Veritas InfoScale™ 7.4.2 Storage and Availability Management for DB2 Databases - AIX, Linux

VERITAS™

Last updated: 2020-06-01

Legal Notice

Copyright © 2020 Veritas Technologies LLC. All rights reserved.

Veritas and the Veritas Logo are trademarks or registered trademarks of Veritas Technologies LLC or its affiliates in the U.S. and other countries. Other names may be trademarks of their respective owners.

This product may contain third-party software for which Veritas is required to provide attribution to the third-party (“Third-Party Programs”). Some of the Third-Party Programs are available under open source or free software licenses. The License Agreement accompanying the Software does not alter any rights or obligations you may have under those open source or free software licenses. Refer to the third-party legal notices document accompanying this Veritas product or available at:
https://www.veritas.com/about/legal/license-agreements

The product described in this document is distributed under licenses restricting its use, copying, distribution, and decompilation/reverse engineering. No part of this document may be reproduced in any form by any means without prior written authorization of Veritas Technologies LLC and its licensors, if any.

THE DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. VERITAS TECHNOLOGIES LLC SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION. THE INFORMATION CONTAINED IN THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT NOTICE.

The Licensed Software and Documentation are deemed to be commercial computer software as defined in FAR 12.212 and subject to restricted rights as defined in FAR Section 52.227-19 "Commercial Computer Software - Restricted Rights" and DFARS 227.7202, et seq. "Commercial Computer Software and Commercial Computer Software Documentation," as applicable, and any successor regulations, whether delivered by Veritas as on premises or hosted services. Any use, modification, reproduction release, performance, display or disclosure of the Licensed Software and Documentation by the U.S. Government shall be solely in accordance with the terms of this Agreement.

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

Technical Support

Technical Support maintains support centers globally. All support services will be delivered in accordance with your support agreement and the then-current enterprise technical support policies. For information about our support offerings and how to contact Technical Support, visit our website:
https://www.veritas.com/support

You can manage your Veritas account information at the following URL:
https://my.veritas.com

If you have questions regarding an existing support agreement, please email the support agreement administration team for your region as follows:

Worldwide (except Japan)	CustomerCare@veritas.com
Japan	CustomerCare_Japan@veritas.com

Documentation

Make sure that you have the current version of the documentation. Each document displays the date of the last update on page 2. The latest documentation is available on the Veritas website:
https://sort.veritas.com/documents

Documentation feedback

Your feedback is important to us. Suggest improvements or report errors or omissions to the documentation. Include the document title, document version, chapter title, and section title of the text on which you are reporting. Send feedback to:
infoscaledocs@veritas.com

You can also see documentation information or ask a question on the Veritas community site:
http://www.veritas.com/community/

Veritas Services and Operations Readiness Tools (SORT)

Veritas Services and Operations Readiness Tools (SORT) is a website that provides information and tools to automate and simplify certain time-consuming administrative tasks. Depending on the product, SORT helps you prepare for installations and upgrades, identify risks in your datacenters, and improve operational efficiency. To see what services and tools SORT provides for your product, see the data sheet:
https://sort.veritas.com/data/support/SORT_Data_Sheet.pdf

Contents

Section 1 Storage Foundation High Availability (SFHA) management solutions for DB2 databases 11

Chapter 1 Overview of Storage Foundation for Databases 12

Introducing Storage Foundation High Availability (SFHA) Solutions for DB2 12
About Veritas File System 13
About the Veritas File System intent log 14
About extents 15
About file system disk layouts 15
About Veritas Volume Manager 16
About Dynamic Multi-Pathing (DMP) 17
About Cluster Server 17
About Cluster Server agents 17
About Veritas InfoScale Operations Manager 18
Feature support for DB2 across Veritas InfoScale 7.4.2 products 18
Use cases for Veritas InfoScale products 21

Section 2 Deploying DB2 with Veritas InfoScale products 25

Chapter 2 Deployment options for DB2 in a Storage Foundation environment 26

DB2 deployment options in a Veritas InfoScale environment 26
DB2 on a single system with Storage Foundation 27
DB2 on a single system with off-host in a Storage Foundation environment 28
DB2 in a highly available cluster with Storage Foundation High Availability 29
DB2 in a parallel cluster with SF Cluster File System HA 31

Deploying DB2 and Storage Foundation in a virtualization environment 33
Deploying DB2 with Storage Foundation SmartMove and Thin Provisioning 34

Chapter 3 Deploying DB2 with Storage Foundation 35

Tasks for deploying DB2 databases 35
About selecting a volume layout for deploying DB2 36
Setting up disk group for deploying DB2 37
Disk group configuration guidelines for deploying DB2 38
Creating volumes for deploying DB2 39
Volume configuration guidelines for deploying DB2 39
Creating VxFS file system for deploying DB2 40
File system creation guidelines for deploying DB2 41
Mounting the file system for deploying DB2 41
Installing DB2 and creating database 42

Chapter 4 Deploying DB2 in an off-host configuration with Storage Foundation 43

Requirements for an off-host database configuration 43

Chapter 5 Deploying DB2 with High Availability 45

Tasks for deploying DB2 in an HA configuration 45
Configuring VCS to make the database highly available 45

Section 3 Configuring Storage Foundation for Database (SFDB) tools 46

Chapter 6 Configuring and managing the Storage Foundation for Databases repository database 47

About the Storage Foundation for Databases (SFDB) repository 47
Requirements for Storage Foundation for Databases (SFDB) tools 48
Storage Foundation for Databases (SFDB) tools availability 48
Configuring the Storage Foundation for Databases (SFDB) tools repository 49
Locations for the SFDB repository 49

Updating the Storage Foundation for Databases (SFDB) repository after adding a node 49
Updating the Storage Foundation for Databases (SFDB) repository after removing a node 50
Removing the Storage Foundation for Databases (SFDB) repository 50

Chapter 7 Configuring authentication for Storage Foundation for Databases (SFDB) tools 52

Configuring vxdbd for SFDB tools authentication 52
Adding nodes to a cluster that is using authentication for SFDB tools 53
Authorizing users to run SFDB commands 54

Section 4 Improving DB2 database performance 56

Chapter 8 About database accelerators 57

About Veritas InfoScale product components database accelerators 57

Chapter 9 Improving database performance with Quick I/O 60

About Quick I/O 60
How Quick I/O improves database performance 61
Tasks for setting up Quick I/O in a database environment 63
Preallocating space for Quick I/O files using the setext command 64
Accessing regular VxFS files as Quick I/O files 66
Converting DB2 containers to Quick I/O files 68
About sparse files 72
Displaying Quick I/O status and file attributes 73
Extending a Quick I/O file 75
Monitoring tablespace free space with DB2 and extending tablespace containers 77
Recreating Quick I/O files after restoring a database 80
Disabling Quick I/O 81

Chapter 10 Improving DB2 database performance with Veritas Concurrent I/O 83

About Concurrent I/O 83
How Concurrent I/O works 84
Tasks for enabling and disabling Concurrent I/O 84
Enabling Concurrent I/O for DB2 84
Disabling Concurrent I/O for DB2 87

Section 5 Using point-in-time copies 88

Chapter 11 Understanding point-in-time copy methods 89

About point-in-time copies 89
When to use point-in-time copies 90
About Storage Foundation point-in-time copy technologies 91
Point-in-time copy solutions supported by SFDB tools 92
About snapshot modes supported by Storage Foundation for Databases (SFDB) tools 93
Volume-level snapshots 93
Persistent FastResync of volume snapshots 94
Data integrity in volume snapshots 94
Third-mirror break-off snapshots 95
Storage Checkpoints 95
How Storage Checkpoints differ from snapshots 96
How a Storage Checkpoint works 97
About Database Rollbacks using Storage Checkpoints 100
Storage Checkpoints and Rollback process 101
Storage Checkpoint space management considerations 102

Chapter 12 Considerations for DB2 point-in-time copies 104

Considerations for DB2 database layouts 104
Supported DB2 configurations 105

Chapter 13 Administering third-mirror break-off snapshots 106

Database FlashSnap for cloning 106
Database FlashSnap advantages 107
Preparing hosts and storage for Database FlashSnap 107
Setting up hosts 107
Creating a snapshot mirror of a volume or volume set used by the database 108

Creating a clone of a database by using Database FlashSnap 111
Resynchronizing mirror volumes with primary volumes 116
Cloning a database on the secondary host 118

Chapter 14 Administering Storage Checkpoints 120

About Storage Checkpoints 120
Database Storage Checkpoints for recovery 121
Advantages and limitations of Database Storage Checkpoints 122
Creating a Database Storage Checkpoint 122
Deleting a Database Storage Checkpoint 123
Mounting a Database Storage Checkpoint 124
Unmounting a Database Storage Checkpoint 124
Creating a database clone using a Database Storage Checkpoint 125
Restoring database from a Database Storage Checkpoint 126
Gathering data for offline-mode Database Storage Checkpoints 127

Chapter 15 Backing up and restoring with Netbackup in an SFHA environment 129

About Veritas NetBackup 129
About using Veritas NetBackup for backup and restore for DB2 130
Using NetBackup in an SFHA Solutions product environment 131
Clustering a NetBackup Master Server 131
Backing up and recovering a VxVM volume using NetBackup 132
Recovering a VxVM volume using NetBackup 134

Section 6 Optimizing storage costs for DB2 135

Chapter 16 Understanding storage tiering with SmartTier 136

About SmartTier 136
About VxFS multi-volume file systems 138
About VxVM volume sets 139
About volume tags 139
SmartTier file management 139
SmartTier sub-file object management 140
SmartTier in a High Availability (HA) environment 140

Chapter 17 SmartTier use cases for DB2 141

SmartTier use cases for DB2 141
Relocating old archive logs to tier two storage using SmartTier 142
Relocating inactive tablespaces or segments to tier two storage 144
Relocating active indexes to premium storage 147
Relocating all indexes to premium storage 148

Section 7 Storage Foundation for Databases administrative reference 152

Chapter 18 Storage Foundation for Databases command reference 153

vxsfadm command reference 153
FlashSnap reference 156
FlashSnap configuration parameters 156
FlashSnap supported operations 158
Database Storage Checkpoints reference 159
Database Storage Checkpoints configuration parameters 159
Database Storage Checkpoints supported operations 161

Chapter 19 Tuning for Storage Foundation for Databases 163

Additional documentation 163
About tuning Veritas Volume Manager (VxVM) 164
About obtaining volume I/O statistics 164
About tuning VxFS 165
How monitoring free space works 165
How tuning VxFS I/O parameters works 167
About tunable VxFS I/O parameters 167
About obtaining file I/O statistics using the Quick I/O interface 171
About I/O statistics data 171
About I/O statistics 173
About tuning DB2 databases 173
DB2_USE_PAGE_CONTAINER_TAG 173
DB2_PARALLEL_IO 174
PREFETCHSIZE and EXTENTSIZE 175
INTRA_PARALLEL 176
NUM_IOCLEANERS 176
NUM_IOSERVERS 177

CHNGPGS_THRESH 177
Table scans 177
Asynchronous I/O 177
Buffer pools 178
Memory allocation 178
TEMPORARY tablespaces 178
DMS containers 178
Data, indexes, and logs 179
Database statistics 179
About tuning AIX Virtual Memory Manager 179

Chapter 20 Troubleshooting SFDB tools 183

About troubleshooting Storage Foundation for Databases (SFDB) tools 183
Running scripts for engineering support analysis for SFDB tools 184
Storage Foundation for Databases (SFDB) tools log files 184
About the vxdbd daemon 184
Starting and stopping vxdbd 184
Configuring listening port for the vxdbd daemon 185
Limiting vxdbd resource usage 185
Configuring encryption ciphers for vxdbd 186
Troubleshooting vxdbd 186
Resources for troubleshooting SFDB tools 187
SFDB logs 187
SFDB error messages 188
SFDB repository and repository files 188
Upgrading Storage Foundation for Databases (SFDB) tools from 5.0.x to 7.4.2 (2184482) 189

Section 1

Storage Foundation High Availability (SFHA) management solutions for DB2 databases

- Chapter 1. Overview of Storage Foundation for Databases

Chapter 1

Overview of Storage Foundation for Databases

This chapter includes the following topics:

- Introducing Storage Foundation High Availability (SFHA) Solutions for DB2
- About Veritas File System
- About Veritas Volume Manager
- About Dynamic Multi-Pathing (DMP)
- About Cluster Server
- About Cluster Server agents
- About Veritas InfoScale Operations Manager
- Feature support for DB2 across Veritas InfoScale 7.4.2 products
- Use cases for Veritas InfoScale products

Introducing Storage Foundation High Availability (SFHA) Solutions for DB2

This guide documents the deployment and key use cases of the SFDB tools with Storage Foundation High Availability (SFHA) Solutions products in DB2 database environments. It is a supplemental guide to be used in conjunction with SFHA Solutions product guides.

The Storage Foundation for Databases tools provide enhanced management options for DB2 databases. The SFDB tools provide enhanced ease-of-use commands which can be run by a database administrator without root privileges to optimize

storage for an DB2 database environment. This guide documents the deployment and use of the SFDB tools included with SFHA Solutions enterprise products.

Note: The commands used for the Red Hat Enterprise Linux (RHEL) operating system in this document also apply to supported RHEL-compatible distributions.

About Veritas File System

A file system is simply a method for storing and organizing computer files and the data they contain to make it easy to find and access them. More formally, a file system is a set of abstract data types (such as metadata) that are implemented for the storage, hierarchical organization, manipulation, navigation, access, and retrieval of data.

Veritas File System (VxFS) was the first commercial journaling file system. With journaling, metadata changes are first written to a log (or journal) then to disk. Since changes do not need to be written in multiple places, throughput is much faster as the metadata is written asynchronously.

VxFS is also an extent-based, intent logging file system. VxFS is designed for use in operating environments that require high performance and availability and deal with large amounts of data.

The maximum size of the file system you can create depends on the block size.

Block Size	Currently-Supported Maximum File System Size
1024 bytes	68,719,472,624 sectors (≈32 TB)
2048 bytes	137,438,945,248 sectors (≈64 TB)
4096 bytes	274,877,890,496 sectors (≈128 TB)
8192 bytes	549,755,780,992 sectors (≈256 TB)

VxFS major components include:

File system logging	About the Veritas File System intent log
Extents	About extents
File system disk layouts	About file system disk layouts

About the Veritas File System intent log

Most file systems rely on full structural verification by the `fsck` utility as the only means to recover from a system failure. For large disk configurations, this involves a time-consuming process of checking the entire structure, verifying that the file system is intact, and correcting any inconsistencies. VxFS provides fast recovery with the VxFS intent log and VxFS intent log resizing features.

VxFS reduces system failure recovery times by tracking file system activity in the VxFS intent log. This feature records pending changes to the file system structure in a circular intent log. The intent log recovery feature is not readily apparent to users or a system administrator except during a system failure. By default, VxFS file systems log file transactions before they are committed to disk, reducing time spent recovering file systems after the system is halted unexpectedly.

During system failure recovery, the VxFS `fsck` utility performs an intent log replay, which scans the intent log and nullifies or completes file system operations that were active when the system failed. The file system can then be mounted without requiring a full structural check of the entire file system. Replaying the intent log might not completely recover the damaged file system structure if there was a disk hardware failure; hardware problems might require a complete system check using the `fsck` utility provided with VxFS.

The `mount` command automatically runs the VxFS `fsck` command to perform an intent log replay if the mount command detects a dirty log in the file system. This functionality is only supported on a file system mounted on a Veritas Volume Manager (VxVM) volume, and is supported on cluster file systems.

See the `fsck_vxfs`(1M) manual page and `mount_vxfs`(1M) manual page.

The VxFS intent log is allocated when the file system is first created. The size of the intent log is based on the size of the file system—the larger the file system, the larger the intent log. You can resize the intent log at a later time by using the `fsadm` commnad.

See the `fsadm_vxfs`(1M) manual page.

The maximum default intent log size for disk layout Version 7 or later is 256 megabytes.

Note: Inappropriate sizing of the intent log can have a negative impact on system performance.

About extents

An extent is a contiguous area of storage in a computer file system, reserved for a file. When starting to write to a file, a whole extent is allocated. When writing to the file again, the data continues where the previous write left off. This reduces or eliminates file fragmentation. An extent is presented as an address-length pair, which identifies the starting block address and the length of the extent (in file system or logical blocks). Since Veritas File System (VxFS) is an extent-based file system, addressing is done through extents (which can consist of multiple blocks) rather than in single-block segments. Extents can therefore enhance file system throughput.

Extents allow disk I/O to take place in units of multiple blocks if storage is allocated in contiguous blocks. For sequential I/O, multiple block operations are considerably faster than block-at-a-time operations; almost all disk drives accept I/O operations on multiple blocks.

Extent allocation only slightly alters the interpretation of addressed blocks from the inode structure compared to block-based inodes. A VxFS inode references 10 direct extents, each of which are pairs of starting block addresses and lengths in blocks.

Disk space is allocated in 512-byte sectors to form logical blocks. VxFS supports logical block sizes of 1024, 2048, 4096, and 8192 bytes. The default block size is 1 KB for file system sizes of up to 2 TB, and 8 KB for file system sizes 2 TB or larger.

About file system disk layouts

The disk layout is the way file system information is stored on disk. On Veritas File System (VxFS), several disk layout versions are supported to provide new features and specific UNIX environments.

You can use one of the following commands to upgrade the disk layout version.

`vxupgrade`	Upgrades an existing VxFS file system to a supported disk layout version while the file system remains online. See the `vxupgrade`(1M) manual page.
`vxfsconvert`	Upgrades a no-longer supported disk layout version to a supported version while the file system is not mounted. The `vxfsconvert` command can also be used to convert a native file system (ext2, ext3, and ext4) to VxFS, while the file system is not mounted. See the `vxfsconvert`(1M) manual page.

Table 1-1 lists the supported disk layout versions.

Table 1-1 Supported disk layout versions

Version	Supported features
Version 12	Version 12 supports 128 node cluster on CFS
Version 13	■ Added support for WORM ■ Clone creation performance improvement when extended file attributes are used
Version 14	Supports SmartIO FEL-based caching
Version 15	■ Performance enhancements in SELinux attribute storage and retrieval ■ Secure Clock support for WORM files
Version 16	Audit Logging support for WORM files

Currently, only versions 12, 13, 14, 15, and 16 can be created and mounted. Versions 6, 7, 8, 9, 10, and 11 can be mounted, but only for upgrading to a supported version.

About Veritas Volume Manager

Veritas™ Volume Manager (VxVM) by Veritas is a storage management subsystem that allows you to manage physical disks and logical unit numbers (LUNs) as logical devices called volumes. A VxVM volume appears to applications and the operating system as a physical device on which file systems, databases, and other managed data objects can be configured.

VxVM provides easy-to-use online disk storage management for computing environments and Storage Area Network (SAN) environments. By supporting the Redundant Array of Independent Disks (RAID) model, VxVM can be configured to protect against disk and hardware failure, and to increase I/O throughput. Additionally, VxVM provides features that enhance fault tolerance and fast recovery from disk failure or storage array failure.

VxVM overcomes restrictions imposed by hardware disk devices and by LUNs by providing a logical volume management layer. This allows volumes to span multiple disks and LUNs.

VxVM provides the tools to improve performance and ensure data availability and integrity. You can also use VxVM to dynamically configure storage while the system is active.

About Dynamic Multi-Pathing (DMP)

Dynamic Multi-Pathing (DMP) provides multi-pathing functionality for the operating system native devices that are configured on the system. DMP creates DMP metadevices (also known as DMP nodes) to represent all the device paths to the same physical LUN.

DMP metadevices support the OS native logical volume manager (LVM). You can create LVM volumes and volume groups on DMP metadevices.

DMP supports the LVM volume devices that are used as the paging devices.

Veritas Volume Manager (VxVM) volumes and disk groups can co-exist with LVM volumes and volume groups. But, each device can only support one of the types. If a disk has a VxVM label, then the disk is not available to LVM. Similarly, if a disk is in use by LVM, then the disk is not available to VxVM.

About Cluster Server

Cluster Server (VCS) is a clustering solution that provides the following benefits:

- Minimizes downtime.
- Facilitates the consolidation and the failover of servers.
- Effectively manages a wide range of applications in heterogeneous environments.

Before you install the product, read the *Veritas InfoScale 7.4.2 Release Notes*.

To configure the product, follow the instructions in the *Cluster Server Generic Application Agent Configuration Guide.*

About Cluster Server agents

Veritas InfoScale agents provide high availability for specific resources and applications. Each agent manages resources of a particular type. For example, the DB2 agent manages DB2 databases. Typically, agents start, stop, and monitor resources and report state changes.

Before you install VCS agents, review the configuration guide for the agent.

In addition to the agents that are provided in this release, other agents are available through an independent Veritas InfoScale offering called the Cluster Server Agent Pack. The agent pack includes the currently shipping agents and is re-released quarterly to add the new agents that are now under development.

Contact your Veritas InfoScale sales representative for the following details:

- Agents that are included in the agent pack

- Agents under development
- Agents available through Veritas InfoScale Consulting Services

You can download the latest agents from the Veritas Services and Operations Readiness Tools website:

http://sort.veritas.com/agents

About Veritas InfoScale Operations Manager

Veritas InfoScale Operations Manager provides a centralized management console for Veritas InfoScale products. You can use Veritas InfoScale Operations Manager to monitor, visualize, and manage storage resources and generate reports.

Veritas recommends using Veritas InfoScale Operations Manager to manage Storage Foundation and Cluster Server environments.

You can download Veritas InfoScale Operations Manager from https://sort.veritas.com/.

Refer to the Veritas InfoScale Operations Manager documentation for installation, upgrade, and configuration instructions.

The Veritas Enterprise Administrator (VEA) console is no longer packaged with Veritas InfoScale products. If you want to continue using VEA, a software version is available for download from https://www.veritas.com/product/storage-management/infoscale-operations-manager. Storage Foundation Management Server is deprecated.

Feature support for DB2 across Veritas InfoScale 7.4.2 products

Veritas InfoScale Storage solutions and use cases for DB2 are based on the shared management features of Veritas InfoScale Storage Foundation and High Availability (SFHA) Solutions products. Clustering features are available separately through Cluster Server (VCS) as well as through the SFHA Solutions products.

Table 1-2 lists the features supported across SFHA Solutions products. Table 1-3 lists the high availability and disaster recovery features available in VCS.

Table 1-2 Storage management features in Veritas InfoScale products

Storage management feature	Veritas InfoScale Foundation	Veritas InfoScale Storage	Veritas InfoScale Availability	Veritas InfoScale Enterprise
Quick I/O **Note:** Not supported on Linux	N	Y **Note:** Supported in the cluster-exclusive mode which allows access to the feature only from a single node.	N	Y
Cached Quick I/O **Note:** Not supported on Linux	N	Y	N	Y
Concurrent I/O	N	Y	N	Y
Compression	N	Y	N	Y
Flexible Storage Sharing	N	Y	N	Y
SmartIO FS (Read Caching)	N	Y	N	Y
SmartIO FS (Writeback Caching)	N	Y	N	Y
SmartMove	N	Y	N	Y
SmartTier	N	Y	N	Y
Thin Reclamation	N	Y	N	Y
Portable Data Containers	N	Y	N	Y
Database FlashSnap	N	Y	N	Y
Database Storage Checkpoints	N	Y	N	Y
Advanced support for virtual storage	Y	Y	Y	Y
Clustering features for high availability (HA)	N	N	Y	N

Table 1-2 Storage management features in Veritas InfoScale products *(continued)*

Storage management feature	Veritas InfoScale Foundation	Veritas InfoScale Storage	Veritas InfoScale Availability	Veritas InfoScale Enterprise
Disaster recovery features (HA/DR)	N	N	Y	N
Dynamic Multi-pathing	Y	Y	Y	Y

Table 1-3 Availability management features in Veritas InfoScale solutions products

Availability management feature	VCS HA/DR
Clustering for high availability (HA)	Y
Database and application/ISV agents	Y
Advanced failover logic	Y
Data integrity protection with I/O fencing	Y
Advanced virtual machines support	Y
Virtual Business Services	Y
Campus or stretch cluster	Y
Global clustering (GCO)	Y

Notes:

- Y=Feature is included in your license.
- N=Feature is not supported with your license.

Notes:

- SmartTier is an expanded and renamed version of Dynamic Storage Tiering (DST).
- All features listed in Table 1-2 and Table 1-3 are supported on AIX, Linux except as noted. Consult specific product documentation for information on supported operating systems.

Use cases for Veritas InfoScale products

Veritas InfoScale Storage Foundation and High Availability (SFHA) Solutions product components and features can be used individually and in concert to improve performance, resilience and ease of management for your storage and applications. This guide documents key use cases for the management features of SFHA Solutions products.

Note: The commands used for the Red Hat Enterprise Linux (RHEL) operating system in this document also apply to supported RHEL-compatible distributions.

Table 1-4 Key use cases for SFHA Solutions products

Use case	Veritas InfoScale feature
Improve database performance using SFHA Solutions database accelerators to enable your database to achieve the speed of raw disk while retaining the management features and convenience of a file system. See "About Veritas InfoScale product components database accelerators" on page 57.	Quick I/O See "About Quick I/O" on page 60. Cached Quick I/O **Note:** Quick I/O and Cached Quick I/O are not supported on Linux. Concurrent I/O See "About Concurrent I/O" on page 83.
Protect your data using SFHA Solutions Flashsnap, Storage Checkpoints, and NetBackup point-in-time copy methods to back up and recover your data. See "About point-in-time copies" on page 89.	FlashSnap Storage Checkpoints NetBackup with SFHA Solutions
Process your data off-host to avoid performance loss to your production hosts by using SFHA Solutions volume snapshots.	FlashSnap
Optimize copies of your production database for test, decision modeling, and development purposes by using SFHA Solutions point-in-time copy methods.	FlashSnap

Table 1-4 Key use cases for SFHA Solutions products *(continued)*

Use case	Veritas InfoScale feature
Make file level point-in-time snapshots using SFHA Solutions space-optimized FileSnap when you need finer granualarity for your point-in-time copies than file systems or volumes. You can use FileSnap for cloning virtual machines.	FileSnap
Maximize your storage utilization using SFHA Solutions SmartTier to move data to storage tiers based on age, priority, and access rate criteria. See "About SmartTier" on page 136.	SmartTier
Maximize storage utilization for data redundancy, high availability, and disaster recovery, without physically shared storage.	Flexible Storage Sharing
Improve your data efficiency on solid state drives (SSDs) through I/O caching using advanced, customizable hueristics to determine which data to cache and how that data gets removed from the cache.	SmartIO read caching for applications running on VxVM volumes SmartIO read caching for applications running on VxFS file systems SmartIO write caching for applications running on VxFS file systems SmartIO caching for databases on VxFS file systems SmartIO caching for databases on VxVM volumes SmartIO write-back caching for databases is not supported on SFRAC See the *Veritas InfoScale 7.4.2 SmartIO for Solid-State Drives Solutions Guide*.
Plan a maintenance of virtual machines in a vSphere environment for a planned failover and recovery of application during unplanned failure using the Just In Time Availability solution.	Just In Time Availability solution

Table 1-4 Key use cases for SFHA Solutions products *(continued)*

Use case	Veritas InfoScale feature
Improve the native and optimized format of your storage devices using the Veritas InfoScale solution which provides support with the advanced format or 4K (4096 bytes) sector devices (formatted with 4KB) in storage environments.	Veritas InfoScale 4K sector device support solution
Multiple parallel applications in a data warehouse that require flexible sharing of data such as ETL pipeline, where output of one stage becomes input for the next stage. (for example, accounting system needs to combine data from different applications such as sales, payroll and purchasing)	Verita InfoScale application isolation More information: Application isolation in CVM environments with disk group sub-clustering Enabling the application isolation feature in CVM environments Disabling the application isolation feature in a CVM cluster Setting the sub-cluster node preference value for master failover Changing the disk group master manually For information, see the *Storage Foundation Cluster File System High Availability Administrator's Guide*.
Relax complete zoning requirement of SAN storage to all CVM nodes. This enables merging of independent clusters for better manageability.	Verita InfoScale application isolation More information: Application isolation in CVM environments with disk group sub-clustering Enabling the application isolation feature in CVM environments Disabling the application isolation feature in a CVM cluster Setting the sub-cluster node preference value for master failover Changing the disk group master manually For information, see the *Storage Foundation Cluster File System High Availability Administrator's Guide*.

Table 1-4 Key use cases for SFHA Solutions products *(continued)*

Use case	Veritas InfoScale feature
Enabling multiple independent clustered applications to use a commonly shared pool of scalable DAS storage. This facilitates adding of storage-only nodes to cluster for growing storage capacity and compute nodes for dedicated application use.	Verita InfoScale application isolation More information: Application isolation in CVM environments with disk group sub-clustering Enabling the application isolation feature in CVM environments Disabling the application isolation feature in a CVM cluster Setting the sub-cluster node preference value for master failover Changing the disk group master manually For information, see the *Storage Foundation Cluster File System High Availability Administrator's Guide*.

Section 2

Deploying DB2 with Veritas InfoScale products

- Chapter 2. Deployment options for DB2 in a Storage Foundation environment
- Chapter 3. Deploying DB2 with Storage Foundation
- Chapter 4. Deploying DB2 in an off-host configuration with Storage Foundation
- Chapter 5. Deploying DB2 with High Availability

Chapter 2

Deployment options for DB2 in a Storage Foundation environment

This chapter includes the following topics:

- DB2 deployment options in a Veritas InfoScale environment
- DB2 on a single system with Storage Foundation
- DB2 on a single system with off-host in a Storage Foundation environment
- DB2 in a highly available cluster with Storage Foundation High Availability
- DB2 in a parallel cluster with SF Cluster File System HA
- Deploying DB2 and Storage Foundation in a virtualization environment
- Deploying DB2 with Storage Foundation SmartMove and Thin Provisioning

DB2 deployment options in a Veritas InfoScale environment

You can deploy DB2 with Storage Foundation High Availability Solutions (SFHA Solutions) products in the following setups:

- DB2 on a single system in a Storage Foundation environment
- DB2 on a single system with off-host in a Storage Foundation environment
- DB2 in a cluster to make it highly available with Storage Foundation High Availability (SFHA)

- DB2 with Storage Foundation Cluster Server High Availability (SFCFSHA)

Storage Foundation for Databases (SFDB) tools support all of these setups.

DB2 on a single system with Storage Foundation

If you are deploying DB2 databases with Storage Foundation, your setup configuration will reflect the following conditions:

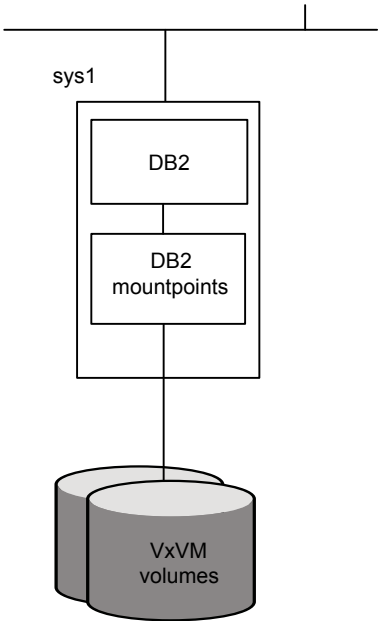
- The DB2 databases are set up on *sys1* with SF.
- The DB2 databases are online on *sys1*.
- You must run the SFDB tools commands on *sys1*.

For information about Storage Foundation for Databases (SFDB) repository or repository database:

See "About the Storage Foundation for Databases (SFDB) repository" on page 47.

Figure 2-1 shows DB2 on single system deployment in a Storage Foundation environment.

Figure 2-1 DB2 database on a single system with Storage Foundation



DB2 on a single system with off-host in a Storage Foundation environment

If you are deploying single instance DB2 with Storage Foundation in an off-host setup, your configuration will reflect the following conditions:

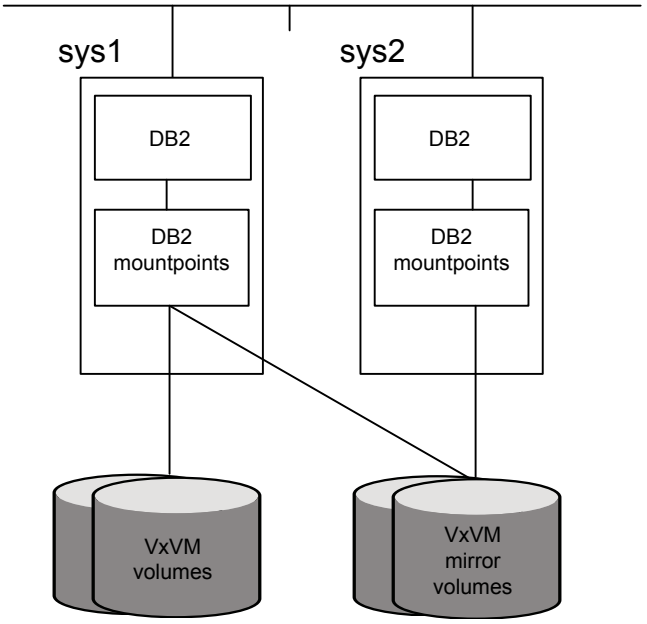
- The DB2 databases are set up on *sys1* with SF.
- The DB2 databases are online on *sys1*.
- *sys1* and *sys2* share the same storage.

For information about Storage Foundation for Databases (SFDB) repository or repository database:

See "About the Storage Foundation for Databases (SFDB) repository" on page 47.

Figure 2-2 shows DB2 on single system with off-host deployment in a Storage Foundation environment.

Figure 2-2 DB2 on a single system with off-host setup in Storage Foundation environment



DB2 in a highly available cluster with Storage Foundation High Availability

If you are deploying DB2 with Storage Foundation High Availability (SFHA), your setup configuration will reflect the following conditions:

- A highly available DB2 database is set up on *sys1* and *sys2* with SFHA
- The database and datafiles are online on *sys1*.
- You must run the SFDB tools commands on *sys1* where the database is online.

For information about Storage Foundation for Databases (SFDB) repository or repository database:

See "About the Storage Foundation for Databases (SFDB) repository" on page 47.

Figure 2-3 and Figure 2-4 show a single system DB2 failover deployment in a Storage Foundation environment.

Figure 2-3 DB2 on a single system with SFHA

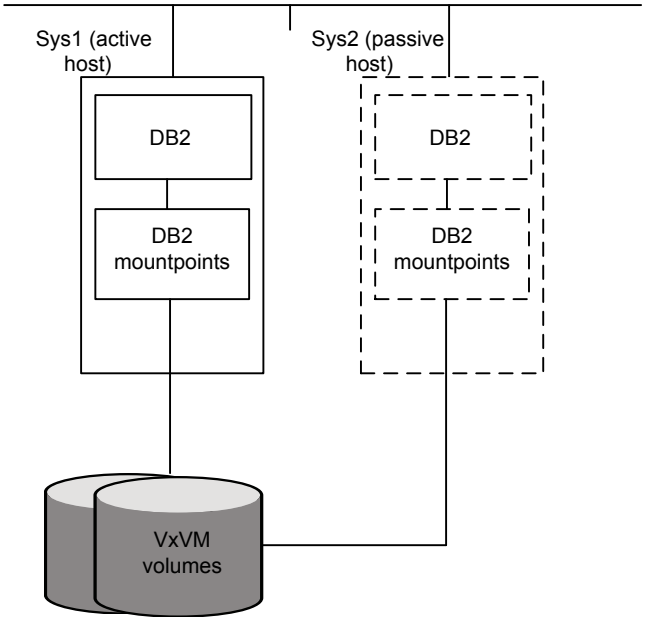
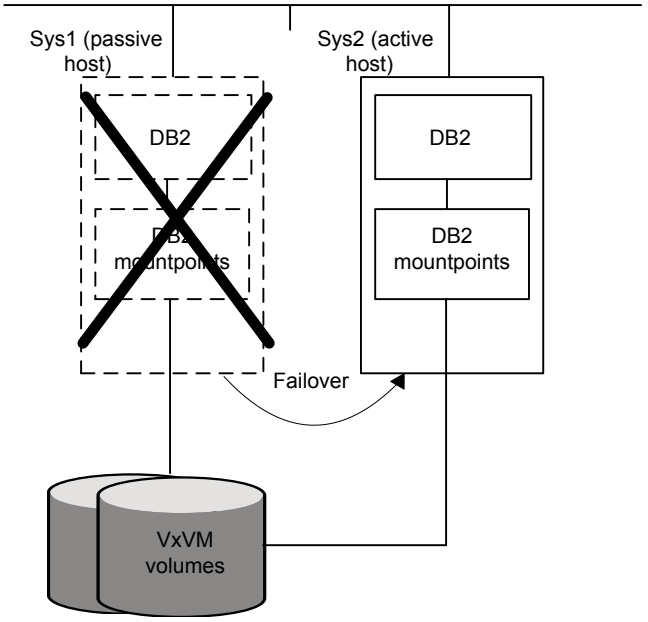


Figure 2-4 DB2 on a single system with SFHA failover setup



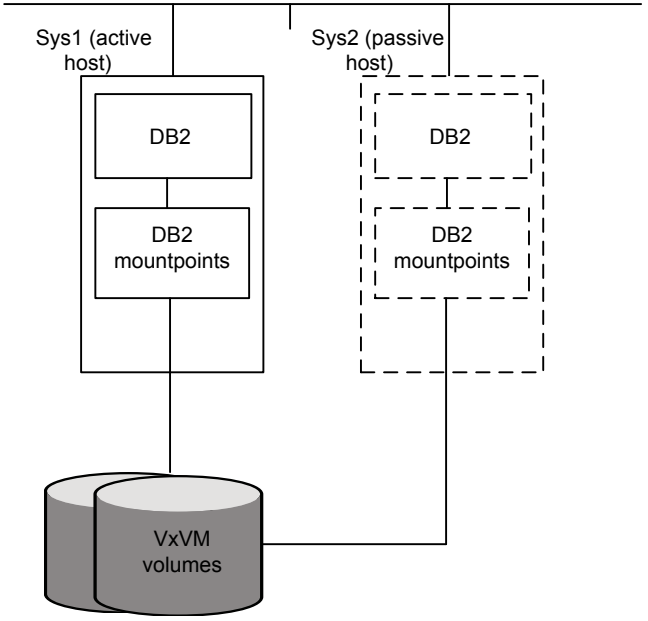
DB2 in a parallel cluster with SF Cluster File System HA

If you are deploying DB2 databases with SF Cluster File System HA, your setup configuration will reflect the following conditions:

- A highly available parallel cluster with a DB2 is set up on *sys1* and *sys2* with SF Cluster File System HA.
- The database is online on *sys1*.
- The datafiles are mounted and shared on *sys1* and *sys2*.
- The database repository is mounted and shared on *sys1* and *sys2*.
- The SFDB tools commands will fail on *sys2*.

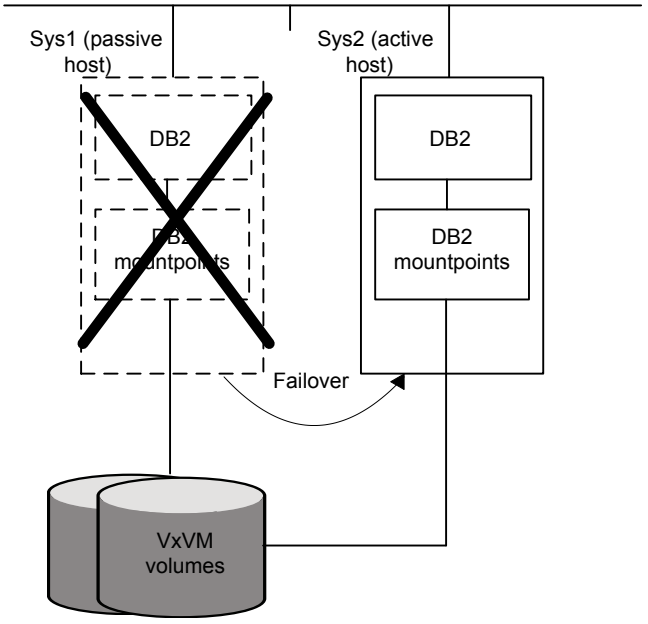
In the figures below the repository directory resides in the DB2 mount points.

Figure 2-5 DB2 on a single system with Storage Foundation HA



The failover to the backup system is automatic rather than manual for SF Cluster File System HA.

Figure 2-6 DB2 on a single system with Storage Foundation HA failover setup



Deploying DB2 and Storage Foundation in a virtualization environment

If you are deploying the Storage Foundation for Databases (SFDB) tools in a virtual machine environment, the following are supported:

- WPARs on AIX
- VMware on Linux

Deploying DB2 with Storage Foundation SmartMove and Thin Provisioning

You can use SmartMove and Thin Provisioning with Storage Foundation products and your DB2 database.

When data files are deleted, you can reclaim the storage space used by these files if the underlying devices are thin reclaimable LUNs. For this purpose, use the Storage Foundation Thin Reclamation feature.

See the *Storage Foundation Administrator's Guide*.

Chapter 3

Deploying DB2 with Storage Foundation

This chapter includes the following topics:

- Tasks for deploying DB2 databases
- About selecting a volume layout for deploying DB2
- Setting up disk group for deploying DB2
- Creating volumes for deploying DB2
- Creating VxFS file system for deploying DB2
- Mounting the file system for deploying DB2
- Installing DB2 and creating database

Tasks for deploying DB2 databases

If you are deploying a DB2 database on a single system in a Storage Foundation environment, complete these tasks in the order listed below:

Create a volume layout.	See “About selecting a volume layout for deploying DB2” on page 36. See “Setting up disk group for deploying DB2” on page 37. See “Disk group configuration guidelines for deploying DB2” on page 38. See “Creating volumes for deploying DB2” on page 39. See “Volume configuration guidelines for deploying DB2” on page 39.
Create and mount file systems.	See “Creating VxFS file system for deploying DB2” on page 40. See “File system creation guidelines for deploying DB2” on page 41. See “Mounting the file system for deploying DB2” on page 41.
Install DB2 and create database.	See “Installing DB2 and creating database” on page 42.

About selecting a volume layout for deploying DB2

Veritas Volume Manager (VxVM) offers a variety of layouts that enables you to configure your database to meet performance and availability requirements. The proper selection of volume layouts provides optimal performance for the database workload.

Disk I/O is one of the most important determining factors of database performance. Having a balanced I/O load usually means optimal performance. Designing a disk layout for the database objects to achieve balanced I/O is a crucial step in configuring a database. When deciding where to place tablespaces, it is often difficult to anticipate future usage patterns. VxVM provides flexibility in configuring storage for the initial database set up and for continual database performance improvement as needs change. VxVM can split volumes across multiple drives to provide a finer level of granularity in data placement. By using striped volumes, I/O can be balanced across multiple disk drives. For most databases, ensuring that different containers or tablespaces, depending on database, are distributed across the available disks may be sufficient.

Striping also helps sequential table scan performance. When a table is striped across multiple devices, a high transfer bandwidth can be achieved by setting the DB2 parameter `DB_FILE_MULTIBLOCK_READ_COUNT` to a multiple of full stripe size divided by `DB_BLOCK_SIZE`.

Another very important consideration when using the DB2 database, which by default performs striping at the tablespace container level, is setting the `DB2_STRIPED_CONTAINERS` variable.

If you plan to use the Database FlashSnap feature (point-in-time copy) for your DB2 database and use it on either the same host or for off-host processing or backup, the layout of volumes should meet the FlashSnap requirements.

Setting up disk group for deploying DB2

Before creating volumes and filesystem for a database, you must set up a disk group for each database.

Review the disk group configuration guidelines before creating disk groups.

See "Disk group configuration guidelines for deploying DB2" on page 38.

To create a disk group

- Use the `vxdg` command as follows.

  ```
  # /opt/VRTS/bin/vxdg init disk_group disk_name=disk_device
  ```

  For example, to create a disk group named *PRODdg* on a raw disk partition, where the disk name *PRODdg01* references the disk within the disk group:

  AIX

  ```
  # /opt/VRTS/bin/vxdg init PRODdg PRODdg01=Disk_0
  ```

  Linux

  ```
  # /opt/VRTS/bin/vxdg init PRODdg PRODdg01=sda
  ```

To add disks to a disk group

- Use the `vxdg` command as follows.

  ```
  # /opt/VRTS/bin/vxdg -g disk_group adddisk disk_name=disk_device
  ```

For example, to add a disk named PRODdg02 to the disk group PRODdg:

AIX

```
# /opt/VRTS/bin/vxdg -g PRODdg adddisk PRODdg02=Disk_0

# /opt/VRTS/bin/vxdg -g PRODdg adddisk PRODdg03=Disk_1

# /opt/VRTS/bin/vxdg -g PRODdg adddisk PRODdg04=Disk_2
```

Linux

```
# /opt/VRTS/bin/vxdg -g PRODdg adddisk PRODdg02=sda

# /opt/VRTS/bin/vxdg -g PRODdg adddisk PRODdg03=sdb

# /opt/VRTS/bin/vxdg -g PRODdg adddisk  PRODdg04=sdc
```

Disk group configuration guidelines for deploying DB2

Follow these guidelines when setting up disk groups.

- Only disks that are online and do not already belong to a disk group can be used to create a new disk group.
- Create one disk group for each database.
- The disk group name must be unique. Name each disk group using the DB2 database name specified by the environment variable `$DB2DATABASE` and a `dg` suffix. The `dg` suffix helps identify the object as a disk group.
- Each disk name must be unique within the disk group.
- Do not share a disk group between different DB2 instances. Although it is not recommended, sharing a disk group among all databases in the same instance may make sense if the instance contains several small databases. In this case, name the disk group using the DB2 instance name specified by the environment variable `$DB2INSTANCE` and a `dg` suffix.
- Never create container files using file systems or volumes that are not in the same disk group.

Note: You must have root privileges to execute all the disk group related VxVM commands.

See the *Storage Foundation Administrator's Guide*.

Creating volumes for deploying DB2

Veritas Volume Manager (VxVM) uses logical volumes to organize and manage disk space. A volume is made up of portions of one or more physical disks, so it does not have the limitations of a physical disk.

Review the volume configuration guidelines before creating volumes.

See "Volume configuration guidelines for deploying DB2" on page 39.

To create a volume

- Use the `vxassist` command as follows.

```
# /opt/VRTS/bin/vxassist -g disk_group make volume_name volume_size
 disk_name
```

The following is an example of creating a volume using the `vxassist` command:

To create a 1 GB volume called db01 on the PRODdg disk group:

```
#/opt/VRTS/bin/vxassist -g PRODdg make db01 1g PRODdg01
```

Volume configuration guidelines for deploying DB2

Follow these guidelines when selecting volume layouts.

- Put the database log files on a file system created on a striped and mirrored (RAID-0+1) volume separate from the index or data tablespaces. Stripe multiple devices to create larger volumes if needed. Use mirroring to improve reliability. Do not use VxVM RAID-5 for redo logs.
- When normal system availability is acceptable, put the tablespaces on filesystems created on striped volumes for most OLTP workloads.
- Create striped volumes across at least four disks. Try to stripe across disk controllers.
  For sequential scans, ensure that the `NUM_IOSERVERS` and the `DB2_PARALLEL_IO` settings are tuned to match the number of disk devices used in the stripe.
- For most workloads, use the default 64 K stripe-unit size for striped volumes.
- When system availability is critical, use mirroring for most write-intensive OLTP workloads. Turn on Dirty Region Logging (DRL) to allow fast volume resynchronization in the event of a system crash.
- For most decision support system (DSS) workloads, where sequential scans are common, experiment with different striping strategies and stripe-unit sizes. Put the most frequently accessed tables or tables that are accessed together on separate striped volumes to improve the bandwidth of data transfer.

Creating VxFS file system for deploying DB2

To create a Veritas File System (VxFS) file system, use the `mkfs` or the `mkfs_vxfs` commands.

Review the file system creation guidelines before creating VxFS file systems.

See "File system creation guidelines for deploying DB2" on page 41.

To create a VxFS file system on an existing volume

- Use the `mkfs` command as follows:
  - AIX

    ```
    # /usr/sbin/mkfs -V vxfs generic_options\
     -o specific_options special size
    ```

  - Linux

    ```
    # /usr/sbin/mkfs -t vxfs generic_options\
     -o specific_options special size
    ```

Where:

- `vxfs` is the file system type
- `generic_options` are the options common to most file systems
- `specific_options` are options specific to the VxFS file system
- `special` is the full path name of the raw character device or the VxVM volume on which to create the file system
- (optional) `size` is the size of the new file system

If you do not specify size, the file system will be as large as the underlying volume.

For example, to create a VxFS file system that has an 8 KB block size and supports files larger than 2 GB on the newly created db01 volume:

- `# /usr/sbin/mkfs -V vxfs -o largefiles,bsize=8192,logsize=2000 \ /dev/vx/rdsk/PRODdg/db01`

The `-o largefiles` option allows you to create files larger than 2GB.

Note: Because size is not specified in this example, the size of the file system will be calculated automatically to be the same size as the volume on which the file system is created.

File system creation guidelines for deploying DB2

Follow these guidelines when creating VxFS file systems.

- Specify the maximum block size and log size when creating file systems for databases.
- Do not disable the intent logging feature of the file system.
- Create separate file systems for redo logs, control files, data files, tmp files, and archive redo logs.
- When using the command line, use the mount points to name the underlying volumes. For example, if a file system named /db01 is to be created on a mirrored volume, name the volume db01 and the mirrors db01-01 and db01-02 to relate to the configuration objects. If you are using the vxassist command or the GUI, this is transparent.
- The block size of your DB2 database should be a multiple of the file system block size. If possible, keep them of the same size.

See the *Storage Foundation Administrator's Guide*.

Mounting the file system for deploying DB2

After creating a VxFS file system, as a root user, mount the file system using the `mount` command.

See the man pages for the `mount` and the `mount_vxfs` commands for more information.

To mount a file system

- Use the `mount` command as follows:
  - AIX

    ```
    # /usr/sbin/mount -V vxfs special /mount_point
    ```

  - Linux

    ```
    # /usr/sbin/mount -t vxfs special /mount_point
    ```

Where:

- `vxfs` is the file system type
- `special` is a block special device
- `/mount_point` is the directory where the file system will be mounted

For example, to mount a file system named `/db01` that supports large files on volume `/dev/vx/dsk/PRODdg/db01`

- `# /usr/sbin/mount -V vxfs -o largefiles /dev/vx/dsk/PRODdg/db01 \ /db01`

Installing DB2 and creating database

Review database layouts considerations and supported configurations for deploying DB2.

See "Considerations for DB2 database layouts" on page 104.

See "Supported DB2 configurations" on page 105.

For information on installing the DB2 software and creating DB2 databases, refer to DB2 documentation.

Chapter 4

Deploying DB2 in an off-host configuration with Storage Foundation

This chapter includes the following topics:

- Requirements for an off-host database configuration

Requirements for an off-host database configuration

If you are using Storage Foundation Database (SFDB) tools to set up a DB2 database in an off-host configuration, ensure the following.

- All the tasks for deploying a DB2 database in a Veritas InfoScale Storage Foundation environment are completed.
  See "Tasks for deploying DB2 databases" on page 35.
- The following requirements are met.
  - All files are on VxFS file systems over VxVM volumes. Raw devices are not supported.
  - There are no symbolic links to database files.
  - The product versions installed on the primary and secondary hosts are the same.
  - The same version of DB2 is installed on both hosts, the DB2 binaries and data files are on different volumes and disks.

- The UNIX login for the database user and group must be the same on both hosts. The UNIX UID and GID must also be the same.
- You must have an Veritas InfoScale Enterprise license on both hosts.

Chapter 5

Deploying DB2 with High Availability

This chapter includes the following topics:

- Tasks for deploying DB2 in an HA configuration
- Configuring VCS to make the database highly available

Tasks for deploying DB2 in an HA configuration

If you are deploying a DB2 database in a Storage Foundation High Availability (SFHA) environment, complete the following tasks.

Complete the tasks for deploying a DB2 database in a Storage Foundation enviornment.	See "Tasks for deploying DB2 databases" on page 35.
Configure VCS to make the database highly available.	See "Configuring VCS to make the database highly available" on page 45.

Configuring VCS to make the database highly available

To make your DB2 database highly available, you need to bring your database configuration under Cluster Server (VCS) control.

See the *Veritas InfoScale Cluster Server Administrator's Guide*.

Section 3

Configuring Storage Foundation for Database (SFDB) tools

- Chapter 6. Configuring and managing the Storage Foundation for Databases repository database
- Chapter 7. Configuring authentication for Storage Foundation for Databases (SFDB) tools

Chapter 6

Configuring and managing the Storage Foundation for Databases repository database

This chapter includes the following topics:

- About the Storage Foundation for Databases (SFDB) repository
- Requirements for Storage Foundation for Databases (SFDB) tools
- Storage Foundation for Databases (SFDB) tools availability
- Configuring the Storage Foundation for Databases (SFDB) tools repository
- Updating the Storage Foundation for Databases (SFDB) repository after adding a node
- Updating the Storage Foundation for Databases (SFDB) repository after removing a node
- Removing the Storage Foundation for Databases (SFDB) repository

About the Storage Foundation for Databases (SFDB) repository

The Storage Foundation for Databases (SFDB) repository or repository database stores metadata information required by the Storage Foundation for Databases tools.

Note: The repository database requires only occasional interaction outside of the initial installation and configuration of Veritas InfoScale Enterprise products.

In this release of Storage Foundation products, the SFDB repository is stored in a relational database and is managed by SQLite3.

Requirements for Storage Foundation for Databases (SFDB) tools

Product requirements are included in the *Veritas InfoScale 7.4.2 Release Notes*.

The hardware compatibility list contains information about supported hardware and is updated regularly. Refer to the latest information on supported hardware at:

https://www.veritas.com/content/support/en_US/doc/infoscale_hcl_73_731_74_unix

For the most current information on the DB2 versions that are supported with InfoScale products, refer to the database support matrix at:

https://www.veritas.com/bin/support/docRepoServlet?bookId=112638608-112638611-1&requestType=pdf

Review the current DB2 documentation to confirm the compatibility of your hardware and software.

Storage Foundation for Databases (SFDB) tools availability

SFDB tools for DB2 databases are included for the following products:

- Storage Foundation, which supports host systems with DB2

  Note: Veritas InfoScale Enterprise licensing required.

- Storage Foundation for Cluster File System HA, which supports clustered host systems with automatic failover and DB2

For information on SFDB tools feature changes and issues for this release, see the *Veritas InfoScale 7.4.2 Release Notes* for the most current and complete information.

Configuring the Storage Foundation for Databases (SFDB) tools repository

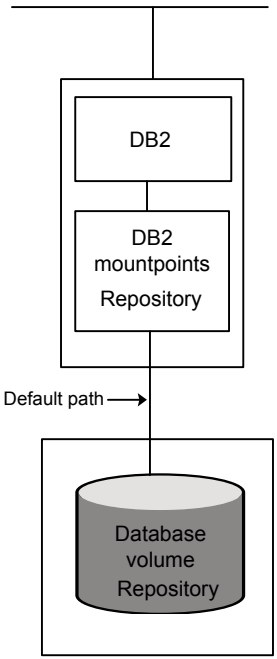
The SFDB repository is automatically created when you run `vxsfadm` for the first time in a DB2 setup. No other steps are required.

Locations for the SFDB repository

The repository location is the same as the DBPATH.

In the figure below the repository directory resides in the DB2 mount points.

Figure 6-1 Location for the SFDB repository



Updating the Storage Foundation for Databases (SFDB) repository after adding a node

After adding a node to a cluster, update the SFDB repository to enable access for the new node.

To update the SFDB repository after adding a node

1. Copy the `/var/vx/vxdba/rep_loc` file from one of the nodes in the cluster to the new node.
2. If the `/var/vx/vxdba/auth/user-authorizations` file exists on the existing cluster nodes, copy it to the new node.

   If the `/var/vx/vxdba/auth/user-authorizations` file does not exist on any of the existing cluster nodes, no action is required.

   This completes the addition of the new node to the SFDB repository.

Updating the Storage Foundation for Databases (SFDB) repository after removing a node

After removing a node from a cluster, you do not need to perform any steps to update the SFDB repository.

For information on removing the SFDB repository after removing the product:

See "Removing the Storage Foundation for Databases (SFDB) repository" on page 50.

Removing the Storage Foundation for Databases (SFDB) repository

After removing the product, you can remove the SFDB repository file and any backups.

Removing the SFDB repository file disables the SFDB tools.

To remove the SFDB repository

1. Identify the SFDB repositories created on the host.
2. Remove the directory identified by the `location` key.

   DB2 9.5 and 9.7:

   ```
   # rm -rf /db2data/db2inst1/NODE0000/SQL00001/.sfae
   ```

   DB2 10.1 and 10.5:

   ```
   # rm -rf /db2data/db2inst1/NODE0000/SQL00001/MEMBER0000/.sfae
   ```

3. Remove the repository location file.

   ```
   # rm -rf /var/vx/vxdba/rep_loc
   ```

   This completes the removal of the SFDB repository.

Chapter 7

Configuring authentication for Storage Foundation for Databases (SFDB) tools

This chapter includes the following topics:

- Configuring vxdbd for SFDB tools authentication
- Adding nodes to a cluster that is using authentication for SFDB tools
- Authorizing users to run SFDB commands

Configuring vxdbd for SFDB tools authentication

To configure vxdbd, perform the following steps as the root user

1 Run the `sfae_auth_op` command to set up the authentication services.

```
# /opt/VRTS/bin/sfae_auth_op -o setup
Setting up AT
Starting SFAE AT broker
Creating SFAE private domain
Backing up AT configuration
Creating principal for vxdbd
```

2 Stop the `vxdbd` daemon.

```
# /opt/VRTS/bin/sfae_config disable
vxdbd has been disabled and the daemon has been stopped.
```

3 Enable authentication by setting the AUTHENTICATION key to yes in the /etc/vx/vxdbed/admin.properties configuration file.

If /etc/vx/vxdbed/admin.properties does not exist, then usecp /opt/VRTSdbed/bin/admin.properties.example /etc/vx/vxdbed/admin.properties.

4 Start the vxdbd daemon.

```
# /opt/VRTS/bin/sfae_config enable
vxdbd has been enabled and the daemon has been started.
It will start automatically on reboot.
```

The vxdbd daemon is now configured to require authentication.

Adding nodes to a cluster that is using authentication for SFDB tools

To add a node to a cluster that is using authentication for SFDB tools, perform the following steps as the root user

1 Export authentication data from a node in the cluster that has already been authorized, by using the -o export_broker_config option of the sfae_auth_op command.

Use the -f option to provide a file name in which the exported data is to be stored.

```
# /opt/VRTS/bin/sfae_auth_op \
-o export_broker_config -f exported-data
```

2 Copy the exported file to the new node by using any available copy mechanism such as scp or rcp.

3 Import the authentication data on the new node by using the -o import_broker_config option of the sfae_auth_op command.

Use the -f option to provide the name of the file copied in Step 2.

```
# /opt/VRTS/bin/sfae_auth_op \
 -o import_broker_config -f exported-data
Setting up AT
Importing broker configuration
Starting SFAE AT broker
```

4 Stop the `vxdbd` daemon on the new node.

```
# /opt/VRTS/bin/sfae_config disable
vxdbd has been disabled and the daemon has been stopped.
```

5 Enable authentication by setting the `AUTHENTICATION` key to `yes` in the `/etc/vx/vxdbed/admin.properties` configuration file.

If `/etc/vx/vxdbed/admin.properties` does not exist, then use `cp /opt/VRTSdbed/bin/admin.properties.example /etc/vx/vxdbed/admin.properties`

6 Start the `vxdbd` daemon.

```
# /opt/VRTS/bin/sfae_config enable
vxdbd has been enabled and the daemon has been started.
It will start automatically on reboot.
```

The new node is now authenticated to interact with the cluster to run SFDB commands.

Authorizing users to run SFDB commands

To authorize users to run SFDB commands, perform the following step as the root user

- Use the `-o auth_user` option of the `sfae_auth_op` command and provide the user name as an argument to the `-u` option.

```
# /opt/VRTS/bin/sfae_auth_op \
-o auth_user -u db2inst1
Creating principal db2inst1@sys1.example.com
```

With an off-host setup in which the off-host node is not part of the same cluster, use an additional `-h` option and perform the following steps:

1. On each primary node or nodes, which are part of a cluster, add **-h *off-host hostname*** to the `sfae_auth_op` command, for example:

```
# /opt/VRTS/bin/sfae_auth_op -o auth_user -u oragrid -h myoffhost
Creating principal oragrid@dblxx64-2-v6.vxindia.veritas.com
```

2. On the off-host node, add **-h *primary node*** to the `sfae_auth_op` command, for example:

```
# /opt/VRTS/bin/sfae_auth_op -o auth_user -u oragrid -h dblxx64-2-v6
Creating principal oragrid@myoffhost.vxindia.veritas.com
```

Section 4

Improving DB2 database performance

- Chapter 8. About database accelerators
- Chapter 9. Improving database performance with Quick I/O
- Chapter 10. Improving DB2 database performance with Veritas Concurrent I/O

Chapter 8

About database accelerators

This chapter includes the following topics:

- About Veritas InfoScale product components database accelerators

About Veritas InfoScale product components database accelerators

The major concern in any environment is maintaining respectable performance or meeting performance service level agreements (SLAs). Veritas InfoScale product components improve the overall performance of database environments in a variety of ways.

Table 8-1 Veritas InfoScale product components database accelerators

Veritas InfoScale database accelerator	Supported databases	Use cases and considerations
Oracle Disk Manager (ODM)	Oracle	■ To improve Oracle performance and manage system bandwidth through an improved Application Programming Interface (API) that contains advanced kernel support for file I/O. ■ To use Oracle Resilvering and turn off Veritas Volume Manager Dirty Region Logging (DRL) to increase performance, use ODM. ■ To reduce the time required to restore consistency, freeing more I/O bandwidth for business-critical applications, use SmartSync recovery accelerator.
Cached Oracle Disk Manager (Cached ODM)	Oracle	To enable selected I/O to use caching to improve ODM I/O performance, use Cached ODM.
Quick I/O (QIO)	Oracle DB2 Sybase	To achieve raw device performance for databases run on VxFS file systems, use Quick I/O.
Cached Quick I/O (Cached QIO)	Oracle DB2 Sybase	To further enhance database performance by leveraging large system memory to selectively buffer the frequently accessed data, use Cached QIO.
Concurrent I/O	DB2 Sybase	Concurrent I/O (CIO) is optimized for DB2 and Sybase environments To achieve improved performance for databases run on VxFS file systems without restrictions on increasing file size, use Veritas InfoScale Concurrent I/O.

These database accelerator technologies enable database performance equal to raw disk partitions, but with the manageability benefits of a file system. With the Dynamic Multi-pathing (DMP) feature of Storage Foundation, performance is maximized by load-balancing I/O activity across all available paths from server to

array. DMP supports all major hardware RAID vendors, hence there is no need for third-party multi-pathing software, reducing the total cost of ownership.

Veritas InfoScale database accelerators enable you to manage performance for your database with more precision.

For details about using ODM, Cached ODM, QIO, and Cached QIO for Oracle, see *Veritas InfoScale Storage and Availability Management for Oracle Databases*.

For details about using QIO, Cached QIO, and Concurrent I/O for DB2, see *Veritas InfoScale Storage and Availability Management for DB2 Databases*.

For details about using ODM and Cached ODM for Oracle, see *Veritas InfoScale Storage and Availability Management for Oracle Databases*.

For details about using Concurrent I/O for DB2, see *Veritas InfoScale Storage and Availability Management for DB2 Databases*.

Chapter 9

Improving database performance with Quick I/O

This chapter includes the following topics:

- About Quick I/O
- Tasks for setting up Quick I/O in a database environment
- Preallocating space for Quick I/O files using the setext command
- Accessing regular VxFS files as Quick I/O files
- Converting DB2 containers to Quick I/O files
- About sparse files
- Displaying Quick I/O status and file attributes
- Extending a Quick I/O file
- Monitoring tablespace free space with DB2 and extending tablespace containers
- Recreating Quick I/O files after restoring a database
- Disabling Quick I/O

About Quick I/O

Veritas Quick I/O is a VxFS feature included in Veritas InfoScale Storage Foundation Standard and Enterprise products that enables applications access preallocated VxFS files as raw character devices. Quick I/O provides the administrative benefits

of running databases on file systems without the typically associated degradation in performance.

Note: Quick I/O is not supported on Linux.

Using Quick I/O is recommended on DB2 databases prior to DB2 8.1 with FixPak 4. With DB2 8.1 with FixPak 4 or later, you can use the Veritas Concurrent I/O feature, which provides high-speed access for SMS tablespaces as well as DMS tablespaces.

See “About Concurrent I/O” on page 83.

How Quick I/O improves database performance

The benefits of using Quick I/O are:

- Improved performance and processing throughput by having Quick I/O files act as raw devices.
- Ability to manage Quick I/O files as regular files, which simplifies administrative tasks such as allocating, moving, copying, resizing, and backing up DB2 containers.

Note: Quick I/O is not supported on Linux.

Quick I/O's ability to access regular files as raw devices improves database performance by:

Table 9-1

Quick I/O feature	Advantage
Supporting direct I/O	I/O on files using `read`() and `write`() system calls typically results in data being copied twice: once between user and kernel space, and later between kernel space and disk. In contrast, I/O on raw devices is direct. That is, data is copied directly between user space and disk, saving one level of copying. As with I/O on raw devices, Quick I/O avoids extra copying.

Table 9-1 *(continued)*

Quick I/O feature	Advantage
Avoiding kernel write locks on database files	When database I/O is performed using the `write()` system call, each system call acquires and releases a write lock inside the kernel. This lock prevents multiple simultaneous write operations on the same file. Because database systems usually implement their own locking to manage concurrent access to files, per file writer locks unnecessarily serialize I/O operations. Quick I/O bypasses file system per file locking and lets the database server control data access.
Avoiding double buffering	Most database servers maintain their own buffer cache and do not need the file system buffer cache. Database data cached in the file system buffer is therefore redundant and results in wasted memory and extra system CPU utilization to manage the buffer. By supporting direct I/O, Quick I/O eliminates double buffering. Data is copied directly between the relational database management system (RDBMS) cache and disk, which lowers CPU utilization and frees up memory that can then be used by the database server buffer cache to further improve transaction processing throughput.
For AIX: Supporting AIX Fastpath asynchronous I/O	AIX Fastpath asynchronous I/O is a form of I/O that performs non-blocking system level reads and writes, allowing the system to handle multiple I/O requests simultaneously. Operating systems such as AIX provide support for asynchronous I/O on raw devices, but not on regular files. As a result, even if the database server is capable of using asynchronous I/O, it cannot issue asynchronous I/O requests when the database runs on file systems. Lack of asynchronous I/O significantly degrades performance. Quick I/O lets the database server take advantage of kernel-supported asynchronous I/O on file system files accessed using the Quick I/O interface.

Tasks for setting up Quick I/O in a database environment

Quick I/O is included in the VxFS package shipped with Veritas InfoScale Storage Foundation Standard and Enterprise products. By default, Quick I/O is enabled when you mount a VxFS file system.

If Quick I/O is not available in the kernel, or a Veritas InfoScale Storage or Veritas InfoScale Enterprise product license is not installed, a file system mounts without Quick I/O by default, the Quick I/O file name is treated as a regular file, and no error message is displayed. If, however, you specify the `-o qio` option, the `mount` command prints the following error message and terminates without mounting the file system.

```
VxFDD: You don't have a license to run this program
vxfs mount: Quick I/O not available
```

To use Quick I/O, you must:

- Preallocate files on a VxFS file system
  Preallocating database files for Quick I/O allocates contiguous space for the files. The file system space reservation algorithms attempt to allocate space for an entire file as a single contiguous extent. When this is not possible due to lack of contiguous space on the file system, the file is created as a series of direct extents. Accessing a file using direct extents is inherently faster than accessing the same data using indirect extents. Internal tests have shown performance degradation in OLTP throughput when using indirect extent access. In addition, this type of preallocation causes no fragmentation of the file system.
  You must preallocate Quick I/O files because they cannot be extended through writes using their Quick I/O interfaces. They are initially limited to the maximum size you specify at the time of creation.
  See “Extending a Quick I/O file” on page 75.

- Use a special file naming convention to access the files
  VxFS uses a special naming convention to recognize and access Quick I/O files as raw character devices. VxFS recognizes the file when you add the following extension to a file name:
  `::cdev:vxfs:`
  Whenever an application opens an existing VxFS file with the extension `::cdev:vxfs:` (`cdev` being an acronym for character device), the file is treated as if it were a raw device. For example, if the file `temp01` is a regular VxFS file, then an application can access `temp01` as a raw character device by opening it with the name:
  `.temp01::cdev:vxfs:`

Note: We recommend reserving the `::cdev:vxfs:` extension only for Quick I/O files. If you are not using Quick I/O, you could technically create a regular file with this extension; however, doing so can cause problems if you later enable Quick I/O.

Depending on whether you are creating a new database or are converting an existing database to use Quick I/O, you have the following options:

If you are creating a new database to use Quick I/O:

- You can use the `qiomkfile` command to preallocate space for database files and make them accessible to the Quick I/O interface.
- You can use the `setext` command to preallocate space for database files and create the Quick I/O files.
  See "Preallocating space for Quick I/O files using the setext command" on page 64.

If you are converting an existing database:

- You can create symbolic links for existing VxFS files, and use these symbolic links to access the files as Quick I/O files.
  See "Accessing regular VxFS files as Quick I/O files" on page 66.

Preallocating space for Quick I/O files using the setext command

As an alternative to using the `qiomkfile` command, you can also use the VxFS `setext` command to preallocate space for database files.

Before preallocating space with `setext`, make sure the following conditions have been met:

Prerequisites	■ The `setext` command requires superuser (`root`) privileges.
Usage notes	■ You can use the `chown` command to change the owner and group permissions on the file after you create it. See the `setext` (1M) manual page for more information.

To create a Quick I/O database file using setext

1. Access the VxFS mount point and create a file:

```
# cd /mount_point

# touch .filename
```

2. Use the `setext` command to preallocate space for the file:

```
# /opt/VRTS/bin/setext -r size -f noreserve -f chgsize  \
.filename
```

3. Create a symbolic link to allow databases or applications access to the file using its Quick I/O interface:

```
# ln -s .filename::cdev:vxfs: filename
```

4 Change the owner and group permissions on the file.

For example, for DB2:

```
# chown user:group .filename

# chmod 660 .dbfile
```

For example, for Sybase:

```
# chown sybase:sybase .filename

# chmod 660 .filename
```

5 To access the mountpoint for the database:

For example, for */db01*, create a container, preallocate the space, and change the permissions:

```
# cd /db01
# touch .dbfile
# /opt/VRTS/bin/setext -r 100M -f noreserve -f chgsize .dbfile
# ln -s .dbfile::cdev:vxfs: dbfile
```

For DB2:

```
# chown db2inst1:db2iadm1 .dbfile

# chmod 660 .dbfile
```

For Sybase:

```
# chown sybase:sybase .dbfile

# chmod 660 .dbfile
```

Accessing regular VxFS files as Quick I/O files

You can access regular VxFS files as Quick I/O files using the `::cdev:vxfs:` name extension.

While symbolic links are recommended because they provide easy file system management and location transparency of database files, the drawback of using symbolic links is that you must manage two sets of files (for instance, during database backup and restore).

Usage notes

- If possible, use relative path names instead of absolute path names when creating symbolic links to access regular files as Quick I/O files. Using relative path names prevents copies of the symbolic link from referring to the original file when the directory is copied. This is important if you are backing up or moving database files with a command that preserves the symbolic link.

  However, some applications require absolute path names. If a file is then relocated to another directory, you must change the symbolic link to use the new absolute path. Alternatively, you can put all the symbolic links in a directory separate from the data directories. For example, you can create a directory named `/database` and put all the symbolic links there, with the symbolic links pointing to absolute path names.

To access an existing regular file as a Quick I/O file on a VxFS file system

1 Access the VxFS file system mount point containing the regular files:

```
$ cd /mount_point
```

2 Create the symbolic link:

```
$ mv filename .filename
$ ln -s .filename::cdev:vxfs: filename
```

This example shows how to access the VxFS file `dbfile` as a Quick I/O file:

```
$ cd /db01
$ mv dbfile .dbfile
$ ln -s .dbfile::cdev:vxfs: dbfile
```

This example shows how to confirm the symbolic link was created:

```
$ ls -lo .dbfile dbfile
```

For DB2:

```
-rw-r--r--  1 db2inst1 104890368 Oct 2 13:42  .dbfile

lrwxrwxrwx  1 db2inst1    19     Oct 2 13:42   dbfile ->
.dbfile::vxcdev:vxfs:
```

For Sybase:

```
$ ls -lo .dbfile dbfile

-rw-r--r--  1 sybase  104890368  Oct 2 13:42  .dbfile

lrwxrwxrwx  1 sybase      19     Oct 2 13:42   dbfile ->
.dbfile::cdev:vxfs:
```

Converting DB2 containers to Quick I/O files

Special commands available in the `/opt/VRTS/bin` directory are provided to assist you in converting an existing database to use Quick I/O. You can use the `qio_getdbfiles` command to extract a list of file names from the database system tables and the `qio_convertdbfiles` command to convert this list of database files to use Quick I/O.

Before converting database files to Ouick I/O files, the following conditions must be met:

Prerequisites

- Files you want to convert must be regular files on VxFS file systems or links that point to regular VxFS files

Usage notes

- Converting existing database files to Quick I/O files may not be the best choice if the files are fragmented. Use the `-f` option to determine the fragmentation levels and choose one of two approaches: Either exclude files that are highly fragmented and do not have sufficient contiguous extents for Quick I/O use, or create new files with the `qiomkfile` command, rather than convert them with the `qio_convertdbfiles` command.
- `qio_getdbfiles` skips any tablespaces that have a type of system managed space (SMS), as these tablespaces are based on a directory format and not suitable for conversion.
  The `qio_getdbfiles` command retrieves a list of database files and saves them in a file named `mkqio.dat` in the current directory.
- Instead of using the `qio_getdbfiles` command, you can manually create the `mkqio.dat` file containing the DB2 instance filenames that you want to convert to Quick I/O files.
- The `qio_convertdbfiles` command exits and prints an error message if any of the database files are not on a VxFS file system. If this happens, you must remove any non-VxFS files from the `mkqio.dat` file before running the `qio_convertdbfiles` command.

The following options are available for the `qio_getdbfiles` command:

`-T` Lets you specify the type of database as `db2`. Specify this option only in environments where the type of database is ambiguous (for example, when multiple types of database environment variables, such as `$ORACLE_SID`, `SYBASE`, `DSQUERY`, and `$DB2INSTANCE`, are present on a server).

The following options are available for the `qio_convertdbfiles` command:

`-a` Changes regular files to Quick I/O files using absolute path names. Use this option when symbolic links need to point to absolute path names (for example, at a site that uses SAP).

`-h` Displays a help message.

`-T` Enables you to specify the type of database as `db2`. Specify this option only in environments where the type of database is ambiguous (for example, when multiple types of database environment variables, such as `$ORACLE_SID`, `SYBASE`, `DSQUERY`, and `$DB2INSTANCE` are present on a server).

`-u` Changes Quick I/O files back to regular files. Use this option to undo changes made by a previous run of the `qio_convertdbfiles` script.

To extract a list of DB2 containers to convert

- With the database instance up and running, run the `qio_getdbfiles` command from a directory for which you have write permission:

```
$ cd /extract_directory

$ export DB2DATABASE=database_name

$ /opt/VRTS/bin/qio_getdbfiles
```

The `qio_getdbfiles` command extracts the list file names from the database system tables and stores the file names and their size in bytes in a file called `mkqio.dat` under the current directory.

Note: Alternatively, you can manually create the `mkqio.dat` file containing the DB2 database container names that you want to convert to use Quick I/O. You can also manually edit the `mkqio.dat` file generated by `qio_getdbfiles`, and remove files that you do not want to convert to Quick I/O files.

Note: To run the `qio_getdbfiles` command, you must have permission to access the database and permission to write to the `/extract_directory`.

The `mkqio.dat` list file should look similar to the following:

```
/data11r1/VRTS11r1/redo01.log 52428800
/data11r1/VRTS11r1/redo02.log 52428800
/data11r1/VRTS11r1/redo03.log 52428800
/data11r1/VRTS11r1/sysaux01.dbf 632553472
/data11r1/VRTS11r1/system01.dbf 754974720
/data11r1/VRTS11r1/undotbs01.dbf 47185920
/data11r1/VRTS11r1/users01.dbf 5242880
/data11r1/nqio1.dbf 104857600
```

To convert the DB2 database files to Quick I/O files

1. Make the database inactive by either shutting down the instance or disabling user connections.

   Warning: Running the `qio_convertdbfiles` command while the database is up and running can cause severe problems with your database, including loss of data and corruption.

2. Run the `qio_convertdbfiles` command from the directory containing the `mkqio.dat` file:

   ```
   $ cd /extract_directory

   $ export DB2DATABASE=database_name

   $ /opt/VRTS/bin/qio_convertdbfiles
   ```

   The list of files in the `mkqio.dat` file is displayed. For example:

   ```
   file1 --> .file1::cdev:vxfs:
   file2 --> .file2::cdev:vxfs:
   file3 --> .file3::cdev:vxfs:
   file4 --> .file4::cdev:vxfs:
   file5 --> .file5::cdev:vxfs:
   ```

   Run the `qio_convertdbfiles` command (with no options specified) to rename the file *filename* to `.filename` and creates a symbolic link to `.filename` with the Quick I/O extension. By default, the symbolic link uses a relative path name.

   The `qio_convertdbfiles` script exits and prints an error message if any of the database files are not on a VxFS file system. If this happens, you must remove any non-VxFS files from the `mkqio.dat` file before running the `qio_convertdbfiles` command again.

3. Make the database active again.

   You can now access these database files using the Quick I/O interface.

To undo the previous run of qio_convertdbfiles and change Quick I/O files back to regular VxFS files

1. If the database is active, make it inactive by either shutting down the instance or disabling user connections.
2. Run the following command from the directory containing the `mkqio.dat` file:

   ```
   $ cd /extract_directory

   $ export DB2DATABASE=database_name

   $ /opt/VRTS/bin/qio_convertdbfiles -u
   ```

   The list of Quick I/O files in the `mkqio.dat` file is displayed. For example:

   ```
   .file1::cdev:vxfs: --> file1
   .file2::cdev:vxfs: --> file2
   .file3::cdev:vxfs: --> file3
   .file4::cdev:vxfs: --> file4
   .file5::cdev:vxfs: --> file5
   ```

   The `qio_convertdbfiles` command with the undo option (`-u`) specified renames the files from <.filename> to <filename> and undoes the symbolic link to `.filename` that was created along with the Quick I/O files.

About sparse files

Support for sparse files lets applications store information (in inodes) to identify data blocks that have only zeroes, so that only blocks containing non-zero data have to be allocated on disk.

For example, if a file is 10KB, it typically means that there are blocks on disk covering the whole 10KB. Assume that you always want the first 9K to be zeroes. The application can go to an offset of 9KB and write 1KB worth of data. Only a block for the 1KB that was written is allocated, but the size of the file is still 10KB.

The file is now sparse. It has a hole from offset 0 to 9KB. If the application reads any part of the file within this range, it will see a string of zeroes.

If the application subsequently writes a 1KB block to the file from an offset of 4KB, for example, the file system will allocate another block.

The file then looks like:

- 0-4KB - hole
- 4-5KB - data block

- 5-9KB - hole
- 9-10KB - data block

So a 1TB file system can potentially store up to 2TB worth of files if there are sufficient blocks containing zeroes. Quick I/O files cannot be sparse and will always have all blocks specified allocated to them.

Displaying Quick I/O status and file attributes

You can obtain and display information about Quick I/O status and file attributes using various options of the `ls` command:

-al	Lists all files on a file system, including Quick I/O files and their links.
-1L	Shows if Quick I/O was successfully installed and enabled.
-a1L	Shows how a Quick I/O file name is resolved to that of a raw device.

To list all files on the current file system, including Quick I/O files and their links

- Use the `ls -al` command with the file names:

  ```
  $ ls -al filename .filename
  ```

  The following example shows how to use the `-a` option to display the absolute path name created using `qiomkfile`:

  ```
  $ ls -al d* .d*
  ```

  For DB2:

  ```
  -rw-r--r--  1 db2inst1 db2iadm1 104890368  Oct 2 13:42  .dbfile

  lrwxrwxrwx  1 db2inst1 db2iadm1   19       Oct 2 13:42   dbfile ->
   .dbfile::cdev:vxfs:
  ```

  For Sybase:

  ```
  -rw-r--r--  1 sybase  sybase  104890368  Oct 2 13:42  .dbfile

  lrwxrwxrwx  1 sybase  sybase     19      Oct 2 13:42   dbfile ->
   .dbfile::cdev:vxfs:
  ```

To determine if Quick I/O is installed and enabled for DB2

- Use the `ls` command as follows:

  ```
  $ ls -lL filename
  ```

  The following example shows how to determine if Quick I/O is installed and enabled:

  ```
  $ ls -lL dbfile
  ```

  where the first character, `c`, indicates it is a raw character device file, and the major and minor device numbers are displayed in the size field. If you see a `No such file or directory` message, Quick I/O did not install properly or does not have a valid Veritas InfoScale Enterprise and Storage license keys.

To determine if a Sybase segment has been converted to Quick I/O

- Use the `ls` command as follows:

  ```
  $ ls -lL filename
  ```

  The following example shows how to determine if Quick I/O is installed and enabled:

  ```
  $ ls -lL dbfile

  crw-r--r--   1 sybase  dba  45, 1  Oct 2 13:42  dbfile
  ```

To show a Quick I/O file resolved to a raw device

- Use the `ls` command with the file names as follows:

  ```
  $ ls -alL filename .filename
  ```

  The following example shows how the Quick I/O file name `dbfile` is resolved to that of a raw device:

  ```
  $ ls -alL d* .d*
  ```

  For DB2:

  ```
  crw-r--r--   1 db2inst1 db2iadm1 45,  1        Oct 2 13:42    dbfile
  -rw-r--r--   1 db2inst1 db2iadm1 104890368     Oct 2 13:42    .dbfile
  ```

  For Sybase:

  ```
  crw-r--r--   1 sybase   sybase   45,  1        Oct 2 13:42    dbfile
  -rw-r--r--   1 sybase   sybase   104890368     Oct 2 13:42    .dbfile
  ```

Extending a Quick I/O file

Although Quick I/O files must be preallocated, they are not limited to the preallocated sizes. You can grow or "extend" a Quick I/O file by a specific amount or to a specific size, using options to the `qiomkfile` command. Extending Quick I/O files is a fast, online operation and offers a significant advantage over using raw devices.

Before extending a Quick I/O file, make sure the following conditions have been met:

Prerequisites

- You must have sufficient space on the file system to extend the Quick I/O file.

Usage notes

- You can also grow VxFS file systems online (provided the underlying disk or volume can be extended) using the `fsadm` command. You can expand the underlying volume and the filesystem with the `vxresize` command.
- You must have superuser (`root`) privileges to resize VxFS file systems using the `fsadm` command.
- For Sybase: although you have the ability to extend a Quick I/O file, you cannot resize a database device in Sybase once it is initialized. However, with the ability to grow the volumes and file systems online, you can easily allocate new database devices to be used for new segments and to extend existing segments.
- See the `fsadm_vxfs` (1M) and `qiomkfile` (1M) manual pages for more information.

The following options are available with the `qiomkfile` command:

`-e`	Extends the file by a specified amount to allow resizing.
`-r`	Increases the file to a specified size to allow resizing.

To extend a Quick I/O file

1 If required, ensure the underlying storage device is large enough to contain a larger VxFS file system (see the `vxassist`(1M) manual page for more information), and resize the VxFS file system using `fsadm` command:

For Sybase, for example:

```
# /opt/VRTS/bin/fsadm -b newsize /mount_point
```

where:

- `-b` is the option for changing size
- *newsize* is the new size of the file system in bytes, kilobytes, megabytes, blocks, or sectors

- *mount_point* is the file system's mount point

2 Extend the Quick I/O file using the `qiomkfile` command:

```
# /opt/VRTS/bin/qiomkfile -e extend_amount /mount_point/filename
```

or

```
# /opt/VRTS/bin/qiomkfile -r newsize /mount_point/filename
```

An example to show how to grow VxFS file system:

`/db01` to 500MB and extend the `tbs1_cont001` Quick I/O file by 20MB:

```
# /opt/VRTS/bin/qiomkfile -e 20M /db01/tbs1_cont001
```

```
# /opt/VRTS/bin/fsadm -b 500M /db01
```

An example to show how to grow VxFS file system for DB2:

`/db01` to 500MB and resize the `tbs1_cont001` Quick I/O file to 300MB:

```
# /opt/VRTS/bin/fsadm -b 500M /db01
```

```
# /opt/VRTS/bin/qiomkfile -r 300M /db01/tbs1_cont001
```

An example to show how to grow VxFS file system for Sybase:

`/db01` to 500MB and resize the `dbfile` Quick I/O file to 300MB:

```
# /opt/VRTS/bin/fsadm -b 500M /db01
```

```
# /opt/VRTS/bin/qiomkfile -r 300M /db01/dbfile
```

Monitoring tablespace free space with DB2 and extending tablespace containers

DB2 does not automatically make use of extended DMS files. When tablespace space needs to be extended, a number of DB2 commands must be run. Unlike raw devices, a Database Administrator can easily extend Quick I/O files online. Using this method, a Database Administrator can monitor the free space available in the DB2 tablespaces and use the `qiomkfile` command to grow the Quick I/O files online as needed (typically when the file is about 80 to 90% full). This method does not require you to lock out unused disk space for Quick I/O files. The free space on the file system is available for use by other applications.

Before extending tablespaces, make sure the following conditions have been met:

Prerequisites
- Log on as the DB2 instance owner.

Usage notes
- Monitor the free space available in the Quick I/O file, and grow the file as necessary with the `qiomkfile` command.
- A Database Administrator can grow underlying VxFS file systems online (provided the underlying disk or volume can be extended) using the `fsadm` command. See the `fsadm` (1M) manual page for more information.
- It is strongly recommended that if a DB2 tablespace is running out of space, that all containers are grown by the same amount. It is possible to add extra containers to the tablespace, but this results in DB2 performing a relayout of the data to balance usage across all available containers in the tablespace.
  Also refer to the DB2 Administration Guide section on *Managing Storage and the DB2 SQL Reference Guide* for information on the `ALTER TABLESPACE` command.

To monitor the free space available in a DB2 tablespace

- Use the following DB2 commands:

```
$ db2 connect to database
$ db2 list tablespaces show detail
$ db2 terminate
```

To extend a Quick I/O file using qiomkfile

- Use the `qiomkfile` command to extend the Quick I/O file (if the container is running low on free blocks):

```
# /opt/VRTS/bin/qiomkfile -e extend_amount filename
```

To extend a DB2 tablespace by a fixed amount

- Use the following DB2 commands:

```
$ db2 connect to database

$ db2 alter tablespace tablespace-name extend (ALL amount)

$ db2 terminate
```

This example shows how to monitor the free space on the tablespaces in database `PROD`:

```
$ db2 connect to PROD

$ db2 list tablespaces show detail

$ db2 terminate
```

This example shows how to extend the three DB2 containers owned by tablespace `EMP` by 500MB using the `qiomkfile` command:

```
# /opt/VRTS/bin/qiomkfile -e 500M tbsEMP_cont001

# /opt/VRTS/bin/qiomkfile -e 500M tbsEMP_cont002

# /opt/VRTS/bin/qiomkfile -e 500M tbsEMP_cont003
```

This example shows how to notify DB2 that all containers in tablespace `EMP` have grown by 500MB:

```
$ db2 connect to PROD

$ db2 alter tablespace EMP extend (ALL 500M)

$ db2 terminate
```

This example shows how to verify the newly allocated space on the tablespace `EMP` in database `PROD`:

```
$ db2 connect to PROD

$ db2 list tablespaces show detail

$ db2 terminate
```

Recreating Quick I/O files after restoring a database

If you need to restore your database and were using Quick I/O files, you can use the `qio_recreate` command to automatically recreate the Quick I/O files after you have performed a full database recovery. The `qio_recreate` command uses the `mkqio.dat` file, which contains a list of the Quick I/O files used by the database and the file sizes.

For information on recovering your database, refer to the documentation that came with your database software.

Before recreating Quick I/O with the `qio_recreate` command, make sure the following conditions have been met:

DB2 Prerequisites
- Recover your database before attempting to recreate the Quick I/O files.
- Log in as the DB2 instance owner (typically, the user ID db2inst1) to run the `qio_recreate` command.
- In the directory from which you run the `qio_recreate` command, you must have an existing `mkqio.dat` file.
- For DB2:
  The DB2DATABASE environment variable must be set.
- For Sybase:
  The SYBASE and DSQUERY environment variables must be set.

Usage notes
- The `qio_recreate` command supports only conventional Quick I/O files.
- See the `qio_recreate`(1M) manual page for more information.

To recreate Quick I/O files after recovering a database

- As DBA, use the `qio_recreate` command as follows:

  For DB2:

  ```
  $ /opt/VRTS/bin/qio_recreate
  ```

  For Sybase:

  ```
  $ /opt/VRTS/bin/qio_recreate
  ```

You will not see any output if the command is successful.

When you run the `qio_recreate` command, the following actions occur:

If...	Then...
a Quick I/O file is missing	the Quick I/O file is recreated.
a symbolic link from a regular VxFS file to a Quick I/O file is missing	the symbolic link is recreated.
a symbolic link and its associated Quick I/O file are missing	both the link and the Quick I/O file are recreated.
a Quick I/O file is missing and the regular VxFS file that it is symbolically linked to is not the original VxFS file	the Quick I/O file is not recreated and a warning message is displayed.
a Quick I/O file is smaller than the size listed in the `mkqio.dat` file	the Quick I/O file is not recreated and a warning message is displayed.

Disabling Quick I/O

Before disabling Quick I/O, make sure the following condition has been met:

Prerequisite

The file system you are planning to remount must be located in the `/etc/filesystems` file.

The file system you are planning to remount must be located in the `/etc/fstab` file.

For DB2: If you need to disable the Quick I/O feature, you need to remount the VxFS file system using a special mount option.

If you need to disable the Quick I/O feature, you first need to convert any Quick I/O files back to regular VxFS files. Then, remount the VxFS file system using a special mount option.

Warning: For DB2: do not disable Quick I/O on VxFS file systems containing Quick I/O containers of type `DEVICE`. Doing so causes the tablespace containing them to go offline.

To remount the file system with Quick I/O disabled For DB2

- Use the `mount -o noqio` command as follows:

```
# /opt/VRTS/bin/mount -V vxfs -o remount,noqio special
     /mount_point
```

For example, to remount file system `db01` with Quick I/O disabled:

```
# /opt/VRTS/bin/mount -V vxfs -o remount,noqio \
 /dev/vx/dsk/dbdg/vol01 /db01
```

To disable Quick I/O for DB2

1. If the database is active, make it inactive by either shutting down the instance or disabling user connections.
2. To remount the file system with Quick I/O disabled, use the command as follows:

```
# /opt/VRTS/bin/mount -V vxfs -o remount,noqio /mount_point
```

To disable Quick I/O for Sybase

1. If the database is running, shut it down.
2. To remount the file system with Quick I/O disabled, use the command as follows:

```
# /opt/VRTS/bin/mount -V vxfs -o remount,noqio /mount_point
```

Chapter 10

Improving DB2 database performance with Veritas Concurrent I/O

This chapter includes the following topics:

- About Concurrent I/O
- Tasks for enabling and disabling Concurrent I/O

About Concurrent I/O

Concurrent I/O improves the performance of regular files on a VxFS file system without the need for extending namespaces and presenting the files as devices. This simplifies administrative tasks and allows databases, which do not have a sequential read/write requirement, to access files concurrently. This chapter describes how to use the Concurrent I/O feature.

Concurrent I/O improves the performance of regular files on a VxFS file system. This simplifies administrative tasks and allows databases, which do not have a sequential read/write requirement, to access files concurrently. This chapter describes how to use the Concurrent I/O feature.

With DB2 8.2.2 or later, you can use the Concurrent I/O feature to improve data write performance for DMS tablespaces with FILE containers. Concurrent I/O can also improve data write performance for most SMS tablespaces.

Quick I/O is an alternative solution for DMS tablespaces.

In some cases (for example, if the system has extra memory), Cached Quick I/O may further enhance performance.

How Concurrent I/O works

Traditionally, Linux semantics require that read and write operations on a file occur in a serialized order. Because of this, a file system must enforce strict ordering of overlapping read and write operations. However, databases do not usually require this level of control and implement concurrency control internally, without using a file system for order enforcement.

Traditionally, UNIX semantics require that read and write operations on a file occur in a serialized order. Because of this, a file system must enforce strict ordering of overlapping read and write operations. However, databases do not usually require this level of control and implement concurrency control internally, without using a file system for order enforcement.

The Concurrent I/O feature removes these semantics from the read and write operations for databases and other applications that do not require serialization.

The benefits of using Concurrent I/O are:

- Concurrency between a single writer and multiple readers
- Concurrency among multiple writers
- Minimalization of serialization for extending writes
- All I/Os are direct and do not use file system caching
- I/O requests are sent directly to file systems
- Inode locking is avoided

Tasks for enabling and disabling Concurrent I/O

Concurrent I/O is not turned on by default and must be enabled manually. You will also have to manually disable Concurrent I/O if you choose not to use it in the future.

You can perform the following tasks:

- Enable Concurrent I/O
- Disable Concurrent I/O

Enabling Concurrent I/O for DB2

Because you do not need to extend name spaces and present the files as devices, you can enable Concurrent I/O on regular files.

For DB2, you can enable an entire file system to use Concurrent I/O or you can enable specific SMS containers to use Concurrent I/O. If you enable a specific SMS container, the rest of the file system will use the regular buffer I/O.

Before enabling Concurrent I/O, review the following:

Prerequisites

- To use the Concurrent I/O feature, the file system must be a VxFS file system.
- Make sure the mount point on which you plan to mount the file system exists.
- Make sure the DBA can access the mount point.

Usage notes

- Files that are open and using Concurrent I/O cannot be opened simultaneously by a different user not using the Concurrent I/O feature.
- Veritas NetBackup cannot backup a database file if the file is open and using Concurrent I/O. However, you can still backup the database online using the `DB2 BACKUP` utility.
- When a file system is mounted with the Concurrent I/O option, do not enable Quick I/O. DB2 will not be able to open the Quick I/O files and the instance start up will fail. (Quick I/O is not available on Linux.)
- If the Quick I/O feature is availabe, do not use any Quick I/O tools if the database is using Concurrent I/O.
- See the `mount_vxfs`(1M) manual page for more information about mount settings.

For DB2, /*mount_point* is the directory in which you can put data containers of the SMS tablespaces using the Concurrent I/O feature.

Note: This applies to both creating a new tablespace to use Concurrent I/O or enabling an existing tablespace to use Concurrent I/O.

For example for DB2 to mount a file system named `/datavol` on a mount point named `/db2data`:

```
  # /usr/sbin/mount -V vxfs -o cio /dev/vx/dsk/db2dg/datavol \
/db2data

  # /usr/sbin/mount -t vxfs -o cio /dev/vx/dsk/db2dg/datavol \
/db2data
```

To enable Concurrent I/O on a new SMS container using the namefs -o cio option

- Using the `mount`command, mount the directory in which you want to put data containers of the SMS tablespaces using the Concurrent I/O feature.

  ```
  # /usr/sbin/mount -Vt namefs -o cio /path_name /new_mount_point
  ```

where:

- */path_name* is the directory in which the files that will be using Concurrent I/O reside
- */new_mount_point* is the new target directory that will use the Concurrent I/O feature

The following is an example of mounting a directory (where the new SMS containers are located) to use Concurrent I/O.

To mount an SMS container named `/container1` on a mount point named `/mysms`:

```
# /usr/sbin/mount -Vt namefs -o cio /datavol/mysms/container1 /mysms
```

To enable Concurrent I/O on an existing SMS container using the namefs -o cio option

1. Stop the DB2 instance using the `db2stop` command.
2. Make the directory that will have Concurrent I/O turned on available using the `mv` command.

   ```
   # mv /mydb/mysmsdir /mydb/mysmsdir2
   ```

3. Remount `/mydb/mysmsdir2` on `/mydb/mysmsdir` using the `mount` command with the `-o cio` option.

   ```
   # mount -Vt namefs -o cio /mydb/mysmsdir2 /mydb/mysmsdir
   ```

4. Start the DB2 instance using the `db2start` command.

   ```
   # db2stop
   # mv /mydb/mysmsdir /mydb/mysmsdir2
   # mount -Vt namefs -o cio /mydb/mysmsdir2 /mydb/mysmsdir
   # db2start
   ```

This example shows how to mount a directory for an existing SMS container to use Concurrent I/O.

To enable Concurrent I/O on a DB2 tablespace when creating the tablespace

1. Use the `db2 -v "create regular tablespace..." no file system caching` command with the `no file system caching` option.
2. Set all other parameters according to your system requirements.

To enable Concurrent I/O on an existing DB2 tablespace

- Use the DB2 `no file system caching` option as follows:

```
# db2 -v "alter tablespace tablespace_name no file system caching"
```

  where *tablespace_name* is the name of the tablespace for which you are enabling Concurrent I/O.

To verify that Concurrent I/O has been set for a particular DB2 tablespace

1. Use the DB2 `get snapshot` option to check for Concurrent I/O.

```
# db2 -v "get snapshot for tablespaces on dbname"
```

   where *dbname* is the database name.

2. Find the tablespace you want to check and look for the `File system caching` attribute. If you see `File system caching = No`, then Concurrent I/O is enabled.

Disabling Concurrent I/O for DB2

If you need to disable Concurrent I/O, use the DB2 `file system caching` option.

Note: If you used the `namefs -o cio` option with the `mount`command to mount a directory to enable Concurrent I/O, make sure you remount without that option as well. Also, if you follow the directions for enabling Concurrent I/O on an existing SMS container, rename the directory back to the original name.

To disable Concurrent I/O on a DB2 tablespace

- Use the DB2 `file system caching` option as follows:

```
# db2 -v "alter tablespace tablespace_name file system caching"
```

  where *tablespace_name* is the name of the tablespace for which you are disabling Concurrent I/O.

Section 5

Using point-in-time copies

- Chapter 11. Understanding point-in-time copy methods
- Chapter 12. Considerations for DB2 point-in-time copies
- Chapter 13. Administering third-mirror break-off snapshots
- Chapter 14. Administering Storage Checkpoints
- Chapter 15. Backing up and restoring with Netbackup in an SFHA environment

Chapter 11

Understanding point-in-time copy methods

This chapter includes the following topics:

- About point-in-time copies
- When to use point-in-time copies
- About Storage Foundation point-in-time copy technologies
- Point-in-time copy solutions supported by SFDB tools
- About snapshot modes supported by Storage Foundation for Databases (SFDB) tools
- Volume-level snapshots
- Storage Checkpoints

About point-in-time copies

Storage Foundation offers a flexible and efficient means of managing business-critical data. Storage Foundation lets you capture an online image of an actively changing database at a given instant, called a point-in-time copy.

More and more, the expectation is that the data must be continuously available (24x7) for transaction processing, decision making, intellectual property creation, and so forth. Protecting the data from loss or destruction is also increasingly important. Formerly, data was taken out of service so that the data did not change

while data backups occured; however, this option does not meet the need for minimal down time.

A point-in-time copy enables you to maximize the online availability of the data. You can perform system backup, upgrade, or perform other maintenance tasks on the point-in-time copies. The point-in-time copies can be processed on the same host as the active data, or a different host. If required, you can offload processing of the point-in-time copies onto another host to avoid contention for system resources on your production server. This method is called off-host processing. If implemented correctly, off-host processing solutions have almost no impact on the performance of the primary production system.

When to use point-in-time copies

The following typical activities are suitable for point-in-time copy solutions implemented using Veritas InfoScale FlashSnap:

- Data backup —Many enterprises require 24 x 7 data availability. They cannot afford the downtime involved in backing up critical data offline. By taking snapshots of your data, and backing up from these snapshots, your business-critical applications can continue to run without extended downtime or impacted performance.
- Providing data continuity —To provide continuity of service in the event of primary storage failure, you can use point-in-time copy solutions to recover application data. In the event of server failure, you can use point-in-time copy solutions in conjunction with the high availability cluster functionality of SFCFSHA or SFHA.
- Decision support analysis and reporting—Operations such as decision support analysis and business reporting may not require access to real-time information. You can direct such operations to use a replica database that you have created from snapshots, rather than allow them to compete for access to the primary database. When required, you can quickly resynchronize the database copy with the data in the primary database.
- Testing and training—Development or service groups can use snapshots as test data for new applications. Snapshot data provides developers, system testers and QA groups with a realistic basis for testing the robustness, integrity and performance of new applications.
- Database error recovery—Logic errors caused by an administrator or an application program can compromise the integrity of a database. You can recover a database more quickly by restoring the database files by using Storage Checkpoints or a snapshot copy than by full restoration from tape or other backup media.

Use Storage Checkpoints to quickly roll back a database instance to an earlier point in time.

- Cloning data—You can clone your file system or application data. This functionality enable you to quickly and efficiently provision virtual desktops.

All of the snapshot solutions mentioned above are also available on the disaster recovery site, in conjunction with Volume Replicator.

For more information about snapshots with replication, see the *Veritas InfoScale 7.4.2 Replication Administrator's Guide*.

Storage Foundation provides several point-in-time copy solutions that support your needs, including the following use cases:

- Creating a replica database for decision support.
- Backing up and recovering a database with snapshots.
- Backing up and recovering an off-host cluster file system
- Backing up and recovering an online database.

About Storage Foundation point-in-time copy technologies

This topic introduces the point-in-time copy solutions that you can implement using the Veritas FlashSnap™ technology. Veritas FlashSnap technology requires a Veritas InfoScale Enterprise or Storage licenses.

Veritas InfoScale FlashSnap offers a flexible and efficient means of managing business critical data. It allows you to capture an online image of actively changing data at a given instant: a point-in-time copy. You can perform system backup, upgrade and other maintenance tasks on point-in-time copies while providing continuous availability of your critical data. If required, you can offload processing of the point-in-time copies onto another host to avoid contention for system resources on your production server.

The following kinds of point-in-time copy solution are supported by the FlashSnap license:

- Volume-level solutions. There are several types of volume-level snapshots. These features are suitable for solutions where separate storage is desirable to create the snapshot. For example, lower-tier storage. Some of these techniques provided exceptional offhost processing capabilities.
- File system-level solutions use the Storage Checkpoint feature of Veritas File System. Storage Checkpoints are suitable for implementing solutions where storage space is critical for:

- File systems that contain a small number of mostly large files.
- Application workloads that change a relatively small proportion of file system data blocks (for example, web server content and some databases).
- Applications where multiple writable copies of a file system are required for testing or versioning.

See "Storage Checkpoints" on page 95.

- File level snapshots.
  The FileSnap feature provides snapshots at the level of individual files.

Point-in-time copy solutions supported by SFDB tools

Storage Foundation for Databases (SFDB) tools provide a database-specific command line to create point-in-time copies of your DB2 database. SFDB tools use the underlying features of Storage Foundation to perform these operations.. For ease of use, the SFDB command line enables you to perform the point-in-time copy operations on the DB2 database with fewer steps. Also, the SFDB command line enables you to perform functions specific to DB2 databases.

Table 11-1 provides a comparison of the point-in-time copy solutions supported by SFDB tools.

Table 11-1 Comparison of Point-in-time Copy Solutions

	FlashSnap	Database Storage Checkpoints
Underlying technology	Volume snapshots (third-mirror break-off snapshots)	File system checkpoints
Possibility of off-host processing	Yes	Yes (requires Cluster File System)
Additional storage requirements	Additional mirror plexes are required. Plexes are full copies of the original data.	Minimal (uses copy-on-write)
Performance impact after taking the point-in-time copy	None	Copy-on-write penalty
Support for multiple clones from a single point-in-time copy	No. However, different mirrors with different snapshots can be used to create multiple clones.	Yes
Supported snapshot modes	Online, Offline	Online, Offline

About snapshot modes supported by Storage Foundation for Databases (SFDB) tools

The snapshot modes supported by SFDB tools are:

- Online
- Offline

Table 11-2 Descriptions of Snapshot Modes

Snapshot mode	Description
Online	The online snapshot mode: ■ Is like an online or a hot backup of the application or the database and it is suitable as a backup image. ■ Is the default and the recommended snapshot mode. In this mode, the DB2 database is put into write suspend mode during the snapshot operation. ■ Has the least performance impact on the application or the database. ■ Allows the user to perform manual point-in-time recovery of a clone based on the snapshot or of the primary application after a restore operation.
Offline	The offline snapshot mode: ■ Is like a cold backup of the application or the database and it is suitable as a backup image. ■ Requires the application to be offline. ■ Requires the snapshot configuration to be validated when the application is online. ■ Is fastest amongst the snapshot modes.

Volume-level snapshots

A volume snapshot is an image of a Veritas Volume Manager (VxVM) volume at a given point in time. You can also take a snapshot of a volume set.

Volume snapshots allow you to make backup copies of your volumes online with minimal interruption to users. You can then use the backup copies to restore data that has been lost due to disk failure, software errors or human mistakes, or to create replica volumes for the purposes of report generation, application development, or testing.

Volume snapshots can also be used to implement off-host online backup.

Physically, a snapshot may be a full (complete bit-for-bit) copy of the data set, or it may contain only those elements of the data set that have been updated since snapshot creation. The latter are sometimes referred to as allocate-on-first-write snapshots, because space for data elements is added to the snapshot image only when the elements are updated (overwritten) for the first time in the original data set. Storage Foundation allocate-on-first-write snapshots are called space-optimized snapshots.

Persistent FastResync of volume snapshots

If persistent FastResync is enabled on a volume, VxVM uses a FastResync map to keep track of which blocks are updated in the volume and in the snapshot.

When snapshot volumes are reattached to their original volumes, persistent FastResync allows the snapshot data to be quickly refreshed and re-used. Persistent FastResync uses disk storage to ensure that FastResync maps survive both system and cluster crashes. If persistent FastResync is enabled on a volume in a private disk group, incremental resynchronization can take place even if the host is rebooted.

Persistent FastResync can track the association between volumes and their snapshot volumes after they are moved into different disk groups. After the disk groups are rejoined, persistent FastResync allows the snapshot plexes to be quickly resynchronized.

Data integrity in volume snapshots

A volume snapshot captures the data that exists in a volume at a given point in time. As such, VxVM does not have any knowledge of data that is cached in memory by the overlying file system, or by applications such as databases that have files open in the file system. Snapshots are always crash consistent, that is, the snapshot can be put to use by letting the application perform its recovery. This is similar to how the application recovery occurs after a server crash. If the `fsgen` volume usage type is set on a volume that contains a mounted Veritas File System (VxFS), VxVM coordinates with VxFS to flush data that is in the cache to the volume. Therefore, these snapshots are always VxFS consistent and require no VxFS recovery while mounting.

For databases, a suitable mechanism must additionally be used to ensure the integrity of tablespace data when the volume snapshot is taken. The facility to temporarily suspend file system I/O is provided by most modern database software. The examples provided in this document illustrate how to perform this operation. For ordinary files in a file system, which may be open to a wide variety of different applications, there may be no way to ensure the complete integrity of the file data other than by shutting down the applications and temporarily unmounting the file system. In many cases, it may only be important to ensure the integrity of file data

that is not in active use at the time that you take the snapshot. However, in all scenarios where application coordinate, snapshots are crash-recoverable.

Third-mirror break-off snapshots

A plex break-off snapshot uses an additional mirror to create the snapshot. Although you can create a plex break-off snapshot for a single plex volume, typically you take a snapshot of a mirrored volume. A mirrored volume has more than one plex or mirror, each of which is a copy of the data. The snapshot operation "breaks off" the plex, which becomes the snapshot volume. You can break off an existing plex or add a new plex specifically to serve as the snapshot mirror. Generally, you want to maintain redundancy for the original volume. If the original volume is a mirrored volume with two plexes, you add a third mirror for the snapshot. Hence, this type of snapshot is also known as a third-mirror snapshot.

The snapshot plex must be on a different disk from the existing plexes in the volume, within the same disk group. The disk must have enough disk space to contain the contents of the existing volume. If you have a one terabyte volume, you must have an additional one terabyte of disk space.

When you create the snapshot, the plexes are separated into two volumes. The original volume retains its original plex or plexes. The snapshot volume contains the snapshot plex. The original volume continues to take on I/O. The snapshot volume retains the data at the point of time when the snapshot was created, until you choose to perform processing on that volume.

You can make multiple snapshots, so you can have multiple copies of the original data.

Third-mirror break-off snapshots are suitable for write-intensive volumes (such as for database redo logs) where the copy-on-write mechanism of space-optimized or full-sized instant snapshots might degrade performance.

Storage Checkpoints

A Storage Checkpoint is a persistent image of a file system at a given instance in time. Storage Checkpoints use a copy-on-write technique to reduce I/O overhead by identifying and maintaining only those file system blocks that have changed since a previous Storage Checkpoint was taken. Storage Checkpoints have the following important features:

- Storage Checkpoints persist across system reboots and crashes.
- A Storage Checkpoint can preserve not only file system metadata and the directory hierarchy of the file system, but also user data as it existed when the Storage Checkpoint was taken.

- After creating a Storage Checkpoint of a mounted file system, you can continue to create, remove, and update files on the file system without affecting the image of the Storage Checkpoint.
- Unlike file system snapshots, Storage Checkpoints are writable.
- To minimize disk space usage, Storage Checkpoints use free space in the file system.

Storage Checkpoints and the Storage Rollback feature of Storage Foundation for Databases enable rapid recovery of databases from logical errors such as database corruption, missing files and dropped table spaces. You can mount successive Storage Checkpoints of a database to locate the error, and then roll back the database to a Storage Checkpoint before the problem occurred.

How Storage Checkpoints differ from snapshots

Storage Checkpoints differ from Veritas File System snapshots in the following ways because they:

- Allow write operations to the Storage Checkpoint itself.
- Persist after a system reboot or failure.
- Share the same pool of free space as the file system.
- Maintain a relationship with other Storage Checkpoints by identifying changed file blocks since the last Storage Checkpoint.
- Can have multiple, read-only Storage Checkpoints that reduce I/O operations and required storage space because the most recent Storage Checkpoint is the only one that accumulates updates from the primary file system.
- Can restore the file system to its state at the time that the Storage Checkpoint was taken.

Various backup and replication solutions can take advantage of Storage Checkpoints. The ability of Storage Checkpoints to track the file system blocks that have changed since the last Storage Checkpoint facilitates backup and replication applications that only need to retrieve the changed data. Storage Checkpoints significantly minimize data movement and may promote higher availability and data integrity by increasing the frequency of backup and replication solutions.

Storage Checkpoints can be taken in environments with a large number of files, such as file servers with millions of files, with little adverse impact on performance. Because the file system does not remain frozen during Storage Checkpoint creation, applications can access the file system even while the Storage Checkpoint is taken. However, Storage Checkpoint creation may take several minutes to complete depending on the number of files in the file system.

How a Storage Checkpoint works

The Storage Checkpoint facility freezes the mounted file system (known as the primary fileset), initializes the Storage Checkpoint, and thaws the file system. Specifically, the file system is first brought to a stable state where all of its data is written to disk, and the freezing process momentarily blocks all I/O operations to the file system. A Storage Checkpoint is then created without any actual data; the Storage Checkpoint instead points to the block map of the primary fileset. The thawing process that follows restarts I/O operations to the file system.

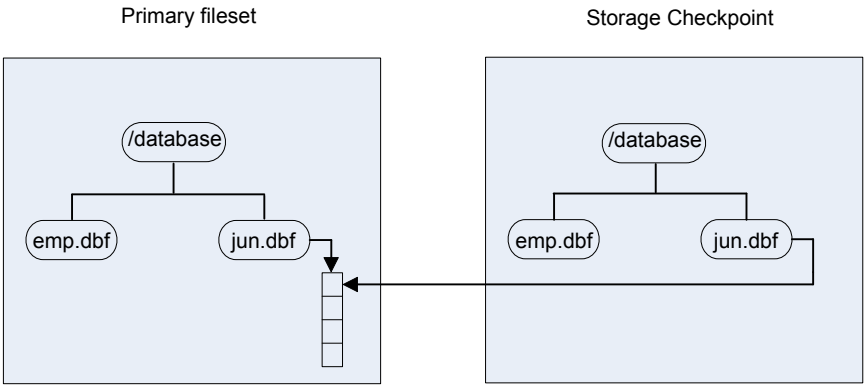
You can create a Storage Checkpoint on a single file system or a list of file systems. A Storage Checkpoint of multiple file systems simultaneously freezes the file systems, creates a Storage Checkpoint on all of the file systems, and thaws the file systems. As a result, the Storage Checkpoints for multiple file systems have the same creation timestamp. The Storage Checkpoint facility guarantees that multiple file system Storage Checkpoints are created on all or none of the specified file systems, unless there is a system crash while the operation is in progress.

Note: The calling application is responsible for cleaning up Storage Checkpoints after a system crash.

A Storage Checkpoint of the primary fileset initially contains only pointers to the existing data blocks in the primary fileset, and does not contain any allocated data blocks of its own.

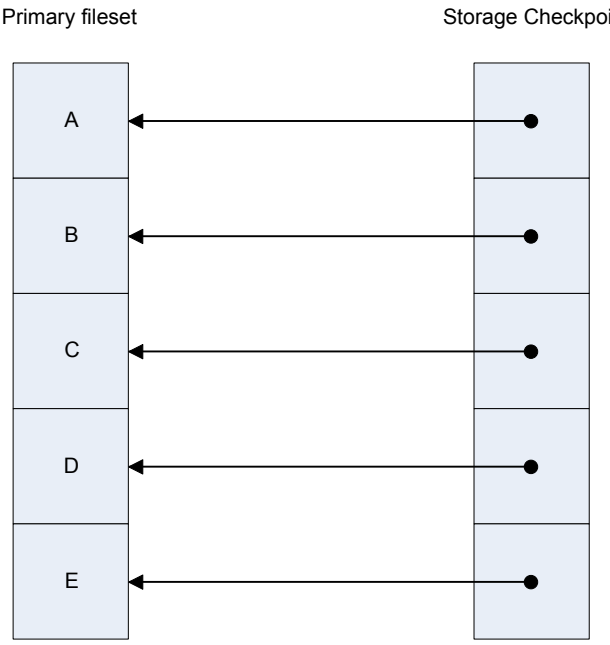
Figure 11-1 shows the file system `/database` and its Storage Checkpoint. The Storage Checkpoint is logically identical to the primary fileset when the Storage Checkpoint is created, but it does not contain any actual data blocks.

Figure 11-1 Primary fileset and its Storage Checkpoint



In Figure 11-2, a square represents each block of the file system. This figure shows a Storage Checkpoint containing pointers to the primary fileset at the time the Storage Checkpoint is taken, as in Figure 11-1.

Figure 11-2 Initializing a Storage Checkpoint



The Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. VxFS updates a Storage Checkpoint by using the copy-on-write technique.

See "Copy-on-write" on page 98.

Copy-on-write

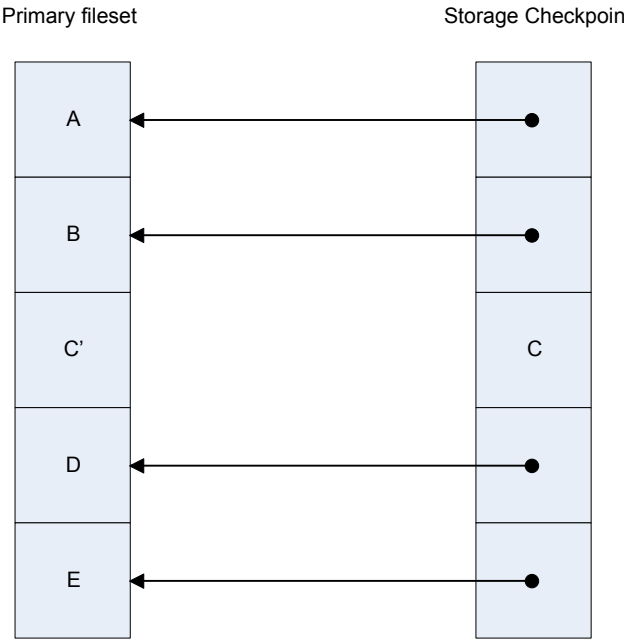
In Figure 11-3, the third data block in the primary fileset originally containing C is updated.

Before the data block is updated with new data, the original data is copied to the Storage Checkpoint. This is called the copy-on-write technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken.

Every update or write operation does not necessarily result in the process of copying data to the Storage Checkpoint because the old data needs to be saved only once. As blocks in the primary fileset continue to change, the Storage Checkpoint

accumulates the original data blocks. In this example, subsequent updates to the third data block, now containing C', are not copied to the Storage Checkpoint because the original image of the block containing C is already saved.

Figure 11-3 Updates to the primary fileset



Storage Checkpoint visibility

With the `ckptautomnt` mount option, all Storage Checkpoints are made accessible automatically through a directory in the root directory of the file system that has the special name `.checkpoint`, which does not appear in directory listings. Inside this directory is a directory for each Storage Checkpoint in the file system. Each of these directories behave as a mount of the corresponding Storage Checkpoint, with the following exceptions:

- External applications, such as NFS, see the files as part of the original mount point. Thus, no additional NFS exports are necessary.
- Inode numbers exposed to applications can be made unique, depending on a mount option.

The Storage Checkpoints are automounted internally, but the operating system does not know about the automounting. This means that Storage Checkpoints cannot be mounted manually, and they do not apear in the list of mounted file

systems. When Storage Checkpoints are created or deleted, entries in the Storage Checkpoint directory are automatically updated. If a Storage Checkpoint is removed with the `-f` option while a file in the Storage Checkpoint is still in use, the Storage Checkpoint is force unmounted, and all operations on the file fail with the EIO error.

If there is already a file or directory named `.checkpoint` in the root directory of the file system, such as a directory created with an older version of Veritas File System (VxFS) or when Storage Checkpoint visibility feature was disabled, the fake directory providing access to the Storage Checkpoints is not accessible. With this feature enabled, attempting to create a file or directory in the root directory with the name `.checkpoint` fails with the EEXIST error.

Note: If an auto-mounted Storage Checkpoint is in use by an NFS mount, removing the Storage Checkpoint might succeed even without the forced (`-f`) option.

Storage Checkpoints and 64-bit inode numbers

The inode number of a file is the same across Storage Checkpoints. For example, if the file `file1` exists in a file system and a Storage Checkpoint is taken of that file system, running the `stat` command on `file1` in the original file system and in the Storage Checkpoint returns the same value in `st_ino`. The combination of `st_ino` and `st_dev` should uniquely identify every file in a system. This is usually not a problem because Storage Checkpoints get mounted separately, so `st_dev` is different. When accessing files in a Storage Checkpoint through the Storage Checkpoint visibility extension, `st_dev` is the same for all Storage Checkpoints as well as for the original file system. This means files can no longer be identified uniquely by `st_ino` and `st_dev`.

In general, uniquely identifying all files in a system is not necessary. However, there can be some applications that rely on unique identification to function properly. For example, a backup application might check if a file is hard-linked to another file by calling `stat` on both and checking if `st_ino` and `st_dev` are the same. If a backup application were told to back up two clones through the Storage Checkpoint visibility extension at the same time, the application can erroneously deduce that two files are the same even though the files contain different data.

By default, Storage Foundation (SF) does not make inode numbers unique. However, you can specify the `uniqueino` mount option to enable the use of unique 64-bit inode numbers. You cannot change this option during a remount.

About Database Rollbacks using Storage Checkpoints

Each Storage Checkpoint is a consistent, point-in-time image of a file system, and Storage Rollback is the restore facility for these on-disk backups. Storage Rollback

rolls back changed blocks contained in a Storage Checkpoint into the primary file system for faster database restoration.

Storage Checkpoints and Rollback process

A Storage Checkpoint is a disk and I/O efficient snapshot technology for creating a "clone" of a currently mounted file system (the primary file system). Like a snapshot file system, a Storage Checkpoint appears as an exact image of the snapped file system at the time the Storage Checkpoint was made. However, unlike a snapshot file system that uses separate disk space, all Storage Checkpoints share the same free space pool where the primary file system resides.

Note: A database Storage Checkpoint can be mounted as read only or read-write, allowing access to the files as if it were a regular file system.

Initially, a Storage Checkpoint contains no data. The Storage Checkpoint only contains the inode list and the block map of the primary fileset. This block map points to the actual data on the primary file system. Because only the inode list and block map are required and no data is copied, creating a Storage Checkpoint takes only a few seconds and very little space.

A Storage Checkpoint initially satisfies read requests by finding the data on the primary file system, using its block map copy, and returning the data to the requesting process. When a write operation changes a data block in the primary file system, the old data is first copied to the Storage Checkpoint, and then the primary file system is updated with the new data. The Storage Checkpoint maintains the exact view of the primary file system at the time the Storage Checkpoint was taken. Subsequent writes to block n on the primary file system do not result in additional copies to the Storage Checkpoint because the old data only needs to be saved once. As data blocks are changed on the primary file system, the Storage Checkpoint gradually fills with the original data copied from the primary file system, and less and less of the block map in the Storage Checkpoint points back to blocks on the primary file system.

Database Storage Rollback restores a database on the primary file systems to the point-in-time image created during a Storage Checkpoint.

Database Storage Rollback is accomplished by copying the "before" images from the appropriate Storage Checkpoint back to the primary file system. As with Storage Checkpoints, Database Storage Rollback restores at the block level, rather than at the file level. Database Storage Rollback is executed using the `vxsfadm` command with the `-o restore` operation.

For example:

```
$ $ /opt/VRTSdbed/bin/vxsfadm -s checkpoint /
 -a db2 -o restore --checkpoint_name checkpoint1
```

Mountable Storage Checkpoints can be used for a wide range of application solutions including the following:

- Backups
- Investigations into data integrity
- Staging upgrades
- Database modifications
- Data replication solutions

If you mount a Storage Checkpoint as read-write, the command will not allow you to roll back to this Storage Checkpoint. This ensures that any Storage Checkpoint data that has been modified incorrectly cannot be a source of any database corruption. When a Storage Checkpoint is mounted as read-write, then a "shadow" Storage Checkpoint of the original Storage Checkpoint gets created, and this "shadow" Storage Checkpoint is mounted as read-write. This allows the database to still be rolled back to the original Storage Checkpoint.

Storage Checkpoint space management considerations

Several operations, such as removing or overwriting a file, can fail when a file system containing Storage Checkpoints runs out of space. If the system cannot allocate sufficient space, the operation will fail.

Database applications usually preallocate storage for their files and may not expect a write operation to fail. During user operations such as `create` or `mkdir`, if the file system runs out of space, removable Storage Checkpoints are deleted. This ensures that applications can continue without interruptions due to lack of disk space. Non-removable Storage Checkpoints are not automatically removed under such `ENOSPC` conditions. Veritas recommends that you create only removable Storage Checkpoints. However, during certain administrative operations, such as using the `fsadm` command, using the `qiomkfile` command, and creating a Storage Checkpoint with the `fsckptadm` command, even if the file system runs out of space, removable Storage Checkpoints are not deleted.

When the kernel automatically removes the Storage Checkpoints, it applies the following policies:

- Remove as few Storage Checkpoints as possible to complete the operation.
- Never select a non-removable Storage Checkpoint.

- Select a nodata Storage Checkpoint only when data Storage Checkpoints no longer exist.
- Remove the oldest Storage Checkpoint first.
- Remove a Storage Checkpoint even if it is mounted. New operations on such a removed Storage Checkpoint fail with the appropriate error codes.
- If the oldest Storage Checkpoint is non-removable, then the oldest removable Storage Checkpoint is selected for removal. In such a case, data might be required to be pushed to a non-removable Storage Checkpoint, which might fail and result in the file system getting marked for a FULLFSCK. To prevent this occurrence, Veritas recommends that you only create removable Storage Checkpoints.

Chapter 12

Considerations for DB2 point-in-time copies

This chapter includes the following topics:

- Considerations for DB2 database layouts
- Supported DB2 configurations

Considerations for DB2 database layouts

The following considerations for database layouts apply if you are using Storage Foundation for Databases (SFDB) tools:

- All database files must be on VxFS file systems. These include all paths listed in the SYSIBMADM.DB_PATHS view.
- All the underlying volumes must be VxVM volumes.
- For third-mirror break-off snapshots (FlashSnap), and when creating snapshots of archived logs, the archive log location must be on VxFS on a separate VxVM volume.
- When performing vxsfadm operations, certain operations may fail, and when vxsfadm attempts to recover from this failure the recovery may also fail. Due to this, the configuration would be in an error state, and some of the application changes would not be reverted. In this scenario, you may need to revert the changes manually followed by an error recovery.

  For example, when creating a checkpoint for a DB2 database, the database is put into the write-suspend mode and then a checkpoint is created following which the database is removed from write-suspend. If the checkpoint creation fails, and the recovery operation is unsuccessful, then the database would remain in the write-suspend mode.

Supported DB2 configurations

For information on supported DB2 versions, refer to the database support matrix:

https://www.veritas.com/support/en_US/article.DOC5082

Storage Foundation for Databases (SFDB) tools are supported with DB2 10.1, DB2 10.5, DB2 9.5 Fixpack 2 or later, and 9.7 releases on AIX. The following considerations apply.

- Partitioned DB2 databases are not supported.
- The following point-in-time copy operations are supported in this release.
  - Database Storage Checkpoints
  - Third-mirror break-off snapshots (FlashSnap)

Storage Foundation for Databases (SFDB) tools are supported with DB2 10.1, DB2 10.5, DB2 9.5 Fixpack 2 or later, and 9.7 releases Linux platforms. The following considerations apply

- Partitioned DB2 databases are not supported.
- The following point-in-time copy operations are supported in this release.
  - Database Storage Checkpoints
  - Third-mirror break-off snapshots (FlashSnap)

Chapter 13

Administering third-mirror break-off snapshots

This chapter includes the following topics:

- Database FlashSnap for cloning
- Preparing hosts and storage for Database FlashSnap
- Creating a clone of a database by using Database FlashSnap
- Resynchronizing mirror volumes with primary volumes
- Cloning a database on the secondary host

Database FlashSnap for cloning

Veritas Database FlashSnap helps to create a point-in-time copy of a database for backup and off-host processing. Database FlashSnap lets you make backup copies of your volumes online and with minimal interruption to users.

Database FlashSnap lets you capture an online image of an actively changing database at a given instant that is known as a snapshot. A snapshot copy of the database is referred to as a database snapshot. You can use a database snapshot on the same host as the production database or on a secondary host sharing the same storage. A database snapshot can be used for off-host processing applications, such as backup, data warehousing, and decision-support queries. When the snapshot is no longer needed, the database administrator can import the original snapshot back to the primary host and resynchronize the snapshot to the original database volumes. Database FlashSnap commands are executed from the command line interface.

Database FlashSnap advantages

Database FlashSnap provides the following advantages:

- The database snapshot can be used on the same host as the production database or on a secondary host sharing the same storage.
- In many companies, there is a clear separation between the roles of system administrators and database administrators. Creating database snapshots typically requires superuser (root) privileges, the privileges that database administrators do not usually have. Because superuser privileges are not required, Database FlashSnap overcomes these obstacles by enabling database administrators to easily create consistent snapshots of the database.

Preparing hosts and storage for Database FlashSnap

Review the following details to prepare the hosts and storage for Database FlashSnap.

Setting up hosts

Database FlashSnap requires sufficient disk space in the disk group to add a mirror of equal size of the existing database.

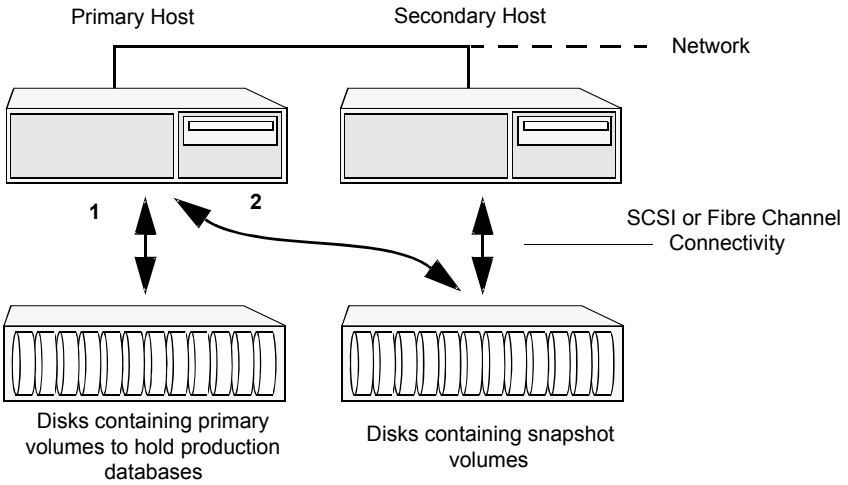
Setting up a storage configuration for Database FlashSnap operations is a system administrator's responsibility and requires superuser (root) privileges. Database FlashSnap utilities do not address setting up an appropriate storage configuration.

Database FlashSnap off-host configuration

A Database FlashSnap off-host configuration allows CPU- and I/O-intensive operations to be performed for online backup and decision support without degrading the performance of the primary host running the production database. Both the primary and secondary hosts share the storage in which the snapshot database is created. Both the primary and secondary hosts have access to the disks containing the snapshot volumes.

Figure 13-1 shows a Database FlashSnap off-host configuration.

Figure 13-1 Example of an off-host Database FlashSnap solution



Note: If you plan to use the FlashSnap feature in a VVR environment, then perform the FlashSnap setup prerequisite instructions after configuring VVR. In case you already have VVR configured, then make sure that the storage for mirror or snapshot volumes is deportable.

For information on host and storage requirements for an off-host configuration:

See "Requirements for an off-host database configuration" on page 43.

Creating a snapshot mirror of a volume or volume set used by the database

With Database FlashSnap, you can mirror the volumes used by the database to a separate set of disks, and those mirrors can be used to create a snapshot of the database. These snapshot volumes can be split and placed in a separate disk group. This snapshot disk group can be imported on a separate host, which shares the same storage with the primary host. The snapshot volumes can be resynchronized periodically with the primary volumes to get recent changes of the datafiles. If the primary datafiles become corrupted, you can quickly restore them from the snapshot volumes. Snapshot volumes can be used for a variety of purposes, including backup and recovery, and creating a clone database.

You must create snapshot mirrors for all of the volumes used by the database datafiles before you can create a snapshot of the database. This section describes the procedure used to create snapshot mirrors of volumes.

Use the `vxsnap` command to create a snapshot mirror or synchronize a snapshot mirror.

Prerequisites

- You must be logged in as superuser (root).
- The disk group must be version 110 or later.
  For more information on disk group versions, see the vxdg(1M) online manual page.
- Be sure that a data change object (DCO) and a DCO log volume are associated with the volume for which you are creating the snapshot.
- Persistent FastResync must be enabled on the existing database volumes and disks must be assigned for the snapshot volumes.
  FastResync optimizes mirror resynchronization by tracking updates to stored data that have been missed by a mirror. When a snapshot mirror is reattached to its primary volumes, only the updates that were missed need to be re-applied to resynchronize it. FastResync increases the efficiency of the volume snapshot mechanism to better support operations such as backup and decision support.
  For detailed information about FastResync, see the *Storage Foundation Administrator's Guide*.
- Snapshot mirrors and their associated DCO logs should be on different disks than the original mirror plexes, and should be configured correctly for creating snapshots by the system administrator.
- When creating a snapshot mirror, create the snapshot on a separate controller and separate disks from the primary volume.
- Allocate separate volumes for archive logs.

Usage Notes

- Create a separate disk group for DB2 database-related files.
- Do not share volumes between DB2 database files and other software.
- Resynchronization speed varies based on the amount of data changed in both the primary and snapshot volumes during the break-off time.
- Do not share any disks between the original mirror and the snapshot mirror.
- Snapshot mirrors for datafiles and archive logs should be created so that they do not share any disks with the data of the original volumes. If they are not created in this way, the VxVM disk group cannot be split and, as a result, Database FlashSnap will not work.

Note: Database FlashSnap commands support third-mirror break-off snapshots only. The snapshot mirror must be in the SNAPDONE state.

The following sample procedure is for existing volumes without existing snapshot plexes or associated snapshot volumes. In this procedure, *volume_name* is the name of either a volume or a volume set.

Note: You must be logged in as superuser (root) to issue the commands in the following procedure.

To create a snapshot mirror of a volume or volume set

1 To prepare the volume for being snapshot, use the `vxsnap prepare` command:

```
# vxsnap -g diskgroup prepare volume \
alloc="storage_attribute ..."
```

The `vxsnap prepare` command automatically creates a DCO and DCO volumes and associates them with the volume, and enables Persistent FastResync on the volume. Persistent FastResync is also set automatically on any snapshots that are generated from a volume on which this feature is enabled.

For enabling persistent FastResync on a volume either from the command line or from within a script, use the `vxsnap prepare` command as described above.

2 To verify that FastResync is enabled on the volume, use the `vxprint` command:

```
# vxprint -g diskgroup -F%fastresync volume_name
```

This returns on if FastResync is on. Otherwise, it returns off.

3 To verify that a DCO and DCO log volume are attached to the volume, use the `vxprint` command:

```
# vxprint -g diskgroup -F%hasdcolog volume_name
```

This returns on if a DCO and DCO log volume are attached to the volume. Otherwise, it returns off.

4 Create a mirror of a volume:

```
# vxsnap -g diskgroup addmir volume_name alloc=diskname
```

Example of creating 3 mirrors for a particular volume:

```
# vxsnap -g diskgroup addmir datavol \
nmirror=3 alloc=disk1,disk2,disk3
```

5 List the available mirrors:

```
# vxprint -g diskgroup -F%name -e"pl_v_name in \"volume_name\""
```

6 Enable database FlashSnap to locate the correct mirror plexes when creating snapshots:

- Set the dbed_flashsnap tag for the data plex you want to use for breaking off the mirror. You can choose any tag name you like, but it needs to match the `SNAPSHOT_PLEX_TAG` attribute specified in the configuration or snapplan.

  ```
  # vxedit -g diskgroup set putil2=dbed_flashsnap plex_name
  ```

- Verify that the dbed_flashsnap tag has been set to the desired data plex:

  ```
  # vxprint -g diskgroup -F%name -e"pl_v_name in \
  \"volume_name\" && p2 in \"dbed_flashsnap\""
  ```

If you require a backup of the data in the snapshot, use an appropriate utility or operating system command to copy the contents of the snapshot to tape or to some other backup medium.

Creating a clone of a database by using Database FlashSnap

You can use Database FlashSnap to create a clone of a database by performing the steps outlined in Figure 13-2.

Figure 13-2 Creating a Clone - Workflow



See "vxsfadm command reference" on page 153.

See "FlashSnap configuration parameters" on page 156.

See "FlashSnap supported operations" on page 158.

For details, refer to `vxsfadm-flashsnap`(1M) and `vxsfadm-db2-flashsnap`(1M) man pages.

To create a clone of a DB2 database by using FlashSnap

1 Create a configuration file.

```
$ /opt/VRTS/bin/vxsfadm -s flashsnap \
-a db2 -o setdefaults --db2instance db2inst1 \
--db2database proddb --flashsnap_name daily_snap -c dailyconfig
Written config file dailyconfig
```

This command creates a default configuration file with all the parameters and default values. You can change the parameters, if required.

Note: If you have exported in environment the DB2INSTANCE and DB2DATABASE parameters, you do not need to include them on the command line. In the following steps, it is assumed that DB2INSTANCE and DB2DATABASE are available from the environment.

---

2 Validate the setup.

```
$ /opt/VRTS/bin/vxsfadm -s flashsnap \
-a db2 -o validate -c dailyconfig
Validating database configuration for third-mirror-break-off snapshot:
DB2INSTANCE = db2inst1
DB2DATABASE = proddb
APP_MODE = online
SNAPSHOT_ARCHIVE_LOG = auto
ARCHIVELOG_DEST = /db2arch/
Database validation successful.
Validating database volume layout for third-mirror-break-off snapshot:
Data volumes ready for snapshot:
Volume/volume-set db2datavol of diskgroup db2dg mounted on /db2data.
Archivelog volume ready for snapshot:
Volume/volume-set db2archvol of diskgroup db2dg mounted on /db2arch.
Storage units to be used for snapshot from diskgroup db2dg:
ds4100-0_9 ds4100-0_7
SNAPSHOT_VOL_PREFIX = SNAP_
SNAPSHOT_DG_PREFIX = SNAP_
Database volume layout validated successfully.
```

This command validates the configuration file and the database environment. In case of any problem, appropriate error messages are displayed that you can use to correct the problem and then retry.

3 Create a snapshot of the database.

```
$ /opt/VRTS/bin/vxsfadm -s flashsnap \
-a db2 -o snap -c dailyconfig
Validating database configuration for third-mirror-break-off snapshot:
DB2INSTANCE = db2inst1
DB2DATABASE = proddb
APP_MODE = online
SNAPSHOT_ARCHIVE_LOG = auto
ARCHIVELOG_DEST = /db2arch/
Database validation successful.
snapshot started at Tue Mar 20 00:39:41 2012.
Putting database in write-suspend mode... Done
Validating database volume layout for third-mirror-break-off snapshot:
Data volumes ready for snapshot:
Volume/volume-set db2datavol of diskgroup db2dg mounted on /db2data.
Archivelog volume ready for snapshot:
Volume/volume-set db2archvol of diskgroup db2dg mounted on /db2arch.
Storage units to be used for snapshot from diskgroup db2dg:
ds4100-0_9 ds4100-0_7
SNAPSHOT_VOL_PREFIX = SNAP_
SNAPSHOT_DG_PREFIX = SNAP_
Database volume layout validated successfully.
Creating snapshot volumes for data volumes ... Done
Taking database out of write-suspend mode... Done
Creating snapshot volume for archivelog volume ... Done
Copying snapshot information to snapshot volume ... Done
Creating snapshot diskgroups ... Done
Deporting snapshot diskgroups ... Done
SNAP_db2dg
snaphot ended at Tue Mar 20 00:40:23 2012.
```

This command breaks the user-specified mirror (parameter `SNAPSHOT_PLEX_TAG`) from the primary volumes and creates a new disk group with the name starting with the string defined in the `snap_dg_prefix` parameter. The default value of this parameter is SNAP_.

Note: At the validation stage, all the parameters, including the mandatory parameters `--db2instance`, `--db2database`, and `--flashsnap_name`, are read and stored in the repository.

If you need to change any parameter, change the configuration file and specify it with the `-c` option.

4 Mount the snapshot.

```
$ /opt/VRTS/bin/vxsfadm -s flashsnap \
-a db2 -o mount -c dailyconfig
Retrieving snapshot information ... Done
Importing snapshot diskgroups ... Done
Mounting snapshot volumes ... Done
```

Note: This command mounts the snapshot on the host running the DB2 instance. The secondary host is the system defined in the SECONDARY_HOST parameter of the configuration file.

By default, volumes are mounted under the `/var/tmp` file system.

If you need to specify an alternate location for mounting snapshot volumes, either provide CLONE_PATH on the command line or from the configuration file.

For performing off-host operations, specify the SFDB repository host using the `-r` option of the `vxsfadm` command.

Note: Ensure that the DB2 user has the required permissions to create the `/clonedb2` directory, if it does not exist.

---

5 Clone the database based on the snapshot.

```
$ /opt/VRTS/bin/vxsfadm -s flashsnap \
-a db2 -o clone -c dailyconfig
Retrieving snapshot information ... Done
Importing snapshot diskgroups ... Done
Mounting snapshot volumes ... Done
Relocating/ Renaming clone database clone1 ... Done
Initializing clone database clone1 ... Done
Activating clone database clone1 ... Done
```

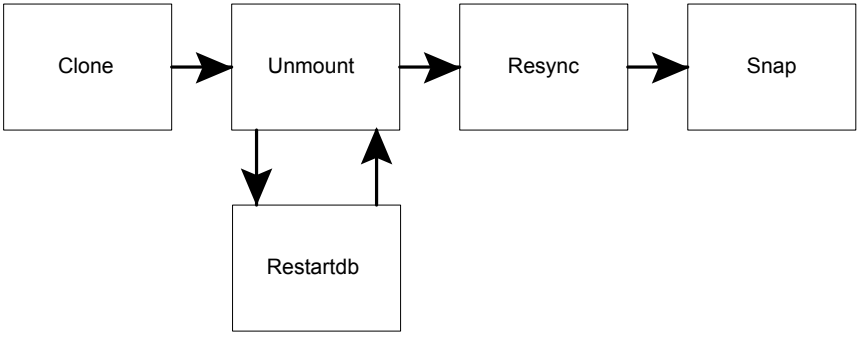
If you have not specified *clone_name*, it is automatically generated.

Note: If you have already specified the *clone_name* and the *clone_path* parameters in the configuration file that was used during the validate operation, the *clone_name* parameter is not required on the command line.

Resynchronizing mirror volumes with primary volumes

After creating a clone of your database, you can refresh mirror volumes with primary volumes by using the steps outlined in Figure 13-3.

Figure 13-3 Resynchronizing Mirror Volumes



To resynchronize mirror volumes with primary volumes

1 Unmount the clone database.

```
$ /opt/VRTS/bin/vxsfadm -s flashsnap \
-a db2 -o umount -c dailyconfig
Shutting down clone database... Done
Retrieving snapshot information ... Done
Unmounting snapshot volumes ... Done
Deporting snapshot diskgroups ... Done
```

This command stops the clone database gracefully and unmounts the file systems. The unmounted database can be restarted by using the clone operation.

You can use the `-o clone` option to restart a clone database after it is unmounted.

```
$ /opt/VRTS/bin/vxsfadm -s flashsnap \
-a db2 -o clone -c dailyconfig
Retrieving snapshot information ... Done
Importing snapshot diskgroups ... Done
Mounting snapshot volumes ... Done
Activating clone database clone1 ... Done
```

This command mounts the snapshot file systems and restarts the cloned database.

2 Refresh mirror volumes with primary volumes.

```
$ /opt/VRTS/bin/vxsfadm -s flashsnap \
-a db2 -o resync -c dailyconfig
resync started at Tue Mar 20 00:46:29 2012.
Importing snapshot diskgroups ... Done
Joining snapshot diskgroups to original diskgroups ... Done
Reattaching snapshot volumes to original volumes ... Done
resync ended at Tue Mar 20 00:46:57 2012.
```

This command resynchronizes all mirror volumes that were broken during snapshot state with primary volumes. After the resync operation is complete and mirror volumes are in the SNAPDONE state, you can take fresh snapshots.

You can use the `vxprint` command to view the state of the volumes.

Cloning a database on the secondary host

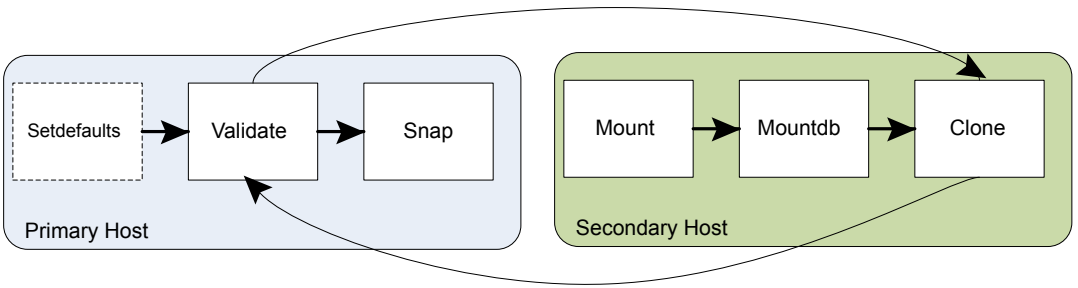
You can use the `vxsfadm` command to perform end operations without performing the intermediate operations. For example, you need to perform the steps outlined in Figure 13-4.

Figure 13-4 Creating a Clone - Basic Workflow



However, you can use `vxsfadm` to go directly from Validate to Clone for cloning a database on the secondary host. Figure 13-5 outlines the steps for doing this operation.

Figure 13-5 Creating a Clone - Without Intermediate Operations



To clone a database on the secondary host

- Enter the following command.

```
$ vxsfadm -s flashsnap \
-a db2 -o clone --db2instance db2inst1 \
--db2database proddb --flashsnap_name dailysnap \
--secondary_host host2 --app_mode online \
--clone_path /tmp/testclonepath
--clone_name clone1
Retrieving snapshot information ... Done
Importing snapshot diskgroups ... Done
Mounting snapshot volumes ... Done
Relocating/ Renaming clone database clone1 ... Done
Initializing clone database clone1 ... Done
Activating clone database clone1 ... Done
```

In a 2-host setup with the primary host host1 and the secondary host host2, this command creates a clone database on host2.

The database name of clone database is clone1 and it is mounted in the `/tmp/testclonepath` directory. The DB2INSTANCE and the uid of the db2insatnce user should be same on both the hosts. Default values are assigned to all of the other parameters.

Chapter 14

Administering Storage Checkpoints

This chapter includes the following topics:

- About Storage Checkpoints
- Database Storage Checkpoints for recovery
- Creating a Database Storage Checkpoint
- Deleting a Database Storage Checkpoint
- Mounting a Database Storage Checkpoint
- Unmounting a Database Storage Checkpoint
- Creating a database clone using a Database Storage Checkpoint
- Restoring database from a Database Storage Checkpoint
- Gathering data for offline-mode Database Storage Checkpoints

About Storage Checkpoints

Veritas File System (VxFS) provides a Storage Checkpoint feature that quickly creates a persistent image of a file system at an exact point in time. Storage Checkpoints significantly reduce I/O overhead by identifying and maintaining only the file system blocks that have changed since the last Storage Checkpoint or backup via a copy-on-write technique.

See "Copy-on-write" on page 98.

Storage Checkpoints provide:

- Persistence through reboots and crashes.

- The ability for data to be immediately writeable by preserving the file system metadata, the directory hierarchy, and user data.

Storage Checkpoints are actually data objects that are managed and controlled by the file system. You can create, remove, and rename Storage Checkpoints because they are data objects with associated names.

See "How a Storage Checkpoint works" on page 97.

Unlike a disk-based mirroring technology that requires a separate storage space, Storage Checkpoints minimize the use of disk space by using a Storage Checkpoint within the same free space available to the file system.

After you create a Storage Checkpoint of a mounted file system, you can also continue to create, remove, and update files on the file system without affecting the logical image of the Storage Checkpoint. A Storage Checkpoint preserves not only the name space (directory hierarchy) of the file system, but also the user data as it existed at the moment the file system image was captured.

You can use a Storage Checkpoint in many ways. For example, you can use them to:

- Create a stable image of the file system that can be backed up to tape.
- Provide a mounted, on-disk backup of the file system so that end users can restore their own files in the event of accidental deletion. This is especially useful in a home directory, engineering, or email environment.
- Create a copy of an application's binaries before installing a patch to allow for rollback in case of problems.
- Create an on-disk backup of the file system in that can be used in addition to a traditional tape-based backup to provide faster backup and restore capabilities.
- Test new software on a point-in-time image of the primary fileset without jeopardizing the live data in the current primary fileset by mounting the Storage Checkpoints as writable.

Database Storage Checkpoints for recovery

A Database Storage Checkpoint creates an exact image of a database instantly and provides a consistent image of the database from the point in time the Database Storage Checkpoint was created. The Database Storage Checkpoint image is managed and available through the command line interface (CLI).

Because each Database Storage Checkpoint is a consistent point-in-time image of a file system, Storage Rollback is the restore facility for these on-disk backups. Storage Rollback rolls back the changed blocks that are contained in a Database Storage Checkpoint into the primary file system for faster database restoration.

The combination of data redundancy (disk mirroring) and Database Storage Checkpoints is recommended for highly critical data to protect them from both physical media failure and logical errors.

Advantages and limitations of Database Storage Checkpoints

Database Storage Checkpoints and rollback provides the following advantages:

- Initially, a Database Storage Checkpoint contains no data—it contains only the inode list and the block map of the primary fileset. The block map points to the actual data on the primary file system.
- Because only the inode list and block map are needed and no data is copied, creating a Database Storage Checkpoint takes only a few seconds and very little space.
- A Database Storage Checkpoint keeps track of block change information and thereby enables incremental database backup at the block level.
- A Database Storage Checkpoint helps recover data from incorrectly modified files.
- A Database Storage Checkpoint can be mounted, allowing regular file system operations to be performed. Mounted Database Storage Checkpoints can be used for a wide range of application solutions that include backup, investigations into data integrity, staging upgrades or database modifications, and data replication solutions.

The limitations of Database Storage Checkpoints are as follows:

- Database Storage Checkpoints can only be used to restore from logical errors (for example, a human error).
- Because all the data blocks are on the same physical device, Database Storage Checkpoints cannot be used to restore files due to a media failure. A media failure requires a database restore from a tape backup or a copy of the database files that are kept on a separate medium.

Creating a Database Storage Checkpoint

You can use the Storage Checkpoint feature of Storage Foundation to create a Database Storage Checkpoint of a database by performing the following procedure.

See “vxsfadm command reference” on page 153.

See “Database Storage Checkpoints configuration parameters” on page 159.

See “Database Storage Checkpoints supported operations” on page 161.

For details, refer to `vxsfadm-checkpoint`(1M) and `vxsfadm-db2-checkpoint`(1M) man pages.

To create a Database Storage Checkpoint

- Use the following command.

```
$ /opt/VRTSdbed/bin/vxsfadm -s checkpoint -o create -a db2 \
--db2database proddb --checkpoint_name XYZ \
--app_mode online --removable
Putting database in backup mode... Done
Creating Storage Checkpoint XYZ ... Done
Storage Checkpoint XYZ created
Removing the database from write suspend... Done
```

This command creates a removable online Database Storage Checkpoint of the DB2 database of the specified name. If the *checkpoint_name* parameter is not specified, a name is automatically generated.

Deleting a Database Storage Checkpoint

You can delete a Database Storage Checkpoint as follows.

To delete a Database Storage Checkpoint

- Use the following command.

```
$ /opt/VRTSdbed/bin/vxsfadm -s checkpoint -o delete -a db2 \
--db2database proddb --checkpoint_name XYZ
Deleting the Checkpoint XYZ...
Deleting Checkpoint from /db2data... Done
```

This command deletes the Database Storage Checkpoint XYZ. If it is mounted, this command unmounts the Database Storage Checkpoint and then deletes it.

You can use the following command to delete a clone database created from a Database Storage Checkpoint.

```
$ /opt/VRTSdbed/bin/vxsfadm -s checkpoint -o delete -a db2 \
--db2instance db2inst1 --db2database proddb \
--checkpoint_name=XYZ --clone_name clone1
```

This command shuts down the clone database clone1 and then unmounts and deletes the Database Storage Checkpoint.

Mounting a Database Storage Checkpoint

You can mount a Database Storage Checkpoint as follows.

To mount a Database Storage Checkpoint

- Do one of the following.
  - Use the following command to mount the Database Storage Checkpoint in the read-only mode.

    ```
    $ /opt/VRTSdbed/bin/vxsfadm -s checkpoint -o mount -a db2 \
    --db2instance db2inst1 --db2database proddb \
    --checkpoint_name XYZ --clone_path /tmp/testclonepath

    Mounting Checkpoint to /tmp/testclonepath... Done
    ```

    This command mounts the Database Storage Checkpoint in the read-only mode to the specified mount point.
    If the *checkpoint_name* parameter is not specified, a name is automatically generated.
  - Use the following command to mount the Database Storage Checkpoint in the read-write mode.

    ```
    $ /opt/VRTSdbed/bin/vxsfadm -s checkpoint -o mountrw -a db2 \
    --db2instance db2inst1 --db2database proddb \
    --checkpoint_name XYZ --clone_path /tmp/testclonepath
    Creating Storage Checkpoint XYZ_rw_1332191432 ... Done
    Storage Checkpoint XYZ_rw_1332191432 created
    Mounting Checkpoint to /tmp/testclonepath... Done
    ```

    This command creates a Storage Checkpoint of the Database Storage Checkpoint XYZ and then mounts the newly created Database Storage Checkpoint to the specified mount point in the read-write mode.
    If the *checkpoint_name* parameter is not specified, a name is automatically generated.

    Note: This command maintains the point-in-time copy of the original Database Storage Checkpoint.

Unmounting a Database Storage Checkpoint

You can unmount a Database Storage Checkpoint as follows.

To unmount a Database Storage Checkpoint

- Use the following command.

```
$ /opt/VRTSdbed/bin/vxsfadm -s checkpoint -o umount -a db2 \
--db2instance db2inst1 --db2database proddb \
--checkpoint_name XYZ
Shutting down the clone database clone1 ... Done
Unmounting the checkpoint... Done
```

This command unmounts the Database Storage Checkpoint XYZ.

You can use the following command to unmount a clone database created from a Database Storage Checkpoint.

```
$ /opt/VRTSdbed/bin/vxsfadm -s checkpoint -o umount -a db2 \
--db2instance db2inst1 --db2database proddb \
--checkpoint_name XYZ --clone_name clone1
```

This command shuts down the clone database clone1 and then unmounts the Database Storage Checkpoint.

Creating a database clone using a Database Storage Checkpoint

You can create a database clone by using a Database Storage Checkpoint as follows.

To create a clone of a Database Storage Checkpoint

- Use the following command.

```
$ /opt/VRTSdbed/bin/vxsfadm -s checkpoint -o clone -a db2 \
--db2instance db2inst1 --db2database proddb \
--checkpoint_name XYZ --clone_name clone1 \
--clone_path /tmp/testclonepath
Creating Storage Checkpoint XYZ_rw_1334030056 ... Done
Storage Checkpoint XYZ_rw_1334030056 created
Mounting Checkpoint to /tmp/testclonepath... Done
Relocating/ Renaming clone database clone1 ... Done
Initializing clone database clone1 ... Done
Activating clone database clone1 ... Done
```

This command creates a Storage Checkpoint of the Database Storage Checkpoint XYZ and then mounts the newly created Database Storage Checkpoint to the specified mount point in the read-write mode and recovers the mounted Storage Checkpoint.

If you do not specify *checkpoint_name* and *clone_name*, they are automatically generated.

Note: This command maintains the point-in-time copy of the original Database Storage Checkpoint.

Restoring database from a Database Storage Checkpoint

You can restore your database from a Database Storage Checkpoint as follows.

Note: The following operation requires your primary database to be deactivated. The database needs to have a `logarchmeth` setup, because after recovery the database needs to be rollforwarded.

To restore the database from a Database Storage Checkpoint

- Use the following command.

```
$ /opt/VRTSdbed/bin/vxsfadm -s checkpoint -o restore -a db2 \
--db2instance db2inst1 --db2database proddb \
--checkpoint_name=XYZ
Rolling back the application files... Done
```

This command restores all the database to the point-in-time when the Checkpoint was created.

Note: Rollback of Database Storage Checkpoints that are mounted in the read-write mode is not supported.

After the restore operation, you can bring up the database by using standard recovery techniques.

For example:

```
$ db2initdb proddb as mirror
$ db2 rollforward db proddb to end of logs and stop
```

Note: The TEMP tablespaces cannot be rolled back.

Gathering data for offline-mode Database Storage Checkpoints

You can gather data for offline-mode Database Storage Checkpoints as follows.

Note: You can perform the following operation only when the database is up.

To gather information necessary to create a Storage Checkpoint when the database is offline

- Use the following command.

```
$ /opt/VRTS/bin/vxsfadm -s checkpoint -a db2 -o getappdata \
--db2instance dbinst1 --db2database proddb
Gathering offline data... Done
```

Note: If you attempt to create an offline Database Storage Checkpoint without performing the data gathering operation, the Checkpoint creation fails.

Chapter 15

Backing up and restoring with Netbackup in an SFHA environment

This chapter includes the following topics:

- About Veritas NetBackup
- About using Veritas NetBackup for backup and restore for DB2
- Using NetBackup in an SFHA Solutions product environment

About Veritas NetBackup

NetBackup provides backup, archive, and restore capabilities for database files and directories contained on client systems in a client-server network. NetBackup server software resides on platforms that manage physical backup storage devices. The NetBackup server provides robotic control, media management, error handling, scheduling, and a repository of all client backup images.

Administrators can set up schedules for automatic, unattended full and incremental backups. These backups are managed entirely by the NetBackup server. The administrator can also manually back up clients. Client users can perform backups, archives, and restores from their client system, and once started, these operations also run under the control of the NetBackup server.

NetBackup, while not a shipped component of Storage Foundation Enterprise products, can be purchased separately.

About using Veritas NetBackup for backup and restore for DB2

With Veritas NetBackup, you can perform high performance, online (hot) backups of databases that must be available on a 24x7 basis. NetBackup supports the Extended Edition (EE) and the Enterprise Extended Edition (EEE) environments. NetBackup also supports Database Partitioning Feature (DPF) for DB2 8.1 and higher.

Veritas NetBackup enables you to back up and restore database files and directories. You can set up schedules for automatic, unattended database backup, as well as full or incremental backup. These backups are managed entirely by the NetBackup server. You can also manually back up database files from any of the NetBackup clients. Client users can perform database backups and restores from their client systems on demand.

Veritas NetBackup can be configured for DB2 in an Extended Edition (EE), Extended-Enterprise Edition (EEE), or Database Partitioning Feature (DPF) environment. Two types of DB2 backup policies are required. One is used to backup the catalog nodes and the other is used to backup all the nodes, including the catalog node. Detailed information and instructions on configuring DB2 for EEE is available in the system administrator's guide.

See the *Veritas NetBackup for DB2 System Administrator's Guide for UNIX*.

Veritas NetBackup for DB2 has the following features:

- Media and device management
- Scheduling facilities
- Multiplexed backups and restores
- Transparent execution of both DB2 and regular file system backup and restore operations
- Shared devices and tapes used during other file backups
- Centralized and networked backup operations
- Parallel backup and restore operations
- Incremental backups of DB2 databases

Table 15-1 Options for backing up DB2 with NetBackup

	Automatically	Manually	`DB2 BACKUP DATABASE` command
DB2 database log backups	Supported	Supported	Supported
DB2 archive log backups	Supported	Supported	Supported
DB2 policy backups	Supported	Supported	

Setting up schedules for automatic backups is the most convenient way to back up your database.

See 'Performing a Backup' in the *Veritas NetBackup for DB2 System Administrator's Guide for UNIX*.

The procedure for restoring a DB2 database depends on the database involved and the problems that you have on your system. You can browse the backups using the `db2 list history` command or using the NetBackup `bplist` command before restoring.

See the *DB2 UDB Administration Guide Data Recovery and High Availability Guide*.

Using NetBackup in an SFHA Solutions product environment

You can enhance the ease of use and efficiency of your SFHA Solutions product and NetBackup by integrating them as follows:

- Clustering a NetBackup Master Server
- Backing up and recovering a VxVM volume using NetBackup

Clustering a NetBackup Master Server

To enable your NetBackup Master Server to be highly available in a cluster environment, use the following procedure.

To make a NetBackup Master Server, media, and processes highly available

1. Verify that your versions of NetBackup and Cluster Server are compatible. Detailed combination information is included in the NetBackup cluster compatibility list:
   - For NetBackup 7.x cluster compatibility:
     See https://www.veritas.com/support/en_US/article.TECH126902

- For NetBackup 6.x cluster compatibility:
  See https://www.veritas.com/support/en_US/article.TECH43619
- For NetBackup 5.x cluster compatibility:
  See https://www.veritas.com/support/en_US/article.TECH29272
- For more on NetBackup compatibility, see
  https://www.veritas.com/support/en_US/dpp.15145.html

2 The steps to cluster a Master Server are different for different versions of NetBackup. See the applicable NetBackup guide for directions.

https://sort.veritas.com

To verify the robustness of the VCS resources and NetBackup processes

1 Verify that you can online the Netbackup master.

2 Verify that you can offline the Netbackup master.

3 Verify that you can monitor all the NetBackup resources.

Backing up and recovering a VxVM volume using NetBackup

To enable NetBackup to backup objects on a VxVM volume, use the following procedure. This procedure enables an Instant Recovery (IR) using a VxVM volume.

To back up objects in a VxVM volume using NetBackup

1. Create a VxVM disk group with six disks. The number of disks may vary depending on the volume size, disk size, volume layout, and snapshot method.

   If the system this test is running on is a clustered system, create a shared disk group using the `-s` option.

   ```
   # vxdg -s init database_dg disk1 disk2 disk3 \
   disk4 disk5 disk6
   ```

2. Create a "mirror-striped" VxVM volume with a size of 10 Gbytes or the maximum size of the disk, whichever is larger.

   ```
   # vxassist -g database_dg make vol_name 10G \
   layout=mirror-stripe init=active
   # vxvol -g database_dg set fastresync=on vol_name
   # vxassist -g database_dg snapstart nmirror=1 vol_name
   ```

   Note: There are three types of snapshot: mirror, full-size instant, and space-optimized instant shanpshots. The example uses an Instant Recovery (IR) snapshot. For snapshot creation details:

   See pages 104-107 of the *NetBackup Snapshot Client Administrator's Guide* for 7.6.

   See https://www.veritas.com/support/en_US/article.DOC6459

   ---

3. Make the file system on the volume.
4. Mount a VxFS file system on the volume.

   If the VxVM volume is a clustered volume, mount the VxFS file system with the "-o cluster" option.
5. Fill up the VxFS file system up to the desired level. For example, you can fill to 95% full, or to whatever level is appropriate for your file system.
6. Store the `cksum`(1) for these files.
7. Un-mount the VxFS file system.
8. Enable the following Advanced Client option:
   - Perform Snapshot Backup.
   - Set **Advanced Snapshot Options** to **vxvm**.

- Enable **Retain snapshots for instant recovery**.

9 Back up the VxVM volume with the NetBackup policy.

See *NetBackup Snapshot Client Administrator's Guide* for 7.6.

See https://www.veritas.com/support/en_US/article.DOC6459

Recovering a VxVM volume using NetBackup

To enable NetBackup to recover objects on a VxVM volume, use the following procedure. This procedure performs an Instant Recovery (IR) using a VxVM volume.

To recover objects in a VxVM volume using NetBackup

1 Initialize the VxVM volume to zeros.

2 Recover the VxVM volume to the newly initialized VxVM volume.

3 Mount the VxFS file system on the empty VxVM volume.

4 Verify the cksum(1) values against the files recovered.

Section 6

Optimizing storage costs for DB2

- Chapter 16. Understanding storage tiering with SmartTier
- Chapter 17. SmartTier use cases for DB2

Chapter 16

Understanding storage tiering with SmartTier

This chapter includes the following topics:

- About SmartTier
- SmartTier in a High Availability (HA) environment

About SmartTier

SmartTier matches data storage with data usage requirements. After data matching, the data can then be relocated based upon data usage and other requirements determined by the storage or database administrator (DBA).

As more and more data is retained over a period of time, eventually, some of that data is needed less frequently. The data that is needed less frequently still requires a large amount of disk space. SmartTier enables the database administrator to manage data so that less frequently used data can be moved to slower, less expensive disks. This also permits the frequently accessed data to be stored on faster disks for quicker retrieval.

Tiered storage is the assignment of different types of data to different storage types to improve performance and reduce costs. With SmartTier, storage classes are used to designate which disks make up a particular tier. There are two common ways of defining storage classes:

- Performance, or storage, cost class: The most-used class consists of fast, expensive disks. When data is no longer needed on a regular basis, the data can be moved to a different class that is made up of slower, less expensive disks.
- Resilience class: Each class consists of non-mirrored volumes, mirrored volumes, and n-way mirrored volumes.

For example, a database is usually made up of data, an index, and logs. The data could be set up with a three-way mirror because data is critical. The index could be set up with a two-way mirror because the index is important, but can be recreated. The redo and archive logs are not required on a daily basis but are vital to database recovery and should also be mirrored.

SmartTier is a VxFS feature that enables you to allocate file storage space from different storage tiers according to rules you create. SmartTier provides a more flexible alternative compared to current approaches for tiered storage. Static storage tiering involves a manual one- time assignment of application files to a storage class, which is inflexible over a long term. Hierarchical Storage Management solutions typically require files to be migrated back into a file system name space before an application access request can be fulfilled, leading to latency and run-time overhead. In contrast, SmartTier allows organizations to:

- Optimize storage assets by dynamically moving a file to its optimal storage tier as the value of the file changes over time
- Automate the movement of data between storage tiers without changing the way users or applications access the files
- Migrate data automatically based on policies set up by administrators, eliminating operational requirements for tiered storage and downtime commonly associated with data movement

---

Note: SmartTier is the expanded and renamed feature previously known as Dynamic Storage Tiering (DST).

SmartTier policies control initial file location and the circumstances under which existing files are relocated. These policies cause the files to which they apply to be created and extended on specific subsets of a file systems's volume set, known as placement classes. The files are relocated to volumes in other placement classes when they meet specified naming, timing, access rate, and storage capacity-related conditions.

In addition to preset policies, you can manually move files to faster or slower storage with SmartTier, when necessary. You can also run reports that list active policies, display file activity, display volume usage, or show file statistics.

SmartTier leverages two key technologies included with Veritas InfoScale Storage Foundation Enterprise products: support for multi-volume file systems and automatic policy-based placement of files within the storage managed by a file system. A multi-volume file system occupies two or more virtual storage volumes and thereby enables a single file system to span across multiple, possibly heterogeneous, physical storage devices. For example the first volume could reside on EMC Symmetrix DMX spindles, and the second volume could reside on EMC CLARiiON

spindles. By presenting a single name space, multi-volumes are transparent to users and applications. This multi-volume file system remains aware of each volume's identity, making it possible to control the locations at which individual files are stored. When combined with the automatic policy-based placement of files, the multi-volume file system provides an ideal storage tiering facility, which moves data automatically without any downtime requirements for applications and users alike.

In a database environment, the access age rule can be applied to some files. However, some data files, for instance are updated every time they are accessed and hence access age rules cannot be used. SmartTier provides mechanisms to relocate portions of files as well as entire files to a secondary tier.

To use SmartTier, your storage must be managed using the following features:

- VxFS multi-volume file system
- VxVM volume set
- Volume tags
- SmartTier management at the file level
- SmartTier management at the sub-file level

About VxFS multi-volume file systems

Multi-volume file systems are file systems that occupy two or more virtual volumes. The collection of volumes is known as a volume set, and is made up of disks or disk array LUNs belonging to a single Veritas Volume Manager (VxVM) disk group. A multi-volume file system presents a single name space, making the existence of multiple volumes transparent to users and applications. Each volume retains a separate identity for administrative purposes, making it possible to control the locations to which individual files are directed.

This feature is available only on file systems meeting the following requirements:

- The minimum disk group version is 140.
- The minimum file system layout version is 7 for file level SmartTier.
- The minimum file system layout version is 8 for sub-file level SmartTier.

To convert your existing VxFS system to a VxFS multi-volume file system, you must convert a single volume to a volume set.

The VxFS volume administration utility (fsvoladm utility) can be used to administer VxFS volumes. The fsvoladm utility performs administrative tasks, such as adding, removing, resizing, encapsulating volumes, and setting, clearing, or querying flags on volumes in a specified Veritas File System.

See the `fsvoladm` (1M) manual page for additional information about using this utility.

About VxVM volume sets

Volume sets allow several volumes to be represented by a single logical object. Volume sets cannot be empty. All I/O from and to the underlying volumes is directed via the I/O interfaces of the volume set. The volume set feature supports the multi-volume enhancement to Veritas File System (VxFS). This feature allows file systems to make best use of the different performance and availability characteristics of the underlying volumes. For example, file system metadata could be stored on volumes with higher redundancy, and user data on volumes with better performance.

About volume tags

You make a VxVM volume part of a placement class by associating a volume tag with it. For file placement purposes, VxFS treats all of the volumes in a placement class as equivalent, and balances space allocation across them. A volume may have more than one tag associated with it. If a volume has multiple tags, the volume belongs to multiple placement classes and is subject to allocation and relocation policies that relate to any of the placement classes.

Warning: Multiple tagging should be used carefully.

A placement class is a SmartTier attribute of a given volume in a volume set of a multi-volume file system. This attribute is a character string, and is known as a volume tag.

SmartTier file management

SmartTier enables administrators of multi-volume VxFS file systems to manage the placement of files on individual volumes in a volume set by defining placement policies that control both initial file location and the circumstances under which existing files are relocated. These placement policies cause the files to which they apply to be created and extended on specific subsets of a file system's volume set, known as placement classes. The files are relocated to volumes in other placement classes when they meet the specified naming, timing, access rate, and storage capacity-related conditions.

File-based movement:

- The administrator can create a file allocation policy based on filename extension before new files are created, which will create the datafiles on the appropriate tier during database creation.
- The administrator can also create a file relocation policy for database files or any types of files, which would relocate files based on how frequently a file is used.

SmartTier sub-file object management

SmartTier enables administrators of multi-volume VxFS file systems to manage the placement of file objects as well as entire files on individual volumes.

Using sub-file based movement you can:

- Move a set of ranges of a specified set of files of a specified set of mounts to a desired set of tiers on command.
- Move segments of files using automation to:
  - Monitor a set of files for collecting I/O statistics
  - Periodically collect and persist the statistics, cluster-wide if applicable
  - Periodically enforce the ranges of the registered sets of files based on their relative frequency of access to a desired set of tiers
  - Track the historical movements of those ranges

SmartTier in a High Availability (HA) environment

Cluster Server does not provide a bundled agent for volume sets. If issues arise with volumes or volume sets, the issues can only be detected at the DiskGroup and Mount resource levels.

The DiskGroup agent brings online, takes offline, and monitors a Veritas Volume Manager (VxVM) disk group. This agent uses VxVM commands. When the value of the StartVolumes and StopVolumes attributes are both 1, the DiskGroup agent onlines and offlines the volumes during the import and deport operations of the disk group. When using volume sets, set StartVolumes and StopVolumes attributes of the DiskGroup resource that contains the volume are set to 1. If a file system is created on the volume set, use a Mount resource to mount the volume set.

The Mount agent brings online, takes offline, and monitors a file system or NFS client mount point.

For additional information, see the *Cluster Server Bundled Agents Reference Guide*.

Chapter 17

SmartTier use cases for DB2

This chapter includes the following topics:

- SmartTier use cases for DB2
- Relocating old archive logs to tier two storage using SmartTier
- Relocating inactive tablespaces or segments to tier two storage
- Relocating active indexes to premium storage
- Relocating all indexes to premium storage

SmartTier use cases for DB2

Veritas InfoScale products include SmartTier, a storage tiering feature which enables you to tier your data to achieve optimal use of your storage.

Example procedures illustrate the following use cases:

- Relocating archive logs older than 2 days to Tier-2 storage
- Relocating inactive tablespaces or segments to Tier-2 storage
- Relocating active indexes to Tier-0 storage
- Relocating all indexes to Tier-0 storage

Relocating old archive logs to tier two storage using SmartTier

A busy database can generate few hundred gigabytes of archivelogs per day. Restoring these archive logs from tape backup is not ideal because it increases database recovery time. Regulatory requirements could mandate that these archive logs be preserved for several weeks.

To save storage costs, you can relocate archive logs older than two days (for example) into tier two storage. To achieve this you must create a policy file, for example, archive_policy.xml.

Note: The relocating archive logs use case applies for DB2 environments.

To relocate archive logs that are more than two days old to Tier-2

1 Create a policy file. A sample XML policy file is provided below.

```
<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM "/opt/VRTSvxfs/etc\
   /placement_policy.dtd">
<PLACEMENT_POLICY Version="5.0" Name="access_age_based">
    <RULE Flags="data" Name="Key-Files-Rule">
         <COMMENT>
            This rule deals with key files such as archive logs.
         </COMMENT>

    <SELECT Flags="Data">
         <COMMENT>
             You want all files. So choose pattern as '*'
         </COMMENT>
         <PATTERN>  *  </PATTERN>
     </SELECT>

    <CREATE>
        <ON>
            <DESTINATION>
                <CLASS> tier1 </CLASS>
            </DESTINATION>
        </ON>
    </CREATE>

    <RELOCATE>
    <TO>
          <DESTINATION>
              <CLASS> tier2 </CLASS>
          </DESTINATION>
    </TO>
          <WHEN>
                  <ACCAGE Units="days">
                      <MIN Flags="gt">2</MIN>
                  </ACCAGE>
          </WHEN>
    </RELOCATE>

    </RULE>
</PLACEMENT_POLICY>
```

Notice the `ACCAGE` units in the `WHEN` clause.

2 To locate additional sample policy files, go to `/opt/VRTSvxfs/etc`.

The access age-based policy is appropriate for this use case. Pay attention to the `CREATE ON` and `RELOCATE TO` sections of the XML file.

To apply a policy file

1 As root, validate *archive_policy.xml*

```
# fsppadm validate /DBarch archive_policy.xml
```

2 If the validation process is not successful, correct the problem. Validate *archive_policy.xml* successfully before proceeding.

3 Assign the policy to /DBarch filesystem

```
# fsppadm assign /DBarch archive_policy.xml
```

4 Enforce the policy. The relocation of two day old archive logs happens when the enforcement step is performed. The policy enforcements must be done every day to relocate aged archive logs. This enforcement can be performed on demand as needed or by using a cron- like scheduler.

```
# fsppadm enforce /DBarch
```

Relocating inactive tablespaces or segments to tier two storage

It is general practice to use partitions in databases. Each partition maps to a unique tablespace. For example in a shopping goods database, the orders table can be portioned into orders of each quarter. Q1 orders can be organized into *Q1_order_tbs tablespace*, Q2 order can be organized into *Q2_order_tbs*.

As the quarters go by, the activity on older quarter data decreases. By relocating old quarter data into Tier-2, significant storage costs can be saved. The relocation of data can be done when the database is online.

For the following example use case, the steps illustrate how to relocate Q1 order data into Tier-2 in the beginning of Q3. The example steps assume that all the database data is in the */DBdata* filesystem.

To prepare to relocate Q1 order data into Tier-2 storage for DB2

1 Obtain a list of containers belonging to *Q1_order_tbs*.

```
$ db2inst1$  db2 list  tablespaces
```

2 Find the tablespace-id for the tablespace *Q1_order_tbs*.

```
$ db2inst1$ db2  list  tablespace containers for <tablespace-id>
```

3 Find the path names for the containers and store them in file *Q1_order_files.txt*.

```
#cat  Q1_order_files.txt
          NODE0000/Q1_order_file1.f
          NODE0000/Q1_order_file2.f
          ...
          NODE0000/Q1_order_fileN.f
```

To relocate Q1 order data into Tier-2

1 Prepare a policy XML file. For the example, the policy file name is *Q1_order_policy.xml*. Below is a sample policy.

This is policy is for unconditional relocation and hence there is no `WHEN` clause. There are multiple `PATTERN` statements as part of the `SELECT` clause. Each `PATTERN` selects a different file.

```
<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM "/opt/VRTSvxfs/etc/\
placement_policy.dtd">
<PLACEMENT_POLICY Version="5.0" Name="selected files">
    <RULE Flags="data" Name="Key-Files-Rule">
        <COMMENT>
            This rule deals with key important files.
        </COMMENT>

        <SELECT Flags="Data">
            <DIRECTORY Flags="nonrecursive" > NODE0000</DIRECTORY>
            <PATTERN> Q1_order_file1.f  </PATTERN>
            <PATTERN> Q1_order_file2.f  </PATTERN>
            <PATTERN> Q1_order_fileN.f  </PATTERN>
        </SELECT>

        <RELOCATE>
             <COMMENT>
                Note that there is no WHEN clause.
             </COMMENT>
             <TO>
                <DESTINATION>
                    <CLASS> tier2 </CLASS>
                </DESTINATION>
             </TO>
         </RELOCATE>

    </RULE>
</PLACEMENT_POLICY>
```

2 Validate the policy *Q1_order_policy.xml*.

```
# fsppadm validate /DBdata Q1_order_policy.xml
```

3 Assign the policy.

```
# fsppadm assign /DBdata Q1_order_policy.xml
```

4 Enforce the policy.

```
# fsppadm enforce /DBdata
```

Relocating active indexes to premium storage

The database transaction rate depends upon how fast indexes can be accessed. If Indexes reside on slow storage, the database transaction rate suffers. Tier-0 storage is generally too expensive to be practical to relocate the entire table data to Tier-0. Indexes are generally much smaller in size and are created to improve the database transaction rate, hence it is more practical to relocate active indexes to Tier-0 storage. Using SmartTier you can move active indexes to Tier-0 storage.

For the following telephone company database example procedure, assume the *call_details* table has an index *call_idx* on the column *customer_id*.

To prepare to relocate *call_idx* to Tier-0 storage for DB2

1 Find the tablespace where *call_idx* resides.

```
$ db2inst1$  db2 connect to PROD
$ db2inst1$  db2 select index_tbspace from syscat.tables \
where tabname='call_details'
```

2 In this example, the index is in tablespace *tbs_call_idx*. To get the tablespace id for *tbs_call_idx* and the list of containers:

```
$ db2inst1$  db2  list tablespaces
```

Note the tablespace id for *tbs_call_idx*.

3 List the containers and record the filenames in the tabelspace *tbs_call_idx*.

```
$ db2inst1$  db2 list tablespace containers for <tablespace-id>
```

4 Store the files in *index_files.txt*.

```
# cat  index_files.txt
/DB2data/NODE0000/IDX/call1.idx
/DB2data/NODE0000/IDX/call2.idx
/DB2data/NODE0000/IDX/call3.idx
```

To relocate *call_idx* to Tier-0 storage

1 Prepare the policy *index_policy.xml*.

Example policy:

```
<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM "/opt/VRTSvxfs/etc/\
placement_policy.dtd">
<PLACEMENT_POLICY Version="5.0" Name="selected files">
    <RULE Flags="data" Name="Key-Files-Rule">
        <COMMENT>
            This rule deals with key important files.
        </COMMENT>

        <SELECT Flags="Data">
            <DIRECTORY Flags="nonrecursive" > NODE0000</DIRECTORY>
            <PATTERN> call*.idx  </PATTERN>
        </SELECT>

        <RELOCATE>
            <COMMENT>
                Note that there is no WHEN clause.
         </COMMENT>
         <TO>
            <DESTINATION>
                <CLASS> tier0 </CLASS>
            </DESTINATION>
         </TO>
         </RELOCATE>

    </RULE>
</PLACEMENT_POLICY>
```

2 Assign and enforce the policy.

```
# fsppadm validate /DBdata index_policy.xml
# fsppadm assign /DBdata index_policy.xml
# fsppadm enforce /DBdata
```

Relocating all indexes to premium storage

It is a common practice for DBAs to name index files with some common extensions. For example, all index files are named with “.inx” extensions. If your Tier-0 storage

has enough capacity, you can relocate all indexes of the database to Tier-0 storage. You can also make sure all index containers created with this special extension are automatically created on Tier-0 storage by using the `CREATE` and `RELOCATE` clause of policy definition.

To relocate all indexes to Tier-0 storage

1 Create a policy such as the following example:

```
# cat index_policy.xml

<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM "/opt/VRTSvxfs/etc/\
placement_policy.dtd">
<PLACEMENT_POLICY Version="5.0" Name="selected files">
    <RULE Flags="data" Name="Key-Files-Rule">
        <COMMENT>
            This rule deals with key important files.
        </COMMENT>

        <SELECT Flags="Data">
            <PATTERN> *.inx  </PATTERN>
        </SELECT>

        <CREATE>
            <COMMENT>
                Note that there  are two DESTINATION.
            </COMMENT>
            <ON>
                <DESTINATION>
                    <CLASS> tier0 </CLASS>
                </DESTINATION>
                <DESTINATION>
                    <CLASS> tier1</CLASS>
                </DESTINATION>
            </ON>
        </CREATE>

         <RELOCATE>
             <COMMENT>
                 Note that there is no WHEN clause.
             </COMMENT>
             <TO>
                 <DESTINATION>
                     <CLASS> tier0 </CLASS>
                 </DESTINATION>
             </TO>
         </RELOCATE>

    </RULE>
</PLACEMENT_POLICY>
```

2 To make sure file creation succeeds even if Tier-0 runs out of space, add two ON clauses as in the example policy in 1.

3 Assign and enforce the policy.

```
# fsppadm validate /DBdata index_policy.xml
# fsppadm assign /DBdata index_policy.xml
# fsppadm enforce /DBdata
```

Section 7

Storage Foundation for Databases administrative reference

- Chapter 18. Storage Foundation for Databases command reference
- Chapter 19. Tuning for Storage Foundation for Databases
- Chapter 20. Troubleshooting SFDB tools

Chapter 18

Storage Foundation for Databases command reference

This chapter includes the following topics:

- vxsfadm command reference
- FlashSnap reference
- Database Storage Checkpoints reference

vxsfadm command reference

`vxsfadm` is a command line utility that can be used to perform point-in-time copy operations such as FlashSnap and Database Storage Checkpoints for DB2 databases. This utility uses the underlying features of Storage Foundation to perform these operations. The vxsfadm command can be run by DB2 instance owner only.

Note: SFDB tools do not support taking point-in-time copies while structural changes to the database are in progress, such as adding or dropping tablespaces and adding or dropping data files. However, once a point-in-time copy is taken, a clone can be created at any time, regardless of the status of the database.

The `vxsfadm` utility is supported in multiple database configurations including:

- DB2 single-partition database in a standalone setup
- DB2 single-partition database with off-host
- DB2 single-partition database in a highly available setup (VCS cluster)

- DB2 single-partition database with Storage Foundation Cluster File System High Availability

The syntax of the `vxsfadm` command is as follows:

```
vxsfadm -s <service_type> -a <application-name> -o <operation>
[ -c config-file ]
[ -r repository-host ]
[ service-specific parameters ]
[ application-specific parameters ]

vxsfadm -s <service-type> -a application-name> -o list
[ service-specific parameters ]
[ application-specific parameters ]

vxsfadm -s <service-type> -a application-name> -o setdefaults
[ service-specific parameters ]
[ application-specific parameters ]
```

The following are the `vxsfadm` parameters.

`-s service_type`

Specifies the service type for the point-in-time copy operation.

Supported service types are:

- flashsnap: Volume-level snapshots
- checkpoint: File system checkpoints

`-o operation`

Specifies the point-in-time copy operation that is being performed. The supported operations vary for the different service types. For more information on the operations supported for a service type, refer to the respective `vxsfadm` man page.

The following operations are common to all service types:

- `-o list`: Lists all the configurations/services for the specified application. If *service_type* is specified then all existing configurations for the specified *service_type* are displayed.
- `-o setdefaults`: Generates the default configuration that can be saved to a file and then it can be used for the remainder of the operations.

`-a application_name`	Specifies the application for which the point-in-time-copy operation is being performed.
`-c config_file`	All the command line options specific to applications apart from service_type, application_name, and operation can be provided in the configuration file. The information in the configuration file should be in the format PARAMETER_NAME=VALUE. For information about the parameters supported for a service, refer to the respective `vxsfadm` man page.
`-r repository-host`	Specifies the host of the SFDB repository for off-host operations.
`service-specific parameters`	These parameters are required by a service for processing. The most important service-specific parameter is *--service*_name. For example, if you want to create a third-mirror break-off snapshot (flashsnap), the service_name parameter is flashsnap_name and you can specify a unique identifier for this parameter as follows: `--flashsnap_name=dailysnap`. This identifier can be used for all operations performed on that copy. These parameters can be provided in the configuration file or they can be exported in the environment. For more information, refer to the `vxsfadm` man pages.

`application-specific parameters`	These parameters are required by application for processing. The most important application-specific parameter is the one that uniquely identifies an application instance. For example, if there are two DB2 databases running on a system, db2instance and db2database parameters uniquely identify the application instance. These parameter are used for all the point-in-time copy operations for the specific application instance. These parameters can be provided in the configuration file or they can be exported in the environment. For more information, refer to the `vxsfadm` man pages.

Note: The `vxsfadm` parameters specified on the command line override the parameters specified in the configuration file and the parameters exported in the environment. The parameters specified in the configuration file override the parameters exported in the environment.

You can use `vxsfadm` logs and error messages for troubleshooting.

See "Resources for troubleshooting SFDB tools" on page 187.

For more information, refer to the following man pages:

```
vxsfadm-flashsnap(1M), vxsfadm-checkpoint(1M)

vxsfadm-db2-flashsnap(1M), vxsfadm-db2-checkpoint(1M)
```

FlashSnap reference

This section describes FlashSnap configuration parameters and supported operations.

FlashSnap configuration parameters

Table 18-1 lists the FlashSnap configuration parameters that can be provided in the configuration file. If you want to specify the parameter on the command line, refer to the Command Line Option column of the table.

Table 18-1 FlashSnap - Configuration Parameters

Parameter	Description	Accepted Values	Command Line Option
VERSION	The version of the configuration file format.	n.n Example: 6.0	NA
FLASHSNAP_NAME (*)	A unique identifier of the FlashSnap configuration.	String Example: snap1	`--flashsnap_name snap1` OR `--name snap1`
DB2INSTANCE (*)	The DB2 instance name.	String Example: db2inst1	`--db2instance db2inst1` OR `-I db2inst1`
DB2DATABASE (*)	The DB2 database name.	String Example: proddb	`--db2database proddb` OR `-D proddb`
APP_MODE	The mode of the application when the snapshot operation is being performed.	offline online instant	`--app_mode offline` OR `online` OR `instant`
SNAPSHOT_PLEX_TAG	The value of the putil2 attribute tag for the plexes that must be a part of the snapshot.	String Example: dbed_snap1	`--snapshot_plex_tag dbed_snap1` OR `--plex dbed_snap1`
SNAPSHOT_VOL_PREFIX	The string prefixed to volume names to create snapshot volume names.	String Example: SNAPVOL_	`--snapshot_vol_prefix SNAPVOL_`
SNAPSHOT_DG_PREFIX	The string prefixed to disk group names to create snapshot disk group names.	String Example: SNAPDG_	`--snapshot_dg_prefix SNAPDG_`
SNAPSHOT_MIRROR	The number of mirrors that need to broken off to form the snapshot volumes.	Number Default: 1	`--snapshot_mirror 2` OR `--n 2`
SNAPSHOT_ARCHIVE_LOG	If this parameter is set, the snapshot operation is also performed on the archive log volumes.	Yes No auto (**) Default: auto	`--snapshot_archive_log y` OR `--no_snapshot_archive_log`

Table 18-1 FlashSnap - Configuration Parameters *(continued)*

Parameter	Description	Accepted Values	Command Line Option
SECONDARY_HOST	The host on which the snapshot can be mounted and the application can be cloned.	Host name	`--secondary_host sys4`

MAPPED_MOUNTS	The volume-to-mountpoint mapping that specifies the paths where the snapshot volumes should be mounted.	dg1:volume1=path1;
		dg2:volume2=path2
		Example:
		mydg:datavol=/tmp/fsnp;
			`--mapped_mounts mydg:datavol=/tmp/fsnp`

CLONE_PATH	The file system path under which the the clone application instance must be mounted.	Valid path `/tmp/so1`	`--clone_path /tmp/so1` OR `mountpath /tmp/so1`
CLONE_NAME	The name of the clone DB2 database that being created in the clone operation.	String Example: clone1	`--clone_name clone1`

Note: (*) denotes that the parameter is mandatory.

Note: (**) If the SNAPSHOT_ARCHIVE_LOG parameter is set to auto, the snapshot operation is performed on the archive logs depending on whether log archiving is enabled or not. If log archiving is not enabled, the snapshot operation is not performed on archive logs. If log archiving is enabled, and if at least one of the archive log destinations, specified by the logarchmeth1 and logarchmeth2 parameters, is set to a local "DISK:" destination, then the snapshot operation is performed on archive logs.

FlashSnap supported operations

Table 18-2 lists the FlashSnap operations that you can perform by using the `vxsfadm` command.

Table 18-2 FlashSnap Supported Operations

Operation	Operation alias	Description
setdefaults	None	Generates the default configuration file for an application.
valid	validate	Validates the configuration file and the environment.
snap	snapshot, break	Takes a snapshot by breaking the mirrors, and splits the disk groups.
clone	clonedb	Creates an application clone. You can also use the clone option to restart a clone if it is unmounted.
mount	snapmount	Mounts the mirror volumes.
unmount	umount, snapunmount, snapumount	Unmounts the snapshots and if a clone is created, shuts down the clone.
destroy	delete, remove	Destroys the FlashSnap configuration from the repository.
list	None	Lists the available snapshot configurations.

Database Storage Checkpoints reference

This section describes FlashSnap configuration parameters and supported operations.

Database Storage Checkpoints configuration parameters

Table 18-3 lists the Database Storage Checkpoints configuration parameters that can be provided in the configuration file. If you want to specify the parameter on the command line, refer to the Command Line Option column of the table.

Table 18-3 Database Storage Checkpoints - Configuration Parameters

Parameter	Description	Accepted Values	Command Line Option
VERSION	The version of the configuration file format.	n.n Example: 6.0	NA
CHECKPOINT_NAME (*)	A unique identifier of the storage checkpoint.	String Example: ckpt1	`--checkpoint_name ckpt1`
DB2INSTANCE (*)	The DB2 instance name.	String Example: db2inst1	`--db2instance db2inst1` OR `-I db2inst1`
DB2DATABASE (*)	The DB2 database name.	String Example: /proddb	`--db2database proddb` OR `-D proddb`
APP_MODE	The mode of the application when the checkpoint operation is being performed.	offline online instant Default: online	`--app_mode offline` OR `online` OR `instant`
CLONE_NAME	The name of the application instance that is created during the clone operation.	String Example: clone1	`--clone_name clone1`
CLONE_PATH	The path to which the clone files are mounted.	Valid path `/tmp/ckpt1`	`--clone_path /tmp/ckpt1`
SNAPSHOT_REMOVABLE	A parameter to specify whether a removable or a non-removable storage checkpoint is being created.	Y or N Default: removable Y	`--snapshot_removable` OR `--no_snapshot_removable`

Note: (*) denotes that the parameter is mandatory.

Database Storage Checkpoints supported operations

Table 18-4 lists the Database Storage Checkpoints operations that you can perform by using the `vxsfadm` command.

Table 18-4 Database Storage Checkpoints Supported Operations

Operation	Operation alias	Description
setdefaults	None	Generates the default configuration file for an application.
ckpt	create, checkpoint, fsckpt, snap, snapshot	Creates a Database Storage Checkpoint for the application.
mount	snapmount, ckptmount	Mounts the Database Storage Checkpoint in the read-only mode or the read-write mode to the specified mount point.
mountrw	snapmountrw, ckptmountrw	Mounts the Database Storage Checkpoint in the read-write mode (a shadow checkpoint of the original storage checkpoint is created and it is mounted as read_write).
unmount/umount	ckptumount, unmount, snapunmount, snapumount	Unmounts the Database Storage Checkpoint.
delete	destroy, remove, ckptremove	Removes the Database Storage Checkpoint.
clone	ckptclone, clonedb	Creates a clone of the Database Storage Checkpoint.
restore	None	Restores the data files from the Database Storage Checkpoint.
list	None	Lists the Database Storage Checkpoints.

Table 18-4 Database Storage Checkpoints Supported Operations *(continued)*

Operation	Operation alias	Description
getappdata	None	Gathers database information when the database is online. This information is used for offline checkpoint processing.

Chapter 19

Tuning for Storage Foundation for Databases

This chapter includes the following topics:

- Additional documentation
- About tuning Veritas Volume Manager (VxVM)
- About tuning VxFS
- About tuning DB2 databases
- About tuning AIX Virtual Memory Manager

Additional documentation

Use the tuning tips and information provided in this chapter in conjunction with other more in-depth publications, such as:

- *Database Performance on AIX in DB2 UDB and Oracle Environments* (IBM Corporation)
- *IBM Configuration and Performance RedBooks* (IBM Corporation)
- *DB2 UDB V8.2 Performance Tuning Guide* (IBM Corporation)
- *DB2 High Performance Design and Tuning* (Prentice Hall)
- *Storage Foundation Administrator's Guide*, chapter on “VxVM Performance Monitoring”

About tuning Veritas Volume Manager (VxVM)

Veritas Volume Manager (VxVM) is tuned for most configurations ranging from small systems to larger servers. On smaller systems with less than a hundred drives, tuning should not be necessary and Veritas Volume Manager should be capable of adopting reasonable defaults for all configuration parameters. On very large systems, however, there may be configurations that require additional tuning of these parameters, both for capacity and performance reasons.

Various mechanisms exist for tuning VxVM. Many parameters can be tuned using AIX's System Management Interface Tool (SMIT). Other values can only be tuned using the command line interface for VxVM.

For more information on tuning VxVM, see the *Storage Foundation Administrator's Guide*.

About obtaining volume I/O statistics

If your database is created on a single file system that is on a single volume, there is typically no need to monitor the volume I/O statistics. If your database is created on multiple file systems on multiple volumes, or the volume configurations have changed over time, it may be necessary to monitor the volume I/O statistics for the databases.

Use the `vxstat` command to access information about activity on volumes, plexes, subdisks, and disks under VxVM control, and to print summary statistics to the standard output. These statistics represent VxVM activity from the time the system initially booted or from the last time the counters were reset to zero. If no VxVM object name is specified, statistics from all volumes in the configuration database are reported. Use the `-g` option to specify the database disk group to report statistics for objects in that database disk group.

VxVM records the following I/O statistics:

- count of operations
- number of blocks transferred (one operation can involve more than one block)
- average operation time (which reflects the total time through the VxVM interface and is not suitable for comparison against other statistics programs)

VxVM records the preceding three pieces of information for logical I/Os, including reads, writes, atomic copies, verified reads, verified writes, plex reads, and plex writes for each volume. VxVM also maintains other statistical data such as read failures, write failures, corrected read failures, corrected write failures, and so on. In addition to displaying volume statistics, the `vxstat` command is capable of

displaying more detailed statistics on the components that form the volume. For detailed information on available options, refer to the `vxstat`(1M) manual page.

To reset the statistics information to zero, use the `-r` option. You can reset the statistics information for all objects or for only those objects that are specified. Resetting just prior to an operation makes it possible to measure the impact of that particular operation.

The following is an example of output produced using the `vxstat` command:

```
   OPERATIONS           BLOCKS           AVG TIME(ms)

TYP  NAME         READ      WRITE      READ       WRITE     READ      WRITE

vol  log2            0       6312         0       79836       .0        0.2

vol  db02      2892318    3399730   0283759     7852514     20.6       25.5
```

Additional information is available on how to use the `vxstat` output to identify volumes that have excessive activity and how to reorganize, change to a different layout, or move these volumes.

Additional volume statistics are available for RAID-5 configurations.

See the `vxstat`(1M) manual page.

See the “Performance Monitoring” section of the “Performance Monitoring and Tuning” chapter in the *Storage Foundation Administrator's Guide*.

About tuning VxFS

Veritas File System provides a set of tuning options to optimize file system performance for different application workloads. VxFS provides a set of tunable I/O parameters that control some of its behavior. These I/O parameters help the file system adjust to striped or RAID-5 volumes that could yield performance far superior to a single disk. Typically, data streaming applications that access large files see the largest benefit from tuning the file system.

Most of these tuning options have little or no impact on database performance when using Quick I/O. However, you can gather file system performance data when using Quick I/O, and use this information to adjust the system configuration to make the most efficient use of system resources.

How monitoring free space works

In general, VxFS works best if the percentage of free space in the file system is greater than 10 percent. This is because file systems with 10 percent or more of

free space have less fragmentation and better extent allocation. Regular use of the `df` command to monitor free space is desirable. Full file systems may have an adverse effect on file system performance. Full file systems should therefore have some files removed or should be expanded.

See the `fsadm_vxfs`(1M) manual page.

About monitoring fragmentation

Fragmentation reduces performance and availability. Regular use of `fsadm`'s fragmentation reporting and reorganization facilities is therefore advisable.

The easiest way to ensure that fragmentation does not become a problem is to schedule regular defragmentation runs using the `cron` command.

Defragmentation scheduling should range from weekly (for frequently used file systems) to monthly (for infrequently used file systems). Extent fragmentation should be monitored with `fsadm`command.

There are three factors that can be used to determine the degree of fragmentation:

- Percentage of free space in extents that are less than eight blocks in length
- Percentage of free space in extents that are less than 64 blocks in length
- Percentage of free space in extents that are 64 or more blocks in length

An unfragmented file system will have the following characteristics:

- Less than 1 percent of free space in extents that are less than eight blocks in length
- Less than 5 percent of free space in extents that are less than 64 blocks in length
- More than 5 percent of the total file system size available as free extents that are 64 or more blocks in length

A badly fragmented file system will have one or more of the following characteristics:

- More than 5 percent of free space in extents that are less than 8 blocks in length
- More than 50 percent of free space in extents that are less than 64 blocks in length
- Less than 5 percent of the total file system size available as free extents that are 64 or more blocks in length

The optimal period for scheduling extent reorganization runs can be determined by choosing a reasonable interval, scheduling `fsadm` runs at the initial interval, and running the extent fragmentation report feature of `fsadm` before and after the reorganization.

The “before” result is the degree of fragmentation prior to the reorganization. If the degree of fragmentation approaches the percentages for bad fragmentation, reduce the interval between `fsadm`. If the degree of fragmentation is low, increase the interval between `fsadm` runs.

How tuning VxFS I/O parameters works

VxFS provides a set of tunable I/O parameters that control some of its behavior. These I/O parameters are useful to help the file system adjust to striped or RAID-5 volumes that could yield performance far superior to a single disk. Typically, data streaming applications that access large files see the biggest benefit from tuning the file system.

If VxFS is being used with Veritas Volume Manager, the file system queries VxVM to determine the geometry of the underlying volume and automatically sets the I/O parameters. VxVM is queried by `mkfs` when the file system is created to automatically align the file system to the volume geometry. If the default alignment from `mkfs` is not acceptable, the `-o align=n` option can be used to override alignment information obtained from VxVM. The `mount` command also queries VxVM when the file system is mounted and downloads the I/O parameters.

If the default parameters are not acceptable or the file system is being used without VxVM, then the `/etc/vx/tunefstab` file can be used to set values for I/O parameters. The `mount` command reads the `/etc/vx/tunefstab` file and downloads any parameters specified for a file system. The `tunefstab` file overrides any values obtained from VxVM. While the file system is mounted, any I/O parameters can be changed using the `vxtunefs` command, which can have tunables specified on the command line or can read them from the `/etc/vx/tunefstab` file.

The `vxtunefs` command can be used to print the current values of the I/O parameters.

See the `vxtunefs`(1M) and `tunefstab`(4) manual pages.

About tunable VxFS I/O parameters

The following are tunable VxFS I/O parameters:

`read_pref_io` — The preferred read request size. The file system uses this parameter in conjunction with the `read_nstream` value to determine how much data to read ahead. The default value is 64K.

`write_pref_io`	The preferred write request size. The file system uses this parameter in conjunction with the `write_nstream` value to determine how to do flush behind on writes. The default value is 64K.
`read_nstream`	The number of parallel read requests of size `read_pref_io` that you can have outstanding at one time. The file system uses the product of `read_nstream` multiplied by `read_pref_io` to determine its read ahead size. The default value for `read_nstream` is 1.
`write_nstream`	The number of parallel write requests of size `write_pref_io` that you can have outstanding at one time. The file system uses the product of `write_nstream` multiplied by `write_pref_io` to determine when to do flush behind on writes. The default value for `write_nstream` is 1.
`discovered_direct_iosz`	Any file I/O requests larger than the `discovered_direct_iosz` are handled as discovered direct I/O. A discovered direct I/O is unbuffered similar to direct I/O, but does not require a synchronous commit of the inode when the file is extended or blocks are allocated. For larger I/O requests, the CPU time for copying the data into the page cache and the cost of using memory to buffer the I/O data becomes more expensive than the cost of doing the disk I/O. For these I/O requests, using discovered direct I/O is more efficient than regular I/O. The default value of this parameter is 256K.
`initial_extent_ size`	Changes the default initial extent size. VxFS determines the size of the first extent to be allocated to the file based on the first write to a new file. Normally, the first extent is the smallest power of 2 that is larger than the size of the first write. If that power of 2 is less than 8K, the first extent allocated is 8K. After the initial extent, the file system increases the size of subsequent extents (see `max_seqio_extent_size`) with each allocation. Since most applications write to files using a buffer size of 8K or less, the increasing extents start doubling from a small initial extent. `initial_extent_size` can change the default initial extent size to be larger, so the doubling policy will start from a much larger initial size and the file system will not allocate a set of small extents at the start of file. Use this parameter only on file systems that will have a very large average file size. On these file systems, it will result in fewer extents per file and less fragmentation. `initial_extent_size` is measured in file system blocks.

`max_direct_iosz`	The maximum size of a direct I/O request that will be issued by the file system. If a larger I/O request comes in, then it is broken up into `max_direct_iosz` chunks. This parameter defines how much memory an I/O request can lock at once, so it should not be set to more than 20 percent of memory.
`max_diskq`	Limits the maximum disk queue generated by a single file. When the file system is flushing data for a file and the number of pages being flushed exceeds `max_diskq`, processes will block until the amount of data being flushed decreases. Although this doesn't limit the actual disk queue, it prevents flushing processes from making the system unresponsive. The default value is 1MB.
`max_seqio_extent_size`	Increases or decreases the maximum size of an extent. When the file system is following its default allocation policy for sequential writes to a file, it allocates an initial extent that is large enough for the first write to the file. When additional extents are allocated, they are progressively larger (the algorithm tries to double the size of the file with each new extent) so each extent can hold several writes' worth of data. This is done to reduce the total number of extents in anticipation of continued sequential writes. When the file stops being written, any unused space is freed for other files to use. Normally, this allocation stops increasing the size of extents at 2048 blocks, which prevents one file from holding too much unused space. `max_seqio_extent_size` is measured in file system blocks.
	Enables or disables caching on Quick I/O files. The default behavior is to disable caching. To enable caching, set `qio_cache_enable` to 1. On systems with large memories, the database cannot always use all of the memory as a cache. By enabling file system caching as a second level cache, performance may be improved. If the database is performing sequential scans of tables, the scans may run faster by enabling file system caching so the file system will perform aggressive read-ahead on the files.

`write_throttle`

Warning: The `write_throttle` parameter is useful in special situations where a computer system has a combination of a lot of memory and slow storage devices. In this configuration, sync operations (such as `fsync()`) may take so long to complete that the system appears to hang. This behavior occurs because the file system is creating dirty pages (in-memory updates) faster than they can be asynchronously flushed to disk without slowing system performance.

Lowering the value of `write_throttle` limits the number of dirty pages per file that a file system will generate before flushing the pages to disk. After the number of dirty pages for a file reaches the `write_throttle` threshold, the file system starts flushing pages to disk even if free memory is still available. The default value of `write_throttle` typically generates a lot of dirty pages, but maintains fast user writes. Depending on the speed of the storage device, if you lower `write_throttle`, user write performance may suffer, but the number of dirty pages is limited, so sync operations will complete much faster.

Because lowering `write_throttle` can delay write requests (for example, lowering `write_throttle` may increase the file disk queue to the `max_diskq` value, delaying user writes until the disk queue decreases), it is recommended that you avoid changing the value of `write_throttle` unless your system has a a large amount of physical memory and slow storage devices.

If the file system is being used with VxVM, it is recommended that you set the VxFS I/O parameters to default values based on the volume geometry.

If the file system is being used with a hardware disk array or volume manager other than VxVM, align the parameters to match the geometry of the logical disk. With striping or RAID-5, it is common to set `read_pref_io` to the stripe unit size and `read_nstream` to the number of columns in the stripe. For striping arrays, use the same values for `write_pref_io` and `write_nstream`, but for RAID-5 arrays, set `write_pref_io` to the full stripe size and `write_nstream` to 1.

For an application to do efficient disk I/O, it should issue read requests that are equal to the product of `read_nstream` multiplied by `read_pref_io`. Generally, any multiple or factor of `read_nstream` multiplied by `read_pref_io` should be a good size for performance. For writing, the same rule of thumb applies to the `write_pref_io` and `write_nstream` parameters. When tuning a file system, the best thing to do is try out the tuning parameters under a real-life workload.

If an application is doing sequential I/O to large files, it should issue requests larger than the `discovered_direct_iosz`. This causes the I/O requests to be performed as discovered direct I/O requests, which are unbuffered like direct I/O but do not require synchronous inode updates when extending the file. If the file is too large to fit in the cache, then using unbuffered I/O avoids throwing useful data out of the cache and lessons CPU overhead.

About obtaining file I/O statistics using the Quick I/O interface

The `qiostat`command provides access to activity information on Quick I/O files on VxFS file systems. The command reports statistics on the activity levels of files from the time the files are first opened using their Quick I/O interface. The accumulated `qiostat` statistics are reset once the last open reference to the Quick I/O file is closed.

The qiostat command displays the following I/O statistics:

- Number of read and write operations
- Number of data blocks (sectors) transferred
- Average time spent on read and write operations

When Cached Quick I/O is used, `qiostat` also displays the caching statistics when the `-l` (the long format) option is selected.

The following is an example of `qiostat` output:

```
                    OPERATIONS        FILE BLOCKS             AVG TIME(ms)

 FILENAME          READ      WRITE     READ      WRITE       READ      WRITE

 /db01/file1          0         00        0        0.0        0.0

 /db01/file2          0         00        0        0.0        0.0

 /db01/file3      73017     181735   718528    1114227       26.8       27.9

 /db01/file4      13197      20252   105569     162009       25.8      397.0

 /db01/file5          0         00        0        0.0        0.0
```

For detailed information on available options, see the `qiostat`(1M) manual page.

About I/O statistics data

Once you gather the file I/O performance data, you can use it to adjust the system configuration to make the most efficient use of system resources.

There are three primary statistics to consider:

- file I/O activity
- volume I/O activity
- raw disk I/O activity

If your database is using one file system on a striped volume, you may only need to pay attention to the file I/O activity statistics. If you have more than one file system, you may need to monitor volume I/O activity as well.

First, use the `qiostat -r` command to clear all existing statistics. After clearing the statistics, let the database run for a while during a typical database workload period. For example, if you are monitoring a database with many users, let the statistics accumulate for a few hours during prime working time before displaying the accumulated I/O statistics.

To display active file I/O statistics, use the `qiostat` command and specify an interval (using `-i`) for displaying the statistics for a period of time. This command displays a list of statistics such as:

```
                    OPERATIONS        FILE BLOCKS          AVG TIME(ms)

FILENAME          READ      WRITE     READ      WRITE      READ      WRITE

/db01/cust1        218         36      872        144      22.8       55.6

/db01/hist1          0         10        4        0.0      10.0

/db01/nord1         10         14       40         56      21.0       75.0

/db01/ord1          19         16       76         64      17.4       56.2

/db01/ordl1        189         41      756        164      21.1       50.0

/db01/roll1          0         50        0        200       0.0       49.0

/db01/stk1        1614        238     6456        952      19.3       46.5

/db01/sys1           0         00        0        0.0       0.0

/db01/temp1          0         00        0        0.0       0.0

/db01/ware1          3         14       12         56      23.3       44.3

/logs/log1           0         00        0        0.0       0.0

/logs/log2           0        217 0   2255        0.0       6.8
```

File I/O statistics help identify files with an unusually large number of operations or excessive read or write times. When this happens, try moving the “hot” files or busy file systems to different disks or changing the layout to balance the I/O load.

```
Mon May 11 16:21:20 2015

/db/dbfile01               813     0      813       0     0.3   0.0

/db/dbfile02                 0   813        0     813     0.0   5.5

Mon May 11 16:21:25 2015

/db/dbfile01               816     0      816       0     0.3    0.0

/db/dbfile02                 0   816        0             816    0.0  5.3

Mon May 11 16:21:30 2015

/db/dbfile01                 0     0        0               0    0.0  0.0

/db/dbfile02                 0     0        0               0    0.0  0.0
```

About I/O statistics

When running your database through the file system, the read-write lock on each file allows only one active write per file. When you look at the disk statistics using `iostat`, the disk reports queueing time and service time. The service time is the time that I/O spends on the disk, and the queueing time is how long it waits for all of the other I/Os ahead of it. At the volume level or the file system level, there is no queueing, so `vxstat`and `qiostat`do not show queueing time.

For example, if you send 100 I/Os at the same time and each takes 10 milliseconds, the disk reports an average of 10 milliseconds of service and 490 milliseconds of queueing time. The `vxstat`and `qiostat`report an average of 500 milliseconds service time.

About tuning DB2 databases

To achieve optimal performance on your DB2 database, the database needs to be tuned to work with VxFS. There are a number of DB2 parameters that you can tune to improve your DB2 database performance.

DB2_USE_PAGE_CONTAINER_TAG

By default, DB2 stores a container tag in the first extent of each DMS container, whether it is a file or a device. The container tag is the metadata for the container.

(Before DB2 v8.1, the container tag was stored in a single page, so it required less space in the container.) It is recommended that you keep this variable set to `OFF`.

The `DB2_USE_PAGE_CONTAINER_TAG` variable is set using the `db2set` command.

```
$ db2set DB2_USE_PAGE_CONTAINER_TAG=OFF
$ db2stop
$ db2start
```

If you set this registry variable to `ON` when you use RAID devices for containers, I/O performance might degrade. Because for RAID devices you create table spaces with an extent size equal to or a multiple of the RAID stripe size, setting the `DB2_USE_PAGE_CONTAINER_TAG` to `ON` causes the extents not to line up with the RAID stripes. As a result, an I/O request might need to access more physical disks than would be optimal. Users are strongly advised against enabling this registry variable.

DB2_PARALLEL_IO

This setting is used to force parallel I/O to occur on tablespaces. This is important in combination with the `DB2_STRIPED_CONTAINERS` setting, as RAID devices have more than one physical disk and therefore can sustain a greater I/O load than non-RAID devices. DB2 achieves this parallelism by enabling multiple prefetch threads on enabled tablespaces.

The `DB2_PARALLEL_IO` variable is set using the `db2set` command. To enable parallel I/O on all tablespaces, you would run the commands:

```
$ db2set DB2_PARALLEL_IO=*
$ db2stop ; db2start
```

To enable parallel I/O on a subset of all tablespaces, you need to know the tablespace identifying number and supply a list of tablespace ids, comma separated, to the `db2set` command:

```
$ db2 connect to PROD
$ db2 list tablespaces
$ db2 terminate
$ db2set DB2_PARALLEL_IO=3,4,8,9
$ db2stop ; db2start
```

As per the examples, you must stop and restart your instance after modifying the `DB2_PARALLEL_IO` setting. It is also recommended that DB2_PARALLEL_IO be enabled for tablespaces residing on RAID devices when PREFETCHSIZE > EXTENTSIZE.

PREFETCHSIZE and EXTENTSIZE

Prefetching is a behavior that increases database performance in DSS type environments, or environments where data are large enough that they cannot be maintained in the database memory. The extentsize is important in environments where DB2 tablespaces and containers reside upon RAID devices. In general, the EXTENTSIZE should always be equal to or a multiple of the RAID stripe size

By setting DB2_PARALLEL_IO, the tablespace PREFETCHSIZE takes on special meaning. PREFETCHSIZE is divided by the EXTENTSIZE to arrive at the degree of I/O parallelism. Without this environment variable set, the degree of I/O parallelism is normally derived from the number of containers. Because RAID often has only one container, it is important to set the PREFETCHSIZE as a multiple of the EXTENTSIZE, to provide a sufficient number of IO_SERVERS (at least one per physical disk), and to assign the tablespace to a bufferpool that is sufficiently large to accommodate to prefetch requests.

In the general case, we calculate EXTENTSIZE based on the physical attributes of the volume. PREFETCHSIZE should be at least EXTENTSIZE * the number of containers in order to obtain a good I/O parallelism. When dealing with RAID devices however, we may have only a single container within a tablespace and so the number of containers would be substituted with the number of devices or columns in the volume.

When using DMS device containers, such as Quick I/O files, the operating system does not perform any prefetching or caching.

When you need to have a greater control over when and where memory is allocated to caching and prefetching of DB2 tablespace data, use Cached Quick I/O.

If you prefer to assign more system memory permanently to DB2 bufferpools, set PREFETCHSIZE and the DB2_PARALLEL_IO settings for tablespaces.

For example, we have a VxVM RAID0 volume striped across 10 physical disks with a stripe column size of 64k. We have created a VxFS file system on this volume and are about to create a tablespace of DMS containers:

```
$ qiomkfile -s 1G /db2_stripe/cont001

$ db2 connect to PROD

$ db2 create tablespace DATA1 managed by database \

using (device '/db2_stripe/cont001' 128000 ) \
pagesize 8k extentsize 8 prefetchsize 80
```

```
using (FILE '/db2_stripe/cont001' 128000) \
pagesize 8k extentsize 8 prefetchsize 80 \
no file system caching

$ db2 terminate
```

In this example, we ensure that each read of an extent will span 1 physical drive (column width is 64k and our extentsize is 8 * 8k pagesize). When prefetching, we take a full stripe read at a time (there are 10 disks in the stripe, so 10 * an extent is 80 pages). Observe that the `PREFETCHSIZE` remains a multiple of the `EXTENTSIZE`. These settings would provide a good environment for a database which in general uses clusters of data around 640k or less. For larger database objects or more aggressive prefetch on data, the specified `PREFETCHSIZE` can be multiplied.

If the database's main workload requires good sequential I/O performance, such as a DSS workload, then the settings for Cached Quick I/O and `PREFETCHSIZE` becomes even more important.

There are some cases where setting the `PREFETCHSIZE` to large values or having prefetching at all may degrade performance. In OLTP environments where data access is very random, you may need to turn off prefetching on a tablespace, or minimize the effect by setting PREFETCHSIZE equal to EXTENTSIZE.

It is still very important in these types of environment to ensure that access to indexes is very fast and preferably all heavily accessed indexes are cached by Cached Quick I/O or in bufferpool memory.

INTRA_PARALLEL

The INTRA_PARALLEL setting is usually set on machines with multiple CPUs when large and complex queries are being executed. This may not provide any performance advantage in OLTP environments, as queries in these types of environments are normally very simple, short and highly repetitive. However, for DSS or OLAP environments, enabling this option may provide significant performance improvements.

NUM_IOCLEANERS

Specifies the number of async page cleaners. The cleaners flush dirty pages from the buffer pool, freeing the space for the threads pulling data in from storage. Important to tune this if the PREFETCH settings for the database are being modified. To avoid I/O wait, set this parameter higher if insert/update/delete is heavy or prefetch large.

NUM_IOSERVERS

Specifies the number of I/O servers for the database. These servers implement prefetch and async I/O operations. Should be set to at least the number of physical devices on the host system in order to maximize I/O parallelism.

CHNGPGS_THRESH

Specifies the threshold at which the IOCLEANERS start flushing dirty pages. A lower value indicates that cleaning should being earlier.

Table scans

Quick I/O in its default mode performs all I/O as direct I/O.

In the case of single-threaded sequential scans (common in decision support system (DSS) workloads), using buffered reads can yield better performance. Because the file system detects these sequential reads and performs read-aheads, the next few blocks that are requested by DB2 are readily available in the system buffer cache and are simply copied to the DB2 buffer pool. Because access from memory is inherently faster than access from disk, this achieves a significant reduction in response time.

To handle large sequential scans when using Quick I/O, two methods are available to improve performance:

- Modify the DB2 PREFETCH setting to force reading in data before it is required.
- The second method is to enable Cached Quick I/O for the files that would be read by the DB2 sequential scan process. Cached Quick I/O enables buffered reads, and the automatic file system read-ahead helps lower response times by pre-loading data. A major advantage of using Cached Quick I/O is that it does not require any database level changes and so does not require the database to be restarted for changes to take effect.

Asynchronous I/O

Asynchronous I/O allows the DB2 database to schedule multiple I/Os without waiting for the I/O to complete. When the I/O completes, the kernel notifies the `DB2` using an interrupt.

Quick I/O supports kernel asynchronous I/O (KAIO), which reduces CPU utilization and improves transaction throughput. The DB2 database engine, by default, will make use of asynchronous I/O when using DMS containers.

Buffer pools

The UNIX buffer cache plays an important role in performance when using UFS, HFS, or JFS in buffered I/O mode. However, when using Quick I/O, the database buffer pools must be tuned as if raw devices are being used. You can allocate more memory to the database buffer pools because Quick I/O bypasses the file system cache to improve database performance. Memory pages normally allocated to the file system cache can be allocated to the database buffer pools. Adjusting the size and number of buffer pools requires restarting the database. Cached Quick I/O can be used to dynamically modify memory allocation to the database without requiring a restart.

Memory allocation

Never configure DB2 to use more than 75% of the physical memory available on the system. DB2 may have to compete with other processes for system memory resources, and all of these potential processes must be considered when sizing and allocating memory. In the ideal configuration, a system that is dedicated to DB2 simplifies the tuning and monitoring issues and ensures best performance.

TEMPORARY tablespaces

When more than one TEMPORARY tablespace exists in the database, they will be used in round-robin fashion in order to balance their usage. See the Administration Guide for information on using more than one tablespace, rebalancing and recommended values for EXTENTSIZE, PREFETCHSIZE, OVERHEAD, and TRANSFERRATE.

DMS containers

When you have more than one container in a DMS tablespace, it is important to ensure that all containers are the same physical, and logically declared, size. DB2 stripes data across available containers in a tablespace, writing in a round-robin fashion. If containers are not sized the same, then once the tablespace becomes sufficiently full, all I/O activity could be occurring to one physical file or device. This will incur a heavy performance penalty, especially when coupled with high values of the `NUM_IOCLEANERS`, `NUM_IOSERVERS` and `PREFETCHSIZE` configuration settings.

When extending tablespace containers using the `qiomkfile`command, ensure that you maintain this equal length across all containers in a tablespace.

Data, indexes, and logs

It is always important to separate database data and log files. The write patterns for these types of object are very different and so mixing them on the same device will adversely affect performance. Log writes are always sequential and high bandwidth, whereas writes to data tablespaces can range from random to large and sequential. It is important to ensure that log writes are fast and do not suffer from device latency in order to provide the highest performing database environment.

When using SMS tablespaces, it is not possible to separate data and indexes onto different devices. This means that there is no way to reduce contention for I/O and memory between these two types of database object. However, when using DMS devices, it is possible to place the data and indexes of tables into different tablespaces. This can provide much improved performance in environments which have very heavy usage of indexes and/or constrained memory.

In addition to being able to separate and therefore easily monitor I/O to the data and indexes, assigning indexes to a separate tablespace allows you to assign a dedicated bufferpool to the indexes or enable Cached Quick I/O on the index containers as required. This can greatly improve performance in environments where you want to ensure that indexes are always in memory and that there is no contention between data and indexes for a single bufferpools resources.

Database statistics

The DB2 database maintains internal information and statistics about the physical layout of data in the database. These internal statistics are used by the prefetch and I/O scheduling threads to plan operations in advance and can therefore have a very large impact on performance. With regular database activity, the statistics can become incorrect and therefore begin to have an adverse affect on I/O planning. This is especially true after major loads of new data, creating indexes on tables and heavy table activity involving large numbers of delete or update queries.

DB2 provides several tools to assist in updating these statistics and therefore enable continued and accurate I/O planning. These tools can be run from the db2 command prompt and are called RUNSTATS, REORG and REORGCHK tools. They should be run regularly to ensure optimal database performance.

See the System Catalog Statistics section in the *DB2 Administration Guide* and the section on CLP commands in the *DB2 Command Reference*.

About tuning AIX Virtual Memory Manager

If you are using either Cached Quick I/O or buffered I/O (that is, plain VxFS files without Quick I/O or mount options specified), it is recommended that you monitor

any paging activity to the swap device on your database servers. To monitor swap device paging, use the `vmstat -I` command. Swap device paging information appears in the `vmstat -I` output under the columns labeled `pi` and `po` (for paging in and paging out from the swap device, respectively). Any nonzero values in these columns indicates swap device paging activity.

For example:

```
# /usr/bin/vmstat -I

kthr      memory              page                          faults     cpu
--------  ------------------- ----------------------------- ---------- -----------
r  b  p   avm     fre      fi    fo pi  po  fr    sr    in    sy     cs    us sy id wa

5  1  0   443602  1566524  661   20 0   0   7     28    4760  37401  7580  11 7  43 38

1  1  0   505780  1503791  18    6  0   0   0     0     1465  5176   848   1  1  97 1

1  1  0   592093  1373498  1464  1  0   0   0     0     4261  10703  7154  5  5  27 62

3  0  0   682693  1165463  3912  2  0   0   0     0     7984  19117  15672 16 13 1  70

4  0  0   775730  937562   4650  0  0   0   0     0     10082 24634  20048 22 15 0  63

6  0  0   864097  715214   4618  1  0   0   0     0     9762  26195  19666 23 16 1  61

5  0  0   951657  489668   4756  0  0   0   0     0     9926  27601  20116 24 15 1  60

4  1  0   1037864 266164   4733  5  0   0   0     0     9849  28748  20064 25 15 1  59

4  0  0   1122539 47155    4476  0  0   0   0     0     9473  29191  19490 26 16 1  57

5  4  0   1200050 247      4179  4  70  554 5300  27420 10793 31564  22500 30 18 1  52

6  10 0   1252543 98       2745  0  138 694 4625  12406 16190 30373  31312 35 14 2  49

7  14 0   1292402 220      2086  0  153 530 3559  17661 21343 32946  40525 43 12 1  44

7  18 0   1319988 183      1510  2  130 564 2587  14648 21011 28808  39800 38 9  3  49
```

If there is evidence of swap device paging, proper AIX Virtual Memory Manager (VMM) tuning is required to improve database performance. VMM tuning limits the amount of memory pages allocated to the file system cache. This prevents the file

system cache from stealing memory pages from applications (which causes swap device page-out) when the VMM is running low on free memory pages.

The command to tune the AIX VMM subsystem is:

```
# /usr/samples/kernel/vmtune
```

Changes made by `vmtune` last until the next system reboot. The VMM kernel parameters to tune include: `maxperm`, `maxclient`, and `minperm`. The `maxperm` and `maxclient` parameters specify the maximum amount of memory (as a percentage of total memory) that can be used for file system caching. The maximum amount of memory for file system caching should not exceed the amount of unused memory left by the AIX kernel and all active applications. Therefore, it can be calculated as:

100*(T-A)/T

where T is the total number of memory pages in the system and A is the maximum number of memory pages used by all active applications.

The `minperm` parameter should be set to a value that is less than or equal to `maxperm`, but greater than or equal to 5.

For more information on AIX VMM tuning, see the `vmtune`(1) manual page and the performance management documentation provided with AIX.

The following is a tunable VxFS I/O parameter:

VMM Buffer Count

(`-b` <value> option)

Sets the virtual memory manager (VMM) buffer count. There are two values for the VMM: a default value based on the amount of memory, and a current value. You can display these two values using `vxtunefs -b`. Initially, the default value and the current value are the same. The `-b` value option specifies an increase, from zero to 100 per cent, in the VMM buffer count from its default. The specified value is saved in the file `/etc/vx/vxfssystem` to make it persistent across VxFS module loads or system reboots.

In most instances, the default value is suitable for good performance, but there are counters in the kernel that you can monitor to determine if there are delays waiting for VMM buffers. If there appears to be a performance issue related to VMM, the buffer count can be increased. If there is better response time on the system, it is a good indication that VMM buffers were a bottleneck.

The following fields displayed by the `kdb vmker` command can be useful in determining bottlenecks.

`THRPGIO buf wait (_waitcnt) value`

This field may indicate that there were no VMM buffers available for pagein or pageout. The thread was blocked waiting for a VMM buffer to become available. The count is the total number of waits since cold load. This field, together with pages “paged in” and pages “paged out” displayed by the kdb vmstat command can be used to determine if there are an adequate number of VMM buffers. The ratio:

waitcnt / pageins+pageouts

is an indicator of waits for VMM buffers, but cannot be exact because pageins + pageouts includes page I/Os to other file systems and pageing space. It is not possible to give a typical value for this ratio because it depends on the amount of memory and page I/Os to file systems other than VxFS. A number greater than 0.1 may indicate a VMM buffer count bottleneck. Other relevant fields displayed by `kdb vmker` are:

- `THRPGIO partial cnt (_partialcnt) value`
  This field indicates page I/O was done in two or more steps because there were fewer VMM buffers available than the number of pages requiring I/O.
- `THRPGIO full cnt (_fullcnt) value`
  All the VMM buffers were found for all the pages requiring I/O.

Chapter 20

Troubleshooting SFDB tools

This chapter includes the following topics:

- About troubleshooting Storage Foundation for Databases (SFDB) tools
- About the vxdbd daemon
- Troubleshooting vxdbd
- Resources for troubleshooting SFDB tools
- Upgrading Storage Foundation for Databases (SFDB) tools from 5.0.x to 7.4.2 (2184482)

About troubleshooting Storage Foundation for Databases (SFDB) tools

Storage Foundation for Databases (SFDB) tools are deployed with several Storage Foundation products, and as a result can be affected by any issue with those products. The first step in case of trouble should be to identify the source of the problem. It is rare to encounter problems in Storage Foundation for Databases (SFDB) tools; more commonly the problem can be traced to setup issues or problems in the base products.

Use the information in this chapter to diagnose the source of problems. Indications may point to base product set up or configuration issues, in which case solutions may require reference to other Storage Foundation documentation. In cases where indications point to a component product or to DB2 as the source of a problem, it may be necessary to refer to the appropriate documentation to resolve it.

For troubleshooting Storage Foundation product issues:

- *Storage Foundation Administrator's Guide*
- *Storage Foundation for Cluster File System High Availability Administrator's Guide*

Running scripts for engineering support analysis for SFDB tools

Troubleshooting scripts gather information about the configuration and status of your product and its modules. The scripts identify package information, debugging messages, console messages, and information about disk groups and volumes. Forwarding the output of these scripts to Veritas Tech Support can assist with analyzing and solving any problems.

To obtain SFDB repository and log information

- Run:

  ```
  # /opt/VRTSspt/VRTSexplorer/VRTSexplorer
  ```

  Send the output to Support.

Storage Foundation for Databases (SFDB) tools log files

Checking the following log files can provide useful diagnostic information.

SFDB tools commands log files are located in the /var/vx/vxdba/logs directory.

About the vxdbd daemon

The SFDB commands are run as the DBA user. DBA users need to perform several operations, such as creating snapshots and mounting file systems, as the root user. The `vxdbd` daemon is used by the SFDB commands to run privileged commands, or when communicating with the SFDB repository on a different host.

Starting and stopping vxdbd

The `vxdbd` daemon is configured to automatically start when the system boots up. The script at /opt/VRTS/bin/sfae_config can be used to stop and start the daemon, and to query its status. Only the root user can start and stop the daemon.

To query the daemon status

- Run the command:

  ```
  # /opt/VRTS/bin/sfae_config status
  ```

To start the daemon

- Run the command:

```
# /opt/VRTS/bin/sfae_config enable
```

To stop the daemon

- Run the command:

```
# /opt/VRTS/bin/sfae_config disable
```

Note: Most SFDB commands require that the `vxdbd` daemon be running.

Configuring listening port for the vxdbd daemon

The `vxdbd` daemon listens on TCP port 3233, by default. If this port is in use by some other application, `vxdbd` can be configured to listen on an alternate port. In cluster environments, `vxdbd` must be configured to listen on the same port on all the nodes.

To configure listening port for the vxdbd daemon

1. Stop the `vxdbd` daemon:

```
# /opt/VRTS/bin/sfae_config disable
```

2. Set VXDBD_PORT to the desired port number by editing the `/etc/vx/vxdbed/admin.properties` configuration file.
3. Start the `vxdbd` daemon:

```
# /opt/VRTS/bin/sfae_config enable
```

Limiting vxdbd resource usage

Although the `vxdbd` daemon is light-weight in terms of resource usage, system administrators might want to additionally restrict its resource usage. This can be controlled by using two configuration parameters in `/etc/vx/vxdbed/admin.properties`:

- MAX_CONNECTIONS: This parameter controls the maximum number of simultaneous requests that vxdbd should allow.
- MAX_REQUEST_SIZE: This parameter controls the maximum size of a single request (in bytes).

Setting these parameters too low may cause SFDB commands to fail. The following are the suggested minimum values for these parameters:

- MAX_CONNECTIONS: 5
- MAX_REQUEST_SIZE: 1048576 (1 MB)

Note: Restart vxdbd after making any changes to these parameters for the changes to take effect.

Configuring encryption ciphers for vxdbd

Communication with the vxdbd daemon is encrypted. The encryption algorithms used for communication can be configured by editing the `/var/vx/vxdba/auth/vxdbd/root/.VRTSat/profile/VRTSatlocal.conf` configuration file. The SSLCipherSuite configuration parameter specifies the ciphers that are allowed when establishing a secure connection. Refer to the OpenSSL ciphers(1) man page for details on the acceptable values of this parameter.

For example, to configure `vxdbd` to deny communication using the medium-strength and low-strength ciphers, set the SSLCipherSuite parameter as follows:

```
"SSLCipherSuite"="HIGH:!MEDIUM:!eNULL:!aNULL:!SSLv2:!LOW"
```

Troubleshooting vxdbd

If the `vxdbd` daemon is not running, SFDB commands may fail with an error message. The following steps can help in troubleshooting the instances of the daemon that are down:

- Use the `/opt/VRTS/bin/sfae_config status` to verify that the daemon is running.
- If the output states that the daemon is not running, use the `/opt/VRTS/bin/sfae_config enable` command to start the daemon.
- If the daemon fails to start, verify that no other process is listening on the same port. If there is such a process, stop that process, or configure `vxdbd` to listen on a different port.
- The daemon generates logs in the file at `/var/vx/vxdba/logs/vxsfaed.log`. To enable verbose logging, edit the `/etc/vx/vxdbed/admin.properties` configuration file and set LOG_LEVEL to DEBUG.

Resources for troubleshooting SFDB tools

If Storage Foundation for Databases (SFDB) commands fail, use the following resources to troubleshoot.

See "SFDB logs" on page 187.

See "SFDB error messages" on page 188.

See "SFDB repository and repository files" on page 188.

SFDB logs

The SFDB commands generate logs that can be used to narrow down to the actual problem.

Log files:

- Log files are generated in the location `/var/vx/vxdba/logs`.
- There are two kind of logs:
  - User logs are generated in the <user> folder.
  - Logs from vxdbd and other root operations are generated in the logs folder.
- The user log files have the naming convention:
  log_<service>_<app>_<service_id><app_id>.log.
  A system.log is also present until `vxsfadm` can recognize the service and the application identifiers.
- The vxdbd logs have the name vxsfaed.log.
  A system.log also exists for all root operations performed.
- The log files are archived after they reach a threshold of 1MB and are backed up as
  log_<service><application><application_identifier><service_identifier>.log.<randomnumber>
  Every log file has a pointer to the previously archived log.

Log levels:

- Log levels can be set using the environment variable SFAE_LOG_LEVEL.
- The following additional environment variables can be set that override SFAE_LOG_LEVEL:
  - APP_LOG_LEVEL: Log application-specific operations.
  - SER_LOG_LEVEL: Log VxFS/VxVM stack specific operations.
  - REP_LOG_LEVEL: Log repository operations.
  - FSM_LOG_LEVEL: Log vxsfadm engine-specific operations.

- The log levels can be set to the following levels:
  - FATAL: Logs only fatal messages.
  - ERROR: Logs errors and above messages.
  - WARN: Logs warning and above messages.
  - INFO: Logs info and above messages.
  - DEBUG: Logs debug and above messages.
- The default log level is DEBUG.

Log messages:

- The actual log messages appear in the following format:

  `yyyy/mm/dd hh:mm:ss: <loglevel> : <module> : <message>`

  For example:

SFDB error messages

Each error message is based on a standard template wherein the following fields are displayed:

- MESSAGE: The actual error message.
- REASON: The reason for the error.
- ACTION: The action to be taken to correct the error.

These fields provide you with precise information about the cause of a problem.

SFDB repository and repository files

Caution: Any troubleshooting that involves operations related to the SFDB repository must be done under the supervision of a trained Veritas Engineer.

The name of the repository database is in the following format:
`dbed-<application>-<application identifier>-repository.db`.

For example: `dbed-db2-db2inst1_sfaedb-repository.db`

The repository database can be queried to view a variety of information about an application instance. This includes the following tables:

- `_fsm_state_`: Displays the progress of various services.

- `_operational_data_`: Displays the various configuration values set for various services.
- `files`: Displays the files used by the services.

SFDB tools create files under `<repositorylocation>/files`.

- These files are used for various processing activities for all services.
- The `files` table from the repository points to the various file names used by a service under the `files` directory.

Upgrading Storage Foundation for Databases (SFDB) tools from 5.0.x to 7.4.2 (2184482)

The `sfua_rept_migrate`command results in an error message after upgrading SFHA or SF for Oracle RAC version 5.0 or 5.0MP3 to SFHA or SF for Oracle RAC 7.4.2.

The `sfua_rept_migrate`command results in an error message after upgrading SFHA or SF for Oracle RAC version 5.0 to SFHA or SF for Oracle RAC 7.4.2.

When upgrading from Veritas InfoScale products version 5.0 or 5.0MP3 to Veritas InfoScale products 7.4.2 the S*vxdbms3 startup script is renamed to NO_S*vxdbms3. The S*vxdbms3 startup script is required by `sfua_rept_upgrade`. Thus when `sfua_rept_upgrade` is run, it is unable to find the S*vxdbms3 startup script and gives the error message:

```
/sbin/rc3.d/S*vxdbms3 not found
SFORA sfua_rept_migrate ERROR V-81-3558 File:  is missing.
SFORA sfua_rept_migrate ERROR V-81-9160 Failed to mount repository.
```

Workaround: Before running `sfua_rept_migrate`, rename the startup script NO_S*vxdbms3 to S*vxdbms3.