROM>
    <SOURCE>
      <CLASS> tier3 </CLASS>
    </SOURCE>
  </FROM>
  <WHEN>
    <ACCAGE Units="minutes">
      <MIN Flags="gt">60</MIN>
    </ACCAGE>
  </WHEN>
</UNCOMPRESS>
```

次の例では、単一ボリュームの拡張子 dbf を持つファイルのうち、1 分以上アクセスされなかったものをすべて圧縮解除します。

```
<SELECT Flags="Data">
  <PATTERN> *.dbf </PATTERN>
</SELECT>

<UNCOMPRESS>
  <WHEN>
    <ACCAGE Units="minutes">
      <MIN Flags="gt">1</MIN>
    </ACCAGE>
  </WHEN>
</UNCOMPRESS>
```

I/O 頻度とアクセス頻度の計算

VxFS SmartTier の重要なアプリケーションには、非アクティブファイルの安価なストレージへの自動再配置があります。ファイルシステムのスキャンでは、<ACCAGE> 要素に指定された期間にアクセスのなかったファイルが、下位階層のストレージに再配置されるようにスケジュールされます。ただし、最後のアクセスからの時間だけでは、負荷ベースの再配置基準としては不十分です。

その理由は次のとおりです。

- アクセス期間はバイナリ測定値です。ファイルの最後のアクセスからの時間は、ファイルのメタデータの POSIX atime から fsppadm enforce コマンドの発行時間を引いて求められます。ファイルが fsppadm enforce コマンド発行の前の日にオープンされた場合、そのファイルの最後のアクセスからの時間は、たとえその前の 1 カ月間

アクティブにならなくても、1 日です。ポリシールールの目的が、非アクティブなファイルを下位階層のボリュームに再配置することであれば、<ACCAGE> 値に定義された期間に偶然にアクセスされたファイルには不適切な処理を行ってしまいます。

- アクセス期間は重要なアクティビティの続行を示すインジケータとしては不適切です。非アクティブなファイルを下位階層ボリュームに再配置するための基準として、最後のアクセスからの時間である ACCAGE を使うと、実行する必要がある再配置をスケジュールできない場合があります。ただし、少なくともこの方法を使うと再配置アクティビティが必要よりも少なくなります。以前にアクティブにならなかったファイルをアクティビティが再開したことで再配置するような基準に、ACCAGE を使うのはさらに不適切です。この方法は正当な理由のない再配置アクティビティをスケジュールするようなものだからです。ポリシールールの目的が、最近 I/O 負荷のあったファイルを、パフォーマンスが速く、耐障害性の優れていると思われるストレージに再配置することであれば、ACCAGE はフィルタとしては粗すぎます。たとえば、ルールに、過去 3 日間にアクセスされた tier2 ボリュームのファイルを tier1 ボリュームに再配置するように指定されている場合、1 人のユーザーに参照されたファイルと実際にアプリケーションで何度も使われたファイルは区別されません。

SmartTier は、このような不足を克服するため I/O 頻度とアクセス頻度の概念を実装します。ファイルの I/O 頻度は、指定期間中のファイルへの（またはファイルからの）転送バイト数をファイルサイズで割った値です。たとえば、fsppadm enforce 操作時に、1 MB のストレージを占有するファイルがあり、そのデータに対して過去 3 日間に 15 回完全な読み取りまたは書き込みが行われた場合、VxFS はそのファイルの 3 日間の平均 I/O 頻度を 5 と計算します（15 MB I/O ÷ 1 MB ファイルサイズ ÷ 3 日）。

同様に、ファイルの平均アクセス頻度とは、指定した期間数 x 24 時間におけるファイルに対する読み取りまたは書き込みの要求数を、指定した期間数で割った値です。I/O 頻度とは異なり、アクセス頻度はファイルサイズに関係しません。2 日間で 20 回の I/O 要求があった大容量ファイルと 2 日間で 20 回のアクセスがあった小容量ファイルの平均アクセス頻度は同じです。

ファイルシステムのアクティブな配置ポリシーに <IOTEMP> または <ACCESSTEMP> 節が記述されている場合、VxFS はポリシーの実施を開始し、ファイルシステムの FCL ファイルの情報を使って、ファイルシステムの全ファイルに対する平均 I/O 負荷を計算します。計算の対象期間は、ポリシーに指定した最長の <PERIOD> です。これよりも短い期間の指定は無視されます。VxFS はこれらの計算結果を使って、I/O 頻度ベースの再配置または削除の対象ファイルを決定します。

p.1101 の「Veritas File System ファイルの変更ログファイルについて」を参照してください。

メモ: FCL がオフになっている場合は、I/O 頻度ベースの再配置は不正確になります。FCL がオフになっている場合は、fsppadm enforce コマンドを呼び出す時に、警告が表示されます。

FCL（File Change Log）は、名前が意味するとおり、VxFS ファイルシステム内でのファイルに対する変更情報を記録します。作成、削除、拡張を記録する他に、FCL はファイル別に I/O 負荷（読み取りと書き込みのバイト数）の累積量を定期的に取得します。ファイルの I/O 負荷は、ファイルが開かれるまたは閉じられるたびに FCL に記録されます。さらに、定期的な間隔で長い期間開かれたままになっているファイルの情報が取得されます。

ファイルシステムのアクティブなファイル配置ポリシーに <IOTEMP> 節がある場合、fsppadm enforce コマンドの実行で、FCL のスキャンが開始され、ポリシーで指定された所定の期間の I/O 負荷情報が抽出されます。所定の期間とは、fsppadm enforce コマンドの発行時間と、その時間からアクティブなポリシーの <PERIOD> 要素に指定された最長の間隔値を引いた時間との間です。

最長間隔の期間に I/O 負荷のあったファイルに対し、VxFS は読み取り処理、書き込み処理、合計データ転送（両方の合計）処理の量を概算で求めます。これは、ファイルに保持された最も新しい FCL レコードの I/O レベルから、最も古いものを引いた値です。次に、各ファイルの I/O 負荷を Tscan 時のファイルサイズで割って、各ファイルの I/O 頻度を計算します。ファイルサイズで割ることは、大容量ファイルの再配置は小容量ファイルの再配置よりも多くの I/O リソースを消費することを勘案しています。このアルゴリズムを使う場合、大容量ファイルが所定の I/O 頻度に達するにはより多くのアクティビティが必要となるため、再配置のリソースコストが正当化されます。

この計算結果はいくつかの方法による概算ですが、容易に計算できます。またさらに重要なことは、最近の I/O 負荷の偏たりのない相対的な算出値であるということで、これを基に適切な再配置を決定できます。

ファイルの再配置と削除は、読み取り、書き込み、合計 I/O 負荷のいずれかを基に決定できます。

次の XML の抜粋は、ポリシールールに IOTEMP を記述して、アクティビティが少ないファイルを tier1 から tier2 ボリュームに再配置する例です。

```
<RELOCATE>
  <FROM>
    <SOURCE>
      <CLASS>tier1</CLASS>
    </SOURCE>
  </FROM>
  <TO>
    <DESTINATION>
      <CLASS>tier2</CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <IOTEMP Type="nrwbytes}">
      <MAX Flags="lt">3</MAX>
```

```
      <PERIOD Units="days">4</PERIOD>
    </IOTEMP>
  </WHEN>
 </RELOCATE>
```

この例では、ルールが適用されたファイルで 4 日間の I/O 頻度が 3 未満のものは tier1 から tier2 ボリュームに再配置されます。Type="nrwbytes}" XML 属性を指定すると、合計データ転送処理(読み取りと書き込みの合計バイト数)が計算に使われます。たとえば、fsppadm enforce スキャン直前の 4 日間に、150 MB 未満のデータ転送を行った 50 MB のファイルは再配置対象となります。VxFS は、所定の期間にアクティビティのなかったファイルは I/O 頻度がゼロであるとみなします。VxFS は、ファイルシステムのディレクトリツリーをスキャンする時に見つけた対象ファイルの順番で再配置を行います。

I/O 頻度またはアクセス頻度を使うと、POSIX atime または mtime などのアクティビティのバイナリ指標を使うよりも、所定の期間に偶然にアクセスされただけのファイルを再配置しない機会を少なくすることができます。ファイルへの(またはファイルからの)転送バイト数が少量しかない大容量ファイルの I/O 頻度は低い値です。したがってこのようなファイルは、直近にアクティビティがあったとしても、tier2 ボリュームへの再配置対象となります。

ただし、ファイル再配置基準としての I/O 頻度またはアクセス頻度が高い値になれば、上方再配置の対象となります。つまり、以前にアクティブになっていないか低頻度のためにストレージ階層の下位に再配置されていたファイルは、そのファイルに対する I/O 負荷レベルの上昇が検出されると、ストレージ階層の上位に再配置されます。

次の XML の抜粋は、アクティビティレベルが上昇したファイルを tier2 から tier1 ボリュームに再配置する例です。

```
 <RELOCATE>
   <FROM>
     <SOURCE>
       <CLASS>tier2</CLASS>
     </SOURCE>
   </FROM>
   <TO>
     <DESTINATION>
       <CLASS>tier1</CLASS>
     </DESTINATION>
   </TO>
   <WHEN>
     <IOTEMP Type="nrbytes">
       <MAX Flags="gt">5</MAX>
       <PERIOD Units="days">2</PERIOD>
     </IOTEMP>
```

```
   </WHEN>
 </RELOCATE>
```

<RELOCATE> 文には、tier2 ボリュームのファイルで、2 日間の読み取りバイト数を使って計算された I/O 頻度が 5 より大きなものは、tier1 ボリュームに再配置されるように指定されています。所定の期間におけるこのファイルへの書き込みバイト数は、この計算では使われません。

アクティビティのバイナリ指標ではなく I/O 頻度をファイル再配置基準として使うことで、管理者は細かいレベルで自動ファイル再配置を制御でき、これを使ってポリシーをアプリケーションの必要条件に合わせることができます。たとえば、上方再配置文の <PERIOD> 要素に大きな値を指定すれば、ファイルに対する持続的な I/O 負荷がないかぎり、ファイルが再配置されることはありません。代わりに、高い頻度と短い期間を指定すれば、短期間に I/O 負荷の集中したファイルが再配置されるようになります。

I/O 頻度とアクセス頻度は、i ノードでインデックス付けされた一時テーブル作成のために sqlite3 データベースを活用します。この一時テーブルを使って、I/O 頻度とアクセス頻度を基にしたファイルのフィルタ処理を行います。一時テーブルは、マウントポイントの lost+found ディレクトリにあるデータベースファイル .__fsppadm_fcliotemp.db に格納されています。

ファイル配置ポリシールール文の複数基準

場合によって、ファイル配置ポリシールール文に、操作に影響を与える複数の節が含まれていることがあります。一般的に、ルール文に任意の種類の複数の節がある場合、その文を有効にするには、すべての節を満たす必要があります。複数の節は、次の 4 つのケースで使われます。

SELECT 文の節での複数ファイル選択基準

1 つの SELECT 文内のすべての選択基準節が同じ種類であれば、1 つの選択リストとして扱われます。特定の種類の基準を 1 つ満たすだけで、ファイルは指定されます。

次の例では、ファイルが db/datafiles、db/indexes、db/logs ディレクトリ（すべてファイルシステムのマウントポイントへの相対パス）のいずれかにあれば選択されます。

```
 <SELECT>
   <DIRECTORY Flags="nonrecursive">db/datafiles</DIRECTORY>
   <DIRECTORY Flags="nonrecursive">db/indexes</DIRECTORY>
   <DIRECTORY Flags="nonrecursive">db/logs</DIRECTORY>
 </SELECT>
```

種類の異なる選択基準節の取り扱いは、この例とは正反対です。SELECT 文に複数種類のファイル選択基準がある場合、ルールの動作文を適用するには、ファイルは各種類の基準を 1 つずつ満たす必要があります。

次の例では、ファイルが動作対象として指定されるには、ファイルが db/datafiles、db/indexes、db/logs のいずれかにあり、さらに所有者が DBA_Manager、MFG_DBA、HR_DBA のいずれかである必要があります。

```
<SELECT>
  <DIRECTORY Flags="nonrecursive">db/datafiles</DIRECTORY>
  <DIRECTORY Flags="nonrecursive">db/indexes</DIRECTORY>
  <DIRECTORY Flags="nonrecursive">db/logs</DIRECTORY>
  <USER>DBA_Manager</USER>
  <USER>MFG_DBA</USER>
  <USER>HR_DBA</USER>
</SELECT>
```

ルールに複数の SELECT 文がある場合、ファイルはその中の 1 つを満たすだけで動作対象として選択されます。このプロパティを使ってファイル選択の代替条件を指定できます。

次の例では、ファイルが db/datafiles、db/indexes、db/logs のいずれかにあるか、または所有者が DBA_Manager、MFG_DBA、HR_DBA のいずれかであれば、そのファイルは動作対象として指定されます。

```
<SELECT>
  <DIRECTORY Flags="nonrecursive">db/datafiles</DIRECTORY>
  <DIRECTORY Flags="nonrecursive">db/indexes</DIRECTORY>
  <DIRECTORY Flags="nonrecursive">db/logs</DIRECTORY>
</SELECT>
<SELECT>
  <USER>DBA_Manager</USER>
  <USER>MFG_DBA</USER>
  <USER>HR_DBA</USER>
</SELECT>
```

CREATE 文の <ON> 節と RELOCATE 文の <TO> 節での複数配置クラス

CREATE 文の <ON> 節も RELOCATE 文の <TO> 節も、複数の <DESTINATION> XML 要素を使って優先度順に配置クラスリストを指定することができます。VxFS は、ファイルの作成用または再配置用のいずれかのリストで、1 番目に記述された配置クラスのボリュームを可能であれば使います。1 番目に記述されたクラスのボリュームに十分な空き領域がないか、またはファイルシステムのボリュームセットにその配置クラスのボリュームがない場合、VxFS は 2 番目に記述されたクラスのボリュームを可能であれば使います。2 番目に記述されたクラスに使用可能なボリュームがない場合、3 番目に記述されたボリュームを可能であれば使います。以降同様にしてボリュームを決めます。

次は、CREATE 文の <ON> 節に 3 つの配置クラスを指定した例です。

```
<CREATE>
  <ON>
    <DESTINATION>
      <CLASS>tier1</CLASS>
    </DESTINATION>
    <DESTINATION>
      <CLASS>tier2</CLASS>
    </DESTINATION>
    <DESTINATION>
      <CLASS>tier3</CLASS>
    </DESTINATION>
  </ON>
</CREATE>
```

この例では、VxFS は、ルールの SELECT 文で指定され新しく作成されたファイル用として、tier1 ボリュームに領域を割り当てます(領域が利用可能な場合)。tier1 ボリュームに十分な空き領域がない場合、VxFS は tier2 ボリュームへの領域の割り当てを試みます。tier2 ボリュームに十分な空き領域がない場合、VxFS は tier3 ボリュームに割り当てを試みます。指定した 3 つのボリュームに領域を十分に割り当てられるボリュームがない場合、ファイルシステムのボリュームセットの他の配置クラスに十分な領域のあるボリュームがあっても、割り当ては失敗し、ENOSPC エラーになります。

RELOCATE 文の <TO> 節も同様に動作します。VxFS は、対象のファイルを 1 番目に指定された配置クラスのボリュームに可能であれば再配置します。不可能であれば 2 番目のボリューム。以降同様にしてボリュームを決めます。指定したすべてのクラスが満杯になっている場合などで再配置先の基準を満たすことができない場合、対象のファイルは再配置されません。ただしこの場合にエラーは出力されません。

RELOCATE と DELETE 文の <FROM> 節での複数配置クラス

RELOCATE 文と DELETE 文の <FROM> 節には、再配置元または削除元の配置クラスを複数含めることができます。ただし、<ON> や <TO> 節とは異なり、<FROM> 節に順序や優先度の意味は含まれません。<FROM> 節に指定したいずれかの配置クラスのボリュームに対象ファイルが存在する場合、そのファイルは再配置または削除されます。<FROM> 節のクラスリスト上の配置クラスの位置は関係ありません。

RELOCATE と DELETE 文の <WHEN> 節での複数条件

RELOCATE 文と DELETE 文の <WHEN> 節には、再配置基準を複数含めることができます。<ACCAGE>、<MODAGE>、<SIZE>、<IOTEMP> のいずれかまたはすべてを指定できます。複数の条件を指定する場合、再配置や削除の対象として選択されるファイルは、すべての条件を満たす必要があります。

次の例では、再配置または削除の対象として選択されるファイルは、アクティビティ(つまりアクセス)が 31 日以上なく、さらに 100 MB よりも大きい必要があります。

```
<WHEN>
  <ACCAGE Units="days">
    <MIN Flags="gt">30</MIN>
  </ACCAGE>
  <SIZE Units="MB">
    <MIN Flags="gt">100</MIN>
  </SIZE>
</WHEN>
```

ファイルが、複数の再配置または削除基準のうちの 1 つを満たす場合、指定した 1 つのファイルセットに対して再配置または削除するようなルールは記述できません。

ファイル配置ポリシールールと文の順序

4 種類のファイル配置ポリシー文書のいずれかを作成するには、SmartTier の GUI が使えます。代わりに、テキストエディタまたは XML エディタを使って、XML ポリシー文書を直接作成することもできます。GUI を使うとポリシールール文は目的の操作が実行されるように正しい順序に配置されます。テキストエディタを使う場合、ユーザーは、ポリシールールとその中のポリシー文を目的の結果が得られるように順序付ける必要があります。

配置ポリシーを構成するルールは任意の順序で指定できますが、ファイルの割り当てと `fsppadm enforce` 再配置スキャンのときにファイルを評価するルールは、最初にファイルが指定された `SELECT` 文のあるルールのみです。したがって、特別クラスのファイル用のルールを一般的に適用する操作より優先させたい場合は、そのルールをファイル配置ポリシー文書の中で一般ルールの前に指定する必要があります。

次の XML の抜粋は、意図した結果が得られない可能性のある誤ったルール配置の例を示します。

```
<?xml version="1.0"?>
<!DOCTYPE FILE_PLACEMENT_POLICY SYSTEM "placement.dtd">
<FILE_PLACEMENT_POLICY Version="5.0">
  <RULE Name="GeneralRule">
    <SELECT>
      <PATTERN>*</PATTERN>
    </SELECT>
    <CREATE>
      <ON>
        <DESTINATION>
          <CLASS>tier2</CLASS>
        </DESTINATION>
```

```
      </ON>
    </CREATE>
    other_statements
  </RULE>
  <RULE Name="DatabaseRule">
    <SELECT>
      <PATTERN>*.db</PATTERN>
    </SELECT>
    <CREATE>
      <ON>
        <DESTINATION>
          <CLASS>tier1</CLASS>
        </DESTINATION>
      </ON>
    </CREATE>
    other_statements
  </RULE>
</FILE_PLACEMENT_POLICY>
```

GeneralRule は、ファイルシステムに作成するすべてのファイル（<PATTERN>*</PATTERN> による指定）を tier2 ボリュームに作成するように指定されています。DatabaseRule は、名前に .db 拡張子が含まれるファイルは、tier1 ボリュームに作成するように指定されています。このファイルシステムに作成するすべてのファイルは、名前パターンが *.db のファイルも含め、GeneralRule が適用されます。したがって、DatabaseRule ルールはいかなるファイルにも適用されません。この誤りは、2 つのルールの順序を入れ替えることで矯正できます。DatabaseRule ルールがポリシー文書の最初に指定されていれば、VxFS は最初にそのルールを見つけ、名前パターンが *.db ファイルの新しい配置先を決定するので、*.db ファイルの領域は tier1 ボリュームに正しく割り当てられます。DatabaseRule ルールが適用されないファイルに関しては、VxFS はポリシーのスキャンを続け、GeneralRule ルールの CREATE 文に指定された仕様に従って領域を割り当てます。

配置ポリシールールの文にも同じ考え方が適用されます。VxFS は、この文を順番に処理し、ファイルに関連する文を見つると、そのファイルのための処理を終了します。これにより、意図しない操作が実行される可能性もあります。

次の XML の抜粋は、過去 30 日以内にアクセスのなかったファイルを再配置し、過去 90 日以内にアクセスのなかったファイルを削除しようとするルールの RELOCATE 文と DELETE 文です。

```
<RELOCATE>
  <TO>
    <DESTINATION>
      <CLASS>tier2</CLASS>
```

```
    </DESTINATION>
  </TO>
  <WHEN>
    <ACCAGE Units="days">
      <MIN Flags="gt">30</MIN>
    </ACCAGE>
  </WHEN>
</RELOCATE>
<DELETE>
  <WHEN>
    <ACCAGE Units="days">
      <MIN Flags="gt">90</MIN>
    </ACCAGE>
  </WHEN>
</DELETE>
```

RELOCATE 文を DELETE 文よりも前に記述すると、どのファイルも削除されません。理由は、少なくとも 30 日間アクセスがないために選択されたすべてのファイルに、RELOCATE 文の <WHEN> 節が適用されるからです。これには、90 日間アクセスのないファイルも含まれます。VxFS は、ファイルに適用する文を特定した時点で、配置ポリシーに対するそのファイルの処理を終了するので、DELETE 文が処理されることはありません。この例は、RELOCATE 文と DELETE 文の一般的なポイントを示しています。それは、ファイル配置ポリシー文書の中で、小さな包括基準を指定した RELOCATE 文と DELETE 文は、大きな包括基準を指定した文よりも前に記述するということです。GUI を使うと、文が正しい順序で自動的に並べられ、ポリシーが作成されます。

ファイル配置ポリシーとファイルの拡張

ファイル配置ポリシーがアクティブな VxFS ファイルシステムでは、ファイルが存在するボリュームの配置クラスは、ファイルのメタデータの中にあり、ファイルが作成される時と再配置でファイルが更新される時に設定されます。アプリケーションがファイルを拡張する時に、VxFS はファイルが占有しているボリュームに増分領域を割り当てます（可能な場合）。不可能な場合、VxFS は同じ配置クラスの別のボリュームに領域を割り当てます。たとえば、ファイルが tier1 ボリュームに作成され、後に tier2 ボリュームに再配置された場合、再配置前のファイルの拡張では tier1 ボリュームに領域が割り当てられますが、再配置後の拡張では tier2 ボリュームに領域が割り当てられます。この例では、ファイルが再配置されるときに、ファイルに割り当てられたすべての領域（拡張により取得された領域を含む）は tier2 ボリュームに再配置されます。

ソリッドステートディスクでの SmartTier の使用

SmartTier 配置ポリシーは次の機能によって SSD ベース階層をサポートします。

- <IOTEMP> と <ACCESSTEMP> 基準の単位として時間を許可するなど、頻度を詳細に調整できる
  p.889 の「ソリッドステートディスクとの微粒子の気温」を参照してください。
- <IOTEMP> と <ACCESSTEMP> 基準で Prefer 属性を使用できる
  p.890 の「ソリッドステートディスクの Prefer 機構」を参照してください。
- 平均 I/O アクティビティに基づいて再配置する機構
  p.890 の「ソリッドステートディスクの Average I/O アクティビティ基準」を参照してください。
- メモリ、CPU、I/O 帯域幅などのリソースに対する影響を最小化するために、スキャンの強度と期間が縮小している
  p.891 の「ソリッドステートディスクでの SmartTier のスキャン頻度」を参照してください。
- コールドファイルのクイック識別
  p.891 の「ソリッドステートディスクのコールドファイルのクイック識別」を参照してください。

これらの拡張機能を利用するには、最新の DTD に基づいて既存の配置ポリシーを修正してから、ポリシーを割り当て直す必要があります。ただし、既存の配置ポリシーは従来どおり機能します。これらの新機能を使わない場合は、配置ポリシーを更新する必要はありません。

ソリッドステートディスクとの微粒子の気温

SmartTier 機能では、SSD（Solid State Disk）拡張が導入されるまでは、頻度値が 1 日単位で計算されていました。つまり、日単位の I/O アクティビティが 1 日以上にわたって計算されます。したがって、<IOTEMP> と <ACCESSTEMP> 基準の <PERIOD> 要素は日単位で指定する必要がありました。SSD の場合は、Veritas File System（VxFS）が I/O アクティビティを 1 日よりも短い期間で測定するので、それに基づいて 1 日よりも短い単位で再配置を決定することをお勧めします。このことを踏まえて、<IOTEMP> と <ACCESSTEMP> 基準の Units 属性値に「時間」を指定できるようになりました。

p.850 の「I/O 頻度再配置基準の指定」を参照してください。

次の配置ポリシーの例では、期間として 4 時間を指定しています。

```
<RELOCATE>
   ...
   <WHEN>
      <IOTEMP Type="nwbytes">
         <MIN Flags="gteq"> 2 </MIN>
            <PERIOD Units="hours"> 4 </PERIOD>
      </IOTEMP>
```

```
    </WHEN>
</RELOCATE>
```

ソリッドステートディスクの Prefer 機構

<IOTEMP> と <ACCESSTEMP> 基準に Prefer 属性値を指定できるようになり、ファイルを再配置するときの優先設定を設定します。

p.852 の「Prefer 属性」を参照してください。

ソリッドステートディスク(SSD)ベースの階層の場合は、I/O アクティビティに著しい増加が見られる場合は、ファイルをすぐに SSD に再配置することをお勧めします。ただし、ファイルが SSD に再配置された後は、ファイルの移動の繰り返しを避けるために、アクティビティが高い間はファイルを SSD 上に残しておく方がいい場合もあります。ファイルを SSD から移動することを決める前に、SSD に再配置したときのアクティビティ監視時間よりも長くアクティビティを監視するようにしてください。

次の配置ポリシーは、Prefer 基準の例です。

```
<RELOCATE>
    ...
    <WHEN>
        <IOTEMP Type="nrbytes" Prefer="high">
            <MIN Flags="gteq"> 3.4 </MIN>
            <PERIOD Units="hours"> 6 </PERIOD>
        </IOTEMP>
    </WHEN>
</RELOCATE>
```

I/O 頻度が最小値より高いファイルがいくつかある場合には、頻度の低いファイルよりも頻度の高いファイルが最初に RELOCATE 操作の対象になります。これは SSD の場合に特に役に立ちます。SSD は大きさが限られていて高価であるためです。したがって SSD は、ほとんどアクティブなファイルに使用することをお勧めします。

ソリッドステートディスクの Average I/O アクティビティ基準

ソリッドステートディスク(SSD)拡張以前には、SmartTier 配置ポリシーで ACCESSTEMP 基準と IOTEMP 基準を使うときに頻度の絶対値を指定する必要がありました。ただし、そのような絶対値を求めることは難しく、データアクセスパターンを一定期間実験して観察する必要があります。さらに、時間の経過とともにアクセスパターンが変更するため、この値の変更が必要になる場合があります。このため、実験を繰り返さなければならないことがあります。ACCESSTEMP と IOTEMP ベースのポリシーを簡単に作成できるように、新しい基準 Average が導入されました。

p.852 の「Average I/O アクティビティ基準」を参照してください。

ソリッドステートディスクでの SmartTier のスキャン頻度

Units 属性値に「時間」を指定することで、I/O 統計情報収集期間を以前のリリースより大幅に短縮できます。ソリッドステートディスク(SSD)を使わないときは、Units 属性値に「日」を指定するだけでかまいません。ただし、SSD を使う環境では、候補のファイルとそれらのアクティビティレベルが 1 日の中で変化する可能性があるので、日よりも短い期間が必要になります。その結果、SmartTier によるスキャン頻度を高める必要があるため、ホストシステムへのスキャン負荷が高まります。

次の競合する要件を同時に満たす必要があります。

- 頻度収集期間を時間レベルにする。
- CPU、I/O、メモリなどのリソースに対するスキャン回数の増加による影響を軽減する。

頻繁にスキャンすることで発生する負荷を軽減する方法をいくつか紹介します。

- -C オプション付きで fsppadm コマンドを実行して FCL(File Change Log)にアクティビティが表示されるファイルだけを対象にすることで、期間中にアクティブなファイルだけをスキャンする。
  p.891 の「ソリッドステートディスクのコールドファイルのクイック識別」を参照してください。
- 頻繁にスキャンする(数時間ごとなど)。頻繁にスキャンすることで、VxFS がアクセスして FCL(File Change Log)ファイルにログを記録する i ノードの数が減少し、それによって各スキャンの時間が短縮される可能性があります。つまり、より少ないアクティブファイルの詳細について、前回のスキャンからの変更が FCL ファイルに収集されることになります。
- <IOTEMP> と <ACCESSTEMP> 基準を使って、より頻繁にファイルを SSD に再配置する。コールドファイルを SSD に残すようにします。

ソリッドステートディスクのコールドファイルのクイック識別

配置機構では通常、コールドファイルの非アクティブ状態が続く場合には、それらがソリッドステートディスク(SSD)に置かれたままになります。これによって、アクティブファイルを SSD に移動する必要がある場合はそれらのファイルの領域が不足し、ストレージの使用効率が低下することになります。この問題は、コールドファイルを識別するための SSD 拡張によって簡単に解決します。

この拡張は、SmartTier の特定の階層でファイルをすばやく識別し、必要に応じてファイルを再配置するための方法です。この方法は、ストレージデバイスをそれらに含まれるファイルの i ノードに関連付けるマップで構成されます。

ファイル場所のマップは次のときに更新されます。

- SmartTier 自体のファイルが再配置されるとき

- SmartTier のスコープの外部で行われた変更のために、ファイルシステムの FCL（File Change Log）が検査されるとき

どちらの更新も SmartTier の再配置スキャン中に行われます（通常は定期的に行うように設定されています）。ただし、ファイル場所のマップは、-T オプション付きで fsppadm コマンドを実行することでいつでも更新できます。

-C オプションは、アクティブファイルを他のファイルの前に処理するときに使用できます。-C オプションは -T オプションと一緒に指定することをお勧めします。fsppadm コマンドに -T オプションと -C オプションを両方指定すると、-C オプションによって SSD 階層に移動するアクティブファイル用の領域を SSD ティアに作成するために、まずコールドファイルが退避されます。-T と一緒に -C を指定することで、スキャンのスコープが限定され、消費される時間とリソースが少なくなるため、頻繁なスキャンが可能になり、データ配置要件に動的に対応できます。

p.837 の「配置ポリシーの実施」を参照してください。

fsppadm（1M）のマニュアルページを参照してください。

ファイルシステム全体をスキャンする代わりにマップを使うことによって、SSD 階層のファイルと FCL に記録されるアクティブファイルだけにスキャンを限定できます。この方法によって、I/O 頻度を少なくしながらスキャン負荷を減らすという 2 つの目的を達成できる可能性があります。

ソリッドステートディスクを使うときの配置ポリシーの例

配置ポリシーをソリッドステートディスク（SSD）ベースの階層で使う場合の例です。

```
<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM
"/opt/VRTSvxfs/etc/placement_policy.dtd">
<PLACEMENT_POLICY Version="5.0" Name="SSD_policy">
  <RULE Flags="data" Name="all_files">
    <COMMENT>
      The first two RELOCATEs will do the evacuation
      out of SSDs to create room for any relocations
      into the SSDs by the third RELOCATE. The parameters
      that can be tuned are basically values for PERIOD and
      the values of MIN and/or MAX as the per the case.
      The values for MIN and MAX are treated as multiples of
      average activity over past 24 hour period.
    </COMMENT>
    <SELECT>
      <PATTERN> * </PATTERN>
    </SELECT>
```

```
<CREATE>
  <COMMENT>
    create files on ssdtier, failing which
    create them on other tiers
  </COMMENT>
  <ON>
    <DESTINATION Flags="any">
      <CLASS> ssdtier </CLASS>
    </DESTINATION>
  </ON>
</CREATE>

<RELOCATE>
  <COMMENT>
    Move the files out of SSD if their last 3 hour
    write IOTEMP is more than 1.5 times the last
    24 hour average write IOTEMP. The PERIOD is
    purposely shorter than the other RELOCATEs
    because we want to move it out as soon as
    write activity starts peaking. This criteria
    could be used to reduce SSD wear outs.
  </COMMENT>
  <FROM>
    <SOURCE>
      <CLASS> ssdtier </CLASS>
    </SOURCE>
  </FROM>
  <TO>
    <DESTINATION>
      <CLASS> nonssd_tier </CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <IOTEMP Type="nwbytes" Average="*">
      <MIN Flags="gt"> 1.5 </MIN>
      <PERIOD Units="hours"> 3 </PERIOD>
    </IOTEMP>
  </WHEN>
</RELOCATE>

<RELOCATE>
  <COMMENT>
    OR move the files out of SSD if their last 6 hour
```

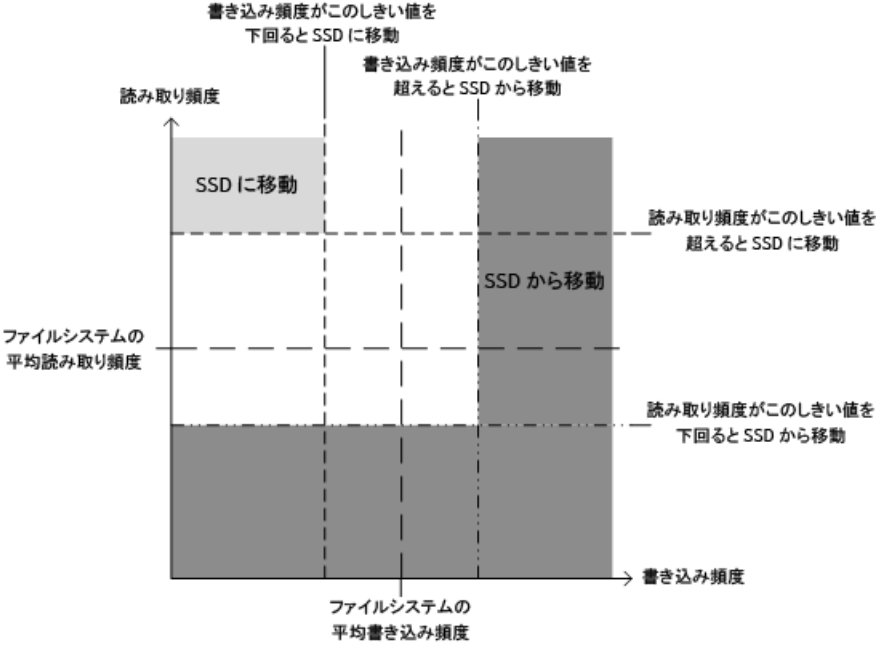
```
    read IOTEMP is less than half the last 24 hour
    average read IOTEMP. The PERIOD is longer,
    we may want to observe longer periods
    having brought the file in. This avoids quickly
    sending the file out of SSDs once in.
  </COMMENT>
  <FROM>
    <SOURCE>
      <CLASS> ssdtier </CLASS>
    </SOURCE>
  </FROM>
  <TO>
    <DESTINATION>
      <CLASS> nonssd_tier </CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <IOTEMP Type="nrbytes" Average="*">
      <MAX Flags="lt"> 0.5 </MAX>
      <PERIOD Units="hours"> 6 </PERIOD>
    </IOTEMP>
  </WHEN>
</RELOCATE>

<RELOCATE>
  <COMMENT>
    OR move the files into SSD if their last 3 hour
    read IOTEMP is more than or equal to 1.5 times
    the last 24 hour average read IOTEMP AND
    their last 6 hour write IOTEMP is less than
    half of the last 24 hour average write IOTEMP
  </COMMENT>
  <TO>
    <DESTINATION>
      <CLASS> ssd_tier </CLASS>
    </DESTINATION>
  </TO>
  <WHEN>
    <IOTEMP Type="nrbytes" Prefer="high" Average="*">
      <MIN Flags="gteq"> 1.5 </MIN>
      <PERIOD Units="hours"> 3 </PERIOD>
    </IOTEMP>
    <IOTEMP Type="nwbytes" Average="*">
```

```
            <MAX Flags="lt"> 0.5 </MAX>
            <PERIOD Units="hours"> 3 </PERIOD>
          </IOTEMP>
        </WHEN>
      </RELOCATE>
    </RULE>
</PLACEMENT_POLICY>
```

この配置ポリシーでは、使用可能な領域があれば SSD 階層に、使用可能な領域がなければ他の場所に、新しいファイルが作成されます。ポリシーが適用されると、現在 SSD にあるファイルは、指定の期間にその書き込みアクティビティの増加がしきい値を超えるか、読み取りアクティビティがしきい値を下回る場合に、SSD から移動されます。この目的は最初の 2 つの RELOCATE で行われます。ただし、指定の期間に読み取りアクティビティがしきい値を超えているけれども、書き込みアクティビティがしきい値を超えていないファイルは、SSD に移動されます。読み取りアクティビティが高いファイルが優先されます。

次の図は、配置ポリシー例の動作を示しています。



I/O アクティビティが薄い灰色領域にあるファイルは、SSD ストレージに移動する候補として適しています。これらのファイルは書き込みアクティビティが少ないのでウェアレベリングへの影響が少なく、SSD への書き込み時間が遅いことはそれほど問題になりません。読み取りアクティビティが非常に多いことも、これらのファイルは SSD に配置するファイルとして適していることの理由です。読み取りアクティビティはウェアレベリングに影響せ

ず、SSDから読み取る方が速く読み取れるためです。一方、I/Oアクティビティが濃い灰色領域にあるファイルは、書き込みアクティビティが多いか読み取りアクティビティが少ないので、SSD ストレージから移動する候補として適しています。書き込みアクティビティが大きくなると、ウェアレベリングによる書き込み回数が増えるのでSSDの書き込み時間が遅くなり、ファイルシステムの処理効率は低下します。これらのファイルでは読み取りアクティビティが少ないので、SSD の読み取り速度が高速であることの利点はありません。

サブファイルリロケーション

サブファイルリロケーション機能は指定したファイルのデータ範囲を指定されたターゲット階層に再配置します。指定されたマウントの指定されたノードで一度に1つのインスタンスのみが可能です。

サブファイルデータは `fsppadm subfilemove` コマンドを使用して移動できます。このフレームワークを使ったアプリケーションは、外部スケジュール機能を介して任意の間隔で定期的に `fsppadm subfilemove` コマンドを呼び出して再配置を実行します。クラスタファイルシステムで負荷分散が必要な場合は、クラスタの各ノードでアプリケーションが `subfilemove` 呼び出さなければならない場合があります。また、アプリケーションがこれらのノードまたはマウントのサブファイルリロケーションを必要とする場合、アプリケーションは新しいマウントと再ブートのためにこの再配置の開始を調整する必要があります。

クラスタでは、各ノードが同じ間隔で統計を収集することになっていても複数のノードから実施が起こる場合があるため、データベースへの各ノードの永続性は各ノード間でわずかに非同期になることがあります。実施は統計情報の収集に続くため、各ノードの実施を数分のラグとともにスケジュールし、すべてのノードがその時間までに同期する統計を完了できるようにすることを推奨します。ほとんどの場合は、5 分のタイムラグで十分です。

メモ: サブファイルリロケーション機能を使っている間は、SmartTier を使用してファイルを圧縮することはできません。

ファイルのサブファイルデータは特定のターゲット階層への移動

`fsppadm`(1M)のマニュアルページを参照してください。

次の例は、オフセット 64 MB から 96 MB までの `test.dbf` ファイルの合計 32 MB を既存の階層から `tier2` に移動します:

```
# cat /var/tmp/list
test.dbf 67108864 100663296 tier2
# fsppadm subfilemove -f /var/tmp/list /mount1
```

38

ホットリロケーションの管理

この章では以下の項目について説明しています。

ホットリロケーションについて

ボリュームにディスク I/O 障害がある場合（たとえば、ディスクに修正不能なエラーがある場合など）、Veritas Volume Manager（VxVM）は障害に関係するプレックスを切断します。そのプレックスの I/O 処理は停止しますが、ボリュームの他のプレックスの I/O 処理は継続します。

ディスク全体に障害が発生した場合、VxVM はそのディスクをディスクグループから切断できます。そのディスク上のすべてのプレックスが無効になります。切断するディスク上に非ミラーボリュームが存在する場合は、そのボリュームも切断されます。

ディスク障害のような症状が見られる場合でも、その原因は物理ディスクメディアまたはディスクコントローラの障害ではなく、ケーブル、ホストバスアダプタ、電源などの中間的または補助的コンポーネントの障害にある可能性があります。

VxVM のホットリロケーション機能は、ディスク障害を自動的に検出し、システム管理者と指定ユーザーに電子メールで通知します。また、スペアディスクと空きディスク領域を利用して冗長性を復元したり、ミラーボリュームと RAID 5 ボリュームへのアクセスを保持しようとします。

p.898 の「ホットリロケーションの動作方法」を参照してください。

ホットリロケーションが無効になっている場合または電子メールを受信しなかった場合は、`vxprint` コマンドまたはグラフィカルユーザーインターフェースを使って、ディスクの状態を検査する必要があります。コンソールやシステムメッセージファイルからドライバエラーメッセージを表示することもできます。

障害が起きたディスクは、手動で削除および交換する必要があります。

p.1057 の「ディスクの削除と交換」を参照してください。

ハードウェアが故障した後にボリュームとそのデータをリカバリする方法について詳しくは、『Veritas InfoScale トラブルシューティングガイド』を参照してください。

ホットリロケーションの動作方法

ホットリロケーションは、冗長性のある（ミラーまたは RAID 5）VxVM オブジェクトの I/O 障害に自動的に反応し、冗長性とこれらのオブジェクトへのアクセスを復元する機能です。VxVM はオブジェクトの I/O 障害を検出し、影響を受けたサブディスクをスペアディスクとして指定されているディスクまたはディスクグループ内の空き領域に再配置します。そして、VxVM は障害発生前に存在したオブジェクトを復元し、冗長化して再度アクセス可能にします。

部分的なディスク障害が発生した場合（すなわち、ディスク上の一部のサブディスクにのみ影響する障害の場合）は、障害発生部分の冗長データが再配置されます。影響を受けていないディスク部分の既存ボリュームには引き続きアクセスできます。

ホットリロケーションは、障害が発生したディスク上の冗長性のある（ミラーまたは RAID 5）サブディスクに対してのみ実行されます。障害が発生したディスク上の冗長性のないサブディスクは再配置されませんが、その障害はシステム管理者に通知されます。

ホットリロケーションはデフォルトで有効になっており、障害発生時にはシステム管理者が操作をしなくても実行されます。

ホットリロケーションデーモン `vxrelocd` は、次のタイプの障害を示す VxVM イベントを検出し、対処します。

ディスク障害	これは通常 VxVM オブジェクトの I/O 障害の結果として検出されます。VxVM はエラーの修正を試みます。エラーを修正できない場合、VxVM はディスクのプライベートリージョンにある設定情報へのアクセスを試みます。プライベートリージョンにアクセスできない場合は、ディスクに障害が発生したものと見なします。
プレックス障害	これは通常、プレックス内の修正不能な I/O エラー(プレックス内のサブディスクに影響を及ぼすエラー)の結果検出されます。ミラーボリュームの場合、プレックスが切断されます。
RAID 5 サブディスク障害	これは通常、修正不能な I/O エラーの結果として検出されます。サブディスクが切断されます。

vxrelocd デーモンでこのような障害が検出されると、次の手順が実行されます。

- vxrelocd デーモンは、システム管理者(および他の指定ユーザー)に電子メールで障害の発生と、影響を受けた VxVM オブジェクトを通知します。
  p.901 の「部分的なディスク障害発生時のメールメッセージ」を参照してください。
  p.902 の「障害発生時のメールメッセージ」を参照してください。
  p.919 の「ホットリロケーションの動作の変更」を参照してください。
- vxrelocd デーモンは次に、サブディスクの再配置が可能かどうかを確認します。vxrelocd デーモンは、障害が発生したディスクグループ内でホットリロケーションスペアとして予約(spare フラグを設定)されているディスク上で適切な領域を検索します。そして、この領域を使用してサブディスクを再配置します。
- スペアディスクが利用できないかまたは追加領域が必要な場合、vxrelocd デーモンは同じディスクグループ内のディスクの空き領域を利用します。ただし、ホットリロケーションの適用対象から除外(nohotuse フラグを設定)されているディスクの領域は利用されません。サブディスクの再配置が完了すると、再配置した各サブディスクをそのプレックスに再接続します。
- 最後に、vxrelocd は適切なリカバリ手順を開始します。たとえば、ミラーボリュームの場合はミラー再同期を、RAID 5 ボリュームの場合はデータリカバリを実行します。また、実行したホットリロケーションとリカバリ処理をシステム管理者に通知します。

再配置できるサブディスクが存在しない場合、vxrelocd はシステム管理者にその旨を通知し、その他の処理は実行しません。

警告: ホットリロケーションでは、再配置後にデータのレイアウトや処理効率が変わる場合があります。管理者は、ホットリロケーションの実行後に設定変更が必要かどうかを確認する必要があります。

次の場合は、障害が発生したサブディスクを再配置できません。

- 障害が発生したサブディスクが非冗長ボリューム（すなわち、ミラーまたは RAID 5 以外のタイプのボリューム）上にある場合。
- ディスクグループに十分なスペアディスクまたは空のディスク領域がない場合。
- 冗長化されていない使用可能領域が、障害が発生したプレックスのミラーを含むディスク上にある場合。
- 冗長化されていない使用可能領域が、RAID 5 ログプレックスまたは 1 つ以上の正常なサブディスクを含むディスク上にある場合。RAID 5 プレックス内の、障害の発生しているサブディスクを再配置することはできません。
- ミラーボリュームのデータプレックスに DRL（Dirty Region Log）のログサブディスクが含まれている場合、そのプレックスに所属するサブディスクの障害発生時に再配置を実行することはできません。
- RAID 5 ボリュームログプレックスまたはミラーボリューム DRL ログプレックスに障害が発生した場合は、新しいログプレックスが別の場所に作成されます。この場合は障害が発生したサブディスクを再配置する必要はありません。

`vxrelocd`（1M）マニュアルページを参照してください。

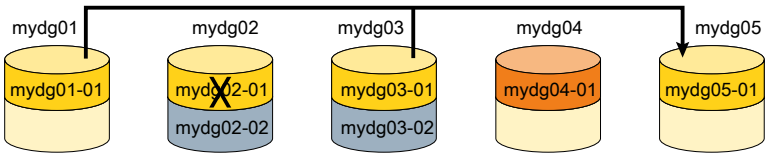
図 38-1 に、RAID 5 ボリュームのサブディスクの 1 つに障害が発生した場合のホットリロケーションプロセスを示します。

図 38-1 RAID 5 ボリュームのサブディスクのホットリロケーション例

a ディスクグループに 5 つのディスクが含まれている。 2 つの RAID 5 ボリュームが 4 つのディスクにまたがって設定されている。
ホットリロケーションに利用できるスペアディスクが 1 つある。

b 1 つの RAID 5 ボリュームのサブディスク mydg02-01 に障害が発生したとする。
ホットリロケーションによってスペアディスクにサブディスク mydg05-01 が作成され、サブディスク mydg02-01 と置き換えられる。その後、RAID 5 ボリュームのリカバリが開始される。

c RAID 5 のリカバリでは、サブディスク mydg01-01 と mydg03-01 上に残っているデータとパリティ情報に基づき、サブディスク mydg05-01 上にサブディスク mydg02-01 のデータとパリティが再作成される。



部分的なディスク障害発生時のメールメッセージ

ホットリロケーションが有効になっているときにプレックスやディスクが障害のため切断されると、障害が発生したオブジェクトを示すメールが root に送信されます。部分的なディスク障害が発生した場合、メールには障害が発生したプレックスが明記されます。たとえば、ミラーボリュームを含むディスクに障害が発生した場合は、次に例示するようなメール情報が送信されます。

```
To: root
Subject: Volume Manager failures on host teal
Failures have been detected by the Veritas Volume Manager:
failed plexes:
home-02
src-02
```

メールは root 以外のユーザーにも送信できます。

p.919 の「ホットリロケーションの動作の変更」を参照してください。

次のコマンドを使うと、前述のメッセージで通知された障害の原因となっているディスクを特定できます。

```
# vxstat -g mydg -s -ff home-02 src-02
```

-s オプションを指定すると、個々のサブディスクの情報が要求されます。-ff オプションを指定すると、失敗した読み取り処理と書き込み処理の数が表示されます。通常の出力表示は次のようになります。

```
                            FAILED
TYP NAME                   READS    WRITES
sd mydg01-04                   0         0
sd mydg01-06                   0         0
sd mydg02-03                   1         0
sd mydg02-04                   1         0
```

この例は、ディスク mydg02 のサブディスク mydg02-03 および mydg02-04 からの読み取りが失敗したことを示しています。

影響を受けたサブディスクがホットリロケーションによって自動的に再配置され、必要なリカバリ手順が開始されます。ただし、再配置が実行できない場合やホットリロケーション機能が無効になっている場合は、問題を調査し、プレックスのリカバリを実行する必要があります。エラーはケーブルの障害によって生じることもあるため、ディスクをシステムに接続しているケーブルをチェックしてください。問題が明らかな場合は、次のコマンドを使って問題を修正し、プレックスを修復します。

```
# vxrecover -b -g mydg home src
```

これにより、障害が発生したプレックスのリカバリがバックグラウンドで開始します(コマンドプロンプトは操作が完了する前に再表示されます)。この後にエラーメッセージが再表示されたり、明らかなケーブル障害がないにもかかわらずプレックスが再度切断された場合は、ディスクを交換してください。

p.1057 の「ディスクの削除と交換」を参照してください。

障害発生時のメールメッセージ

ホットリロケーションが有効になっているときに全体的なディスク障害が発生すると、障害が発生したディスクとそのディスクを使うすべてのプレックスがメールメッセージに一覧表示されます。たとえば、次に例示するようなメールが送信されます。

```
To: root
Subject: Volume Manager failures on host teal
```

```
Failures have been detected by the Veritas Volume Manager:

failed disks:
mydg02

failed plexes:
home-02
src-02
mkting-01

failing disks:
mydg02
```

このメッセージは、mydg02 が障害のため切断されたことを示しています。ディスクが切断されると、I/O はそのディスクに到達できなくなります。プレックス home-02、src-02 および mkting-01 も切断されています（原因はディスク障害と考えられます）。

問題の考えられる原因の 1 つにケーブルエラーがあります。

p.901 の「部分的なディスク障害発生時のメールメッセージ」を参照してください。

ケーブルエラー以外の問題の場合は、ディスクを交換してください。

p.1057 の「ディスクの削除と交換」を参照してください。

再配置領域の選択方法

スペアディスクを再配置に利用するには、初期化してスペアとしてディスクグループに配置しておく必要があります。障害が発生したときにスペアとして指定されているディスクがない場合、VxVM は自動的に、障害が発生したディスクグループの利用可能な空き領域を利用します。十分なスペアディスク領域がない場合、スペアディスク領域と空き領域を組み合わせて利用します。

ホットリロケーションでは、再配置の領域を選択する際に、再配置されるサブディスクが所属する VxVM オブジェクトの冗長特性が保持されます。たとえば、障害が発生したプレックスのサブディスクが、障害が発生したプレックスのミラーが保存されているディスクに再配置されることはありません。利用可能なスペアディスクや空き領域をすべて利用しても冗長性を保持できない場合は、ホットリロケーションは実行されません。再配置が実行できない場合はシステム管理者にその旨が通知され、その他の処理は実行されません。

ホットリロケーションには、条件を満たすディスクの中から障害が発生したディスクに最も近いディスクが利用されます。 近さの評価は、障害が発生したディスクのコントローラとディスク番号によって決まります。障害が発生したディスクと同じコントローラ上のディスクは、別のコントローラ上のディスクより近いものと判定されます。

ホットリロケーションでは、可能なかぎり、障害が発生しているドライブのすべてのサブディスクが同じディスクに移動されます。

ホットリロケーションが実行されると、障害が発生したサブディスクが設定データベースから削除され、VxVM は障害が発生したサブディスクが使っていたディスク領域が空き領域として再度使われないようにします。

FSS 環境でのホットリロケーションの動作方法

FSS 環境では、ホットリロケーションはストレージのエラーの修復にポリシーベースのメカニズムを採用しています。ストレージのエラーには、ディスクメディアのエラーやストレージをアクセス不能にするノードのエラーがあります。この機構では、チューニングパラメータを使って、ホットリロケーションの開始前にストレージがオンラインになるまで VxVM が待機する時間を決定します。指定した時間内にストレージがオンラインにならないと、VxVM は、エラーが発生したディスクを再配置します。

通常、ノード障害の最大の原因は、計画された保守のアクティビティです。これに対し、ディスクメディアの障害が発生する最大の原因は、物理ディスクまたはディスクコントローラの障害です。VxVM は、再配置のタイムアウト期間を設定および変更するために次のチューニングパラメータを使用します。

`storage_reloc_timeout` ディスクメディアでエラーが発生してから何分後に VxVM でホットリロケーションを開始するかを指定します。

`node_reloc_timeout` ノードでエラーが発生してから何分後に VxVM でホットリロケーションを開始するかを指定します。

`storage_reloc_timeout` チューニングパラメータのデフォルト値は 30 分で、`node_reloc_timeout` チューニングパラメータのデフォルト値は 120 分です。ビジネスニーズに合わせてチューニングパラメータ値を変更できます。

`vxtune` コマンドを使用すると、チューニングパラメータの設定を表示または更新することができます。

DAS 環境でのホットリロケーションプロセスは、共有環境と比べて若干異なります。DAS ディスクが失敗すると、VxVM は DCO が失敗していなくてもパフォーマンス上の理由から、同じノード上の別のディスクにデータボリュームとその関連 DCO ボリュームを再配置しようとします。再配置時に、VxVM は再配置に適切な空き領域の検索に失敗した場合、利用可能なスペアディスクを最優先します。

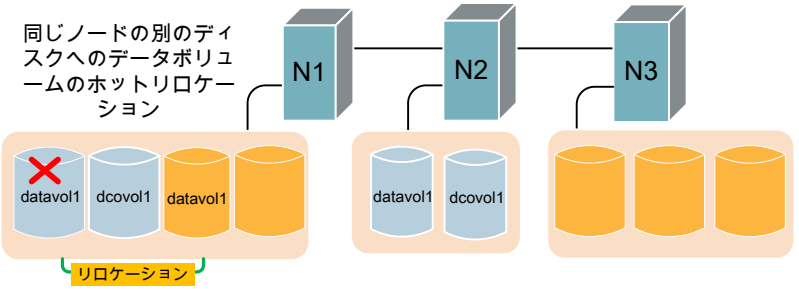
FSS 環境でのポリシーベースのホットリロケーションは、ディスクグループ バージョン 230 で作成された新しいディスクグループでサポートされています。既存のディスクグループでもポリシーベースのホットリロケーションを設定できますが、既存のディスクグループではスペアディスク、同じホストのリロケーション、データのコロケーション、DCO ボリュームに対する優先設定などの機能が利用できない場合があります。

DAS ストレージを使用するホットリロケーションのシナリオ

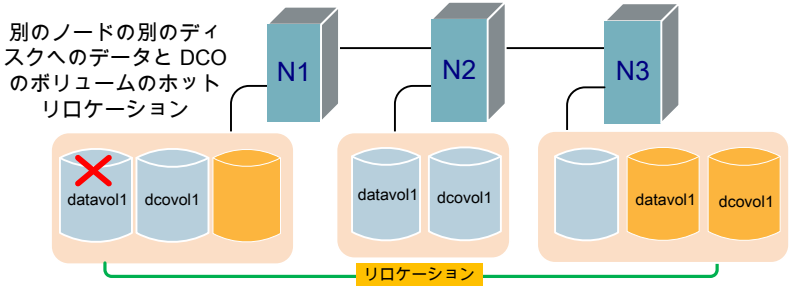
DAS ストレージのみに配置されたデータボリュームには、関連付けられた DCO ボリュームがあり、失効したミラーの迅速で効率的な再同期を促進します。パフォーマンス上の理由で、VxVM はホットリロケーション中に両方のボリュームを一緒に再配置しようとします。

データと DCO ボリュームが同じノード上にあるが、ディスク下位のデータボリュームのみが失敗する

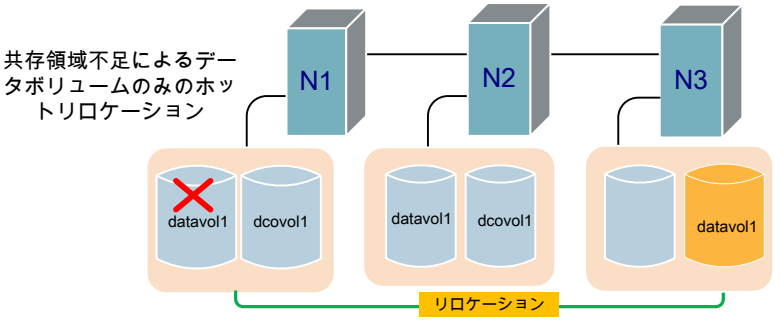
VxVM は、同じノード上の余剰 (空き) 領域である別のディスクにデータボリュームを再配置しようとします。



ローカルノードに必要な領域がない場合、VxVM はスペアディスクまたは十分な空き領域があり、データのミラーの複製が含まれないクラスタ内の別のノードにデータと DCO ボリュームを再配置します。



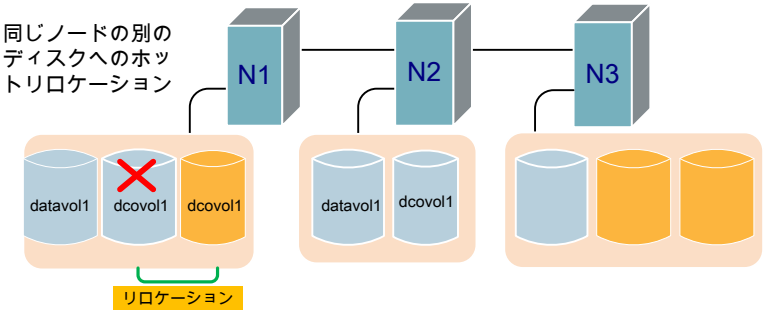
クラスタ内のノードにデータと DCO ボリュームの両方を再配置するための十分な領域がまったくない場合は、データボリュームのみが再配置されます。



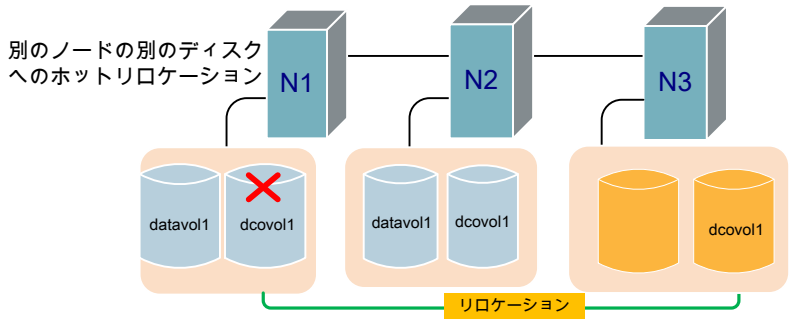
クラスタ内のノードにスペアディスクまたは十分な領域がまったくない場合、VxVM はエラーが発生したデータディスクを再配置しません。

データと DCO ボリュームが同じノード上にあるが、ディスク下位の DCO ボリュームのみが失敗する

VxVM は、同じノード上の余剰 (空き) 領域である別のディスクに DCO ボリュームを再配置しようとします。



ローカルノードに必要な領域がない場合、VxVM はスペアディスクまたは十分な空き領域があり、DCO ボリュームのミラーの複製が含まれないクラスタ内の別のノードに DCO ボリュームを再配置します。

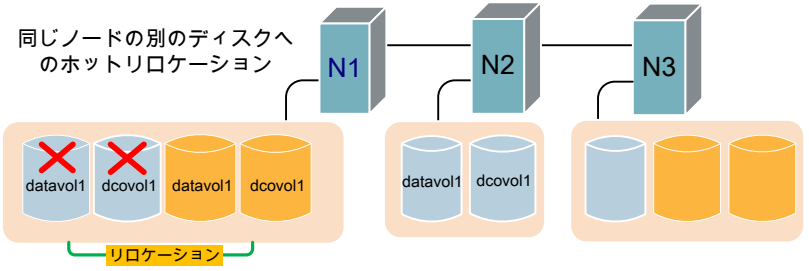


この場合、データボリュームは DCO とともに再配置されません。

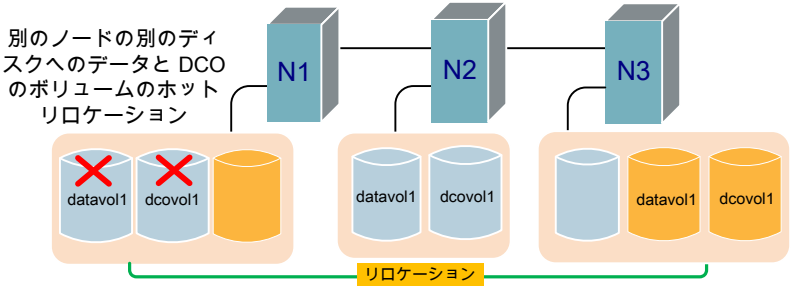
クラスタ内のノードにスペアディスクまたは十分な領域がまったくない場合、VxVM はエラーが発生した DCO ディスクを再配置しません。

データと DCO ボリュームが同じノード上にあるが、ディスク下位のデータと DCO ボリュームの両方が失敗する

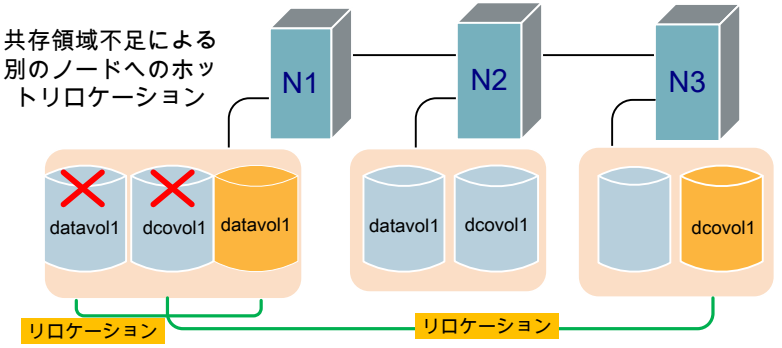
VxVM は、同じノード上の別のディスクまたは利用可能な空き領域にデータと DCO ボリュームを一緒に再配置しようとします。



ローカルノードに必要な領域がない場合、VxVM はスペアディスクまたは十分な空き領域があり、データボリュームと DCO ボリュームのミラーの複製が含まれないクラスタ内の別のノードにデータと DCO ボリュームを再配置します。



クラスタ内のノードにデータと DCO ボリュームの両方を一緒に再配置するための十分な領域がない場合、VxVM は別のノードでこれらのホットリロケーションを実行します。



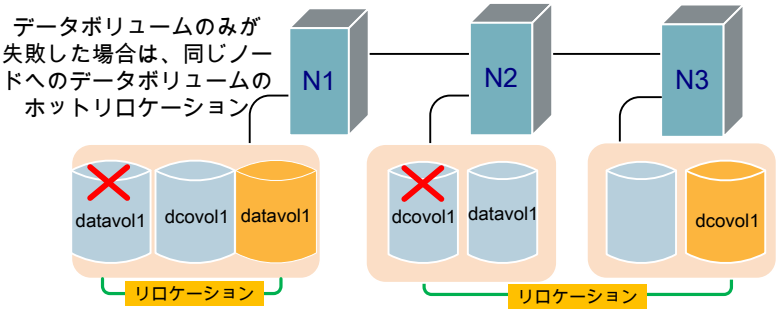
クラスタ内のノードにスペアディスクまたは十分な領域がまったくない場合、VxVM はエラーが発生したディスクを再配置しません。

データと DCO ボリュームが異なるノード上にあるが、ディスク下位のデータと DCO ボリュームのいずれかまたは両方が失敗する

データディスクのみにエラーが発生する場合、VxVM はスペアディスクまたは空き領域の可用性に応じて同じノードの別のディスクまたは他のノードにデータディスクを再配置します。DCO はデータボリュームと共存する場所に再配置されません。

同様に、DCO のみにエラーが発生する場合は DCO のみが再配置されます。

両方にエラーが発生する場合、VxVM はスペアディスクまたは空き領域の可用性に応じて両方をローカルまたは他のノードに再配置します。



共有 FSS 環境全体または一部でのでホットリロケーションのシナリオ

共有環境全体または一部でデータディスクにエラーが発生する場合は、エラーが発生したデータディスクのみを再配置します。DCO はエラーが発生しなければ再配置されません。エラーが DAS ストレージで発生した場合は、その DAS ストレージ内で再配置します。同様に、エラーが共有ストレージで発生した場合は、共有ストレージ内で再配置します。

図 38-2 に、共有 FSS 環境全体でのホットリロケーションを示します。

図 38-2　　共有 FSS 環境全体でのホットリロケーション

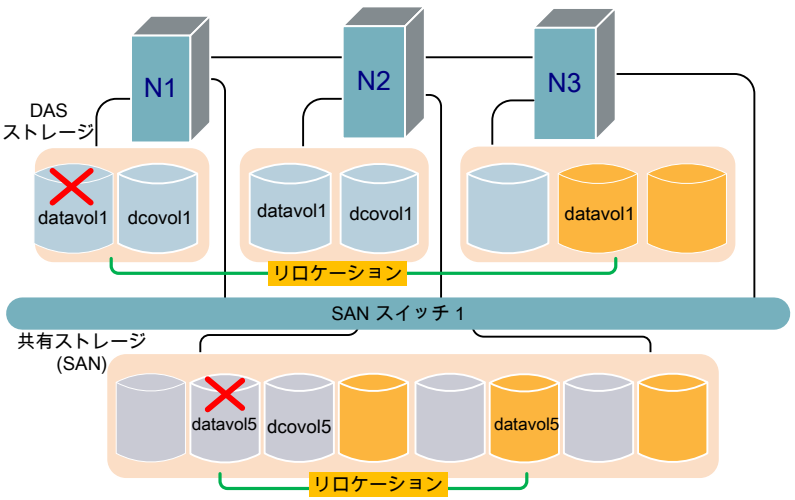
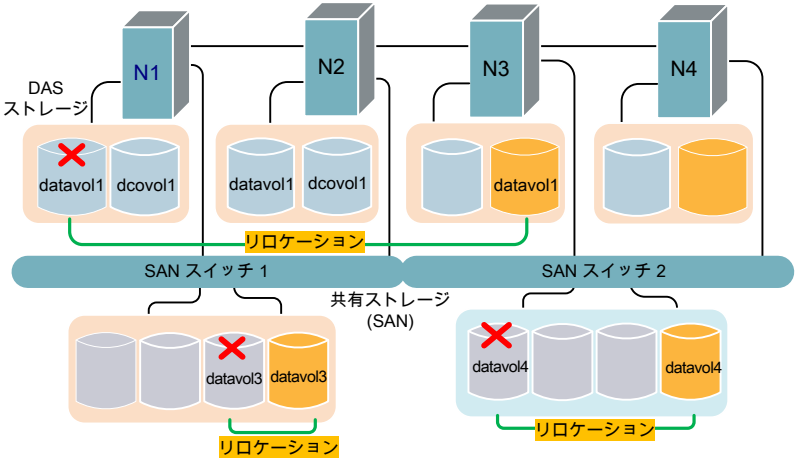


図 38-3 に、共有 FSS 環境の一部でのホットリロケーションを示します。

図 38-3 共有 FSS 環境の一部でのホットリロケーション



システムのホットリロケーション設定

スペアディスクを指定し、ディスク上の空き領域をホットリロケーションに利用できるように設定すると、ディスク障害発生時のサブディスクの再配置に伴うディスク領域の利用方法を制御することができます。空き領域とスペアディスクの領域を組み合わせても領域が不足する場合や、冗長性の制約を満たさない場合は、サブディスクは再配置されません。

スペアディスクまたはホットリロケーションの適用対象外のディスクを確認できます。

p.911 の「スペアディスク情報の表示」を参照してください。

ホットリロケーションの準備を整えるには、ディスクグループごとに 1 つ以上のディスクをホットリロケーションのスペアとして指定します。

p.911 の「ホットリロケーションのスペアディスクの設定」を参照してください。

必要に応じて、ホットリロケーションのスペアディスクの指定を解除できます。

p.913 の「ホットリロケーションスペアディスクの設定解除」を参照してください。

障害発生時に利用できるスペアがない場合またはスペアに十分な領域がない場合は、障害が発生したディスクと同じディスクグループ内のディスクの空き領域が自動的に利用されます。ただし、その領域がホットリロケーションの適用対象から除外されている場合はこの限りではありません。

p.913 の「ディスクのホットリロケーション適用対象からの除外」を参照してください。

p.914 の「ディスクのホットリロケーション適用対象からの除外を解除」を参照してください。

再配置されたサブディスクの位置によっては、再配置後にサブディスクを別の場所に移動することができます。

p.915 の「ホットリロケーションでスペアディスクのみを利用する設定」を参照してください。

再配置が正常に実行されたら、障害が発生したディスクを削除し、交換します。

p.1057 の「ディスクの削除と交換」を参照してください。

スペアディスク情報の表示

再配置に利用可能なスペアディスクに関する情報を表示するには、次のコマンドを使います。

```
# vxdg [-g diskgroup] spare
```

出力例を次に示します。

```
GROUP DISK      DEVICE       TAG       OFFSET    LENGTH    FLAGS
mydg  mydg02    sdc          sdc       0         658007    s
```

この場合、mydg ディスクグループの mydg02 がスペアとして指定されている唯一のディスクです。[長さ(LENGTH)]フィールドは、現在再配置に利用可能な mydg02 上のスペア領域の量を示しています。

現在スペアとして指定されているディスクに関する情報を表示するには、次のコマンドも使えます。

- vxdisk list を使うと、ディスク情報が一覧表示され、スペアディスクが spare フラグ付きで表示されます。
- vxprint を使うと、ディスク情報やその他の情報が一覧表示され、スペアディスクが SPARE フラグ付きで表示されます。
- vxdiskadm メインメニューの list メニュー項目を使うと、スペアディスクが一覧表示されます。

ホットリロケーションのスペアディスクの設定

ホットリロケーションは、I/O 障害に自動的に反応して冗長性のあるサブディスクを別のディスクに再配置する機能です。再配置に続いて、影響を受けた VxVM (Veritas Volume Manager) オブジェクトとデータが復元されます。ディスクグループ内にスペアとして指定されたディスクがすでに存在する場合、障害が発生したディスクのサブディスクはスペアディスクに再配置されます。

スペアディスクとして指定されたディスクがない場合、ディスクグループ内の適切な空き領域が利用されます。

ホットリロケーションのスペアディスクを指定するには、次のコマンドを入力します。

```
# vxedit [-g diskgroup] set spare=on diskname
```

ここで、*diskname* にはディスクメディア名を指定します。

たとえば、ディスクグループ mydg の mydg01 をスペアとして指定するには、次のコマンドを入力します。

```
# vxedit -g mydg set spare=on mydg01
```

vxdisk list コマンドを使うと、このディスクがスペアに指定されたことを確認できます。mydg01 は spare フラグ付きで表示されます。

これで、このディスクグループ内の VxVM ディスクはいずれも障害発生時にこのディスクをスペアとして利用できるようになります。ディスクに障害が発生すると(可能な場合)ホットリロケーションが自動的に実行されます。ユーザーには、電子メールで障害と再配置の情報が通知されます。再配置が正常に終了したら、障害が発生したディスクを交換できます。

vxdiskadm を使ってホットリロケーションのスペアディスクを指定するには

1. vxdiskadm のメインメニューで[ホットスペアディスクを設定(Mark a disk as a spare for a disk group)]を選択します。
2. 次のプロンプトでディスクメディア名を入力します(mydg01 など)。

   ```
   Enter disk name [<disk>,list,q,?] mydg01
   ```

   ディスクがスペアとして設定されると、次の注意メッセージが表示されます。

   ```
   VxVM NOTICE V-5-2-219 Marking of mydg01 in mydg as a spare disk
   is complete.
   ```

3. 次のプロンプトで、他のディスクをスペアとして設定する(y)か、vxdiskadm メインメニューに戻る(n)かを指定します。

   ```
   Mark another disk as a spare? [y,n,q,?] (default: n)
   ```

   これで、このディスクグループ内の VxVM ディスクはいずれも障害発生時にこのディスクをスペアとして利用できるようになります。ディスクに障害が発生すると(可能な場合)ホットリロケーションが自動的に実行されます。ユーザーには、電子メールで障害と再配置の情報が通知されます。再配置が正常に終了したら、障害が発生したディスクを交換できます。

ホットリロケーションスペアディスクの設定解除

スペアとして指定されているディスク上の領域は、そのディスクグループ内の VxVM オブジェクトの作成には使われません。必要に応じて、スペアディスクをホットリロケーションのディスクプールから削除して、一般の用途に解放できます。

ホットリロケーションプールからスペアを削除するには、次のコマンドを使います。

```
# vxedit [-g diskgroup] set spare=off diskname
```

ここで、*diskname* にはディスクメディア名を指定します。

たとえば、mydg01 をディスクグループ mydg で一般の用途に使えるように設定するには、次のコマンドを使います。

```
# vxedit -g mydg set spare=off mydg01
```

vxdiskadm を使ってディスクをホットリロケーションプールから削除するには

1. vxdiskadm のメインメニューで[ホットスペアディスクの設定を解除(Turn off the spare flag on a disk)]を選択します。
2. 次のプロンプトでスペアディスクのディスクメディア名を入力します(mydg01 など)。

   ```
   Enter disk name [<disk>,list,q,?] mydg01
   ```

   次の確認メッセージが表示されます。

   ```
   VxVM NOTICE V-5-2-143 Disk mydg01 in mydg no longer marked as
   a spare disk.
   ```

3. 次のプロンプトで、他のスペアディスクの設定を解除する(y)か、vxdiskadm メインメニューに戻る(n)かを指定します。

   ```
   Turn off spare flag on another disk? [y,n,q,?] (default: n)
   ```

ディスクのホットリロケーション適用対象からの除外

ディスクをホットリロケーションの適用対象から除外するには、次のコマンドを使います。

```
# vxedit [-g diskgroup] set nohotuse=on diskname
```

ここで、*diskname* にはディスクメディア名を指定します。

vxdiskadm を使ってディスクをホットリロケーションへの適用対象から除外するには

1. vxdiskadm のメインメニューで[ディスクのホットリロケーション適用対象からの除外(Exclude a disk from hot-relocation use)]を選択します。
2. 次のプロンプトでディスクメディア名を入力します(mydg01 など)。

   ```
   Enter disk name [<disk>,list,q,?] mydg01
   ```

   次の確認メッセージが表示されます。

   ```
   VxVM INFO V-5-2-925 Excluding mydg01 in mydg from hot-
   relocation use is complete.
   ```

3. 次のプロンプトで、他のディスクをホットリロケーションで利用可能な状態に設定する(y)か、vxdiskadm メインメニューに戻る(n)かを指定します。

   ```
   Exclude another disk from hot-relocation use? [y,n,q,?]
   (default: n)
   ```

ディスクのホットリロケーション適用対象からの除外を解除

障害が発生したサブディスクを再配置するのに十分なスペア領域がない場合、ホットリロケーションには空き領域が自動的に利用されます。ホットリロケーションに利用しない空きディスクを指定することによって、ホットリロケーションに利用できる空き領域を制限できます。ディスクがホットリロケーションの適用対象から除外されている場合は、必要に応じて除外を解除してディスクをホットリロケーションのプールに追加できます。

ディスクをホットリロケーションで利用可能な状態に設定するには、次のコマンドを使います。

```
# vxedit [-g diskgroup] set nohotuse=off diskname
```

vxdiskadm を使ってディスクをホットリロケーションが利用可能な状態にするには

1. vxdiskadm のメインメニューで[ディスクのホットリロケーション適用対象からの除外を解除(Make a disk available for hot-relocation use)]を選択します。
2. 次のプロンプトでディスクメディア名を入力します(mydg01 など)。

   ```
   Enter disk name [<disk>,list,q,?] mydg01
   ```

   次の確認メッセージが表示されます。

   ```
   V-5-2-932 Making mydg01 in mydg available for hot-relocation
   use is complete.
   ```

3. 次のプロンプトで、他のディスクをホットリロケーションで利用可能な状態に設定する(y)か、vxdiskadm メインメニューに戻る(n)かを指定します。

   ```
   Make another disk available for hot-relocation use? [y,n,q,?]
   (default: n)
   ```

ホットリロケーションでスペアディスクのみを利用する設定

VxVM がホットリロケーションでスペアディスクのみを利用するよう設定するには、次の行をファイル /etc/default/vxassist に追加します。

```
spare=only
```

スペアとして設定されているディスクのストレージが十分でない場合、再配置は失敗します。スペア以外のディスクの空き領域は利用されません。

再配置されたサブディスクの移動

ホットリロケーションが実行されると、サブディスクはスペアディスクまたはディスクグループ内の空き領域に再配置されます。新しいサブディスクの場所では、ホットリロケーションが実行される前とはパフォーマンスやデータのレイアウトが変わる場合があります。ホットリロケーションの完了後、再配置されたサブディスクを移動して、パフォーマンスの改善を図ることができます。

また、スペアディスクの再配置されたサブディスクを移動し、将来ホットリロケーションの必要があるときのためにスペアディスクに空き領域を確保しておくこともできます。サブディスクを移動するもう 1 つの理由として、ホットリロケーションの実行前にあった設定を作成し直すことがあります。

ホットリロケーション中に、root に送られる電子メールメッセージの 1 つの例を次に示します。

```
To: root
Subject: Volume Manager failures on host teal

Attempting to relocate subdisk mydg02-03 from plex home-02.
Dev_offset 0 length 1164 dm_name mydg02 da_name sdh.
The available plex home-01 will be used to recover the data.
```

このメッセージには、再配置前のサブディスクについての情報が含まれており、再配置の後にサブディスクの移動先を決定するためにこの情報を使うことができます。

再配置されたサブディスクの新しい場所を示すメッセージの例を示します。

```
To: root
Subject: Attempting VxVM relocation on host teal

Volume home Subdisk mydg02-03 relocated to mydg05-01,
but not yet recovered.
```

再配置されたサブディスクを移動する前に、失敗したディスクを修復する、または置き換えます。

p.1057 の「ディスクの削除と交換」を参照してください。

この実行後は、以降に記述されているように、元のディスクに戻って再配置されたサブディスクを移動できます。

警告: サブディスクの移動操作中、RAID-5 ボリュームは冗長ではなくなります。

vxunreloc を使った再配置されたサブディスクの移動

VxVM のホットリロケーションによって、システムはサブディスクレベルで冗長性のある VxVM オブジェクト上の I/O 障害に自動的に反応し、オブジェクトを再び使えるようにするために必要な処理を実行することができます。この機構によって、サブディスクの I/O 障害の検出、再配置およびそのサブディスクに関連付けられているプレックスのリカバリが実行されます。ディスクを交換した後、vxunreloc コマンドによってディスク障害発生前のシステムの設定を復旧できます。vxunreloc を使うと、ホットリロケーションされたサブディスクを障害のため交換されたディスク上に戻すことができます。

vxunreloc コマンドが呼び出されたら、ホットリロケーションされたサブディスクが属していたもとのディスクメディアの名前を指定する必要があります。vxunreloc コマンドでは、サブディスクはもとのオフセットに移動されます。障害が発生したもとのディスクより小さいディスクに再配置しようとすると、vxunreloc コマンドは何も実行せず、エラーを返します。

vxunreloc コマンドには、最初に再配置されたディスクとは異なるディスクにサブディスクを移動するオプションがあります。また、ディスクにすべてのサブディスクを収容できるだけの容量がある場合は、再配置処理を解除してサブディスクを別のオフセットに配置することもできます。

vxunreloc でサブディスクをもとのオフセットに戻せない場合は、指定ディスクの別のオフセットにサブディスクを移動する強制オプションを利用することができます。

vxunreloc(1M)マニュアルページを参照してください。

vxunreloc コマンドの使用例を次に示します。

サブディスクの再配置処理を解除してもとのディスクに配置

mydg01 に障害が発生し、すべてのサブディスクが再配置されたとします。mydg01 を交換した後、vxunreloc コマンドを使って、ホットリロケーションされたすべてのサブディスクを mydg01 に戻すことができます。

```
# vxunreloc -g mydg mydg01
```

再配置処理されたサブディスクを別のディスクに移動

vxunreloc ユーティリティには、最初に再配置されたディスクとは異なるディスクにサブディスクを移動する -n オプションがあります。

mydg01 に障害が発生し、そのすべてのサブディスクが他のディスクにホットリロケーションされたとします。vxunreloc コマンドには、最初に再配置されたディスクとは異なるディスクにサブディスクを配置するオプションがあります。修復されたディスクは、別の名前(たとえば、mydg05)でディスクグループに戻されます。ホットリロケーションされたサブディスクをすべて新しいディスクに配置するには、次のコマンドを使います。

```
# vxunreloc -g mydg -n mydg05 mydg01
```

新しいディスクには少なくともともとのディスクと同じ大きさのストレージが必要です。十分な領域がない場合は、再配置解除処理は失敗し、サブディスクは移動されません。

再配置処理されたサブディスクを別のオフセットに強制配置

デフォルトでは、vxunreloc コマンドはホットリロケーションされたサブディスクをもとのオフセットに移動しようとします。ただし、他のサブディスクがすでに配置先ディスクの一部または全部を占有している場合は、vxunreloc は失敗します。このような場合、次の 2 つの選択肢があります。

- 配置先ディスク上の既存のサブディスクを別の場所に移動し、vxunreloc コマンドを再実行します。

- サブディスクを新しいディスクに移動する際に vxunreloc コマンドによってディスク上の空き領域があるオフセット値を検索するようにするには、vxunreloc コマンドの -f オプションを使います。サブディスクを保存するディスクにすべてのサブディスクを収容できるだけの容量がある場合は、ホットリロケーションが行われたサブディスクはすべてもとのオフセットとは異なるオフセットに再配置解除されます。

mydg01 に障害が発生してサブディスクが再配置された場合に、ホットリロケーションが行われたサブディスクを、すでに他のサブディスクがもとのオフセット上に存在する mydg05 に配置するとします。強制オプションを使うと、ホットリロケーションが行われたサブディスクを mydg05 に配置することができます。ただし、オフセットは同じではありません。

```
# vxunreloc -g mydg -f -n mydg05 mydg01
```

ディスクからホットリロケーションされたサブディスクの検証

複数のディスク障害のため、サブディスクが複数回にわたってホットリロケーションが行われた場合も、再配置処理を解除してもとの位置に配置できます。たとえば、mydg01 に障害が発生して mydg01-01 という名前のサブディスクが mydg02 に配置され、次に mydg02 に障害が発生した場合、ホットリロケーションされたサブディスクも含め、mydg02 上のすべてのサブディスクが再度移動されます。mydg02 を交換し、mydg02 の vxunreloc コマンドを実行しても、ホットリロケーションされたサブディスク mydg01-01 に対しては何も実行されません。ただし、mydg01 を交換し、その後に vxunreloc コマンドを実行すると、mydg01-01 は mydg01 に戻されます。

障害が発生したディスクを修復または交換した後、vxunreloc コマンドを使って、再配置処理されたすべてのサブディスクをもとのディスク位置に戻すことができます。サブディスクがホットリロケーションされると、もとのディスクメディア名とディスクへのオフセットが設定データベースに保存されます。vxunreloc コマンドを使ってサブディスクをもとのディスクまたは新しいディスクに配置すると、この情報は削除されます。もとのディスクメディア名ともとのオフセットはサブディスクレコードに保存されます。mydg ディスクグループの mydg01 からホットリロケーションされたすべてのサブディスクを表示するには、次のコマンドを使います。

```
# vxprint -g mydg -se 'sd_orig_dmname="mydg01"'
```

vxunreloc のエラー後の再起動

vxunreloc コマンドは、サブディスクを次の 3 段階で移動します。

- vxunreloc コマンドは、再配置を解除するサブディスクと同じ数のサブディスクを宛先ディスクに作成します。文字列 UNRELOC が各サブディスクレコードの comment フィールドに入力されます。
  サブディスクはすべて作成されるかまったく作成されないかのどちらかです。vxunreloc コマンドですべてのサブディスクを正常に作成できない場合は、サブディスクは 1 つも作成されず、vxunreloc コマンドは終了します。

- vxunreloc コマンドは、各サブディスクのデータを、それぞれに対応して宛先ディスクに新しく作成されたサブディスクに移動します。
- すべてのサブディスクデータが正常に移動されると、vxunreloc コマンドは、現在 comment フィールドの設定が UNRELOC になっている宛先ディスクの各サブディスクについて、comment フィールドを null 文字列に再設定します。

宛先ディスク上のサブディスクの comment フィールドは、この 3 段階の処理が完了するまで、いずれも UNRELOC に設定されたままになります。実行が中断された場合、vxunreloc コマンドは中断前に宛先ディスクに作成したサブディスクを再度使うことはできますが、宛先ディスクに移動されたデータは使いません。

サブディスクのデータの移動が失敗すると、vxunreloc コマンドはエラーメッセージを表示して終了します。移動が失敗した原因を特定し、修正してから vxunreloc コマンドを再実行します。

宛先ディスクに新しいサブディスクが作成された後、すべてのデータが移動される前にシステムがダウンした場合は、システムの再ブート後に vxunreloc を再実行します。

警告: サブディスクレコードの comment フィールドの文字列 UNRELOC は変更しないでください。

ホットリロケーションの動作の変更

vxrelocd デーモンの処理の起動中、ホットリロケーションは有効になっています。障害発生時にホットリロケーション機能を利用するには、通常、この機能を有効のままにしておいてください。ディスクの空き領域を再配置に使わない場合など、ホットリロケーションを無効にする場合は、vxrelocd を呼び出す vxvm-recover 起動ファイルを編集することで、システムの起動時に vxrelocd デーモンが起動されないようにできます。

ホットリロケーションデーモンを無効にすると、削除されたボリューム上の自動ストレージ再生利用も無効になります。

vxrelocd デーモンの動作を次のように変更できます。

1 vxrelocd が起動されないようにするには、起動スクリプトファイルで vxrelocd を呼び出すエントリをコメントアウトします。

```
# nohup vxrelocd root &
```

2 デフォルトでは、障害が検出され、再配置処理が実行されると、vxrelocd デーモンによって root ユーザーに電子メールが送信されます。その他のユーザーにも通知されるようにするには、次に示すように該当するユーザー名を追加します。

```
# nohup vxrelocd root user1 user2 &
```

3 リカバリによるシステムのパフォーマンスの低下を緩和するには、次の例に示すように、ボリュームの各領域のリカバリ間の遅延時間を長くするよう、vxrelocd デーモンに指定します。

```
# nohup vxrelocd -o slow[=IOdelay] root &
```

ここで、オプションの値 *IOdelay* は、任意の遅延時間（ミリ秒）を示します。遅延時間のデフォルト値は 250 ミリ秒です。

39

データの重複排除

この章では以下の項目について説明しています。

データの重複排除について

データ重複排除機能はファイルシステムをまたがるブロックの比較によって、データが使用するブロックの重複を排除します。データ重複排除機能が重複したブロックを見つけると、使用領域は削除して代わりに共通ブロックへのポインタを作成します。重複ファイルを変更すると、ファイルが同じブロックを共有しなくなるため、ポインタの代わりに変更されたブロックがディスクに保存されます。CPU オーバーヘッドの継続的な費用をかけずに重複したデータを排除するために、ファイルシステムではプロセス後の定期的な重複排除を実行できます。データがオンデマンドで複製されたかどうかを確認し、その後に効率的かつ安全に重複を排除します。重複排除プロセスでは、次のタスクが実行されます。

- 変更のためにファイルシステムをスキャンする
- データのフィンガープリントを取る
- 重複を特定する
- 重複を確認したら重複を排除する

重複排除の結果取得できる空き領域は、データによって異なります。異なるデータの重複を排除する場合、空き領域も異なります。

fsdedupadm コマンドを実行してデータを重複排除します。

fsdedupadm(1M)マニュアルページを参照してください。

重複排除機能は Veritas InfoScale Storage ライセンスと Veritas InfoScale Enterprise ライセンスの両方で使用できます。

重複排除のチャンクサイズについて

重複排除のチャンクサイズは、重複排除の詳細度とも呼ばれ、フィンガープリントが計算される単位です。有効なチャンクサイズは、4k から 128k の間の 2 の累乗です。設定されると、チャンクサイズを変更する唯一の方法は、ファイルシステム上で重複排除を削除し、再び有効にする方法です。

重複排除とリソース要件に重大な影響を及ぼすため、チャンクサイズは慎重に選択する必要があります。サイズは、重複排除データベースのフィンガープリントレコード数、およびこれらのレコードのソートに必要な一時領域に直接影響します。チャンクサイズが小さいとフィンガープリントの数が多くなり、重複排除データベースにかなりの領域が必要になります。

重複排除を行って節約できるストレージ領域はデータセットおよびデータセット内での重複の分布に大きく依存しますが、節約できるストレージ領域にはチャンクサイズも大きく影響します。重複排除後に最善の結果を得るには、使用しているデータセットについて理解する必要があります。一般的には、チャンクサイズが小さいほど、より多くのストレージを節約できます。チャンクサイズが小さいとフィンガープリントがより細かくなり、結果として、通常、より多くの重複を特定できます。ただし、チャンクが小さいと、データベースサイズ、重複排除時間、さらに重要なことには断片化の面での負荷が高くなります。チャンクサイズが小さいと、重複排除データベースサイズはかなり大きくなることがあります。断片化の程度が高いと、通常、ファイルシステムメタデータが多くなり、結果としてより多くのストレージが必要になります。重複排除データベースが消費する領域と、ファイルシステムのメタデータの増加により、重複排除で節約できる領域が少なくなります。さらに、断片化も、パフォーマンスに悪影響を及ぼすことがあります。Veritas File System(VxFS)の重複排除アルゴリズムでは、連続した複数の重複チャンクをまとめることによって、断片化を減らすようにします。

チャンクサイズが大きいと、通常、重複排除データベースのサイズが小さくなり、重複排除の処理が高速化し、断片化が少なくなります。これらの利点により、節約されるストレージ領域が少なくなることがあります。サイズの小さい重複ファイルが多数存在する場合は、ファイルサイズよりも大きいチャンクサイズを選択する方法があります。チャンクサイズの方が大きいと、それよりも小さいファイルの重複排除には影響しません。この場合、フィンガープリントはファイル全体を対象に計算され、重複排除されます。

Veritasは仮想マシンイメージの複数のコピーに NFS を使ってアクセスする SFCFSHA のチャンクサイズは、4 K を推奨します。他のデータセットでは、チャンクサイズを 16 K 以上に設定することを推奨します。

重複排除データベースが消費する領域は、ファイルシステムと重複排除チャンクサイズのデータ量で表されます。重複排除データベースが消費する領域は、新しいデータがファイルシステムに追加されるにつれて時間とともに大きくなります。フィンガープリントのソートなどに一時的に使用するため、追加ストレージが必要になります。一時ストレージは、作業完了後に解放できます。重複排除が正常に完了できるように、十分な空き領域があることを確認してください。ファイルシステムの空き領域が 15% 未満であると、重複排除は開始しないことがあります。チャンクサイズが小さい場合に重複排除を実行するには、空き領域が 15% 以上必要になることがあります。一般に、チャンクサイズが大きいほど、消費される領域は大幅に少なくなります。チャンクサイズが 4 K の場合は、約 20% の空き領域があることを推奨します。

重複排除とファイルシステムパフォーマンス

Veritas File System（VxFS）の重複排除では、重複データを特定するとき、ストレージを節約するために共有エクステントを使います。共有エクステントにより、特定のタイプのアプリケーションでは、読み取り処理速度を大幅に高めることができます。これらの利点は、共有エクステントに存在するデータのファイルシステムのページキャッシュの斬新な使用法の結果としてもたらされます。

数多くの仮想マシンイメージにサービスを提供する SFCFSHA は、共有エクステントを使用することで、パフォーマンス上の重要なメリットを生み出しています。

共有エクステントについての詳細については、FileSnaps 機能の説明で取り上げています。

p.675 の「FileSnap について」を参照してください。

一般に、複数のファイルを介して共有エクステントに存在するデータを読み取るアプリケーションまたはアプリケーションセットでは、読み取り処理速度の向上が期待されます。

重複排除スケジューラについて

重複排除スケジューラは、すべてのノード上で実行され、ユーザー指定のスケジュールに従ってデータの重複排除を行うデーモンです。

ファイルシステムの重複排除を有効にした後、重複排除のスケジューラは自動的に開始しません。スケジューラを外部から開始する必要があります。

各ファイルシステムには、それぞれのスケジュールがあります。特定のファイルシステムのスケジュールと他の設定情報は、ファイルシステムに保存されます。設定ファイルの場所は `lost+found/dedup/local_config` です。

スケジューラは、設定ファイルの変更を 30 分ごとにチェックして、変更内容があれば組み込みます。この定期的なチェックでは、新しくマウントされたファイルシステムも対象です。スケジューラを再起動して、設定変更をすぐに組み込むことができます。

スケジューラを使用してファイルシステムのデータを自動的に重複排除するとき、ファイルシステムの変更の評価は FCL（File Change Log）機能によって行われます。日の観点

から重複排除のスケジュールがほとんど行われない場合、FCL がロールオーバーする可能性があり、このために FCL 機能でファイルシステムへの変更が見落とされる場合があります。

システムのアクティビティがあまり行われないときに重複排除をスケジュール設定することを推奨します。これで、スケジューラがシステムの通常の作業に干渉しないようになります。

データの重複排除

fsdedupadm コマンドを実行してデータを重複排除します。fsdedupadm コマンドを実行すると、次の操作が行われます。

機能	コマンドの構文
ファイルシステムの重複排除を有効にします。	`fsdedupadm enable [-c chunk_size] [-q] mount_point`
ファイルシステムの重複排除を無効にします。	`fsdedupadm disable [-q] mount_point`
ファイルシステムの重複排除の設定をクエリーします。	`fsdedupadm list mount_point|all`
ファイルシステムで重複排除の実行を開始します。	`fsdedupadm start [-s] [-q] mount_point`
ファイルシステムで重複排除の実行を停止します。	`fsdedupadm stop [-q] mount_point`
ファイルシステムの重複排除の状態をクエリーします。	`fsdedupadm status mount_point|all`
共有エクステントのスキップを有効または無効にします。	`fsdedupadm skipshared {true|false} mount_point`
スケジュール設定された重複排除ジョブが実行されるノードを設定します。	`fsdedupadm setnodelist nodelistmount_point|all`
ファイルシステムの重複排除スケジュールを設定します。	`fsdedupadm setschedule timemount_point`
ファイルシステムで重複排除のドライランを開始します。	`fsdedupadm dryrun [-o threshold=#] mount_point`

機能	コマンドの構文
ファイルシステムの重複排除設定ファイルと重複排除データベースを削除します。	fsdedupadm remove *mount_point*

コマンドについて詳しくは、fsdedupadm(1M)のマニュアルページを参照してください。

次の例では、ファイルシステムを作成し、ファイルシステム上に重複データを作成して、ファイルシステムを重複排除します。

ファイルシステムの重複排除の例

1 ファイルシステム fsvol1 を作成します。

```
# mkfs -t vxfs /dev/vx/rdsk/fsdg/fsvol1
```

2 ファイルシステムを /mnt1 としてマウントします。

```
# mount -t vxfs /dev/vx/dsk/fsdg/fsvol1 /mnt1
```

3 /mnt1 に一時ディレクトリ temp1 を作成し、ディレクトリに file1 ファイルをコピーします。

```
# mkdir /mnt1/temp1
# cd /mnt1/temp1
# cp /root/file1 .
# /opt/VRTS/bin/fsadm -S shared /mnt1
```

```
Mountpoint    Size(KB)    Available(KB)   Used(KB)   Logical_Size(KB) Space_Saved(KB)
          /mnt1         20971520      19335962      346609           602609         0
```

file1 ファイルのサイズは、fsadm コマンドの出力に示すようにおよそ 250 MB です。

4 別の一時ディレクトリ temp2 を作成し、新しいディレクトリに同じファイル file1 をコピーします。

```
# mkdir /mnt1/temp2
# cd /mnt1/temp2
# cp /root/file1 .
# /opt/VRTS/bin/fsadm -S shared /mnt1
```

```
Mountpoint  Size(KB) Available(KB) Used(KB) Logical_Size(KB) Space_Saved(KB)
/mnt1       4194304  3588700       548740   548740           0%
```

temp2 に同じファイルをコピーすることで、データが重複されました。fsadm コマンドの出力は、領域の 2 倍が使用されていることを示します。

5 マウントポイント /mnt1 の重複排除を有効にします。

```
# /opt/VRTS/bin/fsdedupadm enable -c 4096 /mnt1
# /opt/VRTS/bin/fsdedupadm list /mnt1
```

```
Chunksize Enabled SkipShared Schedule NodeList            Filesystem
-----------------------------------------------------------------------
4096      YES     True       NONE     node1               /mnt1
```

6 マウントポイント /mnt1 で重複排除の実行を開始します。

```
# /opt/VRTS/bin/fsdedupadm start /mnt1
UX:vxfs fsdedupadm: INFO: V-3-20: 0000: deduplication is started
on /mnt1.
```

7 重複排除の状態をチェックします。

```
# /opt/VRTS/bin/fsdedupadm status /mnt1
Saving    Status    Node            Type        Filesystem
-------------------------------------------------------------------------------
42%       COMPLETED fsaixp702-v05   MANUAL      /mnt1
2014/09/23 22:48:22 Begin full scan.
2014/09/23 22:51:45 End detecting duplicates and filesystem
changes.
```

8 使用されている領域をチェックして、ファイルシステムが重複排除されたかどうかを確認します。

```
# /opt/VRTS/bin/fsadm -S shared /mnt1
```

```
Mountpoint    Size(KB)    Available(KB)   Used(KB)   Logical_Size(KB) Space_Saved(KB)
            /mnt1       20971520      19335962       346609           602609
256000
```

出力では、ファイルシステムに file1 ファイルのコピーが 1 つだけある場合に使用されている領域とほぼ同一であることを示します。

ファイルシステムの重複排除の有効化と無効化

重複排除機能を使用するには、事前に fsdedupadm enable コマンドを使用してファイルシステムの重複排除を有効にする必要があります。

次の例では、/mnt1 にマウントされたファイルシステムの重複排除を有効にし、重複排除のチャンクサイズとして 4096 を指定します。

```
# /opt/VRTS/bin/fsdedupadm enable -c 4096 /mnt1
```

fsdedupadm disable コマンドを使うと、ファイルシステムの重複排除を無効にできます。

次の例では、/mnt1 にマウントされたファイルシステムの重複排除を無効にします。

```
# /opt/VRTS/bin/fsdedupadm disable /mnt1
```

ファイルシステムの重複排除のスケジュール設定

fsdedupadm setschedule コマンドを使用して、ファイルシステムが自動的に重複排除されるようにスケジュールを設定できます。周期の実行または周期の入力の 2 つのカテゴリのスケジュールオプションを指定できます。スケジュールの詳細設定は、時刻と日に限定されます。fsdedupadm コマンドは、スケジュールを設定するときに、関連する File Change Log のチューニングパラメータに適用されます。

p.1102 の「Veritas File System ファイルの変更ログの管理インターフェース」を参照してください。

スケジュールを設定するには、事前にファイルシステムの重複排除を有効にする必要があります。

p.926 の「ファイルシステムの重複排除の有効化と無効化」を参照してください。

重複排除は、毎時間または指定した時間ごと、毎日または指定した日数ごとに実行されるようにスケジュール設定できます。また、重複排除を実際の実行する時間を設定したり、指定時刻が経過すると指定した時間ごとに実行されるようにスケジュール設定することもできます。重複が排除されない場合、重複排除が実行されるとデータベースのフィンガープリントの更新のみが行われます。

スケジュールコマンドは累積されません。何らかの理由で前の重複排除プロセスが実行されているときに重複排除のスケジュールが開始すると、後の重複排除は破棄され、警告メッセージが表示されます。

スケジュールに対して二重引用符文字(")で囲んだ空の文字列を指定すると、スケジュールを削除できます。

fsdedupadm(1M)マニュアルページを参照してください。

タスクをスケジュールする前に fsdedupschd デーモンを起動してください。

RHEL 7、SLES 12、およびサポート対象の RHEL 互換配布の場合:

```
# systemctl enable fsdedupschd
# systemctl start fsdedupschd
```

以前のバージョンの RHEL、SLES、およびサポート対象の RHEL 互換配布の場合:

```
# chkconfig --add fsdedupschd
# service fsdedupschd start
```

次の例では、ファイルシステム /vx/fs1 の重複排除は、1 日おきに深夜に実行されます。

```
# fsdedupadm setschedule "0 */2" /vx/fs1
```

次の例では、ファイルシステム /vx/fs1 の重複排除は、1 日 2 回、深夜と正午に実行されます。

```
# fsdedupadm setschedule "0,12 *" /vx/fs1
```

次の例では、ファイルシステム /vx/fs1 の重複排除は 1 日 4 回実行されますが、実際には 4 回目の重複排除のみがファイルシステムの重複排除を行います。その他の実行では、スキャンと処理が実行されます。このオプションでは、システムだけでなくクラスタ全体の負荷を分散します。

```
# fsdedupadm setschedule "0,6,12,18 * 4" /vx/fs1
```

次の例では、/vx/fs1 ファイルシステムの重複排除スケジュールを削除します。

```
# fsdedupadm setschedule "" /vx/fs1
```

重複排除のドライランの実行

ファイルシステムを実際に変更せずに、重複排除の保存領域を判断するには、ドライランを実行します。ドライランを実行するには、事前にファイルシステムの重複排除を有効にする必要があります。以前に重複を排除されなかったファイルシステムのみにドライランを実行できます。

p.926 の「ファイルシステムの重複排除の有効化と無効化」を参照してください。

次のコマンドでは、ファイルシステム /mnt1 の重複排除のドライランを開始します。

```
# fsdedupadm dryrun /mnt1
```

-o threshold オプションを指定すると、実際の重複排除を実行するために fsdedupadm を指定できます。この場合、予測された領域保存が指定されたしきい値に到達すると、fsdedupadm コマンドは実際に重複排除を実行します。

次のコマンドでは、ファイルシステム /mnt1 の重複排除のドライランを開始し、予測された領域保存がしきい値 60 パーセントを超えると実際の重複排除が実行されます。

```
# fsdedupadm dryrun -o threshold=60 /mnt1
```

-o threshold オプションを指定すると、fsdedupadm コマンドが Storage Checkpoint を取り、ファイルシステムの File Change Log を有効にします。

ファイルシステムの重複排除の状態のクエリー

fsdedupadm status コマンドを使用して、ファイルシステムの重複排除の状態をクエリーできます。

重複排除の状態をクエリーするには、事前にファイルシステムの重複排除を有効にする必要があります。

p.926 の「ファイルシステムの重複排除の有効化と無効化」を参照してください。

次のコマンドでは、ファイルシステム /mnt1 の重複排除の状態をクエリーします。

```
# fsdedupadm status /mnt1
```

次のコマンドでは、実行中のすべての重複排除ジョブの重複排除の状態をクエリーします。

```
# fsdedupadm status all
```

重複排除スケジューラデーモンの起動と停止

重複排除スケジューラデーモン fsdedupschd の状態は、再ブートされた後も維持されます。再ブートする前に fsdedupschd デーモンを起動した場合、デーモンは再ブート後も自動的に再起動します。再ブートする前に fsdedupschd デーモンを停止した場合、再ブート後も停止したままです。デフォルトの fsdedupschd デーモンは停止状態です。

スケジューラデーモンを開始または停止するには、事前にファイルシステムの重複排除を有効にする必要があります。

p.926 の「ファイルシステムの重複排除の有効化と無効化」を参照してください。

次のコマンドによって fsdedupschd デーモンが起動します。

RHEL 7、SLES 12、およびサポート対象の RHEL 互換配布の場合:

```
# systemctl enable fsdedupschd
# systemctl start fsdedupschd
```

以前のバージョンの RHEL、SLES、およびサポート対象の RHEL 互換配布の場合:

```
# chkconfig --add fsdedupschd
# service fsdedupschd start
```

次のコマンドによって fsdedupschd デーモンが停止します。

RHEL 7、SLES 12、およびサポート対象の RHEL 互換配布の場合:

```
# systemctl stop fsdedupschd
# systemctl disable fsdedupschd
```

以前のバージョンの RHEL、SLES、およびサポート対象の RHEL 互換配布の場合:

```
# service fsdedupschd stop
# chkconfig --del fsdedupschd
```

重複排除の結果

データの性質は、重複排除を有効にするかどうか決定する場合に非常に重要です。JPEG、MP3、MOV などのデータベースやメディアファイルには、重複データがないか非常に少ないため、重複排除に適さない場合があります。仮想マシンのブートイメージファイル（vmdk ファイル）、ユーザーのホームディレクトリ、複数のファイルコピーがあるファイルシステムは、重複排除に適しています。通常は、重複排除のチャンクサイズが小さい場合はストレージの保存性が高くなりますが、重複排除には時間がかかり、より大きい重複排除データベースが必要です。

重複排除のサポート

VxFS (Veritas File System) はファイルシステムのディスクレイアウトバージョン 9 以降で重複排除をサポートします。

重複排除の使用例

次のリストは、重複排除機能を使用するいくつかの例を示しています。

ホームディレクトリ
: ユーザーのホームディレクトリには、複数バージョンの同一ファイルが類似した内容を持ち、それによって重複排除できる冗長データが存在することがよくあります。

ソースコードのディレクトリ
: 通常、ソースコードのリポジトリには、段階的に変更されている複数のファイルがあります。あるファイルから次のファイルに変更されないデータは、重複排除できます。

vmdk ファイル
: FileSnap 機能を使用して複数の仮想マシンのクローンを作成されると、クローンされた仮想マシンは、存続期間に渡ってシステムとセキュリティパッチを操作することになりがちです。そのようなアクティビティの結果としてソースから派生した共通のソース（ゴールデンイメージ）からクローンを作成される個々の仮想マシンとして、共通の内容が多く存在します。やがて、初期のストレージ節約は消失します。これらのファイルに追加される新しいブロックを重複排除すると、ストレージの節約がリストアされます。

重複排除の制限事項

重複排除機能には次の制限があります。

- 重複排除された VxFS（Veritas File System）ファイルシステムの完全バックアップでは、重複排除されなかったファイルシステムと同様の領域をターゲットに必要とする場合があります。たとえば、重複排除の後のファイルシステムに 1 TB と同様のディスク領域を占める 2 TB のデータがある場合、ファイルシステムをバックアップするには、バックアップのターゲットで重複排除が行われないことを想定して、このデータはターゲットに 2 TB の領域を必要とします。同様に、そのようなファイルシステムをリストアする場合、完全なデータをリストアするにはファイルシステムに 2 TB が必要です。ただし、新たにリストアされたファイルシステムは、領域保存を元に戻すために、再び重複排除できます。完全なファイルシステムのリストアを実行したら、`fsdedupadm remove` コマンドを使って既存の重複排除設定を削除し、`fsdedupadm enable` コマンドを使って重複排除を再設定することを推奨します。
- 重複排除は、ボリュームのプライマリファイルセットに限定されます。
- 重複排除はマウントされたクローンとスナップショットによってマウントされたファイルシステムをサポートしません。
- バックアップからデータをリストアしたら、重複排除による領域保存を元に戻すために、リストアされたデータの重複排除を行う必要があります。
- プラットフォーム間でのデータの変換にクロスプラットフォームデータシェアリング機能を使用する場合、重複排除設定ファイルと重複排除データベースを削除し、重複排除を再び有効にして、変換後に重複排除を再起動する必要があります。次の例に、実行する必要のあるコマンドを示します。コマンドは、記載されている順序で実行してください。

```
# /opt/VRTS/bin/fsdedupadm remove /mnt1
# /opt/VRTS/bin/fsdedupadm enable /mnt1
# /opt/VRTS/bin/fsdedupadm start /mnt1
```

- NetBackup の FlashBackup 機能をデータ重複排除機能と併用することはできません。FlashBackup では、ディスクレイアウトバージョン 8 と 9 はサポートされません。

40

ファイルの圧縮

この章では以下の項目について説明しています。

圧縮ファイルについて

ファイルを圧縮すると、アクセシビリティとアプリケーションに対する透過性を確保しながらも、使われる領域を削減できます。圧縮ファイルの見た目と操作性は、圧縮されていないファイルとほぼ同じです。圧縮ファイルは同じ名前を持ち、圧縮されていないファイルと同様に読み取りと書き込みができます。読み取りを行うと、メモリではデータが圧縮解除されます。ただし、ファイルのディスク上のコピーは圧縮された状態のままです。それに対して、書き込みが行われると、新しいデータはディスク上で圧縮解除されます。

ユーザーのデータのみ圧縮可能です。VxFS（Veritas File System）のメタデータは圧縮できません。

ファイルの圧縮後、i ノード番号は変更されず、圧縮の前に開かれるファイル記述子は圧縮の後も引き続き有効です。

圧縮はファイルのプロパティです。したがって、たとえばディレクトリのファイルをすべて圧縮した場合、そのディレクトリに後でコピーされたファイルは、自動的には圧縮されません。新しいファイルは、ディレクトリのファイルを再圧縮することでいつでも圧縮できます。

`vxcompress` コマンドを使用してファイルを圧縮します。

p.934 の「vxcompress コマンドを使用したファイルの圧縮」を参照してください。

vxcompress(1)マニュアルページを参照してください。

ファイルを圧縮するには、ディスクレイアウトバージョン 9 以降の VxFS ファイルシステムが必要です。

メモ: テープに圧縮ファイルをバックアップする際、バックアッププログラムは未圧縮の形式でデータを格納します。ファイルはメモリで圧縮解除され、続いてテープに書き込まれます。この結果、圧縮ファイルをバックアップする際に CPU とメモリ使用量が増加します。

圧縮ファイル形式について

圧縮ファイルとは、圧縮されたエクステントのファイルです。vxcompress 呼び出しによって、ファイルのすべてのエクステントが圧縮されます。ただし、ファイルに書き込みがされると、影響されるエクステントは圧縮解除されます。その結果、ファイルは圧縮されたエクステントと圧縮解除されたエクステントの両方になります。

ファイル圧縮の属性について

vxcompress コマンドを使用してファイルを圧縮するときに、vxcompress は i ノードに次の情報を添付します。

- 圧縮アルゴリズム
- 1 から 9 までの数字で表される圧縮の強度
- 圧縮のブロックサイズ

この情報は、ファイル圧縮の属性として参照されます。属性の目的は、圧縮ファイルを作成するために使用されるパラメータの収集です。 情報はこの後、バックアッププログラムによって読み取られます。

ファイル圧縮の属性によって、特定の圧縮ファイルが 1 つのタイプおよび強度の圧縮を使用できることが保証されます。 異なる属性を使用してファイルを再圧縮すると、失敗します。ファイル圧縮の属性を変更するには、すべてのエクステントがすでに圧縮解除されている場合でも、まず明示的に圧縮解除し、その後に新しいオプションを使用して再圧縮する必要があります。

ファイル圧縮の属性は、すべてのエクステントが圧縮されている場合でも、そのことを示しません。一部のエクステントが圧縮不可だったり、他のエクステントまたすべてのエクステントが書き込みによって圧縮解除される場合がありますが、ファイル圧縮の属性はそのまま残ります。ファイルの圧縮解除が明示的に示された場合のみ、属性は削除されます。

ファイル圧縮のブロックサイズについて

ファイル圧縮のアルゴリズムは、指定されたブロックサイズのデータを圧縮します。デフォルトは 1 MB に設定されます。各圧縮ブロックでは、i ノードに独自のエクステント記述子があります。ファイルまたは最新のエクステントが圧縮されたブロックサイズより小さい場合、より小さいサイズの方が圧縮されます。最大ブロックサイズは 1 MB です。

圧縮できないデータのエクステントは、引き続き圧縮済みのエクステントとしてマークされます。エクステントを圧縮できなかった場合でも、それらのエクステントに圧縮済みのマークを付けることによって、時間を節約するためにこの後の圧縮実行でこれらのエクステントがスキップされます。共有エクステントは圧縮できず、圧縮済みのマークもされません。ファイル圧縮のアルゴリズムは固定サイズのブロックを参照するため、アルゴリズムでは、ファイル圧縮のブロックサイズ単位でこれらの非圧縮性エクステントを見つけます。

vxcompress コマンドを使用したファイルの圧縮

vxcompress コマンドを使用してファイルを圧縮できます。vxcompress コマンドを実行すると、次の操作が行われます。

機能	コマンドの構文
ファイルまたはディレクトリツリーの圧縮	`vxcompress [-r] file_or_dir ...`
ファイルまたはディレクトリツリーの圧縮解除	`vxcompress -u [-r] file_or_dir ...`
ファイルまたはディレクトリツリーの圧縮保存を報告する	`vxcompress {-l|-L} [-r] file_or_dir ...`
サポート対象の圧縮アルゴリズムをリストする	`vxcompress -a`

vxcompress(1)マニュアルページを参照してください。

1 つ以上のファイル名を指定できます。-r オプションを指定すると、ディレクトリを指定でき、vxcompress コマンドはディレクトリで再帰的に動作します。

vxcompress -t コマンドを使用して、ファイル圧縮アルゴリズムと強度を指定できます。デフォルトのアルゴリズムは、現在唯一のサポート対象アルゴリズムである gzip です。強度は 1 から 9 の数値で表され、デフォルトは 6 です。強度 1 に設定すると、最小の圧縮で最も早いパフォーマンスを得られ、強度 9 に設定すると、最大の圧縮で最も遅いパフォーマンスが得られます。たとえば、強度 3 の gzip 圧縮は「gzip-3」と指定します。

ファイルの圧縮の詳細を報告するときに、vxcompress -l コマンドまたは vxcompress -L コマンドを実行すると次の情報が表示されます。

- 圧縮アルゴリズム
- 強度
- 圧縮のブロックサイズ
- 圧縮によって節約されるファイルデータの割合（%）
- 圧縮されたエクステントの割合（%）
  これは、エクステントのサイズに関係なく、圧縮されたファイルのエクステントの割合です。この割合を参考にして、ファイルを再圧縮する価値があるかどうかを判断できます。圧縮の後は、割合は常に 100% です。ただし、共有エクステントは圧縮解除として計算され、ファイルがエクステントを共有した場合は、割合は 100% 未満です。

vxcompress コマンドを使用してファイルの圧縮を試み、エクステントに圧縮できないデータがある場合、このコマンドはファイルを圧縮済みとしてマークし、このエクステントを圧縮済みのエクステント記述子と置き換えます。

ファイルを再圧縮する場合、vxcompress コマンドを使用してオプションを指定する必要はありません。コマンドは、以前にファイルを圧縮するときに使用したオプションを自動的に使います。

次のコマンドは、デフォルトのアルゴリズムと強度 gzip-6 を使用して、file1 ファイルを圧縮します。

```
$ vxcompress file1
```

次のコマンドは、最大強度（9）で gzip アルゴリズムを使用して、dir1 ディレクトリ下のすべてのファイルを再帰的に圧縮します。

```
$ vxcompress -r -t gzip-9 dir1
```

次のコマンドは、強度 3 で gzip アルゴリズムを使用して、file2 ファイルと dir2 ディレクトリ下のすべてのファイルを圧縮します。このとき、vxcompress コマンドは 1 つのスレッドに限定され、スケジュール設定の優先度は低く設定されます。

```
$ vxcompress -r -t gzip-3 file2 dir2
```

次のコマンドは、file1 ファイルの圧縮結果を分かりやすい単位で表示します。

```
$ vxcompress -L file1
%Comp    Physical    Logical  %Exts  Alg-Str  BSize  Filename
  99%       1 KB     159 KB   100%   gzip-6  1024k  file1
```

次のコマンドは、file1 ファイルを圧縮解除します。

```
$ vxcompress -u file1
```

圧縮ファイルと他のコマンドの相互関係

「表 40-1」では、圧縮ファイルと他の Storage Foundation コマンドや基本オペレーティングシステムコマンドとの相互関係について説明します。

表 40-1

コマンド	圧縮ファイルとの相互関係
df	df コマンドはファイルシステムによって使用されている実際のブロックを示します。この数値には圧縮の節約が含まれていますが、コマンドは節約を明示的には表示しません。 df(1)マニュアルページを参照してください。
du	du コマンドは通常はブロック数を使用するため、圧縮の結果を暗黙的に示します。ただし、GNU の du コマンドには、圧縮によって変更されないファイルサイズを代わりに使用するオプションがあります。 du(1)のマニュアルページを参照してください。
fsadm -S	fsadm -S compressed コマンドは、圧縮ファイルによるスペースの節約を報告します。 fsadm_vxfs(1)のマニュアルページを参照してください。
fsmap -p	fsmap コマンドを使用して、-p オプションで圧縮されたエクステントと圧縮解除されたエクステントの情報を報告できます。報告された論理サイズは圧縮解除されたデータのサイズであり、報告されたエクステントサイズはディスク上の圧縮されたデータのサイズです。圧縮されたエクステントでは、2 つのサイズが異なる場合があります。 fsmap(1)のマニュアルページを参照してください。
ls -l ls -s	統計呼び出しによって報告される i ノードのサイズは、ls -l コマンドで示すように論理サイズです。このサイズは圧縮によって影響されません。つまり、ブロック数には使用される実際のブロックが反映されます。したがって、ls -s コマンドは圧縮の結果を示します。 ls(1)のマニュアルページを参照してください。
vxdump	vxdump コマンドが実行されると、圧縮されたエクステントを圧縮解除します。つまり、圧縮はバックアップまたはリストア操作全体に渡って保存されないことを意味します。
vxquota	クォータの使用状況は、圧縮によって節約された領域に基づいて減少します。 vxquota(1M)のマニュアルページを参照してください。

圧縮ファイルと他の機能の相互関係

「表 40-2」では、圧縮ファイルと他の Storage Foundation 機能との相互関係について説明します。

表 40-2

機能	圧縮ファイルとの相互関係
CDS（Cross-Platform Data Sharing）	圧縮をサポートするプラットフォームから圧縮をサポートしないプラットフォームに対してディスクまたはファイルシステムを変換する場合、そのファイルシステムに圧縮ファイルが含まれていると、`fscdsconv` コマンドを実行したときに、CDS 制限に違反するファイルが存在することが通知され、変換を続行するかどうかを確認するメッセージが表示されます。続行すると、変換は正常に完了しますが、新しいプラットフォームで圧縮ファイルにアクセスすることはできません。
FCL（File Change Log）	File Change Log 機能では、ファイルの圧縮と圧縮解除は検出されません。
共有エクステント（FileSnap と重複排除）	共有エクステントは圧縮されません。 圧縮ファイルは `vxfilesnap` コマンドを使用して共有できますが、それらのファイルにアクセスするときのパフォーマンスに影響します。
SmartTier	SmartTier 機能は圧縮をサポートしません。配置ポリシーは、既存の圧縮されたエクステントを移動できません。新しく割り当てられた圧縮済みのエクステントは、既存の配置ポリシーに従います。
領域の予約（`setext -r`）	ファイルが圧縮されると、`setext -r` コマンドによってファイルの終端を越えて予約された領域は破棄され、ファイルが圧縮解除されるときにリストアされません。`setext -r` コマンドは、圧縮されたファイルの領域を予約するためには使用できません。
Storage Checkpoint	ファイルシステムに圧縮ファイルが含まれ、そのファイルシステムの Storage Checkpoint を作成する場合、それらのファイルには通常は Storage Checkpoint を介してアクセスできます。ただし、マウント済みの Storage Checkpoint にすでにあるファイルは、圧縮または圧縮解除できません。

圧縮ファイルとアプリケーションの相互関係

通常は、圧縮されたエクステントへの読み取りと書き込みが圧縮解除されたエクステントへの読み取りと書き込みに比べて遅くても、アプリケーションでは圧縮されたファイルと圧縮解除されたファイルの違いを認識できません。アプリケーションが圧縮ファイルを読み

取るとき、CPU 負荷が高まることを避けるため、ファイルシステムは通常は必要とされる readahead を実行しません。ただし、プライマリファイルセットからの読み取りの場合は、ファイルシステムが圧縮ブロック全体（デフォルトでは 1 MB）を圧縮解除し、これらのページはページキャッシュに残されます。このため、圧縮ブロックの境界を超えると、ファイルの順次読み取り操作では通常は余分なアクセスの損失が発生します。Storage Checkpoint のファイルを読み取る場合は、状況は異なります。この場合、実際に要求されるデータ以上のページキャッシュは行われません。Storage Checkpoint でアクセスする圧縮ファイルの読み取りパフォーマンスを最適化するには、アプリケーションは圧縮ブロックサイズと同じサイズで読み取る必要があります。

圧縮されたエクステントに書き込むときには、書き込みによってエクステントが圧縮解除されるため、新しく圧縮解除されたエクステントに十分なディスク領域とディスクのクォータ制限があることを確認してください。十分なディスク領域がない場合、書き込みは失敗して ENOSPC エラーが表示されます。十分なディスククォータがない場合、書き込みは失敗して EDQUOT エラーが表示されます。

tar、cpio、cp、vi のように、圧縮ファイルからデータを読み込んでファイルを他の場所にコピーするアプリケーションでは、圧縮を新しいデータに保存しません。バックアッププログラムでも同様です。

名前空間を介してファイルデータを読み込むバックアッププログラムでは、ファイルが圧縮済みであることを認識しません。バックアッププログラムは圧縮解除されたデータを受信し、圧縮は失われます。

NetBackup の FlashBackup 機能をファイル圧縮機能と併用することはできません。FlashBackup では、ディスクレイアウトバージョン 8 と 9 はサポートされません。

ファイル圧縮の使用例

次のリストは一般的な使用例のカテゴリが含まれます。

- 「圧縮ファイルとデータベース」
- 「特定の条件を満たすすべてのファイルの圧縮」

圧縮ファイルとデータベース

ファイルの圧縮は、データベース環境のストレージコストの削減をサポートします。Oracle データベースでは、アーカイブログ、パーティションに分割されたテーブル、低い頻度でアクセスされる表領域とデータファイルのストレージコストを削減する場合に、圧縮は優れた価値を提供します。データベースファイルの圧縮比は、データファイルに保存されるオブジェクトのタイプによって決まります。従来 Oracle は、TABLE と INDEX をデータファイルに保存しており、TABLE および INDEX に格納されるキーのタイプに関連付けられる列のタイプに応じて、圧縮比が検証されます。また、Oracle には、TABLE 内にある XML、表計算、MS Word 文書、写真など非構造型のデータを Secured Files 機能を使用して保存する機能もあります。これらのタイプの非構造型データは、圧縮に非常に適

した候補です。アーカイブログは最大 90%、Oracle データファイルとインデックスは約 50% から 65% の圧縮を達成できます。

Oracle データベースファイルは、データベースが有効であれば、必要に応じて圧縮および圧縮解除できます。ただし、データベースのパフォーマンスに重大な影響を与えます。I/O 応答時間が減少するだけでなく、Oracle データベースがオンラインになっており、ファイルに対するトランザクションをアクティブに実行している間に、圧縮はシームレスに実行されます。圧縮は、ダイレクト I/O、非同期 I/O、同時 I/O、ODM、Cached ODM など、高度な I/O の方法でシームレスに動作します。データファイルに更新や新しい挿入があると、書き込みに関連する部分のファイルが圧縮解除されます。クエリーはメモリ内の圧縮されたデータを取得し、ファイルは圧縮解除された状態のままになります。

メモ: DBA ユーザーとして `vxcompress` コマンドを実行できます。

次の使用例はデータベースに適用されます。

- 「サポートされるデータベースバージョンと環境」
- 「アーカイブログの圧縮」
- 「読み取り専用表領域の圧縮」
- 「アクセス頻度が低いテーブルパーティションの圧縮」
- 「アクセス頻度が低いデータファイルの圧縮」
- 「Oracle データベースのファイルの圧縮の推奨設定」

サポートされるデータベースバージョンと環境

SF (Storage Foundation) 、SFHA (Storage Foundation and High Availability) 、SF Oracle RAC (Storage Foundation for Oracle RAC) 、SFCFSHA (Storage Foundation Cluster File System High Availability) で圧縮がサポートされます。SF Oracle RAC、SFCFSHA などのクラスタ環境では、Veritas は負荷が最小のノードでファイルを圧縮することを推奨します。高速フェールオーバーの SFCFSHA 環境では、Veritas はデータベースがオフラインになっているパッシブノードでファイルを圧縮することを推奨します

アーカイブログの圧縮

アーカイブログは、データベースのリカバリに必要な重要ファイルです。ビジー状態の OLTP（オンライントランザクション処理）データベースでは、毎日アーカイブログが数 GB 生成されます。企業のガイドラインでは、アーカイブログを数日間保存するように指示することが多いです。Oracle のアーカイブログは読み取り専用ファイルで、生成された後に更新されることはありません。リカバリの間に、Oracle はアーカイブログを順番に読み取ります。したがって、アーカイブログは非常に圧縮に適した候補者です。また、アーカイブログは高圧縮性です。

次の例の手順では、1 日以上前のアーカイブログをすべて圧縮します。

1 日以上前のアーカイブログをすべて圧縮するには

1 Oracle DBA として次のクエリーを実行し、アーカイブログの場所を取得します。

```
SQL> select destination from v$archive_dest where status = 'VALID'
and valid_now = 'YES';
```

アーカイブログの送信先として /oraarch/MYDB を想定します。

2 1 日以上前のアーカイブログをすべて圧縮します。

```
$ find /oraarch/MYDB -mtime +1 -exec /opt/VRTS/bin/vxcompress {}
 ¥;
```

この手順は、cron のようなスケジューラを使用して毎日実行できます。

読み取り専用表領域の圧縮

大規模のデータベース環境では、読み取り専用のモードで変更がない静的な表領域を維持する一般的な方法です。読み取り専用表領域の主な目的は、大規模データベースの静的な部分のバックアップとリカバリを実行する必要性を取り除くことです。また、ユーザーが履歴データを変更できないように、履歴データを保護する方法も提供します。表領域を読み取り専用にすると、表領域に存在するすべてのテーブルとオブジェクトが、ユーザー更新の権限レベルに関係なく、更新されなくなります。これらのタイプの読み取り専用表領域は、圧縮に非常に適した候補です。月末レポートなどの例では、これらの読み取り専用表領域に対して実行される大きいクエリーがある場合があります。レポートをより迅速に実行するには、月次レポートを実行する前に、表領域をオンデマンドで圧縮解除できます。

次の例では、スポーツ用品の会社で、インベントリが winter_items と summer_items という 2 つの表領域に分割されています。春シーズンが終わると winter_item 表領域を圧縮し、summer_item 表領域を圧縮解除できます。夏シーズンの終わりには逆に処理できます。次の例の手順で、これらのタスクを実行します。

季節ごとに表領域を圧縮または圧縮解除するには

1 SQL を使用して、各表領域のファイルのリストを取得し、その結果を summer_files ファイルと winter_files ファイルに保存します。

```
SQL> select file_name from dba_data_files where
tablespace_name = 'WINTER_ITEM';
```

結果を winter_files ファイルに保存します。

```
SQL> select  file_name from dba_data_files where
tablespace_name = 'SUMMER_ITEM';
```

結果を summer_files ファイルに保存します。

2 winter_files ファイルを圧縮します。

```
$ /opt/VRTS/bin/vxcompress `/bin/cat winter_files`
```

3 summer_files ファイルを圧縮解除します。

```
$ /opt/VRTS/bin/vxcompress -u `/bin/cat summer_files`
```

アクセス頻度が低いテーブルパーティションの圧縮

テーブルのパーティション分割は、大規模の Oracle データベースで頻繁に使用される機能です。テーブルのパーティション分割は、パラレルクエリーを使用するトランザクションを並列化できるため、データベースのクエリーと更新の効率が向上します。また、データベースのメンテナンスを簡単にし、テーブルの可用性を向上させます。パーティションがダウンしている場合、テーブルの対応する部分のみがオフライン状態になり、テーブルの残りの部分がオンライン状態のままになります。通信環境では、一般的なのは「call_details」テーブルを月単位または四半期単位でパーティションに分割する方法です。パーティションの内容は、パーティションが古くなるにつれてあまり使用されなくなります。新しいパーティションに新しいレコードが追加され、以前の四半期のレコードは更新されません。通常は通信データベースは非常に大きいため、去年のデータを圧縮すると大幅に節約されます。

次の例では、テーブル「CALL_DETAIL」は四半期単位でパーティションに分割され、パーティションの名前は CALL_2010_Q1、CALL_2010_Q2、CALL_2011_Q1 などのように想定されます。2011 年の第 1 四半期では、CALL_2010_Q1 のデータを圧縮できます。

CALL_2010_Q1 パーティションを圧縮するには

1 CALL_2010_Q1 パーティションに属するファイル名を取得するには、SQL を使用します。

```
SQL> select tablespace_name from dba_tab_partitions
where table_name = 'CALL_DETAIL' and partition_name =
'CALL_2010_Q1';
```

クエリーが「TBS_2010_Q1」を返すことを想定します。

2 my_compress_files ファイルに名前を保存します。

```
SQL> select file_name from dba_data_files where
tablespace_name = 'TBS_2010_Q1';
```

結果を my_compress_files ファイルに保存します。

3 ファイルを圧縮します。

```
$ /opt/VRTS/bin/vxcompress `/bin/cat my_compress_files`
```

アクセス頻度が低いデータファイルの圧縮

多くの顧客データベースでは、Oracle のパーティション分割機能は使用されません。パーティション分割が使用されない場合、あまり使用されないデータファイルを特定するために Oracle カタログのクエリーを使用できます。カタログのテーブルを定期的にクエリーし、最も使用頻度の低いデータファイルを特定して、次の例の手順に示すようにそれらのファイルを圧縮します。

最も使用頻度の低いデータファイルを特定して圧縮するには

1 v$filestat をクエリーし、最も使用頻度の低いデータファイルを特定します。

```
SQL> select  name, phyrds + phywrts 'TOT_IO' from v$datafile d
and v$filestat f where d.file# = f.file# order by TOT_IO;
```

2 最も I/O 負荷が低いファイルをレポートから選択し、それらのファイルを圧縮します。

```
$ /opt/VRTS/bin/vxcompress file1 file2 file3 ...
```

3 定期的にクエリーを何度も実行して、圧縮ファイルの I/O 負荷が増加しないように確認します。I/O 負荷が増加する場合、ファイルを圧縮解除します。

```
$ /opt/VRTS/bin/vxcompress -u file1 file2 file3 ...
```

Oracle データベースのファイルの圧縮の推奨設定

ファイルが圧縮されているときにOracleデータベースがエラーなしで動作する場合でも、圧縮ファイルのI/O増加は、データベースパフォーマンスを低下させます。Oracleデータファイルの圧縮には次のガイドラインを使用します。

- データベース制御ファイルを圧縮しないでください。
- 一時表領域に属するファイルを圧縮しないでください。
- システム表領域とSYSAUX表領域に属するファイルを圧縮しないでください。
- 圧縮ファイルのI/O負荷を定期的に監視し、I/O負荷が増加したらファイルを圧縮解除してください。

特定の条件を満たすすべてのファイルの圧縮

特定の条件を満たすすべてのファイルを検索して、それらのファイルすべてを圧縮するためにvxcompressコマンドに検索結果を組み合わせます。次の例では、30日以上変更されなかった/mntのすべてのファイルを圧縮します。

```
$ find /mnt -mtime +30 | xargs /opt/VRTS/bin/vxcompress
```

10

ストレージの管理

41

ボリュームとディスクグループの管理

この章では以下の項目について説明しています。

デフォルトのディスクグループの名前の付け方

-g オプションを使える Veritas Volume Manager（VxVM）コマンドには、このオプションを使ってディスクグループを指定してください。ディスクグループを指定しない場合、VxVM は次の順序でルールを適用してディスクグループの名前を決定します。

- 環境変数 VXVM_DEFAULTDG で指定されたデフォルトのディスクグループ名を使います。この変数を、システム全体で予約済みのディスクグループ名、bootdg、defaultdg、nodg のいずれかに設定することもできます。
  p.946 の「システム全体のブートディスクグループの表示」を参照してください。
  この変数が定義されていない場合には、次のルールが適用されます。
- システム全体でデフォルトのディスクグループのエイリアスである defaultdg に割り当てられたディスクグループを使います。
  p.946 の「システム全体のデフォルトのディスクグループの表示と指定」を参照してください。
  このエイリアスが設定されていない場合には、次のルールが適用されます。
- ディスクアクセスレコードの編集操作など、ディスクグループ名が不明でも操作を実行できる場合は、そのまま操作を実行します。

前述のルールが 1 つも該当しない場合、要求した操作は実行できません。

警告: VxVM 4.0 より前のリリースでは、あるコマンドの操作対象となっているオブジェクト名を検索して、そのディスクグループの決定を試みるコマンド群がありました。4.0 以上ではこの機能はサポートされません。オブジェクト名からのディスクグループの決定に依存しているスクリプトを実行すると、失敗することがあります。

システム全体のブートディスクグループの表示

現在定義されているシステム全体のブートディスクグループを表示するには、次のコマンドを入力します。

```
# vxdg bootdg
```

vxdg（1M）マニュアルページを参照してください。

システム全体のデフォルトのディスクグループの表示と指定

Veritas Volume Manager（VxVM）でシステム全体のデフォルトのディスクグループを定義できます。

現在定義されているシステム全体のデフォルトのディスクグループを表示するには、次のコマンドを入力します。

```
#  vxdg defaultdg
```

デフォルトのディスクグループが定義されていない場合は、nodg と表示されます。

vxdg(1M)マニュアルページを参照してください。

また、次のコマンドを使ってデフォルトのディスクグループを表示することもできます。

```
# vxprint -Gng defaultdg 2>/dev/null
```

この場合には、デフォルトのディスクグループが定義されていないと、何も表示されません。

vxprint(1M)マニュアルページを参照してください。

defaultdg によってエイリアスが設定されているディスクグループの名前を指定するには、次のコマンドを使います。

```
# vxdctl defaultdg diskgroup
```

diskgroup は次のいずれかになります。

- 指定したディスクグループ名。
  指定したディスクグループが、システム上に存在する必要はありません。
- bootdg
  デフォルトのディスクグループを現在定義されたシステム全体のブートディスクグループと同じに設定します。
- nodg
  デフォルトのディスクグループが未定義であることを指定します。

vxdctl(1M)マニュアルページを参照してください。

ボリュームまたはディスクの移動

ここでは、ボリュームまたはディスクの移動について説明します。

VxVM ディスクからのボリュームの退避

ディスクを無効化または削除する前に、そのディスクから、システム上の十分な領域がある別のディスクへデータを退避できます。

ディスクからボリュームを移動するには、次の手順を実行します。

1. vxdiskadm メインメニューから、[ディスクからのボリュームの移動(Move volumes from a disk)]を選択します。
2. 次のプロンプトで、移動するボリュームのあるディスクのディスク名を次のように入力します。

```
Enter disk name [<disk>,list,q,?] mydg01
```

次の表示画面で、オプションとしてボリュームの移動先ディスクの一覧を指定できます。プロンプトで、次のいずれかを実行します。

- Enter キーを押してディスクグループの使用可能な領域にボリュームを移動します。
- 次のように、使う必要があるディスクグループ内のディスクを指定します。

```
Enter disks [<disk ...>,list]
VxVM NOTICE V-5-2-283 Requested operation is to move all
volumes from disk mydg01 in group mydg.

NOTE: This operation can take a long time to complete.

Continue with operation? [y,n,q,?] (default: y)
```

ボリュームがディスクから移動されると、vxdiskadm プログラムによって操作の状態が表示されます。

```
VxVM vxevac INFO V-5-2-24 Move volume voltest ...
```

ボリュームがすべて移動されると、vxdiskadm プログラムによって次のメッセージが表示されます。

```
VxVM INFO V-5-2-188 Evacuation of disk mydg02 is complete.
```

3 次のプロンプトで、ボリュームを他のディスクから移動する(y)か、vxdiskadm メインメニューに戻る(n)かを指定します。

```
Move volumes from another disk? [y,n,q,?] (default: n)
```

ディスクグループ間のディスク移動

未使用のディスクをディスクグループ間で移動するには、ディスクをあるディスクグループから削除して、そのディスクを他のディスクグループに追加します。たとえば、(salesdg04というディスク名で接続されている)物理ディスク sdc を、ディスクグループ salesdg から移動して、ディスクグループ mktdg に追加するには、次のコマンドを使います。

```
# vxdg -g salesdg rmdisk salesdg04
# vxdg -g mktdg adddisk mktdg02=sdc
```

警告: この手順では、ディスク上の設定情報もデータも保持されません。

また、vxdiskadmコマンドを使ってディスクを移動することもできます。メインメニューから[ディスクの削除(Remove a disk)]を選択してから、[ディスクの追加または初期化(Add or initialize a disk)]を選択します。

ディスクを移動し、これらのディスク上のデータをボリュームなどの VxVM オブジェクトと共に保持するには

p.956 の「ディスクグループ間のオブジェクト移動」を参照してください。

ディスクグループの内容の再編成

次のような場合には、既存のディスクグループの内容を再編成することができます。

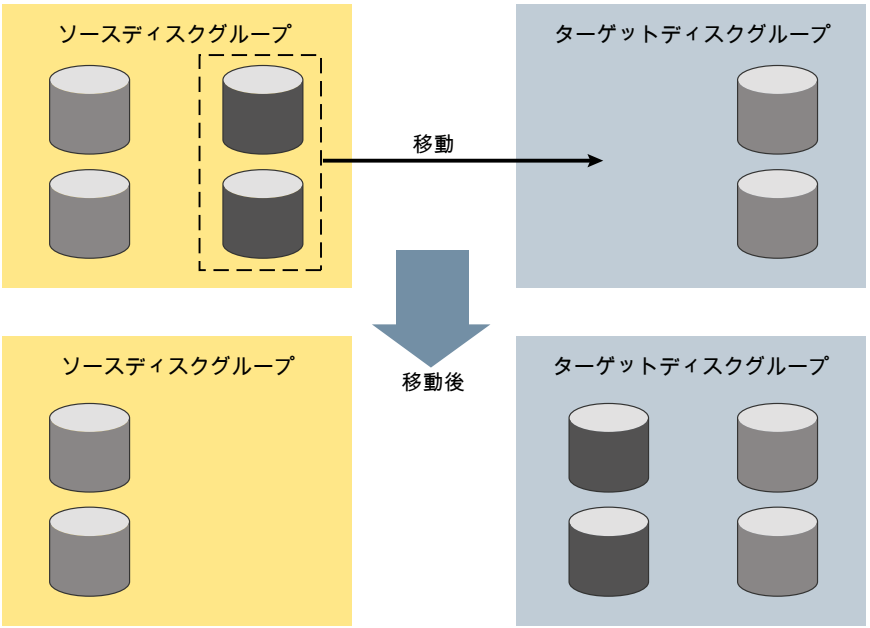
- 組織のニーズの変化に応じて、ボリュームやディスクのグループを変更する。たとえば、部門の境界に合わせてディスクグループを分割したり、部門の合併に伴いディスクグループを結合することができます。
- ディスクグループからボリュームまたはディスクを切り離し、同じホストまたは別のホストで個々に処理する。これにより、バックアップまたは意思決定支援システムのために、オフホスト処理を実装できます。
- プライベートリージョンの空き領域がほとんどなくなった場合に、ディスクグループを統合して設定データベースのサイズを縮小する。これは、プライベートリージョンを増やすよりはるかに簡単な解決法です。
- オンラインメンテナンスとアップグレードの目的で分割し、再結合できる耐障害性のあるシステムで、オンラインメンテナンスとアップグレードを実行する。

ディスクグループを再編成するには、vxdg コマンドを使います。

vxdg コマンドでは、ディスクグループの再編成に必要な次の操作を実行できます。

- move 操作は、インポート済みディスクグループ間で、ひとまとまりの自己完結型 VxVM オブジェクトを移動します。この操作によってソースディスクグループのディスクがすべて削除される場合、操作は失敗します。ボリュームの状態は移動後も保持されます。
  図 41-1 に、move 操作を示します。

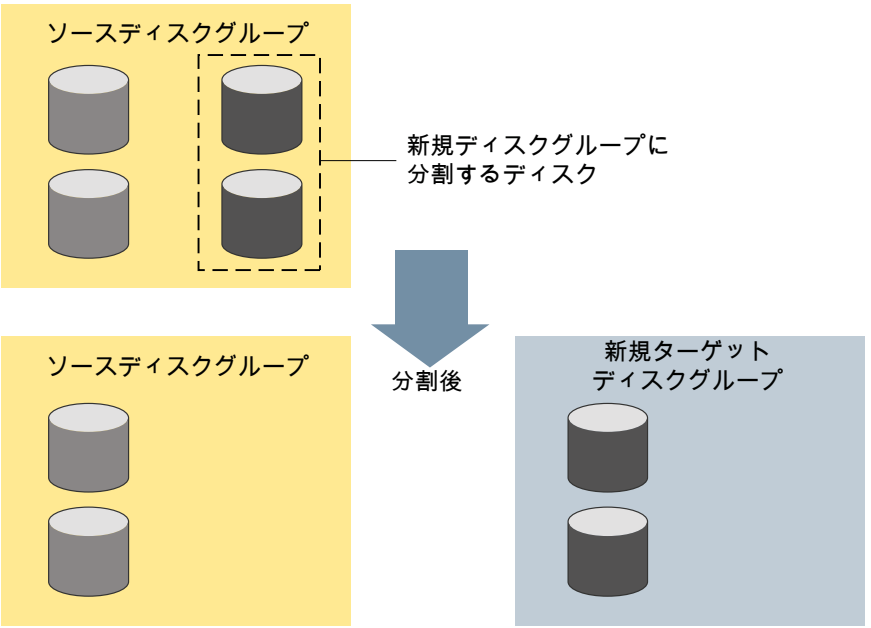
図 41-1 ディスクグループの移動操作



- `split` 操作は、インポート済みディスクグループからひとまとまりの自己完結型 VxVM オブジェクトを分割し、新しく作成したターゲットディスクグループに移動します。この操作によってソースディスクグループのディスクがすべて削除される場合またはインポート済みディスクグループの名前がターゲットディスクグループと同一である場合、操作は失敗します。

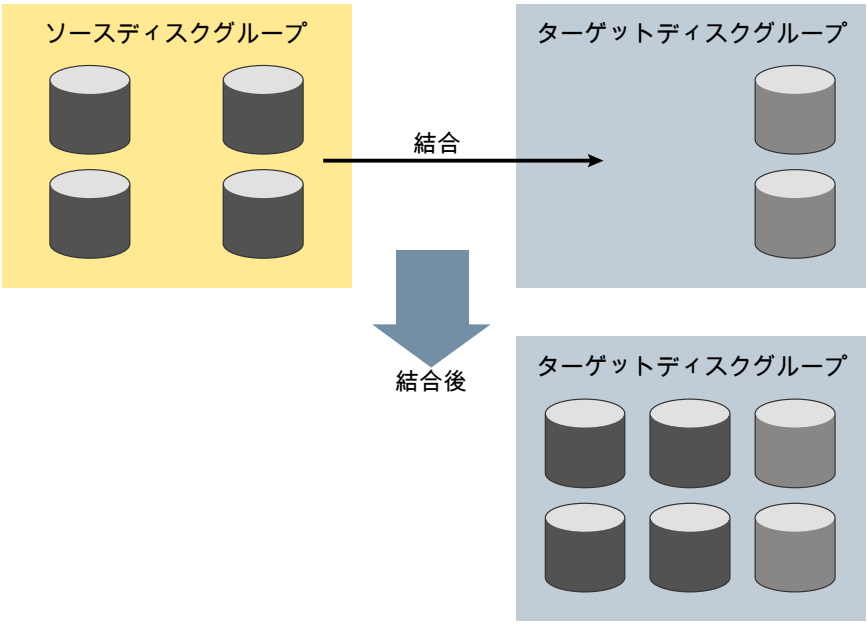
図 41-2 に、`split` 操作を示します。

図 41-2 ディスクグループの分割操作



- `join` 操作は、インポート済みディスクグループから VxVM オブジェクトをすべて削除し、インポート済みターゲットディスクグループに結合します。ソースディスクグループは、結合が完了すると削除されます。
  図 41-3 に、`join` 操作を示します。

図 41-3 ディスクグループの結合操作



これらの操作はディスクやトップレベルのボリュームなどの VxVM オブジェクトで実行され、サブボリューム、プレックス、サブディスクといったすべてのコンポーネントオブジェクトが対象となります。移動するオブジェクトは、自己完結型のオブジェクトである必要があります。つまり、移動するディスクに移動対象外のオブジェクトが含まれないようにする必要があります。

サイトの一貫性が設定されたディスクグループの場合、移動される VxVM オブジェクトが移動の後でサイトの一貫性の条件を満たさない場合は、いずれの移動操作(移動、分割、結合)も失敗します。たとえば、移動されるボリュームが、移動先のディスクグループで設定されているいずれかのサイトのプレックスを持たない場合があります。このようなボリュームは、移動先ディスクグループでの allsites フラグの条件を満たしません。オブジェクトの allsites フラグをオフにすることによって操作を成功させるには、-f(force)オプションを使います。

移動するディスクを 1 つ以上指定すると、それらのディスク上の VxVM オブジェクトがすべて移動されます。`-o expand` オプションを使うと、指定したオブジェクトが設定されているディスクがすべて `vxdg` によって移動されます。常に期待どおりの結果になるとは限らないため、この操作を実行する場合は注意が必要です。`vxdg` の `listmove` 操作を使うと、指定した一連のオブジェクトに対応するひとまとまりの自己完結型オブジェクトの確認に役立つ情報を取得することができます。

警告: ディスクグループ間でボリュームを移動する前に、そのボリュームにアクセスしているアプリケーションをすべて停止し、ボリュームに設定されているすべてのファイルシステムのマウントを解除する必要があります。

システムがクラッシュするかハードウェアのサブシステムに障害が発生した場合、システムの再起動時またはハードウェアのサブシステムの修復時に、VxVM は未完了のディスクグループの再設定を完了するか、元に戻そうと試みます。どちらの処理が実行されるかは、再設定の進行状況によって決まります。別のホストにインポートされたため、またはすでに存在しないためにディスクグループの 1 つが使えなくなっている場合は、手作業でディスクグループのリカバリを行う必要があります。

詳しくは、『Veritas InfoScale トラブルシューティングガイド』を参照してください。

ディスクグループの分割と結合に関する制限

ディスクグループの分割および結合機能には、次の制限があります。

- 移動、分割または結合に関係するディスクグループは、バージョン 90 またはそれ以上のバージョンである必要があります。
  p.983 の「ディスクグループバージョンのアップグレード」を参照してください。
- 再設定は、物理ディスクを示す整数に基づいて実行されます。
- 移動するオブジェクトには、起動しているボリュームが含まれないようにしてください。
- CDS 互換ディスクグループと非 CDS ディスクグループ間でディスクを移動することはできません。
- デフォルトでは、VxVM は自動的にリカバリし、ディスクグループの移動、分割、結合に続いてボリュームを起動します。自動リカバリ機能をオフにした場合、ボリュームは移動、分割、結合の後に無効になります。ボリュームを修復して再起動するには、`vxrecover -m` および `vxvol startall` コマンドを使います。
  p.991 の「ボリュームの自動リカバリの設定」を参照してください。
- 永続 FastResync の関連付けが解除されたデータ変更オブジェクト(DCO)とスナップオブジェクトを、ディスクグループ間で移動することはできません。
- Veritas Volume Replicator(VVR)オブジェクトをディスクグループ間で移動することはできません。
- ディスクグループの移動を問題なく終了するには、移動後に設定データベースのコピーを格納できるディスクが、ソースディスクグループに少なくとも 1 つ含まれている必要があります。
- ディスクグループの分割を問題なく終了するには、分割後に設定データベースのコピーを格納できるディスクが、ソースディスクグループとターゲットディスクグループの両方に少なくとも 1 つ含まれている必要があります。

- ディスクグループの移動または結合を問題なく終了するには、移動後のディスクグループ内にあるすべてのオブジェクトに関する情報がターゲットディスクグループの設定データベースに格納される必要があります。
- ボリュームを別のディスクグループに分割したり、別のディスクグループに移動すると、ボリュームレコード ID が変更されます。
- この操作は、ソースディスクグループまたはターゲットディスクグループが共有ディスクグループである場合に、クラスタのマスターモードでのみ実行できます。
- クラスタ環境では、移動または結合に関係するディスクグループは、両方とも専有または両方とも共有である必要があります。
- 分割または移動するキャッシュオブジェクトまたはボリュームが ISP ボリュームを使っている場合は、それらのボリュームを含むストレージプールも指定する必要があります。

移動により影響を受ける可能性のあるオブジェクトの一覧表示

リスト形式でオブジェクトを指定して、移動される VxVM オブジェクトを表示するには、次のコマンドを使います。

```
# vxdg [-o expand] listmove sourcedg targetdg object ...
```

次の例では、ボリューム vol1 をディスクグループ mydg から newdg に移動することによって影響を受けるオブジェクトが一覧表示されます。

```
# vxdg listmove mydg newdg vol1
mydg01 sda mydg05 sde vol1 vol1-01 vol1-02 mydg01-01 mydg05-01
```

ただし、次のコマンドを実行すると、ディスク mydg01 に設定されているのはボリューム vol1 の一部にすぎないためエラーが起きます。

```
# vxdg listmove mydg newdg mydg01
VxVM vxdg ERROR V-5-2-4597 vxdg listmove mydg newdg failed
VxVM vxdg ERROR V-5-2-3091 mydg05 : Disk not moving, but
subdisks on it are
```

次のように -o expand オプションを指定すると、移動するオブジェクトの一覧に、vol1 に設定される他のディスク(この場合は mydg05)も含まれるようになります。

```
# vxdg -o expand listmove mydg newdg mydg01
mydg01 sda mydg05 sde vol1 vol1-01 vol1-02 mydg01-01
mydg05-01
```

ディスクグループ間の DCO ボリュームの移動

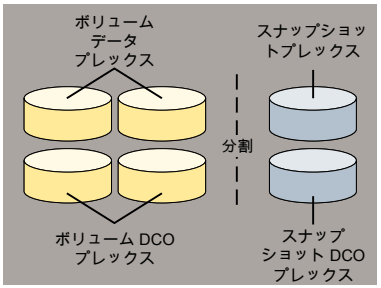
親ボリューム（スナップショットボリュームなど）を別のディスクグループに移動する場合は、DCO ボリュームも親ボリュームに付随している必要があります。vxassist addlog、vxmake または vxdco コマンドを使って DCO を設定した場合は、親ボリュームの移動時に、DCO ボリュームのプレックスを含むディスクが親ボリュームに付随しているかどうかを確認する必要があります。vxprint コマンドを使って、ボリュームに関連付けられている DCO ボリュームの設定情報を確認してください。

vxassist コマンドを使ってボリュームとその DCO を作成するか、または vxsnap prepare コマンドを使ってボリュームに DCO を追加した場合、DCO プレックスは自動的に親ボリュームのデータプレックスとは異なるディスク上に配置されます。以前のリリースでは、バージョン 0 の DCO プレックスは、ディスクグループの分割または移動操作を実行する際の利便性を高めるため、データプレックスと同じディスク上に配置されていました。バージョン 20 の DCO の場合は、永続 FastResync に加えて DRL（dirty region logging）もサポートしているため、DCO プレックスはデータプレックスと切り離しておくことをお勧めします。これにより、ボリュームに対する I/O の処理効率が向上し、DRL ログに障害許容力が備わります。

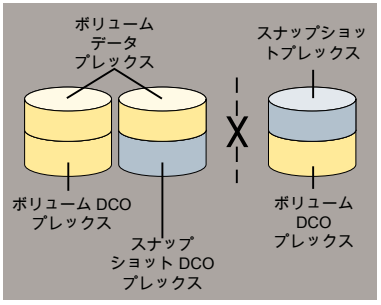
図 41-4 に、ディスクグループのディスク上の DCO プレックスの位置が原因でディスクグループを分割できないケースを示します。

p.87 の「ボリュームスナップショット」を参照してください。

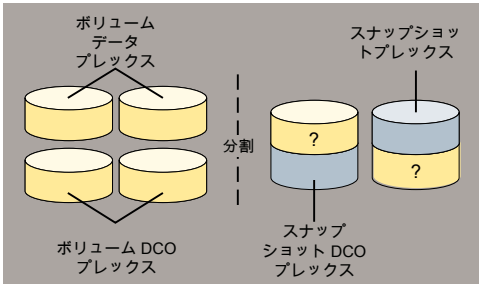
図 41-4 分割できるディスクグループと分割できないディスクグループの例



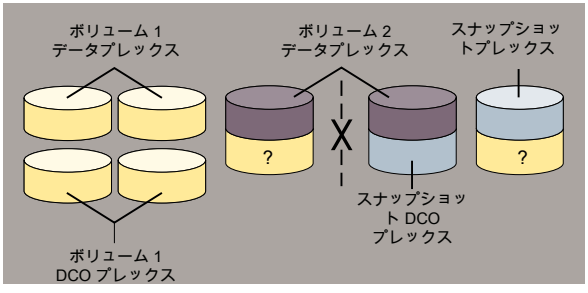
DCO プレックスが専用のディスク上にあり、ボリュームデータを含むディスクに付随して移動できるため、このディスクグループは分割できる。



DCO プレックスをボリュームに付随して移動できないため、このディスクグループは分割できない。1 つの解決策は、DCO プレックスを再配置すること。この例では、再配置が必要な DCO プレックスを スワップする際に、ディスクグループ内の別のディスクを媒介として使う。あるいは、DRL の処理効率と障害許容力を 向上させるために、DCO プレックスを 専用のディスクに割り当てる。



DCO プレックスをボリュームに付随して移動できるため、このディスクグループは分割できる。ただし、「?」マークの付いたディスク部分のデータも同時に移動されるため、あまり望ましくない。



このディスクグループを分割するとボリューム 2 のデータプレックスを格納しているディスクが切り離されるため、このディスクグループは分割できない。考えられる解決策は、スナップショットプレックスを格納しているディスク、または移動できる別の適切なディスクにスナップショット DCO プレックスを再配置すること。

ディスクグループ間のオブジェクト移動

一連の自己完結型 VxVM オブジェクトを、インポート済みソースディスクグループからインポート済みターゲットディスクグループに移動するには、次のコマンドを使います。

```
# vxdg [-o expand] [-o override|verify] move sourcedgtargetdg ¥
  object ...
```

-o expand オプションを使うと、指定したオブジェクトまたはそれらが格納するオブジェクトに関連付けられているサブディスクを格納する他のディスクもすべて移動対象になります。

EMC 社のアレイ内のライセンスされているディスクを移動する場合、vxdg は、移動に関係する各ディスクについてデフォルトで EMC ディスクの互換性検査を実行します。互換性検査が問題なく終了すると、移動処理が実行されます。その後、vxdg が再び検査を行い、互換性検査の実行後に設定が変更されていないことを確認します。設定が変更されていた場合は、vxdg によって移動処理全体が再度実行されます。

メモ: -o override オプションと -o verify オプションは、EMC アレイを有効な timefinder ライセンスで使っている場合にのみ使ってください。これらのいずれかのオプションを指定したときに、アレイとライセンスの必要条件を満たしていない場合は、警告メッセージが表示され、操作は無視されます。

-o override オプションを指定すると、この検査をまったく行わずに移動処理を実行することができます。

-o verify オプションを指定すると、移動するディスクのアクセス名が返されます。移動処理は実行されません。

次に示す vxprint の出力は、ディスクグループ rootdg と mydg の内容を示しています。

出力は 2 つのユーティリティフィールド、TUTIL0 と PUTIL0 を含んでいます。VxVM は異なるコマンドと Veritas InfoScale 製品の間のオブジェクトと通信を管理するためにこれらのフィールドを作成します。TUTIL0 の値は一時的です。これらは再ブートで維持されません。PUTIL0 の値は永続的です。これらは再ブートで維持されます。

```
# vxprint
Disk group: rootdg
TY NAME         ASSOC      KSTATE   LENGTH     PLOFFS   STATE   TUTIL0   PUTIL0
dg rootdg       rootdg     -        -          -        -       -        -
dm rootdg02     sdb        -        17678493   -        -       -        -
dm rootdg03     sdc        -        17678493   -        -       -        -
dm rootdg04     csdd       -        17678493   -        -       -        -
dm rootdg06     sdf        -        17678493   -        -       -        -

Disk group: mydg
TY NAME         ASSOC      KSTATE   LENGTH     PLOFFS   STATE   TUTIL0   PUTIL0
dg mydg         mydg       -        -          -        -       -        -
dm mydg01       sda        -        17678493   -        -       -        -
```

```
dm mydg05       sde         -         17678493   -         -        -        -
dm mydg07       sdg         -         17678493   -         -        -        -
dm mydg08       sdh         -         17678493   -         -        -        -
v  vol1         fsgen       ENABLED   2048       -         ACTIVE   -        -
pl  vol1-01     vol1        ENABLED   3591       -         ACTIVE   -        -
sd mydg01-01    vol1-01     ENABLED   3591       0         -        -        -
pl vol1-02      vol1        ENABLED   3591       -         ACTIVE   -        -
sd mydg05-01    vol1-02     ENABLED   3591       0         -        -        -
```

次のコマンドは、ディスク mydg01 を指定することによって間接的に指定される一連の自己完結型オブジェクトを、ディスクグループ mydg から rootdg に移動します。

```
# vxdg -o expand move mydg rootdg mydg01
```

デフォルトでは、VxVM は自動的にリカバリし、ディスクグループの移動に続いてボリュームを起動します。自動リカバリ機能をオフにした場合、ボリュームは移動後に無効になります。次のコマンドを使って、ターゲットディスクグループ内のボリュームのリカバリおよび再起動を行います。

```
# vxrecover -g targetdg -m [volume ...]
# vxvol -g targetdg startall
```

移動後の vxprint の出力は、mydg01 だけでなくボリューム vol1 および mydg05 も rootdg に移動され、ディスクグループ mydg 内には、mydg07 および mydg08 のみが残っていることを示しています。

```
# vxprint
Disk group: rootdg
TY NAME         ASSOC       KSTATE    LENGTH     PLOFFS    STATE    TUTIL0   PUTIL0
dg rootdg       rootdg      -         -          -         -        -        -
dm mydg01       sda         -         17678493   -         -        -        -
dm rootdg02     sdb         -         17678493   -         -        -        -
dm rootdg03     sdc         -         17678493   -         -        -        -
dm rootdg04     sdd         -         17678493   -         -        -        -
dm mydg05       sde         -         17678493   -         -        -        -
dm rootdg06     sdf         -         17678493   -         -        -        -
v  vol1         fsgen       ENABLED   2048       -         ACTIVE   -        -
pl  vol1-01     vol1        ENABLED   3591       -         ACTIVE   -        -
sd mydg01-01    vol1-01     ENABLED   3591       0         -        -        -
pl vol1-02      vol1        ENABLED   3591       -         ACTIVE   -        -
sd mydg05-01    vol1-02     ENABLED   3591       0         -        -        -

Disk group: mydg
TY NAME         ASSOC       KSTATE    LENGTH     PLOFFS    STATE    TUTIL0   PUTIL0
dg mydg         mydg        -         -          -         -        -        -
```

```
dm mydg07       sdg         -          17678493   -        -       -        -
dm mydg08       sdh         -          17678493   -        -       -        -
```

ボリュームがパスワードまたはパスフレーズで暗号化されている場合、VxVM でのボリュームの再起動時にパスワードまたはパスフレーズの入力を求めるメッセージが表示されます。

また、次のコマンドを実行しても、同じ結果を得ることができます。

```
# vxdg move mydg rootdg mydg01 mydg05
# vxdg move mydg rootdg vol1
```

p.450 の「共有ディスクグループ間のオブジェクト移動」を参照してください。

ディスクグループの分割

一連の自己完結型 VxVM オブジェクトを、インポート済み分割元のディスクグループから新しいディスクグループに分割するには、次のコマンドを使います。

```
# vxdg [-o expand] [-o override|verify] split sourcedgtargetdg ¥
  object ...
```

p.956 の「ディスクグループ間のオブジェクト移動」を参照してください。

次に示す vxprint の出力は、ディスクグループ rootdg の内容を示しています。

出力は 2 つのユーティリティフィールド、TUTIL0 と PUTIL0 を含んでいます。VxVM は異なるコマンドと Veritas InfoScale 製品の間のオブジェクトと通信を管理するためにこれらのフィールドを作成します。TUTIL0 の値は一時的です。これらは再ブートで維持されません。PUTIL0 の値は永続的です。これらは再ブートで維持されます。

```
# vxprint
Disk group: rootdg
TY NAME         ASSOC       KSTATE    LENGTH     PLOFFS    STATE    TUTIL0   PUTIL0
dg rootdg       rootdg      -         -          -         -        -        -
dm rootdg01     sda         -         17678493   -         -        -        -
dm rootdg02     sdb         -         17678493   -         -        -        -
dm rootdg03     sdc         -         17678493   -         -        -        -
dm rootdg04     sdd         -         17678493   -         -        -        -
dm rootdg05     sde         -         17678493   -         -        -        -
dm rootdg06     sdf         -         17678493   -         -        -        -
dm rootdg07     sdg         -         17678493   -         -        -        -
dm rootdg08     sdh         -         17678493   -         -        -        -
v  vol1         fsgen       ENABLED   2048       -         ACTIVE   -        -
pl  vol1-01     vol1        ENABLED   3591       -         ACTIVE   -        -
sd rootdg01-01 vol1-01     ENABLED   3591       0         -        -        -
```

```
pl vol1-02     vol1       ENABLED   3591       -         ACTIVE   -        -
sd rootdg05-01 vol1-02    ENABLED   3591       0         -        -        -
```

次のコマンドは、ディスク rootdg07 および rootdg08 を rootdg から削除して、新しいディスクグループ mydg を形成します。

```
# vxdg -o expand split rootdg mydg rootdg07 rootdg08
```

デフォルトでは、VxVM は自動的にリカバリし、ディスクグループの分割に続いてボリュームを起動します。自動リカバリ機能をオフにした場合、ボリュームは分割後に無効になります。次のコマンドを使って、ターゲットディスクグループ内のボリュームのリカバリおよび再起動を行います。

```
# vxrecover -g targetdg -m [volume ...]
# vxvol -g targetdg startall
```

ボリュームがパスワードまたはパスフレーズで暗号化されている場合、VxVM でのボリュームの再起動時にパスワードまたはパスフレーズの入力を求めるメッセージが表示されます。

分割処理後の vxprint の出力には、新しいディスクグループ mydg が表示されます。

```
# vxprint
Disk group: rootdg
TY NAME         ASSOC      KSTATE     LENGTH     PLOFFS     STATE    TUTIL0     PUTIL0
dg rootdg       rootdg     -          -          -          -        -          -
dm rootdg01     sda        -          17678493   -          -        -          -
dm rootdg02     sdb        -          17678493   -          -        -          -
dm rootdg03     sdc        -          17678493   -          -        -          -
dm rootdg04     sdd        -          17678493   -          -        -          -
dm rootdg05     sde        -          17678493   -          -        -          -
dm rootdg06     sdf        -          17678493   -          -        -          -
v vol1          fsgen      ENABLED    2048       -          ACTIVE   -          -
pl vol1-01      vol1       ENABLED    3591       -          ACTIVE   -          -
sd rootdg01-01 vol1-01     ENABLED    3591       0          -        -          -
pl vol1-02      vol1       ENABLED    3591       -          ACTIVE   -          -
sd rootdg05-01 vol1-02     ENABLED    3591       0          -        -          -
Disk group: mydg
TY NAME         ASSOC      KSTATE     LENGTH     PLOFFS     STATE    TUTIL0     PUTIL0
dg mydg         mydg       -          -          -          -        -          -
dm rootdg07     sdg        -          17678493   -          -        -          -
dm rootdg08     sdh        -          17678493   -          -        -          -
```

p.450 の「共有ディスクグループの分割」を参照してください。

ディスクグループの結合

すべての VxVM オブジェクトを、インポート済み結合操作元のディスクグループからインポート済みディスクグループに結合するには、次のコマンドを使います。

```
# vxdg [-o override|verify] join sourcedgtargetdg
```

p.956 の「ディスクグループ間のオブジェクト移動」を参照してください。

メモ: join 操作では、rootdg をソースディスクグループとして指定することはできません。

次に示す vxprint の出力は、ディスクグループ rootdg と mydg の内容を示しています。

出力は 2 つのユーティリティフィールド、TUTIL0 と PUTIL0 を含んでいます。VxVM は異なるコマンドと Veritas InfoScale 製品の間のオブジェクトと通信を管理するためにこれらのフィールドを作成します。TUTIL0 の値は一時的です。これらは再ブートで維持されません。PUTIL0 の値は永続的です。これらは再ブートで維持されます。

```
# vxprint
Disk group: rootdg
TY NAME         ASSOC       KSTATE     LENGTH     PLOFFS     STATE     TUTIL0     PUTIL0
dg rootdg       rootdg      -          -          -          -         -          -
dm rootdg01     sda         -          17678493   -          -         -          -
dm rootdg02     sdb         -          17678493   -          -         -          -
dm rootdg03     sdc         -          17678493   -          -         -          -
dm rootdg04     sdd         -          17678493   -          -         -          -
dm rootdg07     sdg         -          17678493   -          -         -          -
dm rootdg08     sdh         -          17678493   -          -         -          -

Disk group: mydg
TY NAME         ASSOC       KSTATE     LENGTH     PLOFFS     STATE     TUTIL0     PUTIL0
dg mydg         mydg        -          -          -          -         -          -
dm mydg05       sde         -          17678493   -          -         -          -
dm mydg06       sdf         -          17678493   -          -         -          -
v  vol1         fsgen       ENABLED    2048       -          ACTIVE    -          -
pl  vol1-01     vol1        ENABLED    3591       -          ACTIVE    -          -
sd mydg01-01    vol1-01     ENABLED    3591       0          -         -          -
pl vol1-02      vol1        ENABLED    3591       -          ACTIVE    -          -
sd mydg05-01    vol1-02     ENABLED    3591       0          -         -          -
```

次のコマンドは、ディスクグループ mydg を rootdg に結合します。

```
# vxdg join mydg rootdg
```

デフォルトでは、VxVM は自動的にリカバリし、ディスクグループの結合に続いてボリュームを起動します。自動リカバリ機能をオフにしてある場合は、ボリュームは結合の後で無効になります。次のコマンドを使って、ターゲットディスクグループ内のボリュームのリカバリおよび再起動を行います。

```
# vxrecover -g targetdg -m [volume ...]
# vxvol -g targetdg startall
```

ボリュームがパスワードまたはパスフレーズで暗号化されている場合、VxVM でのボリュームの再起動時にパスワードまたはパスフレーズの入力を求めるメッセージが表示されます。

結合処理後の vxprint の出力では、ディスクグループ mydg は結合され、削除されています。

```
# vxprint
Disk group: rootdg
TY NAME         ASSOC       KSTATE     LENGTH      PLOFFS     STATE    TUTIL0     PUTIL0

dg rootdg       rootdg      -          -           -          -        -          -
dm mydg01       sda         -          17678493    -          -        -          -
dm rootdg02     sdb         -          17678493    -          -        -          -
dm rootdg03     sdc         -          17678493    -          -        -          -
dm rootdg04     sdd         -          17678493    -          -        -          -
dm mydg05       sde         -          17678493    -          -        -          -
dm rootdg06     sdf         -          17678493    -          -        -          -
dm rootdg07     sdg         -          17678493    -          -        -          -
dm rootdg08     sdh         -          17678493    -          -        -          -
v  vol1         fsgen       ENABLED    2048        -          ACTIVE   -          -
pl  vol1-01     vol1        ENABLED    3591        -          ACTIVE   -          -
sd mydg01-01    vol1-01     ENABLED    3591        0          -        -          -
pl vol1-02      vol1        ENABLED    3591        -          ACTIVE   -          -
sd mydg05-01    vol1-02     ENABLED    3591        0          -        -          -
```

p.450 の「共有ディスクグループの結合」を参照してください。

タスクの監視と制御

VxVM タスクモニタは、タスクの作成、管理、完了を監視することによって、システムリカバリの進行状況を追跡します。タスクモニタによって、タスクの進行状況を監視し（たとえば、システム処理効率への影響を軽減するために）一時停止やリカバリ速度などのタスクの特性を変更できます。

メモ: VxVM は、この機能を専用ディスクグループに対してのみサポートしており、CVM 環境の共有ディスクグループに対してはサポートしていません。

システムで作成されるタスクは診断のために /etc/vx/log/ ディレクトリの tasklog ファイルに記録されます。このファイルはタスク関連のすべての操作(作成、完了、一時停止、再開および中止)のエントリをログに記録します。ログのエントリ例を次に示します。

```
# 159571211, 16905,  Thu Aug 21 02:54:18 2014
  184     -   SNAPSYNC  Starting 00.00% 0/65536/0 SNAPSYNC full1-v1
 vvrdg
```

タスクタグの指定

各タスクには一意のタスク ID が指定されています。これは、各タスクに対応する数字の ID であり、特に 1 つのタスクを識別するために vxtask ユーティリティに対して指定できます。いくつかの VxVM ユーティリティでは、-t オプションを使って、最長 16 文字の英数字のタグを指定することもできます。この機能によって、複数のタスクを同じタグで関連付け、グループ化することができます。

次のユーティリティは -t オプションを受け入れます。

- vxassist
- vxevac
- vxmirror
- vxplex
- vxrecover
- vxrelayout
- vxresize
- vxsd
- vxvol

たとえば、vxrecover コマンドを実行し、タスクタグ myrecovery を持つグループとして生じるタスクを追跡するには、次のコマンドを使います。

```
# vxrecover -g mydg -t myrecovery -b mydg05
```

結果として得られるタスクを追跡するには、次のコマンドを使います。

```
# vxtask monitor myrecovery
```

vxrecover で呼び出したユーティリティによって起動されたタスクは、タスク ID およびタスクタグを継承するため、親子のタスク関係が確立されます。

タスクタグをサポートするユーティリティについて詳しくは、各マニュアルページを参照してください。

vxtask 操作

vxtask コマンドでは次の操作をサポートします。

中止

指定のタスクを停止します。通常は、I/O エラーが発生した場合と同様に操作が取り消され、その時点までに実行された処理が可能なかぎり元に戻されます。

リスト

システムで実行されているタスクをそれぞれ 1 行にまとめて表示します。-l オプションを指定すると、タスクが長い形式で表示されます。-h オプションを指定すると、親タスクの後に子タスクというように、タスクが階層的に表示されます。デフォルトでは、システムで実行されているタスクがすべて表示されます。taskid 引数を含めた場合、出力されるタスクはその taskid またはタスクタグが taskid と一致するものに制限されます。残りの引数はタスクをフィルタ処理し、一覧表示するタスクを限定します。

このリリースでは、vxtask list コマンドは SmartMove とシン再利用操作をサポートします。

- SmartMove を使ってボリューム、プレックス、サブディスクの再同期または同期を行う場合、vxtask list では、その操作が SmartMove を使っているかどうかが表示されます。
- LUN レベルの再生では、vxtask list コマンドにより、各 LUN で実行される再生量の情報が提供されます。
- シンボリュームに対して init=zero を指定すると、シンボリュームで再生がトリガされる可能性があり、進行状況は vxtask list コマンドで参照されます。

監視

タスク情報が変更されると、タスクやタスクグループに関する情報を継続的に表示します。これで、タスクの進行状況を追跡できます。-l を指定すると、長い形式で表示されます。デフォルトでは、1 行にまとめられたリストが表示されます。タスク状態が変更されるとタスク情報が表示され、さらに、タスク完了時にも出力が生成されます。この場合、タスクの状態は EXITED と表示されます。

一時停止

実行中のタスクを一時停止し、操作を中断します。

再開

一時停止していたタスクの操作を再開します。

設定

タスクの変更可能なパラメータを変更します。現時点では、変更可能なパラメータは *slow[=iodelay]* の 1 つのみです。これを使って、コピー操作によるシステムパフォーマンスへの影響を低減できます。slow を指定すると、このような操作の間に iodelay のデフォルト値である 250 ミリ秒の遅延が発生します。iodelay に指定した値が大きくなるほどタスクの進行が遅くなり、一定時間内に消費できるシステムリソースが減少します(vxplex、vxvol、vxrecover の各コマンドでは、slow 属性も受け入れられます)。

vxtask コマンドの使用

システム上で実行されているすべてのタスクを一覧表示するには、次のコマンドを使います。

```
# vxtask list
```

親タスクの後に子タスクというように、タスクを階層的に表示するには、次のように -h オプションを指定します。

```
# vxtask -h list
```

ディスクグループ mydg 内の一時停止中のすべてのタスク、およびタグ sysstart の付いたすべてのタスクをトレースするには、次のコマンドを使います。

```
# vxtask -g mydg -p -i sysstart list
```

一時停止中のすべてのタスクを一覧表示するには、vxtask -p list コマンドを使います。実行を継続するには(タスクはタスク ID やタスクタグで指定できます)、vxtask resume を使います。

```
# vxtask -p list
# vxtask resume 167
```

タグ myoperation の付いたすべてのタスクを監視するには、次のコマンドを使います。

```
# vxtask monitor myoperation
```

タグ recovall の付いたタスクをすべて中断するには、次のコマンドを使います。

```
# vxtask abort recovall
```

このコマンドを実行すると、VxVM はこれまでの操作の進行を元に戻そうとします。たとえば、オンライン再レイアウトを中止すると、VxVM はボリュームをもとのレイアウトに戻します。

p.100 の「オンライン再レイアウトの進行状況の制御」を参照してください。

vxnotify による設定の変更の監視

vxnotify ユーティリティを使って、ディスクに関連するイベントや、vxconfigd 設定デーモンで管理される設定の変更を監視することができます。VxVM クラスタ機能が有効なシステムで vxnotify を実行すると、そのシステムのクラスタ状態の変更に関連するイベントが表示されます。vxnotify ユーティリティでは、要求されたイベントタイプの表示を、強制終了されるか、指定された数のイベントを受け取るか、または指定された期間が経過するまで続けます。

検出される設定イベントは、コントローラ、パスおよび DMP ノードの有効化と無効化、RAID 5 ボリュームの縮退モードへの移行、ディスク、プレックスおよびボリュームの切断、ノードとクラスタの結合と切断などです。

たとえば、ディスク、プレックスおよびボリュームが切断された場合にそれらに関する情報をすべて表示するには、次の vxnotify コマンドを実行します。

```
# vxnotify -f
```

共有ディスクグループのインポートとデポートなど、クラスタ設定の変更に関する情報を表示するには、次のコマンドを実行します。

```
# vxnotify -s -i
```

vxnotify(1M)マニュアルページを参照してください。

オンライン再レイアウトの実行

vxassist relayout コマンドを使うと、ボリュームをオフラインにせずに、ボリュームのレイアウトを再設定できます。このコマンドの一般的な形式は次のとおりです。

```
# vxassist [-b] [-g diskgroup] relayout volume [layout=layout] ¥
  [relayout_options]
```

-b オプションを指定すると、ボリュームの再レイアウトはバックグラウンドタスクになります。

メモ: SmartIO VxVM キャッシュが有効になっているボリュームに再レイアウト操作を行うと、ボリュームのキャッシュ内容が無効になる場合があります。

次の再設定レイアウト属性がサポートされています。

連結ミラー	連結ミラー
連結	連結
nomirror	連結

nostripe	連結
RAID5	RAID 5（共有ディスクグループではサポート対象外）
span	連結
ストライプ	ストライプ

p.967 の「可能な再レイアウト変換」を参照してください。

たとえば、次のコマンドは、ディスクグループ mydg 内の連結ボリューム vol02 をストライプボリュームに変更します。デフォルトでは、ストライプボリュームには、2 つのカラムと 64 KB でストライプ化されたユニットサイズがあります。

```
# vxassist -g mydg relayout vol02 layout=stripe
```

場合によっては、ボリューム上ではなくプレックス上で再レイアウトの実行が必要になることがあります。

p.971 の「オンライン再レイアウト用のプレックスの指定」を参照してください。

可能な再レイアウト変換

表 41-1 に、連結ボリュームに対してサポートされている再レイアウト変換を示します。

表 41-1　連結ボリュームに対してサポートされている再レイアウト変換

再レイアウト後のタイプ	連結ボリュームから
連結	不可。
連結ミラー	不可。代わりに、ミラーを追加してから、vxassist convert コマンドを使います。
mirror-concat	不可。代わりにミラーを追加します。
mirror-stripe	不可。代わりに、ストライプ化ミラーボリュームへの再レイアウト後に vxassist convert コマンドを使います。
raid5	はい。ストライプユニットサイズとカラム数を定義できます。
ストライプ	はい。ストライプユニットサイズとカラム数を定義できます。
stripe-mirror	はい。ストライプユニットサイズとカラム数を定義できます。

表 41-2 に、連結ミラーボリュームに対してサポートされている再レイアウト変換を示します。

表 41-2　連結ミラーボリュームに対してサポートされている再レイアウト変換

再レイアウト後のタイプ	連結ミラーボリュームから
連結	不可。代わりに、vxassist convert を使い、作成されたミラー化連結ボリュームから不必要なミラーを削除します。
連結ミラー	不可。
mirror-concat	不可。代わりに、vxassist convert コマンドを使います。
mirror-stripe	不可。代わりに、ストライプ化ミラーボリュームへの再レイアウト後に vxassist convert コマンドを使います。
raid5	可。
ストライプ	はい。この再レイアウトによりミラーが削除され、ストライプ化が追加されます。ストライプユニットサイズとカラム数を定義できます。
stripe-mirror	はい。ストライプユニットサイズとカラム数を定義できます。

表 41-3 に、RAID 5 ボリュームに対してサポートされている再レイアウト変換を示します。

表 41-3　RAID-5 ボリュームに対してサポートされている再レイアウト変換

再レイアウト後のタイプ	RAID-5 から
連結	可。
連結ミラー	可。
mirror-concat	不可。代わりに、連結ミラーボリュームに再レイアウトした後に vxassist convert コマンドを使います。
mirror-stripe	不可。代わりに、ストライプ化ミラーボリュームへの再レイアウト後に vxassist convert コマンドを使います。
raid5	はい。ストライプユニットサイズとカラム数を変更できます。
ストライプ	はい。ストライプユニットサイズまたはカラム数を変更する必要があります。
stripe-mirror	はい。ストライプユニットサイズまたはカラム数を変更する必要があります。変更しない場合は、vxassist convert コマンドを使います。

表 41-4 に、ミラー化連結ボリュームに対してサポートされている再レイアウト変換を示します。

表 41-4 ミラー化連結ボリュームに対してサポートされている再レイアウト変換

再レイアウト後のタイプ	ミラー化連結ボリュームから
連結	不可。代わりに、不必要なミラーを削除します。
連結ミラー	不可。代わりに、`vxassist convert` コマンドを使います。
mirror-concat	不可。
mirror-stripe	不可。代わりに、ストライプ化ミラーボリュームへの再レイアウト後に `vxassist convert` コマンドを使います。
raid5	はい。ストライプユニットサイズとカラム数を定義できます。再レイアウトを実行する、既存のミラーボリューム内のプレックスを選択します。他のプレックスは、再レイアウト操作の最後に削除されます。
ストライプ	可。
stripe-mirror	可。

表 41-5 に、ミラー化ストライプボリュームに対してサポートされている再レイアウト変換を示します。

表 41-5 ミラー化ストライプボリュームに対してサポートされている再レイアウト変換

再レイアウト後のタイプ	ミラー化ストライプボリュームから
連結	可。
連結ミラー	可。
mirror-concat	不可。代わりに、連結ミラーボリュームに再レイアウトした後に `vxassist convert` コマンドを使います。
mirror-stripe	不可。代わりに、ストライプ化ミラーボリュームへの再レイアウト後に `vxassist convert` コマンドを使います。
raid5	はい。ストライプユニットサイズとカラム数を変更できます。
ストライプ	はい。ストライプユニットサイズまたはカラム数を変更する必要があります。
stripe-mirror	はい。ストライプユニットサイズまたはカラム数を変更する必要があります。変更しない場合は、`vxassist convert` コマンドを使います。

表 41-6 に、ミラー化されていないストライプボリュームと階層化されたストライプ化ミラーボリュームに対してサポートされている再レイアウト変換を示します。

表 41-6　ミラー化されていないストライプボリューム、および階層化されたストライプ化ミラーボリュームに対してサポートされている再レイアウト変換

再レイアウト後のタイプ	ストライプまたはストライプ化ミラーから
連結	可。
連結ミラー	可。
mirror-concat	不可。代わりに、連結ミラーボリュームに再レイアウトした後に vxassist convert コマンドを使います。
mirror-stripe	不可。代わりに、ストライプ化ミラーボリュームへの再レイアウト後に vxassist convert コマンドを使います。
raid5	はい。ストライプユニットサイズとカラム数を変更できます。
ストライプ	はい。ストライプユニットサイズまたはカラム数を変更する必要があります。
stripe-mirror	はい。ストライプユニットサイズまたはカラム数を変更する必要があります。

非デフォルトレイアウトの指定

以下の再レイアウトオプションを 1 つ以上指定して、デフォルトのレイアウト設定を変更できます。

`ncol=`*`number`*　カラム数を指定

`ncol=+`*`number`*　追加するカラム数を指定

`ncol=-`*`number`*　削除するカラム数を指定

`stripeunit=`*`size`*　ストライプ幅を指定

以下の例では、vxassist を使って、ディスクグループ dbasedg 内のストライプボリュームのストライプ幅とカラム数を変更しています。

```
# vxassist -g dbasedg relayout vol03 stripeunit=64k ncol=6
# vxassist -g dbasedg relayout vol03 ncol=+2
# vxassist -g dbasedg relayout vol03 stripeunit=128k
```

次の例では、連結ボリュームを 4 カラムの RAID 5 ボリュームに変更しています。

```
# vxassist -g dbasedg relayout vol04 layout=raid5 ncol=4
```

オンライン再レイアウト用のプレックスの指定

ディスクグループ内に十分なディスクや領域がある場合は、どのレイアウトも RAID 5 に変更できます。ミラーボリュームを RAID 5 へ変換するには、どのプレックスを変換するかを指定する必要があります。変換が終わると、他のプレックスはすべて削除され、他の目的のために領域が解放されます。ミラーボリュームを RAID 5 以外のレイアウトに変換する場合、変換されていないプレックスは削除されません。次のように、ボリュームの代わりにプレックスの名前を指定して、変換するプレックスを指定します。

```
# vxassist [-g diskgroup] relayout plex [layout=layout] ¥
  [relayout_options]
```

オンライン再レイアウト操作のタグ設定

再レイアウト操作を一時停止したりもとに戻すなど、操作の進行状況を制御するには、vxassist に -t オプションを指定して、操作にタスクタグを設定します。たとえば、次の再レイアウト操作はバックグラウンドタスクとして実行され、そのタスクにはタグ myconv が設定されます。

```
# vxassist -b -g dbasedg -t myconv relayout vol04 layout=raid5 ¥
  ncol=4
```

p.971 の「オンライン再レイアウトの状態の表示」を参照してください。

p.100 の「オンライン再レイアウトの進行状況の制御」を参照してください。

オンライン再レイアウトの状態の表示

オンライン再レイアウト操作の実行には時間がかかります。vxrelayout コマンドを使って、再レイアウト操作の状態に関する情報を取得できます。たとえば、次のコマンドを考えてみます。

```
# vxrelayout -g mydg status vol04
```

このコマンドにより次のような出力が表示されます。

```
STRIPED, columns=5, stwidth=128--> STRIPED, columns=6,
stwidth=128
Relayout running, 68.58% completed.
```

この例では、ストライプボリュームのカラム数 5 から 6 への再設定が実行中であり、ちょうど 3 分の 2 以上が完了したところです。

vxrelayout（1M）マニュアルページを参照してください。

再レイアウトの開始時に vxassist でタスクタグを指定した場合、このタグを vxtask コマンドで使って、再レイアウトの進行状況を監視できます。たとえば、myconv とタグが設定されたタスクを監視するには、次のように入力します。

```
# vxtask monitor myconv
```

ボリュームへのミラーの追加

次のように vxassist コマンドを使って、ボリュームにミラーを追加できます。

```
# vxassist [-b] [-g diskgroup] mirror volume
```

-b オプションを指定すると、追加されたミラーの同期処理はバックグラウンドタスクとして実行されます。

たとえば、ディスクグループ mydg 内のボリューム voltest のミラーを作成するには、次のコマンドを使います。

```
# vxassist -b -g mydg mirror voltest
```

また、次のコマンドを使い、プレックスを作成してからボリュームに接続することによって、ボリュームをミラー化できます。

```
# vxmake [-g diskgroup] plex plex sd=subdisk ...
# vxplex [-g diskgroup] att volume plex
```

すべてのボリュームのミラー化

ディスクグループ内のすべてのボリュームを使用可能なディスク領域にミラー化するには、次のコマンドを使います。

```
# /etc/vx/bin/vxmirror -g diskgroup -a
```

デフォルトでミラーボリュームを作成するように VxVM を設定するには、次のコマンドを使います。

```
# vxmirror -d yes
```

このような変更を行っても、vxassist コマンドに nmirror=1 を属性として指定することによって、非ミラーボリュームを作成することができます。たとえば、20 GB の非ミラーボリューム nomirror をディスクグループ mydg 内に作成するには、次のコマンドを使います。

```
# vxassist -g mydg make nomirror 20g nmirror=1
```

VxVM ディスク上でのボリュームのミラー化

ボリュームをミラー化すると、ボリュームの 1 つ以上のコピーが別のディスク上に作成されます。ボリュームのミラーコピーを作成することによって、ディスクに障害が発生した場合にデータ損失からボリュームを保護します。

このタスクをルートディスク上で使って、代替起動ボリュームを別のディスク上に構成することができます。ルートディスクで障害が発生している場合でも、このようにしておくとシステムを起動できます。

メモ: このタスクでは、連結ボリュームのミラー化のみ行われます。すでにミラー化されているボリュームや、複数のディスクにサブディスクが存在するボリュームについては無視されます。

ディスク上でボリュームをミラー化するには、次の手順を実行します。

1. ターゲットディスクにソースディスク以上の領域があることを確認します。
2. vxdiskadm メインメニューから、[ディスク上でのボリュームのミラー化(Mirror volumes on a disk)]を選択します。
3. プロンプトで、ミラー化するディスクのディスク名を入力します。

   ```
   Enter disk name [<disk>,list,q,?] mydg02
   ```

4. プロンプトで、ターゲットディスク名を入力します(このディスクは元のディスク以上のサイズである必要があります)。

   ```
   Enter destination disk [<disk>,list,q,?] (default: any) mydg01
   ```

5. プロンプトで、Return キーを押してミラーを作成します。

   ```
   Continue with operation? [y,n,q,?] (default: y)
   ```

   vxdiskadm プログラムによって、次のようにミラー操作の状態が表示されます。

   ```
   VxVM vxmirror INFO V-5-2-22 Mirror volume voltest-bk00
      .
      .
      .
   VxVM INFO V-5-2-674 Mirroring of disk mydg02 is complete.
   ```

6. プロンプトで、ボリュームを他のディスク上にミラー化する(y)か、vxdiskadm メインメニューに戻る(n)かを指定します。

   ```
   Mirror volumes on another disk? [y,n,q,?] (default: n)
   ```

SmartMove の設定

デフォルトでは、SmartMove ユーティリティはすべてのボリュームに対して有効になります。デフォルトの動作を変更したい場合のみ、またはこの機能を以前に無効にした場合にのみ、SmartMove 機能の設定が必要になります。

SmartMove には、SmartMove を適用するかどうかを指定するために 3 つの値が用意されています。3 つの値について説明します。

値	意味
none	SmartMove をまったく使いません。
thinonly	シン対応 LUN にのみ SmartMove を使います。
all	すべてのタイプの LUN に SmartMove を使います。 これはデフォルト値です。

SmartMove 値を設定するには

1 現在およびデフォルトの SmartMove 値を表示するには、次のコマンドを入力します。

```
# vxdefault list
KEYWORD                         CURRENT-VALUE   DEFAULT-VALUE
usefssmartmove                  all             all
...
```

2 SmartMove 値を設定するには、次のコマンドを入力します。

```
# vxdefault set usefssmartmove value
```

value は none、thinonly、all のいずれかです。

ミラーの削除

ミラーが不要になった場合は、そのミラーを削除してディスク領域を解放できます。

メモ: VxVM では、ボリュームに関連付けられている最後の有効なプレックスを削除できません。

ボリュームからミラーを削除するには、次のコマンドを使います。

```
# vxassist [-g diskgroup] remove mirror volume
```

ストレージ属性を使って、削除するストレージを指定することもできます。たとえば、ディスク mydg01 上のミラーをボリューム vol01 から削除するには、次のように入力します。

メモ: ! 文字は一部のシェルでの特殊文字です。次の例では、bash シェルでこの文字をエスケープ処理する方法を示します。

```
# vxassist -g mydg remove mirror vol01 ¥!mydg01
```

p.248 の「指定したディスクにおけるボリュームの作成」を参照してください。

また、ボリュームとミラーの関連付けを解除してミラーを削除する場合は、次のコマンドを使います。

```
# vxplex [-g diskgroup] -o rm dis mirror
```

たとえば、ディスクグループ mydg からミラー vol01-02 の関連付けを解除して削除するには、次のコマンドを使います。

```
# vxplex -g mydg -o rm dis vol01-02
```

このコマンドでは、ミラー vol01-02 および関連付けられたすべてのサブディスクが削除されます。次のコマンドを別々に入力しても同じ結果が得られます。

```
# vxplex -g mydg dis vol01-02
# vxedit -g mydg -r rm vol01-02
```

ボリュームでのタグ設定

ボリュームタグは SmartTier 機能を実装します。また、下に示すのと同じ vxvm コマンド構文を使って、ボリュームセットにタグを適用することもできます。

次の形式の vxassist コマンドを使って、以下の操作を実行できます。

- ボリュームに名前付きタグとオプションのタグ値を設定する。
- タグを置換する。
- ボリュームからタグを削除する。

```
# vxassist [-g diskgroup] settag volume|vsettagname[=tagvalue]
# vxassist [-g diskgroup] replacetag volume|vsetoldtagnewtag
# vxassist [-g diskgroup] removetag volume|vsettagname
```

ボリュームに関連付けされたすべてのタグの一覧を表示するには、次のコマンドを使います。

```
# vxassist [-g diskgroup] listtag [volume|vset]
```

ボリューム名を指定しないと、ディスクグループ内にあるすべてのボリュームとボリュームセットが表示されます。TY フィールドの略称 vt は、ボリュームセットを示します。

listtag コマンドの例を次に示します。

```
# vxassist -g dg1 listtag vol
```

指定したタグ名を持つボリュームの一覧を表示するには、次のコマンドを使います。

```
# vxassist [-g diskgroup] list tag=tagnamevolume
```

タグの名前と値は、256 文字までの大文字と小文字を区別する文字列です。タグ名には、次の ASCII 文字を使えます。

- 英字(A から Z、a から z)
- 数字(0 から 9)
- ハイフン(-)
- 下線文字(_)
- ピリオド(.)

タグ名は、英字または下線文字から始める必要があります。ディスクグループのディスクの名前と同じタグ名は指定できません。

タグ名 site、udid、vdid は予約されているため、使わないでください。今後の製品機能との競合を防ぐため、asl、be、nbu、sf、symc、vx で始まるタグ名を使わないでください。

タグ値には、32 から 127 までの 10 進値を持つ任意の ASCII 文字を使えます。タグ値にスペースがある場合は、シェルからそのタグ値を保護するために、次のように指定を引用符で囲みます。

```
# vxassist -g mydg settag myvol "dbvol=table space 1"
```

list 操作を使うと、ドット区切りのタグ階層を確認できます。たとえば、tag=a.b のリストには、a.b で始まるタグ名が設定されたボリュームがすべて含まれます。

ディスクグループの管理

ここでは、ディスクグループの管理について説明します。

ディスクグループバージョン

各ディスクグループには、バージョン番号が関連付けられています。VxVM (Veritas Volume Manager) の各主要リリースでは、ディスクグループバージョンが導入されています。各リリースの新機能をサポートするには、ディスクグループが最新のディスクグルー

プバージョンになっている必要があります。デフォルトでは、VxVM は最新のディスクグループバージョンでディスクグループを作成します。たとえば、Veritas Volume Manager 7.4.2 は、ディスクグループをバージョン 280 で作成します。

VxVM の各リリースでは、それぞれ特定のディスクグループバージョンがサポートされます。VxVM は、サポートされる任意のバージョンのディスクグループをインポートして各種操作を実行できます。ただし、実行できる操作は、そのディスクグループバージョンでサポートされている機能および操作によって制限されます。以前のバージョンからディスクグループをインポートすると、最新の機能を利用できない場合があります。新しいバージョンの VxVM の機能を使おうとすると、次のようなエラーメッセージが表示されます。

```
VxVM vxedit ERROR V-5-1-2829 Disk group version doesn't support
feature
```

こうした機能を使うには、適切なディスクグループバージョンにディスクグループを明示的にアップグレードする必要があります。

p.983 の「ディスクグループバージョンのアップグレード」を参照してください。

表 41-7は、特定のディスクグループバージョンの導入とサポートを行う Veritas Volume Manager リリースをまとめたものです。また、各ディスクグループバージョンでのサポート対象機能の概略を示します。

表 41-7　　ディスクグループバージョンの割り当て

VxVM リリース	導入されているディスクグループバージョン	サポートされる新機能	サポートされているディスクグループバージョン *
7.4.2	290	DCO での DCM ログ記録 ディスクグループレベルの暗号化と再キー機能	20、30、40、50、60、70、80、90、110、120、130、140、150、160、170、180、190、200、220、230、240、250、260、280
7.4.1	280	技術プレビュー: VVR のアダプティブ同期モード	20、30、40、50、60、70、80、90、110、120、130、140、150、160、170、180、190、200、220、230、240、250、260

VxVM リリース	導入されているディスクグループバージョン	サポートされる新機能	サポートされているディスクグループバージョン *
7.4	260	ボリュームレベルの I/O 転送 インテントロック 保存データと転送中のデータの暗号化	20、30、40、50、60、70、80、90、110、120、130、140、150、160、170、180、190、200、220、230、240、250
7.3.1	240	レプリケーション用のボリュームの暗号化	20、30、40、50、60、70、80、90、110、120、130、140、150、160、170、180、190、200、220、230
7.2、7.3	230	FSS 環境内のホットリロケーション イレイジャコーディングされたボリューム (技術プレビュー) 4K セクタサイズディスクのサポート	20、30、40、50、60、70、80、90、110、120、130、140、150、160、170、180、190、200、220
7.1	220	■ VxVM ボリュームに対するブロックレベルの暗号化のサポート ■ VxVM に対する最大 IOPS のサポート	20、30、40、50、60、70、80、90、110、120、130、140、150、160、170、180、190、200
7.0、6.2	200	■ Atomic Write I/O のサポート ■ 共有ボリュームの SmartIO サポート	20、30、40、50、60、70、80、90、110、120、130、140、150、160、170、180、190
6.1	190	■ SmartIO のキャッシュ ■ Flexible Storage Sharing ■ CVM 拡張	20、30、40、50、60、70、80、90、110、120、130、140、150、160、170、180

VxVM リリース	導入されているディスクグループバージョン	サポートされる新機能	サポートされているディスクグループバージョン *
6.0.1	180	■ SSD (Solid State Device) の TRIM サポート ■ CVM 可用性の拡張	20、30、40、50、60、70、80、90、110、120、130、140、150、160、170
6.0	170	■ VVR の圧縮 ■ VVR のセカンダリログ ■ CVM 可用性の拡張 ■ DCO バージョン 30 ■ 同期タスクのリカバリ	20、30、40、50、60、70、80、90、110、120、130、140、150、160
5.1SP1	160	■ GCO のフェールオーバーの一環としてのバンカーの自動再生 ■ GCO のテイクオーバーの間にプライマリを選択する機能 ■ 32 を超え、64 までのノードを CVM でサポート ■ 大きい LUN (1 TB 超) のレイアウトを CDS でサポート ■ vxrootadm の機能拡張	20、30、40、50、60、70、80、90、110、120、130、140、150、160
5.1	150	SSD デバイスのサポート、ISP dg の移行	20、30、40、50、60、70、80、90、110、120、130、140、150

VxVM リリース	導入されているディスクグループバージョン	サポートされる新機能	サポートされているディスクグループバージョン *
5.0	140	データ移行、リモートミラー、コーディネータディスクグループ (VCS で使われる)、リンクボリューム、スナップショット LUN のインポート	20、30、40、50、60、70、80、90、110、120、130、140
5.0	130	■ VVR の拡張	20、30、40、50、60、70、80、90、110、120、130
4.1	120	■ A/P アレイのクラスタ全体の自動フェールバック ■ 永続 DMP ポリシー ■ 共有ディスクグループ障害ポリシー	20、30、40、50、60、70、80、90、110、120

VxVM リリース	導入されているディスクグループバージョン	サポートされる新機能	サポートされているディスクグループバージョン *
4.0	110	■ CDS (クロスプラットフォームデータシェアリング) ■ DDL (デバイス検出層) 2.0 ■ ディスクグループ設定のバックアップとリストア ■ 特殊ディスクグループとしての rootdg の排除 ■ フルサイズインスタントスナップショットおよび領域最適化インスタントスナップショット ■ ISP (インテリジェントストレージプロビジョニング) ■ シリアルスプリットブレイン検出 ■ ボリュームセット (VxFS での複数デバイスのサポート)	20、30、40、50、60、70、80、90、110

VxVM リリース	導入されているディスクグループバージョン	サポートされる新機能	サポートされているディスクグループバージョン *
3.2、3.5	90	■ クラスタにおける Oracle リシルバリングのサポート ■ ディスクグループの移動、分割および結合 ■ DDL (デバイス検出層) 1.0 ■ クラスタにおける階層化ボリュームのサポート ■ 順次ディスク割り当て ■ OS に依存しない名前の付け方のサポート ■ 永続 FastResync	20、30、40、50、60、70、80、90
3.1.1	80	■ VVR の拡張	20、30、40、50、60、70、80
3.1	70	■ 非永続 FastResync ■ シーケンシャル DRL ■ 再配置の解除 ■ VVR の拡張	20、30、40、50、60、70
3.0	60	■ オンライン再レイアウト ■ RAID 5 サブディスクの安全な移動	20、30、40、60
2.5	50	■ SRVM (現在は Veritas Volume Replicator または VVR と呼ばれる)	20、30、40、50
2.3	40	■ ホットリロケーション	20、30、40

VxVM リリース	導入されているディスクグループバージョン	サポートされる新機能	サポートされているディスクグループバージョン *
2.2	30	■ VxSmartSync Recovery Accelerator	20、30
2.0	20	■ DRL (ダーティリージョンログ) ■ ディスクグループ設定コピーの制限 ■ ミラーボリュームログ ■ 新スタイルのストライプ ■ RAID 5 ボリューム ■ リカバリのチェックポインティング	20
1.3	15		15
1.2	10		10

* 新しい機能をサポートするには、ディスクグループは少なくともその機能が実装されたリリースのディスクグループバージョンである必要があります。

古いバージョンの Veritas Volume Manager を実行するシステムのディスクグループをインポートする必要がある場合は、ディスクグループを以前のディスクグループバージョンで作成できます。

p.984 の「古いディスクグループバージョンのディスクグループの作成」を参照してください。

ディスクグループバージョンのアップグレード

Veritas Volume Manager の各ディスクグループには、バージョン番号が関連付けられています。VxVM の各リリースでは、特定のディスクグループのバージョンがサポートされており、該当するバージョンのディスクグループ上のタスクをインポートして実行することができます。一部の新しい機能とタスクは現在のディスクグループバージョンのディスクグループのみで動作します。

アップグレードするときに、VxVM は既存のディスクグループのバージョンを自動的にアップグレードしません。サポートされるバージョンのディスクグループである場合は、現在のバージョンの機能を使おうとしないかぎり、そのままディスクグループを使えます。ディスクグループがアップグレードされるまで、引き続き、インポート元のリリースにデポートされる可能性があります。

アップグレードされたリリースの機能を使うには、明示的に既存のディスクグループをアップグレードする必要があります。「ダウングレード」機能はありません。アップグレード後のディスクグループには、新しいバージョンをサポートしていない VxVM の旧リリースとの互換性がありません。フェールオーバーまたはオフホスト処理のために複数のサーバー間で共有されるディスクグループの場合は、そのディスクグループを使う可能性のあるすべてのホスト上の VxVM リリースが、アップグレードしようとしているディスクグループのバージョンをサポートしていることを確認します。

Storage Foundation Cluster File System High Availability 7.4.2 にアップグレードした後で、ISP によって編成されるすべての既存のディスクグループをアップグレードする必要があります。バージョンのアップグレードなしで、設定のクエリー操作はうまく動作し続けます。ただし、設定変更操作は正しく機能しません。

ディスクグループのバージョンを表示するには、次のコマンドを使います。

```
# vxdg list dgname
```

また、vxprint コマンドの -l フォーマットオプションを使ってディスクグループのバージョンを判定することもできます。

現在稼動している VxVM のリリースがサポートしている最も新しいバージョンにディスクグループをアップグレードするには、次のコマンドを使います。

```
# vxdg upgrade dgname
```

古いディスクグループバージョンのディスクグループの作成

場合により、古いバージョンの Veritas Volume Manager を実行するシステムでインポートできるディスクグループを作成する必要が生じることがあります。ディスクグループバージョンはダウングレードできないため、ディスクグループを作成するときにディスクグループバージョンを指定する必要があります。

たとえば、Veritas Volume Manager 6.0 が稼動しているシステムで作成されるディスクグループのデフォルトのディスクグループバージョンは 170 です。Veritas Volume Manager 4.1 が稼動しているシステムの場合、そのリリースではバージョン 120 までしかサポートされていないため、このディスクグループをインポートすることはできません。このため、Veritas Volume Manager 6.0 が稼動しているシステムで作成したディスクグループを Veritas Volume Manager 4.1 が稼動しているシステムにインポートできるようにするには、バージョン 120 以前のディスクグループを作成する必要があります。

旧バージョンのディスクグループを作成するには、-T *version* オプションを vxdg init コマンドに指定します。

ディスク情報の表示

既存のディスクグループに関する情報を表示するには、次のコマンドを入力します。

```
# vxdg list
NAME    STATE     ID
rootdg  enabled   730344554.1025.tweety
newdg   enabled   731118794.1213.tweety
```

特定のディスクグループについて、より詳細な情報を表示するには、次のコマンドを使います。

```
# vxdg list diskgroup
```

このコマンドを mydg という名前のディスクグループに適用した場合は、次のように出力されます。

```
# vxdg list mydg

Group: mydg
dgid: 962910960.1025.bass
import-id: 0.1
flags:
version: 160
local-activation: read-write
alignment: 512 (bytes)
ssb: on
detach-policy: local
copies: nconfig=default nlog=default
config: seqno=0.1183 permlen=3448 free=3428 templen=12 loglen=522
config disk sda copy 1 len=3448 state=clean online
config disk sdb copy 1 len=3448 state=clean online
log disk sdc copy 1 len=522
log disk sdd copy 1 len=522
```

特定のディスクに（たとえば、ディスクグループをインポートするため）関連付けられているディスクグループ ID およびディスクグループ名を確認するには、次のコマンドを使います。

```
# vxdisk -s list devicename
```

このコマンドの出力には、指定したディスクに関する次の情報が表示されます。たとえば、ディスク sdc に対する出力は次のようになります。

```
Disk: sdc
type:   simple
flags:  online ready private autoconfig autoimport imported
diskid: 963504891.1070.bass
dgname: newdg
dgid:   963504895.1075.bass
```

```
hostid: bass
info:   privoffset=128
```

ディスクグループの空き領域の表示

ボリュームおよびファイルシステムをシステムに追加する前に、必要な空きディスク領域が存在することを確認します。

システムの空き領域を表示するには、次のコマンドを使います。

```
# vxdg free
```

出力例を次に示します。

```
GROUP DISK      DEVICE    TAG       OFFSET  LENGTH   FLAGS
mydg  mydg01    sda       sda       0       4444228  -
mydg  mydg02    sdb       sdb       0       4443310  -
newdg znewdg01  sdc       sdc       0       4443310  -
newdg newdg02   sdd       sdd       0       4443310  -
oradg oradg01   sde       sde       0       4443310  -
```

ディスクグループの空き領域を表示するには、次のコマンドを使います。

```
# vxdg -g diskgroup free
```

ここで、*-g diskgroup* は、ディスクグループを指定するオプションです。

たとえば、ディスクグループ mydg の空き領域を表示するには、次のコマンドを使います。

```
# vxdg -g mydg free
```

次の出力例では、空き領域の長さがセクタ単位で示されています。

```
DISK   DEVICE    TAG       OFFSET  LENGTH    FLAGS
mydg01 sda       sda       0       4444228   -
mydg02 sdb       sdb       0       4443310   -
```

ディスクグループの作成

ディスクグループは、少なくとも 1 つのディスクに関連付ける必要があります。vxdiskadm コマンドのメインメニューで Add or initialize one or more disks を選択してディスクを VxVM の制御に追加すると、新しいディスクグループを作成できます。ディスクグループに追加するディスクは、既存のディスクグループに所属していないディスクである必要があります。ディスクグループ名にピリオド (.) の文字を含めることはできません。

共有ディスクグループを作成できます。

また、vxdiskadd コマンドを使って新しいディスクグループを作成することもできます。コマンドダイアログは、vxdiskadm コマンドを実行した場合と似ています。

次の例の sdd は現在ディスクグループに割り当てられていないディスクのデバイス名です。

```
# vxdiskadd sdd
```

p.389 の「VxVM へのディスクの追加」を参照してください。

次の vxdg init コマンドを使ってディスクグループを作成することもできます。

```
# vxdg init diskgroup [cds=on|off] diskname=devicename
```

たとえば、デバイス sdc 上に mktdg という名前のディスクグループを作成するには、次のように入力します。

```
# vxdg init mktdg mktdg01=sdc
```

デバイス名 sdc で指定するディスクは、vxdiskadd または vxdiskadm を使って事前に初期化している必要があります。このディスクは、現在ディスクグループに所属していないディスクである必要があります。

vxdg init コマンドとともに cds 属性を使って、新しいディスクグループがクロスプラットフォームデータ共有 (CDS) 機能と互換性があるかどうかを確認できます。新しく作成したディスクグループは、デフォルトで CDS と互換性があります (cds=on を指定する場合と同等)。この設定を変更する場合はファイル /etc/default/vxdg を編集し、新しいディスクグループを作成する前にこのファイル内で属性値を cds=off に設定します。

この属性は、次のコマンドを使ってディスクグループに対して設定することもできます。

```
# vxdg -g diskgroup set cds=on|off
```

-o encrypted オプションを使って、ディスクグループ内のボリュームを暗号化できます。ディスクグループ内で作成されたすべてのボリュームが暗号化されます。vxencrypt list コマンド出力には、このようなディスクグループに「デフォルトで暗号化されています」というメッセージが表示されます。

```
# vxdg -o encrypted init mydg disk1 disk2 disk3
# vxencrypt list
```

ホスト上のすべてのボリュームを暗号化する場合はファイル /etc/default/vxassist を更新して次の行を含めます。

```
encrypted=on
```

ディスクグループからのディスクの削除

ディスクグループから最後のディスクを削除する前に、ディスクグループを無効にする必要があります。

p.1015 の「ディスクグループの無効化」を参照してください。

ディスクグループを無効にする代わりに、ディスクグループを破棄することができます。

p.1015 の「ディスクグループの破棄」を参照してください。

ディスクにサブディスクが含まれていない場合は、次のコマンドを使って、所属するディスクグループからそのディスクを削除できます。

```
# vxdg [-g diskgroup ] rmdisk diskname
```

たとえば、ディスクグループ mydg から mydg02 を削除するには、次のように入力します。

```
# vxdg -g mydg rmdisk mydg02
```

削除しようとしたディスクにサブディスクがある場合は、次のエラーメッセージが表示されます。

```
VxVM vxdg ERROR V-5-1-552 Disk diskname is used by one or more
subdisks
Use -k to remove device assignment.
```

-k オプションを使うと、サブディスクが存在する場合でもディスクを削除できます。

vxdg(1M)マニュアルページを参照してください。

警告: vxdg の -k オプションを使うと、データが失われることがあります。

所属するディスクグループからディスクを削除した後、(オプションで)VxVM の制御下からそのディスクを完全に削除できます。次を入力します。

```
# vxdiskunsetup devicename
```

たとえば、VxVM の制御下からディスク sdc を削除するには、次のように入力します。

```
# vxdiskunsetup sdc
```

ボリュームのサブディスクが定義されているディスクを削除することができます。たとえば、1 つのディスク上にすべてのボリュームを統合することができます。vxdiskadm を使ってディスクを削除する場合は、そのディスク上のボリュームを他のディスクに移動するよう選択することができます。そのためには、vxdiskadm を実行して、メインメニューから[ディスクの削除(Remove a disk)]を選択します。

ディスクがボリュームで使われている場合は、次のメッセージが表示されます。

```
VxVM ERROR V-5-2-369 The following volumes currently use part of
disk mydg02:

home usrvol

Volumes must be moved from mydg02 before it can be removed.

Move volumes to other disks? [y,n,q,?] (default: n)
```

y を選択した場合は、可能であれば、すべてのボリュームがディスクから移動されます。移動できないボリュームもあります。ボリュームを移動できない最も一般的な理由は、次のとおりです。

- 残りのディスクに十分な領域が存在しない。
- ボリューム上の既存のプレックスやストライプサブディスクから異なるディスク上に、プレックスまたはストライプサブディスクを割り当てることができない場合。

vxdiskadm で移動できないボリュームがある場合は、ディスク削除の操作を進める前に、いくつかのディスクからプレックスを削除して空き領域を増やしてください。

ディスクグループのデポート

ディスクグループをデポートすると、システムによってアクセスが有効化(インポート)されているディスクグループへのアクセスを無効にできます。ディスクグループ内のディスクを別のシステム上に移動する場合は、ディスクグループをデポートします。

ディスクグループをデポートするには、次の手順を実行します。

1. デポートするディスクグループに設定されているボリュームに対して行われる、アプリケーションの動作をすべて停止します。ボリューム上に設定されているファイルシステムのマウントを解除し、データベースを停止します。

   対象のディスクグループに(たとえば、マウントされているファイルシステムやデータベースが)使用中のボリュームが含まれている場合は、デポートが失敗します。

2. ディスクグループ内のボリュームを停止するには、次のコマンドを使います。

   ```
   # vxvol -g diskgroup stopall
   ```

3. vxdiskadm のメインメニューから、[ディスクグループのデポート(Remove access to(deport)a disk group)]を選択します。

4. プロンプトで、デポートするディスクグループの名前を入力します。次の例では、newdg になっています。

   ```
   Enter name of disk group [<group>,list,q,?](default: list)
   newdg
   ```

5 このディスクグループ内のディスクを削除する場合は、次のプロンプトで y を入力します。

```
Disable (offline) the indicated disks?[y,n,q,?](default: n) y
```

6 次のプロンプトで、リターンキーを押して操作を続行します。

```
Continue with operation?[y,n,q,?](default: y)
```

ディスクグループをデポートすると、vxdiskadm ユーティリティによって次のメッセージが表示されます。

```
VxVM INFO V-5-2-269 Removal of disk group newdg was
successful.
```

7 次のプロンプトで、他のディスクグループを無効にする(y)か、vxdiskadm メインメニューに戻る(n)かを指定します。

```
Disable another disk group?[y,n,q,?](default: n)
```

次の vxdg コマンドを使ってディスクグループをデポートできます。

```
# vxdg deport diskgroup
```

ディスクグループのインポート

ディスクグループをインポートすると、システムによるディスクグループへのアクセスが有効にされます。あるシステムから別のシステムにディスクグループを移動するには、まず元のシステム上のディスクグループを無効化(デポート)し、そのディスクをシステム間で移動して、ディスクグループを有効化(インポート)します。

デフォルトでは、ディスクグループをインポートすると、VxVM はディスクグループ内の無効にされていたすべてのボリュームをリカバリして起動します。VxVM が無効にされているボリュームをリカバリしないようにするには、自動リカバリ機能を無効にします。たとえば、ディスクグループをインポートした後、ボリュームを起動する前に保守をしたい場合もあります。

p.991 の「ボリュームの自動リカバリの設定」を参照してください。

ディスクグループをインポートするには

1 デポートしたディスクグループ内のディスクが online 状態であることを確認するには、次のコマンドを使います。

```
# vxdisk -s list
```

2 vxdiskadm のメインメニューから、[ディスクグループのインポート(Enable access to(import)a disk group)]を選択します。

3 次のプロンプトで、インポートするディスクグループの名前(この例では newdg)を入力します。

```
Select disk group to import [<group>,list,q,?] (default: list)
  newdg
```

インポートが終わると、vxdiskadm ユーティリティによって次のメッセージが表示されます。

```
VxVM INFO V-5-2-374 The import of newdg was successful.
```

4 次のプロンプトで、他のディスクグループをインポートする(y)か、vxdiskadm メインメニューに戻る(n)かを指定します。

```
Select another disk group? [y,n,q,?] (default: n)
```

また、ディスクグループをインポートする次の vxdg コマンドを使うこともできます。

```
# vxdg import diskgroup
```

また、ディスクグループを共有ディスクグループとしてインポートすることもできます。

p.448 の「共有ディスクグループのインポート」を参照してください。

ボリュームの自動リカバリの設定

デフォルトでは、ディスクグループをインポートすると、VxVM はディスクグループ内の無効にされていたすべてのボリュームをリカバリして起動します。VxVM が無効にされているボリュームをリカバリしないようにするには、ボリュームの自動リカバリを無効にします。たとえば、ディスクグループをインポートした後、ボリュームを起動する前に保守をしたい場合もあります。

ボリュームの自動リカバリ機能を無効にするには

- システム全体で自動ボリュームリカバリを無効にするには、次のコマンドを使います。

  ```
  # vxtune autostartvolumes off
  ```

  または

  特定のディスクグループのインポート操作で自動ボリュームリカバリを無効にするには、noautostart オプションを使います。

  ```
  # vxdg -o noautostart import diskgroup
  ```

マイナー番号競合時の対処

インポートするディスクグループ内のボリュームデバイスのマイナー番号が、既存のボリュームデバイスと競合する場合があります。リリース 5.1 の前にリリースされた VxVM リリースでは、このような競合はエラーになっていました。ディスクグループのインポート操作が失敗していたか、またはスレーブノードが共有ディスクグループへの結合に失敗していました。こうような状況が発生した場合は、vxdg reminor コマンドを手動で実行して、マイナー番号の競合を解決する必要がありました。リリース 5.1 からは、VxVM がマイナー番号の競合を自動的に解決します。

ディスクグループのインポート時にマイナー番号の競合が存在する場合、VxVM はディスクグループに新しいベースマイナー番号を自動的に割り当てて、新しいベースマイナー番号に基づいてディスクグループ内のボリュームにマイナー番号を付け直します。マイナー番号の競合を解決するために vxdg reminor コマンドを実行する必要はありません。

共有ディスクグループと専用ディスクグループの間の競合を回避するために、マイナー番号は共有プールと専用プールに分けられています。VxVM は、共有ディスクグループのマイナー番号を共有プールからのみ割り当てて、専用ディスクグループのマイナー番号は専用プールからのみ割り当てます。専用ディスクグループを共有ディスクグループとして、またはその逆にインポートすると、デバイスマイナー番号は正しいプールから再度割り当てられます。ディスクグループへのマイナー番号の再割り当ては動的に実行されます。

デフォルトでは、専用マイナー番号は 0 から 32999 の範囲であり、共有マイナー番号は 33000 から始まります。この区切りは、必要に応じて変更できます。たとえば、共有マイナー番号の範囲が、より小さい番号から始まるように設定できます。このような範囲を利用すると、共有ディスクグループ用のマイナー番号を増やし、専用ディスクグループ用のマイナー番号を減らせます。

通常、専用プールと共有プールのマイナー番号の数は十分にあるため、区切りを変更する必要はありません。

メモ: 新しい区切りを有効にするには、デフォルトファイル内のチューニングパラメータが変更された後、vxdctl enable を実行するか、vxconfigd を再起動する必要があります。ノードの結合、ボリュームの作成、ディスクグループのインポート操作でのノード障害を防止するために、すべてのクラスタノード上の区切りが正確に同じである必要があります。

共有マイナー番号と専用マイナー番号の間の区切りを変更するには

1 デフォルトファイル /etc/default/vxsf にチューニングパラメータ sharedminorstart を追加します。たとえば、共有マイナー番号の範囲が 20000 から始まるように変更するには、/etc/default/vxsf ファイル内に次の行を設定します。

```
sharedminorstart=20000
```

共有マイナー番号を 1000 未満の番号から始まるように設定することはできません。sharedminorstart が 0 から 999 の値に設定されている場合、専用マイナー番号と共有ディスクグループのマイナー番号の区切りは 1000 に設定されます。0 の値を指定すると、番号の動的な再割り当ては無効になります。

2 次のコマンドを実行します。

```
# vxdctl enable
```

特定のシナリオでは、共有マイナー番号と専用マイナー番号の間の区切りを無効にすることが必要になる場合があります。たとえば、以前のリリースからのアップグレード時に、デバイスマイナー番号が変更されないようにする場合が挙げられます。この場合は、新しい VxVM RPMをインストールする前に、マイナー番号の動的な再割り当てを無効にします。

共有マイナー番号と専用マイナー番号の間の区切りを無効にするには

1 デフォルトファイル /etc/default/vxsf のチューニングパラメータ sharedminorstart を 0(ゼロ)に設定します。/etc/default/vxsf ファイル内に次の行を設定します。

```
sharedminorstart=0
```

2 次のコマンドを実行します。

```
# vxdctl enable
```

システム間でのディスクグループの移動

ディスクグループの重要な特長は、システム間で移動できることです。ディスクグループ内のすべてのディスクが 1 つのシステムから別のシステムに移動できれば、ディスクグループは 2 番目のシステムで使うことができます。設定を再指定する必要がありません。

ディスクグループをシステム間で移動するには

1. 移動先のシステム上で、ディスクグループ内のすべてのディスクが認識可能であることを確認します。これにはマスキングとゾーン化の変更が必要になる場合があります。
2. 移動元システムで、ディスクグループ内のすべてのボリュームを停止し、次のコマンドを使ってディスクグループをデポート(ローカルアクセスを無効化)します。

   ```
   # vxdg deport diskgroup
   ```

3. すべてのディスクを移動先システムに移動し、新しいディスクを認識するために、移動先システムおよび VxVM に対してシステムに応じた必要な手順を実行します。

   これには再ブートが必要な場合があり、それにより、`vxconfigd` デーモンが再起動し、新しいディスクが認識されます。再ブートしない場合は、`vxconfigd` プログラムを再起動するために `vxdctl enable` コマンドを使い、VxVM でもディスクが認識されるようにします。

4 このコマンドを使い、移動先システムのディスクグループをインポート(ローカルアクセスを有効化)します。

```
# vxdg import diskgroup
```

警告: ディスクグループのすべてのディスクは、もう一方のシステムに移動する必要があります。それらが移動されないと、インポートに失敗します。

5 デフォルトでは、ディスクグループをインポートすると、VxVM はディスクグループ内の無効にされていたすべてのボリュームを有効にして起動します。

p.991 の「ボリュームの自動リカバリの設定」を参照してください。

自動リカバリ機能をオフにしてある場合は、次のコマンドを使ってすべてのボリュームを開始します。

```
# vxrecover -g diskgroup -sb
```

クラッシュしたシステムからディスクを移動することもできます。この場合、移動元システムからディスクグループをデポートすることはできません。ディスクグループが作成されるか、システムでインポートされると、そのシステムはディスクグループのすべてのディスクに対してロックを書き込みます。

警告: ロックの目的は、SAN アクセスされたディスクが、両方のシステムによって同時に使われないようにすることです。2 つのシステムが同じディスクに同時にアクセスを試みる場合、VxVM のクラスタ化機能のようなソフトウェアを使用してこれを管理する必要があります。アクセスが管理されないと、ディスク上に格納されたデータおよび設定情報が破損し、使用できなくなる可能性があります。

ディスクのインポート時に発生するエラーの処理

クラッシュしたシステムまたはディスクグループが見つからないシステムからディスクを移動する場合は、ディスク上に保存されているロックを解除する必要があります。システムは次のエラーメッセージを返します。

```
VxVM vxdg ERROR V-5-1-587 disk group groupname: import failed:
Disk is in use by another host
```

次のメッセージは、ディスクグループに有効なディスクが含まれていないことを示しています(ディスクがまったく含まれていないことを示しているのではありません)。

```
VxVM vxdg ERROR V-5-1-587 Disk group groupname: import failed:
No valid disk found containing disk group
```

設定コピー内のホスト ID と /etc/vx/volboot ファイルに格納されているホスト ID が一致しないために、無効なディスクと見なされている可能性があります。

一連のデバイスを指定してロックを解除するには、次のコマンドを使います。

```
# vxdisk clearimport devicename ...
```

インポート時にロックを解除するには、次のコマンドを使います。

```
# vxdg -C import diskgroup
```

警告: 同じディスクを SAN 経由で参照しているシステムで vxdisk clearimport コマンドまたは vxdg -C import コマンドを使う場合は、注意が必要です。ロックを解除すると、それらのディスクに複数のホストから同時にアクセスできるようになり、その結果、データが破損されることがあります。

ディスクグループが最後にインポートされたときに表示できたすべてのディスクにアクセスできる場合、そのディスクグループを正常にインポートできます。ただし、一部のディスクを使用できない場合、-f オプションを指定してディスクグループを強制的にインポートする必要があります。import 操作に失敗した場合、エラーメッセージが表示されます。

次のエラーメッセージは致命的なエラーを示します。このメッセージが表示された場合は、ハードウェアの修復、新規ディスクグループの作成、ディスクグループ設定情報とデータのリカバリが必要です。

```
VxVM vxdg ERROR V-5-1-587 Disk group groupname: import failed:
Disk group has no valid configuration copies
```

次のエラーメッセージは、リカバリが可能なエラーであることを示します。

```
VxVM vxdg ERROR V-5-1-587 Disk group groupname: import failed:
Disk for disk group not found
```

ディスクグループの一部のディスクに障害が起きた場合は、vxdg import コマンドに -f オプションを指定すると、ディスクグループを強制的にインポートできます。

```
# vxdg -f import diskgroup
```

警告: -f オプションを使う場合は注意が必要です。このオプションを指定すると、別々のディスクセットから同じディスクグループが 2 回インポートされます。これによって、ディスクグループ設定に不整合が起きる可能性があります。

---

p.1008 の「競合する設定コピーの扱い方」を参照してください。

-f オプションを使って、不完全なディスクグループのインポートを強制的に正常なインポートとして処理するため、以降のインポートでこのオプションを指定しない場合でも、不

完全なディスクグループがインポートされることがあります。これは、ユーザーの望む処理ではない可能性があります。

また、ディスクグループを共有ディスクグループとしてインポートすることもできます。

p.448 の「共有ディスクグループのインポート」を参照してください。

これらの操作は、vxdiskadm ユーティリティを使って実行することもできます。vxdiskadm を使ってディスクグループをデポートするには、メインメニューから[ディスクグループのデポート(Remove access to(deport)a disk group)]を選択します。ディスクグループをインポートするには、[ディスクグループのインポート(Enable access to(import)a disk group)]を選択します。vxdiskadm によるインポート操作では、ホストインポートロックがチェックされ、検出されたホストインポートロックを解除するかどうかを確認するメッセージが表示されます。この操作では、ディスクグループ内のボリュームも起動されます。

ディスクグループのマイナー番号の予約

デバイスマイナー番号は、デバイスを制御するデバイスドライバに対してデバイスの一定の特性を重複しないように識別します。多くの場合、個々のデバイスの一定の特性モードを識別したり、1 つのコントローラの制御下に置かれている各デバイスを識別するために使われます。VxVM で管理するオブジェクト(ボリューム、プレックス、サブディスク、ディスクまたはディスクグループ)には、それぞれ重複のないデバイスマイナー番号が割り当てられます。

ディスクグループをシステム間で移動する際には、以前のシステムで使われていたマイナー番号が、新しいシステム上の VxVM で認識されているオブジェクトのマイナー番号と重なる可能性があります。この問題を回避するには、各ディスクグループのマイナー番号に重複しない範囲を割り当てる必要があります。VxVM は、ディスクグループのディスクからボリュームオブジェクトが作成されるときに指定された範囲のマイナー番号を使います。再ブートまたは再設定を行っても、各ボリュームのマイナー番号は変わりません。したがって、コンピュータ間でディスクグループを移動する際にデバイス番号の衝突が発生することはなくなります。

VxVM は、このディスクグループから作成されたオブジェクトに対して、ベースマイナー番号 *base_minor* から始まるマイナーデバイス番号を選択します。マイナー番号は、この値から 65,535(2.6 以上のカーネルの場合)までの範囲の値を取ることができます。万が一デバイスマイナー番号の競合が発生したときに一時デバイス番号の再マッピングに使えるように、この範囲の先頭近くに、ある程度のマイナー番号を未割り当てのまま残しておくことをお勧めします。

VxVM では、マイナー番号の 0 - 999 が予約済となり、ブートディスクグループ内のボリューム用に使われます。たとえば、rootvol ボリュームには常にマイナー番号 0 が割り当てられます。

ディスクグループのマイナー番号範囲のベースを指定しない場合は、VxVM が無作為に 1 つ選択します。選択される番号は 1000 以上の 1000 の倍数であり、1000 個のデバイス番号を含む範囲が使用可能になります。また、選択される番号は現在インポートされて

いるどのディスクグループに対する1000単位の範囲とも重複せず、現在割り当てられているどのボリュームデバイス番号とも重複しません。

メモ: デフォルトのポリシーを用いることで、少数のディスクグループはひとまとまりのコンピュータ間で正常に認識されます。ただし、ディスクグループがフェールオーバー機構を使って自動的に認識される場合には、マイナー番号の衝突を回避するため使う範囲を指定します。

既存のディスクグループのベースマイナー番号を表示するには、vxprint コマンドを使います。ディスクグループ mydg の例を次に示します。

```
# vxprint -l mydg | grep minors
minors: >=45000

# vxprint -g mydg -m | egrep base_minor
base_minor=45000
```

作成するディスクグループにボリュームデバイスのベースマイナー番号を設定するには、次のコマンドを使います。

```
# vxdg init diskgroup minor=base_minor disk_access_name ...
```

たとえば、次のコマンドを実行すると、指定したディスクを含むディスクグループ newdg が作成され、ベースマイナー番号が 30000 に設定されます。

```
# vxdg init newdg minor=30000 sdc sdd
```

ディスクグループがすでに存在する場合は、vxdg reminor コマンドを使って、ベースマイナー番号を変更できます。

```
# vxdg -g diskgroup reminor new_base_minor
```

たとえば、次のコマンドを実行すると、ディスクグループ mydg のベースマイナー番号が 30000 に変更されます。

```
# vxdg -g mydg reminor 30000
```

ボリュームが起動している場合、システムを再ブートするかディスクグループをデポートして再インポートするまでは古いデバイス番号が有効になっています。起動しているボリュームを停止すると、vxdg reminor を再度実行して、再ブートまたは再インポートを行わなくても番号を付け直すことができるようになります。

ベースマイナー番号を変更する必要があるディスクグループの一例としては、クラスタ共有ディスクグループが挙げられます。共有ディスクグループ内のボリュームは、すべてのノードで同じマイナー番号が設定されている必要があります。ノードをクラスタに結合しようとした場合にマイナー番号に矛盾があると、結合は失敗します。この矛盾を解消するに

は、クラスタ内にあるノードで reminor 操作を実行します。2 つ以上のノードが結合されたクラスタでは、すべてのノードで矛盾がないベースマイナー番号を使ってください。

vxdg(1M)マニュアルページを参照してください。

p.992 の「マイナー番号競合時の対処」を参照してください。

プラットフォーム間でのディスクグループの互換性

ディスクグループでクロスプラットフォームデータシェアリング(CDS)機能を使う場合、AIX、HP-UX、Linux(カーネル 2.6 以上)および Solaris では、オペレーティングシステム間の互換性を保つために、マイナー番号の上限が 65,535 に制限されます。

カーネル 2.6 より前の Linux では、マイナー番号の数が 256 に制限されます(ベースは 0)。すなわち、システム全体でサポートできるボリュームおよびディスクの数が、他のオペレーティングシステムプラットフォームで許可されているよりも少ない数に制限されます。カーネル 2.6 より前の Linux でサポートされるディスクの数は、通常は数百に制限されます。Linux 上の VxVM 4.0 で拡張メジャー番号体系が実装されている場合、15 個の拡張メジャー番号の連続したブロックが使用可能であれば、設定できるボリュームの最大数は 4079 でした。

VxVM 4.1 以降のリリースは 2.6 バージョンの Linux カーネルで動作します。このカーネルでは、メジャーデバイス番号ごとに設定できるマイナーデバイスの最大数が、256 から 65,536 に増加しています。これにより、多数のボリュームおよびディスクデバイスをシステム上で設定できます。そのようなシステムで設定できる DMP 数およびボリュームデバイス数の理論上の制限は、それぞれ 65,536 および 1,048,576 です。ただし、実際には 1 つのディスクグループで設定できる VxVM デバイスの数は、プライベートリージョンのサイズによって制限されます。

CDS と互換性があるディスクグループをカーネル 2.6 より前の Linux システムにインポートした場合は、VxVM によってマイナー番号が再度割り当てられます。

CDS と互換性があるディスクグループをカーネル 2.6 より前の Linux などのオペレーティングシステム間で移動できるようにするには、次のコマンドを使って、ディスクグループで maxdev 属性を設定します。

```
# vxdg -g diskgroup set maxdev=4079
```

メモ: この属性を設定したディスクグループでも、ボリュームに割り当てられるシステム上のマイナー番号の数が 4079 を超えたり、使用可能な拡張メジャー番号の数が 15 未満だと、カーネル 2.6 より前の Linux にはインポートできない場合があります。

次のコマンドを使って、Linux ホスト上の VxVM でサポートされている最大ボリューム数を確認することができます。

```
# cat /proc/sys/vxvm/vxio/vol_max_volumes
4079
```

vxdg(1M)マニュアルページを参照してください。

ハードウェアクローンディスクを含むディスクグループのインポート

ディスクグループをインポートすると、VxVM はディスクセットに一貫があり、重複したディスクが含まれていないことを確認しようとします。デフォルトでは、VxVM は udid_mismatch フラグまたは clone_disk フラグが付いているディスクをインポートしません。ディスクグループにクローンディスクと非クローンディスクが両方とも含まれている場合、デフォルトでは元の(非クローン)ディスクのみが含まれるディスクグループがインポートされます。VxVM では同じディスクグループ内のクローンディスクと非クローンディスクを一緒にインポートすることができません。

p.101 の「VxVM のハードウェアクローンまたはスナップショットの処理方法」を参照してください。

クローンディスク(ディスクコピー)をインポートする場合は、次のいずれかを実行できます。

- UDID を更新し、クローンディスクのフラグを消去する。
  クローンディスクを標準データディスクとして使う場合は、ディスクの UDID を更新して、clone_disk フラグを消去できます。ディスクが実際はクローンでない場合、または別のホストのクローンディスクのセット全体を使うため、クローンディスクと元のディスクの違いを維持することに気を遣う必要がない場合は、このオプションを使います。
  p.1001 の「ディスクへの新しい UDID の書き込み」を参照してください。
- クローンディスクを含むディスクグループをインポートする。
  クローンディスクのみをインポートし、元の標準ディスクはインポートしません。useclonedev オプションを指定します。デフォルトでは、clone_disk フラグがディスクに残るため、引き続き元のディスクとコピーを区別することができます。
  p.1002 の「クローンディスクのみを含む既存ディスクグループのインポート」を参照してください。
- クローンディスクを新しいディスクグループとしてインポートする。
  クローンディスクのみを新しいディスクグループにインポートします。VxVM はディスクをコピーとして認識しなくなるため、clone_disk フラグは削除されます。ディスクグループはディスクのコピーを含む標準ディスクグループになります。このオプションが適用されるのは、ディスクグループ内に一貫性のある単一のクローンディスクセットが存在する場合のみです。複数のディスクに同じ UDID がある場合、VxVM はディスクグループのインポートを禁止します。たとえば、2 つの異なるポイントインタイムスナップショットが作成されている場合は、ディスクグループに各ディスクのコピーが 2 つ含まれているため、このオプションを使えません。
  p.1003 の「新しい標準ディスクグループとしてのクローンディスクのインポート」を参照してください。
- タグの付いた、一貫性のあるクローンディスクセットをインポートする。
  ハードウェアスナップショットが複数あるため、インポート元のディスクグループに同じディスクのコピーが複数存在する場合は、VxVM ディスクのタグ付けを使って一貫性

のあるクローンディスクセットを識別できます。その後、タグの付いたディスクをインポートできます。
p.1003 の「タグの付いたクローンディスクセットのインポート」を参照してください。

EMC CLARiiON のスナップショット LUN を使う場合の注意事項

元の LUN と同じホストにプライマリ LUN のスナップショット LUN をインポートする必要がある場合は、次の制限があることに注意してください。

アレイボリューム ID（AVID）が有効な状態でエンクロージャに基づく命名（EBN）を使う場合は、もとのホストにスナップショット LUN をインポートする前に、デバイス検出中の名前の永続性を無効にします。

名前の永続性を無効にするには、次のコマンドを使います。

```
# vxddladm set namingscheme=ebn persistence=no use_avid=yes
```

DDL が LUN を認識した後、次のコマンドを使って名前の永続性を有効にします。

```
# vxddladm set namingscheme=ebn persistence=yes use_avid=yes
```

ディスクへの新しい UDID の書き込み

VxVM に `udid_mismatch` フラグまたは `clone_disk` フラグの付いたディスクがある場合は、そのディスクに格納された UDID を更新できます。この手順では、デバイス検出層（DDL）データベースに格納されている現在の UDID 値を使ってプライベートリージョンの値を訂正します。クローンディスクと元のディスクの違いを維持する必要がない場合は、このオプションを使います。たとえば、ディスクが実際はクローンでない場合や、別のホストのクローンディスクセット全体を使っている場合などです。

ディスクの UDID を更新するには

- インポートされたディスクグループに含まれない 1 つ以上のディスクの固有ディスク識別子（UDID）を更新するには、次のコマンドを使います。

  ```
  # vxdisk [-cf] [-g diskgroup ] updateudid disk ...
  ```

  たとえば、次のコマンドはディスク `sdg` と `sdh` の UDID を更新します。

  ```
  # vxdisk updateudid sdg sdh
  ```

  VxVM がディスクに対して `udid_mismatch` フラグを設定していない場合は、`-f` オプションを指定する必要があります。

  VxVM がクローンでないディスクに `udid_mismatch` フラグを設定している場合は、`-c` オプションを指定して `udid_mismatch` フラグと `clone` フラグを削除します。

クローンディスクのみを含む既存ディスクグループのインポート

ディスクグループ内の標準(非クローン)ディスクがインポートされていない場合は、クローンディスクのみを含む既存のディスクグループをインポートできます。デフォルトでは、clone_disk フラグがディスクに設定されているため、引き続き元のディスクとクローンディスクを区別することができます。

この手順は一時的なシナリオに便利です。たとえば、PITC を検証するためにクローンディスクのみをインポートする場合などです。クローンディスクを検証した後にクローンディスクをデポートし、標準ディスクを再度インポートできます。

元のディスクの単一時点のコピーである、一貫性のあるクローンディスクセットをインポートしてください。各ディスクには udid_mismatch フラグ、clone_disk フラグ、または両方が設定されている必要があります。どの 2 つのディスクでも UDID が同じにならないようにしてください。つまり、同じ元のディスクのコピーが 2 つ存在してはなりません。

システム上にディスクグループ内のディスクの複数のコピーがある場合は、ディスクタグを使う必要があります。

p.1003 の「タグの付いたクローンディスクセットのインポート」を参照してください。

VxVM はクローンディスクと非クローンディスクを両方含むディスクグループをサポートしません。クローンディスクと標準ディスクの両方を同時にインポートする場合は、クローンディスクグループに新しいディスクグループ名を指定する必要があります。

p.1003 の「新しい標準ディスクグループとしてのクローンディスクのインポート」を参照してください。

クローンディスクのみを含むディスクグループのみをインポートするには

1. 1 つ以上のクローンディスクに、プライベートリージョン内の現在の設定データベースのコピーがあることを確認します。

   p.1005 の「ディスクグループの設定データベースコピー(メタデータ)の設定」を参照してください。

2. この例に示すように、vxdg import コマンドに -o useclonedev=on オプションを指定して、クローンディスクのみをインポートします。

   ```
   # vxdg -o useclonedev=on [-o updateid] import mydg
   ```

   このコマンド形式を実行すると、クローンディスクのみをインポートできます。クローン以外のすべてのディスクはインポートされません。

   -o updateid オプションを指定して新しい識別属性をディスクに書き込み、ディスクに clone_disk フラグを設定します(フラグの設定には、vxdisk set clone=on コマンドも使えます)。

新しい標準ディスクグループとしてのクローンディスクのインポート

ディスクグループ内の標準（非クローン）ディスクがすでにインポート済みの場合は、同じディスクグループ内のクローンディスクを同時にインポートできません。VxVM はクローンディスクと非クローンディスクを両方含むディスクグループをサポートしません。クローンディスクをインポートする場合は、クローンディスクを含むディスクグループに新しいディスクグループ名を指定する必要があります。

元のディスクの単一時点のコピーである、一貫性のあるクローンディスクセットをインポートしてください。各ディスクには `udid_mismatch` フラグ、`clone_disk` フラグ、または両方が設定されている必要があります。どの 2 つのディスクでも格納された UDID が同じにならないようにしてください。つまり、同じ元のディスクのコピーが 2 つ存在してはなりません。

1 つ以上のクローンディスクのコピーが複数存在する場合は、タグを使う必要があります。

p.1003 の「タグの付いたクローンディスクセットのインポート」を参照してください。

インポート操作を行うと、新しいディスクグループのディスクから `udid_mismatch` フラグおよび `clone_disk` フラグが消去されます。新しいディスクグループは標準ディスクグループになり、ディスクは標準ディスクになります。

クローンディスクを新しいディスクグループとしてインポートするには

1. 1 つ以上のクローンディスクに、プライベートリージョン内の現在の設定データベースのコピーがあることを確認します。

2. この例に示すように、`vxdg import` コマンドに `-o useclonedev=on` オプションを指定して、クローンディスクのみをインポートします。

   ```
   # vxdg -n clonedg -o useclonedev=on -o tag=my_tagged_disks ¥
     import mydg
   ```

   このコマンド形式を実行すると、クローンディスクのみをインポートできます。

タグの付いたクローンディスクセットのインポート

同じディスクセットを複数回コピーすると、各ディスクに複数のクローンディスクが存在することになります。VxVM はコピー元のディスクとクローンディスクの違いを認識しますが、一貫性のあるデータセットを表すクローンディスクを判別することはできません。たとえば、異なる時刻に複数のハードウェアスナップショットを作成した場合は、1 組のクローンディスクが特定時点における各スナップショットを表します。

ディスクグループのクローンディスクをインポートしようとした場合（`-o useclonedev` オプションを使用）、VxVM が同じディスクの複数のクローンを検出すると、インポート操作は失敗します。この動作により、一貫性のないディスクセットのインポートは防止されます。たとえば、さまざまな時点のスナップショットディスクが混在している場合です。

複数のクローンセットが存在する場合は、インポート用に選択するクローンディスクをVxVMに指示する必要があります。一貫性のあるセットを構成するディスクを識別して、これらのディスクに VxVM タグを割り当てます。その後で、指定されたタグの付いたクローンディスクをインポートできます。

タグの付いたクローンディスクセットをインポートするには

1 セットとしてまとめてインポートする必要があるディスクを識別します。

```
# vxdisk -o alldgs list

DEVICE       TYPE         DISK    GROUP  STATUS
EMC0_4       auto:cdsdisk mydg01  mydg   online
EMC0_6       auto:cdsdisk mydg02  mydg   online
EMC0_8       auto:cdsdisk -       (mydg)  online udid_mismatch
EMC0_15      auto:cdsdisk -       (mydg)  online udid_mismatch
EMC0_18      auto:cdsdisk -       (mydg)  online udid_mismatch
EMC0_24      auto:cdsdisk -       (mydg)  online udid_mismatch
```

2 ディスクにまだタグが付いていない場合は、次のコマンドを使って、インポートするディスクグループ内のすべてのディスクにタグを付けます。

```
# vxdisk [-g diskgroup ] settag tagnamedisk ...
```

ここで、*tagname* は最大 128 文字の文字列であり、スペースやタブは含められません。

たとえば、udid_mismatch ディスクが 2 つの異なるポイントインタイムスナップショットを表しているとします。これらのスナップショットを区別するには、次のようにディスクにタグを付けます。

```
# vxdisk settag snaptag1=snap1 EMC0_8 EMC0_15

# vxdisk settag snaptag2=snap2 EMC0_18 EMC0_24
```

vxdisk(1M)マニュアルページを参照してください。

3 どのディスクにタグが付いているかを調べるには、vxdisk listtag コマンドを使います。

```
# vxdisk listtag

DEVICE       NAME     VALUE
EMC0_8       snaptag1 snap1
EMC0_15      snaptag1 snap1
EMC0_18      snaptag2 snap2
EMC0_24      snaptag2 snap2
```

4 snaptag1 のタグの付いたクローンディスクをインポートするには、UDID を更新します。mydg ディスクグループはすでにインポートされているため、mydg 以外のディスクグループ名を割り当てる必要があります。

```
# vxdg -n bcvdg -o useclonedev=on -o tag=snaptag1 -o updateid ¥
import mydg
# vxdisk -o alldgs list

DEVICE        TYPE          DISK    GROUP  STATUS
EMC0_4        auto:cdsdisk mydg01  mydg   online
EMC0_6        auto:cdsdisk mydg02  mydg   online
EMC0_8        auto:cdsdisk mydg01  bcvdg  online
EMC0_15       auto:cdsdisk mydg02  bcvdg  online
EMC0_18       auto:cdsdisk -       (mydg) online udid_mismatch
EMC0_24       auto:cdsdisk -       (mydg) online udid_mismatch
```

EMC0_18 と EMC0_24 クローンディスクは、snaptag1 のタグが付いていないので、インポートされません。

インポートされたクローンディスクの状態が online udid_mismatch から online に変更されました。このディスクは新しいディスクグループに含まれているため、VxVM は clone_disk フラグを削除します。

vxdg(1M)マニュアルページを参照してください。

ディスクグループの設定データベースコピー(メタデータ)の設定

各 VxVM ディスクグループには、ディスクグループ内のオブジェクトに関する永続的な設定データ(メタデータ)が格納された設定データベースがあります。ディスクグループをインポートすると、VxVM はこのデータベースに問い合わせます。設定データベースはディスクグループ内の 1 つ以上の VxVM ディスクのプライベートリージョンに格納されます。ディスクグループごとに、格納される設定のコピー数が設定されています。

vxdg(1M)マニュアルページを参照してください。

ディスクグループ内の一部のディスクセットのみをインポートする場合は、少なくとも 1 つのインポート済みディスクに現在の設定データベースのコピーが含まれていることを確認する必要があります。

ディスクセットの設定コピーを設定するには

1 次のコマンドを使うと、ディスクグループ内の指定したタグを共有するすべてのディスクに設定データベースとカーネルログのコピーを配置できます。

```
# vxdg [-g diskgroup] set tagmeta=on tag=tagname nconfig=all ¥
  nlog=all
```

2 ディスクグループに tagmeta=on が設定されている場合は、次のコマンドを使ってディスクのタグと、設定コピー数の設定値を表示します。値が -1 の場合は、タグの付いたすべてのディスクで設定コピーまたはログコピーが維持されています。

```
# vxdg listmeta diskgroup
```

特定のディスクの設定コピーを設定するには

◆ ディスクグループの配置ポリシーに関係なく、指定されたディスクに設定コピー（メタデータ）のコピーを配置するには、次のコマンドを使います。この属性は、ディスクグループにディスクを追加する前でも、追加した後でも設定できます。

```
# vxdisk [-g diskgroup] set disk keepmeta=always
```

ディスクグループ名の変更

ディスクグループの名前はシステム内で重複していないことが必要です。ターゲットシステムに同じ名前のディスクグループがすでに存在する場合は、ディスクグループをインポートまたはデポートすることはできません。この問題を回避するために、VxVM ではインポートまたはデポートの際にディスクグループ名を変更することができます。

インポート時にディスクグループ名を変更するには、次のコマンドを使います。

```
# vxdg [-t] -n newdg import diskgroup
```

-t オプションを指定すると、そのインポートは一時的なものとなり、再起動すると無効になります。この場合には、元のホストで設定したディスクグループ名は変更されずに残りますが、インポートするホスト側ではこのディスクグループは一時的に *newdg* で指定された名前として認識されます。-t オプションを指定しないと、名前の変更は永続的になります。

たとえば、次のコマンドを使うと、インポート時にディスクグループ mydg の名前が一時的に mytempdg に変更されます。

```
# vxdg -t -n mytempdg import mydg
```

デポート時にディスクグループの名前を変更するには、次のコマンドを使います。

```
# vxdg [-h hostname] -n newdg deport diskgroup
```

デポート時に名前を変更する場合は、*-h hostname* オプションを指定して、オフホストにロックを割り当てることができます。この操作により、オフホストの再起動時に必ず当該ディスクグループが自動的にインポートされます。

たとえば、次のコマンドを使うと、ディスクグループ mydg の名前が myexdg に変更され、ホスト jingo にデポートされます。

```
# vxdg -h jingo -n myexdg deport mydg
```

ブートディスクグループの場合は、マウントされているディスクグループが使用中のボリューム(/ など)が含まれているため、この方法で名前を変更することはできません。ブートディスクグループの名前を変更するには、まずルートディスクのミラー化およびカプセル化を解除し、次に別のディスクグループで再度ミラー化およびカプセル化を実行する必要があります。このディスクグループが新しいブートディスクグループになります。

p.1070 の「ルータビリティ」を参照してください。

bootdg **ブートディスクグループを(ルートボリュームの修復作業などのため)あるホストから別のホストに一時的に移動し、また、作業後に元に戻すには、次の手順を実行します。**

1. 元のホスト上で次のコマンドを実行し、インポートするディスクグループ bootdg のディスクグループ ID を確認します。

   ```
   # vxdisk -g bootdg -s list

   dgname: rootdg
   dgid: 774226267.1025.tweety
   ```

   この例では、管理者がブートディスクグループ名を rootdg としています。このディスクグループの ID は 774226267.1025.tweety です。

   この手順は、ブートディスクグループ内のすべてのディスクに両方のホストからアクセスできることを前提としています。

2. 元のホストを停止します。

3 インポート側のホスト上で次のコマンドを実行し、ディスクグループ rootdg のインポートおよび名前の変更を行います。

```
# vxdg -tC -n newdg import diskgroup
```

-t オプションはインポート時に一時的なディスクグループ名を使うことを示し、-C オプションはインポートロックを解除することを示します。-n オプションは、インポートする rootdg に代替名を設定して、既存の rootdg と競合しないようにします。*diskgroup* は、インポートするディスクグループのディスクグループ名またはディスクグループ ID（例: 774226267.1025.tweety）を指定します。

この時点で再起動またはクラッシュが発生すると、一時的にインポートされたディスクグループのインポートが解除され、再インポートが必要になります。

4 インポートしたディスクグループに対して必要な作業を完了したら、次のコマンドを実行してデポートし、元のホストに戻します。

```
# vxdg -h hostname deport diskgroup
```

ここで、*hostname* は、rootdg が返されるシステムの名前です（システム名は uname -n コマンドで確認できます）。

このコマンドは、インポートしたディスクグループをインポート側のホストから削除し、元のホストにロックを返します。元のホストは、次回の再起動時に自動的にこのブートディスクグループをインポートします。

競合する設定コピーの扱い方

不完全なディスクグループを複数の異なるシステムにインポートすると、ディスクグループの設定コピーに不整合が発生することがあります。この状態は手動で解消する必要があります。この項と次の項では、このような状態が発生するしくみと、この状態を修正する方法について説明します（分割したクラスタで生じるこのような状態のことを、一般にシリアルスプリットブレイン（Serial Split Brain）状態と呼びます）。

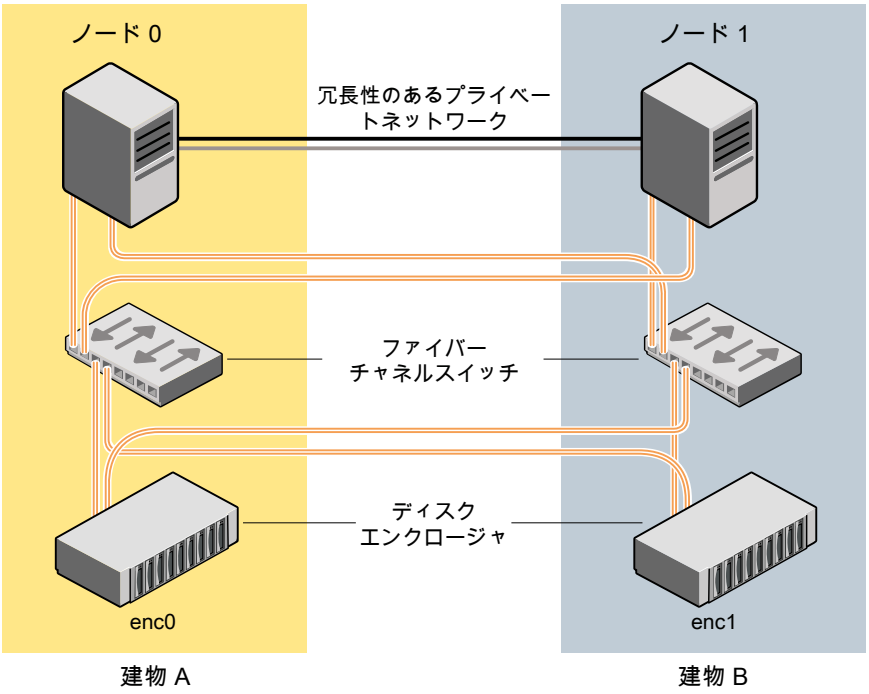
クラスタでのシリアルスプリットブレイン条件の例

この項では、クラスタ内の共有ディスクグループにシリアルスプリットブレイン条件がどのように発生するかについて説明します。別のシステムにディスクグループの一部をインポートした場合には、クラスタまたは非クラスタ設定の専用ディスクグループでも設定コピーに矛盾が生じることがあります。

通常、キャンパスクラスタ（別名ストレッチクラスタまたはリモートミラー設定）は、2 ノードクラスタで構成され、クラスタの各コンポーネント（サーバー、スイッチ、ストレージ）は別々の建物内に配置されます。

図 41-5 は、建物 A にノード 0、ファイバーチャネルスイッチ、ディスクエンクロージャ enc0、建物 B にノード 1、もう 1 つのスイッチ、エンクロージャ enc1 を配置した 2 ノードクラスタを示しています。

図 41-5　　2 ノードキャンパスクラスタの代表的な構成



各ノードと各エンクロージャの間の冗長ループアクセスを実現するために、ファイバーチャネル接続は多重冗長型になっています。また、2 つのノードは冗長性のあるプライベートネットワークでも接続されています。

シリアルスプリットブレイン条件が起きるのは、ノード 1 がノード 0 のフェールオーバーノードとして設定されたクラスタで、ノード 0 に専有（非共有）ディスクグループがインポートされた場合です。

ノード間のネットワーク接続が切断されると、両方のノードはそれぞれ他方のノードに障害が発生したものと認識します（これは、クラスタ環境でスプリットブレイン条件が生じる一般的な原因です）。1 つのディスクグループが 2 つのエンクロージャ、enc0 と enc1 の間で分割されると、各部とディスクグループの他の部分との接続が失われます。ノード 0 はアクセス可能なディスクグループ各部にあるディスクへの更新を続け、フェールオーバーノードとして機能するノード 1 は（-f オプションが設定されているため）ディスクグループの他の部分をインポートして、認識できるディスクの更新を開始します。

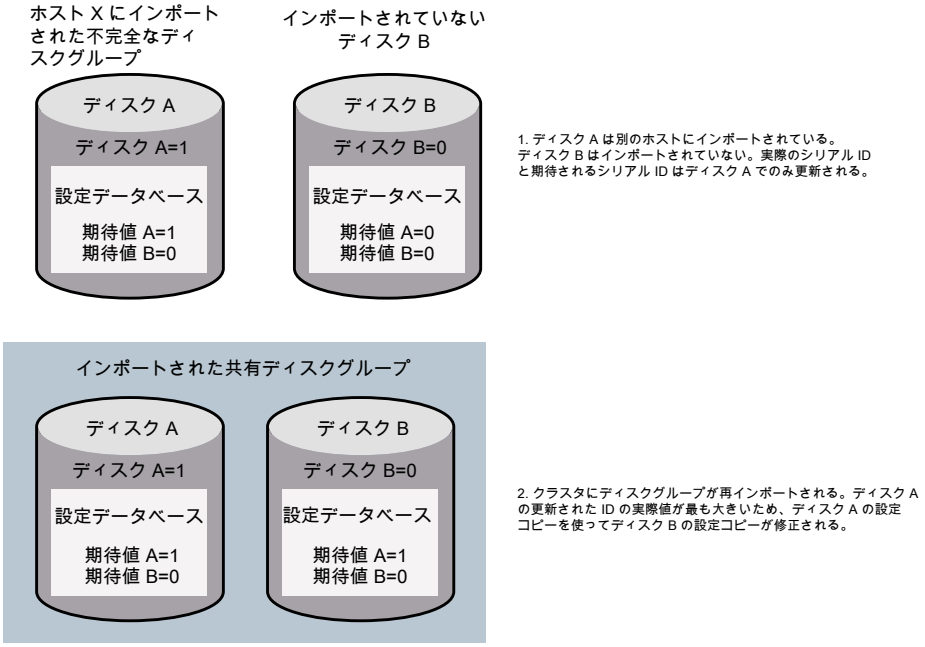
この状態でネットワークリンクが復元された場合、ノード 0 上のディスクグループに存在しないディスクを再接続したり、どちらかのノードにディスクグループ全体をインポートしようとしても、操作は失敗します。インポートされた各ディスクのディスクメディアレコードのシリアル ID は、これらのディスク上のすべてのディスクグループ設定データベースとインポートされた各ディスクのプライベートリージョンで、VxVM によって順に増分されます。設定データベースに保存される値は、ディスクグループに順に割り当てられていくシリアル ID であるのに対して、ディスクのプライベートリージョンに保存されるシリアル ID は、実際の値と見なされます。VxVM は、接続されているディスクの実際のシリアル ID と、インポートされたディスクグループのディスクグループ設定データベース内のシリアル ID が一致しない場合、シリアルスプリットブレインを検出します。

(物理的な切断や停電などによって)ディスクグループから一部のディスクが失われた後、別のホストによってこれらのディスクがインポートされると、設定データベースに保存されたディスクのコピーのシリアル ID と各ディスクのプライベートリージョンのシリアル ID は同じホスト上で別々に更新されます。この後で元の共有ディスクグループにディスクを再インポートすると、ディスクの実際のシリアル ID とディスクグループ内の他のディスクの設定コピーの期待値が一致しないという状態が生じます。

このような設定データベース間の不整合を解消するには、分割されたディスクグループの各部で何が起こったかに応じて 2 つの方法が考えられます。

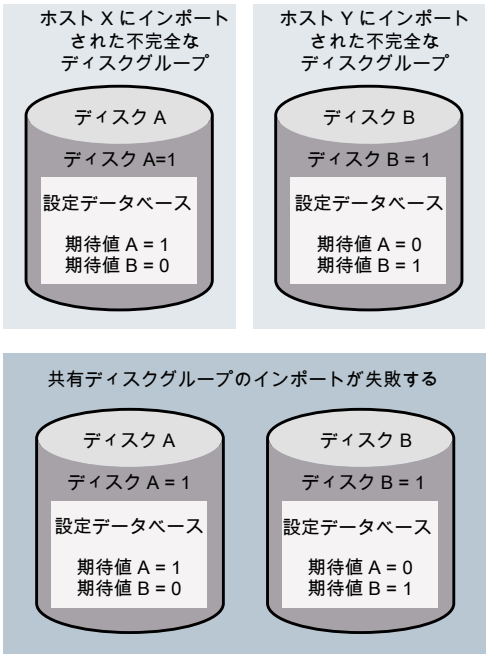
- ディスクグループの他方のディスクが別のホストにインポートされていない場合、VxVM は最も大きい更新済み ID(`vxdg list` *`diskgroup`* コマンドの出力の `update_id`)の値を持つディスクの設定データベースのバージョンを使って、シリアル ID の矛盾する値を調整します。
  図 41-6 に、VxVM による自動的な解消が可能なシリアルスプリットブレイン条件の例を示します。

図 41-6 自動的な解消が可能なシリアルスプリットブレイン条件の例



- 他方のディスクも別のホストにインポートされている場合、2 つのディスクの設定データベースはいずれも信頼できないものと見なされます。
  図 41-7 に、VxVM により自動的に解消できない、真のシリアルスプリットブレイン条件の例を示します。

図 41-7　　自動的に解消できない真のシリアルスプリットブレイン条件の例



この場合には、ディスクグループのインポートは失敗し、vxdg ユーティリティの終了前に次のようなエラーメッセージが表示されます。

```
VxVM vxconfigd NOTICE V-5-0-33 Split Brain. da id is 0.1, while dm
id
is 0.0 for DM mydg01
VxVM vxdg ERROR V-5-1-587 Disk group newdg: import failed: Serial
Split Brain detected. Run vxsplitlines
```

vxdg コマンドに -f フラグを指定しても、インポートは失敗します。

この場合でも、設定 ID（vxdg list *diskgroup*| grep config コマンドの出力の seqno の値として表示される）が最も大きい設定データベースを使うと、矛盾を解消できる場合があります。ただし、すべての状況で有効とはかぎりません。

p.1013 の「競合する設定情報の修正」を参照してください。

p.588 の「サイトとリモートミラーについて」を参照してください。

競合する設定情報の修正

設定情報の競合を解消するには、どちらのディスクのディスクグループ設定データベースが正しいかを判断する必要があります。これには、vxsplitlines コマンドを実行し、ディスクグループの各ディスクの実際のシリアル ID と設定データベースの期待されるシリアル ID を表示します。このコマンドを実行すると、それぞれのディスクについて実行可能な vxdg コマンドが表示されます。ディスクグループをインポートするために使う適切なコピーとなる設定データベースを選択するには、このコマンドを実行する必要があります。

メモ: ディスクグループのバージョン番号は、最低でも 110 である必要があります。

---

newdg ディスクグループで vxsplitlines コマンドを実行した場合の出力例を次に示します。

```
# vxsplitlines -v -g newdg

VxVM. vxsplitlines NOTICE V-0-0-0 There are 2 pools
All the disks in the first pool have the same config copies
All the disks in the second pool may not have the same config copies
```

ディスクから設定のコピーを参照するには、次のコマンドを入力します。

```
# /etc/vx/diag.d/vxprivutil dumpconfig private path
```

ディスクから設定のコピーと一緒にディスクグループをインポートするには、次のコマンドを入力します。

```
# /usr/sbin/vxdg (-s) -o selectcp=diskid import newdg

             Pool 0
DEVICE DISK DISK ID DISK PRIVATE PATH
newdg1 sdp 1215378871.300.vm2850lx13 /dev/vx/rdmp/sdp5
newdg2 sdq 1215378871.300.vm2850lx13 /dev/vx/rdmp/sdp5

              Pool 1
DEVICE DISK DISK ID DISK PRIVATE PATH
newdg3 sdo 1215378871.294.vm2850lx13 /dev/vx/rdmp/sdo5
```

-v オプションを指定しないと、コマンドの出力は次のようになります。

```
# vxsplitlines -g mydg listssbinfo

VxVM vxdg listssbinfo NOTICE V-0-0-0 There are 2 pools
All the disks in the first pool have the same config copies
```

```
All the disks in the second pool may not have the same config copies
Number of disks in the first pool: 1
Number of disks in the second pool: 1
```

最初のプールから設定のコピーと一緒にディスクグループをインポートするには、次のコマンドを入力します。

```
# /usr/sbin/vxdg (-s) -o selectcp=1221451925.395.vm2850lx13 import
mydg
```

2 番目のプールから設定のコピーと一緒にディスクグループをインポートするには、次のコマンドを入力します。

```
# /usr/sbin/vxdg (-s) -o selectcp=1221451927.401.vm2850lx13 import
mydg
```

この例では、ディスクグループは 4 つのディスクを持ち、分割の両方の側で 2 つのディスクが表れるように分割されます。

-c オプションを vxsplitlines に対して指定すると、ディスクのアクセス名によって指定されるディスク上の設定のコピーから、各ディスク ID についての詳しい情報を出力できます。

```
# vxsplitlines  -g newdg -c sde

DANAME(DMNAME) || Actual SSB      || Expected SSB
sdd( sdd ) || 0.1                 || 0.0 ssb ids don't match
sde( sde ) || 0.1                 || 0.1 ssb ids match
sdf( sdf ) || 0.1                 || 0.1 ssb ids match
sdg( sdg ) || 0.1                 || 0.0 ssb ids don't match

Please note that even though some disks ssb ids might match
that does not necessarily mean that those disks' config copies
have all the changes. From some other configuration copies,
those disks' ssb ids might not match. To see the configuration
from this disk, run
/etc/vx/diag.d/vxprivutil dumpconfig /dev/vx/dmp/sde
```

シリアルスプリットブレイン条件がどのように生じたかに基づいて、ディスクグループのインポートに使われる 1 つのディスク設定を選択する必要があります。たとえば、次のコマンドでは、分割の 0 側にある設定のコピーを使ってディスクグループがインポートされます。

```
# /usr/sbin/vxdg -o selectcp=1045852127.32.olancha import newdg
```

優先される設定のコピーを選択するときに、ディスクグループがインポート済みの場合、VxVM によってインポートされたシリアル ID が 0 にリセットされます。その時点でインポー

トされていないディスクグループ内のすべてのディスクの実際のシリアル ID と予想されるシリアル ID は、変更されないままになります。

ディスクグループの無効化

ディスクグループを無効にするには、ディスクグループ内のすべてのボリュームのマウントを解除して停止し、次のコマンドを使ってディスクグループをデポートします。

```
# vxdg deport diskgroup
```

ディスクグループを無効にしても、ディスクグループは実際には削除されません。この操作では、システムによるディスクグループの使用が無効にされます。デポートされたディスクグループ内のディスクは再使用または再初期化することができ、また、他のディスクグループに追加したり、他のシステムにインポートして使うことも可能です。ディスクグループへのアクセスを再び有効にするには、`vxdg import` コマンドを使います。

ディスクグループの破棄

`vxdg` コマンドは、システムからディスクグループを削除し、再初期化設定のためそのディスクグループ内のディスクを解放する `destroy` オプションを提供します。

```
# vxdg destroy diskgroup
```

警告: このコマンドは、ディスクのすべてのデータを破棄します。

ディスクグループが破棄されると、解放されたディスクは他のディスクグループで再利用することができます。

破棄されたディスクグループのリカバリ

ディスクグループが誤って破棄された場合、ディスクグループのディスクが他で変更または再利用されていなければ、リカバリできます。

破棄されたディスクグループをリカバリするには

1 次のコマンドを入力して、ディスクグループ内のいずれかのディスクのグループ ID (dgid)を確認します。

```
# vxdisk -s list disk_access_name
```

ディスクは、sdc などのディスクアクセス名で指定する必要があります。コマンドの出力で、ディスクグループ ID を示す次のような行を探します。

```
dgid:   963504895.1075.bass
```

2 そのディスクグループ ID を使ってディスクグループをインポートします。

```
# vxdg import dgid
```

ディスクグループ設定データのバックアップとリストア

ディスクグループ設定のバックアップとリストアの機能を実行すると、ディスクグループと、ディスクグループ内に設定されたボリュームなどの VxVM オブジェクトの設定データすべてをバックアップおよび復元できます。vxconfigbackupd デーモンは、VxVM 設定の変更を監視し、設定が変更されると自動的に設定変更を記録します。デフォルトでは、vxconfigbackup は設定のバックアップとリストア(cbr)データのコピーを 5 つ保存します。cbr コピーの数は 1 ～ 5 コピーの間でカスタマイズできます。

vxconfigbackupd(1M)マニュアルページを参照してください。

VxVM では、ディスクグループの VxVM 設定のバックアップおよびリストア用に、vxconfigbackup と vxconfigrestore というユーティリティが用意されています。

詳しくは、『Veritas InfoScale トラブルシューティングガイド』を参照してください。

vxconfigbackup(1M)マニュアルページを参照してください。

vxconfigrestore(1M)マニュアルページを参照してください。

FSS ディスクグループ設定データのバックアップと復元

ディスクグループ設定のバックアップと復元機能を使うと、FSS(Flexible Storage Sharing)ディスクグループの設定データをバックアップして復元することもできます。vxconfigbackupd デーモンにより、すべてのクラスタノードに行われた設定変更がすべて自動的に記録されます。FSS ディスクグループの設定データを復元する場合は、最初にクラスタ内のセカンダリ(スレーブ)ノードにある設定データを復元し、ローカルに接続されたすべてのディスクをエクスポートしてリモートディスクを作成する必要があります。セカンダリノードの設定データを復元した後、ディスクグループをインポートするプライマリ(マスター)ノードにある設定データを復元する必要があります。

FSS ディスクグループの設定データをバックアップするには

- ディスクグループ内の 1 つ以上のディスクに接続されたすべてのクラスタノードで FSS ディスクグループの設定データをバックアップするには、次のコマンドを入力します。

```
# /etc/vx/bin/vxconfigbackup -T diskgroup
```

FSS ディスクグループの設定データを復元するには

1. マスターノードを識別します。

```
# vxclustadm nidmap
```

2. プライマリノードがディスクグループ内の 1 つ以上のディスクに接続されているかどうかを調べます。ディスクには DAS（Direct Attached Storage）ディスク、部分的に共有されているディスク、または完全に共有されているディスクを指定できます。
3. プライマリノードがディスクグループ内のどのディスクにも接続されていない場合は、次のコマンドを使って、1 つ以上の DAS または部分的に共有されたディスクに接続されたノードにプライマリノードを切り替えます。

```
# vxclustadm setmaster node_name
```

4. すべてのセカンダリノードで設定データを復元します。

```
# vxconfigrestore diskgroup
```

メモ: ディスクグループ内の 1 つ以上のディスクに接続されていないすべてのセカンダリノードの設定データを復元する必要があります。

---

5. プライマリノードの設定データを復元します。

```
# vxconfigrestore diskgroup
```

6. 設定データを確認します。

```
# vxprint -g diskgroup
```

7. 設定データが正しい場合は、設定をコミットします。

```
# vxconfigrestore -c diskgroup
```

FSS ディスクグループの設定のリストアを中止またはコミット解除するには

1 マスターノードを識別します。

```
# vxclustadm nidmap
```

2 マスターノードの設定データを中止またはコミット解除します。

```
# vxconfigrestore -d diskgroup
```

3 すべてのセカンダリノードの設定データを中止またはコミット解除します。

```
# vxconfigrestore -d diskgroup
```

メモ: ディスクグループ内の 1 つ以上のディスクに接続されているすべてのセカンダリノードと、プレコミットをトリガしたすべてのセカンダリノードの設定データを中止またはコミット解除する必要があります。

『Veritas InfoScale 7.4.2 トラブルシューティングガイド』を参照してください。

`vxconfigbackup`(1M)マニュアルページを参照してください。

`vxconfigrestore`(1M)マニュアルページを参照してください。

既存の ISP ディスクグループの使用

Veritas Volume Manager(VxVM)の ISP(Intelligent Storage Provisioning)機能は非推奨になりました。このリリースでは、ISP ディスクグループの作成はサポートされていません。既存の ISP ディスクグループがある場合は、ディスクグループのバージョンをアップグレードしないでディスクグループをインポートできます。この場合、設定が変更されるような ISP ボリュームの操作は実行できません。さらに、アップグレードされたディスクグループのバージョンが必要になる現行リリースの機能はいずれも使うことができません。

ISP ディスクグループを現在のディスクグループのバージョンにアップグレードできます。この操作では、すべての ISP ボリュームが標準の(非 ISP)ボリュームに変換され、ISP 固有のオブジェクトは削除されます。st プール、ボリュームテンプレート、ケイパビリティ、ルールなどが ISP 固有のオブジェクトです。この操作は非 ISP ボリュームには影響を与えません。

メモ: ISP ディスクグループをアップグレードすると、インテントとストレージプールの情報がすべて失われます。この条件が受け入れ可能なときのみ、ディスクグループをアップグレードしてください。

ディスクグループが ISP ディスクグループかどうかを判断するには

◆ 次のコマンドを使ってストレージプールの存在を確認します。

```
# vxprint
```

サンプル出力:

```
Disk group: mydg
TY NAME       ASSOC        KSTATE  LENGTH   PLOFFS STATE     TUTIL0 PUTIL0
dg mydg       mydg         -       -        -      ALLOC_SUP -      -
dm mydg2      ams_wms0_359 -       4120320  -      -         -      -
dm mydg3      ams_wms0_360 -       4120320  -      -         -      -
st mypool     -            -       -        -      DATA      -      -
dm mydg1      ams_wms0_358 -       4120320  -      -         -      -
v  myvol0     fsgen        ENABLED 20480    -      ACTIVE    -      -
pl myvol0-01 myvol0        ENABLED 20480    -      ACTIVE    -      -
sd mydg1-01   myvol0-01    ENABLED 20480    0      -         -      -
v  myvol1     fsgen        ENABLED 20480    -      ACTIVE    -      -
pl myvol1-01 myvol1        ENABLED 20480    -      ACTIVE    -      -
sd mydg1-02   myvol1-01    ENABLED 20480    0      -         -      -
```

出力例では、`st mypool` が、`mydg` は ISP ディスクグループであることを示しています。

ISP ディスクグループをアップグレードするには

◆ 次のコマンドを使って、ISP ディスクグループをアップグレードします。

```
# vxdg upgrade ISP_diskgroup
```

ISP ディスクグループをそのまま使うには

◆ ISP ディスクグループをインポートするには、次のコマンドを使います。

```
# vxdg import ISP_diskgroup
```

ディスクグループ内の ISP ボリュームは、ディスクグループがアップグレードされるまで、すべての設定変更が許可されません。ISP ボリュームに対して拡張/縮小、ミラーの追加、ディスクグループの分割/結合などの操作を試みると、次のエラーが表示されます。

```
This disk group is a ISP disk group. Dg needs to be migrated to
non-ISP dg to allow any configuration changes. Please upgrade
the dg to perform the migration.
```

メモ: ISP ディスクグループ内の非 ISP ボリュームや VxVM ボリュームは影響を受けません。

アップグレードしないで ISP ディスクグループに対してまだ機能する操作は次のとおりです

- ボリュームタグの設定、削除、置換。
- ボリューム、dg、プレックスなどの VxVM オブジェクトの名前変更。
- プレックスの接続と切断。
- インテント情報は失われますが `vxconfigbackup` と `vxconfigrestore` のコマンドを使えます。

プレックスとサブディスクの管理

ここでは、プレックスとサブディスクの管理について説明します。

サブディスクは連続した一連のディスクブロックです。VxVM は、サブディスクを使ってディスク領域を割り当てます。

プレックスとは、物理ディスクサイズなどの制約を受けないディスク領域エリアを作成する、サブディスクの論理集合です。ディスクデータの複製（ミラー）は、1 つのボリュームに複数のデータプレックスを作成することによって設定されます。ミラーボリューム内の各データプレックスには、ボリュームデータと同一のコピーが格納されます。

プレックスをボリュームに接続すると、そのボリュームに対して有効なプレックスになります プレックスを接続すると、プレックスとボリュームが関連付けられ、プレックスが使用可能になります。

プレックスの再接続

ミラープレックスで回復不能なエラーが起きた場合、Veritas Volume Manager（VxVM）はミラーボリュームからプレックスを切断します。管理者は、vxplex や vxassist などのユーティリティを使用してプレックスを手動で切断する場合もあります。以前にボリュームに接続されたプレックスを使用するには、プレックスはボリュームに再接続されている必要があります。また、再接続操作では、プレックスミラーがボリューム内の他のプレックスに再同期されていることも確認できます。

p.1023 の「プレックスの同期」を参照してください。

プレックスを再接続するには、次の方法を使用できます。

- デフォルトでは、VxVM は障害が発生した下位のディスクまたは LUN が認識可能になったときに影響があるミラープレックスを自動的に再接続します。デバイスがオンラインであることが VxVM で検出されると、VxVM は関係する LUN のボリュームコンポーネントを自動的にリカバリします。VxVM はプレックスを再同期し、ミラーが使用可能になります。
  p.1021 の「プレックスの自動再接続」を参照してください。

- 自動再接続機能が無効の場合、プレックスを手動で再接続する必要があります。また、自動的に再接続されないデバイスのプレックスを手動で再接続する必要がある場合もあります。たとえば、VxVM では、サイトの一貫性が設定されたボリューム上のプレックスは自動的に再接続されません。
  p.1022 の「手動によるプレックスの再接続」を参照してください。

プレックスの自動再接続

ミラープレックスで回復不能なエラーが起きた場合、Veritas Volume Manager（VxVM）はミラーボリュームからプレックスを切断します。デフォルトでは、VxVM は障害が発生した下位のディスクまたは LUN が認識可能になったときに影響があるミラープレックスを自動的に再接続します。デバイスがオンラインであることを VxVM が検出すると、関係する LUN 上の VxVM ボリュームコンポーネントが自動的にリカバリされ、ミラーが使用可能になります。

VxVM は障害が発生した LUN の DMP プローブを使って、デバイスがいつオンラインになったかを検出します。再接続のタイミングは、チューニングパラメータである `dmp_restore_interval` によって決まります。再接続した LUN の数は、プレックスが再接続されるのに必要な時間にも影響することがあります。

VxVM では、サイトの一貫性が設定されたボリューム上のプレックスは自動的に再接続されません。

VxVM がインストールされるか、システムが再ブートされると、VxVM は `vxattachd` デーモンを起動します。`vxattachd` デーモンはプレックスとサイト両方の自動再接続を処理します。`vxattachd` デーモンはプレックスの再同期プロセスも開始します。プレックスが正常に再接続されると、`vxattachd` が root への通知を行います。

プレックスの自動接続を無効にするには、起動スクリプトから `vxattachd` の記述を削除します。`vxattachd` を無効にすると、プレックスとサイト両方の自動再接続機能が無効になります。

Cluster Volume Manager（CVM）では、次の注意事項があります。

- グローバル切断ポリシーが設定されている場合、任意のノードでストレージの障害が起きると、そのストレージにあるすべてのプレックスが一括で切断されます。ストレージが任意のノードに再接続されたときに `vxattachd` デーモンが再接続を開始するのは、マスターノードのプレックスのみです。
- 自動再接続機能はノードに対してローカルです。この機能がノードで有効にされると、そのノードでインポートされたすべてのディスクグループが監視されます。自動再接続機能がマスターノードで無効にされると、マスターノードでインポートされたすべての共有ディスクグループと専用ディスクグループで機能が無効になります。
- `vxattachd` デーモンは、`vxnotify` を使って操作をトリガするために、`dmpnode` online イベントの応答を準備します。このため、`vxattachd` の動作中は `dmpnode` online イ

ベントが生成されないかぎり、自動再接続がトリガされません。一般的な例を次に示します。

- ストレージは vxattachd が起動される前、たとえば起動中に、再接続されます。
- CVM のアクティブ/パッシブアレイでは、アレイコントローラに接続するパスの設定がすべてのノードで共通していないと、I/O 障害のためにプレックスが切断される可能性があります。このような場合、dmpnode は無効にされません。したがって、接続がリストアされた後、dmpnode online イベントは生成されず、プレックスの自動再接続はトリガされません。

これらの CVM の注意事項は、サイトの自動再接続にもあてはまります。

p.608 の「サイトの自動再接続」を参照してください。

手動によるプレックスの再接続

ここでは、自動再接続機能が無効になっている場合に、プレックスを手動で再接続する方法について説明します。この手順はまた、自動的に再接続されないデバイスに必要になることがあります。たとえば、VxVM では、サイトの一貫性が設定されたボリューム上のプレックスは自動的に再接続されません。

ディスクの修復や交換が完了し、再度使えるようになったら、プレックスを再びオンラインに設定(プレックスを ACTIVE 状態に設定)する必要があります。プレックスを ACTIVE 状態に設定するには、ボリュームの状態に応じて次の手順のいずれかを使います。

- ボリュームが現在 ENABLED カーネル状態である場合、次のコマンドを使ってプレックスを再接続します。

  ```
  # vxplex [-g diskgroup] att volumeplex ...
  ```

  たとえば、プレックス vol01-02 を、ディスクグループ mydg 内のボリューム vol01 に接続するには、次のコマンドを使います。

  ```
  # vxplex -g mydg att vol01 vol01-02
  ```

  OFFLINE 状態のプレックスが再び ACTIVE 状態になると、このコマンドによりプレックスの内容のリカバリが開始され、リカバリが完了すると、プレックスは ACTIVE 状態に設定されます。

- ボリュームが使われていない(ENABLED カーネル状態ではない)場合、次のコマンドを使ってプレックスが使えるように再度有効にします。

  ```
  # vxmend [-g diskgroup] on plex
  ```

  たとえば、ディスクグループ mydg 内のプレックス vol01-02 を再度有効にするには、次のように入力します。

  ```
  # vxmend -g mydg on vol01-02
  ```

この場合、vol01-02 は STALE 状態に設定されます。ボリュームを次回起動するときには、プレックス上のデータが別のプレックスから修復され、プレックスが ACTIVE 状態に設定された状態でボリュームに組み込まれます。
vxinfo コマンドによりボリュームが Unstartable と表示された場合、次のコマンドを使ってプレックスのいずれかを CLEAN 状態に設定します。

```
# vxmend [-g diskgroup] fix clean plex
```

次のコマンドを使ってボリュームを起動します。

```
# vxvol [-g diskgroup] start volume
```

詳しくは、『Veritas InfoScale トラブルシューティングガイド』を参照してください。

プレックスの同期

ボリュームの各プレックスまたはミラーは、データの完全な複製です。プレックスがボリュームに接続されると、このプレックスのデータを、ボリューム内の他のプレックスのデータと同期する必要があります。接続されているプレックスは、新しいミラーまたは以前接続されたプレックスの場合があります。新しいミラーは完全に同期されている必要があります。以前に接続されたプレックスは、プレックスが切断されたため、適用された変更のみを必要とします。

次の操作は、プレックスの同期をトリガします。

- vxsd コマンドでサブディスクを移動またはコピーする。この操作は、元のサブディスクと同期される一時プレックスを作成します。
- vxassist mirror コマンドでミラーを追加する。
- vxassist make コマンドでミラーを使用するボリュームを作成する。
- vxplex att コマンドでプレックスを手動で再接続する。
- vxrecover コマンドでボリュームをリカバリする。
- vxsnap addmir コマンドでスナップショットにミラーを追加する。
- vxsnap コマンドでスナップショットを再接続またはリストアする。

プレックスの同期は、同期する必要があるボリュームのサイズとデータの量によっては、時間のかかる操作になります。Veritas Volume Manager は、プレックスの同期の効率を向上させるために、いくつかの機能を提供します。

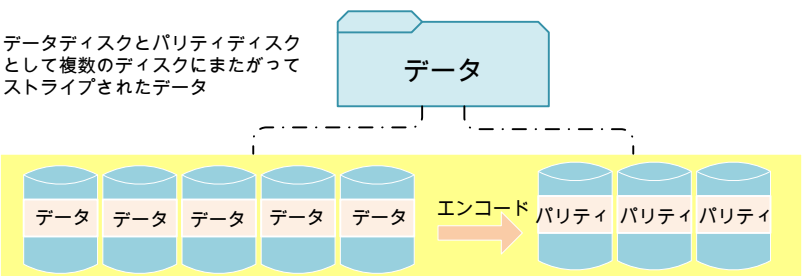
- FastResync
  FastResync 機能が有効な場合、VxVM はボリューム上の FastResync のマップを保守します。VxVM は、ミラーに反映されなかった更新のみを適用するために FastResync マップを使用します。この動作はプレックスを再同期する効率的な方法を提供します。

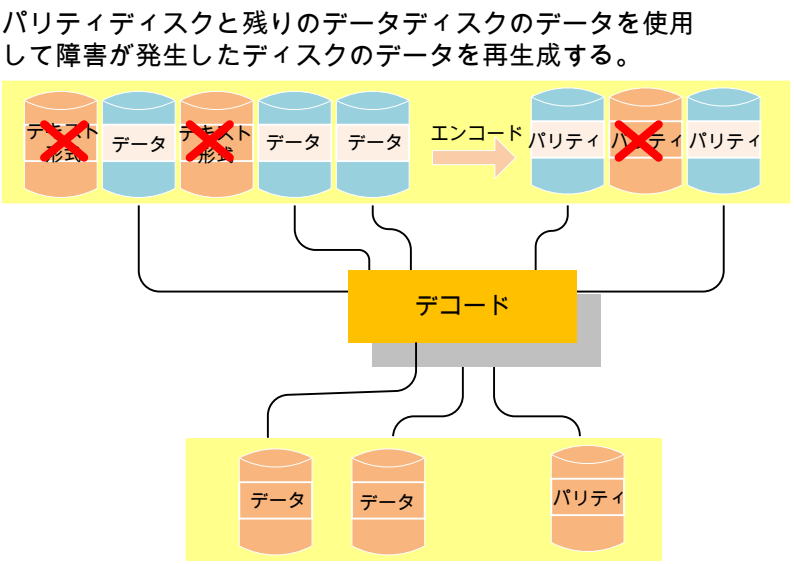
- SmartMove
  マウントされた VxFS ファイルシステムでは、SmartMove™ 機能を使うと、VxVM のボリュームにプレックスを接続または再接続するために必要な時間と I/O が削減されます。SmartMove 機能では、VxFS 情報を使って空きエクステントが検出され、空きエクステントがコピーされません。
  SmartMove 機能が有効になっている場合、ホストとストレージネットワークを経由してディスクまたは LUN に送信される I/O は減少します。SmartMove 機能を使えば、プレックスの作成やアレイの移行を短時間で行うことができます。
- 同期タスクのリカバリ
  このリリースでは、VxVM は次のコマンドのプレックスの同期を追跡します。`vxplex att`、`vxassist mirror`、`vxsnap addmir`、`vxsnap reattach`、`vxsnap restore`。システムがクラッシュした場合、または `vxconfigd` デーモンが失敗した場合、VxVM は同期タスクに自動リカバリを提供します。システムがリカバリされると、VxVM は失敗した時点から同期を再開します。同期はバックグラウンドで実行されるため、ボリュームは遅延なく使用できます。

Veritas InfoScale Storage 環境の Erasure coding

Erasure coding は、スタンドアロン環境と FSS 環境でサポートされます。

ストレージシステムが拡大し、複雑性が増すにつれ、従来のデータ保護機構ではエラーに対して不十分であることがわかってきています。Erasure coding は重要なストレージアーカイブの冗長性と耐障害性において、より堅牢なソリューションを提供します。Erasure coding ではデータを断片化して拡張し、冗長データでエンコードして異なる場所またはストレージメディアに格納します。1 つ以上のディスクでエラーが発生した場合、エラーが発生したディスク上のデータは、エンコードされたディスクのパリティ情報と、残りのディスクのデータを使って再構築されます。Erasure Code ボリュームは、ディスクグループバージョン 240 以降を使用して作成する必要があります。





Veritas InfoScale では、Reed-Solomon アルゴリズムを使用して Erasure coding を構築します。

Erasure coding の使用例

Erasure Code ボリュームは、次の使用例で使用できます。

- トランザクションストアやファイルストアなどの一般的な作業負荷用デバイスとしての EC (Erasure Coded) ボリューム。
- Erasure Code (EC) ボリュームは、オブジェクトストアなどの特別な作業負荷に対して最適なパフォーマンスに合わせてチューニングされます。

一般的な用途 (トランザクション作業負荷を含む)

他のボリュームレイアウトと同じように、トランザクションやその他の作業負荷用デバイスとして EC ボリュームを使用できます。この場合、システムがクラッシュしたときにデータの一貫性を確保するために、非同期の書き込みログが使われます。各書き込みは、まずログに記録または書き込まれてから通知されます。その後、ログからの書き込みは、非同期的にデータ領域にフラッシュされます。ユーザーは、`vxassist` に ecloglen オプションを使用して log_length を指定できます。これは、カラムごとのログサイズを指定します。ログの長さは、並列書き込み数の関数である必要があります。ecloglen のデフォルト値は 1 GB です。

システムがクラッシュ後にオンライン状態になると、ログ再生が実行されます。書き込み側ノード (ホスト) の一部がクラッシュして復帰した場合、そのログのすべての書き込みは FMR で追跡され、追跡された領域がリカバリされます。クラッシュが発生した場合は、このようにしてデータの一貫性が維持されます。障害が発生し、そのことがログに書き込ま

れただけのカラムへの書き込みに対応する領域は、カラムのリカバリを開始する前にダーティとしてマークされます。

オブジェクトストア

オブジェクトストアでは、データはオブジェクトとして管理および格納されます。これは、データがファイル階層やディスクセクタまたはブロックとして管理および格納される、その他のアーキテクチャとは異なります。標準的なオブジェクトストアアーキテクチャでは、次の条件が保証されます。

- オブジェクトストアへのデータの保存中に、新しいオブジェクトが作成されます。
- データのインプレースでの変更は許可されません。古いオブジェクトは、新しいオブジェクトとして読み込まれて、変更および格納されて、古いオブジェクトは削除されます。

EC ボリュームは、このような使用例で最適な状態に調整できます。この場合に EC ボリュームを使用するには、ユーザーは stripe_aligned=yes オプションを指定して EC ボリュームを作成する必要があります。EC ボリュームを作成するときに「stripe_aligned = yes」オプションを指定すると、ログとロックのオーバーヘッドがなくなり、パフォーマンスが向上します。また、オブジェクトの割り当てを EC ストライプの長さと揃える必要があることと、オブジェクトサイズを EC ストライプの長さと同じにするか、EC ストライプの長さの倍数にする必要があることに注意してください。

このリリースでサポートされる RHEL、SLES、サポート対象の RHEL 互換配布のバージョンでは、Erasure Code ボリュームを設定できます。

分散パリティの使用

分散パリティを使用して EC ボリュームを作成するときは、専用のデータやパリティ列は存在せず、すべてのノードのストレージ帯域幅を最適な利用を実現します。Erasure Code ボリュームは、デフォルトでは分散パリティを使用して作成されます。ただし、専用のパリティを使用するには、ボリュームの作成中に vxassist コマンドで ecrotate = no オプションを指定します。

例: 分散パリティ

一般的な用途の使用例 (トランザクション作業負荷など) 向けに 1 GB の Erasure Code ボリューム vol1 (ディスク 1 台の障害を許容し、2 台のディスクにデータをストライプ化) を作成し、分散パリティディスクを使用する場合は、次のコマンドを実行します。

```
# vxassist -g sfsdg make ecvol1 1g layout=ecoded ncols=2 nparity=1
```

この設定を表示するには

```
# vxprint -g sfsdg ecvol1
TY NAME                    ASSOC     KSTATE    LENGTH   PLOFFS  STATE  TUTIL0 PUTIL0
v  ecvol1                   fsgen     ENABLED   2097152    -     SYNC     -      -
```

```
pl ecvol1-01                    ecvol1    ENABLED   2097152    -     ACTIVE   -       -
sd sflrcluster_04_disk_2-07     ecvol1-01 ENABLED   409600     0     ECLOG    -       -
sd sflrcluster_04_disk_2-06     ecvol1-01 ENABLED   1048576    0       -      -       -
sd sflrcluster_03_disk_0-07     ecvol1-01 ENABLED   409600     0     ECLOG    -       -
sd sflrcluster_03_disk_0-06     ecvol1-01 ENABLED   1048576    0       -      -       -
sd sflrcluster_02_disk_2-08     ecvol1-01 ENABLED   409600     0     ECLOG    -       -
sd sflrcluster_02_disk_2-07     ecvol1-01 ENABLED   1048576    0       -      -       -
dc ecvol1_dco                   ecvol1       -        -        -       -      -       -
v  ecvol1_dcl   gen                       ENABLED   67840      -     ACTIVE   -       -
pl ecvol1_dcl-01                ecvol1_dclENABLED   67840            ACTIVE   -       -
sd sflrcluster_04_disk_2-08 ecvol1_dcl-01ENABLED   67840      0       -      -       -
pl ecvol1_dcl-02              ecvol1_dcl ENABLED   67840      -     ACTIVE   -       -
sd sflrcluster_03_disk_0-08 ecvol1_dcl-02ENABLED   67840      0       -      -       -
```

この例では、分散パリティはデフォルトのオプションであるため、特定のパリティカラムはありません。

上記の例では、専用のパリティカラムが存在するErasure Codeボリュームを作成するには、次のように vxassist コマンドで ecrotate = no オプションを指定する必要があります。

```
# vxassist -g sfsdg make ecvol2 1g layout=ecoded ncols=2 nparity=1
ecrotate=no
```

この設定を表示するには

```
# vxprint -g sfsdg ecvol2

TY NAME                     ASSOC        KSTATE   LENGTH   PLOFFS   STATE   TUTIL0
PUTIL0
v  ecvol2                   fsgen        ENABLED  2097152  -        SYNC      -
-
pl ecvol2-01                ecvol2       ENABLED  2097152  -        ACTIVE    -
-
sd sflrcluster_04_disk_2-10 ecvol2-01    ENABLED  409600   0        ECLOG     -
-
sd sflrcluster_04_disk_2-09 ecvol2-01    ENABLED  1048576  0         -        -
-
sd sflrcluster_03_disk_0-10 ecvol2-01    ENABLED  409600   0        ECLOG     -
-
sd sflrcluster_03_disk_0-09 ecvol2-01    ENABLED  1048576  0         -        -
-
sd sflrcluster_02_disk_2-10 ecvol2-01    ENABLED  409600   0        ECLOG     -
-
```

```
sd sflrcluster_02_disk_2-09 ecvol2-01    ENABLED  1048576   0       PARITY    -
-
dc ecvol2_dco               ecvol2         -        -       -        -        -
-
v  ecvol2_dcl               gen            ENABLED  67840     -       ACTIVE    -
-
pl ecvol2_dcl-01             ecvol2_dcl  ENABLED  67840     -       ACTIVE    -
-
sd sflrcluster_04_disk_2-11 ecvol2_dcl-01ENABLED  67840     0          -      -
-
pl ecvol2_dcl-02            ecvol2_dcl   ENABLED  67840     -       ACTIVE    -
-
sd sflrcluster_03_disk_0-11 ecvol2_dcl-02ENABLED  67840     0          -      -
-
```

この例では、State カラムにパリティ列が表示されています。

別のディスクへのログの割り当て

デフォルトでは、Erasure Code ボリュームのカラムレベルのログのログサブディスクは、すべてのカラムの最初のデータディスクに割り当てられます。そのため、すべてのカラムの最初のディスクでは、ログサブディスクとデータサブスクリプションディスクの両方が割り当てられ、そのためにディスクのボトルネックが生じる可能性があります。ディスクのボトルネックを避けるには、プロビジョニングを追加し、`eclogdisk` オプションを使用して別のディスクにログを割り当てます。ディスクのカンマ区切りのリストを指定する必要があります。ここで、ディスクの数は、データとパリティカラムの合計数以上になります。また、すべてのディスクが異なる障害ドメインに属することを確認する必要があります。

例: 3 つのデータカラムと 2 つのパリティカラムが存在する、Erasure Code ボリュームの構成を検討してみましょう。ここで、障害ドメインはホスト (FSS 環境でのデフォルト) で、ホストの数は 5 です。この場合、ユーザーは 5 つの異なるホストに存在する 5 つのログディスクを指定する必要があります。

例: 別のディスク上のログ

1 GB の Erasure Code ボリューム `vol1` (1 つのホストにエラーに対する耐性があり、2 つのホストにデータをストライプ化 (EC 構成 2、1)) を作成し、別のディスクにログを割り当てる場合は、次のコマンドを実行します。

```
# vxassist -g sfsdg make ecvol1 1g layout=ecoded ncols=2 nparity=1
eclogdisk=sflrcluster_02_disk_2,sflrcluster_03_disk_2,sflrcluster_04_disk_2
```

ボリュームの設定を表示するには

```
# vxprint -g sfsdg ecvol1
```

```
TY NAME         ASSOC        KSTATE   LENGTH   PLOFFS   STATE    TUTIL0  PUTIL0
v  ecvol1       fsgen        ENABLED  2097152  -        SYNC     -       -
pl ecvol1-01    ecvol1       ENABLED  2097152  -        ACTIVE   -       -
sd sflrcluster_04_disk_2-06 ecvol1-01 ENABLED 409600 0  ECLOG    -       -
sd sflrcluster_04_disk_3-03 ecvol1-01 ENABLED 1048576 0 -        -       -
sd sflrcluster_03_disk_2-03 ecvol1-01 ENABLED 409600 0  ECLOG    -       -
sd sflrcluster_03_disk_0-06 ecvol1-01 ENABLED 1048576 0 -        -       -
sd sflrcluster_02_disk_2-07 ecvol1-01 ENABLED 409600 0  ECLOG    -       -
sd sflrcluster_02_disk_3-03 ecvol1-01 ENABLED 1048576 0 -        -       -
dc ecvol1_dco   ecvol1       -        -        -        -        -       -
v  ecvol1_dcl   gen          ENABLED  67840    -        ACTIVE   -       -
pl ecvol1_dcl-01 ecvol1_dcl  ENABLED  67840    -        ACTIVE   -       -
sd sflrcluster_04_disk_3-04 ecvol1_dcl-01 ENABLED 67840 0 -      -       -
pl ecvol1_dcl-02 ecvol1_dcl  ENABLED  67840    -        ACTIVE   -       -
sd sflrcluster_03_disk_0-07 ecvol1_dcl-02 ENABLED 67840 0 -      -       -
```

この例では、コマンドラインで `eclogdisk` オプションが指定され、ログは次のディスクに割り当てられます。

```
sflrcluster_02_disk_2,sflrcluster_03_disk_2,sflrcluster_04_disk_2
```

Erasure Code ボリュームの制限事項

Erasure Code ボリュームのサポートは、次の制約によって制限されます。

- 次の管理操作が Erasure Code ボリュームでまだサポートされていません。
  - ミラーの追加と削除
  - スナップショットと関連する操作
  - ディスクグループにより Erasure Code ボリュームでの操作が分割および移行されます。
  - Erasure Code ボリュームの再レイアウト
- シンプロビジョニングディスクまたは再生利用可能 (TP/R) なディスクでは、Erasure Code ボリュームを作成できません。再生すると Erasure Code ボリュームが壊れる可能性があります。

Erasure coding 配備シナリオ

Erasure coding は次の配備シナリオをサポートします。

- シングルノードの SAN
- クラスタ化された DAS

ただし、クラスタ化された SAN 環境ではサポートされません。

DAS または SAN のストレージを使用して 1 つのノード上に Erasure Code ボリュームを作成する

この手順は、ディスクアレイを n 個の LUN に *d1*、*d2*、... *dn* のようにエクスポートすることを想定しています。

DAS または SAN のストレージを使用して 1 つのノードに Erasure Code ボリュームを作成するには

1 必要なディスクセットを含むディスクグループがまだ存在しない場合は、ディスクグループ (dg1) を作成します。

```
# vxdg init dg1 <d1 d2 ... dn>
```

2 ディスクグループ内の **<k>** 台のディスクにデータをストライプ化し、**<m>** 台までの障害を許容する Erasure Code ボリューム (vol1) を作成します。

```
# vxassist -g dg1 make vol1 <size> layout=ecoded ncol=K nparity=m
```

Erasure Code ボリュームの作成時にストライプグループとストライプ制限グループを指定します。p.1046 の「Erasure Code ボリューム作成時のストライプグループとストライプ制限グループの使用」を参照してください。

ディスクグループ内の特定のディスクをボリュームに使用する場合は、次のようにディスクを指定できます。

```
# vxassist -g dg1 make vol1 <size> layout=ecoded ncol=k
nparity=md1 d2 ... dn
```

次の設定例では、各サイズが 256 GB の 10 台のディスクを含むディスクグループ dg1 を使用します。

```
# vxprint
Disk group: dg1

TY   NAME                    ASSOC             KSTATE   LENGTH   PLOFFS STATE TUTIL0 PUTIL0
dg   dg1                      dg1                -        -        -      -     -       -

dm vmr720-18vm3_vmdk0_0  vmr720-18vm3_vmdk0_0 -      4128464  -      -     -       -
dm vmr720-18vm3_vmdk0_1  vmr720-18vm3_vmdk0_1 -      4128464  -      -     -       -
dm vmr720-18vm3_vmdk0_2  vmr720-18vm3_vmdk0_2 -      4128464  -      -     -       -
dm vmr720-18vm3_vmdk0_3  vmr720-18vm3_vmdk0_3 -      4128464  -      -     -       -
dm vmr720-18vm3_vmdk0_4  vmr720-18vm3_vmdk0_4 -      4128464  -      -     -       -
dm vmr720-18vm3_vmdk0_5  vmr720-18vm3_vmdk0_5 -      4128464  -      -     -       -
dm vmr720-18vm3_vmdk0_6  vmr720-18vm3_vmdk0_6 -      4128464  -      -     -       -
dm vmr720-18vm3_vmdk0_7  vmr720-18vm3_vmdk0_7 -      4128464  -      -     -       -
```

例: 一般的な用途の使用例

一般的な用途の使用例 (トランザクション作業負荷など) 向けに 1 GB の Erasure Code ボリューム vol1 (ディスク 3 台の障害を許容し、5 台のディスクにデータをストライプ化) を作成する場合は、次のコマンドを実行します。

```
# vxassist -g dg1 make vol1 1g layout=ecoded nparity=3 ncols=5
```

ボリュームの構成を表示するには、次の操作をします。

```
# vxprint
Disk group: dg1

TY  NAME                 ASSOC              KSTATE   LENGTH  PLOFFS STATE TUTIL0 PUTIL0
dg  dg1                    dg1                -        -       -      -      -      -

dm  vmr720-18vm3_vmdk0_0 vmr720-18vm3_vmdk0_0 -     4128464    -      -      -      -
dm  vmr720-18vm3_vmdk0_1 vmr720-18vm3_vmdk0_1 -     4128464    -      -      -      -
dm  vmr720-18vm3_vmdk0_2 vmr720-18vm3_vmdk0_2 -     4128464    -      -      -      -
dm  vmr720-18vm3_vmdk0_3 vmr720-18vm3_vmdk0_3 -     4128464    -      -      -      -
dm  vmr720-18vm3_vmdk0_4 vmr720-18vm3_vmdk0_4 -     4128464    -      -      -      -
dm  vmr720-18vm3_vmdk0_5 vmr720-18vm3_vmdk0_5 -     4128464    -      -      -      -
dm  vmr720-18vm3_vmdk0_6 vmr720-18vm3_vmdk0_6 -     4128464    -      -      -      -
dm  vmr720-18vm3_vmdk0_7 vmr720-18vm3_vmdk0_7 -     4128464    -      -      -      -

v  vol1                    fsgen            ENABLED 2097280    -    SYNC     -       -
pl vol1-01                  vol1             ENABLED 2097280    -    ACTIVE   -       -
sd vmr720-18vm3_vmdk0_0-02 vol1-01          ENABLED 2097152    0    ECLOG    -       -
sd vmr720-18vm3_vmdk0_0-01 vol1-01          ENABLED 419456     0     -       -       -
sd vmr720-18vm3_vmdk0_1-02 vol1-01          ENABLED 2097152    0    ECLOG    -       -
sd vmr720-18vm3_vmdk0_1-01 vol1-01          ENABLED 419456     0     -       -       -
sd vmr720-18vm3_vmdk0_2-02 vol1-01          ENABLED 2097152    0    ECLOG    -       -
sd vmr720-18vm3_vmdk0_2-01 vol1-01          ENABLED 419456     0     -       -       -
sd vmr720-18vm3_vmdk0_3-02 vol1-01          ENABLED 2097152    0    ECLOG    -       -
sd vmr720-18vm3_vmdk0_3-01 vol1-01          ENABLED 419456     0     -       -       -
sd vmr720-18vm3_vmdk0_4-02 vol1-01          ENABLED 2097152    0    ECLOG    -       -
sd vmr720-18vm3_vmdk0_4-01 vol1-01          ENABLED 419456     0     -       -       -
sd vmr720-18vm3_vmdk0_5-02 vol1-01          ENABLED 2097152    0    ECLOG    -       -
sd vmr720-18vm3_vmdk0_5-01 vol1-01          ENABLED 419456     0     -       -       -
sd vmr720-18vm3_vmdk0_6-02 vol1-01          ENABLED 2097152    0    ECLOG    -       -
sd vmr720-18vm3_vmdk0_6-01 vol1-01          ENABLED 419456     0     -       -       -
sd vmr720-18vm3_vmdk0_7-02 vol1-01          ENABLED 2097152    0    ECLOG    -       -
sd vmr720-18vm3_vmdk0_7-01 vol1-01          ENABLED 419456     0     -       -       -

# vxprint -g dg1 -F%stripe_aligned vol1
```

```
off
```

例: オブジェクトストアの使用例

オブジェクトストアの使用例向けに 1 GB の Erasure Code ボリューム vol1 (ディスク 3 台の障害を許容し、5 台のディスクにデータをストライプ化) を作成する場合は、次のコマンドを実行します。

```
# vxassist -g dg1 make vol1 1g layout=ecoded nparity=3 ncols=5
stripe_aligned=yes
```

ボリュームの設定を表示するには

```
# vxprint
Disk group: dg1

TY    NAME              ASSOC               KSTATE   LENGTH    PLOFFS STATE TUTIL0 PUTIL0
dg    dg1                 dg1                  -        -         -      -     -      -

dm vmr720-18vm3_vmdk0_0 vmr720-18vm3_vmdk0_0  -      4128464    -      -       -
 -
dm vmr720-18vm3_vmdk0_1 vmr720-18vm3_vmdk0_1  -      4128464    -      -       -
 -
dm vmr720-18vm3_vmdk0_2 vmr720-18vm3_vmdk0_2  -      4128464    -      -       -
 -
dm vmr720-18vm3_vmdk0_3 vmr720-18vm3_vmdk0_3  -      4128464    -      -       -
 -
dm vmr720-18vm3_vmdk0_4 vmr720-18vm3_vmdk0_4  -      4128464    -      -       -
 -
dm vmr720-18vm3_vmdk0_5 vmr720-18vm3_vmdk0_5  -      4128464    -      -       -
 -
dm vmr720-18vm3_vmdk0_6 vmr720-18vm3_vmdk0_6  -      4128464    -      -       -
 -
dm vmr720-18vm3_vmdk0_7 vmr720-18vm3_vmdk0_7  -      4128464    -      -       -
 -

v  vol1                    fsgen          ENABLED    2097280    -    ACTIVE    -
 -
pl vol1-01                 vol1           ENABLED    2097280    -    ACTIVE    -
 -
sd vmr720-18vm3_vmdk0_0-01 vol1-01        ENABLED    419456     0      -       -
 -
sd vmr720-18vm3_vmdk0_1-01 vol1-01        ENABLED    419456     0      -       -
 -
```

```
sd vmr720-18vm3_vmdk0_2-01 vol1-01        ENABLED    419456     0       -       -
 -
sd vmr720-18vm3_vmdk0_3-01 vol1-01        ENABLED    419456     0       -       -
 -
sd vmr720-18vm3_vmdk0_4-01 vol1-01        ENABLED    419456     0       -       -
 -
sd vmr720-18vm3_vmdk0_5-01 vol1-01        ENABLED    419456     0       -       -
 -
sd vmr720-18vm3_vmdk0_6-01 vol1-01        ENABLED    419456     0       -       -
 -
sd vmr720-18vm3_vmdk0_7-01 vol1-01        ENABLED    419456     0       -       -
 -
```

```
# vxprint -g dg1 -F%stripe_aligned vol1
on
```

出力のプレックスには、プレックスに関連付けられている 8 台のサブディスク (1 列に 1 台ずつ表示) が含まれ、それぞれ別のディスクに作成されています。ストライプボリュームのレイアウト (RAID-0) のように、1 列に表示するために複数のサブディスクを連結する場合もあります。PARITY という状態マークが付いたサブディスクには、エンコードされたデータが含まれます。ECLOG という状態マークが付いたサブディスクは、ログのサブディスクです。Erasure Code ボリュームのレイアウト属性は ECODED です。

ボリュームのレイアウトは、次のコマンドを実行して確認できます。

```
#vxprint -g dg1 -F%layout vol1-01ECODED
```

FSS 環境での Erasure Code ボリュームの作成

この手順では、クラスタ内でストレージを提供するノードが *n* 個 (N1、N2、... Nn) あり、各ノードにそれぞれ d1、d2、... dn というディスクがあることを前提にしています。

FSS 環境で Erasure Code ボリュームを作成するには

1 EC ボリューム用のストレージを提供する各ノードのディスクを初期化し (まだ初期化していない場合)、クラスタ全体で利用可能にするためにディスクをエクスポートします。

メモ: SAL (Storage Access Layer) 機能を使用してディスクを自動エクスポートしている場合、ディスクをエクスポートする必要はありません。

```
# vxdisk export <disk_name>
```

2 すべてのクラスタノードの必要なディスクのセットを使用して、FSS ディスクグループ (dg1) を作成します。

```
# vxdg -s -o fss init dg1 da1 da2 ... dan
```

3 FSS ディスクグループ内の *k* 個のノードのストレージをストライプ化し、*m* 個までの障害を許容する Erasure Code ボリューム (vol1) を作成します。

```
# vxassist -g dg1 make vol1 <vol_size> layout=ecoded ncol=<k>
nparity=<m>
```

ボリュームにストレージを提供する必要があるホストを指定する場合は、次のように指定できます。

```
# vxassist -g dg1 make vol1 <vol_size> layout=ecoded ncol=k
nparity=mhost:N1 host:N2 ....host:Nn
```

Erasure Code ボリュームの作成時にストライプグループとストライプ制限グループを指定します。p.1046 の「Erasure Code ボリューム作成時のストライプグループとストライプ制限グループの使用」を参照してください。

FSS クラスタの 4 つ目のノードに Erasure Code ボリュームを作成した場合の出力例を次に示します。

```
# vxprint

Disk group: dg1

TY NAME                    ASSOC              KSTATE   LENGTH    PLOFFS   STATE   TUTIL0
PUTIL0
dg dg1                      dg1                -        -         -        -       -
 -

dm vmr720-18vm3_vmdk0_0 vmr720-18vm3_vmdk0_0 -       4128464   -        -       -
 -
dm vmr720-18vm4_vmdk0_0 vmr720-18vm4_vmdk0_0 -       4128464   -       REMOTE   -
 -
dm vmr720-18vm5_vmdk0_0 vmr720-18vm5_vmdk0_0 -       4128464   -       REMOTE   -
 -
dm vmr720-18vm6_vmdk0_0 vmr720-18vm6_vmdk0_0 -       4128464   -       REMOTE   -
 -
```

例: 一般的な用途の使用例

一般的な用途の使用例 (トランザクション作業負荷など) 向けに 2 GB の Erasure Code ボリューム vol1 (ノード/ディスクの障害を許容し、3 台のノード/ディスクにデータをストライプ化) を作成するには、次のコマンドを実行します。

```
# vxassist -g dg1 make vol1 2g layout=ecoded nparity=1 ncols=3
```

4 ノードの FSS クラスタに Erasure Code ボリュームを作成した場合の出力例を次に示します。

```
# vxprint
Disk group: dg1

TY   NAME               ASSOC              KSTATE   LENGTH PLOFFS  STATE  TUTIL0 PUTIL0
dg   dg1                  dg1                 -        -     -       -       -      -

dm vmr720-18vm3_vmdk0_0 vmr720-18vm3_vmdk0_0 -     4128464  -       -       -       -
dm vmr720-18vm4_vmdk0_0 vmr720-18vm4_vmdk0_0 -     4128464  -     REMOTE    -       -
dm vmr720-18vm5_vmdk0_0 vmr720-18vm5_vmdk0_0 -     4128464  -     REMOTE    -       -
dm vmr720-18vm6_vmdk0_0 vmr720-18vm6_vmdk0_0 -     4128464  -     REMOTE    -       -

v  vol1                    fsgen          ENABLED   4194432  -     SYNC      -       -
pl vol1-01                  vol1          ENABLED   4194432  -     ACTIVE    -       -
sd vmr720-18vm3_vmdk0_0-02 vol1-01        ENABLED   2097152 0      ECLOG     -       -
sd vmr720-18vm3_vmdk0_0-01 vol1-01        ENABLED   1398144 0       -        -       -
sd vmr720-18vm4_vmdk0_0-02 vol1-01        ENABLED   2097152 0      ECLOG     -       -
sd vmr720-18vm4_vmdk0_0-01 vol1-01        ENABLED   1398144 0       -        -       -
sd vmr720-18vm5_vmdk0_0-02 vol1-01        ENABLED   2097152 0      ECLOG     -       -
sd vmr720-18vm5_vmdk0_0-01 vol1-01        ENABLED   1398144 0       -        -       -
sd vmr720-18vm6_vmdk0_0-02 vol1-01        ENABLED   2097152 0      ECLOG    -       -
sd vmr720-18vm6_vmdk0_0-01 vol1-01        ENABLED   1398144 0       -        -       -
dc vol1_dco                 vol1             -        -     -       -       -       -
v  vol1_dcl                 gen            ENABLED   67840   -      ACTIVE    -       -
pl vol1_dcl-01             vol1_dcl        ENABLED   67840   -      ACTIVE    -       -
sd vmr720-18vm3_vmdk0_0-03vol1_dcl-01     ENABLED   67840   0         -      -       -
pl vol1_dcl-02             vol1_dcl        ENABLED   67840   -      ACTIVE    -       -
sd vmr720-18vm4_vmdk0_0-03 vol1_dcl-02    ENABLED   67840   0       -        -       -

# vxprint -g dg1 -F% stripe_aligned vol1
off
```

例: オブジェクトストアの使用例

オブジェクトストアの使用例向けに 2 GB の Erasure Code ボリューム vol1 (ノード/ディスク障害を許容し、3 台のノード/ディスクにデータをストライプ化) を作成するには、次のコマンドを実行します。

```
# vxassist -g dg1 make vol1 2g layout=ecoded nparity=1 ncols=3
stripe_aligned=yes
```

4 ノードの FSS クラスタに Erasure Code ボリュームを作成した場合の出力例を次に示します。

```
# vxprint
Disk group: dg1

TY NAME                 ASSOC              KSTATE LENGTH   PLOFFS   STATE    TUTIL0
PUTIL0
dg dg1                  dg1                  -      -        -        -        -
 -

dm vmr720-18vm3_vmdk0_0 vmr720-18vm3_vmdk0_0 -    4128464    -         -       -
 -
dm vmr720-18vm4_vmdk0_0 vmr720-18vm4_vmdk0_0 -    4128464    -      REMOTE     -
 -
dm vmr720-18vm5_vmdk0_0 vmr720-18vm5_vmdk0_0 -    4128464    -      REMOTE     -
 -
dm vmr720-18vm6_vmdk0_0 vmr720-18vm6_vmdk0_0 -    4128464    -      REMOTE     -
 -

v  vol1                 fsgen            ENABLED 4194432    -      ACTIVE     -
 -
pl vol1-01              vol1             ENABLED 4194432    -      ACTIVE     -
 -
sd vmr720-18vm3_vmdk0_0-01 vol1-01       ENABLED 1398144    0        -        -
 -
sd vmr720-18vm4_vmdk0_0-01 vol1-01       ENABLED 1398144    0        -        -
 -
sd vmr720-18vm5_vmdk0_0-01 vol1-01       ENABLED 1398144    0        -        -
 -
sd vmr720-18vm6_vmdk0_0-01 vol1-01       ENABLED 1398144    0        -        -
 -
dc vol1_dco                vol1            -        -       -        -        -
 -
v  vol1_dcl                gen             ENABLED  67840     -      ACTIVE     -
 -
pl vol1_dcl-01            vol1_dcl         ENABLED  67840     -      ACTIVE     -
 -
sd vmr720-18vm3_vmdk0_0-02 vol1_dcl-01     ENABLED  67840     0        -        -
 -
pl vol1_dcl-02             vol1_dcl        ENABLED  67840     -      ACTIVE     -
 -
sd vmr720-18vm4_vmdk0_0-02 vol1_dcl-02     ENABLED  67840     0        -        -
```

-

```
#vxprint -g dg1 -F%stripe_aligned vol1
on
```

ボリュームのプレックスで次のコマンドを実行して、ボリュームのレイアウトを確認できます。

```
# vxprint -g dg1 -F% layout vol1-01
ECODED
```

Erasure Code ボリュームの I/O 操作

Erasure Code ボリュームのレイアウトでは、データをパーティション分割してデータとパリティの列全体に保存します。データ列にはデータをそのまま保存し、パリティ列には、エンコード形式のデータに関する冗長な情報を保存します。

I/O 操作は、任意の基礎ディスクに障害が発生したかどうかに応じて異なる方法で実行されます。

- ストレージ障害なし
  読み取り要求はパリティ列を介さずにデータ列から実行されます。書き込み操作では、すべてのパリティ列がデータとともに更新されます。
- 1 つ以上のディスクに障害が発生した (耐障害性で対応するほどではない)
  読み取り操作では、障害が発生した 1 つ以上のデータ列で、利用可能なデータ列とパリティ列を使用してデータを再構築します。書き込み操作では、利用可能な (障害が発生していない) ディスク上のデータを更新します。利用可能なディスクのパリティを更新するには、新しいデータと組み合わせて障害が発生した列の再構築データを使います。

パリティ列の障害は、読み取り操作でも書き込み操作でも同様の方法で処理されます。I/O 操作で障害が検出されるとすぐに、今後の I/O に対応するディスクを無効化し、DCO をボリュームに設定している場合はボリュームでの書き込みを追跡します。

Erasure Code ボリュームのリカバリ

Erasure Code ボリュームは、次の複数の理由によりリカバリが必要になることがあります。

- 下位ストレージが一時的にデータを古い状態のままにできなかった
- 下位ストレージを退避または移行 (再配置またはホットリロケーション) した

ボリュームのリカバリにより、再生ストレージ上のデータの再構築が必要になります。効率的なデータの再構築 (すなわち、すでに一貫性があるデータの再構築の回避) に Flashsnap 機能を使用できます。Flashsnap は、ディスクが切断されるとボリューム上の

変更時点の追跡を提供します。Flashsnap 機能を使用するには、できれば Erasure Code ボリュームの作成中またはディスクにエラーが発生する前に、最新のサポート対象バージョンの DCO を追加してこのようなリカバリに使うボリュームを準備します。

最適なリカバリを行うためのボリュームを準備するには、vxsnap コマンドを使用します。

```
# vxsnap -g disk_group prepare vol_name ndcomir=n
```

ボリュームに必要な耐障害性を実現するために必要な数のミラーを DCO に作成 (ndcomir 属性を使用して指定) することを推奨します。

Erasure Code ボリュームに DCO がある場合は、いずれかのサブディスクで読み取り/読み込みエラーが発生するとボリューム上の書き込み操作の追跡が有効になり、ストレージのサブシステムを再生およびリカバリするときにサブディスクを最適な状態でリカバリするために同じボリュームを使用します。

リカバリの種類

Erasure Code ボリュームは自動または手動でリカバリできます。

自動リカバリ	ストレージ障害が一時的な場合 (たとえば、FSS クラスタ内のストレージを提供するノードでエラーが発生して再生する場合)、再生したストレージ上のデータは古いので再構築またはリカバリする必要があります。vxattachd デーモンはストレージの再生を自動的に検出します。Erasure Code ボリュームが検出されたストレージに存在する場合、リカバリで自動的に古いディスクの同期を開始します。
手動リカバリ	vxattachd デーモンを実行していない場合は、デーモンを再起動しても Erasure Code ボリュームを自動的にリカバリすることはできません。このような場合は、vxrecover コマンドを使用して Erasure Code ボリュームを手動でリカバリする必要があります。 `# vxrecover -g disk_group`

リカバリタスクの管理

Erasure Code ボリュームをリカバリすると、vxtask コマンドを使用して制御されるタスクが作成されます。vxtask list コマンドを使用して関連するリカバリタスクを検索し、必要に応じてこのタスクを監視または制御します。

詳しくは、vxtask(1M) マニュアルページを参照してください。

Erasure Code ボリュームを含む障害のあるストレージの再配置

Erasure Code ボリュームに含まれるディスクで障害が発生し、それを別のディスクに置き換える場合、Erasure Code ボリュームではホットリロケーションがサポートされるため、この処理が自動的に行われます。新しいディスクは、障害が発生したディスクと同じスト

レージ制約に従う必要があります。この障害が発生したディスクが、ノードのローカルで使用できない場合は、このディスクを別の新しいノードのディスクに再配置または移動することができます。ただし、新しいノードに再配置または移動されたディスクは、どの EC カラムにも指定できません。

障害が発生したディスクを新しいディスクに交換するには、次のコマンドを実行します。

```
# vxassist -r -g disk_group move vol_name ¥
!fail_disk [new_disk]
```

fail_disk は交換する必要がある、障害が発生したディスクの名前です。

new_disk はオプションの引数です。障害が発生したディスクの代わりに使う必要がある新しいディスクを示します。

新しいディスクを指定せず、利用可能なスペアディスクが存在する場合、VxVM はスペアディスクのいずれかを使用して障害が発生したディスクを交換します。ディスクを正常に交換したら、次のコマンドを実行して新しいディスクをリカバリします。

```
# vxrecover -g disk_group
```

Erasure Code ボリュームの初期化

Erasure Code ボリュームの作成時、デフォルトでは Veritas InfoScale は非同期の初期化をボリューム上で実行し、データとパリティがすべての領域で確実に同期するようにします。この操作はバックグラウンドで実行されるため、ボリュームは作成直後にアプリケーションから使用できるようになります。ボリュームは作成後、すべての領域が同期されるまでの間、SYNC 状態として表示されます。この機能は、プライベートと共有または FSS のディスクグループのどちらでもサポートされます。

Erasure Code ボリュームは、ボリュームの作成時に `init=zero` を設定することで、手動で初期化できます。初期化によってすべての領域は初期化され、ボリュームは初期化プロセスが完了するまでの間、使用できなくなります。

初期化中と初期化後のボリュームの状態を確認するには

1 Erasure Code ボリュームを作成します。

```
# vxassist -g dg1 make vol1 2g layout=ecoded ncol=3 nparity=1
```

2 初期化中と初期化の完了後、ボリュームの状態を確認します。

初期化の完了前のボリュームの状態

```
# vxprint
Disk group: dg1

TY   NAME                ASSOC              KSTATE   LENGTH  PLOFFS STATE TUTIL0 PUTIL0
dg   dg1                 dg1                 -        -       -     -      -      -

dm vmr720-18vm3_vmdk0_0 vmr720-18vm3_vmdk0_0 -      4128464    -     -      -      -
dm vmr720-18vm4_vmdk0_0 vmr720-18vm4_vmdk0_0 -      4128464    -   REMOTE   -      -
dm vmr720-18vm5_vmdk0_0 vmr720-18vm5_vmdk0_0 -      4128464    -   REMOTE   -      -
dm vmr720-18vm6_vmdk0_0 vmr720-18vm6_vmdk0_0 -      4128464    -   REMOTE   -      -


v  vol1                 fsgen              ENABLED  4194432    -   SYNC     -      -
pl vol1-01              vol1               ENABLED  4194432    -   ACTIVE   -      -
sd vmr720-18vm3_vmdk0_0-02 vol1-01         ENABLED  2097152    0   ECLOG    -       -
sd vmr720-18vm3_vmdk0_0-01 vol1-01         ENABLED  1398144    0    -       -       -
sd vmr720-18vm4_vmdk0_0-02 vol1-01         ENABLED  2097152    0   ECLOG    -       -
sd vmr720-18vm4_vmdk0_0-01 vol1-01         ENABLED  1398144    0    -       -       -
sd vmr720-18vm5_vmdk0_0-02 vol1-01         ENABLED  2097152    0   ECLOG    -       -
sd vmr720-18vm5_vmdk0_0-01 vol1-01         ENABLED  1398144    0    -       -       -
sd vmr720-18vm6_vmdk0_0-02 vol1-01         ENABLED  2097152    0   ECLOG    -       -
sd vmr720-18vm6_vmdk0_0-01 vol1-01         ENABLED  1398144    0    -       -       -
dc vol1_dco                vol1             -        -       -    -       -       -
v  vol1_dcl                gen              ENABLED   67840     -   ACTIVE   -       -
pl vol1_dcl-01             vol1_dcl         ENABLED   67840     -   ACTIVE   -       -
sd vmr720-18vm3_vmdk0_0-03 vol1_dcl-01      ENABLED   67840     0     -      -       -
pl vol1_dcl-02             vol1_dcl         ENABLED   67840     -   ACTIVE   -       -
sd vmr720-18vm4_vmdk0_0-03 vol1_dcl-02      ENABLED   67840 0   -     -      -       -

# vxtask list
TASKID       PTID      TYPE/STATE          PCT                PROGRESS
22020283     -     ECINIT/R          10.35%          0/4194432/433920 ECINIT vol1
                                                       dg1 auto-throttled
```

初期化の完了後のボリュームの状態

```
# vxprint

Diskgroup: dg1

TY   NAME                ASSOC              KSTATE   LENGTH  PLOFFS STATE TUTIL0 PUTIL0
```

```
dg   dg1                 dg1                 -        -        -     -       -       -

dm vmr720-18vm3_vmdk0_0 vmr720-18vm3_vmdk0_0 -      4128464    -     -       -       -
dm vmr720-18vm4_vmdk0_0 vmr720-18vm4_vmdk0_0 -      4128464    -   REMOTE    -       -
dm vmr720-18vm5_vmdk0_0 vmr720-18vm5_vmdk0_0 -      4128464    -   REMOTE    -       -
dm vmr720-18vm6_vmdk0_0 vmr720-18vm6_vmdk0_0 -      4128464    -   REMOTE    -       -

v  vol1                  fsgen             ENABLED  4194432    -   ACTIVE     -       -
pl vol1-01               vol1              ENABLED  4194432    -   ACTIVE     -       -
sd vmr720-18vm3_vmdk0_0-02 vol1-01         ENABLED  2097152    0   ECLOG      -       -
sd vmr720-18vm3_vmdk0_0-01 vol1-01         ENABLED  1398144    0    -         -       -
sd vmr720-18vm4_vmdk0_0-02 vol1-01         ENABLED  2097152    0   ECLOG      -       -
sd vmr720-18vm4_vmdk0_0-01 vol1-01         ENABLED  1398144    0    -         -       -
sd vmr720-18vm5_vmdk0_0-02 vol1-01         ENABLED  2097152    0   ECLOG      -       -
sd vmr720-18vm5_vmdk0_0-01 vol1-01         ENABLED  1398144    0    -         -       -
sd vmr720-18vm6_vmdk0_0-02 vol1-01         ENABLED  2097152    0   ECLOG      -       -
sd vmr720-18vm6_vmdk0_0-01 vol1-01         ENABLED  1398144    0    -         -       -
dc vol1_dco                 vol1              -        -        -     -       -       -
v  vol1_dcl                 gen               ENABLED   67840     -   ACTIVE    -       -
pl vol1_dcl-01             vol1_dcl           ENABLED   67840     -   ACTIVE    -       -
sd vmr720-18vm3_vmdk0_0-03 vol1_dcl-01        ENABLED   67840     0     -       -       -
pl vol1_dcl-02             vol1_dcl           ENABLED   67840     -   ACTIVE    -       -
sd vmr720-18vm4_vmdk0_0-03 vol1_dcl-02        ENABLED   67840 0   -     -       -
```

Erasure Code ボリュームのサイズ変更

Erasure Code ボリュームは、標準の VxVM ボリュームと同じ方法でサイズを変更できます。

FSS または DAS ストレージ上に設定された Erasure Code ボリュームのサイズを変更するには

1. データとパリティ用にストレージを提供するノードから、新しいディスクをディスクグループに追加します。

   ```
   # vxdg -g dgname adddisk ecdisk1 ecdisk2
   ```

2. 個々のノードのストレージ容量に空きがない場合、データとパリティ用にストレージを提供する既存のノード数と同数のクラスタ内の新しいノードを追加します。新しいノードからのディスクをディスクグループに追加します。

3. ボリュームのサイズを変更します。

   ```
   # vxresize -g dgnamevolname+5G
   ```

カスタマイズされた障害ドメイン

Erasure Code ボリュームのカラムは、デフォルトで 1 つのノードに制限されている障害ドメインです。カスタマイズ済み障害ドメインを使用すると、ユーザーはカラムの制約を指定し、ニーズや利用可能な設定に基づいて障害ドメインを定義できます。この機能を利用することで、ユーザーはボリューム割り当てポリシーと障害ドメインの管理をより詳細に制御できます。次に、カスタマイズ済み障害ドメインを作成して処理できる、障害ドメインの例をいくつか示します。

- 障害ドメインとしての RAC
- 障害ドメインとしての個々のノード
- 障害ドメインとしてのノード内のディスクトレー

カスタム障害ドメインの定義

タグを設定することによって、割り当てで使用されるデバイスをカスタマイズします。タグは、キーと値のペアで、各ボリュームのディスクに保存されます。障害ドメインには、ラック、ノード、トレー、またはディスクなどのタグを適用できるさまざまなレベルがあります。

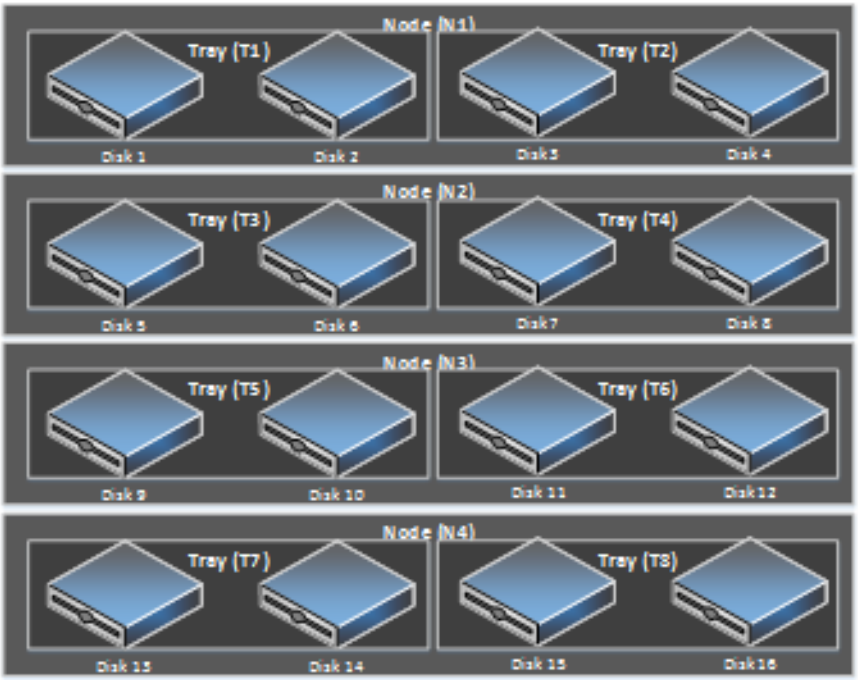
ディスクにタグを割り当て、そのディスクのトレー、ラック、ノードなどを指定するには、次のコマンドを入力する必要があります。このタグに基づいて値が適用されます。

```
# vxdisk settag <tag_name>=<tag_val> <device_name>
```

図 41-8 は、2 つのレベルの障害ドメインの例を示しています。ノードがより高いレベルになります。

- ノードがより高いレベル:
  - タグ名: Node
  - タグ値: N1、N2、N3、N4
- トレーが次のレベル
  - タグ名: Tray
  - タグ値: T1、T2、T3、T4

図 **41-8** 2 つのレベルのカスタム障害ドメイン - ノードとトレー



例として、FSS セットアップ内の 1 つのノードにつき 2 つのディスクが含まれる 4 ノードのクラスタについて考えます。

FSS 環境内のノードとトレーにタグを割り当てるには

1 ディスクグループを作成します。

```
# vxprint
Disk group: ecdg

TY NAME                  ASSOC             KSTATE LENGTH   PLOFFS  STATE   TUTIL0  PUTIL0
dg ecdg                  ecdg                 -      -        -      -       -       -

dm vmr720-18vm3_vmdk0_0 vmr720-18vm3_vmdk0_0 -    4128464   -       -       -       -
dm vmr720-18vm3_vmdk0_1 vmr720-18vm3_vmdk0_1 -    4128464   -       -       -       -
dm vmr720-18vm4_vmdk0_0 vmr720-18vm4_vmdk0_0 -    4128464   -     REMOTE    -       -
dm vmr720-18vm4_vmdk0_1 vmr720-18vm4_vmdk0_1 -    4128464   -     REMOTE    -       -
dm vmr720-18vm5_vmdk0_0 vmr720-18vm5_vmdk0_0 -    4128464   -     REMOTE    -       -
dm vmr720-18vm5_vmdk0_1 vmr720-18vm5_vmdk0_1 -    4128464   -     REMOTE    -       -
dm vmr720-18vm6_vmdk0_0 vmr720-18vm6_vmdk0_0 -    4128464   -     REMOTE    -       -
dm vmr720-18vm6_vmdk0_1 vmr720-18vm6_vmdk0_1 -    4128464   -     REMOTE    -       -
```

2 すべてのディスクにタグを割り当てて、カスタムドメインを指定します。

```
# vxdisk listtag | sort -k2

DEVICE                  NAME      VALUE
vmr720-18vm3_vmdk0_0    node       n1
vmr720-18vm3_vmdk0_1    node       n1
vmr720-18vm4_vmdk0_0    node       n2
vmr720-18vm4_vmdk0_1    node       n2
vmr720-18vm5_vmdk0_0    node       n3
vmr720-18vm5_vmdk0_1    node       n3
vmr720-18vm6_vmdk0_0    node       n4
vmr720-18vm6_vmdk0_1    node       n4
vmr720-18vm3_vmdk0_0    tary       t1
vmr720-18vm3_vmdk0_1    tary       t2
vmr720-18vm4_vmdk0_0    tray       t3
vmr720-18vm4_vmdk0_1    tray       t4
vmr720-18vm5_vmdk0_0    tray       t5
vmr720-18vm5_vmdk0_1    tray       t6
vmr720-18vm6_vmdk0_0    tray       t7
vmr720-18vm6_vmdk0_1    tray       t8
```

Erasure Code ボリューム作成時のストライプグループとストライプ制限グループの使用

Erasure Code ボリュームの作成中に、ユーザーはこれらの制約を両方とも指定できます。何も指定しない場合、ストライプグループの値がホストになります。

- ストライプグループ (SG): ストライプグループは、カラムを割り当てるための分離基準を次のように定義します。
  - 1 つ以上の分離基準メンバーから 1 カラムが割り当てられます。
  - 1 つのストライプグループメンバーは、最大で 1 つのカラムを構成します。
    例: 1 つのストライプグループに 2 つのディスクがある場合、これら 2 つのディスクが、あるボリュームの単一のカラムを構成します。ただし、1 つ目のディスクをあるカラムで使用し、2 つ目のディスクを同じボリュームの別のカラムで使用することはできません。
- ストライプ制限グループ (SCG): ストライプ制限グループは、カラムを割り当てるための制限グループを次のように定義します。
  - 1 つのカラムは、そのストライプ制限クラスで定義された 1 つの障害ドメイン内に制限されます。
  - 1 つのストライプ制限グループメンバーは、1 つ以上の完全なカラムを含むか、何も含まないかのいずれかになります。ただし、SCG メンバーに部分的なカラムが含まれることはありません。
    両方の制約を比べると、SCG の方が SG より高い階層レベルにあります。したがって、SCG がノードの場合、SG はトレーとして指定できます。しかし、SG がノードとして指定されている場合、SCG はトレーとして指定できません。
    例: あるカラム全体が特定の SCG グループメンバー内に含まれる場合、別の SCG グループメンバーが同じカラムを使用することはできません。

メモ: Erasure Code ボリュームを作成すると、そのボリュームに対してデフォルトでログが維持されます。この場合、ストライプ制限グループの制約が指定されていないと、この値はデフォルトでホストとして設定されます。

ボリュームの作成時に SG と SCG を使用するには

1 レイアウト、データカラム数、およびパリティカラム数を指定します。

p.1030 の「DAS または SAN のストレージを使用して 1 つのノード上に Erasure Code ボリュームを作成する」を参照してください。

p.1033 の「FSS 環境での Erasure Code ボリュームの作成」を参照してください。

2 ストライプグループとストライプ制限グループを指定します。

SG を指定するには、次のコマンドを使用します。

```
# stripe=dtag:<tag name>
```

SCG を指定するには、次のコマンドを使用します。

```
# Stripeconfine=dtag:<tag name>
```

3 次のコマンドを使用して、ストライプグループとストライプ制限グループを含むボリュームを作成します。

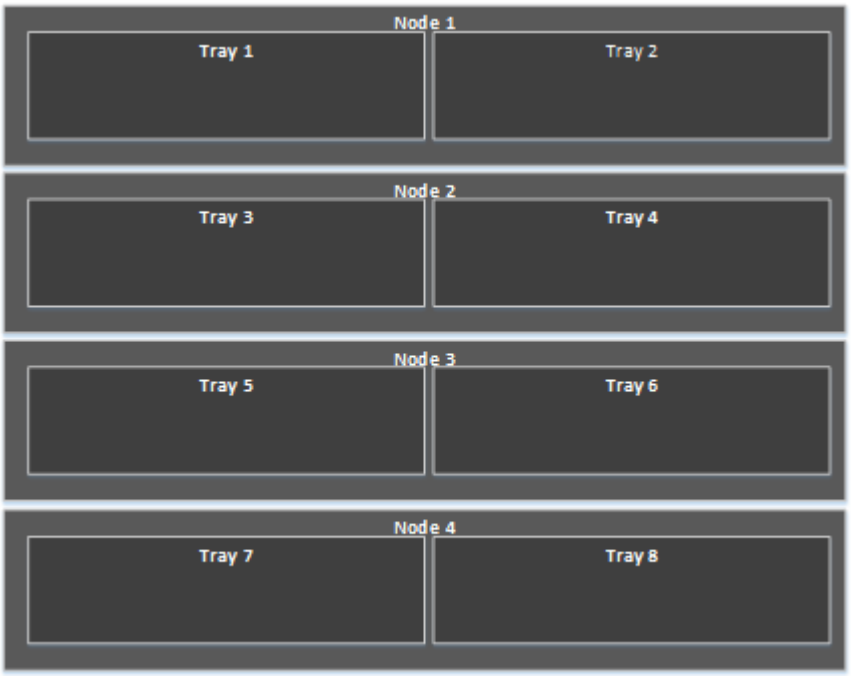
```
# vxassist -g <dg_name> make <vol_name> <vol_size> layout=ecoded
stripe=dtag:<tag name> stripeconfine=dtag:<tag name>
```

ここで、カスタム障害ドメインを使用してボリュームを作成する際のさまざまな構成例を紹介します。

例

次の Erasure Code の構成例では、3 つのデータカラムと 1 つのパリティカラムを使用しています。ストライプグループがトレーで、ストライプ制限グループがノードです。1 つのトレーに完全なカラムまたは部分的なカラムのいずれかがあり (2 カラムではない)、カラムか 2 つのノードにまたがらない構成です。

図 41-9　　2 つのレベルのカスタム障害ドメイン - トレーとノード



この場合、ストライプグループ (SG) とストライプ制限グループの両方を指定します。

```
# vxassist -g ecdg make ecvol1 1g ncol=3 nparity=1 layout=ecoded
stripe=dtag:tray stripeconfine=dtag:node
```

```
# vxprint -g ecdg ecvol1

TY NAME                 ASSOC              KSTATE   LENGTH   PLOFFS  STATE   TUTIL0 PUTIL0
dg ecdg                  ecdg                  -        -        -       -       -      -

dm vmr720-18vm3_vmdk0_0 vmr720-18vm3_vmdk0_0  -      4128464  -       -       -      -
dm vmr720-18vm3_vmdk0_1 vmr720-18vm3_vmdk0_1  -      4128464  -       -       -      -
dm vmr720-18vm4_vmdk0_0 vmr720-18vm4_vmdk0_0  -      4128464  -     REMOTE    -      -
dm vmr720-18vm4_vmdk0_1 vmr720-18vm4_vmdk0_1  -      4128464  -     REMOTE    -      -
dm vmr720-18vm5_vmdk0_0 vmr720-18vm5_vmdk0_0  -      4128464  -     REMOTE    -      -
dm vmr720-18vm5_vmdk0_1 vmr720-18vm5_vmdk0_1  -      4128464  -     REMOTE    -      -
dm vmr720-18vm6_vmdk0_0 vmr720-18vm6_vmdk0_0  -      4128464  -     REMOTE    -      -
dm vmr720-18vm6_vmdk0_1 vmr720-18vm6_vmdk0_1  -      4128464  -     REMOTE    -      -

v  ecvol1                fsgen            ENABLED   2097408  -      SYNC     -
```

```
-
pl ecvol1-01              ecvol1           ENABLED   2097408   -     ACTIVE   -
-
sd vmr720-18vm3_vmdk0_0-02 ecvol1-01       ENABLED   2097152   0     ECLOG    -
-
sd vmr720-18vm3_vmdk0_0-01 ecvol1-01       ENABLED   699136    0      -       -
-
sd vmr720-18vm4_vmdk0_0-02 ecvol1-01       ENABLED   2097152   0     ECLOG    -
-
sd vmr720-18vm4_vmdk0_0-01 ecvol1-01       ENABLED   699136    0      -       -
-
sd vmr720-18vm5_vmdk0_0-02 ecvol1-01       ENABLED   2097152   0     ECLOG    -
-
sd vmr720-18vm5_vmdk0_0-01 ecvol1-01       ENABLED   699136    0      -       -
-
sd vmr720-18vm6_vmdk0_0-02 ecvol1-01       ENABLED   2097152   0     ECLOG    -
-
sd vmr720-18vm6_vmdk0_0-01 ecvol1-01       ENABLED   699136    0      -       -
-
dc ecvol1_dco             ecvol1              -        -       -      -       -
-
v  ecvol1_dcl             gen              ENABLED   67840     -     ACTIVE   -
-
pl ecvol1_dcl-01 ecvol1_dcl                ENABLED   67840     -     ACTIVE   -
-
sd vmr720-18vm3_vmdk0_0-03 ecvol1_dcl-01   ENABLED   67840     0      -       -
-
pl ecvol1_dcl-02         ecvol1_dcl        ENABLED   67840     -     ACTIVE   -
-
sd vmr720-18vm4_vmdk0_0-03 ecvol1_dcl-02   ENABLED   67840     0      -       -
-
```

例

次の Erasure Code ボリュームの構成例では、3 つのカラムと 1 つのパリティを使用し、ノードはストライプグループです。異なるノードで分離されたカラムを含むボリュームを作成します。1 つのノードに完全なカラムまたは部分的なカラムのいずれかがあり、2 カラムは含まない構成です。ただし、カラムが 1 つ以上のノードにまたがることは可能です。

この場合、指定できるのはストライプグループ (SG) のみです。

```
# vxassist -g ecdg make ecvol2 1g ncol=3 nparity=1 layout=ecoded
stripe=dtag:node
```

```
# vxprint -g ecdg ecvol2

TY NAME                    ASSOC    KSTATE   LENGTH   PLOFFS STATE TUTIL0 PUTIL0

v  ecvol2                  fsgen    ENABLED  2097408  -      SYNC    -       -
pl ecvol2-01    ecvol2              ENABLED  2097408  -      ACTIVE  -       -
sd vmr720-18vm5_vmdk0_1-02 ecvol2-01 ENABLED  2097152  0      ECLOG   -       -
sd vmr720-18vm5_vmdk0_1-01 ecvol2-01 ENABLED  699136   0       -      -       -
sd vmr720-18vm6_vmdk0_1-02 ecvol2-01 ENABLED  2097152  0      ECLOG   -       -
sd vmr720-18vm6_vmdk0_1-01 ecvol2-01 ENABLED  699136   0       -      -       -
sd vmr720-18vm3_vmdk0_1-02 ecvol2-01 ENABLED  2097152  0      ECLOG   -       -
sd vmr720-18vm3_vmdk0_1-01 ecvol2-01 ENABLED  699136   0       -      -       -
sd vmr720-18vm4_vmdk0_1-02 ecvol2-01 ENABLED  2097152  0      ECLOG   -       -
sd vmr720-18vm4_vmdk0_1-01 ecvol2-01 ENABLED  699136   0       -      -       -
dc ecvol2_dco              ecvol2       -        -      -       -       -
v  ecvol2_dcl              gen      ENABLED  67840    -      ACTIVE  -       -
pl ecvol2_dcl-01           ecvol2_dclENABLED  67840    -      ACTIVE  -       -
sd vmr720-18vm3_vmdk0_1-03 ecvol2_dcl-01 ENABLED 67840 0       -      -       -
pl ecvol2_dcl-02           ecvol2_dclENABLED  67840    -      ACTIVE  -       -
sd vmr720-18vm4_vmdk0_1-03 ecvol2_dcl-02 ENABLED 67840 0       -      -       -
```

例

次の例は、ストライプグループがノードの場合に、容量がどのように使用されるかを示しています。この場合、Erasure Code ボリューム構成は、4 つのカラムと 2 つのパリティを使用し、ノードはストライプグループです。異なるノードで分離し、ノードに制限したカラムを含むボリュームを作成します。6 (4+2) ノードを必要とする、1 ノードにつき 1 カラムの構成ですが、利用可能なノードは 4 つのみです。このため、ボリュームの作成は失敗します。

```
# vxassist -g ecdg make ecvol3 1g ncol=4 nparity=2 layout=ecoded
stripe=dtag:node

# VxVM vxassist ERROR V-5-1-15315 Cannot allocate space for 2097152 block volume:

Not enough HDD devices that meet specification
```

例

次の例は、ストライプグループがトレーで、利用可能なトレーよりも容量が多い場合を示しています。この場合、Erasure Code ボリュームの構成では、4 つのカラムと 2 つのパリティを使用し、ストライプグループはトレーです。異なるトレーで分離されたカラムを含むボリュームを作成します。6 (4+2) トレーを必要とする、1 トレーにつき 1 カラムの構成ですが、8 (4*2) トレーが利用可能です。ボリュームの作成時に必要なトレーは 6 つですが、利用可能なトレーは 8 つあるため、作成は成功します。

```
                # vxassist -g ecdg make ecvol3 1g ncol=4 nparity=2 layout=ecoded
                stripe=dtag:tray

# vxprint -g ecdg ecvol3

TY NAME                    ASSOC    KSTATE   LENGTH   PLOFFS  STATE  TUTIL0 PUTIL0

v  ecvol3                  fsgen     ENABLED 2097152   -     SYNC     -      -
pl ecvol3-01               ecvol3    ENABLED 2097152   -     ACTIVE   -      -
sd vmr720-18vm3_vmdk0_0-02 ecvol3-01 ENABLED 2097152   0     ECLOG    -      -
sd vmr720-18vm3_vmdk0_0-01 ecvol3-01 ENABLED 524288    0      -       -      -
sd vmr720-18vm4_vmdk0_0-02 ecvol3-01 ENABLED 2097152   0     ECLOG    -      -
sd vmr720-18vm4_vmdk0_0-01 ecvol3-01 ENABLED 524288    0      -       -      -
sd vmr720-18vm5_vmdk0_0-02 ecvol3-01 ENABLED 2097152   0     ECLOG    -      -
sd vmr720-18vm5_vmdk0_0-01 ecvol3-01 ENABLED 524288    0      -       -      -
sd vmr720-18vm6_vmdk0_0-02 ecvol3-01 ENABLED 2097152   0     ECLOG    -      -
sd vmr720-18vm6_vmdk0_0-01 ecvol3-01 ENABLED 524288    0      -       -      -
sd vmr720-18vm3_vmdk0_1-02 ecvol3-01 ENABLED 2097152   0     ECLOG    -       -
sd vmr720-18vm3_vmdk0_1-01 ecvol3-01 ENABLED 524288    0      -       -       -
sd vmr720-18vm4_vmdk0_1-02 ecvol3-01 ENABLED 2097152   0     ECLOG    -       -
sd vmr720-18vm4_vmdk0_1-01 ecvol3-01 ENABLED 524288    0      -       -       -
dc ecvol3_dco              ecvol3       -     -         -      -       -       -
v  ecvol3_dcl              gen       ENABLED 67840     -      ACTIVE  -       -
pl ecvol3_dcl-01           ecvol3_dclENABLED 67840    -       ACTIVE  -       -
sd vmr720-18vm5_vmdk0_0-03 ecvol3_dcl-01 ENABLED 67840 0        -      -       -
pl ecvol3_dcl-02           ecvol3_dclENABLED 67840    -       ACTIVE  -       -
sd vmr720-18vm6_vmdk0_0-03 ecvol3_dcl-02 ENABLED 67840 0        -      -       -
pl ecvol3_dcl-03           ecvol3_dcl ENABLED 67840    -      ACTIVE  -       -
sd vmr720-18vm3_vmdk0_0-03 ecvol3_dcl-03 ENABLED 67840 0        -      -       -
```

カスタマイズされた障害ドメインがある場合のサイズ変更と再配置

サイズ変更と再配置は類似の方法で動作しますが、カスタマイズされた障害ドメインがある場合、別の方法で動作します。ボリュームの作成中に使用される同じ制約かインテントに従います。

ストライプグループとストライプ制限グループの両方が指定されたときに想定される動作

たとえば、トレーをストライプグループ、ノードをストライプ制限グループとして指定するとします。この場合、サイズ変更操作と再配置操作の動作は次のようになります。

- サイズ変更: サイズ変更操作は、ノードであるストライプ制限の障害ドメイン内で領域が利用可能な場合にのみ発生します。ノードのストレージ容量に空きがない場合、拡張操作またはサイズ変更操作はできません。

- 再配置: トレーが失敗し、別の空きトレーが同じノードで使用可能な場合、同じノード上に存在する別の空きトレーで再配置が発生します。同じノードで使用できる空きトレーがない場合、次の基準を満たすクラスタの任意のノード上で再配置が発生します。
  - ノードには 1 つ以上の未使用のトレーが含まれる (ストライプグループ)
  - ノードにはそれらの未使用のトレーに属する 1 つのカラム全体が含まれ、そのようなカラムはノード間にまたがることはない (ストライプ制限グループ)

ストレージの破棄

ここでは、VxVM からディスクとボリュームを削除する方法について説明します。

ボリュームの削除

ボリュームがアクティブでなくなるか、または内容がアーカイブされている場合、そのボリュームはもう必要ありません。その場合は、そのボリュームを削除して、ディスク領域を他の目的のために解放できます。

ボリュームを削除するには、次の手順を実行します。

1. システムで実行中のアプリケーションプログラム (シェルを含む) からのボリュームに対する参照をすべて削除します。
2. ボリュームをファイルシステムとしてマウントしている場合は、次のコマンドを使ってボリュームをマウント解除します。

   ```
   # umount /dev/vx/dsk/diskgroup/volume
   ```

3. このボリュームが `/etc/fstab` ファイルに記述されている場合は、このファイルを編集してそのエントリを削除します。このファイルのフォーマットやファイルの変更方法について詳しくは、オペレーティングシステムのマニュアルを参照してください。

4 次のコマンドを使って、ボリューム上の VxVM によるアクティビティをすべて停止します。

```
# vxvol [-g diskgroup] stop volume
```

5 次のように vxassist コマンドを使って、ボリュームを削除します。

```
# vxassist [-g diskgroup] remove volume volume
```

また、次のように vxedit コマンドを使ってもボリュームを削除できます。

```
# vxedit [-g diskgroup] [-r] [-f] rm volume
```

vxedit の -r オプションは再帰的に削除することを示します。このコマンドでは、ボリュームに関連付けられたすべてのプレックスと、それらのプレックスに関連付けられたすべてのサブディスクが削除されます。vxedit に -f オプションを指定すると、強制的に削除が行われます。ボリュームがまだ有効になっている場合は、このオプションを指定する必要があります。

VxVM 制御下からのディスクの削除

ディスクグループからディスクを削除すると、Veritas Volume Manager の制御下からディスクを完全に削除できます。

警告: vxdiskunsetup コマンドは、VxVM メタデータをディスクから消去して、Veritas Volume Manager の制御下からディスクを削除します。データが失われないようにするには、ディスク上のデータをすべてディスクから退避させておく必要があります。vxdiskunsetup コマンドは、Veritas Volume Manager に精通したシステム管理者のみが十分な注意を払って使う必要があります。

VxVM 制御下からディスクを削除するには

◆ 次のコマンドを入力します。

```
# /usr/lib/vxvm/bin/vxdiskunsetup sdx
```

vxdiskunsetup(1M)のマニュアルページを参照してください。

データの細断について

機密データを含むディスクを廃止するとき、ディスクの残りのデータを破壊する必要がある場合があります。データを単に削除するだけでは、機密データとセキュリティデータが十分保護されない可能性があります。データの削除だけでなく、ディスクに保存されている情報をハッカーがリカバリする可能性を取り除きたいと考えています。規制基準では、機密データとセキュリティデータを無害化、またはデジタルパターンを使用してデータを

上書きするなどして消去することが必要です。VxVM（Veritas Volume Manager）は、1 つ、3 つ、または 7 つのパスのデジタルパターンを持つアドレス指定可能なブロックすべてを上書きする、ディスクの細断操作を提供します。

注意: ボリューム内のデータはすべて、削除すると細断されます。情報が別のストレージメディアにバックアップされて検証されているか、または情報がすでに不要になっているかどうかを確認します。

VxVM は、データがリカバリされる可能性を最小化するために、ディスク上のデータを細断する機能を提供します。ディスクの細断操作を指定すると、VxVM は、既存のディスクラベルを含めディスク全体を細断します。細断操作の後、ディスクがエラー状態になるのを防ぐため、VxVM はディスク上の新しい空のラベルに書き込みます。VxVM 細断操作は、ディスクを上書きする次の方法を提供します。

- 1 つのパスのアルゴリズム
  VxVM は、ランダムに選択したデジタルパターンが付いているディスクを上書きします。このオプションにかかる時間は最小です。デフォルトのタイプは、1 つのパスのアルゴリズムです。
- 3 つのパスのアルゴリズム
  VxVM は合計 3 回ディスクに上書きします。最初のパスでは、VxVM は事前選択したデジタルパターンが付いているデータを上書きします。2 回目は、VxVM はパターンの 2 進補数が付いているデータを上書きします。最終パスでは、VxVM はランダムに選択したデジタルパターンが付いているディスクを上書きします。
- 7 つのパスのアルゴリズム
  VxVM は合計 7 回ディスクに上書きします。各パスでは、VxVM はランダムに選択したデジタルパターンが付いているデータ、または前のパターンで 2 進補数が付いているデータを上書きします。

VxVM は現在、シン再生 LUN の細断はサポートしません。シン再生ディスクで細断操作を開始しようとすると、VxVM に警告メッセージが表示され、ディスクはスキップされます。

暗号化ボリュームのみを含むディスクに対し shred 操作をスキップできます。

VxVM ディスクの細断

機密データを含む VxVM（Veritas Volume Manager）ディスクを廃止するとき、VxVM はディスク上のデータを破壊する機能を提供します。

次の必要条件に注意してください。

- VxVM は、このシステムまたは共有ディスクグループで使用中のディスクは細断しません。

- VxVM は現在、シン再生 LUN の細断はサポートしません。シン再生ディスクで細断操作を開始しようとすると、VxVM に警告メッセージが表示され、ディスクはスキップされます。
- VxVM は、VxVM ディスクではないディスクは細断しません。
- VxVM は、マウントされているディスクは細断しません。
- ソリッドステートドライブ（SSD）を細断することを推奨しません。SSD デバイスを細断するには、細断操作に force（-f）オプションを使用します。

p.1053 の「データの細断について」を参照してください。

注意: ディスク上のすべてのデータは、ディスクを細断すると失われます。情報が別のストレージメディアにバックアップされて検証されているか、または情報がすでに不要になっているかどうかを確認します。

VxVM ディスクを細断するには

1 ディスクを細断するには、次を行います。

```
# /etc/vx/bin/vxdiskunsetup [-Cf] -o shred[=1|3|7] disk...
```

ここで、

force (-f) オプションにより、ソリッドステートドライブ (SSD) の細断が可能になります。

1、3、7 は、パスの番号に対応する細断オプションです。パスのデフォルト数は 1 です。

disk... は、1 つまたは複数のディスク名を表します。複数のディスク名を指定すると、vxdiskunsetup コマンドはそれらを順番に 1 つずつ処理します。

次に例を示します。

```
# /etc/vx/bin/vxdiskunsetup -o shred=3 hds9970v0_14
disk_shred: Shredding disk hds9970v0_14 with type 3
disk_shred: Disk raw size 2097807360 bytes
disk_shred: Writing 32010 (65536 byte size) pages and 0 bytes
to disk
disk_shred: Wipe Pass 0: Pattern 0x3e
disk_shred: Wipe Pass 1: Pattern 0xca
disk_shred: Wipe Pass 2: Pattern 0xe2
disk_shred: Shred passed random verify of 131072 bytes at
offset 160903168
```

vxdiskunsetup shred コマンドを実行して新しいタスクを設定します。

2 vxtask コマンドを実行して、細断操作の進行状況を監視できます。

次に例を示します。

```
# vxtask list
TASKID PTID TYPE/STATE  PCT    PROGRESS
   203   -  DISKSHRED/R 90.16% 0/12291840/11081728 DISKSHRED
nodg nodg
```

細断タスクは、一時停止、中止、再開できます。細断タスクをスロットル調整することはできません。

vxtask (1M) を参照してください

3 ディスクの細断操作が失敗すると、ディスクはラベルなしでエラー状態になる場合があります。

p.1057 の「ディスクの細断操作が失敗するとディスクがラベルなしになる」を参照してください。

ディスクの細断操作が失敗するとディスクがラベルなしになる

ディスクの細断操作によって、ディスクのラベルが破壊され、ラベルは再作成されます。細断操作が途中で中止された場合、またはシステムがクラッシュした場合、ディスクはラベルなしでエラー状態でなる場合があります。

ディスクのエラー状態を修正するには

1. 新しいラベルを手動で作成するか、または次のコマンドを使って VxVM 下でディスクを再初期化します。

   ```
   # /etc/vx/bin/vxdisksetup -i disk
   ```

2. 細断操作を開始します。ディスクが非 VxVM ディスクとして表示されたら、手順 1 で vxdisksetup コマンドを使ってディスクを再初期化してから、断片操作を再開してください。

   ```
   # /etc/vx/bin/vxdiskunsetup [-Cf] -o shred[=1|3|7] disk...
   ```

ディスクの削除と交換

障害の発生したディスクと交換用ディスクの様式は同一である必要があります。つまり、セクタあたりのバイト数、トラックあたりのセクタ数、シリンダあたりのトラック数とセクタ数、シリンダの総数、アクセス可能なシリンダの数が一致する必要があります。

メモ: 物理ディスクを取り外す前に、オペレーティングシステムまたはディスクアレイに固有のコマンドを実行する必要があります。

ディスクに障害が発生し、まだディスク全体で損傷していない場合は、ディスクを交換できます。この操作では、損傷したディスクまたは障害が発生したディスクを、所属するディスクグループから切断し、新しいディスクと交換します。必要に応じて、ディスクの交換を後日に延期できます。

ディスクを取り外したことでボリュームが無効になった場合、データをバックアップから復元できるように、ボリュームを再起動します。

詳しくは、『Veritas InfoScale トラブルシューティングガイド』を参照してください。

ディスクを交換するには、次の手順を実行します。

1. vxdiskadm メインメニューから[ディスクの交換のための削除(Remove a disk for replacement)]を選択します。
2. 次のプロンプトで、交換するディスクのディスク名を入力します(または、ディスクの一覧を表示するために、list を入力します)。

   ```
   Enter disk name [<disk>,list,q,?] mydg02
   ```

3 交換のために削除するディスクを選択すると、この操作で影響の及ぶボリュームすべてが表示されます。例を次に示します。

```
VxVM NOTICE V-5-2-371 The following volumes will lose mirrors
as a result of this operation:

home src

No data on these volumes will be lost.

The following volumes are in use, and will be disabled as a
result of this operation:

mkting

Any applications using these volumes will fail future
accesses. These volumes will require restoration from backup.

Are you sure you want do this? [y,n,q,?] (default: n)
```

ディスクを交換する際に、名前が表示されているボリュームを無効にし、データを保持せずに、ディスクを削除するには、y を入力するかリターンキーを押します。

ディスクの削除の操作を停止し、無効にされるボリュームに関連するデータのバックアップを作成するか、データを移動するには、n または q を入力しリターンキーを押します。

たとえば、ボリューム mkting を mydg02 以外のディスクに移動するには、次のコマンドを使います。

! 文字は一部のシェルでの特殊文字です。次の例では、bash シェルでこの文字をエスケープ処理する方法を示します。

```
# vxassist move mkting ¥!mydg02
```

ボリューム内のデータのバックアップの作成またはデータの移動を行った後に、手順 1 からもう一度始めます。

4 次のプロンプトで(提示された一覧から)交換用ディスクのデバイス名を選択するか、リターンキーを押してデフォルトディスクを選択します。このプロンプトで交換用ディスクを選択せずに物理ディスクを交換する場合は、none を入力します。

```
The following devices are available as replacements:
sdb

You can choose one of these disks now, to replace mydg02.
Select none if you do not wish to select a replacement disk.

Choose a device, or select none
[<device>,none,q,?] (default: sdb)
```

選択一覧に表示されていても、古いディスクドライブを交換用に選択しないでください。必要に応じて、新しいディスクの初期化を選択できます。

物理ディスクを交換する場合は、none を入力します。

p.1060 の「障害が発生したディスクまたは削除したディスクの交換」を参照してください。

5 手順 4 でディスクの交換を選択した場合は、次のプロンプトでリターンキーを押して、この操作を確認します。

```
VxVM NOTICE V-5-2-285 Requested operation is to remove mydg02
from group mydg. The removed disk will be replaced with disk
device
sdb.  Continue with operation? [y,n,q,?] (default: y)
```

vxdiskadm によって、元のディスクの削除中であることを示す次のようなメッセージが表示されます。

```
VxVM NOTICE V-5-2-265 Removal of disk mydg02 completed
successfully.
VxVM NOTICE V-5-2-260 Proceeding to replace mydg02 with device
sdb.
```

6 これで、フォーマットするディスクとして、異なるオペレーティングシステム間で移動できる CDS ディスク、移動できない sliced ディスク、simple ディスクのいずれかを選択できます。

```
Enter the desired format [cdsdisk,sliced,simple,q,?]
(default: cdsdisk)
```

使用目的に適した形式を入力します。多くの場合、デフォルトの cdsdisk を選択します。

7 次のプロンプトで、vxdiskadm により、デフォルトの 65536 ブロック(32 MB)をプライベートリージョンサイズに使うかどうかを尋ねるメッセージが表示されます。リターンキーを押して、デフォルト値の使用を確認するか、別の値を入力します(指定できる最大値は、524288 ブロックです)。

```
Enter desired private region length [<privlen>,q,?]
(default: 65536)
```

8 1 つ以上のミラープレックスがディスクから移動された場合は、FastResync を使ってプレックスを再同期するかどうかを確認するメッセージが表示されます。

```
Use FMR for plex resync? [y,n,q,?] (default: n) y
vxdiskadm displays the following success message:
VxVM NOTICE V-5-2-158 Disk replacement completed successfully.
```

9 次のプロンプトで、他のディスクを削除する(y)か、vxdiskadm メインメニューに戻る(n)かを指定します。

```
Remove another disk? [y,n,q,?] (default: n)
```

サブディスクの再配置処理を解除して、元の交換用ディスクに配置できます。

p.915 の「ホットリロケーションでスペアディスクのみを利用する設定」を参照してください。

障害が発生したディスクまたは削除したディスクの交換

次の手順は、障害が発生したディスクまたは削除したディスクを交換する方法を説明しています。

障害が発生したディスクまたは削除したディスクと交換するディスクを指定するには、次の手順を実行します。

1 vxdiskadm メインメニューで[障害が発生したディスクまたは削除したディスクの交換(Replace a failed or removed disk)]を選択します。

2 次のプロンプトで、交換するディスクのディスク名を入力します(または、ディスクの一覧を表示するために、list を入力します)。

```
Select a removed or failed disk [<disk>,list,q,?] mydg02
```

3 vxdiskadm プログラムは、交換用ディスクとして使用可能なディスクデバイスのデバイス名を表示します。ご使用のシステムによっては、ここに示した例とは異なるデバイス名が使われることがあります。ディスクのデバイス名を入力するか、リターンキーを押してデフォルトのデバイスを選択します。

```
The following devices are available as replacements:
sdb sdk

You can choose one of these disks to replace mydg02.
Choose "none" to initialize another disk to replace mydg02.

Choose a device, or select "none"
[<device>,none,q,?] (default: sdb)
```

4 交換用ディスクがすでに初期化されているかどうかに応じて、次の手順の中から適切なものを実行します。

- ディスクが事前に初期化されていない場合は、次のプロンプトでリターンキーを押してディスクを交換します。

```
VxVM INFO V-5-2-378 The requested operation is to initialize
disk device sdb and to then use that device to
replace the removed or failed disk mydg02 in disk group mydg.
Continue with operation? [y,n,q,?] (default: y)
```

- ディスクがすでに初期化されている場合は、次のプロンプトでリターンキーを押してディスクを交換します。

```
VxVM INFO V-5-2-382 The requested operation is to use the
initialized device sdb to replace the removed or
failed disk mydg02 in disk group mydg.
Continue with operation? [y,n,q,?] (default: y)
```

5 これで、フォーマットするディスクとして、異なるオペレーティングシステム間で移動できる CDS ディスク、移動できない sliced ディスク、simple ディスクのいずれかを選択できます。

```
Enter the desired format [cdsdisk,sliced,simple,q,?]
(default: cdsdisk)
```

使用目的に適した形式を入力します。多くの場合、デフォルトの cdsdisk を選択します。

6 次のプロンプトで、vxdiskadm により、デフォルトの 65536 ブロック(32 MB)をプライベートリージョンサイズに使うかどうかを尋ねるメッセージが表示されます。リターンキーを押して、デフォルト値の使用を確認するか、別の値を入力します (指定できる最大値は、524288 ブロックです)。

```
Enter desired private region length [<privlen>,q,?]
(default: 65536)
```

7 この後、vxdiskadm プログラムにより、ディスクの交換処理が続行されます。処理に成功すると、次のメッセージが返されます。

```
VxVM NOTICE V-5-2-158 Disk replacement completed successfully.
```

次のプロンプトで、他のディスクを交換する(y)か、vxdiskadm メインメニューに戻る(n)かを指定します。

```
Replace another disk? [y,n,q,?] (default: n)
```

42

ルータビリティ

この章では以下の項目について説明しています。

ルートディスクのカプセル化 (RDE) がサポートされない

Linux ディストリビューションのルートディスクのカプセル化 (RDE) は 7.3.1 リリース以降サポートされません。

ディスクのカプセル化

警告: ディスクのカプセル化では、システムの再ブートが何回か必要になります。ユーザーに不都合が生じないように、この手続きの実行をスケジュールしてください。

この項では、VxVM 用ディスクのカプセル化の方法を説明します。カプセル化では、ディスクを VxVM の制御下に置く際に、ディスク上のすべての既存データを保持します。

ルートディスクをカプセル化して、VxVM の制御下に置くことができます。ただし、カプセル化可能なルートディスクのレイアウトと設定には制限があります。

p.1071 の「Linux でのルータビリティの使用に関する制限」を参照してください。

p.1070 の「ルータビリティ」を参照してください。

ルートディスクをカプセル化する前に、ルートディスクのパーティションテーブルをプリントアウトするには、format または fdisk コマンドを使います。詳細については、該当するマニュアルページを参照してください。今後、元のルートディスクを再作成することが必要になった場合は、この情報が必要になります。

カプセル化されたルートディスクに関連付けられているボリューム（rootvol、usrvol、varvol、optvol、swapvol など）を拡大または縮小することはできません。これは、ボリュームは物理的なディスクパーティションにマップされており、連続している必要があるためです。

msdos のラベルの付いたディスクは、auto:sliced ディスクとしてカプセル化できます。ただし、そのディスクに、パブリックリージョンへの割り当てが可能な 1 つ以上の予備のプライマリパーティションと、プライベートリージョンへの割り当てが可能な 1 つの予備のプライマリパーティションまたは論理パーティションが存在することが前提となります。

sun のラベルの付いたディスクも auto:sliced ディスクとしてカプセル化できます。ただし、そのディスクに、パブリックリージョンとプライベートリージョンへの割り当てが可能な 2 つ以上の予備のスライスが存在することが前提となります。

GPT（GUID Partition Table）のラベルの付いた EFI（Extensible Firmware Interface）ディスクも auto:sliced ディスクとしてカプセル化できます。ただし、そのディスクに、パブリックリージョンとプライベートリージョンへの割り当てが可能な 2 つ以上の予備のスライスが存在することが前提となります。

パーティションテーブル内のパブリックリージョンのエントリには、ディスク上の追加領域は必要ありません。代わりに、そのエントリが既存のパーティションで使われているディスク領域を表す（またはカプセル化する）ようにします。

パブリックリージョンとは異なり、プライベートリージョンのパーティションでは、既存のパーティションまたはスライスのいずれにも属さない少量の空き領域がディスクの先頭または末尾に必要です。デフォルトでは、プライベートリージョンに必要な領域は 32 MB です。この値は、整数のシリンダ数に切り上げられます。最近のディスクでは、ほとんどの場合 1 シリンダで十分です。

VxVM 用のディスクをカプセル化するには、次の手順を実行します。

1 ルートディスクをカプセル化する前に、VxVM が使う、デバイスの名前の付け方を永続的になるように設定します。

```
# vxddladm set namingscheme={osn|ebn} persistence=yes
```

たとえば、エンクロージャに基づく名前の付け方と共に永続的な名前の付け方を使うには、次の手順を実行します。

```
# vxddladm  set namingscheme=ebn persistence=yes
```

2 vxdiskadm のメインメニューで、[1 つ以上のディスクのカプセル化(Encapsulate one or more disks)]を選択します。

ご使用のシステムによっては、ここで示した例とは異なるデバイス名が使われることがあります。

次のプロンプトで、カプセル化するディスクのディスクデバイス名を入力します。

```
Select disk devices to encapsulate:
[<pattern-list>,all,list,q,?] device name
```

pattern-list には、単一ディスクまたは一連のディスクを指定できます。*pattern-list* が複数の項目で構成されている場合は、スペースで区切って項目を切り離す必要があります。

カプセル化するディスクのアドレス(デバイス名)が不明な場合は、使用可能なディスクの完全な一覧を要求するプロンプトで、l または list を入力します。

3 この操作を続行するには、次のプロンプトで、y を入力します(またはリターンキーを押します)。

```
Here is the disk selected. Output format: [Device]
device name

Continue operation? [y,n,q,?] (default: y) y
```

4 次のプロンプトで、ディスクの追加先となるディスクグループを選択します。

```
You can choose to add this disk to an existing disk group or to
a new disk group. To create a new disk group, select a disk
group name that does not yet exist.

Which disk group [<group>,list,q,?]
```

5 次のプロンプトで、リターンキーを押してデフォルトのディスク名を使うか、ディスク名を入力します。

```
Use a default disk name for the disk? [y,n,q,?] (default: y)
```

6 操作を続行するには、次のプロンプトで y を入力します(またはリターンキーを押します)。

```
The selected disks will be encapsulated and added to the
disk group name disk group with default disk names.

device name

Continue with operation? [y,n,q,?] (default: y) y
```

7 カプセル化の続行を確認するには、次のプロンプトで y を入力します(またはリターンキーを押します)。

```
The following disk has been selected for encapsulation.
 Output format: [Device]

device name

Continue with encapsulation? [y,n,q,?] (default: y) y
```

次のようなメッセージで、ディスクが VxVM での使用のためにカプセル化されることが確認されます。

```
The disk device device name will be encapsulated and added to
the disk group diskgroup with the disk name diskgroup01.
```

8 非ルートディスクの場合は、ディスクのフォーマットを、異なるオペレーティングシステム間で移動できる CDS ディスクとして行うか、移動できない sliced ディスクとして行うかを選択できます。

```
Enter the desired format [cdsdisk,sliced,simple,q,?]
(default: cdsdisk)
```

使用目的に適した形式を入力します。多くの場合、デフォルトの cdsdisk を選択します。ルートディスク、ブートディスクまたはスワップディスクの場合は、sliced 形式です。

9 次のプロンプトで、vxdiskadm により、デフォルトの 65536 ブロック(32 MB)をプライベートリージョンサイズに使うかどうかを尋ねるメッセージが表示されます。リターンキーを押して、デフォルト値の使用を確認するか、別の値を入力します (指定できる最大値は、524288 ブロックです)。

```
Enter desired private region length [<privlen>,q,?]
(default: 65536)
```

10 cdsdisk 形式を指定した場合は、手順 8 で、この形式に変換できなかった場合に実行するアクションの入力を求めるプロンプトが表示されます。

```
Do you want to use sliced as the format should cdsdisk
fail? [y,n,q,?] (default: y)
```

y を入力した場合、CDS ディスクとしてカプセル化できないディスクは sliced ディスクとしてカプセル化されます。それ以外の場合は、カプセル化が失敗します。

11 この後、vxdiskadm によって、ディスクのカプセル化が続行されます。次のようなコマンドを実行して、できるだけ早い段階でシステムを再ブートしてください。

```
# shutdown -r now
```

/etc/fstab ファイルが更新され、カプセル化されたファイルシステムをマウントするためのボリュームデバイスが追加されます。場合によっては、バックアップスクリプト、データベース、手動作成したスワップデバイスなどの参照項目を更新する必要があります。元の /etc/fstab ファイルは、/etc/fstab.b4vxvm として保存されます。

12 次のプロンプトで、続けてディスクをカプセル化する(y)か、vxdiskadm メインメニューに戻る(n)かを指定します。

```
Encapsulate other disks? [y,n,q,?] (default: n) n
```

ディスクをカプセル化するときに使うデフォルトのレイアウトは変更できます。

ディスクのカプセル化の失敗

ある状況の下では、プライベートリージョンを格納できるだけの空き領域がディスク上にないためにディスクのカプセル化に失敗することがあります。十分な空き領域がない場合は、次のようなエラーメッセージを表示してカプセル化プロセスは突然終了します。

```
VxVM ERROR V-5-2-338 The encapsulation operation failed with the
following error:
It is not possible to encapsulate device, for the following
reason:
<VxVM vxslicer ERROR V-5-1-1108 Unsupported disk layout.>
```

1 つの解決策は、nopriv フォーマットでディスクを設定することです。

p.1068 の「nopriv ディスクを使ったカプセル化」を参照してください。

nopriv ディスクを使ったカプセル化

カプセル化すると、指定されたディスク上の既存パーティションがボリュームに変換されます。いずれかのパーティションにファイルシステムがある場合は、/etc/fstab エントリが変更され、代わりにファイルシステムがボリュームにマウントされます。

ディスクをカプセル化するためには、VxVM がディスクの識別情報と設定情報の保存に使うプライベートリージョンを格納できるだけの十分な空き領域（デフォルトでは 32 MB）がディスク上に必要です。この空き領域を、他のパーティションに含めることはできません

vxencap（1M）マニュアルページを参照してください。

vxdisk ユーティリティを使って、VxVM のプライベートリージョンのパーティション用に使用可能な領域のないディスクを、カプセル化できます。カプセル化するには、プライベートリージョンを持たない nopriv デバイスとしてディスクを設定します。

nopriv デバイスを使う欠点は、VxVM でディスクのアドレスやコントローラの変更をトラッキングできないことです。通常、VxVM では、物理ディスク上のプライベートリージョンに格納された識別情報を使い、物理ディスクの配置の変更をトラッキングします。nopriv デバイスにはプライベートリージョンが存在しないため、物理ディスク上に格納された識別情報も存在せず、トラッキングを行うことができません。

nopriv デバイスの用途の 1 つは、VxVM を使ってデータをディスク外に移動できるように、ディスクをカプセル化することです。ディスク上に使用可能な領域ができたら、nopriv デバイスを削除し、そのディスクを標準ディスクデバイスとしてカプセル化します。

nopriv デバイスのみでディスクグループを構成することはできません。これは、nopriv デバイスに、ディスクグループの設定情報を格納する領域が存在しないためです。設定情報は、ディスクグループ内の少なくとも 1 つのディスクに格納されている必要があります。

カプセル化する nopriv ディスクの作成

警告: nopriv ディスクを使ってルートディスクをカプセル化しないでください。ルートディスクにプライベートリージョン用の十分な空き領域がない場合は、代わりにスワップ領域の一部を使うことができます。

カプセル化する nopriv ディスクを作成するには

1. パーティションが存在しない場合は、VxVM を使って、アクセスする領域用のディスクパーティションを設定します。
2. 次のコマンドを使って、VM ディスクをパーティションにマップします。

   ```
   # vxdisk define partition-device type=nopriv
   ```

   partition-device は、/dev/dsk ディレクトリのデバイスのベース名です。

   たとえば、ディスクデバイス sdc のパーティション 3 をマップするには、次のコマンドを使います。

   ```
   # vxdisk define sdc3 type=nopriv
   ```

nopriv ディスク上の他のパーティション用のボリュームの作成

nopriv ディスク上の他のパーティションにボリュームを作成するには、次の手順を実行します。

1. ディスクグループに該当するパーティションを追加します。
2. カプセル化したパーティション内でパーティションが存在する位置を決定します。
3. パーティション上にデータを保持しない場合は、vxassist を使って、必要なサイズのボリュームを作成します。

警告: vxassist は、デフォルトで、作成するボリュームのデータ領域を再初期化します。パーティション上に保持すべきデータがある場合は、vxassist を使わないでください。代わりに、vxmake を使ってボリュームを作成し、vxvol init active コマンドでボリュームを起動します。

カプセル化後に RHEL 7 環境でデバイス名の形式が変わる

RHEL 7 とサポート対象の RHEL 互換配布では、ルートディスクをカプセル化すると /etc/fstab ファイルのボリュームの形式が変わります。

表 42-1 に、RHEL 7 とサポート対象の RHEL 互換配布環境での変更点を一覧表示します。

表 42-1 RHEL 7 とサポート対象の RHEL 互換配布環境でのボリューム形式の変更点

RHEL 7 より前	RHEL 7
ボリューム形式: /dev/vx/dsk/bootdg/<volume>	ボリューム形式: /dev/vx_dsk_bootdg_<volume>
rootdisk に 2 つのパーティションがある /etc/fstab ファイルの内容。すなわち、/ と swap。 # cat /etc/fstab /dev/vx/dsk/bootdg/rootvol ¥ / ext4 defaults 1 1 /dev/vx/dsk/bootdg/swapvol ¥ swap swap defaults 0 0 #NOTE: volume rootvol (/) ¥ encapsulated partition sda1 #NOTE: volume swapvol (swap) ¥ encapsulated partition sda2	rootdisk に 2 つのパーティションがある /etc/fstab ファイルの内容。すなわち、/ と swap。 # cat /etc/fstab /dev/vx_dsk_bootdg_rootvol ¥ / ext4 defaults 1 1 /dev/vx_dsk_bootdg_swapvol ¥ swap swap defaults 0 0 #NOTE: volume rootvol (/) ¥ encapsulated partition sda1 #NOTE: volume swapvol (swap) ¥ encapsulated partition sda2

メモ: /etc/fstab のデバイス名の形式は変わっても、mount ユーティリティの出力に変更はありません。mount ユーティリティは、引き続き古い形式でマウント済みのボリュームを表示します。

ルータビリティ

VxVM では、ルートファイルシステム、swap デバイス、ルートディスク上の他のファイルシステムに属する各種ファイルを VxVM の制御下に置くことができます。この機能は、ルータビリティと呼ばれます。ルートディスク(すなわち、ルートファイルシステムを含むディスク)をカプセル化の過程を経て VxVM の制御下に置くことができます。

カプセル化すると、ディスク上の既存のパーティションがボリュームに変換されます。一度 VxVM の制御下に置かれると、root デバイスおよび swap デバイスは、ボリュームとなり、他の VxVM ボリュームと同じ特性を持ちます。スワップ領域用として構成されたボリュームはスワップボリュームと呼ばれ、ルートファイルシステムを含むボリュームはルートボリュームと呼ばれます。

メモ: ルートディスクのカプセル化は、ミラー化も併せて行う場合にかぎり実行してください。ルートディスクのカプセル化自体には利点はありません。

rootvol ボリュームおよび swapvol ボリュームは、システムの正常起動に必要なルートディスクの他の部分(/usr など)とともにミラー化できます。これにより、ディスクエラーが発生した場合の冗長性とリカバリ機能が完備されます。ミラー化がないと、root、swap、usr パーティションのいずれかが損失した場合に、残りのディスクからシステムを起動できなくなります。

ブートに必要なディスクドライブをミラー化することにより、単一ディスクのエラーでシステムが使用不能になるのを回避できます。起動に必要なディスクについては、使用可能な別のディスク上にミラーを作成することをお勧めします(vxdiskadm コマンドを使用)。root パーティションおよび swap パーティションを含むディスクに障害が発生した場合は、これらのパーティションのミラーを含むディスクからシステムを再ブートできます。

カプセル化されたルートディスクに障害が発生した場合にシステムをリカバリするには、特別な手順を適用する必要があります。

『Veritas InfoScale トラブルシューティングガイド』を参照してください。

Linux でのルータビリティの使用に関する制限

msdos ディスクラベルの付いた起動可能なルートディスクには、最大 4 つのプライマリパーティションを含めることができます。SCSI ディスクの場合は /dev/sdx1 から /dev/sdx4 まで、IDE ディスクの場合は /dev/hdx1 から /dev/hdx4 までのパーティションです。5 つ以上のパーティションが必要な場合は、プライマリパーティションを拡張パーティションとして設定できます。拡張パーティションには、SCSI ディスクで最大 11 個の論理パーティション(/dev/sdx5 から /dev/sdx15)、IDE ディスクで最大 12 個の論理パーティション(/dev/hdx5 から /dev/sdx16)を含めることができます。

メモ: ルートディスクのカプセル化では、GPT のラベルの付いた EFI ディスクはサポートされません。

ルートディスクをカプセル化するには、パブリックリージョン用の未使用のプライマリパーティション 1 つと、プライベートリージョン用の未使用のプライマリパーティション 1 つまたは未使用の論理パーティション 1 つが必要です。

パーティションテーブル内のパブリックリージョンのエントリには、ディスク上の追加領域は必要ありません。代わりに、そのエントリが既存のパーティションで使われているディスク領域を表す(またはカプセル化する)ようにします。

パブリックリージョンとは異なり、プライベートリージョンのパーティションには、比較的少量のディスク空き領域が必要です。デフォルトでは、プライベートリージョンに必要な領域は

32 MB です。この値は、整数のシリンダ数に切り上げられます。最近のディスクでは、ほとんどの場合 1 シリンダで十分です。

カプセル化するルートディスクのパーティションレイアウトは、次の必要条件を満たす必要があります。

- パブリックリージョン用の未使用のプライマリパーティション 1 つ。
- プライベートリージョンへの割り当てが可能な空きディスク領域または swap パーティション。拡張パーティション内に空き領域または swap パーティションが存在しない場合は、プライベートリージョン用に未使用のプライマリパーティションが 1 つ必要になります。存在する場合は、未使用の論理パーティションが 1 つ必要です。

レイアウトの基準を満たしていないルートディスクをカプセル化しようとすると、vxencap または vxdiskadm コマンドによって次のエラーメッセージが表示されます。

```
Cannot find appropriate partition layout to allocate space
for VxVM public/private partitions.
```

以下の項では、ルートディスクレイアウトでカプセル化がサポートされる例とサポートされない例を紹介しています。

- p.1073 の「カプセル化がサポートされるルートディスクのレイアウト例」を参照してください。
- p.1076 の「カプセル化がサポートされないルートディスクのレイアウト例」を参照してください。

Linux でルータビリティを使う場合、以下のような重要な制限があります。

- ルートディスクのカプセル化は、標準の SCSI または IDE インターフェースを備えたデバイスでのみサポートされています。ベンダー独自のインターフェースを備えた大部分のデバイスでは、サポートされていません。ただし、COMPAQ SMART コントローラと SMARTII コントローラは、/dev/ida/cXdXpX と /dev/cciss/cXdXpX という形式のデバイス名を使っており、ルートディスクのカプセル化をサポートしています。
- ルートディスクのカプセル化は、msdos または sun のラベルの付いたディスクでのみサポートされています。gpt のラベルの付いたディスクではサポートされていません。
- root、boot および swap パーティションが同じディスク上に存在する必要があります。
- SCSI と IDE ディスクのブートローダーとして、GRUB または LILO ブートローダーを使う必要があります。
- ブートローダー設定ファイル内のメニューエントリが有効である必要があります。
- ブートローダー設定ファイルが、ルートディスクのカプセル化プロセスで編集されないようにする必要があります。
- /boot パーティションが、BIOS が最初に認識するディスクに存在し、このパーティションがプライマリパーティションである必要があります。

一部のシステムはローカルディスクを無視するように設定できません。ローカルディスクはカプセル化するときに削除する必要があります。マルチパス設定の変更(複数の HBA システムの場合)は同じ影響をもたらす可能性があります。VxVM がサポートするのは、初期のブートストラップインストール設定がルートのカプセル化のために変更されていないシステムのみです。

- ブートローダーは、ルートディスクまたはルートディスクのミラー上に存在するマスターブートレコード(MBR)内に置かれている必要があります。
- GRUB ブートローダーを使う場合、ルートディスクをカプセル化するには、/boot ディレクトリの root デバイスの場所を、最初のディスクドライブ sd0 または hd0 に設定する必要があります。
- LILO または ELILO ブートローダーを使う場合、ルートディスクをカプセル化した後で、FALLBACK、LOCK または -R オプションを使わないでください。

警告: LILO では VxVM ボリュームのレイアウトが識別されないため、LILO で FALLBACK、LOCK または -R オプションを使うと、システムが起動しなくなる場合があります。

- A/P(アクティブ/パッシブ)アレイ内のセカンダリコントローラのみに接続された、カプセル化されたルートディスクからの起動は、サポートされていません。
- Red Hat のデフォルトのレイアウトでは、ルータビリティを実装できません。ルートディスクのレイアウトを変更する場合は、カプセル化を行う前に、そのルートディスクが起動可能であることを確認してください。
  p.1076 の「例 1: カプセル化がサポートされないルートディスクのレイアウト」を参照してください。
- カプセル化した後のルートディスクから、ボリュームを割り当てないでください。ディスクに保存されているパーティション情報が破棄されます。
- デバイスの名前の付け方は永続的になるように設定する必要があります。

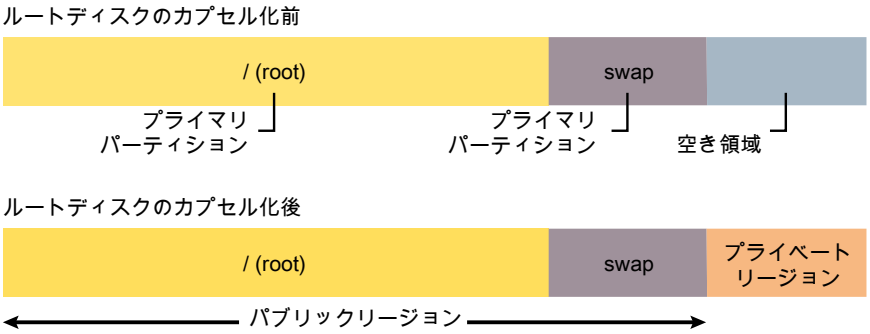
カプセル化がサポートされるルートディスクのレイアウト例

次の例は、カプセル化がサポートされるルートディスクのレイアウトを示しています。

例 1: カプセル化がサポートされるルートディスクのレイアウト

図 42-1 に、サポートされるレイアウトの例を示します。このレイアウトでは、root と swap が 2 つのプライマリパーティション上に設定され、ディスクにいくらかの空き領域があります。

図 42-1　2 つのプライマリパーティション上に設定された root と swap、およびディスク上の空き領域

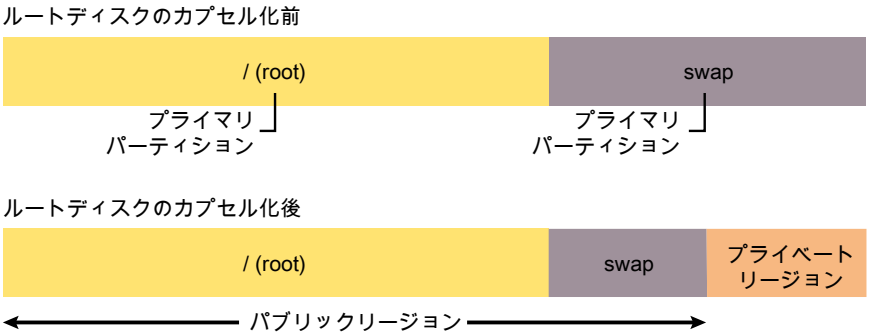


2 つのプライマリパーティションは、/ と swap で使われています。未使用のプライマリパーティションが 2 つあり、ディスク上にはプライベートリージョン用のプライマリパーティションに割り当てることのできる空き領域があります。

例 2: カプセル化がサポートされるルートディスクのレイアウト

図 42-2 に、サポートされるレイアウトの例を示します。このレイアウトでは、root と swap が 2 つのプライマリパーティション上に設定され、ディスクに空き領域がありません。

図 42-2　2 つのプライマリパーティション上に設定された root と swap（空き領域なし）

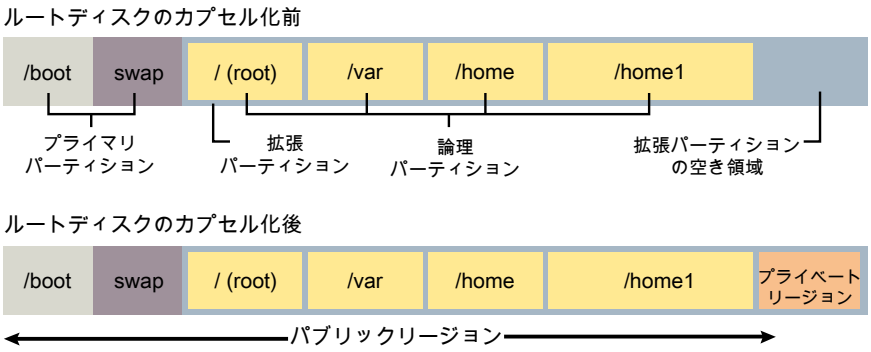


2 つのプライマリパーティションは、/ と swap で使われています。2 つの未使用のプライマリパーティションがあり、swap パーティションの末尾の領域を使うことにより、新しいプライマリパーティションにプライベートリージョンを割り当てることができます。

例 3: カプセル化がサポートされるルートディスクのレイアウト

図 42-3 に、サポートされるレイアウトの例を示します。このレイアウトでは、boot と swap が 2 つのプライマリパーティション上に設定され、拡張パーティションにいくらかの空き領域があります。

図 42-3 2 つのプライマリパーティション上に設定された boot と swap、および拡張パーティション内の空き領域

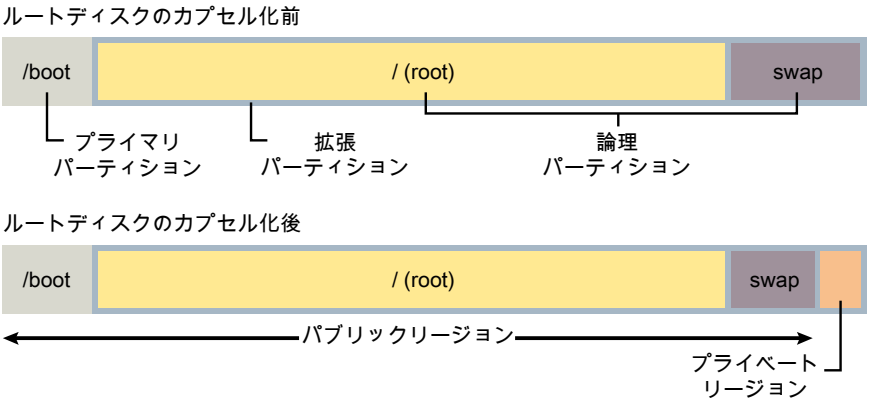


3 つのプライマリパーティションは、/boot、swap、および拡張パーティションで使われています。拡張パーティションには、root を含む 4 つのファイルシステムがあります。拡張プライマリパーティションの末尾の空き領域を使って、プライベートリージョン用の新しい論理パーティションを作成できます。

例 4: カプセル化がサポートされるルートディスクのレイアウト

図 42-4 に、サポートされるレイアウトの例を示します。このレイアウトでは、boot が 1 つのプライマリパーティション上に設定され、root と swap が拡張パーティションに設定されています。

図 42-4 プライマリパーティション上に設定された boot、および拡張パーティション内に設定された root と swap



2 つのプライマリパーティションは、/boot と拡張パーティションで使われています。拡張パーティションには、root ファイルシステムとスワップ領域があります。swap パーティションの末尾の領域を使うことにより、プライベートリージョン用の新しい論理パーティションを作成できます。

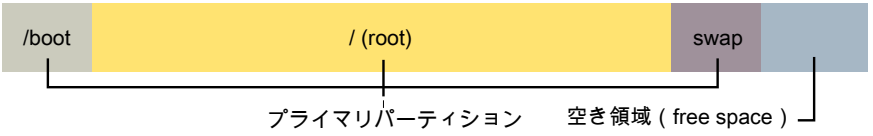
カプセル化がサポートされないルートディスクのレイアウト例

次の例は、カプセル化がサポートされないルートディスクのレイアウトを示しています。

例 1: カプセル化がサポートされないルートディスクのレイアウト

図 42-5 に、サポートされないレイアウトの例を示します。このレイアウトでは、boot、swap、root が 3 つのプライマリパーティション上に設定され、ディスクにいくらかの空き領域があります。

図 42-5 3 つのプライマリパーティション上に設定された boot、swap、root、およびディスク上の空き領域

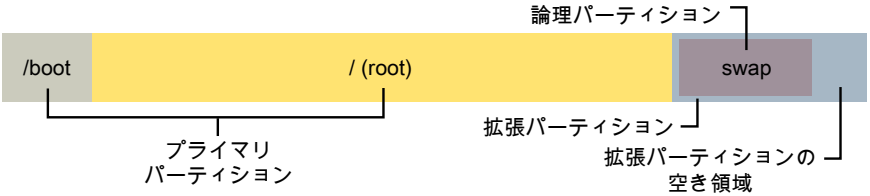


このレイアウトは、Red Hat のデフォルトのレイアウトと同じであり、使用可能な予備のプライマリパーティションが 1 つしか存在せず、拡張パーティション内に swap パーティションも空き領域も存在しないため、カプセル化できません。

図 42-6 にこの問題の解決策を示します。swap パーティションまたは空き領域を拡張パーティションとして設定し、スワップ領域を論理パーティションに移動します。論理パー

ティションには、プライベートリージョンを格納できるだけの十分な空き領域を残しておきます。

図 42-6 解決策: swap を論理パーティションとして再設定する

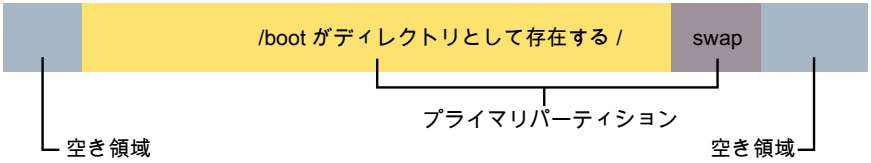


元の swap パーティションは削除する必要があります。再設定が完了すると、このルートディスクをカプセル化できます。

p.1075 の「例 3: カプセル化がサポートされるルートディスクのレイアウト」を参照してください。

図 42-7 にもう 1 つの解決策を示します。/boot を / の下にディレクトリとして再作成し、/boot パーティションを削除して、新しい /boot の位置を使うように LILO または GRUB を再設定します。

図 42-7 解決策: /boot をディレクトリとして再設定する



警告: ルートファイルシステムの先頭が最初の 1024 シリンダ内にない場合は、/boot を移動するとシステムが起動しなくなることがあります。

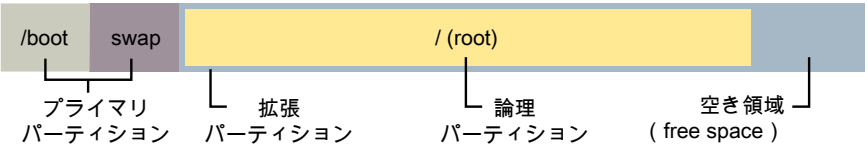
再設定が完了すると、このルートディスクをカプセル化できます。

p.1073 の「例 1: カプセル化がサポートされるルートディスクのレイアウト」を参照してください。

例 2: カプセル化がサポートされないルートディスクのレイアウト

図 42-8 に、サポートされないレイアウトの例を示します。このレイアウトでは、boot と swap が 2 つのプライマリパーティション上に設定され、拡張パーティションに空き領域がありません。

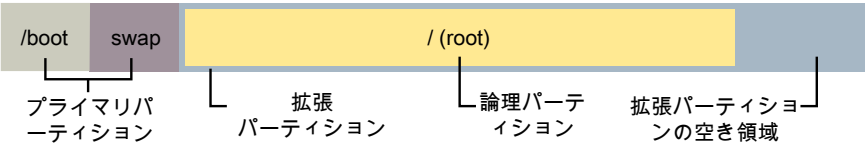
図 42-8 2 つのプライマリパーティション上に設定された boot と swap、および空き領域のない拡張パーティション



このレイアウトは、使用可能な予備のプライマリパーティションが 1 つしか存在せず、拡張パーティション内に swap パーティションも空き領域も存在しないため、カプセル化できません。

図 42-9 にこの問題の解決策を示します。パーティション設定ツールを使って、拡張パーティションをディスク上の空き領域まで拡大します。

図 42-9 解決策: 拡張パーティションを拡大する



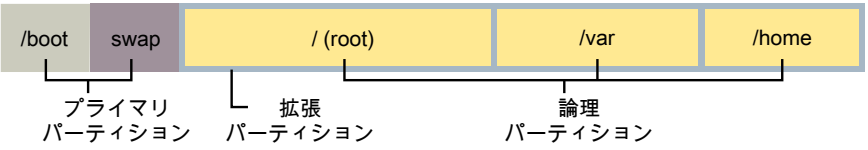
`root` ファイルシステムが含まれている論理パーティションの境界を変更しないように注意する必要があります。再設定が完了すると、このルートディスクをカプセル化できます。

p.1075 の「例 3: カプセル化がサポートされるルートディスクのレイアウト」を参照してください。

例 3: カプセル化がサポートされないルートディスクのレイアウト

図 42-10 に、サポートされないレイアウトの例を示します。このレイアウトでは、`boot` と `swap` が 2 つのプライマリパーティション上に設定され、ディスクに空き領域がありません。

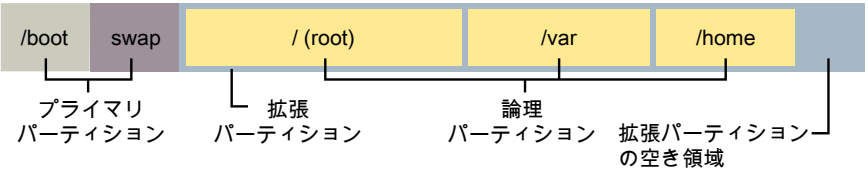
図 42-10 2 つのプライマリパーティション上に設定された boot と swap（空き領域なし）



このレイアウトは、使用可能な予備のプライマリパーティションが 1 つしか存在せず、拡張パーティション内に swap パーティションも追加の論理パーティション用の空き領域も存在しないためカプセル化できません。

図 42-11 にこの問題の解決策を示します。1 つ以上の既存のファイルシステムと対応する論理パーティションを縮小します。

図 42-11　　解決策: 既存の論理パーティションを縮小する



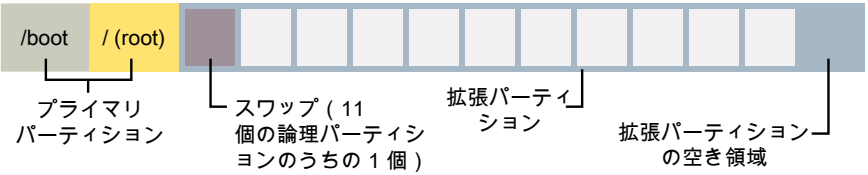
既存の論理パーティションを縮小すると、拡張パーティション内の領域がプライベートリージョン用に解放されます。再設定が完了すると、このルートディスクをカプセル化できます。

p.1075 の「例 3: カプセル化がサポートされるルートディスクのレイアウト」を参照してください。

例 4: カプセル化がサポートされないルートディスクのレイアウト

図 42-12 に、2 つのプライマリパーティションに設定された `boot` と `root`（使用可能な論理パーティションなし）の例を示します。

図 42-12　　2 つのプライマリパーティション上に設定された boot と swap（使用可能な論理パーティションなし）



SCSI ディスクを使う場合、このレイアウトは、使用可能な予備のプライマリパーティションが 1 つしか存在しないためカプセル化できません。また、論理パーティション上に swap が設定され、拡張パーティションに空き領域がある場合でも、追加の論理パーティションは作成できません。この問題は、12 個の論理パーティションが作成されている IDE ディスクでも発生します。

この問題を解決するには、既存の論理パーティションの 1 つからすべてのデータを退避させて、この論理パーティションを削除します。これにより、プライベートリージョン用に 1 つの論理パーティションが使用可能になります。その後、このルートディスクをカプセル化できます。

p.1075 の「例 3: カプセル化がサポートされるルートディスクのレイアウト」を参照してください。

p.1075 の「例 4: カプセル化がサポートされるルートディスクのレイアウト」を参照してください。

ルートボリュームの起動

オペレーティングシステムの起動時に、root ファイルシステムと swap 領域が使用可能な状態にあることを条件として、vxconfigd デーモンによる VxVM 設定のロードやボリュームの起動が開始されます。rootvol ボリュームおよび swapvol ボリュームがシステム起動時に通常のパーティションとして認識され、普通のディスクパーティションと同様にオペレーティングシステムからアクセスできる状態になっている必要があります。

この制限があるため、各 rootvol プレックスおよび swapvol プレックスは、単一のパーティションにマップされる、ディスク上の連続した領域から作成する必要があります。起動に使う rootvol ボリュームまたは swapvol ボリュームのプレックスをストライプ化、連結または分散することはできません。起動に使う可能性のあるこれらのプレックスのミラーについても同様です。

デフォルトのブートディスク以外のディスクから起動するようにシステム BIOS を設定する方法については、ハードウェアベンダーのマニュアルを参照してください。

起動時のボリュームの制限

ルートディスク上のボリュームは、設定について非常に特殊な制限がある点で他のボリュームとは異なります。

- ルートボリューム(rootvol)は、デフォルトのディスクグループ bootdg 内に存在する必要があります。bootdg 以外のディスクグループ内に rootvol という名前の別のボリュームを作成することはできますが、システムの起動に使えるのは、bootdg 内のボリューム rootvol のみです。
- rootvol ボリュームと swapvol ボリュームのマイナーデバイス番号は、それぞれ 0 と 1 に設定されます。ルートディスク上の他のボリュームには、特定のマイナーデバイス番号がありません。
- ルートディスクデバイス上のボリュームの制限付きミラーを作成すると、そのミラーに対応するオーバーレイパーティションが作成されます。オーバーレイパーティションに含まれる領域は、制限付きミラーで使っているディスク領域と正確に一致します。デフォルトのボリューム設定では、起動時に rootvol、varvol、usrvol および swapvol の各ボリュームが完全に設定されるまでの間、ディスク上のデータへのアクセスにはオーバーレイパーティションが使われます。
- 処理効率目的で、rootvol デバイスにストライプ化ミラーを追加することは可能ですが、プライマリプレックスをストライプ化することまたは、プライマリプレックスに障害が発生した場合にシステムのリカバリや起動に必要となる rootvol のミラーをストライプ化することはできません。

- rootvol および swapvol に、連続していない複数のサブディスクを持つプライマリプレックスを分散したり、格納することはできません。カプセル化されたブートディスクに関連付けられているボリューム（rootvol、usrvol、varvol、optvol、swapvol など）は下位の物理的なディスクパーティションにマップされており、連続している必要があるため、拡大または縮小することはできません。これを解決するには、ブートディスクのカプセル化を解除し、ブートディスクのパーティションを設定しなおしてから（必要に応じて、パーティションを拡張または縮小）、再度カプセル化します。
- ブートディスクの一部をミラー化する場合は、ミラーが作成されるディスクに、元のプレックス上のデータを保存する十分な大きさが必要です。領域が足りないと、ミラー化できないことがあります。
- ルートディスク上のボリュームでは、DRL（dirty region logging）を使えません。

これらの必要条件に加えて、root、usr、var、opt および swap の各ボリュームに、少なくとも 1 つの連続した（できればシリンダ位置を合わせた）ミラーを作るのもよい方法です。これにより、通常のディスクパーティションに、これらのボリュームを容易に変換しなおすことができます（たとえば、オペレーティングシステムのアップグレード時）。

ルートディスクの冗長性の確立

1 つのディスクに障害が発生した場合に備えて、ルートディスクのアクティブなバックアップを作成できます。vxrootadm コマンドを使って、ブートされるルートディスクや、ルートディスクグループ内の他のボリュームのミラーを作成します。

バックアップルートディスクを作成するには

◆ vxrootadm addmirror コマンドを使ってミラーを作成します。

```
# vxrootadm [-v] [-Y] addmirror targetdisk
```

ディザスタリカバリ用のアーカイブ化されたバックアップルートディスクの作成

ルートディスクのアクティブなバックアップを用意しておくことに加えて、ブート可能なルートディスクをアーカイブ化したバックアップコピーを保管できます。vxrootadm コマンドを使い、ブートされたルートディスクのスナップショットを作成します。これでミラーが作成され、別個のディスクグループに切り離されます。

アーカイブ化されたバックアップルートディスクを作成するには

1. ブートされたルートディスクグループにディスクを追加します。
2. ブートされたルートディスクのスナップショットを作成します。

```
# vxrootadm [-v] mksnap targetdisk targetdg
```

3. ディザスタリカバリ用のバックアップルートディスクグループをアーカイブ化します。

ルートディスクのカプセル化とミラー化

VxVM では、ルートボリュームと起動に必要な他の領域を、別のディスク上にミラー化することができます。エラーが発生した root ディスクをいずれかのミラーに置き換えると、エラーからリカバリできます。

fdisk ディスクをカプセル化する前に、ルートディスクのパーティションテーブルをプリントアウトするには、sfdisk または root コマンドを使います。詳細については、該当するマニュアルページを参照してください。今後、元のルートディスクを再作成することが必要になった場合は、この情報が必要になります。

『Veritas InfoScale トラブルシューティングガイド』を参照してください。

p.1071 の「Linux でのルータビリティの使用に関する制限」を参照してください。

ルートディスクをカプセル化するには、vxdiskadm コマンドを使います。

p.1063 の「ディスクのカプセル化」を参照してください。

次の例 (ルートディスクは sda) のように、vxencap コマンドを使うこともできます。

```
# vxencap -c -g diskgroup rootdisk=sda
```

diskgroup には、現在のブートディスクグループの名前を指定します。その時点でブートディスクグループが存在しない場合は、指定した名前で作成されます。bootdg という名前はブートディスクグループ名のエイリアスとして予約されているため、使えません。変更を有効にするには、システムを再ブートする必要があります。

ルートディスクをカプセル化する vxdiskadm プロシージャと vxencap プロシージャでは、/etc/fstab ファイルとブートローダー設定ファイル (GRUB ではプラットフォームに応じて /boot/grub/menu.lst または /etc/grub.conf、LILO では /etc/lilo.conf) も更新します。

- rootvol、swapvol およびカプセル化されたルートディスク上のその他のボリュームにある /etc/fstab のエントリを変更します。
- ブートローダー設定ファイルに vxvm_root という特殊なエントリを追加すると、カプセル化されたルートディスクからシステムをブートできるようになります。

元の /etc/fstab とブートローダー設定ファイルの内容は、GRUB では /etc/fstab.b4vxvm、/boot/grub/menu.lst.b4vxvm、または /etc/grub.conf.b4vxvm、LILO では /etc/lilo.conf.b4vxvm というファイルに保存されます。

警告: /etc/fstab とブートローダー設定ファイルを変更するときは、VxVM によって追加されたエントリを壊さないように注意してください。システムが正常に起動されなくなることがあります。

カプセル化の後でルートディスクを別のディスク上にミラー化するには、次の手順を実行します。

1. ミラーに使用するために、既存の `root` ディスクサイズ以上のディスクを選択します。このディスクのジオメトリは、既存のルートディスクと同じジオメトリを持つと Linux で認識され、VxVM またはその他のサブシステム (マウント済みパーティションやスワップ領域など) で使っていないディスクです。ディスクは BIOS (Basic Input Output System) とオペレーティングシステムのブートローダーで表示できる必要があります。
2. `vxdiskadm` メインメニューで `Mirror Volumes on a Disk` を選択してルートディスクのミラーを作成します (これにより、ミラー化操作をルートディスク上で実行した場合は自動的に `vxrootmir` コマンドが呼び出されます)。

   すでに Volume Manager の制御下に置かれているディスクは、ルートミラーに使えません。

3 ルートディスク上のすべてのファイルシステムをミラー化する場合は、次のコマンドを実行します。

```
# vxrootmir mirror_da_namemirror_dm_name
```

mirror_da_name は、ルートディスクをミラー化するディスクのディスクアクセス名、*mirror_dm_name* は、ミラーディスクに割り当てるディスクメディア名です。プライマリルートディスクでエラーが発生した場合にシステムをブートできるように、代替 root ディスクを設定します。たとえば、ルートディスク sda をディスク sdb 上にミラー化して、このディスクに rootmir という名前を付けるには、次のコマンドを実行します。

```
# vxrootmir sdb rootmir
```

ルートディスクのミラーを設定する処理は、完了までに時間がかかることがあります。

ルートディスクをカプセル化してそのミラーを作成した後の vxprint コマンドの出力例を次に示します (簡潔にするため、TUTIL0 フィールド、PUTIL0 フィールド、サブディスクレコードは省略しています)。

```
Disk group: rootdg

TY NAME          ASSOC   KSTATE   LENGTH    PLOFFS   STATE ...
dg rootdg        rootdg  -        -         -        -

dm rootdisk      sda     -        16450497  -        -
dm rootmir       sdb     -        16450497  -        -

v  rootvol       root    ENABLED  12337857  -        ACTIVE
pl mirrootvol-01 rootvol ENABLED  12337857  -        ACTIVE
pl rootvol-01    rootvol ENABLED  12337857  -        ACTIVE

v  swapvol       swap    ENABLED  4112640   -        ACTIVE
pl mirswapvol-01 swapvol ENABLED  4112640   -        ACTIVE
pl swapvol-01    swapvol ENABLED  4112640   -        ACTIVE
```

ルートディスクをカプセル化する際の METADATA サブディスクの割り当て

METADATA サブディスクは、ルートディスクがカプセル化される際にパーティション情報を保護するために作成されます。ルートディスクのカプセル化が解除されると、このサブディスクは自動的に削除されます。

次の fdisk 出力の例は、システムのルートディスクのもとのパーティションテーブルを表します。

```
# fdisk -ul /dev/hda
Disk /dev/hda: 255 heads, 63 sectors, 2431 cylinders
Units = sectors of 1 * 512 bytes

   Device Boot    Start       End   Blocks    Id  System
/dev/hda1             63   2104514   1052226   83  Linux
/dev/hda2       2104515   6297479   2096482+  83  Linux
/dev/hda3       6329610  39054014  16362202+   5  Extended
/dev/hda5       6329673  10522574   2096451   83  Linux
/dev/hda6      10522638  14715539   2096451   83  Linux
/dev/hda7      14715603  18908504   2096451   83  Linux
/dev/hda8      18908568  23101469   2096451   83  Linux
/dev/hda9      23101533  25205984   1052226   82  Linux swap
```

拡張パーティション（hda3）の始まりと、最初の論理パーティション（hda5）の始まりの間にギャップがあります。論理パーティション（hda5 から hda9 まで）では、1 つの論理パーティションの終点と次の論理パーティションの始まりの間にギャップがあります。これらのギャップには、パーティション情報のメタデータが含まれています。これらのメタデータ領域はパブリックリージョンの中にあるため、VxVM は、メタデータ領域が誤ってボリュームに割り当てられるのを防ぐために、これらの領域にサブディスクを割り当てます。

ルートディスクがカプセル化された後で、vxprint コマンドから次のような内容が出力されます。

```
Disk group: rootdg

TY NAME            ASSOC       KSTATE   LENGTH   PLOFFS  STATE   TUTIL0  PUTIL0
dg rootdg          rootdg      -        -        -       -       -       -
dm disk01          sdh         -        17765181 -       -       -       -
dm rootdisk        hda         -        39053952 -       -       -       -

sd meta-rootdisk05 -           ENABLED  63       -       -       -       METADATA
sd meta-rootdisk06 -           ENABLED  63       -       -       -       METADATA
sd meta-rootdisk07 -           ENABLED  63       -       -       -       METADATA
sd meta-rootdisk08 -           ENABLED  63       -       -       -       METADATA
sd meta-rootdisk09 -           ENABLED  63       -       -       -       METADATA
sd meta-rootdisk10 -           ENABLED  63       -       -       -       METADATA
sd rootdiskPriv     -          ENABLED  2049     -       -       -       PRIVATE

v bootvol           fsgen      ENABLED  2104452  -       ACTIVE  -       -
pl bootvol-01       bootvol    ENABLED  2104452  -       ACTIVE  -       -
sd rootdisk-07      bootvol-01 ENABLED  2104452  0       -       -       -

v homevol           fsgen      ENABLED  4192902  -       ACTIVE  -       -
```

```
pl homevol-01       homevol    ENABLED  4192902  -       ACTIVE  -       -
sd rootdisk-05      homevol-01 ENABLED  4192902  0       -       -       -

v optvol            fsgen      ENABLED  4192902  -       ACTIVE  -       -
pl optvol-01        optvol     ENABLED  4192902  -       ACTIVE  -       -
sd rootdisk-04      optvol-01  ENABLED  4192902  0       -       -       -

v rootvol           root       ENABLED  4192902  -       ACTIVE  -       -
pl rootvol-01       rootvol    ENABLED  4192902  -       ACTIVE  -       -
sd rootdisk-02      rootvol-01 ENABLED  4192902  0       -       -       -

v swapvol           swap       ENABLED  2104452  -       ACTIVE  -       -
pl swapvol-01       swapvol    ENABLED  2104452  -       ACTIVE  -       -
sd rootdisk-01      swapvol-01 ENABLED  2104452  0       -       -       -

v usrvol            fsgen      ENABLED  4192965  -       ACTIVE  -       -
pl usrvol-01        usrvol     ENABLED  4192965  -       ACTIVE  -       -
sd rootdisk-06      usrvol-01  ENABLED  4192965  0       -       -       -

v varvol            fsgen      ENABLED  4192902  -       ACTIVE  -       -
pl varvol-01        varvol     ENABLED  4192902  -       ACTIVE  -       -
sd rootdisk-03      varvol-01  ENABLED  4192902  0       -       -       -
```

ルートディスクの新しいパーティションテーブルは次のようになります。

```
# fdisk -ul /dev/hda
Disk /dev/hda: 255 heads, 63 sectors, 2431 cylinders
Units = sectors of 1 * 512 bytes

    Device Boot    Start     End    Blocks  Id  System
/dev/hda1             63 2104514   1052226  83  Linux
/dev/hda2        2104515 6297479   2096482+ 83  Linux
/dev/hda3        6329610 39054014 16362202+  5  Extended
/dev/hda4             63 39054014 19526976  7e  Unknown
/dev/hda5        6329673 10522574 2096451   83  Linux
/dev/hda6       10522638 14715539 2096451   83  Linux
/dev/hda7       14715603 18908504 2096451   83  Linux
/dev/hda8       18908568 23101469 2096451   83  Linux
/dev/hda9       23101533 25205984 1052226   82  Linux swap
/dev/hda10      39051966 39054014 1024+     7f  Unknown
```

この例では、プライマリパーティション `hda4` と、論理パーティション `hda10` が作成され、それぞれが VxVM のパブリックリージョンとプライベートリージョンを表しています。

ルートカプセル化システムでのカーネルのアップグレード

OS ベンダーは、セキュリティの問題や他の小さな製品不具合を解決するために製品の保守パッチを頻繁にリリースします。OS ベンダーは、保守契約に準拠するようにまたは適切なベンダーサポートを受けられるように、これらのパッチを定期的に適用することをユーザーに要求することがあります。このリリースよりも前のリリースでは、ルートカプセル化システムでカーネルのパッチをインストールしたりカーネルをアップグレードしたりできませんでした。これを行うには、システムのカプセル化を解除し、アップグレードを適用して、ルートディスクを再カプセル化する必要がありました。現在は、ルートカプセル化システムで OS カーネルをアップグレードできます。

メモ: このセクションの手順はマイナーなカーネルアップグレードまたはパッチにのみ適用されます。これらの手順は Linux オペレーティングシステムの完全なアップグレードには適用されません。

ルートカプセル化システムで OS カーネルをアップグレードするには

1. マイナーなアップグレードまたはパッチをシステムに適用します。
2. アップグレードを適用した後、次のコマンドを実行します。

   ```
   # . /etc/vx/modinst-vxvm

   # upgrade_encapped_root
   ```

   上記のコマンドはカーネルのアップグレードをカプセル化されたシステムに適用できるかどうかを判断します。アップグレードが成功すると、コマンドは次のメッセージを表示します。

   ```
   # upgrade_encapped_root
   The VxVM root encapsulation upgrade has succeeded.
   Please reboot the machine to load the new kernel.
   ```

   次の再起動後に、システムはパッチが適用されたカーネルとカプセル化された VxVM ルートボリュームを使って再起動されます。

一部のパッチは VxVM のインストールされたバージョンと完全に互換性がない場合があります。この場合、スクリプトは失敗して、次のメッセージが表示されます。

```
# upgrade_encapped_root
FATAL ERROR: Unencapsulate the root disk manually.
VxVM cannot re-encapsulate the upgraded system.
```

アップグレードスクリプトは、システムを以前の設定でブートするのに使うことができるシステム設定ファイル保存しています。アップグレードが失敗したら、次の手順に従って以前の設定をリストアします。

メモ: 厳密な手順はオペレーティングシステムによって変わることがあります。

以前の設定をリストアするには

1 ブートストラップ時にスペースキーを押して、GRuB ブートローダーを割り込みます。

インストールされた各種カーネルバージョンと VxVM ルートのカプセル化バージョンが示された一連の潜在的なブート設定が表示されます。

次に例を示します。

```
Red Hat Enterprise Linux Server (2.6.18-53.el5)
Red Hat Enterprise Linux Server (2.6.18-8.el5)
vxvm_root_backup
vxvm_root
```

2 カプセル化された VxVM ルートディスクを使って以前のカーネルバージョンをブートするように `vxvm_root_backup` オプションを選択します。

ルートカプセル化システムで OS カーネルを手動でアップグレードするには

1 アップグレードスクリプトが失敗した場合、ルートディスクをブートできるようにルートディスクのカプセル化を手動で解除できます。

p.1090 の「ルートディスクのカプセル化の解除」を参照してください。

2 カーネルをアップグレードし、システムを再ブートします。

3 再起動が成功すれば、ルートディスクを再びカプセル化およびミラー化できます。

p.1082 の「ルートディスクのカプセル化とミラー化」を参照してください。

ただし、次の再起動後に VxVM が正しく動作できない場合があり、すべての VxVM のボリュームが利用不能になる場合があります。VxVM ボリュームをリストアするには、次のとおりにカーネルのアップグレードを削除する必要があります。

```
# rpm -e upgrade_kernel_package_name
```

次に例を示します。

```
# rpm -e kernel-2.6.18-53.el5
```

カプセル化されたブートディスクの管理

`vxrootadm`コマンドはカプセル化されたブートディスクのブート可能なスナップショットを作ることを可能にします。

`vxrootadm` コマンドの形式は次のとおりです。

```
# vxrootadm [-v] [-g dg] [-s srcdisk] ... keyword arg ...
```

mksnapキーワードは次の形式がなければなりません:

```
# vxrootadm -s srcdisk mksnap destdisk newdg
```

vxrootadm には次のオプションがあります。

`vxrootadm [-v ] [-D]`	これらは詳細出力とデバッグのメッセージオプションで、省略可能です。
`vxrootadm [-g dg]`	ディスクグループの引数は省略可能です。
`vxrootadm -s srcdisk`	ソースディスクを指定します。

vxrootadm(1M)のマニュアルページを参照してください。

カプセル化されたブートディスクのスナップショットの作成

カプセル化されたブートディスクのスナップショットを作成するには、vxrootadm コマンドを使います。スナップショットのターゲットディスクはソースディスク(ブートディスク)以上のサイズである必要があります。新しいディスクグループ名を使ってターゲットディスクを関連付ける必要があります。

カプセル化されたブートディスクのスナップショットを作成するには

- 次のコマンドを入力します。

  ```
  # vxrootadm -s srcdisk [-g dg] mksnap destdisk newdg
  ```

  次に例を示します。

  ```
  # vxrootadm -s disk_0 -g rootdg mksnap disk_1 snapdg
  ```

  この例では、disk_0 はカプセル化されたブートディスク、rootdg は関連付けられているブートディスクグループ、disk_1 はターゲットディスク、snapdg は新しいディスクグループ名です。

ルートディスクのカプセル化の解除

vxunroot ユーティリティを使って、システムからルータビリティサポートを削除できます。これにより、ルートディスク上の root、swap、home および他のファイルシステムに、ボリュームデバイスを通じてではなく、ディスクパーティションを通じて直接アクセスできるようになります。

また、vxunroot ユーティリティを使って必要な設定変更を行うと、システムを VxVM にまったく依存せずに起動できるようになります。

vxunroot でカプセル化を解除できるのは、カプセル化の時点でルートディスク上に存在していたボリュームのみです。カプセル化後にルートディスク上に作成したボリュームは、vxunroot を実行する前にすべて退避してください。

元のディスクパーティションに対応しているルートディスク上のプレックスを削除しないでください。

警告: この手順を完結するには、システムを再ブートする必要があります。

システムから、ルータビリティを削除するには、次の手順を実行します。

1 vxplex コマンドを使って、ルートディスク以外のディスク上の、rootvol、swapvol、usr、var、opt、home の各ボリュームのプレックスをすべて削除します。

たとえば、次のコマンドは、ディスク rootmir 上に設定された mirrootvol-01 および mirswapvol-01 の各プレックスを削除します。

```
# vxplex -g bootdg -o rm dis mirrootvol-01 mirswapvol-01
```

2 vxunroot ユーティリティを実行します。

```
# vxunroot
```

他のディスク上にプレックスが残っている場合、vxunroot はディスクパーティションへの変換を実行しません。

ルートディスクがカプセル化されてからデバイスの名前の付け方が変更された場合、vxunroot コマンドは次のエラーで失敗します。

```
VxVM vxunroot ERROR V-5-2-4101 The root disk name does not match
the name of the original disk that was encapsulated.
```

このメッセージが表示された場合は、vxddladm assign names コマンドを使い、カプセル化されたルートディスクの永続的なデバイス名を再生成してから、vxunroot コマンドを再試行します。

p.385 の「永続的なデバイス名の再生成」を参照してください。

43

クォータ

この章では以下の項目について説明しています。

- Veritas File System のクォータ制限について
- Veritas File System のクォータファイルについて
- Veritas File System のクォータコマンドについて
- Veritas File System によるクォータのチェックについて
- Veritas File System クォータの使用

Veritas File System のクォータ制限について

VxFS（Veritas File System）では、ユーザークォータおよびグループクォータがサポートされています。クォータシステムは、ファイルシステムの 2 つの主要リソースであるファイルとデータブロックの使用を制限します。リソースごとに各ユーザーおよびグループにクォータを割り当て、その使用を制限できます。

この 2 つのリソースについては、次の制限が用意されています。

ハード制限	いかなる条件下においても超過できない絶対的な制限です。
ソフト制限	ハード制限の範囲内で設定します。一定の期間内であれば、制限を超えることができます。期間はファイルシステム単位でのみ設定できます。VxFS のデフォルトの期間は 7 日間です。

ソフト制限は、通常、ユーザーが大容量の一時ファイルを生成するアプリケーションを実行する必要がある場合に使います。この場合、ユーザーは一定の期間内であればクォータ限度を超えることができます。期間をすぎると、割り当てを実行できなくなります。

`vxedquota` コマンドを使って、制限を設定します。

p.1095 の「Veritas File System クォータの使用」を参照してください。

ファイルとデータブロックの制限はユーザーおよびグループごとに設定できますが、期間はファイルシステム全体に適用されます。クォータ限度情報はユーザー ID とグループ ID に関連付けられており、ユーザーまたはグループのクォータファイルに保存されます。

p.1093 の「Veritas File System のクォータファイルについて」を参照してください。

VxFS でファイルに領域が事前に割り当てられている場合は、クォータのソフト制限を超えることができます。

p.283 の「エクステント属性について」を参照してください。

Veritas File System のクォータファイルについて

どのクォータコマンドも実行できるように、クォータファイル(ファイル名は quotas)は、ファイルシステムのルートディレクトリに配置する必要があります。グループクォータが機能するには、quotas.grp または quotas.grp.64 ファイルが必要です。ファイルシステムのマウントポイント内のファイルは、外部クォータファイルと呼ばれます。VxFS には、VxFS 自体が使うための内部クォータファイルも用意されています。

クォータ管理コマンドは、外部クォータファイルに対して読み取りと書き込みを実行し、使用制限を取得または変更します。VxFS では、内部ファイルを使って、各ユーザーが使うデータブロックと i ノードの使用回数を保持します。クォータが有効になっている場合、クォータ限度は外部クォータファイルから内部クォータファイルにコピーされます。クォータが有効な間は、クォータの変更や使用情報の変更はすべて内部クォータファイルに登録されます。クォータを無効にすると、内部クォータファイルの内容が外部クォータファイルにコピーされるため、2 つのファイルのデータはすべて同期化されます。

VxFS では、ユーザークォータの他にグループクォータもサポートされています。ユーザークォータがユーザー別にファイルシステムリソース(ディスクブロックおよび i ノード数)の使用を制限するように、グループクォータはグループ別にリソースの使用を制限します。ユーザークォータと同様に、グループクォータはファイルシステムリソースにソフト制限とハード制限を設定します。ユーザークォータとグループクォータの両方が有効な場合、実行ユーザーのリソースの使用は、この 2 つのうち制限が厳しい方の値に基づいて制限されます。

グループクォータとユーザークォータを区別するには、VxFS クォータコマンドで -g と -u オプションを使います。どちらのオプションも指定されていない場合、デフォルトでユーザークォータが使われます。-o quota オプションを mount コマンドのオプションとして指定している場合、このルールは適用されません。その場合、ユーザークォータとグループクォータの両方が有効になります。グループクォータのサポートでは、個別にグループクォータファイルも必要になります。VxFS グループクォータファイルは、quotas.grp または quotas.grp.64 という名前です。VxFS ユーザークォータファイルの名前は、quotas または quotas.64 です。quotas という名前が使われるのは、オペレーティングシステムの別のファイルシステムで使われる quotas.user ファイルと区別するためです。

Veritas File System のクォータコマンドについて

メモ: quotacheckコマンドは例外です。VxFS では、このようなコマンドはサポートされていません。

p.1095 の「Veritas File System によるクォータのチェックについて」を参照してください。

さまざまなファイルシステムでクォータをサポートするには、Linux に用意されている汎用コードを使います。ただし、VxFS は、この汎用インターフェースを使いません。VxFS では、この代わりに、VxFS ファイルシステムでのみ動作する、類似したコマンドセットが提供されています。

VxFS では、次のクォータコマンドが提供されています。

vxedquota	ユーザーやグループのクォータ限度を変更します。vxedquota を使って変更された制限は、内部クォータファイルと外部クォータファイルの両方に反映されます。
vxrepquota	クォータとディスクの使用率の情報を取得します。
vxquot	ファイルの所有権と使用率の情報を取得します。
vxquota	クォータ限度と使用率を表示します。
vxquotaon	マウントされている VxFS ファイルシステムのクォータを有効にします。
vxquotaoff	マウントされている VxFS ファイルシステムのクォータを無効にします。

vxquota、vxrepquota、vxquot、vxedquota コマンドは、ヒューマンフレンドリな入力と出力に使用する -H オプションをサポートしています。-H オプションを使用すると、ストレージサイズは、次のヒューマンフレンドリな単位で表示されます。bytes(B)、kilobytes(KB)、megabytes(MB)、gigabytes(GB)、terabyte(TB)、petabytes(PB)、exabytes(EB)。クォータのソフト制限値とハード制限値、クォータの使用状況、特定のユーザーや特定のグループ、またはすべてのユーザーやすべてのグループが消費する総ストレージは、-H オプションを使用して、ヒューマンフレンドリな単位で取得できます。

これらのコマンドに加えて、VxFS mount コマンドは、特殊なマウントオプション(-o quota|usrquota|grpquota)をサポートしています。このオプションは、マウント時にクォータを有効にする場合に使います。また、再マウント中の VxFS ファイルシステムか、マウントされているファイルシステムで、ユーザーまたはグループクォータを有効にするか、無効にするかを選択できます。

クォータコマンドについて詳しくは、vxedquota(1M)、vxrepquota(1M)、vxquot(1M)、vxquota(1M)、vxquotaon(1M)、およびvxquotaoff(1M)のマニュアルページを参照してください。

メモ: VxFS ファイルシステムが NFS を介してエクスポートされる場合、NFS クライアントで VxFS クォータコマンドを使い、クォータへの問い合わせやクォータの編集を行うことはできません。サーバーで VxFS クォータコマンドを使うと、クォータに対する問い合わせや編集を行えます。

Veritas File System によるクォータのチェックについて

クォータの標準的な実装では、全ファイルシステムのマウント後にクォータが個別に検査されます。クォータ検査では、ディスクの i ノードを読み取り、ユーザー別やグループ別に使用量を算出します。この方法は時間がかかります。また、ファイルシステムがマウントされているため、クォータ検査の実行中に使用量が変化することがあります。

VxFS では、quotacheck コマンドはサポートされていません。VxFS では、クォータを有効にした時点で、クォータ検査が自動的に実行されます(必要な場合)。内部クォータファイルに記録された使用量情報に反映されるファイルシステムに変更があった場合は、クォータ検査が必要になります。これは、クォータを無効にした状態でファイルシステムへの書き込みがあった場合、またはファイルシステムにファイルシステム全体の検証が必要となるような構造上の損傷が発生した場合に限られます。

fsck_vxfs(1M)マニュアルページを参照してください。

クォータ検査は、通常、ディスクの i ノードの情報を読み取り、内部クォータファイルを再構築します。クォータが有効でない場合でも、システム管理者によってクォータ限度が変更されることがあります。このような変更は外部クォータファイルに保存されます。クォータを有効にする処理の一部として、クォータ限度は外部クォータファイルから内部クォータファイルに読み取られます。

Veritas File System クォータの使用

Veritas File System(VxFS)クォータコマンドを使って、以下のクォータ機能を実行します。

- 「Veritas File System クォータの有効化」
- 「マウント時に Veritas File System のクォータを有効にする」
- 「Veritas File System クォータの編集」
- 「Veritas File System のクォータの時間制限の修正」
- 「Veritas File System のディスククォータと使用率の表示」
- 「ユーザーまたはグループが所有するブロックの表示」
- 「Veritas File Systemのクォータを無効にする」
- 「64 ビットのクォータのサポート」

Veritas File System クォータの有効化

ファイルシステムでクォータ機能を使うには、クォータを有効にする必要があります。ファイルシステムのマウント時またはマウント後にクォータを有効にできます。

メモ: クォータを有効にする前に、root によって所有される quotas という名前のユーザークォータファイルと quotas.grp という名前のグループクォータファイルをファイルシステムのルートディレクトリに作成する必要があります。

64 ビットのクォータは DLV 10 以上でサポートされます。クォータファイル名は quotas.64 および quotas.grp.64 です。

クォータを有効にするには

1. VxFS のユーザーとグループクォータを有効にするには、次のコマンドを入力します。

```
# vxquotaon /mount_point
```

2. VxFS ファイルシステムのユーザークォータのみを有効にするには、次のコマンドを入力します。

```
# vxquotaon -u /mount_point
```

3. VxFS ファイルシステムのグループクォータのみを有効にするには、次のコマンドを入力します。

```
# vxquotaon -g /mount_point
```

マウント時に Veritas File System のクォータを有効にする

ファイルシステムのマウント時に mount コマンドを使ってクォータを有効にできます。

マウント時にクォータを有効にするには

1 マウント時にファイルシステムのユーザークォータまたはグループクォータを有効にするには、次のように入力します。

```
# mount -t vxfs -o quota special/mount_point
```

ここで、*special* は aVxFS ブロックの特殊デバイスです。

2 ユーザークォータのみを有効にするには、次のコマンドを入力します。

```
# mount -t vxfs -o usrquota special/mount_point
```

3 グループクォータのみを有効にするには、次のコマンドを入力します。

```
# mount -t vxfs -o grpquota special/mount_point
```

メモ:「*special*」は VxFS のブロックスペシャルのデバイスを示します。

Veritas File System クォータの編集

vxedquota コマンドを使って、ユーザークォータとグループクォータを設定できます。クォータを編集するには、スーパーユーザー権限が必要です。

vxedquota は指定されたユーザーに対する一時ファイルを作成します。このファイルに、クォータファイルを持つ、マウントされた各ファイルシステムのディスククォータを保存します。vxedquota を実行するためにクォータを有効にする必要はありません。ただし、ファイルシステムのクォータを有効にしないかぎりクォータ限度は適用されません。

クォータを編集するには

1 -u オプションを指定して、*username* で指定した 1 つ以上のユーザークォータを編集できます。

```
# vxedquota [-u] username
```

-u オプションが指定されていない場合は、デフォルトの状態で 1 つ以上のユーザークォータを編集できます。

2 -g オプションを指定して、*groupname* で指定した 1 つ以上のグループクォータを編集できます。

```
# vxedquota -g groupname
```

Veritas File System のクォータの時間制限の修正

ソフト制限とハード制限では、vxedquota コマンドを使って値を変更したり、割り当てることができます。どのユーザーまたはグループも、クォータが有効になっている場合、ハード制限を超えることはできません。

期限の修正は、ファイルシステム全体に適用されます。ユーザー単位やグループ単位で適用することはできません。

期限を修正するには

1. -t オプションを指定して、どのユーザーの期限も修正できます。

```
# vxedquota [-u] -t
```

2. -g オプションや -t オプションを指定して、どのグループの期限も修正できます。

```
# vxedquota -g -t
```

Veritas File System のディスククォータと使用率の表示

vxquota コマンドを使って、VxFS ファイルシステムでユーザーまたはグループのディスククォータと使用率を表示します。

ディスククォータと使用率を表示するには

1. マウントされたすべての VxFS に quotas ファイルが存在する場合に、ユーザーのクォータとディスクの使用率を表示するには、次のように入力します。

```
# vxquota -v [-u] username
```

2. マウントされたすべての VxFS に quotas.grp ファイルが存在する場合に、グループのクォータとディスクの使用率を表示するには、次のように入力します。

```
# vxquota -v -g groupname
```

ユーザーまたはグループが所有するブロックの表示

vxquot コマンドを使って、ファイルシステムで各ユーザーまたはグループが所有するブロック数を表示します。

ユーザーまたはグループが所有するブロック数を表示するには

1 各ユーザーが所有するファイル数と領域を表示するには、次のように入力します。

```
# vxquot [-u] -f filesystem
```

2 各グループが所有するファイル数と領域を表示するには、次のように入力します。

```
# vxquot -g -f filesystem
```

Veritas File Systemのクォータを無効にする

vxquotaoff コマンドを使って、クォータを無効にします。

クォータを無効にするには

1 マウントされたファイルシステムのクォータを無効にするには、次のコマンドを入力します。

```
# vxquotaoff /mount_point
```

2 VxFS のユーザークォータのみを無効にするには、次のコマンドを入力します。

```
# vxquotaoff -u /mount_point
```

3 VxFS のグループクォータのみを無効にするには、次のコマンドを入力します。

```
# vxquotaoff -g /mount_point
```

64 ビットのクォータのサポート

64 ビットのクォータは DLV 10 以上でサポートされます。この機能を使うには、vxupgrade コマンドを使って、既存のファイルシステムを DLV 10 以上にアップグレードする必要があります。

vxupgrade（1M）のマニュアルページを参照してください。

32 ビットのクォータは、ディスクレイアウトバージョン 9 またはそれ以前で引き続きサポートされます。同じクォータコマンドを、32 ビットと 64 ビットの両方のクォータに使うことができます。

64 ビットのクォータの場合は、2 つの新しいクォータファイルがあります。グループのクォータのファイル名は quotas.grp.64 で、ユーザーのクォータのファイル名は quotas.64 です。これらのファイルは、ディスクレイアウトバージョンのアップグレードが完了した後、各ファイルシステムに作成されます。

44

FCL (File Change Log)

この章では以下の項目について説明しています。

Veritas File System ファイルの変更ログについて

VxFS FCL（File Change Log）は、ファイルシステム内のファイルおよびディレクトリへの変更を記録します。

通常、FCL を使うアプリケーションで次の処理を行う必要があります。

- ファイルシステムまたはサブセット全体のスキャン
- 前回のスキャン以降に変更された箇所の検出

これに該当するアプリケーションには、バックアップユーティリティ、Web クローラ、検索エンジン、複製プログラムが含まれることがあります。

メモ: FCL は、データが変更された時刻を追跡して変更のタイプを記録しますが、実際のデータの変更内容は追跡しません。データが変更されたファイルを検査して、変更されたデータを特定するのはアプリケーションです。

FCL 機能は別ライセンスです。

Veritas File System ファイルの変更ログファイルについて

FCL は、作成、リンク、リンクの解除、名前の変更、データの追加、データの上書き、データの切り捨て、拡張属性の変更、ホールのパンチ、さまざまなファイルプロパティの更新など、ファイルシステムへの変更を記録します。

FCL は、ファイルシステムの名前空間にあるスパースファイルに変更を保存します。FCL ファイルは、`mount_point/lost+found/changelog` に保存されます。FCL ファイルは通常ファイルと同様に操作できますが、実行できない操作もあります。標準のシステムコール `open`(2)、`lseek`(2)、`read`(2)、`close`(2)は FCL のデータにアクセスできますが、`write`(2)、`mmap`(2)、`rename`(2)はアクセスできません。

警告: 標準システムコールは、現在その一部がすでにサポートされていますが、今後の VxFS リリースでは、FCL ファイルは名前空間から取り出されるようになり、これらのシステムコールは動作しなくなります。したがって、新しく開発するアプリケーションはすべてプログラミングインターフェースを使うことを推奨します。

FCL のログファイルには、FCL に関する情報(FCL スーパーブロックに保存される)と、ファイルシステム内のファイルおよびディレクトリへの変更(FCL レコードとして保存される)の両方が格納されます。

p.1104 の「Veritas File System ファイルの変更ログのプログラミングインターフェース」を参照してください。

4.1 リリースでは、FCL の構造は、`/opt/VRTS/include/sys/fs/fcl.h` ヘッダーファイルにより開示されていました。本リリースでは、FCL ファイルの内部構造を公開していません。FCL にアクセスするしくみとしては、`/opt/VRTSfssdk/7.0.0.000/include/vxfsutil.h` ヘッダーファイルに記述されている API の使用が推奨されます。

FCL 4.1 ヘッダーファイルにアクセスするアプリケーションの動作が中断しないように、このリリースには、`/opt/VRTS/include/sys/fs/fcl.h` ヘッダーファイルが含まれています。新しいアプリケーションでは、`/opt/VRTSfssdk/7.0.0.000/include/vxfsutil.h` に記述されている新規 FCL API を使う必要があります。既存のアプリケーションでも、新しい FCL API が使えるように修正してください。

既存のアプリケーションへの後方互換に対応するために、このリリースでは複数の FCL バージョンがサポートされます。ユーザーは、新しい FCL の FCL バージョンを柔軟に指定できます。デフォルトの FCL バージョンは 4 です。

`fcladm`(1M)のマニュアルページを参照してください。

Veritas File System ファイルの変更ログの管理インターフェース

FCL(File Change Log)は、VxFS の管理コマンド fcladm(1M)と vxtunefs(1M)を使って設定およびチューニングできます。

fcladm(1M)および vxtunefs(1M)の各マニュアルページを参照してください。

fcladm の FCL 引数は次のとおりです。

clear	このパラメータ設定後に、監査、オープン、クローズ、統計イベントの記録は無効になります。
dump	オフホストプロセッシングシステムにダウンロードできる FCL ファイルの正規のファイルイメージを作成します。作成されるファイルの形式は、FCL ファイルのものとは異なっています。
on	マウントされているファイルシステムで FCL を有効にします。VxFS 5.0 以降のリリースでは、FCL バージョンの 3 または 4 のいずれかをサポートしています。バージョンを指定しなかった場合は、バージョン 4 がデフォルトで設定されます。バージョンは、fcladm on を使って指定できます。
print	指定したオフセットから始まる FCL ファイルの内容を出力します。
restore	FCL の正規のファイルイメージから FCL ファイルをリストアします。ファイルは dump 引数を指定して作成されています。
rm	FCL ファイルを削除します。FCL ファイルを削除する前に、まず off 引数を指定して FCL を無効にする必要があります。
set	「eventlist」オプションの指定により、イベントの記録を有効にします。 fcladm(1M)のマニュアルページを参照してください。
state	標準出力に対して FCL の現在の状態を書き込みます。
sync	FCL の記録間隔に関連するデータをフラッシュして FCL を静的な状態へと移行させます。

vxtunefs の FCL チューニングパラメータは次のとおりです。

fcl_keeptime
: FCL レコードが、パージされる前に、FCL ファイル内に留まる時間を秒単位で指定します。パージされる場合、最も古いレコードからパージされます。最も古いレコードはファイルの先頭にあります。さらに、FCL ファイルへの割り当てが fcl_maxalloc バイトを越えた場合に、ファイルの最初にあるレコードがパージされる可能性があります。
fcl_keeptime のデフォルト値は、0 です。fcl_maxalloc パラメータが設定されている場合、レコードは、FCL に割り当てられている領域が fcl_maxalloc を超えているとFCL ファイルからパージされます。これは、たとえレコードがログに記録されている経過時間が、fcl_keeptime の値を下回っていたとしても変わることはありません。

fcl_maxalloc
: FCL に割り当てる領域の最大値をバイト単位で指定します。割り当てられた領域が fcl_maxalloc を超えた場合、ファイルの先頭にホールがパンチされます。この結果、レコードがパージされ、最初の有効なオフセット(fc_foff)が更新されます。さらに、最も古いレコードの有効期限が fcl_keeptime に達していない場合、fcl_maxalloc は無視されることがあります。

  fcl_maxalloc の最小値は 4 MB です。デフォルト値は fs_size/33 です。

fcl_winterval
: FCL が上書き、拡張書き込み、または切り捨てを記録する間隔を秒単位で指定します。これにより、FCL ファイル内の繰り返しレコード数を減らすことができます。fcl_winterval タイムアウトは i ノードごとです。i ノードがキャッシュから削除され、再度キャッシュに復帰する場合、その書き込み間隔はリセットされます。結果として、同じ書き込み間隔でファイルに複数のレコードを書き込めることになります。デフォルト値は 3600 秒です。

fcl_ointerval
: 秒単位の間隔を指定します。この間は、後続のファイルオープンで新しい FCL レコードは生成されません。これにより、FCL ファイルに記録される繰り返しレコードの数を減らすことができます。アクセス情報の追跡も有効になっている場合、別のユーザーにオープンされているファイルは、fcl_ointerval の間でも後続のファイルオープンでレコードが生成されることがあります。同様に i ノードがキャッシュから取り消された場合、同じオープン間隔内で 2 つ以上のレコードが生成される場合があります。

  デフォルト値は 600 秒です。

FCL をアクティブにするには、fcl_maxalloc および fcl_keeptime の一方または両方を設定する必要があります。次に、fcladm コマンドの使用例を示します。

マウントしたファイルシステムの FCL を有効にするには、次のコマンドを入力します。

```
# fcladm on mount_point
```

マウントしたファイルシステムの FCL を無効にするには、次のコマンドを入力します。

```
# fcladm off mount_point
```

FCL が無効となっている、マウントされたファイルシステムの FCL ファイルを削除するには、次を入力します。

```
# fcladm rm mount_point
```

マウントされたファイルシステムの現在の FCL 状態を取得するには、次のコマンドを入力します。

```
# fcladm state mount_point
```

FCL における各イベントによりアクセス情報とともにファイルのオープン状態を追跡できるようにするには、次のコマンドを入力します。

```
# fcladm set fileopen,accessinfo mount_point
```

FCL の I/O 統計データの追跡を停止するには、次のコマンドを入力します。

```
# fcladm clear filestats mount_point
```

FCL に関する情報を取得するために、オフセット 0 を使ってオンディスク FCL スーパーブロックをテキスト形式で出力できます。これは、オンディスク FCL スーパーブロックが FCL ファイルの最初のブロックを占有しているため、FCL スーパーブロックを読み取り、fc_foff フィールドをチェックすることで、FCL ファイル内の最初と最後の有効なオフセット値を特定できます。次を入力します。

```
# fcladm print 0 mount_point
```

オフセットで指定される FCL の内容をテキスト形式で出力するには、次のコマンドを入力します（使うオフセットは 32 バイトで整列されています）。

```
# fcladm print offset mount_point
```

Veritas File System ファイルの変更ログのプログラミングインターフェース

VxFS には、次の 2 つの方法で FCL ファイルの読み取りと解析を単純化するための拡張 API が用意されています。

読み取りの単純化	PI により FCL エントリ解析に必要な追加のコードが削減され、ユーザータスクが単純化されます。4.1 では、削除やリンクなどのイベント情報を取得するために、ユーザーは削除やリンクされたファイル名を取得するための追加のコードを書く必要がありました。このリリースでは、ユーザーは API を使ってアセンブルされたレコードを直接読み取ることができます。また、API を使って、目的のイベントレコードのサブセットを示すためのフィルタを指定できます。
後方互換	FCL への API アクセス機能により、アプリケーションの後方互換に対応します。API を使うことで、アプリケーションは FCL のレイアウト変更に関係なく FCL ファイルを解析できます。FCL の隠されたレコードレイアウトが変更されても、API は戻りデータを自動的に変換して求められる出力レコードに一致させます。その結果、ユーザーは、オンディスクの FCL レイアウトの変更により、アプリケーションを修正または再コンパイルする必要はありません。

次のサンプルコードは、FCL スーパーブロックを読み取り、FCL の状態が VX_FCLS_ON であることを確認します。次に、vxfs_fcl_sync への呼び出しを発行して、読み取り先の最後のオフセットを取得し、FCL ファイル内の最初の有効なオフセットを特定します。最後に、このオフセットから 8 K のチャンクでエントリを読み取ります。FCL エントリ処理（process fcl entries）とコメントされた行付近に、FCL 内のエントリに対する処理レコードをアプリケーション開発者が提供する必要があります。

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/fcntl.h>
#include <errno.h>
#include <fcl.h>
#include <vxfsutil.h>
#define FCL_READSZ 8192
char* fclname = "/mnt/lost+found/changelog";
int read_fcl(fclname) char* fclname;
{
    struct fcl_sb fclsb;
    uint64_t off, lastoff;
    size_t size;
    char buf[FCL_READSZ], *bufp = buf;
    int fd;
    int err = 0;
    if ((fd = open(fclname, O_RDONLY)) < 0) {
        return ENOENT;
    }
```

```
    if ((off = lseek(fd, 0, SEEK_SET)) != 0) {
        close(fd);
        return EIO;
    }
    size = read(fd, &fclsb, sizeof (struct fcl_sb));
    if (size < 0) {
        close(fd);
        return EIO;
    }
    if (fclsb.fc_state == VX_FCLS_OFF) {
        close(fd);
        return 0;
    }
    if (err = vxfs_fcl_sync(fclname, &lastoff)) {
        close(fd);
        return err;
    }
    if ((off = lseek(fd, off_t, uint64_t)) != uint64_t) {
        close(fd);
        return EIO;
    }
    while (off < lastoff) {
        if ((size = read(fd, bufp, FCL_READSZ)) <= 0) {
            close(fd);
            return errno;
        }
        /* process fcl entries */
        off += size;
    }
    close(fd);
    return 0;
}
```

Veritas File System FCL API 機能の概略

ここでは、FCL API 関数を簡単にまとめておきます。

`vxfs_fcl_close()` FCL ファイルを閉じて、ハンドルに関連したリソースを削除します。

`vxfs_fcl_cookie()` 現在の FCL のアクティブ化日時とオフセットを埋め込んだ非公開構造を返します。この cookie を保存しておき、後でアプリケーションが最後に停止した箇所から読み取りを続行するために `vxfs_fcl_seek()` 関数に渡すこともできます。

vxfs_fcl_getinfo()	FCL ファイルの状態やバージョンなどの情報を返します。
vxfs_fcl_open()	FCL ファイルを開き、後続の操作で使えるハンドルを返します。
vxfs_fcl_read()	対象の FCL レコードをユーザー指定のバッファに読み込みます。
vxfs_fcl_seek()	指定の cookie からデータを抽出し、指定のオフセットにシークします。
vxfs_fcl_seektime()	指定時間の経過後に FCL の最初のレコードにシークします。

11

参照先

- 付録 A. パス名の逆引きルックアップ
- 付録 B. チューニング可能なパラメータ
- 付録 C. コマンドリファレンス
- 付録 D. スタータデータベースの作成

A

パス名の逆引きルックアップ

この付録では以下の項目について説明しています。

パス名の逆引きルックアップについて

パス名の逆引きルックアップ機能を使うと、ファイルやディレクトリの i ノード番号から、完全パス名を取得できます。i ノード番号は、管理コマンド `vxlsino` またはアプリケーションプログラミングインターフェースのライブラリ関数 `vxfs_inotopath_gen`(3)の引数として指定します。

パス名の逆引きルックアップ機能は、VxFS FCL 機能のクライアント、バックアップユーティリティ、リストアユーティリティ、レプリケーション製品など、さまざまなアプリケーションで使えます。ファイルやディレクトリのパス名は非常に長くなることがあるため、これらのアプリケーションでは、情報を i ノード番号で保存することが普通です。そのため、パス名を簡単に取得する手段が必要になります。

i ノード番号は、ファイルシステム内の各ファイルに付けられた一意の識別番号です。i ノードには、そのファイルに関連付けられたデータとメタデータが格納されます。ただし、その i ノードに対応するファイル名は含まれていません。したがって、i ノード番号からファイルの名前を特定することは、比較的難しくなります。`ncheck` コマンドを実行すると、ファイルシステム内の各ディレクトリがスキャンされ、i ノードの識別子からファイル名が取得されます。ただし、このプロセスには、時間がかかることがあります。VxFS のパス名の逆引きルックアップ機能を使うと、パス名を比較的すばやく取得できます。

メモ: シンボリックリンクにはファイルのパスが含まれていないため、パス名の逆引きルックアップ機能を使っても、ファイルへのシンボリックリンクを追跡することはできません。

名前の変更、リンクの解除および新規ファイルの作成においてエラーが起きた可能性があるので、パス名を指定して `lookup` または `open` を実行し、i ノード番号と取得したパス名が一致するかどうかを確認することをお勧めします。

vxlsino(1M)、vxfs_inotopath_gen(3)、vxfs_inotopath(3)のマニュアルページを参照してください。

B

チューニング可能なパラメータ

この付録では以下の項目について説明しています。

Storage Foundation Cluster File System High Availability のチューニングについて

Storage Foundation Cluster File System High Availability（SFCFSHA）は、パフォーマンスが重要な役割を持つさまざまな環境で幅広く使用されています。SFCFSHA には、SFCFSHA が使用される特定の環境および作業負荷に合わせてスタックのカスタマイズを可能にする、いくつかのチューニングパラメータと設定オプションがあります。このマニュアルは、これらのオプションのいくつかがどのようにパフォーマンスに影響を与えるかを理

解するのに役立ちます。また、それらのオプションのチューニングに関するガイドラインも提供します。

このマニュアルでは、SFCFSHA にとって重要なチューニングパラメータについて詳しく説明します。作業負荷に対して良好なパフォーマンスを得るには、オペレーティングシステム、データベース、ストレージネットワーク、ディスクアレイなど、ほかのスタックのエンティティも調整する必要がある場合があります。このマニュアルではこれらのうちのいくつかのパラメータを簡単に説明しますが、これらのエンティティの調整方法について詳しくは、そのエンティティのベンダーのマニュアルを参照してください。

警告: ベリタスでは、VCS カーネルのチューニングパラメータを変更する場合は、ベリタスのサポートに問い合わせることをお勧めしています。いくつかのチューニングパラメータは、重要なデータ構造のためのメモリを事前に割り当てており、それらの値を変更すると、メモリの使用量が増えたり、パフォーマンスが低下したりします。

VxFS ファイルシステムのチューニング

この項では、次の VxFS カーネルチューニングパラメータについて説明します。

- 「i ノードテーブルサイズのチューニング」
- 「i ノード割り当てのパフォーマンス最適化のチューニング」
- 「ファイルシステムの並行ダイレクト I/O のチューニング」
- 「Veritas Volume Manager の最大 I/O サイズ」
- 「パーティションディレクトリ」

i ノードテーブルサイズのチューニング

VxFS では、i ノードが i ノードテーブルにキャッシュされます。i ノードテーブルへのエントリ数を決定する VxFS のチューニングパラメータは、`vxfs_ninode` です。

VxFS では、i ノードテーブルへのエントリ数として `/etc/modprobe.conf` にある vxfs_ninode の値を使います。デフォルトでは、ファイルシステムはシステムのメモリサイズを基に算出した `vxfs_ninode` 値を使います。値を増やすには、次のように `/etc/modprobe.conf` を変更して再起動します。

```
options vxfs vxfs_ninode=new_value
```

この新しいパラメータは、再起動した後、または VxFS モジュールをロード解除してから再ロードした後に有効になります。VxFS モジュールは、`modprobe` コマンドを使ってロードすることもできますが、ファイルシステムのマウント時に自動的にロードされます。

`modprobe`(8)のマニュアルページを参照してください。

メモ: /etc/modprobe.conf ファイルの新しいパラメータは、insmod vxfs コマンドでは読み取れません。

i ノード割り当てのパフォーマンス最適化のチューニング

delicache_enable チューニングパラメータは、i ノード割り当てのパフォーマンス最適化および新規ファイル作成の間の i ノードの再利用をオンとオフのどちらにするかを指定します。delicache_enable チューニングパラメータは、クラスタファイルシステムでもサポートされます。delicache_enable には次の値を指定できます。

- 0 - delicache の最適化を無効にします。
- 1 - delicache の最適化を有効にします。

delicache_enable のデフォルト値は 1 です。

ファイルシステムの並行ダイレクト I/O のチューニング

VxFS では、各 iovec は readv(2) 呼び出しと writev(2) 呼び出しに対して同期的に実行されます。Single Unix Specification では、readv(2) と writev(2) の両方に対し、readv/writev() 関数は、次に進む前に常に 1 つの領域を完全に満たす必要があることが述べられています。しかし、ダイレクト I/O のために Linux でこの必要条件は無視され、完了を待つ前に、複数の iovec が送信されます。Veritas File System（VxFS）で並行ダイレクト I/O が読み取りと書き込みの両方に対して実行されるようになり、その結果 VxFS のパフォーマンスが向上します。並行ダイレクト I/O のサポートは、VxFS モジュール負荷チューニングパラメータ vx_parallel_dio の設定によって、有効にすることができます。

並行ダイレクト I/O を有効にするには、/etc/modprobe.conf ファイルに次の変更を加え、システムを再ブートします。

```
options vxfs vx_parallel_dio=1
```

パーティションディレクトリ

pdir_enable チューニングパラメータを設定することで、パーティションディレクトリ機能を有効または無効にできます。値に 1 を指定すると、パーティションディレクトリは有効になります。値に 0 を指定すると、パーティションディレクトリは無効になります。

デフォルト値は 0 です。

pdir_threshold チューニングパラメータを設定してディレクトリサイズのしきい値をバイト単位で指定できますが、パーティションディレクトリが有効になっていると、VxFS はそのサイズを超えてディレクトリにパーティションを割り当てることができます。デフォルト値は 32768 です。

fsadmコマンドに -dオプションを指定すると、パーティションディレクトリから空の隠しディレクトリが削除されます。パーティションディレクトリを無効にした場合でも、fsadm -dコマンドを実行すると、パーティションディレクトリは通常のディレクトリに変換されます。

パーティションディレクトリ機能はディスクレイアウトバージョン 8 以降のファイルシステムでのみ動作します。

警告: ディレクトリが巨大な場合、パーティションディレクトリから通常のディレクトリへの変換(またはその逆)は時間を必要とします。ルートディレクトリにすでに多数のファイルが含まれているときにこの機能を有効にすると、ファイルシステムのマウント時にこの変換が開始され、マウントに非常に長い時間がかかる原因になることがあります。Veritas はディレクトリが少ないとき、またはまだ作成されていないときに変換を実行することを推薦します。

Veritas Volume Manager の最大 I/O サイズ

VxFS を Veritas Volume Manager(VxVM)とともに使う場合、VxVM はデフォルトで 256 K を超えた I/O 要求を分割します。ストライピングを使う場合、パフォーマンスの最適化のため、ファイルシステムは I/O 要求をフルストライプのサイズに分割して発行する必要があります。しかし、ストライプサイズが 256 K を超える場合、デフォルトでは、I/O 要求が不必要に分割されてしまいます。

不要な I/O 要求の分割を避けるには、/etc/modprobe.conf ファイルの vol_maxio パラメータ値を変更して、最大 I/O サイズを拡張します。

クローンプロセスによるネーティブ非同期 I/O

vx_allow_cloned_naio チューニングパラメータを設定して、クローンプロセスによるネーティブ非同期 I/Oの有効と無効を切り替えることができます。クローンプロセスによるネーティブ非同期 I/O を有効にするには 1 を、無効にするには 0 を指定します。デフォルト値は 0 です。

CLONE_VM フラグを使ってクローンされたプロセスは、アドレス領域を親と共有します。そのようなスレッドが io_submit() 呼び出しを使ってネーティブ非同期 I/O を発行すると、I/O を完了する前にそれらのスレッドが戻って終了した場合は、システムにパニックが発生します。この問題を回避するには、vx_allow_cloned_naio チューニングパラメータを 0 に設定し、スレッドが I/O を同期的に発行するようにします。

保留中の非同期 I/O とともに終了することのないスレッドを持つ、適切に動作するアプリケーションには、この制限はありません。Sybase など、そのようなアプリケーションを使用する場合、vx_allow_cloned_naio チューニングパラメータを 1 に設定します。これにより、非同期 I/O を同期化するスレッドのパフォーマンスへの影響を回避できます。

DMP チューニングパラメータ

DMP は環境をチューニングするのに使うことができる各種パラメータを提供します。

表 B-1 に、チューニングできる DMP パラメータを示します。チューニングパラメータはオンラインで設定できます。再ブートは不要です。

表 B-1　　チューニング可能な DMP パラメータ

パラメータ	説明
dmp_cache_open	このパラメータを on に設定した場合、デバイスの最初の起動がキャッシュに保存されます。このキャッシュ処理により、デバイスの以降の起動によって発生するオーバーヘッドが最小化されるため、デバイス検出のパフォーマンスが高まります。このパラメータを off に設定した場合、キャッシュへの保存は行われません。 デフォルト値は on です。
dmp_daemon_count	サービスパスのエラー処理、パスリストア、その他の DMP 管理作業に使用できるカーネルスレッド数。 スレッドのデフォルト数は 10 です。
dmp_delayq_interval	アレイがスタンバイパスにフェールオーバーした後、再試行 I/O までの DMP の待機時間。一部のディスクアレイは、フェールオーバー直後の I/O 要求を受け入れることができません。 デフォルト値は 15 秒です。
dmp_display_alua_states	ALUA アレイの場合、このチューニングパラメータは PATH-TYPE[M] 列の PRIMARY または SECONDARY 状態ではなく、非対称アクセスの状態を表示します。 非対称アクセスの状態は次のとおりです。 ■ アクティブ/最適化 (Active/Optimized) ■ アクティブ/非最適化 (Active/Non-optimized) ■ スタンバイ (Standby) ■ 利用不能 (Unavailable) ■ 進行中の移行 (TransitionInProgress) ■ オフライン (Offline) デフォルトのチューニングパラメータの値は on です。

パラメータ	説明
dmp_fast_recovery	DMP が、HBA インターフェースから直接 SCSI エラー情報を取得するかどうかを示します。HBA インターフェースがエラー照会機能をサポートしている場合、値を on に設定するとエラーのリカバリが速くなる可能性があります。このパラメータを off に設定した場合、HBA インターフェースは使われません。 デフォルト設定は on です。
dmp_health_time	DMP は、断続的にエラーになっているパスを検出し、I/O 要求がこれらのパスに送信されないようにします。dmp_health_time の値は、パスが健全であり続けなければならない時間を秒で表します。この時間内にパスの状態が有効から無効に戻った場合、DMP はパスに断続的なエラーが発生していることを示すマークを付けて、dmp_path_age に設定している秒数が経過するまで I/O のパスを有効にしません。 デフォルト値は 60 秒です。 値を 0 に設定すると、DMP は断続的にエラーになっているパスを検出しません。
dmp_log_level	DMP コンソールメッセージで表示される詳細のレベル。次のレベル値が定義されています。 1 - 重要な DMP ログメッセージをすべて表示します。 2 - レベル 1 のメッセージに加えて、パスまたはディスクの追加または削除、SCSI エラー、I/O エラー、DMP ノードの移行に関するメッセージを表示します。 3 - レベル 1 と 2 のメッセージに加えて、パスの調整、障害の可能性のあるパス、アイドル状態のパス、異常なパスのロジックに関するメッセージを表示します。 4 - レベル 1、2、3 のメッセージに加えて、パスの属性の設定または変更に関するメッセージと、チューニングパラメータに関連する変更を表示します。 5 以上 - レベル 1 、2、3、4 のメッセージに加えて、他の詳しいメッセージを表示します。 デフォルト値は 1 です。

パラメータ	説明
dmp_low_impact_probe	リストアデーモンによるパスのプローブを最適化するかどうかを決定します。最適化は on に設定すると有効になり、off に設定すると無効になります。パスのプローブは、リストアポリシーが check_disabled である場合、または check_periodic ポリシーの check_disabled フェーズ中のみ最適化されます。 デフォルト値は on です。
dmp_lun_retry_timeout	HBA と SCSI ドライバが処理しない一時的なエラーを処理するための再試行間隔を指定します。 時間は秒数で指定します。 通常は、このような特殊な処理は不要です。そのため、dmp_lun_retry_timeout チューニングパラメータのデフォルト値は 30 です。ディスクのすべてのパスがエラーになった場合、DMP はアプリケーションの I/O をエラーにします。パスでは、接続性が 1 回のみチェックされます。 DMP による一時的なエラー処理が必要になる特殊な場合では、DMP がアプリケーション I/O をエラーにする処理を短期間延期するように設定します。この期間を指定するには、dmp_lun_retry_timeout チューニングパラメータを 0 以外の値に設定します。LUN のすべてのパスがエラーになっても、I/O を提供する必要がある場合、DMP は指定した期間中、5 秒ごとにパスをプローブします。この期間内にパスが復元された場合、DMP はそのことを検出して I/O を再試行します。どちらが先でも、指定した dmp_lun_retry_timeout が経過するか、またはパスの 1 つで I/O が正常に処理されないかぎり、DMP はエラーが発生したすべてのパスでディスクに I/O を送信する処理をエラーにしません。
dmp_monitor_fabric	DMP が SNIA HAB API から HBA イベントに登録するかどうかを指定します。これらのイベントは障害が切迫している I/O パスを避けることによってプロアクティブにフェールオーバーパフォーマンスを改善します。 この DDL 機能をサポートするようにパッチで修正したリリース 5.0 以前では、デフォルト設定は off です。5.0 以降のリリースでは、デフォルト設定は on です。

パラメータ	説明
dmp_monitor_ownership	ALUA のアレイの所有権の監視を有効にするかどうかを決定します。このチューニングパラメータを on に設定すると、DMP はデバイスをポーリングして LUN 所有権の変更を確認します。ポーリング間隔は dmp_restore_interval チューニングパラメータで指定します。デフォルト値は on です。 dmp_monitor_ownership チューニングパラメータが off の場合、DMP は LUN 所有権の変更を確認するポーリングを行いません。
dmp_native_support	DMP がネーティブデバイスのマルチパスを行うかどうかを決定します。 DMP によってネーティブデバイスのマルチパス化を行う場合は、チューニングパラメータを on に設定します。 Dynamic Multi-Pathing が Storage Foundation Cluster File System High Availability のコンポーネントとしてインストールされる場合、デフォルト値は off です。 Dynamic Multi-Pathing をスタンドアロン製品としてインストールする場合、デフォルト値は on です。
dmp_path_age	断続的にエラーの発生しているパスが継続して健全と判断されなければならない期間を示します。この期間が経過すると、DMP は再度そのパスに I/O 要求のスケジュール設定を試みます。 デフォルト値は 300 秒です。 値を 0 に設定すると、DMP は断続的にエラーになっているパスを検出しません。

パラメータ	説明
dmp_pathswitch_blks_shift	次に使用可能なパスに切り替わる前に DMP パスでアレイに送信される、連続 I/O ブロックのデフォルト数を示します。この値は 2 の累乗の指数(整数)で指定するようになっており、たとえば 9 は 512 ブロックを表します。 デフォルト値は 9 です。この場合、512 ブロック(256 KB)の連続 I/O が切り替え前に DMP パスで送信されます。内部データキャッシュを持つ高機能ディスクアレイの場合、このチューニングパラメータの値を大きくすることで、スループットが向上する可能性があります。たとえば、日立 SANRISE2800 アクティブ/アクティブアレイの場合、シーケンシャルな読み取りまたは書き込みを主として構成される I/O 処理パターンに最適な値は 15 - 17 です。 このパラメータの影響を受けるのは、I/O ポリシーを balanced に設定している場合の動作のみです。パラメータ値を 0 にすると、vxdmpadm コマンドでアレイに別のパーティションサイズを指定した場合を除き、このポリシーではマルチパス化が無効になります。 p.338 の「I/O ポリシーの指定」を参照してください。
dmp_probe_idle_lun	DMP 統計情報の収集が有効になっている場合、DMP パスのリストアスレッドがアイドル状態の LUN をプローブするように、このチューニングパラメータを on (デフォルト) に設定します。機能をオフにするには、このチューニングパラメータを off に設定します。(アイドル状態の LUN とは、I/O 要求がスケジュール設定されていない VM ディスクを指します。) このチューニングパラメータの値は、DMP 統計の収集が有効になっているときにのみ解釈されます。統計の収集をオフにすると、アイドル状態の LUN のプローブも無効になります。 デフォルト値は on です。
dmp_probe_threshold	dmp_low_impact_probe を on に設定している場合、dmp_probe_threshold は同じサブパスフェールオーバーグループに属する他のパスの状態を変更する前に、プローブするパスの数を決定します。 デフォルト値は 5 です。

パラメータ	説明
dmp_restore_cycles	DMP リストアポリシーが check_periodic の場合に、check_all ポリシーが呼び出されるまでのサイクル数です。 デフォルト値は 10 です。 p.351 の「DMP パスリストアポリシーの設定」を参照してください。
dmp_restore_interval	interval 属性値には、パスリストアスレッドがパスを調べる頻度を指定します。時間は秒数で指定します。 デフォルト値は 300 です。 このチューニングパラメータの値は、vxdmpadm start restore コマンドを使って設定することもできます。 p.351 の「DMP パスリストアポリシーの設定」を参照してください。
dmp_restore_policy	次のいずれかの値に設定可能な DMP リストアポリシー。 ■ check_all ■ check_alternate ■ check_disabled ■ check_periodic デフォルト値は check_disabled です。 このチューニングパラメータの値は、vxdmpadm start restore コマンドを使って設定することもできます。 p.351 の「DMP パスリストアポリシーの設定」を参照してください。
dmp_restore_state	このパラメータを enabled に設定すると、パスリストアスレッドの開始が有効になります。 p.351 の「DMP パスリストアポリシーの設定」を参照してください。 このパラメータを disabled に設定した場合、パスリストアスレッドは停止し、無効になります。 このパラメータを stopped に設定した場合、次のデバイスの検出サイクルまでパスリストアスレッドは停止します。 デフォルトは enabled です。 p.353 の「DMP パスリストアスレッドの停止」を参照してください。

パラメータ	説明
dmp_scsi_timeout	DMP 経由で送信される SCSI コマンドにタイムアウト値を設定する必要があるかを判別します。タイムアウト時間内にデバイスに送信されたことを示す SCSI コマンドの応答を HBA が受信しない場合、SCSI コマンドは障害エラーコードとともに返されます。 デフォルト値は 20 秒です。
dmp_sfg_threshold	DMP が同じフェールオーバーグループに属する他のパスの検討を開始する前に、フェールオーバーグループ内で障害が発生する必要があるパスの最小数を決定します。値が 0 の場合、サブパスフェールオーバーグループに基づくフェールオーバーのロジックが無効になります。 デフォルト値は 1 です。
dmp_stat_interval	DMP 統計情報を収集する間隔。 最小値は 1 秒で、これがデフォルトです。

Dynamic Multi-Pathing チューニングパラメータを変更する方法

DMP（Dynamic Multi-Pathing）には、設定をチューニングするために使用できるさまざまなパラメータがあります。

p.1115 の「DMP チューニングパラメータ」を参照してください。

次のいずれかのメソッドを使って、DMP チューニングパラメータを変更します。

vxdmpadm settune コマンドを使って値を表示または変更する。	p.1121 の「vxdmpadm settune コマンドラインを使った DMP パラメータの値の変更」を参照してください。
vxdmpadm コマンドのテンプレートメソッドを使う。	p.1122 の「テンプレートを使った DMP（Dynamic Multi-Pathing）のチューニングについて」を参照してください。

vxdmpadm settune コマンドラインを使った DMP パラメータの値の変更

DMP チューニングパラメータを設定するには、次のコマンドを使います。

```
# vxdmpadm settune dmp_tunable=value
```

DMP チューニングパラメータの値を表示するには、次のコマンドを使います。

```
# vxdmpadm gettune [dmp_tunable]
```

また、テンプレートメソッドを使っても、DMP チューニングパラメータの表示や変更が可能です。

p.1122 の「テンプレートを使った DMP（Dynamic Multi-Pathing）のチューニングについて」を参照してください。

テンプレートを使った DMP（Dynamic Multi-Pathing）のチューニングについて

Dynamic Multi-Pathing には、処理効率を最適化するために設定できる複数のチューニングパラメータや属性があります。DMP は、1 回の操作で複数のチューニングパラメータと属性を更新できるテンプレートメソッドを提供します。テンプレートは DMP 設定の全部か一部を表し、ホストのパラメータと属性の値を示します。

チューニングパラメータの表示や操作を行うには、ファイルに DMP チューニングパラメータの設定値をダンプできます。必要に応じて、パラメータと属性を編集します。その後、ホストにテンプレートファイルをロードして、1 回の操作ですべての値を更新します。

設定ファイルは、同じホストにロードすることも、別の同様のホストにロードすることもできます。テンプレートメソッドは、次のシナリオに有用です。

- 処理効率を最適化するチューニング値を使って、複数の同様のホストを設定する。
  1 つのホストを設定して、処理効率を最適化します。ホストの設定後、テンプレートファイルにチューニングパラメータと属性をダンプします。同様の要件を持つ別のホストに、そのテンプレートファイルをロードできます。ベリタスは同じ設定テンプレートを使うホストは、オペレーティングシステムが同じで、I/O 要件も同様であることを推奨します。
- 複数の特殊なテンプレートを定義して、異なる I/O 負荷要件を処理する。
  ホストの負荷が変更されたときに、異なるテンプレートをロードして処理効率の最適化を図ることができます。この戦略は、予測可能な一時的な I/O 負荷の変更に対応する場合に適切です。システム管理者は、システムの I/O 負荷の動作を定義した後で、特定の負荷に合わせてチューニングテンプレートをカスタマイズできます。スクリプトか cron ジョブで使うことができる単一のロードコマンドがあるため、チューニングを自動化できます。

いつでも、設定はリセットできます。その場合、チューニングパラメータと属性の値は、DMP のデフォルト値に復帰します。

`vxdmpadm config` コマンドを使って、DMP 設定ファイルを管理できます。

`vxdmpadm`（1M）マニュアルページを参照してください。

DMP チューニングテンプレート

テンプレートの機構は、ファイルまたは標準出力に設定値をダンプすることで、DMP のパラメータと属性のチューニングを可能にします。

DMP では、次のタイプの情報について、テンプレートファイルを使ったチューニングがサポートされます。

- DMP チューニングパラメータ。
- エンクロージャ、アレイ名、またはアレイタイプに定義する DMP 属性
- Veritas の名前の付け方のパラメータ。

テンプレートファイルは、次のようなセクションに分かれています。

DMP チューニングパラメータ	すべてのエンクロージャとアレイに適用されます。
Namingscheme	すべてのエンクロージャとアレイに適用されます。
Arraytype	アレイタイプのカスタマイズに使用します。指定したアレイタイプのすべてのエンクロージャに適用されます。
Arrayname	特定のアレイでカスタマイズが必要な場合、つまり、チューニングパラメータがアレイタイプに適用されるものと異なる場合に使用します。 このセクションの属性は、指定したアレイ名のすべてのエンクロージャ適用されます。
Enclosurename	指定した Cab シリアル番号とアレイ名のエンクロージャに適用されます。 特定のエンクロージャでカスタマイズが必要な場合、つまり、チューニングパラメータがアレイタイプとアレイ名に適用されるものと異なる場合に使用します。

ロードはセクション単位で実行されます。DMP は、各セクションにつき、すべての属性が有効な場合のみ、そのセクションをロードします。すべてのセクションが処理されたら、DMP はエラーと警告の一覧をユーザーに報告します。DMP は部分的なロールバックをサポートしません。DMP はロード処理中にチューニングパラメータと属性を検証します。ただし、Veritas はファイルのロード前に、設定テンプレートファイルを調べることを推奨します。設定ファイルが正しいことが確認されるまで、必要に応じて修正してください。

テンプレートにロード時に、各属性は次の順序の優先度が割り当てられます。

エンクロージャセクション > アレイ名セクション > アレイタイプセクション

同じアレイタイプのすべてのエンクロージャに同じ設定が必要な場合は、テンプレートから対応するアレイ名セクションとエンクロージャ名セクションを削除します。アレイタイプセ

クションの設定のみを定義します。一部のエンクロージャかアレイ名で設定をカスタマイズする必要がある場合は、アレイ名またはエンクロージャの属性セクションを保持します。アレイタイプに定義した設定と同じ設定を使う場合は、エンクロージャまたはアレイ名のエントリを削除できます。

ホストから設定ファイルをダンプするときに、そのホストには他のホストでは表示されないアレイが含まれていることがあります。そのようなエンクロージャ、アレイタイプ、またはアレイ名を含まないチューニング先のホストにテンプレートをロードすると、DMP は該当セクションを無視します。

チューニング先ホストの非共有のアレイやホスト固有のアレイに設定を適用しない場合は、該当する各アレイのエンクロージャセクションをテンプレートに必ず定義してください。チューニング先のホストにテンプレートファイルをロードするときに、エンクロージャセクションによって設定が決められます。定義しないと、DMP は各アレイ名またはアレイタイプのセクションの設定を適用します。

DMP チューニングテンプレートの例

このセクションでは、DMP チューニングテンプレートの例を示します。

```
DMP Tunables
        dmp_cache_open=on
        dmp_daemon_count=10
        dmp_delayq_interval=15
        dmp_restore_state=enabled
        dmp_fast_recovery=on
        dmp_health_time=60
        dmp_log_level=1
        dmp_low_impact_probe=on
        dmp_lun_retry_timeout=30
        dmp_path_age=300
        dmp_pathswitch_blks_shift=9
        dmp_probe_idle_lun=on
        dmp_probe_threshold=5
        dmp_restore_cycles=10
        dmp_restore_interval=300
        dmp_restore_policy=check_disabled
        dmp_retry_count=5
        dmp_scsi_timeout=20
        dmp_sfg_threshold=1
        dmp_stat_interval=1
        dmp_monitor_ownership=on
        dmp_monitor_fabric=on
        dmp_native_support=off
Namingscheme
```

```
        namingscheme=ebn
        persistence=yes
        lowercase=yes
        use_avid=yes
Arraytype
        arraytype=CLR-A/PF
        iopolicy=minimumq
        partitionsize=512
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
Arraytype
        arraytype=ALUA
        iopolicy=adaptive
        partitionsize=512
        use_all_paths=no
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
Arraytype
        arraytype=Disk
        iopolicy=minimumq
        partitionsize=512
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
Arrayname
        arrayname=EMC_CLARiiON
        iopolicy=minimumq
        partitionsize=512
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
Arrayname
        arrayname=EVA4K6K
        iopolicy=adaptive
        partitionsize=512
        use_all_paths=no
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
Arrayname
        arrayname=Disk
```

```
        iopolicy=minimumq
        partitionsize=512
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
Enclosure
        serial=CK200051900278
        arrayname=EMC_CLARiiON
        arraytype=CLR-A/PF
        iopolicy=minimumq
        partitionsize=512
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
        dmp_lun_retry_timeout=30
Enclosure
        serial=50001FE1500A8F00
        arrayname=EVA4K6K
        arraytype=ALUA
        iopolicy=adaptive
        partitionsize=512
        use_all_paths=no
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
        dmp_lun_retry_timeout=30
Enclosure
        serial=50001FE1500BB690
        arrayname=EVA4K6K
        arraytype=ALUA
        iopolicy=adaptive
        partitionsize=512
        use_all_paths=no
        recoveryoption=nothrottle
        recoveryoption=timebound iotimeout=300
        redundancy=0
        dmp_lun_retry_timeout=30
Enclosure
        serial=DISKS
        arrayname=Disk
        arraytype=Disk
        iopolicy=minimumq
        partitionsize=512
```

```
recoveryoption=nothrottle
recoveryoption=timebound iotimeout=300
redundancy=0
dmp_lun_retry_timeout=30
```

設定属性テンプレートを使った DMP のホストのチューニング

テンプレートファイルを使って、同じホストまたは別の同様のホストに DMP 設定の一連の変更をアップロードできます。

Veritas はチューニングパラメータ値の設定もとのホストと同様のホストに DMP のテンプレートをロードすることを推奨します。

テンプレートを使ってホストの DMP を設定するには

1 ファイルに現在のホストの設定内容をダンプします。

```
# vxdmpadm config dump file=filename
```

2 ファイルを編集し、テンプレートのチューニングパラメータを必要に応じて変更します。

チューニング先のホストには、非共有のアレイかホスト固有のアレイが含まれることがあります。このようなアレイでアレイ名またはアレイタイプの設定が更新されるのを回避するには、テンプレートで各アレイのエンクロージャセクションを定義します。チューニング先のホストにテンプレートファイルをロードするときに、エンクロージャセクションによって設定が決められます。 定義しないと、DMP は各アレイ名またはアレイタイプのセクションの設定を適用します。

3 DMP チューニングパラメータの値を検証します。

```
# vxdmpadm config check file=filename
```

DMP は設定の検査が成功なら出力を表示しません。ファイルにエラーがある場合は、DMP はエラーを表示します。設定ファイルが有効になるまで、必要に応じて修正してください。たとえば、次のようなエラーが表示されることがあります。

```
VxVM vxdmpadm ERROR V-5-1-0 Template file 'error.file' contains
following errors:

Line No: 22  'dmp_daemon_count' can not be set to 0 or less
Line No: 44  Specified value for 'dmp_health_time' contains
non-digits
Line No: 64  Specified value for 'dmp_path_age' is beyond
the limit of its value
Line No: 76  'dmp_probe_idle_lun' can be set to either on or off
Line No: 281 Unknown arraytype
```

4 チューニング先のホストにファイルをロードします。

```
# vxdmpadm config load file=filename
```

ロード処理中に、DMP はテンプレートの各セクションを検証します。DMP はすべての有効なセクションをロードします。DMP はエラーがあるセクションはロードしません。

DMP 設定ファイルの管理

ホストに前回ロードしたテンプレートファイルの名前を表示できます。この情報には、DMP がテンプレートファイルをロードした日時が含まれます。

ホストが現在使っているテンプレートファイルの名前を表示するには

◆

```
# vxdmpadm config show

TEMPLATE_FILE      DATE             TIME
=============================================
/tmp/myconfig      Feb 09, 2011     11:28:59
```

DMP のチューニングパラメータと属性のデフォルト値へのリセット

DMP は DMP のチューニングパラメータと属性のデフォルト値を維持します。いつでも、ホストにデフォルト値を復元できます。テンプレートファイルを使ってホストに適用した変更は、破棄されます。

DMP チューニングパラメータをデフォルト値にリセットするには

◆ 次のコマンドを実行します。

```
# vxdmpadm config reset
```

テンプレートでサポートされる DMP のチューニングパラメータと属性

DMP では、次のチューニングパラメータと属性について、設定テンプレートを使ったチューニングがサポートされます。

DMP チューニングパラメータ	p.1115 の「DMP チューニングパラメータ」を参照してください。
エンクロージャ、アレイ名、またはアレイタイプに定義する DMP 属性	■ iopolicy ■ partitionsize ■ use_all_paths ■ recoveryoption 属性（retrycount または iotimeout） ■ 冗長性 ■ dmp_lun_retry_timeout
名前の付け方の属性:	■ 名前の付け方 ■ 永続 ■ lowercase ■ use_avid

次のチューニングパラメータは、テンプレートでサポートされません。

- OS のチューニングパラメータ
- TPD モード
- エンクロージャのフェールオーバー属性（failovermode）

VxVM のチューニングパラメータ

VxVM（Veritas Volume Manager）には、環境のチューニングで使うことができるパラメータがいくつか用意されています。VxVM チューニングパラメータは、いくつかのコンポーネントを構成します。

チューニングパラメータの VxVM コンポーネントを次に示します。

- basevm
  VxVM のコア機能をチューニングするためのパラメータ。

p.1130 の「コア VxVM のチューニングパラメータ」を参照してください。

- cvm
  CVM（Cluster Volume Manager）をチューニングするためのパラメータ。
  p.1141 の「CVM のチューニングパラメータ」を参照してください。
- fmr
  FlashSnap 機能（FMR）をチューニングするためのパラメータ。
  p.1136 の「FlashSnap（FMR）のチューニングパラメータ」を参照してください。
- vvr
  VVR（Veritas Volume Replicator）をチューニングするためのパラメータ。
  p.1142 の「VVR のチューニングパラメータ」を参照してください。

コア VxVM のチューニングパラメータ

表 B-2 に VxVM のカーネルチューニングパラメータを一覧表示します。

特に指定のないかぎり、vxtune コマンドまたはオペレーティングシステムのメソッドを使ってパラメータをチューニングできます。

表 B-2　　コア VxVM のカーネルチューニングパラメータ

パラメータ	説明
vol_checkpt_default	回復または再同期化処理を実行しているユーティリティが、処理の済んだ範囲を示すオフセットをカーネルにチェックポイントとして読み込む間隔を示します。このような処理を行っている間にシステム障害が発生した場合は、完全なリカバリを実行する必要がなく、到達した最後のチェックポイントから継続できます。 デフォルト値は 20480 セクタ（10 MB）です。 このサイズを大きくすると、回復処理のチェックポイント操作の負荷が減少しますが、回復中にシステム障害が生じると、その後さらに回復を実行する必要があります。
vol_default_iodelay	ユーティリティが I/O 要求を発行する頻度を軽減するよう指示されており、特定の遅延時間が指定されていない場合に、ユーティリティが一時停止するクロック刻みのカウントを示します。このチューニングパラメータ値は、ミラーの再同期化や RAID 5 カラムの再構築などの操作を実行するユーティリティで使います。 デフォルト値は 50 刻みです。 この値を大きくすると、回復処理が遅くなり、回復実行中のシステムの動作が低下します。

パラメータ	説明
vol_max_adminio_poolsz	管理 I/O 操作で使われるメモリプールの最大サイズ。VxVM は、管理 I/O を調整する場合にこのプールを使います。 デフォルト値は 64 MB です。最大サイズは、voliomem_maxpool_sz パラメータより大きな値にはできません。
vol_max_vol	このパラメータは、vxtune コマンドを使ってチューニングできます。システムで作成できるボリュームの最大数を示します。許可されている最小値は 1 です。許可されている最大値は、システムで表すことのできるマイナー番号の最大数です。 デフォルト値は 65534 です。
vol_maxio	要求を分割しないで実行できる論理 I/O 操作の最大サイズを示します。この値より大きい VxVM への I/O 要求は分割され、並列実行されます。物理 I/O 要求はディスクデバイスの容量に基づいて分割され、この最大論理要求制限値を変更しても影響を受けません。 デフォルト値は 2048 セクタ(1 MB)です。 voliomem_maxpool_sz 値は、少なくとも vol_maxio 値の 10 倍以上である必要があります。 DRL シーケンシャルログを設定した場合には、voldrl_min_regionsz 値は、少なくとも vol_maxio 値の半分の値に設定する必要があります。 vol_maxio の最大値は、物理メモリまたはカーネルの仮想メモリ容量のどちらか小さい方の 20 % です。この制限を超える値は推奨されません。
vol_maxioctl	ioctl 呼び出しにより VxVM に渡すことのできる最大データサイズを示します。この制限値を大きくすると、より大きいサイズの操作を一度に実行できます。ユーティリティによっては、特定のサイズの操作を実行するように設定されており、それより大きいサイズの ioctl 要求を発行すると、予期せずに失敗することがあるため、制限値を小さくすることは一般的にはお勧めしません。 デフォルト値は 32768 バイト(32 KB)です。
vol_maxparallelio	VOL_VOLDIO_WRITE ioctl 呼び出しごとに 1 つの VOL_VOLDIO_READ で、vxconfigd デーモンがカーネルに対して要求できる I/O 操作数を示します。 デフォルト値は 256 です。この値は変更しないでください。

パラメータ	説明
vol_maxspecialio	ioctl 呼び出しで発行できる I/O 要求の最大サイズを示します。ioctl 要求自体は小さいサイズですが、大きい I/O 要求を実行するよう要求できます。このチューニングパラメータで、これらの I/O 要求のサイズが制限されます。必要に応じて、この値を超えた要求をエラーにしたり、要求を分割して並行して実行することができます。 デフォルト値は 2048 セクタです。 この制限値を大きくすると、I/O 要求のサイズが大きくなった場合、プロセスに存在するより多くのメモリやカーネルの仮想マップ領域が必要となるため処理できなくなる可能性があります。このチューニングパラメータの最大制限値は、物理メモリまたはカーネルの仮想メモリより 20 % 小さい値です。この制限値を超えると処理できなくなる可能性があるため、この制限値を超えないようにしてください。 ストライプがこのチューニングパラメータの値より大きい場合は、ストライプ単位の I/O 要求は分割され、ストライプ全体に対する読み取りおよび書き込みは一度に実行されません。これにより、順次 I/O や大規模な I/O 要求に対してボリュームの I/O スループットが低下します。 このチューニングパラメータにより、個々のディスクのレベルより高いレベルの VxVM 内の I/O 要求のサイズが制限されます。たとえば、8 × 64 KB のストライプの場合、256 KB のデフォルト値によって、ストライプのディスクの半分を使う I/O 要求が可能となるため、可能なスループットは半分となります。より多くのカラムがある場合またはより大きいインタリーブ係数を使っている場合は、相対的な処理効率はさらに低下します。 このチューニングパラメータは、少なくとも、最大ストライプ（RAID 0 または RAID 5）のサイズに設定する必要があります。
vol_stats_enable	Veritas Volume Manager オブジェクトの I/O 統計情報収集を有効または無効にします。この機能はデフォルトで有効になっているため、デフォルト値は 1 です。
vol_subdisk_num	1 つのプレックスに接続できる最大サブディスク数を示します。このチューニングパラメータのデフォルト値は 4096 です。

パラメータ	説明
voliomem_chunk_size	システムメモリを割り当てたり解放する場合、VxVM が使うメモリ領域のサイズを示します。サイズが大きいと、VxVM は大きいメモリ容量を保持できるため、メモリ割り当てによる CPU のオーバーヘッドは軽減されます。 このチューニングパラメータの値は、システムのページサイズによって決まります。デフォルト値より大きい値は指定できません。この値を変更した場合、VxVM はシステムを再ブートしたときに値をページサイズに整列させます。 ページサイズが 512 バイトの場合、デフォルト値は 32 KB です。
voliomem_maxpool_sz	VxVM が内部的にシステムに要求する最大メモリサイズを示します。このチューニングパラメータによって、1 つの I/O 操作でシステム内のすべてのメモリが使われることを防ぐことができるため、この値は、VxVM のパフォーマンスに直接影響します。 VxVM では、このサイズまで拡張可能なプールが、RAID 5 ボリュームとミラーボリューム用に 1 つずつ割り当てられます。インスタント(コピーオンライト)スナップショットが存在する場合は追加のプールが割り当てられます。 プールサイズの 1/4 よりも大きい RAID 5 ボリュームへの書き込み要求は分割され、プールサイズの 1/10 ずつ実行されます。 プールサイズよりも大きいミラーボリュームへの書き込み要求は分割され、プールサイズで実行されます デフォルト値は 134217728(128 MB)です。 voliomem_maxpool_szの値は volraid_minpool_sizeの値より大きい必要があります。 voliomem_maxpool_sz 値は、少なくとも vol_maxio 値の 10 倍以上である必要があります。

パラメータ	説明
voliot_errbuf_dflt	エラートレースイベント用に確保されているバッファのデフォルトサイズを示します。このバッファはドライバのロード時に割り当てられ、VxVM 実行中はサイズの調整はできません。 デフォルト値は 16384 バイト(16 KB)です。 このバッファを大きくすると、システムメモリが使われますが、エラーイベント用のストレージが多くなります。このバッファサイズを小さくすると、トレースデバイスによってエラーが検出されないことがあります。エラートレースに応じて応答動作を実行するアプリケーションは、このバッファに影響を受けます。
voliot_iobuf_default	トレースの一部として ioctl が使うカーネルバッファサイズについての指定がない場合に、トレースバッファを作成するためのデフォルトサイズを示します。 デフォルト値は 8192 バイト(8 KB)です。 このバッファサイズが小さすぎるために、トレースデータが損失されることが多い場合は、この値をより大きい値に変更できます。
voliot_iobuf_limit	カーネルでトレースバッファの格納に使えるメモリサイズの上限値を示します。トレースバッファは VxVM カーネルがトレースイベントレコードを格納するために使います。トレースバッファをカーネルに格納するよう要求されると、そのためのメモリがこのプールから引き出されます。 このサイズを大きくすると、システムメモリが使われますが、追加のトレースを実行できます。この値をシステムですぐに使えるメモリサイズより大きい値に設定することはお勧めできません。 デフォルト値は 131072 バイト(128 KB)です。
voliot_iobuf_max	1 つのトレースバッファに使える最大バッファサイズを示します。このサイズより大きいバッファの要求は、このサイズになるよう自動的に切り捨てられます。トレースインターフェースから要求される最大バッファサイズは(使用制限により)このサイズのバッファになります。 デフォルト値は 65536 バイト(64 KB)です。 このバッファを大きくすると、使用率が高いボリュームでも、より大きいトレースを実行できます。 この値は、voliot_iobuf_limit チューニングパラメータの値より大きい値に設定しないでください。

パラメータ	説明
voliot_max_open	同時に開くことのできる最大トレースチャネル数を示します。トレースチャネルはトレースデバイスドライバへのクローンエントリポイントです。システムで実行される各 vxtrace プロセスは 1 つのトレースチャネルを使います。 チャネルのデフォルト数は 32 です。 チャネルが使われていなくても、各チャネルの割り当てには約 20 バイト必要です。
volraid_minpool_size	このパラメータは、vxtune コマンドを使ってチューニングできます。RAID 5 ボリュームの操作のために VxVM がシステムに要求するメモリの初期サイズを示します。このメモリプールの最大サイズは、voliomem_maxpool_sz の値によって制限されます。 デフォルト値は 8192 セクタ(4 MB)です。
volraid_rsrtransmax	RAID 5 ボリュームに、並行して実行できる最大一時再構築操作数を示します。一時再構築操作とは、縮退されていない RAID 5 ボリューム上で生じる予測されていない操作です。同時に発生し得るこれらの操作数を制限すると、システムが多くの再構築操作でメモリが消費される可能性がなくなり、メモリ不足が生じるリスクも軽減されます。 デフォルト値は 1 です。 このサイズを大きくすると、障害が最初に生じたときと、障害オブジェクトの切断を実行する前のシステム上の初期処理効率は改善されますが、メモリが不足する可能性があります。
autostartvolumes	自動ボリュームリカバリをオンまたはオフにします。オンに設定すると、ディスクグループをインポート、結合、移動、分割すると VxVM は自動的にリカバリされ、無効なボリュームが起動します。オフに設定すると、この動作は行われません。デフォルト値は on です。
fssmartmovethreshold	個々のファイルシステムのしきい値です。合計で 100% です。このしきい値に達すると、SmartMove 機能は使われません。デフォルト値は 100 です。
reclaim_on_delete_start_time	シン LUN で、その LUN を使用するボリュームが削除された後に再生が開始される時刻です。24 時間制(HH:MM)で指定します。デフォルト値は 22:10 です。

パラメータ	説明
reclaim_on_delete_wait_period	シン LUN で、その LUN を使用するボリュームが削除された後に領域の再生利用を開始するまでの待機日数です。-1 から 366 までの整数で指定します。-1 を指定するとただちに開始され、366 を指定すると開始されません。デフォルト値は 1 です。
usefssmartmove	SmartMove 機能の状態。有効な値は次のとおりです。 ■ thinonly −シンディスクのみで使用します。 ■ all − すべてのディスクで使用します。 ■ none − SmartMove 機能をオフにします。 デフォルト値は、all です。

FlashSnap（FMR）のチューニングパラメータ

表 B-3 では、FlashSnap のカーネルチューニングパラメータを一覧表示しています。vxtune コマンドは、これらのパラメータを FMR コンポーネントの下に分類します。

特に指定のないかぎり、vxtune コマンドまたはオペレーティングシステムのメソッドを使ってパラメータをチューニングできます。

表 B-3 FlashSnap（FMR）のカーネルチューニングパラメータ

パラメータ	説明
vol_fmr_logsz	非永続 FastResync がボリューム内の変更されたブロックを追跡するために使うビットマップの最大サイズ（KB 単位）を示します。ビットマップの各ビットにマップされるボリューム内のブロック数は、ボリュームのサイズに応じて異なり、ボリュームのサイズが変わると、ブロック数も変わります。 たとえば、ボリュームサイズが 1 GB、システムブロックサイズが 512 バイト、かつこのチューニングパラメータの値が 4 の場合、16,384 ビットのマップが生成されるため、各ビットは 128 ブロックの領域を表します。 ビットマップサイズが大きくなると、各ビットにマップされるブロック数は少なくなります。これにより、再同期化で必要な読み取りおよび書き込みの量が軽減されますが、ビットマップに必要な非ページカーネルメモリがより多くなります。さらに、クラスタ化されたシステムでは、ビットマップサイズが増加すると、I/O 処理効率の遅延が長くなり、クラスタメンバー間のプライベートネットワークの負荷も増加します。これは、クラスタの他のすべてのメンバーは、マップ内のビットが設定されるたびにそれを通知される必要があるからです。 共有ボリュームの場合、領域のサイズはクラスタ内のすべてのノード上で同じである必要があるため、このチューニングパラメータの値がマスターノードとスレーブノードで異なる場合は、マスターノードの値が優先されます。共有ボリュームの値は変更される可能性があるため、このボリュームの寿命中、このチューニングパラメータの値は保持されます。 スナップショットプレックスが接続された何千ものミラーが存在する設定では、メモリオーバーヘッドの総量が、VxVM で通常消費されるメモリのオーバーヘッドより大幅に多くなることがあります。 デフォルト値は 4 KB です。許可されている最小値は 1 KB、最大値は 8 KB です。 KB 単位で値を vxtune に指定します。 **メモ:** このチューニングパラメータの値は、永続 FastResync には影響を及ぼしません。

パラメータ	説明
voldrl_dirty_regions	このパラメータは、拡張された DCO レイアウト（バージョン 30）にのみ適用されます。 同じリージョンへの別の書き込みによって DRL が更新される前にキャッシュするためのダーティリージョン数を表します。この値を小さくすると DRL の更新が頻繁に行われることになり、パフォーマンスが低下します。この値を大きくすると I/O パフォーマンスは向上しますが、DRL で使うメモリ量が増加します。 デフォルト値は 1024 です。
voldrl_max_drtregs	ボリュームの非シーケンシャル DRL 用にシステムに作成できる最大ダーティリージョン数を示します。この値が大きい場合は、リカバリ時間が長くなりますが、システム処理効率が向上します。このチューニングパラメータは、障害発生後のシステムで回復時間が遅い場合のチューニングに使えます。 デフォルト値は 2048 です。
voldrl_max_seq_dirty	シーケンシャル DRL に対して許可されている最大ダーティリージョン数を示します。これは、データベースのログなど、データが連続的に書き込まれるボリュームに便利です。ダーティリージョン数を制限すると、クラッシュが生じたときに、より短時間で回復を実行できます。 デフォルト値は 3 です。
voldrl_min_regionsz	DRL（dirty region logging）のボリューム領域の最小セクタ数を示します。DRL の場合、VxVM では、ボリュームは論理的に一連の連続領域に分割されます。領域のサイズが大きいと、領域のキャッシュヒット率が改善されます。これにより、書き込みの処理速度は向上しますが、回復時間が長くなります。 デフォルト値は 512 セクタです。 DRL シーケンシャルログを設定した場合には、voldrl_min_regionsz 値は、少なくとも vol_maxio 値の半分の値に設定する必要があります。 値をセクタ単位で指定します。
voldrl_volumemax_drtregs	従来の DRL を使用するミラーボリュームのダーティリージョンの 1 ボリューム当たりの最大制限数。頻繁に使われるボリュームの場合、このパラメータの値を大きくしてパフォーマンスを高めます。 デフォルト値は 256 です。

パラメータ	説明
voldrl_volumemax_drtregs_20	バージョン 20 の DCO を使用するミラーボリュームのダーティリージョンの 1 ボリューム当たりの最大制限数。頻繁に使われるボリュームの場合、このパラメータの値を大きくしてパフォーマンスを高めます。デフォルト値は 1024 です。

パラメータ	説明
volpagemod_max_memsz	FastResync のキャッシュとキャッシュオブジェクトメタデータのために割り当てられるメモリのサイズを示します。このキャッシュに割り当てられるメモリは専用で排他的に使われ、別のプロセスやアプリケーションでは使えません。 デフォルト値は 6144 KB（6 MB）です。 インスタントスナップショット操作用に準備されたキャッシュオブジェクトまたはボリュームがシステムに存在する場合は、この値を 512 KB 未満に設定することはできません。バージョン 20 の DCO ボリュームによって実装される FastResync 機能または DRL 機能を使わない場合は、値を 0 に設定することも可能です。後になってこれらの機能を有効にする必要が生じた場合は、この値を適切な値に変更します。 値を KB 単位で指定します。新しい値は自動的にページサイズに整列されますが、指定した実際の値は永続的にされます。 値は、リージョンサイズとインスタンススナップショットを取得するボリューム数に基づいて決定します。ページングモジュールサイズは、少なくとも最大のサイズボリュームに必要なサイズの 2 倍である必要があり、次の公式によって計算されます。 *size_in_KB* = 6 * (*total_volume_size_in_GB*) * (64/*region_size_in_KB*) たとえば、領域のサイズが 64 KB の場合、1 TB の単一のボリュームには約 6 MB のページングメモリが必要です。チューニングパラメータの最小値は、少なくともその 2 倍、つまり 12 MB です。 複数のボリュームがある場合、すべてのボリュームで同じページングモジュールを共有します。最大必要条件は、上記の公式にボリューム数を掛けることで計算されます。ただし、より適切な値は、各ボリュームに対する平負荷によって決まります。たとえば、各ボリュームの 20% しか更新されない場合、パフォーマンスを犠牲にすることなく、ページングモジュールサイズはそれに比例して削減できます。最大のボリュームについての最小必要条件は、依然として満たす必要があります。たとえば、それぞれが 1 TB のボリュームが 10 個ある場合、ページングメモリの当初の計算結果は 60 MB となります。データの 20% しか更新しない場合には、修正値として 12 MB に計算します。

CVM のチューニングパラメータ

表 B-4 では、CVM のカーネルチューニングパラメータを一覧表示しています。特に指定のないかぎり、vxtune コマンドまたはオペレーティングシステムのメソッドを使ってパラメータをチューニングできます。

表 B-4　　CVM のカーネルチューニングパラメータ

autoreminor	自動再マイナー機能をオンまたはオフにします。ディスクグループまたはそのオブジェクトのデバイスのマイナー番号が既存のディスクグループの番号と競合している場合、ディスクグループはインポートできません。autoreminor がオンの場合、VxVM がインポート中に競合を検出すると、VxVM はディスクグループに自動的に新しいマイナー番号を割り当てます。その後、ディスクグループがインポートされます。デフォルト値は on です。 NFS ファイルシステムなどの一部のシナリオでは、新しいマイナー番号を割り当てると問題が発生することがあります。その場合、チューニングパラメータをオフに設定してください。autoreminor パラメータがオフに設定されている場合、force (-f) オプションを指定していても、競合しているマイナー番号の付いているディスクグループをインポートしようとすると失敗します。ディスクグループを手動で再度マイナーすると、ディスクグループをインポートできるようになります。
same_key_for_alldgs	デフォルトでは、CVM はクラスタの共有ディスクグループのそれぞれに対して一意のフェンシングキーを生成します。一部のストレージアレイでは、登録できる一意のキーの合計数に上限があります。持続的な予約が使用されていると、CVM の上限に達する可能性があります。 このチューニングパラメータをオンに設定すると、CVM は作成またはインポートする共有ディスクグループに対して同じキーを生成します。チューニングパラメータを設定するときにすでにインポートされているディスクグループは、チューニングパラメータの値の変更が有効になる前に、デポートして再インポートする必要があります。 デフォルト値は off です。

sharedminorstart	共用(CVM)ディスクグループのデバイスのマイナー番号を割り当てるために使われる範囲の開始番号です。デフォルト値は 33000 です。
storage_connectivity	ストレージ切断に対する CVM の許容範囲を示す clusterwide チューニングパラメータです。 値が非対称の場合(デフォルト)、CVM を使って別のノードを通じてすべてのディスクにアクセスできるノードがクラスタに結合できるように設定できます。同様に、共有ディスクグループのディスクにアクセスできるノードが 1 つでもある場合は、CVM は共有ディスクグループをインポートできます。 値が[耐性がある(resilient)]である場合、CVM はノードを CVM クラスタに結合する前に、ノードが共有ディスクグループ内のすべてのディスクにアクセスできるようになっている必要があります。接続は共有ディスクグループのインポート前にも必要です。 どちらの設定でも、少なくとも 1 つのノードがディスクグループ設定にアクセスできれば、CVM はディスクグループ設定へのアクセスを処理します。

VVR のチューニングパラメータ

表 B-5 では、VVR のチューニングパラメータを一覧表示しています。

特に指定のないかぎり、vxtune コマンドまたはオペレーティングシステムのメソッドを使ってパラメータをチューニングできます。

表 B-5 VVR チューニングパラメータ

チューニングパラメータ名	説明
vol_cmpres_enabled	圧縮をグローバルに有効化または無効化する clusterwide チューニングパラメータ。圧縮はデフォルトでは無効になっているため、デフォルト値は 0 です。
vol_cmpres_threads	プライマリホストの圧縮スレッド数、またはセカンダリホストの圧縮解除スレッド数を 1 から 64 の間で設定できるようにするシステム単位のチューニングパラメータ。デフォルト値は 10 です。CPU 使用率に応じてこの設定を調整できます。

チューニングパラメータ名	説明
vol_dcm_replay_size	このチューニングパラメータは、vxtune コマンドを使って変更できません。 DCM 再生のブロックサイズ。デフォルト値は 256 KB です。
vol_max_nmpool_sz	ネットワークを経由してセカンダリに渡される要求の格納に使えるバッファ領域の大きさ。デフォルト値は 128MB です。
vol_max_rdback_sz	リードバックに使えるバッファ領域の大きさ。デフォルト値は 128 MB です。
vol_max_wrspool_sz	書き込み転送バッファ領域であり、非ログ所有者により送信された書き込み情報を受信するために、ログ所有者に割り当て可能なバッファ領域サイズ。デフォルト値は 64 MB です。
vol_min_lowmem_sz	最小のバッファ領域。VVR は、使用可能なバッファ領域がこのしきい値を下回ると、書き込みバッファを解放します。デフォルト値は 32MB です。 この値は自動調整が可能です。指定した値は初期値として使われ、アプリケーションの書き込み動作に応じて変更できます。
vol_nm_hb_timeout	ハートビートタイムアウト値。デフォルト値は 10 秒です。
vol_rvio_maxpool_sz	受信した書き込みを処理するためにオペレーティングシステム内に割り当てることのできるバッファ領域の大きさ。デフォルト値は 128 MB です。
vol_vvr_use_nat	このチューニングパラメータは、vxtune コマンドを使って変更できません。 このチューニングパラメータは、VVR が NAT ベースのファイアウォールを経由して通信できるように、受け取ったメッセージの変換済みのアドレスを使用するように VVR に指示します。構成内に NAT ベースのファイアウォールがある場合にのみ、このチューニングパラメータを 1 に設定します。

FSS 環境でのホットリロケーションのチューニングパラメータ

表 B-6 はFSS 環境でのホットリロケーションのチューニングパラメータを一覧表示します。Vxtune コマンドを使用してパラメータをチューニングできます。

表 B-6 FSS 環境でのホットリロケーションのチューニングパラメータ

チューニングパラメータ名	説明
storage_reloc_timeout	FSS 環境でホットリロケーションの開始前にストレージがオンラインになるまで VxVM が待機する時間を指定する、クラスタ全体のチューニングパラメータ。 デフォルト値は 30 分です。
node_reloc_timeout	FSS 環境でホットリロケーションの開始前にノードがオンラインになるまで VxVM が待機する時間を指定する、クラスタ全体のチューニングパラメータ。 デフォルト値は 120 分です。

VVR チューニングパラメータ値の変更時の注意点

チューニングパラメータ値を変更するときは次の点に注意してください。

- vol_rvio_maxpool_sz チューニングパラメータの値を減らす場合、ホスト上の RVG すべてを停止しなければなりません。
- チューニングパラメータ vol_max_rdback_sz と vol_max_nmpool_sz のサイズを縮小する場合、RLINK を停止する必要があります。

  メモ: vol_max_wrspool_sz の場合でも、RLINK を停止します。

- vol_min_lowmem_sz チューニングパラメータは自動調整が可能なため、受信した書き込みに応じて、VVR によってチューニングパラメータの値が増減されます。
  自動調整は、チューニングパラメータ vol_min_lowmem_sz でのみサポートされています。

共有ディスク環境では、各ホストに必要なチューニングパラメータのみを設定することもできます。ただし、現在使用されていないチューニングパラメータも、適切に設定することをお勧めします。これは、ログ所有者が変更された場合に、新しいログ所有者のチューニングパラメータが使用されるためです。次のチューニングパラメータは、ログ所有者のみに対して設定し、他のホストには設定する必要はありません。

- vol_max_rdback_sz
- vol_max_nmpool_sz
- vol_max_wrspool_sz
- vol_dcm_replay_size
- vol_nm_hb_timeout

- vol_vvr_use_nat

vxtune コマンドを使ってチューニングパラメータを変更した場合、その変更は、コマンドを実行しているホスト上のチューニングパラメータの値にのみ有効です。そのため、共有ディスクグループ環境では、チューニングパラメータの値を変更するホストごとに、それぞれコマンドを実行する必要があります。

Veritas Volume Manager のチューニングパラメータの変更方法

VxVM（Veritas Volume Manager）には、設定のチューニングで使うことができるさまざまなパラメータが用意されています。

p.1129 の「VxVM のチューニングパラメータ」を参照してください。

次のいずれかの方法で VxVM チューニングパラメータを変更します。

VxVM チューニングパラメータの値を表示または変更するには、vxtune コマンドを使用します。	p.1145 の「vxtune コマンドラインを使った Veritas Volume Manager チューニングパラメータの値の変更」を参照してください。
vxtune コマンドのテンプレート方式を使います。	p.1149 の「テンプレートを使った Veritas Volume Manager チューニングパラメータの値の変更」を参照してください。

vxtune コマンドラインを使った Veritas Volume Manager チューニングパラメータの値の変更

vxtune コマンドを使って、VxVM チューニングパラメータの値を表示または変更します。この変更は永続的であるため、値は以降の再ブート後も保持されます。新しい値を設定する前に、VxVM は値を検証して、そのチューニングパラメータの許容範囲内であることを確認します。値が有効であれば、VxVM はチューニングパラメータを更新します。一部のチューニングパラメータでは、変更した値を有効にする前に再ブートが必要です。VxVM は必要に応じてシステムの再ブートを要求します。

デフォルトでは、vxtuneコマンドは clusterwide チューニングパラメータを除いた、コマンドを実行しているホストのチューニングパラメータの値にのみ有効です。clusterwide 属性は、クラスタのすべてのノード上のチューニングパラメータの値を vxtune コマンドで設定することを指定します。チューニングパラメータが clusterwide でない場合、クラスタのすべてのノードのチューニングパラメータの値は -c オプションを使って変更します。-c オプションをスタンドアロンシステムで使用すると、操作は失敗します。

VxVM はチューニングパラメータの値を /etc/vx/vxtunables ファイルに格納します。

注意: vxtune コマンドを使ってチューニングパラメータの値を変更することをお勧めします。チューニングパラメータの値は、vxtunables ファイルで直接編集しないでください。

ほとんどのチューニングパラメータでは、K、M、Gという単位を示す接尾辞を付けてチューニングパラメータの値を指定します。単位を指定しない場合、vxtune はその値がバイト単位であると想定します。

メモ: 値を入力する場合のデフォルトの単位は、デフォルトの表示単位と異なる場合があります。

VxVM チューニングパラメータの値を変更するには

1 変更するチューニングパラメータの名前と現在の値を検索します。説明を表示するには、-l オプションを使います。

```
# vxtune -l
```

次の例は、切り捨てられた出力を示していて、形式を示しています。

```
Tunable              Current Value  Default Value  Reboot  Clusterwide  Description
-------------------  -------------  -------------  ------  -----------  -----------
vol_checkpt_default          20480          20480       Y            N  Size of VxVM
                                                                        checkpoints (sectors)
vol_cmpres_enabled               0              0       N            Y  Allow enabling
                                                                           compression for
VERITAS
                                                                           Volume Replicator
vol_cmpres_threads              10             10       N            N  Maximum number of
                                                                          compression threads

                                                                           for VERITAS
                                                                           Volume Replicator
vol_default_iodelay             50             50       Y            N  Time to pause between

                                                                           I/O requests from
VxVM
                                                                           utilities (10ms
units)
vol_fmr_logsz                    4              4       Y            N  Maximum size of
bitmap
                                                                           Fast Mirror Resync

                                                                        uses to track changed

                                                                           blocks (KBytes)
vol_max_adminio_poolsz    67108864       67108864       Y            N  Maximum amount of
                                                                           memory used by
                                                                           VxVM admin IO's
(bytes)
.
.
.
```

出力には、デフォルト値と現在の値が表示されます。再ブートフィールドは、チューニングパラメータの値が有効になる前に再ブートが必要かどうかを示します。

Clusterwide フィールドは vxtune がクラスタのすべてのノードに値をデフォルトで適用するかどうかを示します。

vxtune（1M）のマニュアルページを参照してください。

2 それぞれのチューニングパラメータに新しい値を設定します。K、M、G という単位を示す接尾辞を付けて値を指定します。単位を指定しない場合、vxtune コマンドはチューニングパラメータのデフォルトの単位を使います。ほとんどのチューニングパラメータでは、デフォルト値はバイトです。vxtune 出力の説明には、各チューニングパラメータのデフォルトの単位が表示されます。

```
# vxtune [-C] tunable_nametunable_value
```

たとえば、vol_cmpres_enabled の値を 1 に変更するには、次のコマンドを使います。

```
# vxtune vol_cmpres_enabled 1
```

指定したチューニングパラメータが clusterwide でない場合、クラスタのすべてのノードのチューニングパラメータの値は -c オプションを使って設定します。

3 新しい値を確認します。

```
# vxtune tunable_name
```

たとえば、vol_cmpres_enabled について変更された値を表示するには、次のコマンドを使います。

```
# vxtune vol_cmpres_enabled
Tunable              Current Value  Default Value  Reboot
-------------------- -------------- -------------- ------
vol_cmpres_enabled   1              0              N
```

vol_cmpres_enabled チューニングパラメータが clusterwide のため、vxtune コマンドはクラスタのすべてのノードの値を変更しました。

テンプレートを使った Veritas Volume Manager チューニングパラメータの値の変更

VxVM には、チューニングパラメータを変更するためのテンプレートメソッドがあります。このメソッドを使って、チューニングパラメータをファイルにエクスポートし、ファイルを変更してからそのファイルをインポートします。チューニングパラメータは、厳密にそのエクスポートで指定された形式である必要があります。形式が異なっている場合、特定の値は破棄されることになります。

テンプレートを使って VxVM チューニングパラメータの値を変更するには

1 チューニングパラメータとその値をチューニングテンプレートファイルにエクスポートします。すべてのチューニングパラメータをエクスポートすることも、コンポーネントを指定することもできます。

```
# vxtune -o export file=file_name [component]
```

次に例を示します。

```
# vxtune -o export file=vxvm-tunables
```

```
# vxtune -o export file=vvr-tunables vvr
```

2 必要に応じてテンプレートを変更します。エクスポート操作で指定されたファイル形式を保持する必要があります。

3 チューニングテンプレートファイルをシステムにインポートします。インポート操作は、有効な値にのみ適用されます。特定のパラメータについて値が有効でない場合、その特定の値は破棄されます。

```
# vxtune -o import file=file_name
```

次に例を示します。

```
# vxtune -o import file=vxvm-tunables
```

LLT のチューニングパラメータについて

LLT には、LLT モジュールの動作を変更し、制御するための、さまざまな設定とチューニングパラメータがあります。ここでは、実行時と LLT の起動時に変更できる LLT のチューニングパラメータの一部について説明します。

チューニングパラメータは以下の 2 つのカテゴリに分類されます。

- LLT タイマーチューニングパラメータ
  p.1151 の「LLT タイマーチューニングパラメータについて」を参照してください。
- LLT フロー制御チューニングパラメータ
  p.1155 の「LLT フロー制御チューニングパラメータについて」を参照してください。

p.1158 の「LLT タイマーチューニングパラメータの設定」を参照してください。

LLT タイマーチューニングパラメータについて

表 B-7 に、LLT タイマーチューニングパラメータのリストを示します。タイマー値は .01 秒単位で設定します。現在のタイマー値を表示するには、コマンド `lltconfig -T query` を使用します。

表 B-7 LLT タイマーチューニングパラメータ

LLT パラメータ	説明	デフォルト	変更のタイミング	他の LLT チューニングパラメータとの依存関係
peerinact	LLT は、このタイマー間隔だけピアノードのリンクでパケットを受信しないと、そのリンクを「非アクティブ」と指定します。リンクが「非アクティブ」に指定されると、LLT はそのリンクでデータを送信しません。	1600	■ クライアントの通知処理ロジックに従ってノード/リンク非アクティブ通知メカニズムを遅く、または速くする場合に、この値を変更します。 ■ 障害のあるネットワークケーブル/スイッチの計画的な交換の場合は、値を大きくします。 ■ プライベートネットワークリンクが非常に遅いとき、またはネットワークトラフィックのバースト状態が非常に高くなったときは、ピアの終了が誤って通知されることを避けるためにこの値を大きくします。 障害のあるネットワークケーブルまたは障害のあるスイッチの計画的な交換の場合は、高い値に設定します。	このタイマーの値は、peertrouble タイマーの値より常に高くする必要があります。

LLT パラメータ	説明	デフォルト	変更のタイミング	他の LLT チューニングパラメータとの依存関係
rpeerinact	ノードがこのタイマー間隔だけ RDMA リンクでパケットを受信しないと、RDMA リンクの RDMA チャネルを「非アクティブ」と指定します。RDMA チャネルが「非アクティブ」と指定されると、LLT がそのリンクの RDMA チャネルにデータを送信することはなくなります。ただし peerinact が期限切れになるまでそのリンクの非 RDMA チャネルにデータを送信し続ける場合があります。lltstat -nvv -r コマンドを使って、RDMA リンクの RDMA チャネルの状態を表示できます。このパラメータは特定のバージョンの Linux でのみサポートされます。	700	RDMA リンクのエラーからのリカバリを早めるには、この調整可能な値を小さくします。リンクが不安定で、頻繁にアップまたはダウンするようであれば、この値を小さくしないでください。	このタイマー値は必ず peertrouble タイマーの値よりも大きくし、peerinact の値未満にする必要があります。
peertrouble	LLT は、このタイマー間隔だけピアノードの高優先度リンクでパケットを受信しないと、そのリンクを「問題発生」と指定します。リンクが「問題発生」に指定されると、LLT はリンクが起動するまでそのリンクでデータを送信しません。	200	■ プライベートネットワークリンクが非常に遅いとき、またはクラスタのノードが非常にビジー状態であるときは、この値を大きくします。 ■ 障害のあるネットワークケーブル/障害のあるスイッチの計画的な交換の場合は、値を大きくします。	このタイマーの値は、peerinact タイマーの値より常に低くする必要があります。また、デフォルト値に近くしておく必要があります。
peertroublelo	LLT は、このタイマー間隔だけピアノードの低優先度リンクでパケットを受信しないと、そのリンクを「問題発生」と指定します。リンクが「問題発生」に指定されると、LLT はリンクが使用できるようになるまでそのリンクでデータを送信しません。	400	■ プライベートネットワークリンクが非常に遅いとき、またはクラスタのノードが非常にビジー状態であるときは、この値を大きくします。 ■ 障害のあるネットワークケーブル/障害のあるスイッチの計画的な交換の場合は、値を大きくします。	このタイマーの値は、peerinact タイマーの値より常に低くする必要があります。また、デフォルト値に近くしておく必要があります。

LLT パラメータ	説明	デフォルト	変更のタイミング	他の LLT チューニングパラメータとの依存関係
heartbeat	LLT は、各高優先度リンクで heartbeat タイマー間隔が経過するたびに、ピアノードに対して繰り返しハートビートパケットを送信します。	50	プライベートネットワークリンクが非常に遅い（または輻輳している）とき、またはクラスタのノードが非常にビジー状態であるときは、この値を大きくします。	このタイマーの値は、peertrouble タイマーの値より低くする必要があります。また、peertrouble タイマーの値に近くしないようにする必要があります。
heartbeatlo	LLT は、各低優先度リンクで heartbeatlo タイマー間隔が経過するたびにピアノードに対して繰り返しハートビートパケットを送信します。	100	ネットワークリンクが非常に遅いとき、またはクラスタのノードが非常にビジー状態であるときは、この値を大きくします。	このタイマーの値は、peertroublelo タイマーの値より低くする必要があります。また、peertroublelo タイマーの値に近くしないようにする必要があります。
timetoreqhb	LLT が「timetoreqhb」で指定された期間特定のリンクのピアノードからパケットを受信しない場合、LLT は、そのピアノードに対するハートビート要求（同じリンクのピアノードに 5 つの特別なハートビート要求（hbreqs）を送信）を試みます。ピアノードが特別なハートビート要求に応答しない場合、LLT はそのピアノードに対するこのリンクを「期限切れ」と指定します。値は、0 から（peerinact -200）までの範囲で設定できます。値 0 は、ハートビート要求メカニズムを無効にします。	1400	クライアントの通知処理ロジックに従ってノード/リンク非アクティブ通知メカニズムを速くするには、このチューニングパラメータの値を小さくします。 障害のあるネットワークケーブルやスイッチの計画的な交換の場合は、このタイマーを 0 に設定することで、ハートビート要求メカニズムを無効にします。 プライベートネットワークリンクが非常に遅いとき、またはネットワークトラフィックのバースト状態が非常に高くなったときは、このタイマーチューニングパラメータの値を変更しないでください。	このタイマーは、peerinact タイマーが変更されるたびに、「peerinact - 200」に自動的に設定されます。
reqhbtime	この値は、連続する 2 つの特別なハートビート要求の時間間隔を指定します。特別なハートビート要求について詳しくは、timetoreqhb パラメータを参照してください。	40	この値は変更しないことを推奨します。	適用不可能

LLT パラメータ	説明	デフォルト	変更のタイミング	他の LLT チューニングパラメータとの依存関係
timetosendhb	LLT は、LLT タイマーが一定の間隔で実行しないとき、ノードの稼働を保つためにタイマー切れコンテキストのハートビートを送信します。このオプションは、タイマーが動作していない場合にハートビートを送信する前に待つ時間を指定します。 このタイマーチューニングパラメータを 0 に設定すると、タイマー切れコンテキストのハートビートメカニズムは無効になります。	200	障害のあるネットワークケーブル/スイッチの計画的な交換の場合は、このタイマーを 0 に設定することで、タイマー切れコンテキストのハートビートメカニズムを無効にします。 プライベートネットワークリンクが非常に遅いとき、またはクラスタのノードが非常にビジー状態であるときは、この値を大きくします。	このタイマーの値は、peerinact タイマーの値以下にする必要があります。また、peerinact タイマー値の近くにしない必要があります。
sendhbcap	この値は LLT がタイマーコンテキストハートビートを連続して送信する最大時間を指定します。	18000	この値は変更しないことを推奨します。	該当なし
oos	順不同タイマーがノードで期限切れになった場合、LLT はそのノードに適切な NAK を送信します。LLT は oos パケットを受信してすぐには NAK を送信しません。NAK を送信する前に oos タイマーの値だけ待ちます。	10	パフォーマンス上の理由からこの値を変更しないでください。値を下げると、不必要な再送信/NAK トラフィックが発生する可能性があります。 クラスタ(たとえば、キャンパスクラスタ)でラウンドトリップ時間が大きい場合、oos の値を増加できます。	適用不可能
retrans	LLT はこのタイマー間隔の値の間に肯定応答を受信しなければパケットを再送信します。	10	この値は変更しないでください。値を下げると不必要な再送信が発生する可能性があります。 クラスタ(たとえば、キャンパスクラスタ)でラウンドトリップ時間が大きい場合、retrans の値を増加できます。	適用不可能
service	LLT は、service タイマー間隔が経過するたびにサービスルーチン(LLT クライアントにメッセージを配信します)を呼び出します。	100	パフォーマンス上の理由からこの値を変更しないでください。	適用不可能

LLT パラメータ	説明	デフォルト	変更のタイミング	他の LLT チューニングパラメータとの依存関係
arp	このタイマーが期限切れになると、LLT は格納されているピアノードのアドレスをフラッシュし、アドレスを再設定します。	0	この機能はデフォルトでは無効です。	適用不可能
arpreq	このタイマーが期限切れになると、LLT は arp 要求を送信し、クラスタの他のピアノードを検出します。	3000	パフォーマンス上の理由からこの値を変更しないでください。	適用不可能

LLT フロー制御チューニングパラメータについて

表 B-8 に、LLT フロー制御チューニングパラメータのリストを示します。フロー制御の値はパケット数で設定します。 `lltconfig -F query` コマンドを使って、現在のフロー制御の設定を表示できます。

表 B-8 LLT フロー制御チューニングパラメータ

LLT パラメータ	説明	デフォルト	変更のタイミング	他の LLT チューニングパラメータとの依存関係
highwater	ノードの送信キュー内のパケット数が highwater に達すると、LLT はフロー制御されます。	200	クライアントがバースト方式でデータを生成する場合は、着信データ速度に合わせてこの値を増やします。値を大きくするとメモリの消費量が増えるので、不必要にメモリを浪費しないように適切な値を設定してください。 値を小さくすると、クライアントを不必要にフロー制御する可能性があります。	このフロー制御値は lowwater フロー制御値より常に高くする必要があります。
lowwater	LLT がクライアントのフロー制御を行っているときは、ノードに対するポート送信キュー内のパケット数が lowwater に低下するまで、パケットの受け付けを再開しません。	100	ベリタスでは、このチューニングパラメータを変更することは推奨しません。	このフロー制御値は highwater フロー制御値より低くする必要があります。 highwater フロー制御値に近い値を設定しないようにする必要があります。

LLT パラメータ	説明	デフォルト	変更のタイミング	他の LLT チューニングパラメータとの依存関係
rporthighwater	ポートの受信キュー内のパケット数が highwater に達すると、LLT はフロー制御されます。	200	クライアントがバースト方式でデータを生成する場合は、着信データ速度に合わせてこの値を増やします。値を大きくするとメモリの消費量が増えるので、不必要にメモリを浪費しないように適切な値を設定してください。 値を小さくすると、ピアノード上のクライアントを不必要にフロー制御する可能性があります。	このフロー制御値は rportlowwater フロー制御値より常に高くする必要があります。
rportlowwater	LLT がピアノード上のクライアントのフロー制御を行っているときは、ポートに対するポート受信キュー内のパケット数が rportlowwater に低下するまで、そのクライアントに対するパケットの受け付けを再開しません。	100	ベリタスでは、このチューニングパラメータを変更することは推奨しません。	このフロー制御値は rpothighwater フロー制御値より低くする必要があります。rporthighwater フロー制御値に近い値を設定しないようにする必要があります。

LLT パラメータ	説明	デフォルト	変更のタイミング	他の LLT チューニングパラメータとの依存関係
window	LLT が ACK を受信せずに送信するパケットの最大数です。	50	パフォーマンス上の理由により、デフォルトでは、ポート 5（cfs）およびポート 24（cvm）でアダプティブウィンドウ機能が有効になっています。他のポートについては、/etc/sysconfig/llt ファイルの LLT_AW_PORT_LIST パラメータの値を変更することで、アダプティブウィンドウを手動で有効にできます。 ■ アダプティブウィンドウを無効にするには、LLT_ENABLE_AWINDOW パラメータの値をゼロに設定します。 ■ 5 および 24 以外のポートでアダプティブウィンドウを有効にするには、LLT_AW_PORT_LIST にポート番号をカンマで区切って追加します。たとえば、LLT_AW_PORT_LIST=:"5,24,0,1,5,14" のように指定します。 ■ LLT_AW_PORT_LIST パラメータにリストされている以外のポートのウィンドウサイズを設定する場合は、プライベートネットワーク速度に合わせて window パラメータの値を変更します。例: set-flow window: 2000 プライベートネットワークの速度に応じて値を変更します。ネットワークの速度に関係なく値を下げると、ウィンドウ順序が異なるパケットの不必要な再送信が発生する可能性があります。	このフロー制御値は、highwater フロー制御値と lowwater フロー制御値の差よりより大きくしないようにする必要があります。 このパラメータ（window）の値は帯域幅を遅延させる製品の値に合わせる必要があります。

LLT パラメータ	説明	デフォルト	変更のタイミング	他の LLT チューニングパラメータとの依存関係
linkburst	これは、次のリンクが選択される前に LLT がリンク上で送信するバックツーバックパケットの数を表します。	32	パフォーマンス上の理由から、値は 0 または少なくとも 32 にする必要があります。	このフロー制御値は、highwater フロー制御値と lowwater フロー制御値の差よりより大きくしないようにする必要があります。
ackval	LLT は、次のアウトバウンドデータパケットの ACK パケットに便乗することでパケットの肯定応答を送信者ノードに送信します。ACK パケットを便乗させるデータパケットがない場合、LLT は送信者に明示的な ACK を送信する前に ackval 個のパケットを待ちます。	10	パフォーマンス上の理由からこの値を変更しないでください。値を大きくすると不必要な再送信が発生する可能性があります。	適用不可能
sws	Silly Window Syndrome を避けるため、LLT は ACK 未受信パケットの数がこのチューニングパラメータの値より低下した場合にのみ、さらにパケットを伝送します。	40	パフォーマンスの理由のため、window チューニングパラメータの値が変更されたときは常に、次の式に従ってこのパラメータの値を変更する必要があります（sws = window *4/5）。	このパラメータの値は window の値より小さくする必要があります。このパラメータの値は window チューニングパラメータの値に近くする必要があります。
largepktlen	パケットを複数のポートに送信する場合、LLT は 1 つのポートに対して一度に大きいパケットを 1 つ、または小さいパケットを 5 つまで送信します。このパラメータは大きいパケットのサイズを指定します。	1024	ベリタスでは、このチューニングパラメータを変更することは推奨しません。	適用不可能

LLT タイマーチューニングパラメータの設定

LLT のチューニングパラメータは、`lltconfig` コマンドまたは `/etc/llttab` ファイルで設定できます。`lltconfig` コマンドを使用すると、実行時にローカルノードのパラメータを変更できます。ベリタスはクラスタのすべてのノードでコマンドを実行してパラメータの値を変更することを推奨します。システムの再ブート時に LLT パラメータを設定するには、`/etc/llttab` ファイルにパラメータ定義を含める必要があります。/etc/lltttab で何も指定されていない場合は、パラメータのデフォルト値が使われます。/etc/lltttab ファイルで指定されているパラメータ値は、LLT の起動時にのみ有効になります。ベリタスは各ノードの `/etc/llttab` ファイルでチューニングパラメータの同じ定義を指定することを推奨します。

タイマーチューニングパラメータを取得して設定するには:

- lltconfig コマンドを使ってタイマーのチューニングパラメータの現在の一覧を取得するには:

```
# lltconfig -T query
```

- lltconfig コマンドを使ってタイマーチューニングパラメータを設定するには:

```
# lltconfig -T timer tunable:value
```

- /etc/lltab ファイルでタイマーチューニングパラメータを設定するには:

```
set-timer  timer tunable:value
```

フロー制御チューニングパラメータを取得して設定するには

- lltconfig コマンドを使ってフロー制御チューニングパラメータの現在の一覧を取得するには:

```
# lltconfig -F query
```

- lltconfig コマンドを使ってフロー制御チューニングパラメータを設定するには:

```
# lltconfig -F flowcontrol tunable:value
```

- /etc/lltab ファイルでフロー制御チューニングパラメータを設定するには:

```
set-flow  flowcontrol tunable:value
```

lltconfig(1M)マニュアルページと lltab(1M)マニュアルページを参照してください。

GAB のチューニングパラメータについて

GAB には、GAB モジュールの動作を変更し、制御するための、さまざまな設定とチューニングパラメータがあります。

チューニングパラメータは、クラスタ内のノードの最大許容数などの設定を制御するだけでなく、GAB が障害または障害状態を検出したときの GAB の動作方法も制御します。これらのチューニングパラメータのいくつかは、GAB モジュールがシステムにロードされるときに必要です。このようなロード時チューニングパラメータを変更したときは、GAB モジュールのアンロードと再ロード、またはシステムの再ブートが必要です。他のチューニングパラメータ(実行時)は、GAB モジュールがロードされ、構成されて、クラスタが動作している間でも変更できます。このようなチューニングパラメータへの変更は、チューニングパラメータの値と GAB の動作に即時に反映されます。

これらのチューニングパラメータは次のファイルで定義されています。

/etc/sysconfig/gab

p.1160 の「GAB のロード時チューニングパラメータまたは静的チューニングパラメータについて」を参照してください。

p.1161 の「GAB の実行時または動的チューニングパラメータ」を参照してください。

GAB のロード時チューニングパラメータまたは静的チューニングパラメータについて

表 B-9 は、モジュールのロード時に使われる GAB の静的なチューニングパラメータのリストです。このような GAB のチューニングパラメータをすべて表示するには、gabconfig -e コマンドを使います。

これらのチューニングパラメータを変更するには、GAB 設定ファイルに新しい値を追加する必要があります。変更が反映されるのは、再ブート時または GAB モジュールの再ロード時だけです。

表 B-9 GAB の静的チューニングパラメータ

GAB パラメータ	説明	値（デフォルトと範囲）
numnids	クラスタのノードの最大数	デフォルト: 128 範囲: 1 から 128
numports	クラスタのポートの最大数	デフォルト: 32 範囲: 1 から 32
flowctrl	GAB がフロー制御をヒットするまでの、GAB キュー（送信または受信）内の保留中メッセージの数。 この値は、クラスタが動作中の間に gabconfig -Q オプションで上書きできます。このチューニングパラメータの値を制御するには gabconfig コマンドを使います。	デフォルト: 128 範囲: 1 から 1024
logbufsize	GAB の内部ログバッファのサイズ（バイト単位）。	デフォルト: 48100 範囲: 8100 から 65400

GAB パラメータ	説明	値（デフォルトと範囲）
msglogsize	内部メッセージログの最大メッセージ。	デフォルト: 256 範囲: 128 から 4096
isolate_time	分離されたクライアントを待つ最大時間。 実行時に上書きできます。 p.1161 の「GAB の実行時または動的チューニングパラメータ」を参照してください。	デフォルト: 120000 ミリ秒（2 分） 範囲: 160000 から 24000（ミリ秒単位）
kill_ntries	クライアントの強制終了を試みる回数。 実行時に上書きできます。 p.1161 の「GAB の実行時または動的チューニングパラメータ」を参照してください。	デフォルト: 5 範囲: 3 から 10
conn_wait	クラスタの再設定中に、GAB がクラスタからノードを接続解除するまでの待ち時間の最大数（stable timeout パラメータで定義）	デフォルト: 12 範囲: 1 から 256
ibuf_count	GAB のログ記録デーモンが有効と無効かどうか決定します。 GAB のログ記録デーモンはデフォルトでは有効になります。無効にするには、gab_ibuf_count の値を 0 に変更します。 クラスタが起動し、動作している間に、`gabconfig -K` オプションを使って GAB デーモンへのログインを無効にします。このチューニングパラメータの値を制御するには `gabconfig` コマンドを使います。	デフォルト: 8 範囲: 0 から 32
kstat_size	GAB で保持するシステム統計の数。	デフォルト: 60 範囲: 0 から 240

GAB の実行時または動的チューニングパラメータ

GAB の動的なチューニングパラメータは、GAB が設定されている間およびクラスタが動作している間でも変更できます。変更は `gabconfig` コマンドを実行することですぐに反映されます。これらのパラメータの一部は障害または障害状態が検出されたときの GAB の動作も制御することに注意してください。このような状態の一部では、データ破損を防ぐために PANIC がトリガされる場合があります。

gabconfig -l コマンドを使ってデフォルト値を表示できます。これらの値の変更が再ブート時にも保持されるようにするには、適切なコマンドオプションを既存のオプションと共に /etc/gabtab ファイルに追加できます。たとえば、次のような既存の /etc/gabtab ファイルに -k オプションを追加できます。

```
gabconfig -c -n4
```

オプションを追加した後の /etc/gabtab ファイルは次のようになります。

```
gabconfig -c -n4 -k
```

表 B-10 では、gabconfig -l コマンドで表示される GAB の動的チューニングパラメータとその変更コマンドについて説明します。

表 B-10 GAB の動的チューニングパラメータ

GAB パラメータ	説明とコマンド
Control port seed	このオプションはクラスタを形成できるノードの最小数を定義します。このオプションはクラスタの形成を制御します。クラスタのノードの数が gabtab ファイルで指定されている数より少ない場合、クラスタは形成されません。たとえば、gabconfig -c -n4 と入力すると、クラスタは 4 つのノードがすべてクラスタに参加するまで形成されません。gabconfig -x コマンドを使ってこのオプションを有効にした場合、クラスタの他のノードがまだメンバーシップの一部になっていなくても、そのノードはクラスタに参加します。 クラスタを形成できるノード数を設定するには、次のコマンドを使います。 gabconfig -n *count* 制御ポートシードを有効にするには、次のコマンドを使います。ノードは他のノードのメンバーシップを待たずにクラスタを形成できます。 gabconfig -x

GAB パラメータ	説明とコマンド

Halt on process death	デフォルト: 無効
	このオプションは、ユーザープロセスが終了したときに GAB がシステムを一時停止(パニック)させる機能を制御します。_had と _hashadow が `kill -9` を使用して強制終了された場合、システムは高可用性を失う可能性があります。このオプションを有効にした場合、GAB はクライアントプロセスの終了を検出するとシステムを PANIC にします。デフォルトの動作では、このオプションは無効になります。
	プロセス終了時のシステムの一時停止を有効にするには、次のコマンドを使います。
	`gabconfig -p`
	プロセス終了時のシステムの一時停止を無効にするには、次のコマンドを使います。
	`gabconfig -P`

Missed heartbeat halt	デフォルト: 無効
	このオプションを有効にすると、システムでは CVM 環境の VCS エンジンまたは `vxconfigd` デーモンからの最初のハートビートが見つからないためにパニックが発生します。デフォルトのオプションでは即時パニックは無効です。
	この GAB オプションは、VCS エンジンまたは vxconfigd デーモンが GAB にハートビートを送信しない場合に、GAB がノードをパニックにできるかどうかを制御します。VCS エンジンがハングして GAB にハートビートを送信できない場合、GAB はすぐにはシステムをパニックにしません。GAB は、最初に、iofence_timeout(デフォルト値は 15 秒)の間隔で kill_ntries(デフォルト値は 5 回)回だけ SIGABRT を送信することで、プロセスの中止を試みます。これに失敗すると、GAB は isolate_time グローバルチューニングパラメータ(デフォルト値 2 分)で制御される分離タイムアウト期間だけ待ちます。プロセスがそれでも稼働している場合、GAB はシステムを PANIC にします。
	このオプションを有効にすると、GAB はクライアントからのハートビートを受け取らないと即座にシステムを一時停止します。
	プロセスハートビートが失敗したときに、システム停止を有効にするには、次のコマンドを使います。
	`gabconfig -b`
	プロセスハートビートが失敗したときに、システム停止を無効にするには、次のコマンドを使います。
	`gabconfig -B`

GAB パラメータ	説明とコマンド

Halt on rejoin	デフォルト: 無効
	
	このオプションを使用すると、ネットワーク分割の後で 1 つ以上のノードがクラスタに再参加するときの VCS エンジンまたはその他のユーザープロセスの動作を設定できます。デフォルトでは、GAB はVCS エンジンを実行するノードを PANIC にしません。GAB は userland プロセス(VCS エンジンまたは vxconfigd プロセス)を強制終了します。これにより、ユーザーポート(VCS エンジンの場合ポート h)がリサイクルされ、古い世代番号のメッセージがプログラムでクリーンアップされます。プロセスの再起動が必要な場合は、GAB の制御の範囲外で処理する必要があります(hashadow プロセスが _had を再起動する場合など)。
	
	GAB にカーネルクライアント(フェンシング、VxVM、VxFS など)がある場合、ネットワーク分割の後でクラスタに再参加するときにノードは常に PANIC になります。これは GAB がポートを消去して古いメッセージを削除できる唯一の方法なので、PANIC は必須です。
	
	再参加時にシステムの一時停止を有効にするには、次のコマンドを使います。
	
	`gabconfig -j`
	
	再参加時にシステムの一時停止を無効にするには、次のコマンドを使います。
	
	`gabconfig -J`

Keep on killing	デフォルト: 無効 このオプションを有効にした場合、VCS エンジンまたは vxconfigd プロセスが GAB とのハートビートに失敗し、GAB が VCS エンジンまたは vxconfigd プロセスの強制終了に失敗したときに、GAB はシステムが PANICKING になるのを防ぎます。GAB は、継続的に VCS エンジンの強制終了を試み、強制終了が失敗した場合でもパニックに陥りません。 プロセスが終了しない場合にプロセスを強制終了する試行を繰り返します。 `gabconfig -k`

GAB パラメータ	説明とコマンド

Quorum flag	デフォルト: 無効
	GAB のこのオプションを使うと、新しいメンバーシップセットが古いメンバーシップセットの 50% 未満の場合に、ノードは IOFENCE に(結果として PANIC に)なることができます。このオプションは通常は無効にしておき、他の製品と統合するときに使います。
	iofence クォーラムを有効にします。
	`gabconfig -q`
	iofence クォーラムを無効にします。
	`gabconfig -d`

GAB queue limit	デフォルト: 送信キュー制限: 128
	デフォルト: 受信キュー制限: 128
	GAB キュー制限オプションは、GAB がフローを設定するようになる前の保留中メッセージの数を制御します。送信キュー制限は、GAB 送信キューの保留中メッセージの数を制御します。GAB がこの制限に達すると、GAB クライアントの送信プロセスのフロー制御を設定します。GAB 受信キュー制限は、GAB が受信側にフロー制御を送信する前の、GAB 受信キュー内の保留中メッセージの数を制御します。
	指定した値に送信キュー制限を設定します。
	`gabconfig -Q sendq:value`
	指定した値に受信キュー制限を設定します。
	`gabconfig -Q recvq:value`

IOFENCE timeout	デフォルト: 15000(ミリ秒) このパラメータは、GAB が次の処理を実行する前に IOFENCE メッセージに対するクライアントからの応答を待つタイムアウト(ミリ秒)を指定します。kill_ntries の値に基づいて、GAB は SIGABRT 信号を送信することによってクライアントプロセスの強制終了を試みます。GAB が kill_ntries の回数だけクライアントプロセスの強制終了を試みた後もクライアントプロセスがまだ登録している場合、GAB はさらに isolate_timeout の値だけ待ってからシステムを一時停止します。 指定したミリ秒値に iofence タイムアウトの値を設定します。 `gabconfig -f value`

GAB パラメータ	説明とコマンド
Stable timeout	デフォルト: 5000（ミリ秒） 特定のポートのローカルノード接続の状態変化が LLT から最後に報告された後で GAB がメンバーシップの再設定を待機する時間を指定します。接続の状態に何らかの変化があると、GAB は待ち時間を再起動します。 指定した値に安定タイムアウトを設定します。 `gabconfig -t stable`
Isolate timeout	デフォルト: 120000（ミリ秒） このチューニングパラメータは、GAB が送信した SIGKILL 信号に応答してクライアントプロセスが登録を解除するのを GAB が待機するタイムアウト値を指定します。分離タイムアウトの後もプロセスがまだ存在する場合、GAB はシステムを一時停止します。 `gabconfig -S isolate_time:value`
Kill_ntries	デフォルト: 5 このチューニングパラメータは、GAB が SIGABRT 信号を送信することによってプロセスの強制終了を試みる回数を指定します。 `gabconfig -S kill_ntries:value`
Driver state	このパラメータは GAB が設定されているかどうか示します。GAB はまだメンバーシップをシーディングして形成していない場合があります。
Partition arbitration	このパラメータは、GAB が JEOPARDY を無視するように要求されているかどうか示します。 `gabconfig`（1M）マニュアルの `-s` フラグに関する詳細のページを参照してください。

チューニングパラメータ VXFEN について

この項では、VXFEN のチューニングパラメータの概念と、VXFEN モジュールを再設定する方法について説明します。

表 B-11 は、VXFEN ドライバのチューニングパラメータについて説明しています。

表 B-11 VXFEN のチューニングパラメータ

vxfen のパラメータ	説明と値: デフォルト、最小値、最大値
dbg_log_size	デバッグログのバイト単位のサイズです。 ■ 値 デフォルト: 524288（512 KB） 最小値: 65536（64 KB） 最大値: 1048576 (1MB)
vxfen_max_delay	ネットワーク分割が発生した場合により大きいサブクラスタとコーディネータディスクの制御をめぐって競合するまでに小さい方のサブクラスタが待機する最大時間を秒数で指定します。 この値は、vxfen_min_delay の値よりも大きくする必要があります。 ■ 値 デフォルト: 60 最小値: 1 最大値: 600
vxfen_min_delay	ネットワーク分割が発生した場合により大きいサブクラスタとコーディネータディスクの制御をめぐって競合するまでに小さい方のサブクラスタが待機する最小時間を秒数で指定します。 この値は、vxfen_max_delay の値以下にする必要があります。 ■ 値 デフォルト: 1 最小値: 1 最大値: 600
vxfen_vxfnd_tmt	I/O フェンシングドライバ VxFEN が、指定のタスクの完了後に I/O フェンシングデーモン VXFEND が戻ってくるのを待つ時間を秒数で指定します。 ■ 値 デフォルト: 60 最小値: 10 最大値: 600

vxfen のパラメータ	説明と値: デフォルト、最小値、最大値
panic_timeout_offst	スプリットブレインが発生したときに GAB が意思決定を実装するまで待機する時間を、フェンシングがアービトレーションを完了するまで待機する GAB モジュールに渡すために I/O フェンシングドライバ VxFEN が計算する遅延に基づいて、秒数で指定します。vxfenmode ファイル内でこのパラメータを設定し、vxfenadm コマンドを実行して値を調べることができます。vxfen_mode に基づいて、GAB 遅延は次のように計算されます。 ■ scsi3 モードの場合: 1000 * (panic_timeout_offst + vxfen_max_delay) ■ カスタマイズモードの場合: 1000 * (panic_timeout_offst + max (vxfen_vxfnd_tmt, vxfen_loser_exit_delay)) ■ デフォルト: 10

ネットワーク分割イベントの場合、値がより小さいサブクラスタではコーディネータディスクの獲得を競う前に遅延が発生します。この遅延時間を指定することにより、値がより大きいサブクラスタがコーディネータディスクを獲得できるようになります。vxfen_max_delay および vxfen_min_delay の各パラメータは、遅延を秒単位で定義します。

VXFEN モジュールパラメータの設定

カーネルドライバのチューニングパラメータを調整した後、パラメータの変更を有効にするには VXFEN モジュールを再設定する必要があります。

次の手順例を実行すると、vxfen_min_delay パラメータの値が変化します。

各 Linux ノードで、/etc/sysconfig/vxfen ファイルを編集し、vxfen ドライバのチューニンググローバルパラメータである vxfen_max_delay と vxfen_min_delay の値を変更します。

メモ: パラメータ変更を有効にするには、VXFEN モジュールを再起動する必要があります。

VxFEN パラメータを設定し、Vxfen モジュールを再設定するには

1. VCS 下で設定されないすべてのアプリケーションを停止します。アプリケーションを停止するにはネイティブのアプリケーションコマンドを使います。
2. すべてのノード上で VCS を停止します。各ノードで次のコマンドを実行します。

```
# hastop -local
```

3. VxFEN ドライバを停止します。

```
# systemctl stop vxfen
```

4 /etc/sysconfig/vxfen ファイルを編集します。

たとえば、次のエントリがあります。

```
vxfen_min_delay=1
```

このエントリを次のように変更します。

```
vxfen_min_delay=30
```

5 VXFEN モジュールを起動します。

```
# systemctl start vxfen
```

6 VCS 下で設定されていないすべてのアプリケーションを起動します。アプリケーションを起動するには、ネイティブのアプリケーションコマンドを使います。

7 VCS を起動します。

```
#  hastart
```

AMF チューニングパラメータについて

次のコマンドを使ってAMF（Asynchronous Monitoring Framework）カーネルモジュールのチューニングパラメータを設定できます。

```
# amfconfig -T tunable_name=tunable_value,
tunable_name=tunable_value...
```

表 B-12 は AMF カーネルのチューニング可能パラメータの一覧です。

表 B-12 AMF チューニングパラメータ

AMF のパラメータ	説明	値
dbglogsz	AMF はメモリ内デバッグログを保持します。このパラメータ（KB 単位で指定）は、このログ用に割り当てられるカーネルメモリの量を制御します。	最小: 4 最大: 512 デフォルト: 256
processhashsz	AMF は、登録されたイベントを、イベントタイプに固有なハッシュテーブルに保存します。このパラメータは、プロセス関連のイベントを保存するために使われるハッシュテーブル用に割り当てられるバケットの数を制御します。	最小: 64 最大: 8192 デフォルト: 2048

AMF のパラメータ	説明	値
mnthashsz	AMF は、登録されたイベントを、イベントタイプに固有なハッシュテーブルに保存します。このパラメータは、マウント関連のイベントを保存するために使われるハッシュテーブル用に割り当てられるバケットの数を制御します。	最小: 64 最大: 8192 デフォルト: 512
conthashsz	AMF は、登録されたイベントを、イベントタイプに固有なハッシュテーブルに保存します。このパラメータは、コンテナ関連のイベントを保存するために使われるハッシュテーブル用に割り当てられるバケットの数を制御します。	最小: 1 最大: 64 デフォルト: 32
filehashsz	AMF は、登録されたイベントを、イベントタイプに固有なハッシュテーブルに保存します。このパラメータは、ファイル関連のイベントを保存するために使われるハッシュテーブル用に割り当てられるバケットの数を制御します。	最小: 1 最大: 64 デフォルト: 32
dirhashsz	AMF は、登録されたイベントを、イベントタイプに固有なハッシュテーブルに保存します。このパラメータは、ディレクトリ関連のイベントを保存するために使われるハッシュテーブル用に割り当てられるバケットの数を制御します。	最小: 1 最大: 64 デフォルト: 32

更新したパラメータ値は AMF ドライバを再設定した後反映されます。モジュールをロード解除すると、更新済みの値が失われることに注意してください。更新したチューニング可能パラメータを有効にするには、`amfconfig -U` か同等のコマンドを使ってモジュールの設定を解除し、それから `amfconfig -c` コマンドを使って再設定する必要があります。モジュールのロード時にチューニングパラメータを設定する場合には、これらの `amfconfig` コマンドを `amftab` ファイルに書き込んでください。

詳しくは、`amftab`(4)の man ページを参照してください。

C

コマンドリファレンス

この付録では以下の項目について説明しています。

Veritas コマンドに対するコマンド入力補完機能

Storage Foundation Cluster File System High Availability では、Veritas Volume Manager（VxVM）コマンドと Dynamic Multi-Pathing（DMP）コマンドに対するコマンド入力補完機能がサポートされています。

このリリースでは、コマンド入力補完機能は bash シェルでのみサポートされます。シェルは、bash バージョン 2.4 以降である必要があります。

この機能を使うには、サポート対象の VxVM コマンドまたは DMP コマンドの入力時に Tab キーを押します。コマンドは可能なかぎり補完されます。選択肢があるときに、コマンド入力補完機能によって、コマンドに関する次の有効なオプションが表示されます。表示された値のうちの 1 つを入力します。カッコに囲まれた値は、ユーザー指定の値を示します。

メモ: プラットフォーム固有のオプションは、コマンド入力補完機能ではサポートされません。

デフォルトでは、ログインのたびに bash シェルを呼び出すことで、コマンド入力補完機能を使うことができます。コマンド入力補完機能を永続的に有効にするには、次のコマンドを使用します。

```
# vxdctl cmdcompletion enable
```

コマンド入力補完機能を有効にすると、.bash_profile ファイルが作成されます(存在しない場合)。

コマンド入力補完機能を永続的に無効にするには、次のコマンドを使用します。

```
# vxdctl cmdcompletion disable
```

vxdctl(1M)マニュアルページを参照してください。

コマンド入力補完機能をサポートするコマンドは、次のとおりです。

- vxassist
- vxcache
- vxcdsconvert
- vxclustadm
- vxconfigd
- vxdctl
- vxddladm
- vxdg
- vxdisk
- vxdmpadm
- vxdisksetup
- vxdiskunsetup
- vxdmpadm
- vxedit
- vxinstall
- vxplex
- vxprint
- vxreattach
- vxrecover
- vxresize

- vxsd
- vxsnap
- vxstat
- vxtask
- vxtrace
- vxtune
- vxvol
- vxvset

Veritas Volume Manager コマンドの参照

多くの Veritas Volume Manager（VxVM）のコマンド（デーモン、ライブラリコマンド、サポートスクリプトを除く）は、/opt/VRTS/bin ディレクトリから /usr/sbin ディレクトリにリンクされています。PATH 環境変数に次のディレクトリを追加することをお勧めします。

- Bourne または Korn シェル（sh または ksh）を使っている場合は、次のコマンドを使います。

```
$ PATH=$PATH:/usr/sbin:/opt/VRTS/bin:/opt/VRTSvxfs/sbin:¥
  /opt/VRTSdbed/bin:/opt/VRTSob/bin
$ MANPATH=/usr/share/man:/opt/VRTS/man:$MANPATH
$ export PATH MANPATH
```

- C シェル（csh または tcsh）を使っている場合は、次のコマンドを使います。

```
% set path = ( $path /usr/sbin /opt/VRTSvxfs/sbin ¥
  /opt/VRTSdbed/bin /opt/VRTSob/bin /opt/VRTS/bin )
% setenv MANPATH /usr/share/man:/opt/VRTS/man:$MANPATH
```

VxVM の library コマンドとサポートスクリプトは、/usr/lib/vxvm ディレクトリ構造に配置されます。これらを定期的に使う必要がある場合は、これらのディレクトリを自分のパスに含めることができます。

個々のコマンドに関する詳細情報については、該当するマニュアルページの 1M セクションを参照してください。

p.1202 の「Veritas Volume Manager のマニュアルページ」を参照してください。

このコマンドやスクリプトをサポートするための他のコマンドとスクリプトは、通常は使われないため、/opt/VRTS/bin には配置されません。また、マニュアルページもありません。

次の表に、よく使うコマンドの概略を示します。

- 表 C-1 は、VxVM 内のオブジェクトに関する情報を取得するためのコマンドの一覧です。
- 表 C-2 は、ディスクを管理するためのコマンドの一覧です。
- 表 C-3 は、ディスクグループを作成し、管理するためのコマンドの一覧です。
- 表 C-4 は、サブディスクを作成し、管理するためのコマンドの一覧です。
- 表 C-5 は、プレックスを作成し、管理するためのコマンドの一覧です。
- 表 C-6 は、ボリュームを作成するためのコマンドの一覧です。
- 表 C-7 は、ボリュームを管理するためのコマンドの一覧です。
- 表 C-8 は、VxVM でのタスクを監視し、制御するためのコマンドの一覧です。

表 C-1　　VxVM 内のオブジェクトに関する情報の取得

コマンド	説明
`vxdctl license [init]`	VxVM のライセンスされている機能を一覧表示します。 init パラメータは、ホストに対してライセンスが追加または削除され、新しいライセンスを有効にするときに必要になります。
`vxdisk [-g diskgroup] list [diskname]`	VxVM の制御下にあるディスクを一覧表示します。 p.378 の「ディスク情報の表示」を参照してください。 例: `# vxdisk -g mydg list`
`vxdg list [diskgroup]`	ディスクグループに関する情報を一覧表示します。 p.984 の「ディスク情報の表示」を参照してください。 例: `# vxdg list mydg`

コマンド	説明
`vxdg -s list`	共有ディスクグループに関する情報を一覧表示します。 p.446 の「共有ディスクグループの一覧表示」を参照してください。 例: `# vxdg -s list`
`vxdisk -o alldgs list`	ディスクのすべてのディスクグループを一覧表示します。指定したディスクグループは標準として表示され、さらに、他のすべてのディスクグループが一重引用符の中に列挙されます。
`vxdisk -o cluster list`	クラスタ内のすべてのディスク(ローカルおよび共有) のグローバルビューを提供します。

`vxinfo [-g diskgroup] [volume ...]`	ボリュームへのアクセスおよびその使用が可能かどうかを表示します。
	詳しくは、『Veritas InfoScale トラブルシューティングガイド』を参照してください。
	例:
	`# vxinfo -g mydg myvol1 ¥`
	`  myvol2`

`vxprint -hrt [-g diskgroup] [object ...]`	VxVM 内のオブジェクトに関する情報を 1 行で出力します。
	例:
	`# vxprint -g mydg myvol1 ¥`
	`  myvol2`

`vxlist`	Veritas Volume Manager(VxVM)や Veritas File System(VxFS)の情報など、SF 設定の統合表示を提供します。 `vxlist`(1M)マニュアルページを参照してください。

コマンド	説明
`vxprint -st [-g diskgroup] [subdisk ...]`	サブディスクに関する情報を表示します。 例: `# vxprint -st -g mydg`
`vxprint -pt [-g diskgroup] [plex ...]`	プレックスに関する情報を表示します。 例: `# vxprint -pt -g mydg`

表 C-2 ディスクの管理

コマンド	説明
`vxdisk [-o full] reclaim {disk|enclosure|diskgroup}...`	シンプロビジョニング LUN のストレージ再生利用を実行します。
`vxdiskadm`	メニューベースのインターフェースを使って、VxVM 内のディスクを管理します。
`vxdiskadd [devicename ...]`	デバイス名で指定されたディスクを追加します。 p.398 の「vxdiskadd を使った VxVM の制御下へのディスクの配置」を参照してください。 例: `# vxdiskadd sde`
`vxedit [-g diskgroup] rename ¥ olddisk newdisk`	VxVM の制御下にあるディスクの名前を変更します。 p.402 の「ディスク名の変更」を参照してください。 例: `# vxedit -g mydg rename ¥ mydg03 mydg02`

コマンド	説明

`vxedit [-g diskgroup] set ¥ reserve=on|off diskname`	ディスクグループ内のディスクを使用対象外または使用対象に設定します。
	例:
	`# vxedit -g mydg set ¥ reserve=on mydg02`
	`# vxedit -g mydg set ¥ reserve=off mydg02`

`vxedit [-g diskgroup] set ¥ nohotuse=on|off diskname`	ディスク上の空き領域を、ホットリロケーションの適用対象または適用対象外に設定します。
	p.913 の「ディスクのホットリロケーション適用対象からの除外」を参照してください。
	p.914 の「ディスクのホットリロケーション適用対象からの除外を解除」を参照してください。
	例:
	`# vxedit -g mydg set ¥ nohotuse=on mydg03`
	`# vxedit -g mydg set ¥ nohotuse=off mydg03`

`vxedit [-g diskgroup] set ¥ spare=on|off diskname`	ホットリロケーションのスペアのプールから、ディスクを追加または削除します。
	p.911 の「ホットリロケーションのスペアディスクの設定」を参照してください。
	p.913 の「ホットリロケーションスペアディスクの設定解除」を参照してください。
	例:
	`# vxedit -g mydg set ¥ spare=on mydg04`
	`# vxedit -g mydg set ¥ spare=off mydg04`

コマンド	説明
`vxdisk online devicename`	ディスクデバイスのオフライン状態を消去します。 `vxdisk(1M)`マニュアルページを参照してください。 例: `# vxdisk online sde`
`vxdisk offline devicename`	ディスクをオフラインにします。 例: `# vxdisk offline sde`
`vxdg -g diskgroup adddisk diskname`	ディスクグループにディスクを追加します。 p.214 の「新しい LUN の追加による既存のストレージの拡張」を参照してください。 例: `# vxdg -g mydg adddisk mydg02`
`vxdg -g diskgroup rmdisk diskname`	ディスクをディスクグループから削除します。 p.988 の「ディスクグループからのディスクの削除」を参照してください。 例: `# vxdg -g mydg rmdisk mydg02`
`vxdisksetup devicename`	VxVM で使うようにディスクを設定します。 p.213 の「新しいストレージのプロビジョニング」を参照してください。 例: `# /etc/vx/bin/vxdisksetup -i enc1_3`
`vxdiskunsetup devicename`	VxVM の制御下からディスクを削除します。 p.988 の「ディスクグループからのディスクの削除」を参照してください。 例: `# vxdiskunsetup sdg`

表 C-3 ディスクグループの作成と管理

コマンド	説明
`vxdg [-s] init diskgroup ¥ [diskname=]devicename`	初期化済みディスクを使って、ディスクグループを作成します。 p.986 の「ディスクグループの作成」を参照してください。 p.448 の「共有ディスクグループの作成」を参照してください。 例: `# vxdg init mydg ¥ mydg01=sde`
`vxdg -g diskgroup listssbinfo`	矛盾する設定情報を表示します。 p.1008 の「競合する設定コピーの扱い方」を参照してください。 例: `# vxdg -g mydg listssbinfo`
`vxdg [-n newname] deport diskgroup`	ディスクグループをデポートし、オプションでそのディスクグループの名前を変更します。 p.989 の「ディスクグループのデポート」を参照してください。 例: `# vxdg -n newdg deport mydg`
`vxdg [-n newname] import diskgroup`	ディスクグループをインポートし、オプションでそのディスクグループの名前を変更します。 p.990 の「ディスクグループのインポート」を参照してください。 例: `# vxdg -n newdg import mydg`

コマンド	説明

`vxdg [-n newname] -s import diskgroup`	ディスクグループをクラスタ共有ディスクグループとしてインポートし、オプションでその名前を変更します。
	p.448 の「共有ディスクグループのインポート」を参照してください。
	例:
	`# vxdg -n newsdg -s import ¥`
	`  mysdg`

`vxdg [-o expand] listmove sourcedg ¥`
`targetdg object ...`
	ディスクグループの移動により、影響を受ける可能性のあるオブジェクトを一覧表示します。
	p.954 の「移動により影響を受ける可能性のあるオブジェクトの一覧表示」を参照してください。
	例:
	`# vxdg -o expand listmove ¥`
	`  mydg newdg myvol1`

`vxdg [-o expand] move sourcedg ¥`
`targetdg object ...`
	ディスクグループ間でオブジェクトを移動します。
	p.956 の「ディスクグループ間のオブジェクト移動」を参照してください。
	例:
	`# vxdg -o expand move mydg ¥`
	`  newdg myvol1`

`vxdg [-o expand] split sourcedg ¥`
`targetdg object ...`
	ディスクグループを分割し、指定したオブジェクトをターゲットディスクグループに移動します。
	p.959 の「ディスクグループの分割」を参照してください。
	例:
	`# vxdg -o expand split mydg ¥`
	`  newdg myvol2 myvol3`

コマンド	説明
`vxdg join sourcedg targetdg`	2 つのディスクグループを結合します。 p.961 の「ディスクグループの結合」を参照してください。 例: `# vxdg join newdg mydg`
`vxdg -g diskgroup set ¥ activation=ew|ro|sr|sw|off`	クラスタ内の共有ディスクグループのアクティベーションモードを設定します。 p.450 の「共有ディスクグループ上のアクティベーションモードの変更」を参照してください。 例: `# vxdg -g mysdg set ¥ activation=sw`
`vxrecover -g diskgroup -sb`	インポートされたディスクグループ内のすべてのボリュームを起動します。 p.994 の「システム間でのディスクグループの移動」を参照してください。 例: `# vxrecover -g mydg -sb`
`vxdg destroy diskgroup`	ディスクグループを破棄し、そのディスクグループに所属するディスクを解放します。 p.1015 の「ディスクグループの破棄」を参照してください。 例: `# vxdg destroy mydg`

表 C-4 サブディスクの作成と管理

コマンド	説明
vxmake [-g *diskgroup*] sd *subdisk* ¥ *diskname*,*offset*,*length*	サブディスクを作成します。 例: # vxmake -g mydg sd ¥ mydg02-01 mydg02,0,8000
vxsd [-g *diskgroup*] assoc *plex* ¥ *subdisk*...	サブディスクを既存のプレックスに関連付けます。 例: # vxsd -g mydg assoc home-1 ¥ mydg02-01 mydg02-00 ¥ mydg02-01
vxsd [-g *diskgroup*] assoc *plex* ¥ *subdisk1:0 ... subdiskM:N-1*	ストライプボリュームまたはRAID 5ボリューム内のカラムの末尾にサブディスクを追加します。 例: # vxsd -g mydg assoc ¥ vol01-01 mydg10-01:0 ¥ mydg11-01:1 mydg12-01:2
vxsd [-g *diskgroup*] mv *oldsubdisk* ¥ *newsubdisk* ...	サブディスクを交換します。 例: # vxsd -g mydg mv mydg01-01 ¥ mydg02-01
vxsd [-g *diskgroup*] -s *size* split ¥ *subdisk sd1 sd2*	サブディスクを 2 つに分割します。 例: # vxsd -g mydg -s 1000m ¥ split mydg03-02 mydg03-02 ¥ mydg03-03

コマンド	説明
`vxsd [-g diskgroup] join ¥ sd1 sd2 ... subdisk`	2 つ以上のサブディスクを結合します。 例: `# vxsd -g mydg join ¥ mydg03-02 mydg03-03 ¥ mydg03-02`
`vxassist [-g diskgroup] move ¥ volume ¥!olddisk newdisk`	ボリューム内のサブディスクをディスク間で再配置します。 例: `# vxassist -g mydg move ¥ myvol ¥!mydg02 mydg05` **メモ:** ! 文字は一部のシェルでの特殊文字です。次の bash シェルの例では、この文字をエスケープしています。
`vxunreloc [-g diskgroup] original_disk`	サブディスクを元のディスク位置へ再配置します。 p.916 の「vxunreloc を使った再配置されたサブディスクの移動」を参照してください。 例: `# vxunreloc -g mydg mydg01`
`vxsd [-g diskgroup] dis subdisk`	サブディスクとプレックスの関連付けを解除します。 例: `# vxsd -g mydg dis mydg02-01`
`vxedit [-g diskgroup] rm subdisk`	サブディスクを削除します。 例: `# vxedit -g mydg rm mydg02-01`

コマンド	説明
vxsd [-g *diskgroup*] -o rm dis *subdisk*	サブディスクとプレックスの関連付けを解除し、サブディスクをプレックスから削除します。 例: # vxsd -g mydg -o rm dis ¥ mydg02-01

表 C-5 プレックスの作成と管理

コマンド	説明
vxmake [-g *diskgroup*] plex *plex* ¥ sd=*subdisk1*[,*subdisk2*,...]	コンカチネイテッドプレックスを作成します。 例: # vxmake -g mydg plex ¥ vol01-02 ¥ sd=mydg02-01,mydg02-02

vxmake [-g diskgroup] plex *plex* ¥
layout=stripe|raid5 stwidth=*W* ¥
ncolumn=*N* ¥
sd=*subdisk1*[,*subdisk2*,...]
	ストライプ化プレックスまたはRAID 5プレックスを作成します。
	例:
	# vxmake -g mydg plex pl-01 ¥
	layout=stripe stwidth=32 ¥
	ncolumn=2 ¥
	sd=mydg01-01,mydg02-01

vxplex [-g *diskgroup*] att *volumeplex*	プレックスを既存のボリュームに追加します。 p.1022 の「手動によるプレックスの再接続」を参照してください。 例: # vxplex -g mydg att vol01 ¥ vol01-02

コマンド	説明
`vxplex [-g diskgroup] det plex`	プレックスを切断します。 例: `# vxplex -g mydg det vol01-02`
`vxmend [-g diskgroup] off plex`	メンテナンスのため、プレックスをオフラインにします。 例: `# vxmend -g mydg off vol02-02`
`vxmend [-g diskgroup] on plex`	プレックスを使うために再有効にします。 p.1022 の「手動によるプレックスの再接続」を参照してください。 例: `# vxmend -g mydg on vol02-02`
`vxplex [-g diskgroup] mv oldplex ¥ newplex`	プレックスを交換します。 例: `# vxplex -g mydg mv ¥ vol02-02 vol02-03`
`vxplex [-g diskgroup] cp volume ¥ newplex`	ボリュームをプレックスにコピーします。 例: `# vxplex -g mydg cp vol02 ¥ vol03-01`
`vxmend [-g diskgroup] fix clean plex`	起動できないボリューム内のプレックスの状態を CLEAN 状態に設定します。 p.1022 の「手動によるプレックスの再接続」を参照してください。 例: `# vxmend -g mydg fix clean ¥ vol02-02`

コマンド	説明
vxplex [-g *diskgroup*] -o rm dis *plex*	プレックスとボリュームの関連付けを解除し、プレックスをボリュームから削除します。 例: # vxplex -g mydg -o rm dis ¥ vol03-01

表 C-6　　ボリュームの作成

コマンド	説明

vxassist [-g *diskgroup*] maxsize ¥
layout=*layout* [*attributes*]
	作成可能なボリュームの最大サイズを表示します。
	例:
	# vxassist -g mydg maxsize ¥
	layout=raid5 nlog=2

vxassist -b [-g *diskgroup*] make ¥ *volume length* [layout=*layout*] ¥ [*attributes*]	ボリュームを作成します。 p.248 の「指定したディスクにおけるボリュームの作成」を参照してください。 例: # vxassist -b -g mydg make ¥ myvol 20g layout=concat ¥ mydg01 mydg02
vxassist -b [-g *diskgroup*] make ¥ *volume length* layout=mirror ¥ [nmirror=*N*][*attributes*]	ミラーボリュームを作成します。 p.242 の「ミラーボリュームの作成」を参照してください。 例: # vxassist -b -g mydg make ¥ mymvol 20g layout=mirror ¥ nmirror=2

コマンド	説明

`vxassist -b [-g diskgroup] make ¥ volume length layout=layout ¥ exclusive=on [attributes]`	クラスタ内の 1 つのノードで排他的に起動できるボリュームを作成します。
	p.456 の「排他的起動権限を持つボリュームの作成」を参照してください。
	例:
	`# vxassist -b -g mysdg make ¥`
	`mysmvol 20g layout=mirror ¥`
	`exclusive=on`

`vxassist -b [-g diskgroup] make ¥ volume length layout={stripe¦raid5} ¥ [stripeunit=W] [ncol=N] ¥ [attributes]`	ストライプボリュームまたは RAID 5 ボリュームを作成します。
	p.244 の「ストライプボリュームの作成」を参照してください。
	p.246 の「RAID 5 ボリュームの作成」を参照してください。
	例:
	`# vxassist -b -g mydg make ¥`
	`mysvol 20g layout=stripe ¥`
	`stripeunit=32 ncol=4`

`vxassist -b [-g diskgroup] make ¥ volume length layout=mirror ¥ mirror=ctlr [attributes]`	別個のコントローラ上にミラーデータプレックスを持つボリュームを作成します。
	例:
	`# vxassist -b -g mydg make ¥`
	`mymcvol 20g layout=mirror ¥`
	`mirror=ctlr`

`vxmake -b [-g diskgroup] ¥ -Uusage_type vol volume ¥ [len=length] plex=plex,...`	既存のプレックスからボリュームを作成します。
	例:
	`# vxmake -g mydg -Uraid5 ¥`
	`vol r5vol ¥`
	`plex=raidplex,raidlog1,¥`
	`raidlog2`

コマンド	説明
`vxvol [-g diskgroup] start volume`	使うボリュームを初期化して起動します。 例: `# vxvol -g mydg start r5vol`
`vxvol [-g diskgroup] init zero ¥ volume`	使うボリュームを初期化してゼロクリアします。 例: `# vxvol -g mydg init zero ¥ myvol`

表 C-7　　ボリュームの管理

コマンド	説明
`vxassist [-g diskgroup] mirror ¥ volume [attributes]`	ミラーをボリュームに追加します。 p.972 の「ボリュームへのミラーの追加」を参照してください。 例: `# vxassist -g mydg mirror ¥ myvol mydg10`
`vxassist [-g diskgroup] remove ¥ mirror volume [attributes]`	ミラーをボリュームから削除します。 p.974 の「ミラーの削除 」を参照してください。 例: `# vxassist -g mydg remove ¥ mirror myvol ¥!mydg11` **メモ:** ! 文字は一部のシェルでの特殊文字です。次の bash シェルの例では、この文字をエスケープしています。

コマンド	説明
`vxassist [-g diskgroup] ¥ {growto|growby} volume length`	指定したサイズに、または指定した量だけボリュームを拡張します。 例: `# vxassist -g mydg growby ¥ myvol 10g`
`vxassist [-g diskgroup] ¥ {shrinkto|shrinkby} volume length`	指定したサイズに、または指定した量だけボリュームを縮小します。 例: `# vxassist -g mydg shrinkto ¥ myvol 20g`
`vxresize -b -F vxfs [-g diskgroup] ¥ volume length diskname ...`	ボリュームおよびそのボリューム上に作成された Veritas File System のサイズを変更します。 例: `# vxresize -b -F vxfs ¥ -g mydg myvol 20g mydg10 ¥ mydg11`
`vxsnap [-g diskgroup] prepare volume ¥ [drl=on|sequential|off]`	インスタントスナップショットや DRL ログ用のボリュームを準備します。 p.694 の「インスタントスナップの DCO と DCO ボリュームの追加」を参照してください。 例: `# vxsnap -g mydg prepare ¥ myvol drl=on`

コマンド	説明

`vxsnap [-g diskgroup] make ¥`
`source=volume¥`
`/newvol=snapvol¥`
`[/nmirror=number]`
	元のボリュームのプレックスを切り離して、ボリュームのフルサイズインスタントスナップショットを作成します。
	p.693 の「インスタントスナップショットの作成」を参照してください。
	例:
	`# vxsnap -g mydg make ¥`
	`source=myvol/¥`
	`newvol=mysnpvol/¥`
	`nmirror=2`

`vxsnap [-g diskgroup] make ¥`
`source=volume/snapvol=snapvol`
	準備された空のボリュームを使って、ボリュームのフルサイズインスタントスナップショットを作成します。
	p.698 の「フルサイズインスタントスナップショットまたはリンクブレークオフスナップショットに使うボリュームの作成」を参照してください。
	p.693 の「インスタントスナップショットの作成」を参照してください。
	例:
	`# vxsnap -g mydg make ¥`
	`source=myvol/snapvol=snpvol`

コマンド	説明

vxmake [-g *diskgroup*] cache ¥
cache_object cachevolname=*volume* ¥
[regionsize=*size*]
	領域最適化インスタントスナップショットで使うためのキャッシュオブジェクトを作成します。
	p.696 の「共有キャッシュオブジェクトの作成」を参照してください。
	キャッシュボリュームがすでに作成されている必要があります。キャッシュオブジェクトを作成した後、vxcache start コマンドを使ってキャッシュオブジェクトを有効にします。
	次に例を示します。
	`# vxassist -g mydg make ¥`
	`  cvol 1g layout=mirror ¥`
	`  init=active mydg16 mydg17`
	`# vxmake -g mydg cache cobj ¥`
	`  cachevolname=cvol`
	`# vxcache -g mydg start cobj`

vxsnap [-g *diskgroup*] make ¥
source=*volume*/newvol=*snapvol*¥
/cache=*cache_object*
	ボリュームの領域最適化インスタントスナップショットを作成します。
	p.693 の「インスタントスナップショットの作成」を参照してください。
	例:
	`# vxsnap -g mydg make ¥`
	`  source=myvol/¥`
	`  newvol=mysosvol/¥`
	`  cache=cobj`

vxsnap [-g *diskgroup*] refresh *snapshot*	元のボリュームからスナップショットを更新します。
	p.715 の「インスタント領域最適化スナップショットの更新」を参照してください。
	例:
	`# vxsnap -g mydg refresh ¥`
	`  mysnpvol`

コマンド	説明
`vxsnap [-g diskgroup] dis snapshot`	スナップショットを独立したボリュームにします。 p.717 の「インスタントスナップショットの関連付けの解除」を参照してください。 例: `# vxsnap -g mydg dis mysnpvol`

`vxsnap [-g diskgroup] unprepare ¥ volume`	インスタントスナップショットや DRL ログのサポートをボリュームから削除します。
	例:
	`# vxsnap -g mydg unprepare ¥`
	`  myvol`

`vxassist [-g diskgroup] relayout ¥ volume [layout=layout] ¥ [relayout_options]`	ボリュームのオンライン再レイアウトを実行します。
	p.966 の「オンライン再レイアウトの実行」を参照してください。
	例:
	`# vxassist -g mydg relayout ¥`
	`  vol2 layout=stripe`

`vxassist [-g diskgroup] relayout ¥ volume layout=raid5 ¥ stripeunit=W ¥ ncol=N`	ストライプ幅 *W* と *N* カラムを使って RAID-5 ボリュームとしてボリュームの再レイアウトを実行します。
	p.966 の「オンライン再レイアウトの実行」を参照してください。
	例:
	`# vxassist -g mydg relayout ¥`
	`  vol3 layout=raid5 ¥`
	`  stripeunit=16 ncol=4`

コマンド	説明
`vxrelayout [-g diskgroup] -o bg ¥ reverse volume`	一時停止中のボリューム再レイアウトを元に戻します。 例: `# vxrelayout -g mydg -o bg ¥ reverse vol3`
`vxassist [-g diskgroup] convert ¥ volume [layout=layout] ¥ [convert_options]`	階層化ボリュームレイアウトと非階層化ボリュームレイアウト間の変換を行います。 例: `# vxassist -g mydg convert ¥ vol3 layout=stripe-mirror`
`vxassist [-g diskgroup] remove ¥ volume volume`	ボリュームを削除します。 p.1052 の「ボリュームの削除」を参照してください。 例: `# vxassist -g mydg remove ¥ myvol`

表 C-8 タスクの監視と制御

コマンド	説明
`command [-g diskgroup] -t tasktag ¥ [options] [arguments]`	VxVM コマンドにタスクタグを指定します。 p.963 の「タスクタグの指定」を参照してください。 例: `# vxrecover -g mydg ¥ -t mytask -b mydg05`

コマンド	説明
`vxtask [-h] [-g diskgroup] list`	システム上で実行されているタスクを一覧表示します。 p.965 の「vxtask コマンドの使用」を参照してください。 例: `# vxtask -h -g mydg list`
`vxtask monitor task`	タスクの進行状況を監視します。 p.965 の「vxtask コマンドの使用」を参照してください。 例: `# vxtask monitor mytask`
`vxtask pause task`	タスクの操作を中断します。 p.965 の「vxtask コマンドの使用」を参照してください。 例: `# vxtask pause mytask`
`vxtask -p [-g diskgroup] list`	一時停止中のすべてのタスクを一覧表示します。 p.965 の「vxtask コマンドの使用」を参照してください。 例: `# vxtask -p -g mydg list`
`vxtask resume task`	一時停止中のタスクを再開します。 p.965 の「vxtask コマンドの使用」を参照してください。 例: `# vxtask resume mytask`

コマンド	説明
`vxtask abort task`	タスクを中断し、変更を元に戻します。 p.965 の「vxtask コマンドの使用」を参照してください。 例: `# vxtask abort mytask`

スレーブノードでの実行をサポートされる CVM コマンド

表 C-9 に、スレーブノードでの実行がサポートされるコマンドの完全な一覧を示します。

表 **C-9** スレーブノードでの実行をサポートされる CVM コマンドのリスト

コマンド	サポートされる操作
vxdg	

コマンド	サポートされる操作
	vxdg -s init <shared_dg> [cds=on|off]
	vxdg -T < different_versions> -s init <shared_dg> [minor=base-minor] [cds=on|off]
	vxdg [-n newname] [-h new-host-id] deport <shared_dg>
	vxdg [-Cfst] [-n newname] [-o clearreserve] [-o useclonedev={on|off}] [-o updateid] [-o noreonline] [-o selectcp=diskid] [-o dgtype=shared] import <shared_dg>
	vxdg destroy <shared_dg>
	vxdg -g <shared_dg> [-o overridessb] [-f] adddisk [disk=]device
	vxdg-g <shared_dg>addsite site
	vxdg -g <shared_dg> reattachsite site
	vxdg -g <shared_dg> detachsite site
	vxdg -g <shared_dg> rmsite site
	vxdg -g <shared_dg> renamesite oldname newname
	vxdg flush <shared_dg>
	vxdg [-qa] -g <shared_dg> free [medianame...]
	vxdg join sourcedg targetdg（dg は両方とも共有である必要があります）
	vxdg split sourcedg targetdg
	vxdg [-q] [-s] [-o listreserve] list [diskgroup...]
	vxdg [-o expand] move sourcedg targetdg object（dg は両方とも共有である必要があります）
	vxdg -g shared_dg recover
	vxdg -g <shared_dg> [-f] reminor <shared_dg> new-minor-number
	vxdg -g <shared_dg> rmdisk medianame...
	vxdg -g <shared_dg>[-q] spare [medianame...]
	vxdg -g <shared_dg> [-f] [-o retain|replace] settag [encl:<enclosure>] name[=value name[=value]
	vxdg [-q] listtag <shared_dg>
	vxdg -g <shared_dg> rmtag [encl:<enclosure>] name=value
	vxdg -g <shared_dg> set siteconsistent=on
	vxdg upgrade <shared_dg>
	vxdg -g <shared_dg>set attr=value ...

コマンド	サポートされる操作
vxassist	vxassist -g <shared_dg> [-b] convert volume layout=<type> vxassist -g <shared_dg> [-b] addlog volume vxassist -g <shared_dg> [-b] mirror volume vxassist [-b]-g <shared_dg>make volume length [layout=layout] diskname ... vxassist -g <shared_dg> [-b] growby volume lengthchange [attribute ...] vxassist [-b] -g <shared_dg> growto volume newlength vxassist -g <shared_dg> shrinkby volume lengthchange vxassist -g <shared_dg> shrinkto volume newlength vxassist -g <shared_dg> settag volume|vset tagname[=tagvalue] vxassist -g <shared_dg>replacetag volume|vset oldtag newtag vxassist -g <shared_dg> removetag volume|vset tagname vxassist -g <shared_dg> move volume-name storage-spec vxassist -g <shared_dg> relayout {volume-name} layout=<type> vxassist -g <shared_dg> remove {volume|mirror|log} volume-name vxassist -g <shared_dg> snapshot volume-name [snapshot-name] [comment=<comment>] vxassist -g <shared_dg> snapstart volume vxassist -g <shared_dg> maxsize layout=<> nmirror=<> / nlog=<> vxassist -g <shared_dg> maxgrow volume vxassist -g <shared_dg> snapback snapvol vxassist -g <shared_dg> snapclear snapvol1
vxcache	vxcache -g <shared_dg> start cacheobject vxcache -g <shared_dg> stop cacheobject vxcache -g <shared_dg> att volume cacheobject vxcache -g <shared_dg> dis cachevol vxcache -g <shared_dg> shrinkcacheto cacheobject newlength vxcache -g <shared_dg> shrinkcacheby cacheobject lengthchange vxcache -g <shared_dg> growcacheto cacheobject newlength vxcache -g <shared_dg> growcacheby cacheobject lengthchange

コマンド	サポートされる操作
vxdco	vxdco -g <shared_dg> dis dco vxdco -g <shared_dg> att volume dco vxdco -g <shared_dg>[-o force] enable dco
vxedit	vxedit -g <shared_dg> set comment="plex comment" plex1 vxedit -g <shared_dg> -rf rm volume vxedit -g <shared_dg>rename oldname newname vxedit -g <shared_dg> set what=value vxedit-g <shared_dg> set user=value mode=value medianame vxedit -g <shared_dg> set failing=off <disk name> vxedit -g <shared_dg> set fstype volumename vxedit -g <shared_dg> set len subdisk vxedit -g <shared_dg> set orig_dmname subdisk vxedit -g <shared_dg> set orig_dmoffset subdisk vxedit -g <shared_dg> set diskdetpolicy diskgroup
vxmake	vxmake -g <shared_dg> sd name [attr...] vxmake-g <shared_dg>plex plex sd=subdisk1[,subdisk2,...] vxmake -g <shared_dg> -U fsgen vol homevol1 plex=plex-1 vxmake -g <shared_dg> -U fsgen vol volume1 plex=plex1,plex2 vxmake -g <shared_dg> cache name regionsize=<size> vxmake -g <shared_dg> dco volume log=dco
vxmend	vxmend -g <shared_dg> on plex vxmend -g <shared_dg> off plex
vxmirror	vxmirror -g <shared_dg>medianame vxmirror -g <shared_dg> -d [yes|no]

コマンド	サポートされる操作
vxplex	vxplex -g <shared_dg> att volume plex vxplex -g <shared_dg> cp volume new_plex vxplex -g <shared_dg> dis plex1 vxplex -g <shared_dg>mv original_plex new_plex vxplex -g <shared_dg> snapstart vol snapplex vxplex -g <shared_dg> snaphot snapplex vxplex -g <shared_dg> snapback vol snapplex vxplex -g <shared_dg> plex
vxrelayout	vxrelayout -g <shared_dg> status volume vxrelayout -g <shared_dg> start volume vxrelayout -g <shared_dg> reverse volname
vxsd	vxsd -g <shared_dg> assoc plex subdisk1 [subdisk2 subdisk3 ...] vxsd -g <shared_dg> [-o force] dis subdisk vxsd -g <shared_dg> mv old_subdisk new_subdisk [new_subdisk ...] vxsd -g <shared_dg> aslog plex2 sdisk3 vxsd -g <shared_dg> join subdisk1 subdisk2 ... new_subdisk vxsd -g <shared_dg> [-o force] dis subdisk vxsd -g <shared_dg> split subdisk newsd [newsd2]...

コマンド	サポートされる操作
vxsnap	vxsnap -g <shared_dg> addmir volume [nmirror=N] vxsnap -g <shared_dg> prepare volume vxsnap -g <shared_dg> rmmir volume vxsnap -g <shared_dg>unprepare volume vxsnap -g <shared_dg> make snapshot_tuple [snapshot_tuple]... [alloc=storage_attributes] vxsnap [-f] -g <shared_dg> dis volume vxsnap -g <shared_dg> addmap volumename count vxsnap -g <shared_dg> print volumename vxsnap -g <shared_dg> list volumename vxsnap -g <shared_dg> syncwait snapvol vxsnap -g <shared_dg> snapwait vxsnap -g <shared_dg> refresh snapvol source=volume vxsnap -g <shared_dg> restore target source=volname vxsnap -g <shared_dg> split volumename vxsnap -g <shared_dg> reattach volname source=volname
vxsnptadm	vxsnptadm -g <shared_dg> create vol [snptname=snpt] [snapvolname=snapvol] [data={yes|no}] vxsnptadm -g <shared_dg> info vol [snptname=snpt] vxsnptadm -g <shared_dg> remove vol snptname=snpt vxsnptadm -g <shared_dg> removeall vol [cookie=cookie] vxsnptadm -g <shared_dg> rename vol snptname=snpt newname=snpt2

コマンド	サポートされる操作
vxvol	vxvol -g <shared_dg> set logtype=drl | drlseq volume vxvol -g <shared_dg> start volume vxvol -g <shared_dg> stop volume vxvol -g <shared_dg> {startall|stopall} volume vxvol -g <shared_dg> init enable volume vxvol -g <shared_dg> init active volume vxvol -g <shared_dg> maint volumename vxvol -g <shared_dg> set len volumename vxvol -g <shared_dg> set logtype volumename vxvol -g <shared_dg> set loglen volumename
vxvset	vxvset -g <shared_dg> make volume-set-name volume-name vxvset -g <shared_dg> addvol volume-set-name volume-name vxvset -g <shared_dg> list volume-set-name vxvset -g <shared_dg> rmvol volume-set-name volume-name vxvset -g <shared_dg> stop volume-set-name vxvset -g <shared_dg> start volume-set-name
vxevac	vxevac -g <shared_dg> medianame
vxresize	vxresize [-Vsb] [-F fstype] -g <shared_dg> volume length
vxrecover	vxrecover -g <shared_dg> vxrecover -g <shared_dg> volume
vxckdiskrm	vxckdiskrm -g <shared_dg> medianame

Veritas Volume Manager のマニュアルページ

マニュアルページは次の項から構成されます。

1M	管理コマンド
4	ファイルフォーマット

セクション 1M - 管理コマンド

表 C-10 に、Veritas Volume Manager を管理するためのコマンドについて説明しているセクション 1M のマニュアルページを一覧表示します。

表 C-10　セクション 1M マニュアルページ

名前	説明
vxassist	ボリュームの作成、再レイアウト、変換、ミラー、バックアップ、拡張、縮小、削除、および移動に使います。
vxcache	領域最適化スナップショットのキャッシュオブジェクトの管理に使います。
autogrow	必要に応じてキャッシュボリュームを監視、サイズ変更するデーモン。
vxcdsconvert	ディスクおよびディスクグループをシステム間で移植可能な状態に変換します。
vxclustadm	クラスタの開始、停止、および再構成に使います。
vxcmdlog	コマンドログの管理に使います。
vxconfigbackup	ディスクグループ設定のバックアップに使います。
vxconfigbackupd	ディスクグループ設定バックアップデーモンです。
vxconfigd	Veritas Volume Manager 設定デーモンです。
vxconfigrestore	ディスクグループ設定をリストアします。
vxdco	バージョン DCO オブジェクトおよび DCO ボリュームに対して操作を実行するときに使います。
vxdctl	ボリューム設定デーモンを制御します。
vxddladm	デバイス検出層サブシステムの管理に使います。
vxdefault	/etc/default/vxsf で設定される、SmartMove、シン再利用、ボリュームの自動起動、共有ディスクグループのマイナー番号などの設定を指定するデフォルト値を管理します。

名前	説明
vxdg	Veritas Volume Manager ディスクグループの管理に使います。
vxdisk	Veritas Volume Manager ディスクの定義と管理に使います。
vxdiskadd	Veritas Volume Manager で使うディスクを追加します。
vxdmpadm	メニュー駆動型の Veritas Volume Manager ディスク管理プログラムです。
vxdisksetup	Veritas Volume Manager で使うディスクの設定に使います。
vxdiskunsetup	Veritas Volume Manager で使用中のディスクを設定から除外します。
vxdmpadm	DMP サブシステムの管理に使います。
vxdmptune	DMP チューニングパラメータの値を表示および変更します。このユーティリティは非推奨です。代わりに、vxdmpadm ユーティリティを使います。
vxedit	Veritas Volume Manager レコードの作成、削除、および変更に使います
vxencap	新しいディスクでパーティションをカプセル化します。
vxevac	すべてのボリュームをディスクから退避します。
vxinfo	ボリュームのアクセス可能性と可用性を表示します。
vxinitrd	VxVM モジュールをプリロードするための初期 ramdisk イメージを作成します。
vxinstall	メニュー駆動型の Veritas Volume Manager 初回設定プログラムです。
vxintro	Veritas Volume Manager ユーティリティの概要です。
vxiod	Veritas Volume Manager カーネル I/O スレッドの開始、停止、およびレポートに使います。

名前	説明
vxmake	Veritas Volume Manager 設定レコードを作成します。
vxmemstat	Veritas Volume Manager のメモリ統計情報を表示します。
vxmend	設定レコードの単純な問題を修復します。
vxmirror	ディスク上のボリュームのミラー化またはデフォルトミラー化の制御に使います。
vxnotify	Veritas Volume Manager 設定イベントを表示します。
vxplex	プレックス上で Veritas Volume Manager 操作を実行します。
vxprint	Veritas Volume Manager 設定のレコードを表示します。
vxr5check	RAID-5 ボリュームのパリティ検査を実行します。
vxreattach	アクセス可能になったディスクドライブを再接続します。
vxrecover	ボリュームリカバリ操作を実行します。
vxrelayout	オンラインストレージを別のレイアウトに変換します。
vxrelocd	Veritas Volume Manager の障害イベントを監視し、障害の発生したサブディスクを再配置します。
vxresize	ファイルシステムを含むボリュームのサイズを変更します。
vxrootadm	ブートディスクのスナップショットを拡大または作成します。
vxrootmir	ルートディスクを代替ディスクにミラー化します。
vxscsiinq	SCSI 問い合わせデータを表示します。
vxsd	サブディスク上で Veritas Volume Manager 操作を実行します。

名前	説明
vxsnap	ボリューム上の DRL の有効化およびインスタントスナップショットの作成と管理に使います。
vxstat	Veritas Volume Manager 統計情報管理ユーティリティです。
vxtask	Veritas Volume Manager タスクの一覧表示と管理に使います。
vxtrace	ボリューム上の操作をトレースします。
vxtranslog	トランザクションログの管理に使います。
vxtune	Veritas Volume Replicator および Veritas Volume Manager のチューニングパラメータの調整に使います。
vxunreloc	ホットリロケーションされたサブディスクをもとのディスクに移動します。
vxunroot	カプセル化されたルートボリュームから Veritas Volume Manager フックを削除します。
vxvol	ボリューム上で Veritas Volume Manager 操作を実行します。
vxvoltune	VxVM チューニングパラメータの値を表示および変更します。このユーティリティは非推奨です。代わりに、vxtune コマンドを使います。
vxvset	ボリュームセットの作成と管理に使います。

セクション 4 - ファイルフォーマット

表 C-11 に、Veritas Volume Manager で使われるファイルのフォーマットについて説明しているセクション 4 のマニュアルページを一覧表示します。

表 C-11　　セクション 4 マニュアルページ

名前	説明
vol_pattern	ディスクグループ検索の仕様です。
vxmake	vxmake の解説ファイルです。

Veritas File System コマンドの概略

すべての VxFS コマンドの実行形式ファイルへのシンボリックリンクは、/opt/VRTS/bin ディレクトリにインストールされています。コマンドにアクセスするには、このディレクトリを PATH 環境変数の末尾に追加します。

表 C-12 では、VxFS 固有のコマンドを説明します。

表 C-12　　VxFS コマンド

コマンド	説明
df	VxFS ファイルシステムの空きディスクブロックと i ノードの数を出力します。
fcladm	VxFS FCL を管理します。
ff	VxFS ファイルシステムのファイル名と i ノードの情報を一覧表示します。
fiostat	ファイル I/O の統計データを管理します。
fsadm	VxFS ファイルシステムのサイズ変更または断片化の解消を実行します。
fsapadm	VxFS 割り当てポリシーを管理します。
fscat	VxFS ファイルシステムの内容を表示します。
fscdsadm	オンライン CDS 操作を実行します。
fscdsconv	VxFS ファイルシステムでオフラインによる CDS の移行タスクを実行します。
fscdstask	さまざまな CDS 操作を実行します。
fsck	VxFS ファイルシステムの検査と修復を実行します。 Linux の fsck ラッパーの動作上の問題により、等号 (=) が含まれるオプションを指定する場合は、VxFS の fsck コマンド /opt/VRTS/bin/fsck を実行する必要があります。次に例を示します。 # /opt/VRTS/bin/fsck -o zapvol=MyVolName /dev/rdsk/c0t0d1s1
fsckpt_restore	VxFS Storage Checkpoint からファイルシステムをリストアします。
fsclustadm	クラスタにマウントされた VxFS ファイルシステムを管理します。
fsdb	VxFS ファイルシステムをデバッグします。
fsdedupadm	データ重複排除を管理します。
fsfreeze	VxFS ファイルシステムをフリーズし、ファイルシステム上でユーザーコマンドを実行します。
fsmap	VxFS ファイルシステムのエクステント情報を表示します。

コマンド	説明
fsppadm	VxFS 配置ポリシーを管理します。
fsppmk	配置ポリシーを作成します。
fstag	ファイルタグの作成、削除、一覧表示を実行します。
fstyp	指定されたディスクパーティションのファイルシステムタイプを出力します。
fsvmap	VxFS ファイルシステムのボリュームをファイルにマップします。
fsvoladm	VxFS ボリュームを管理します。
glmconfig	GLM（Group Lock Manager）を設定します。
glmdump	クラスタファイルシステムのスタック状態の GLM（Group Lock Manager）ロックを出力します。
glmstat	GLM（Group Lock Manager）の統計情報収集ユーティリティ。
mkdstfs	SmartTier ファイルシステム作成ユーティリティ。
mkfs	VxFS ファイルシステムを作成します。
mount	VxFS ファイルシステムをマウントします。
ncheck	VxFS ファイルシステムの i ノードの番号からパス名を生成します。
setext	VxFS ファイルシステムのファイルにエクステント属性を設定します。
vxcompress	ファイルを圧縮または圧縮解除します。
vxdump	ファイルシステムの増分ダンプを実行します。
vxedquota	VxFS ファイルシステムのユーザークォータを編集します。
vxenablef	特定の VxFS の機能を有効にします。
vxfilesnap	VxFS ファイルシステムのファイルのコピーオンライトコピーを作成します。
vxfsconvert	マウントされていないファイルシステムを VxFS に変換するか、または VxFS ディスクレイアウトバージョンを更新します。
vxfsstat	ファイルシステムの統計を表示します。
vxlsino	VxFS パス名の逆引きルックアップを実行します。
vxquot	VxFS ファイルシステムのファイルシステム所有権のサマリーを表示します。
vxquota	VxFS ファイルシステム上のユーザーディスククォータと使用率を表示します。
vxquotaoffvxquotaon	VxFS ファイルシステムのクォータを有効または無効にします。

コマンド	説明
vxrepquota	VxFS ファイルシステムのクォータのサマリーを出力します。
vxrestore	ファイルシステムの増分リストアを実行します。
vxtunefs	VxFS ファイルシステムをチューニングします。
vxupgrade	マウントされている VxFS ファイルシステムのディスクレイアウトを更新します。

Veritas File System のマニュアルページ

本リリースには、VRTSvxfs RPM の一部として、次のマニュアルページが収録されています。各マニュアルページは、/opt/VRTS/man の下の適切なディレクトリにインストールされます。このディレクトリを MANPATH 環境変数に追加してください。ただし、windex データベースは更新されません。新しい VxFS マニュアルページを正しく表示するには、VRTSvxfs のインストール後に windex データベースを更新する必要があります。

catman(1M)のマニュアルページを参照してください。

表 C-13 では、VxFS 固有のセクション 1 マニュアルページを説明します。

表 C-13　　セクション 1 マニュアルページ

セクション 1	説明
fiostat	ファイル I/O の統計データを管理します。
fsmap	VxFS ファイルシステムのエクステント情報を表示します。
getext	VxFS ファイルシステムのエクステント属性を取得します。
setext	VxFS ファイルシステムのファイルにエクステント属性を設定します。
vxcompress	ファイルを圧縮または圧縮解除します。
vxfilesnap	VxFS ファイルシステムのファイルのコピーオンライトコピーを作成します。

表 C-14 では、VxFS 固有のセクション 1M マニュアルページを説明します。

表 C-14　　セクション 1M マニュアルページ

セクション 1M	説明
df_vxfs	VxFS ファイルシステムの空きディスクブロックと i ノードの数を出力します。
fcladm	VxFS FCL を管理します。

セクション 1M	説明
ff_vxfs	VxFS ファイルシステムのファイル名と i ノードの情報を一覧表示します。
fsadm_vxfs	VxFS ファイルシステムのサイズ変更または再構成を実行します。
fsapadm	VxFS 割り当てポリシーを管理します。
fscat_vxfs	VxFS ファイルシステムの内容を表示します。
fscdsadm	オンライン CDS 操作を実行します。
fscdsconv	VxFS ファイルシステムでオフラインによる CDS の移行タスクを実行します。
fscdstask	さまざまな CDS 操作を実行します。
fsck_vxfs	VxFS ファイルシステムの検査と修復を実行します。
fsckptadm	Storage Checkpoint のクォータの作成、削除、変換、設定、表示など、さまざまな管理作業を行います。 -H オプションを使用すると、クォータをヒューマンフレンドリな形式で表示できます。
fsckpt_restore	VxFS Storage Checkpoint からファイルシステムをリストアします。
fsclustadm	クラスタにマウントされた VxFS ファイルシステムを管理します。
fsdbencap	データベースをカプセル化します。
fsdb_vxfs	VxFS ファイルシステムをデバッグします。
fsdedupadm	データ重複排除を管理します。
fsfreeze	VxFS ファイルシステムをフリーズし、ファイルシステム上でユーザーコマンドを実行します。
fsppadm	VxFS 配置ポリシーを管理します。
fstyp_vxfs	指定されたディスクパーティションのファイルシステムタイプを出力します。
fsvmap	VxFS ファイルシステムのボリュームをファイルにマップします。
fsvoladm	VxFS ボリュームを管理します。
glmconfig	GLM（Group Lock Manager）を設定します。この機能は、Storage Foundation Cluster File System High Availability 製品を使う場合にのみ利用できます。
glmdump	クラスタファイルシステムのスタック状態の GLM（Group Lock Manager）ロックを出力します。
mkdstfs	SmartTier ファイルシステム作成ユーティリティ。
mkfs_vxfs	VxFS ファイルシステムを作成します。

セクション 1M	説明
mount_vxfs	VxFS ファイルシステムをマウントします。
ncheck_vxfs	VxFS ファイルシステムの i ノードの番号からパス名を生成します。
quot	VxFS ファイルシステムの所有権のサマリーを出力します。
quotacheck_vxfs	VxFS ファイルシステムクォータの整合性を検証します。
vxdiskusg	VxFS ディスクアカウントデータをユーザー ID ごとに生成します。
vxdump	ファイルシステムの増分ダンプを実行します。
vxedquota	VxFS ファイルシステムのユーザークォータを編集します。
vxenable	特定の VxFS の機能を有効にします。
vxfsconvert	マウントされていないファイルシステムを VxFS に変換するか、または VxFS ディスクレイアウトバージョンを更新します。
vxfsstat	ファイルシステムの統計を表示します。
vxlsino	VxFS パス名の逆引きルックアップを実行します。
vxquot	VxFS ファイルシステムのファイルシステム所有権のサマリーを表示します。
vxquota	VxFS ファイルシステム上のユーザーディスククォータと使用率を表示します。
vxquotaoffvxquotaon	VxFS ファイルシステムのクォータを有効または無効にします。
vxrepquota	VxFS ファイルシステムのクォータのサマリーを出力します。
vxrestore	ファイルシステムの増分リストアを実行します。
vxtunefs	VxFS ファイルシステムをチューニングします。
vxupgrade	マウントされている VxFS ファイルシステムのディスクレイアウトを更新します。

表 C-15 では、VxFS 固有のセクション 3 マニュアルページを説明します。

表 C-15　　セクション 3 マニュアルページ

セクション 3	説明
vxfs_ap_alloc2	fsap_info2 構造を割り当てます。
vxfs_ap_assign_ckpt	Storage Checkpoint のファイルデータとメタデータに割り当てポリシーを割り当てます。

セクション 3	説明
vxfs_ap_assign_ckptchain	VxFS ファイルシステムのすべての Storage Checkpoint に割り当てポリシーを割り当てます。
vxfs_ap_assign_ckptdef	VxFS ファイルシステムの新しい Storage Checkpoint にデフォルトの割り当てポリシーを割り当てます。
vxfs_ap_assign_file	ファイルデータとメタデータの割り当てポリシーを割り当てます。
vxfs_ap_assign_file_pat	パターンベースの割り当てポリシーをディレクトリに割り当てます。
vxfs_ap_assign_fs	指定したファイルシステム内にあるすべてのファイルデータとメタデータに割り当てポリシーを割り当てます。
vxfs_ap_assign_fs_pat	パターンベースの割り当てポリシーをファイルシステムに割り当てます。
vxfs_ap_define	新しい割り当てポリシーを定義します。
vxfs_ap_define2	新しい割り当てポリシーを定義します。
vxfs_ap_enforce_ckpt	指定された割り当てポリシーに一致するように Storage Checkpoint のブロックを再編成します。
vxfs_ap_enforce_ckptchain	VxFS ファイルシステムのすべての Storage Checkpoint に割り当てポリシーを実施します。
vxfs_ap_enforce_file	指定されたファイル内のすべてのブロックをファイル割り当てポリシーに一致させます。
vxfs_ap_enforce_file2	割り当てポリシーと一致するようにファイル内のブロックを再度割り当てます。
vxfs_ap_enforce_range	割り当てポリシーと一致するように、指定された範囲内のファイルのブロックを再度割り当てます。
vxfs_ap_enumerate	すべての割り当てポリシーに関する情報を返します。
vxfs_ap_enumerate2	すべての割り当てポリシーに関する情報を返します。
vxf_ap_free2	1 つ以上の fsap_info2 構造を解放します。
vxfs_ap_query	指定された割り当てポリシーに関する情報を返します。
vxfs_ap_query2	指定された割り当てポリシーに関する情報を返します。
vxfs_ap_query_ckpt	各 Storage Checkpoint の割り当てポリシーに関する情報を返します。
vxfs_ap_query_ckptdef	VxFS ファイルシステムの新しい Storage Checkpoint のデフォルトの割り当てポリシーを取得します。

セクション 3	説明
vxfs_ap_query_file	指定されたファイルに割り当てられている割り当てポリシーに関する情報を返します。
vxfs_ap_query_file_pat	ディレクトリに割り当てられたパターンベースの割り当てポリシーに関する情報を返します。
vxfs_ap_query_fs	指定されたファイルシステムに割り当てられている割り当てポリシーを取得します。
vxfs_ap_query_fs_pat	ファイルシステムに割り当てられたパターンベースの割り当てポリシーに関する情報を返します。
vxfs_ap_remove	指定された割り当てポリシーを削除します。
vxfs_fcl_sync	VxFS FCL に同期ポイントを設定します。
vxfs_fiostats_dump	ファイルとファイル範囲の I/O 統計データを返します。
vxfs_fiostats_getconfig	ファイル範囲の I/O 統計データの設定値を取得します。
vxfs_fiostats_set	ファイル範囲の I/O 統計データを有効または無効にして、統計カウンタをリセットします。
vxfs_get_ioffsets	VxFS i ノードフィールドオフセットを取得します。
vxfs_inotopath	指定された i ノード番号のパス名を返します。
vxfs_inostat	i ノード番号に基づいてファイルの統計情報を取得します。
vxfs_inotofd	i ノード番号に基づいてファイル記述子を取得します。
vxfs_nattr_check vxfs_nattr_fcheck	名前付きデータストリームが実在しているかどうかをチェックします。
vxfs_nattr_link	名前付きデータストリームにリンクします。
vxfs_nattr_open	名前付きデータストリームを開きます。
vxfs_nattr_rename	名前付きデータストリームの名前を変更します。
vxfs_nattr_unlink	名前付きデータストリームを削除します。
vxfs_nattr_utimes	名前付きデータストリームのアクセスと変更の時間を設定します。
vxfs_vol_add	MVS ファイルシステムにボリュームを追加します。
vxfs_vol_clearflags	MVS ファイルシステムのボリューム上で指定されたフラグを消去します。
vxfs_vol_deencapsulate	MVS ファイルシステム内のボリュームのカプセル化を解除します。

セクション 3	説明
vxfs_vol_encapsulate	MVS ファイルシステム内のボリュームをカプセル化します。
vxfs_vol_encapsulate_bias	MVS ファイルシステム内のボリュームをカプセル化します。
vxfs_vol_enumerate	MVS ファイルシステム内のボリュームに関する情報を返します。
vxfs_vol_queryflags	MVS ファイルシステムのボリューム上にあるフラグのクエリーを発行します。
vxfs_vol_remove	MVS ファイルシステムからボリュームを削除します。
vxfs_vol_resize	MVS ファイルシステム内の特定のボリュームのサイズを変更します。
vxfs_vol_setflags	MVS ファイルシステムのボリューム上で指定されたフラグを設定します。
vxfs_vol_stat	MVS ファイルシステム内のコンポーネントボリュームに関する空き領域情報を返します。

表 C-16 では、VxFS 固有のセクション 4 マニュアルページを説明します。

表 C-16　　セクション 4 マニュアルページ

セクション 4	説明
fs_vxfs	VxFS ファイルシステムのボリュームの形式を提供します。
inode_vxfs	VxFS ファイルシステムの i ノードの形式を提供します。
tunefstab	VxFS ファイルシステムのチューニングパラメータテーブルについて説明します。

表 C-17 では、VxFS 固有のセクション 7 マニュアルページを説明します。

表 C-17　　セクション 7 マニュアルページ

セクション 7	説明
vxfsio	VxFS ファイルシステムの制御機能を説明します。

SmartIO コマンドリファレンス

表 C-18 では、SmartIO 機能を使うためのコマンドのリストを示します。

SmartIO について詳しくは、『Veritas InfoScale SmartIO for Solid State Drives ソリューションガイド』を参照してください。

sfcache（1M）マニュアルページを参照してください。

表 C-18 SmartIO コマンドリファレンス

コマンド	説明
sfcache app	指定したテンプレート名を適用します。
sfcache create	キャッシュ領域を作成します。
sfcache delete	指定したキャッシュ領域を削除します。
sfcache disable	指定したデータオブジェクトに対してキャッシュを無効にします。
sfcache enable	指定したデータオブジェクトに対してキャッシュを有効にします。
sfcache flush	このファイルシステムまたはキャッシュのライトバックデータをフラッシュします。
sfcache list	キャッシュされるファイルシステムまたはボリュームとそのキャッシュ使用率を表示します。
sfcache load	指定したファイルをキャッシュ領域にロードします。
sfcache maxsize	キャッシュ用にすでにプロビジョニングされているデバイスの空き容量を表示します。
sfcache offline	VxFS または VxVM によるキャッシュ領域の使用を停止します。
sfcache online	キャッシュ領域が利用可能であることを明示的に示します。
sfcache pin	ファイルまたはディレクトリが削除、切り捨て、固定解除されるまでそのファイルまたはディレクトリを保留することをマークします。
sfcache purge	指定したファイルシステムのキャッシュされた内容を削除します。
sfcache resize	指定したキャッシュ領域のサイズを変更します。
sfcache restore-access	ライトバックデータが欠けているファイルへの読み取りまたは書き込みアクセスを有効にします。このコマンドにより、欠けているデータが復元されることはありません。

コマンド	説明
sfcache rmdev	キャッシュの使用からデバイスまたは複数のデバイスを削除します。
sfcache set	指定した属性の値を設定します。
sfcache stat	キャッシュヒット率、ミス、平均読み取り/書き込みレイテンシを含むキャッシュの統計を表示します。
sfcache unpin	固定された状態からファイルまたはディレクトリを削除します。

D

スタータデータベースの作成

この付録では以下の項目について説明しています。

データベースの作成

共有 RAW VxVM ボリューム上での Oracle のデータベース表領域の作成

開始する前に、次の前提条件に注意してください。

- CRS デーモンが動作している必要があります。CRS のステータスを検証するには、次のとおりに入力します。

```
# $CRS_HOME/bin/crsctl status resource -t
```

- `ping` コマンドを使って、各ノードのすべてのプライベート IP アドレスが動作していることを確認します。

共有 RAW VxVM ボリューム上でのデータベース表領域の作成

ここでは、共有 RAW VxVM ボリューム上でデータベース表領域の作成方法（オプション 1）について説明します。

共有 RAW VxVM ボリューム上でデータベース表領域を作成するには（オプション 1）

1 任意のクラスタノードで root としてログインします。Oracle データベース表領域用の共有ディスクグループを作成するために使用できる空きディスクを見つけ、次のとおりに入力します。

```
# vxdisk -o alldgs list
DEVICE TYPE          DISK   GROUP     STATUS
sda    auto:none     -      -         online invalid
sdb    auto:none     -      -         online invalid
sdc    auto:cdsdisk  -      tempdg    online shared
sdd    auto:cfsdisk  -      ocrvotedg online shared
sde    auto:cdsdisk  -      -         online shared
sdf    auto:cdsdisk  -      -         online shared
```

この出力例は、共有ディスク sde と sdf が空いていて、Oracle データベース表領域のために使用できることを示します。

2 共有ディスクグループを作成します。

```
# vxdg -s init oradatadg sde sdf
```

3 必須表領域のそれぞれに対するボリュームを共有グループに作成します。

表領域の必要条件を決定するためには、Oracle データベースリリース個別の *Oracle* のマニュアルを参照してください。

次のように実行します。

```
# vxassist -g oradatadg make VRT_sys1 1000M
# vxassist -g oradatadg make VRT_sys2 2 10M
.
.
.
```

4 Oracle データを格納するボリュームに対するアクセス権限とアクセスモードを定義します。$ORACLE_HOME/raw_config に記載されている各ボリュームでは、vxedit コマンドを使用します。

```
# vxedit -g disk_group set group=group user=user mode=660
     volume
```

vxedit（1M）のマニュアルページを参照してください。

次のように実行します。

```
# vxedit -g oradatadg set group=oinstall user=oracle mode=660 ¥
VRT_sys1
```

この例では、VRT_sys1 は 1 つのボリュームの名前です。oradatadg で各ボリュームに対するアクセス権限とアクセスモードを定義するためコマンドを繰り返します。

5 データベースを作成します。

Oracle のマニュアルを参照してください。