Veritas Storage Foundation™ for Sybase Administrator's Guide

Solaris

5.0

N18528F

symantec™

Veritas Storage Foundation™ for Sybase Administrator's Guide

Documentation version 5.0

PN: N18528F

Legal Notice

The Licensed Software and Documentation are deemed to be "commercial computer software" and "commercial computer software documentation" as defined in FAR Sections 12.212 and DFARS Section 227.7202.

Symantec Corporation 20330 Stevens Creek Blvd. Cupertino, CA 95014 USA

http://www.symantec.com

Technical Support

For technical assistance, visit http://support.veritas.com and select phone or email support. Use the Knowledge Base search feature to access resources such as TechNotes, product alerts, software downloads, hardware compatibility lists, and our customer email notification service.

Contents

Chapter 1 Introducing Veritas Storage Foundation for Sybase

About Veritas Storage Foundation for Sybase 11
Components of Veritas Storage Foundation for Sybase 12
How Veritas Volume Manager works 13
About volumes 14
About disk groups 15
About volume layouts 15
About online relayout 18
About volume resynchronization 18
About dirty region logging 19
About volume sets 19
About volume snapshots 20
About Veritas FastResync 20
About disk group split and join 21
About hot-relocation 21
About DMP-supported disk arrays 22
About dynamic LUN expansion 22
About Storage Expert 22
About cluster functionality (optional) 23
About Veritas Volume Replicator (optional) 23
How Veritas File System works 23
About Veritas Quick I/O 23
About Veritas Cached Quick I/O 24
About Veritas Concurrent I/O 25
About extent-based allocation 25
About fast file system and database recovery 25
About online system administration 26
About cross-platform data sharing 27
Support for multi-volume file systems 27
About Quality of Storage Service (optional) 27
Support for large file systems and large files (optional) 27
About restoring file systems using Storage Checkpoints 28
About quotas 28
About cluster functionality (optional) 29
About Veritas Storage Foundation/High Availability for Sybase (optional) 29

Chapter 2 Setting up dataservers

Tasks for setting up new databases 31
About setting up a disk group 32
Disk group configuration guidelines 33
Creating a disk group 33
Adding disks to a disk group 34
About selecting a volume layout 35
How to choose appropriate stripe unit sizes 36
How to choose between mirroring and RAID-5 36
Volume configuration guidelines 37
Creating a volume 37
Creating a volume set 38
Adding a volume to a volume set 39
File system creation guidelines 40
Creating a VxFS file system 41
Large file system and large file support 42
Multi-volume support 43
Mounting a file system 43
Unmounting a file system 44
About fragmentation 45
How to control fragmentation 46
Types of fragmentation 46
How to monitor fragmentation 46
Defragmenting a file system 47
Resizing a file system 49
Resizing a file system and the underlying volume 50
About Quick I/O 57
How Quick I/O works 57
How Quick I/O improves database performance 58
About Quick I/O requirements 59
How to set up Quick I/O 60

Chapter 3 Using Veritas Quick I/O

About Quick I/O 57
How Quick I/O works 57
How Quick I/O improves database performance 58
About Quick I/O requirements 59
How to set up Quick I/O 60
Preallocating space for Quick I/O files using the setext command 61
Creating database files as Quick I/O files using qiomkfile 62
Accessing regular VxFS files as Quick I/O files 65
Converting Sybase files to Quick I/O files 67

Displaying Quick I/O status and file attributes 74
Extending a Quick I/O file 75
Recreating Quick I/O files after restoring a database 76
Disabling Quick I/O 78

Chapter 4 Using Veritas Cached Quick I/O

About Cached Quick I/O 79
How Cached Quick I/O works 79
How Cached Quick I/O improves database performance 80
How to set up Cached Quick I/O 81
Enabling Cached Quick I/O on a file system 81
Enabling and disabling the qio_cache_enable flag 82
Making Cached Quick I/O settings persistent across reboots and mounts 83
Using vxtunefs to obtain tuning information 84
Determining candidates for Cached Quick I/O 85
Collecting I/O statistics 85
About I/O statistics 86
Effects of read-aheads on I/O statistics 87
Other tools for analysis 88
Enabling and disabling Cached Quick I/O for individual files 88
Setting cache advisories for individual files 88
Making individual file settings for Cached Quick I/O persistent 89
Determining individual file settings for Cached Quick I/O using qioadmin 90

Chapter 5 Using Veritas Concurrent I/O

About Concurrent I/O 93
How Concurrent I/O works 93
Enabling and disabling Concurrent I/O 94
Enabling Concurrent I/O 94
Disabling Concurrent I/O 95

Chapter 6 Converting existing database configurations to VxFS

Converting native file systems to VxFS with Quick I/O 97
Upgrading from earlier VxFS version layouts 98
Converting from raw devices 99

Chapter 7 Using volume snapshots for dataserver backup and off-host processing

About snapshot volumes 102
Backup and off-host processing applications 103
FastResync of snapshot volumes 103
Disk group split and join 104
Preparing hosts for database backup or off-host processing 107
Single-host configuration 107
Two-host configuration 108
Upgrading existing volumes to use VxVM 4.0 features 109
Sybase Adaptive Server Enterprise 12.5 quiesce feature 111
How to set up volume snapshots with Sybase ASE 12.5 server 112
Implementing online backup or off-host processing 113
..... 119
Creating a warm standby server 122
Resynchronizing the snapshot to your ASE dataserver 134
Recovering the database from a backup image 139
Refreshing a snapshot database image 143
Dissociating a snapshot volume 145
Removing a snapshot volume 145

Chapter 8 Tuning for performance

Additional documentation 147
About tuning VxVM 147
About obtaining volume I/O statistics 148
About tuning VxFS 149
How monitoring free space works 149
How tuning VxFS I/O parameters works 150
About tunable VxFS I/O parameters 151
About obtaining file I/O statistics using the Quick I/O interface 156
About I/O statistics data 156
About tuning Sybase dataservers 158
Sybase tempdb database 158
Sybase sybsecurity database 159
Placement of the transaction logs 159
Database device layout 160
Nonclustered indexes placement 160
About tuning Solaris for Sybase 160
maxuprc 161
shmmax 161
shmmni 161

shmseg 161
semmap 161
semmni 161
semmns 162
semmnu 162

Appendix A Veritas Storage Foundation for Sybase command line interface

Overview of commands 163
About the command line interface 164
Converting VxFS files to Quick I/O using qio_convertdbfiles 164
Identifying VxFS files to convert to Quick I/O using qio_getdbfiles 166
Recreating Quick I/O files using qio_recreate 167
Managing log files using edgetmsg2 169

Glossary

Index

Chapter 1

Introducing Veritas Storage Foundation for Sybase

This chapter includes the following topics:

- About Veritas Storage Foundation for Sybase
- How Veritas Volume Manager works
- How Veritas File System works
- About Veritas Storage Foundation/High Availability for Sybase (optional)

About Veritas Storage Foundation for Sybase

There are two versions of this product:

- Veritas Storage Foundation for Sybase Standard Edition
- Veritas Storage Foundation for Sybase Enterprise Edition
  The Enterprise Edition contains everything in the Standard Edition plus FastResync, disk group split and join, Quality of Storage Service, and support for large file systems (up to 8 exabytes).

Note: Veritas Storage Foundation/High Availability (HA) for Sybase is available only with the Enterprise Edition.

Unless otherwise noted, features pertain to both the Standard and Enterprise Edition products.

Components of Veritas Storage Foundation for Sybase

Veritas Storage Foundation for Sybase combines the strengths of the core technology products with database-specific enhancements to offer performance, availability, and manageability for Sybase database servers.

Veritas Storage Foundation for Sybase includes the following products:

- Veritas Volume Manager (VxVM)
  A disk management subsystem that supports disk striping, disk mirroring, and simplified disk management for improved data availability and performance.
- Veritas File System (VxFS)
  A high-performance, fast-recovery file system that is optimized for business-critical database applications and data-intensive workloads. VxFS offers online administration, letting you perform most frequently scheduled maintenance tasks (including online backup, resizing, and file system changes) without interrupting data or system availability. VxFS also provides support for large file systems (of more than 8 exabytes in a 64-bit environment) and large files (in the exabyte range in a 64-bit environment).
  Veritas File System offers the following performance-enhancing features that are of particular interest in a database environment:
  - Veritas Quick I/O is a VxFS feature that improves the throughput for Sybase databases built on VxFS file systems. Quick I/O delivers raw device performance to databases run on VxFS, providing the administrative advantages of using file systems without the performance penalties.
  - Veritas Cached Quick I/O further enhances database performance by leveraging large system memory to selectively buffer the frequently accessed data.
  - Veritas 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 relational databases (such as Sybase), which do not have a sequential read/write requirement, to access files concurrently.
- Veritas Enterprise Administrator
  Veritas Enterprise Administrator (VEA) is the infrastructure that allows you to access Veritas Storage Foundation for Sybase, Veritas Volume Manager, and Veritas File System information and features through the GUI.

An optional High Availability (HA) version of Veritas Storage Foundation for Sybase Enterprise Edition, which includes Veritas Cluster Server, is available for customers who have high system-availability requirements.

How Veritas Volume Manager works

Databases require their storage media to be robust and resilient to failure. It is vital to protect against hardware and disk failures and to maximize performance using all the available hardware resources. Using a volume manager provides this necessary resilience and eases the task of management. A volume manager can help you manage hundreds of disk devices and makes spanning, striping, and mirroring easy.

Veritas Volume Manager (VxVM) builds virtual devices called volumes on top of physical disks. Volumes are accessed by a file system, a database, or other applications in the same way physical disk partitions would be accessed. Using volumes, VxVM provides the following administrative benefits for databases:

Table 1-1 Veritas Volume Manager features

Feature	Benefit
Spanning of multiple disks	Eliminates media size limitations.
Striping	Increases throughput and bandwidth.
Mirroring or RAID-5	Increases data availability.
Online relayout	Allows volume layout changes without application or database downtime. Online relayout can be used to change performance or reliability characteristics of unerlying storage.
Volume resynchronization	Ensures that all mirrors contain exactly the same data and that the data and parity in RAID-5 volumes agree.
Dirty Region Logging (DRL)	Speeds the recovery of mirrored volumes after a system crash.
Volume snapshots	Allows backup of volumes based on disk mirroring. VxVM provides full-sized and space-optimized instant snapshots, which are online and off-host point-in-time copy solutions.
FastResync	Separately licensed, optional feature that performs quick and efficient resynchronization of stale mirrors. FastResync is included with the Enterprise Edition and is also included as part of the Veritas FlashSnap option with the Standard Edition.

Table 1-1 Veritas Volume Manager features *(continued)*

Feature	Benefit
Disk group split and join	Separately licensed, optional feature that supports general disk group reorganization and allows you to move volume snapshots to another host for off-host backup. Disk group split and join is included with the Enterprise Edition and is also included as part of the Veritas FlashSnap option with the Standard Edition.
Hot-relocation	Automatically restores data redundancy in mirrored and RAID-5 volumes when a disk fails.
Dynamic multipathing (DMP)	Allows for transparent failover, load sharing, and hot plugging of physical disks.
Volume sets	Allows several volumes to be represented by a single logical mount device.
Dynamic LUN Expansion	Allows you to resize a disk after it has been initialized while preserving the existing data on the disk.
Storage Expert	Helps diagnose configuration problems with VxVM.
Cluster Volume Manager (CVM)	Separately licensed, optional feature that allows you to use VxVM in a cluster environment.
Veritas Volume Replicator (VVR)	Separately licensed, optional feature that provides data replication for disaster recovery solutions.
Free space pool management	Simplifies administration and provides flexible use of available hardware.
Online administration	Allows configuration changes without system or database down time.

For a more detailed description of VxVM and its features, refer to the *Veritas Volume Manager Administrator's Guide*.

About volumes

A volume is a virtual disk device that appears to applications, databases, and file systems like a physical disk partition without the physical limitations of a disk partition. A volume consists of one or more plexes, each holding a copy of the selected data in the volume. Due to its virtual nature, a volume is not restricted

to a particular disk or a specific area of a disk. For example, a volume can span multiple disks and can be used to create a large file system.

Volumes consist of other virtual objects that can be manipulated to change the volume's configuration. Volumes and their virtual components are referred to as Volume Manager objects. You can manipulate Veritas Volume Manager objects in a variety of ways to optimize performance, provide redundancy of data, and perform backups or other administrative tasks on one or more physical disks without interrupting applications. As a result, data availability and disk subsystem throughput are improved.

You can change the configuration of a volume without causing disruption to databases or file systems that are using the volume. For example, you can mirror a volume on separate disks or move the volume to use different disk storage.

About disk groups

A disk group is a collection of disks that share a common configuration (for example, configuration objects that belong to a single database). We recommend creating one disk group for each database.

You can move a disk group and its components as a unit from one host to another host. For example, you can move volumes and file systems that belong to the same database and are created within one disk group as a unit. You must configure a given volume from disks belonging to one disk group.

In releases before Veritas Storage Foundation 4.0 for Sybase, the default disk group was `rootdg`. For VxVM to function, the rootdg disk group had to exist and it had to contain at least one disk. This requirement no longer exists, and VxVM can work without any disk groups configured (although you must set up at least one disk group before you can create any volumes of other VxVM objects).

About volume layouts

A Redundant Array of Independent Disks (RAID) is a disk array in which a group of disks appears to the system as a single virtual disk or a single volume. VxVM supports several RAID implementations, as well as spanning.

The following volume layouts are available to satisfy different database configuration requirements:

- Spanning and concatenation
- Striping (RAID-0)
- Mirroring (RAID-1)
- Mirrored-Stripe Volumes (RAID-0+1)

- Striped-Mirror Volumes (RAID-1+0)
- RAID-5

Caution: Spanning or striping a volume across multiple disks increases the chance that a disk failure will result in failure of that volume. Use mirroring or RAID-5 to substantially reduce the chance of a single volume failure caused by a single disk failure.

How spanning and concatenation work

Concatenation maps data in a linear manner onto one or more subdisks in a plex. To access all of the data in a concatenated plex sequentially, data is first accessed in the first subdisk from beginning to end. Data is then accessed in the remaining subdisks sequentially from beginning to end, until the end of the last subdisk.

You can use concatenation with multiple subdisks when there is insufficient contiguous space for the plex on any one disk. This form of concatenation can be used for load balancing between disks, and for head movement optimization on a particular disk.

Concatenation using subdisks that reside on more than one VxVM disk is called spanning.

Warning: Spanning a plex across multiple disks increases the chance that a disk failure results in failure of the assigned volume. Use mirroring (RAID-1) or striping with parity (RAID-5) to reduce the risk that a single disk failure results in a volume failure.

Spanning is useful when you need to read or write data sequentially (for example, reading from or writing to database redo logs) and there is not sufficient contiguous space.

How striping (RAID-0) works

Striping is a technique of mapping data so that the data is interleaved among multiple physical disks. Data is allocated in equal-sized units (called stripe units) that are interleaved between the disks. Each stripe unit is a set of contiguous blocks on a disk. A stripe consists of the set of stripe units at the same position across all columns. A column is a set of one or more subdisks within a striped plex.

Striping is useful if you need large amounts of data written to or read from physical disks, and performance is important. Striping is also helpful in balancing the I/O

load from multi-user applications across multiple disks. By using parallel data transfer to and from multiple disks, striping significantly improves data-access performance.

When striping across multiple disks, failure of any one disk will make the entire volume unusable.

How mirroring (RAID-1) works

Mirroring is a technique of using multiple copies of the data, or mirrors, to duplicate the information contained in a volume. In the event of a physical disk failure, the mirror on the failed disk becomes unavailable, but the system continues to operate using the unaffected mirrors. For this reason, mirroring increases system reliability and availability. A volume requires at least two mirrors to provide redundancy of data. A volume can consist of up to 32 mirrors. Each of these mirrors must contain disk space from different disks for the redundancy to be effective.

How striping plus mirroring (mirrored-stripe or RAID-0+1) works

VxVM supports the combination of mirroring above striping. The combined layout is called a mirrored-stripe layout. A mirrored-stripe layout offers the dual benefits of striping to spread data across multiple disks, while mirroring provides redundancy of data. For mirroring above striping to be effective, the striped plex and its mirrors must be allocated from separate disks.

The layout type of the data plexes in a mirror can be concatenated or striped. Even if only one is striped, the volume is still termed a mirrored-stripe volume. If they are all concatenated, the volume is termed a mirrored-concatenated volume.

How mirroring plus striping (striped-mirror, RAID-1+0 or RAID-10) works

VxVM supports the combination of striping above mirroring. This combined layout is called a striped-mirror layout and mirrors each column of the stripe. If there are multiple subdisks per column, each subdisk can be mirrored individually instead of each column. A striped-mirror volume is an example of a layered volume.

Compared to a mirrored-stripe volume, a striped-mirror volume offers the dual benefits of striping to spread data across multiple disks, while mirroring provides redundancy of data. A striped-mirror volume enhances redundancy, which makes it more tolerant of disk failure, and reduces recovery time after disk failure.

For databases that support online transaction processing (OLTP) workloads, we recommend either mirrored-stripe or striped-mirror volumes to improve database

performance and reliability. For highest availability, we recommend striped-mirror volumes (RAID 1+0).

How striping with parity (RAID-5) works

RAID-5 provides data redundancy through the use of parity (a calculated value that the system uses to reconstruct data after a failure). While data is written to a RAID-5 volume, parity is also calculated by performing an exclusive OR (XOR) procedure on data. The resulting parity is then written to another part of the volume. If a portion of a RAID-5 volume fails, the data that was on that portion of the failed volume can be recreated from the remaining data and the parity.

RAID-5 offers data redundancy similar to mirroring, while requiring less disk space. RAID-5 read performance is similar to that of striping but with relatively slow write performance. RAID-5 is useful if the database workload is read-intensive (as in many data warehousing applications). You can snapshot a RAID-5 volume and move a RAID-5 subdisk without losing redundancy.

About online relayout

As databases grow and usage patterns change, online relayout lets you change volumes to a different layout, with uninterrupted data access. Relayout is accomplished online and in place. Use online relayout to change the redundancy or performance characteristics of the storage, such as data organization (RAID levels), the number of columns for RAID-5 and striped volumes, and stripe unit size.

About volume resynchronization

When storing data redundantly, using mirrored or RAID-5 volumes, Veritas Volume Manager ensures that all copies of the data match exactly. However, if the system crashes, small amounts of the redundant data on a volume can become inconsistent or unsynchronized. For mirrored volumes, unsynchronized data can cause two reads from the same region of the volume to return different results if different mirrors are used to satisfy the read request. In the case of RAID-5 volumes, unsynchronized data can lead to parity corruption and incorrect data reconstruction.

In the event of a system crash, Veritas Volume Manager ensures that all mirrors contain exactly the same data and that the data and parity in RAID-5 volumes agree. This process is called volume resynchronization. Not all volumes require resynchronization after a system failure. VxVM notices when a volume is first written and marks it as dirty. Only volumes that are marked dirty when the system reboots require resynchronization.

The process of resynchronization can impact system and database performance. However, it does not affect the availability of the database after system reboot. You can immediately access the database after database recovery although the performance may suffer due to resynchronization. For very large volumes or for a very large number of volumes, the resynchronization process can take a long time. You can significantly reduce resynchronization time by using Dirty Region Logging (DRL) for mirrored volumes or by making sure that RAID-5 volumes have valid RAID-5 logs. However, using logs can slightly reduce the database write performance.

For most database configurations, we recommend using dirty region logs or the RAID-5 logs when mirrored or RAID-5 volumes are used. It is also advisable to evaluate the database performance requirements to determine the optimal volume configurations for the databases.

About dirty region logging

Dirty region logging (DRL), if enabled, speeds the recovery of mirrored volumes after a system crash. DRL keeps track of the regions that have changed due to I/O writes to a mirrored volume. DRL uses this information to recover only those portions of the volume that need to be recovered.

Note: If a version 20 data change object (DCO) volume is associated with a volume, a portion of the DCO volume can be used to store the DRL log. There is no need to create a separate DRL log for a volume that has a version 20 DCO volume.

For more information on DCOs and DCO volumes, see the *Veritas Volume Manager Administrator's Guide*.

About volume sets

Volume sets are an enhancement to VxVM that allow several volumes to be represented by a single logical mount device. 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-device 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 snapshots

A volume snapshot is a point-in-time image of a volume. Veritas Volume Manager provides three volume snapshot features based on disk mirroring:

- Full-sized instant snapshots
- Space-optimized instant snapshots
- Emulation of third-mirror snapshots

About Veritas FastResync

Veritas FastResync (previously called Fast Mirror Resynchronization or FMR) is included with the Enterprise Edition. It is also included as part of the Veritas FlashSnap option with the Standard Edition.

Veritas FastResync performs quick and efficient resynchronization of stale mirrors (mirrors that are not synchronized). This increases the efficiency of the VxVM snapshot mechanism, and improves the performance of operations such as backup and decision support. Typically, these operations require that the volume is quiescent, and that they are not impeded by updates to the volume by other activities on the system. To achieve these goals, the snapshot mechanism in VxVM creates an exact copy of a primary volume at an instant in time. After a snapshot is taken, it can be accessed independently of the volume from which it was taken.

How non-persistent FastResync works

Non-persistent FastResync allocates its change maps in memory. If non-persistent FastResync is enabled, a separate FastResync map is kept for the original volume and for each snapshot volume. Unlike a dirty region log (DRL), these maps do not reside on disk nor in persistent store. The advantage is that updates to the FastResync map have little impact on I/O performance, as no disk updates need to be performed. However, if a system is rebooted, the information in the map is lost, so a full resynchronization is required when performing a snapback operation. This limitation can be overcome for volumes in cluster-shareable disk groups, provided that at least one of the nodes in the cluster remains running to preserve the FastResync map in its memory.

How persistent FastResync works

Non-persistent FastResync has been augmented by the introduction of persistent FastResync. Unlike non-persistent FastResync, Persistent FastResync keeps the FastResync maps on disk so that they can survive system reboots and system crashes. When the disk groups are rejoined, this allows the snapshot plexes to be quickly resynchronized. This ability is not supported by non-persistent FastResync.

If persistent FastResync is enabled on a volume or on a snapshot volume, a DCO and a DCO log volume are associated with the volume.

The DCO object is used not only to manage FastResync maps, but also to manage DRL recovery maps and special maps called copy maps that allow instant snapshot operations to be resume following a system crash.

Persistent FastResync can also track the association between volumes and their snapshot volumes after they are moved into different disk groups. When the disk groups are rejoined, this allows the snapshot plexes to be quickly resynchronized. This ability is not supported by non-persistent FastResync.

About disk group split and join

Disk group split and join is included with the Enterprise Edition. It is also included as part of the Veritas FlashSnap option with the Standard Edition.

VxVM provides a disk group content reorganization feature that supports general disk group reorganization and allows you to move volume snapshots to another host for off-host backup. Additional options to the `vxdg` command enable you to take advantage of the ability to remove all VxVM objects from an imported disk group and move them to a newly created target disk group (split), and to remove all VxVM objects from an imported disk group and move them to an imported target disk group (join). The move operation enables you to move a self-contained set of VxVM objects between the imported disk groups.

About hot-relocation

In addition to providing volume layouts that help improve database performance and availability, VxVM offers features that you can use to further improve system availability in the event of a disk failure. Hot-relocation is a feature that allows a system to react automatically to I/O failures on mirrored or RAID-5 volumes and restore redundancy and access to those volumes.

VxVM detects I/O failures on volumes and relocates the affected portions to disks designated as spare disks or free space within the disk group. VxVM then reconstructs the volumes that existed before the failure and makes them redundant and accessible again.

The hot-relocation feature is enabled by default and is recommended for most database configurations. After hot-relocation occurs, we recommend verifying the volume configuration for any possible performance impact. It is also a good idea to designate additional disks as spares to augment the spare pool.

While a disk is designated as a spare, you cannot use the space on that disk for the creation of VxVM objects within its disk group. VxVM also lets you free a spare disk for general use by removing it from the pool of hot-relocation disks.

About DMP-supported disk arrays

VxVM provides administrative utilities and driver support for disk arrays that can take advantage of its Dynamic Multipathing (DMP) feature. Some disk arrays provide multiple ports to access their disk devices. These ports, coupled with the host bus adaptor (HBA) controller and any data bus or I/O processor local to the array, make up multiple hardware paths to access the disk devices. Such disk arrays are called multipathed disk arrays. This type of disk array can be connected to host systems in many different configurations, (such as multiple ports connected to different controllers on a single host, chaining of the ports through a single controller on a host, or ports connected to different hosts simultaneously). DMP is available for multiported disk arrays from various vendors and provides improved reliability and performance by using path failover and load balancing.

See the *Veritas Volume Manager Administrator's Guide*.

See the *Veritas Volume Manager Hardware Notes*.

About dynamic LUN expansion

Dynamic LUN expansion allows you to resize a disk after it has been initialized while preserving the existing data on the disk.

See the *Veritas Volume Manager Administrator's Guide*.

About Storage Expert

Storage Expert consists of a set of simple commands that collect VxVM configuration data and compare it with “best practice.” Storage Expert then produces a summary report that shows which objects do not meet these criteria and makes recommendations for VxVM configuration improvements.

These user-configurable tools help you as an administrator to verify and validate systems and non-optimal configurations in both small and large VxVM installations.

Storage Expert components include a set of rule scripts and a rules engine. The rules engine runs the scripts and produces ASCII output, which is organized and archived by Storage Expert's report generator. This output contains information about areas of VxVM configuration that do not meet the set criteria. By default, output is sent to the screen, but you can redirect it to a file using standard UNIX redirection.

See the *Veritas Volume Manager Administrator's Guide*.

About cluster functionality (optional)

VxVM includes an optional, separately licensable clustering feature, known as Cluster Volume Manager, that enables VxVM to be used in a cluster environment. With the clustering option, VxVM supports up to 16 nodes per cluster.

See the *Veritas Volume Manager Administrator's Guide*.

About Veritas Volume Replicator (optional)

Veritas Volume Replicator (VVR) is an optional, separately licensable feature of VxVM. VVR is a data replication tool designed to maintain a consistent copy of application data at a remote site. It is built to contribute to an effective disaster recovery plan. If the data center is destroyed, the application data is immediately available at the remote site, and the application can be restarted at the remote site.

VVR works as a fully integrated component of VxVM. VVR benefits from the robustness, ease of use, and high performance of VxVM and, at the same time, adds replication capability to VxVM. VVR can use existing VxVM configurations with some restrictions. Any application, even with existing data, can be configured to use VVR transparently.

See the Veritas Volume Replicator documentation.

How Veritas File System works

Veritas File System (referred to as VxFS) is an extent-based, intent logging file system intended for use in UNIX environments that deal with large volumes of data and that require high file system performance, availability, and manageability. VxFS also provides enhancements that make file systems more viable in database environments.

The following sections provide a brief overview of VxFS concepts and features that are relevant to database administration.

See the *Veritas File System Administrator's Guide*.

About Veritas Quick I/O

Databases can run on either file systems or raw devices. Database administrators often create their databases on file systems because it makes common administrative tasks (such as moving, copying, and backing up) easier. However,

running databases on most file systems significantly reduces database performance.

When performance is an issue, database administrators create their databases on raw devices. VxFS with Quick I/O presents regular, preallocated files as raw character devices to the application. Using Quick I/O, you can enjoy the management advantages of databases created on file systems and achieve the same performance as databases created on raw devices.

Prior to Sybase ASE 12.0, Sybase did not recommend placing database devices on UNIX file systems (UFS for VFS) because of data integrity concerns. Writes to UFS or VFS files are buffered by the file system; therefore, Sybase dataservers would not know when an update was reflected on the physical media. Data integrity of Sybase databases using UFS or VFS files is questionable under certain system failure scenarios.

VxFS allows a regular, pre-allocated file to be accessed as a raw device using the Quick I/O interface. Using Quick I/O files not only eliminates any data integrity concerns related to running Sybase on file systems, but also improves overall performance for Sybase servers.

Beginning with Sybase ASE 12.0, placing database devices on UFS or VFS files is fully supported. Sybase uses the UNIX `O_DSYNC` flag when opening a UFS or VFS file for a database device. Writes to a UFS or VFS file opened with the `O_DSYNC`flag occur directly on the physical storage media. Sybase ASE 12.x can recover data on the UFS or VFS files in the event of a system failure. Although Sybase supports UFS or VFS files as its database devices in 12.x, using Quick I/O eliminates the potential performance problems caused by file-level locking and extra memory copying when writing to a file.

See “About Quick I/O” on page 57.

About Veritas Cached Quick I/O

Cached Quick I/O allows databases to make more efficient use of large system memory while still maintaining the performance benefits of Quick I/O. Cached Quick I/O provides an efficient, selective buffering mechanism to back asynchronous I/O. Using Cached Quick I/O, you can enjoy all the benefits of Quick I/O and achieve even better performance.

Cached Quick I/O is first enabled for the file system and then enabled on a per file basis.

See “About Cached Quick I/O” on page 79.

About Veritas Concurrent I/O

Veritas 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.

Veritas Concurrent I/O allows for concurrency between a single writer and multiple readers or between multiple writers. It minimizes serialization for extending writes and sends I/O requests directly to file systems.

See "About Concurrent I/O" on page 93.

About extent-based allocation

The UFS file system supplied with Solaris uses block-based allocation schemes that provide good random access to files and acceptable latency on small files. For larger files, such as database files, this block-based architecture limits throughput. This limitation makes the UFS file system a less than optimal choice for database environments.

When storage is allocated to a file on a VxFS file system, it is grouped in extents, as opposed to being allocated a block at a time as with the UFS file system.

By allocating disk space to files in extents, disk I/O to and from a file can be done in units of multiple blocks. This type of I/O can occur if storage is allocated in units of consecutive 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 of multiple blocks.

The VxFS file system allocates disk space to files in groups of one or more extents. VxFS also allows applications to control some aspects of the extent allocation for a given file. Extent attributes are the extent allocation policies associated with a file.

See "Preallocating space for Quick I/O files using the setext command" on page 61.

About fast file system and database recovery

Veritas File System begins recovery procedures within seconds after a system failure by using a tracking feature called intent logging. 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. 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 completing a full structural check of the

entire file system. Replaying the intent log may not completely recover the damaged file system structure if there was a disk hardware failure; hardware problems may require a complete system check using the `fsck` utility provided with Veritas File System.

About online system administration

The VxFS file system provides online system administration utilities to help resolve certain problems that impact database performance. You can defragment and resize a VxFS file system while it remains online and accessible to users.

How the defragmentation utility works

Free resources are originally aligned in the most efficient order possible and are allocated to files in a way that is considered by the system to provide optimal performance. When a file system is active for extended periods of time, new files are created, old files are removed, and existing files grow and shrink. Over time, the original ordering of free resources is lost and the file system tends to spread across the disk, leaving unused gaps or fragments between areas that are in use. This process, known as fragmentation, leads to degraded performance because the file system has fewer choices when selecting an extent (a group of contiguous data blocks) to assign to a file. You should analyze the degree of fragmentation before creating new database files.

VxFS provides the online administration utility `fsadm` to resolve fragmentation problems. The utility can be run on demand and should be scheduled regularly as a `cron` job.

How the resizing utility works

Changes in database size can result in file systems that are too large or too small for the current database. Without special utilities, expanding or shrinking a file system becomes a a matter of stopping applications, offloading the contents of the file system, rebuilding the file system to a new size, and then restoring the data. Data is unavailable to users while these administrative tasks are performed.

The VxFS file system utility `fsadm` provides a mechanism to resize file systems without unmounting them or interrupting users' productivity. Because the VxFS file system can only be mounted on one device, expanding a file system means that the underlying device must also be expandable while the file system is mounted. Working with VxVM, VxFS provides online expansion capability.

About cross-platform data sharing

Veritas Cross-Platform Data Sharing allows data to be serially shared among heterogeneous systems where each system has direct access to the physical devices that hold the data. This feature can be used only in conjunction with Veritas Volume Manager. Shared or parallel access is possible for read-only data.

See the *Veritas Storage Foundation Cross-Platform Data Sharing Administrator's Guide*.

Support for multi-volume file systems

The multi-volume file system (MVS) feature allows several volumes to be represented by a single logical object. All I/O to and from an underlying logical volume is directed by way of volume sets. A volume set is a container for multiple different volumes. This feature can be used only in conjunction with Veritas Volume Manager.

About Quality of Storage Service (optional)

The Quality of Storage Service (QoSS) feature is included with the Enterprise Edition.

The QoSS option is built on the multi-volume support technology introduced in this release. Using QoSS, you can map more than one device to a single file system. You can then configure policies that automatically relocate files from one device to another, or relocate files by running file relocation commands. Having multiple devices lets you determine where files are located, which can improve performance for applications that access specific types of files and reduce storage-related costs.

Database Dynamic Storage Tiering is built on top of QoSS and automates the migration process for Sybase database objects.

Support for large file systems and large files (optional)

Support for large file systems is included with the Enterprise Edition.

In conjunction with VxVM, VxFS can support file systems up to 8 exabytes in size.

You have the option of creating a file system using:

- Version 4 disk layout, which supports file systems up to one terabyte. The Version 4 disk layout encompasses all file system structural information in files, rather than at fixed locations on disk, allowing for greater scalability.
- Version 5, which supports file systems up to 32 terabytes. Files can be a maximum of two terabytes. File systems larger than one terabyte must be created on a Veritas Volume Manager volume.

- Version 6, which supports file systems up to 8 exabytes. The Version 6 disk layout enables features such as multi-device support, Cross-Platform Data Sharing, named data streams, file change log. File systems created on VxFS 4.1 will by default use the Version 6 disk layout. An online conversion utility, `vxupgrade`, is provided to upgrade existing disk layouts to Version 6 on mounted file systems.

For large database configurations, this eliminates the need to use multiple file systems because of the size limitations of the underlying physical devices.

Changes implemented with the VxFS Version 4 disk layout have greatly expanded file system scalability, including support for large files.

You can create or mount file systems with or without large files by specifying either the `largefiles` or `nolargefiles` option in `mkfs` or `mount` commands.

See "Creating a VxFS file system " on page 41.

About restoring file systems using Storage Checkpoints

Storage Checkpoints can be used by backup and restore applications to restore either individual files or an entire file system. Restoring from Storage Checkpoints can recover data from incorrectly modified files, but typically cannot be used to recover from hardware damage or other file system integrity problems. File restoration can be done using the `fsckpt_restore`(1M) command.

See the *Veritas File System Administrator's Guide*.

About quotas

VxFS supports quotas, which allocate per-user and per-group quotas and limit the use of two principal resources: files and data blocks. You can assign quotas for each of these resources.

Each quota consists of two limits for each resource:

- The hard limit represents an absolute limit on data blocks or files. A user can never exceed the hard limit under any circumstances.
- The soft limit is lower than the hard limit and can be exceeded for a limited amount of time. This allows users to temporarily exceed limits as long as they fall under those limits before the allotted time expires.

You can use quotas to limit the amount of file system space used by Storage Checkpoints.

See the *Veritas File System Administrator's Guide*.

About cluster functionality (optional)

File system clustering is an optional, separately licensed feature of VxFS, where one system is configured as a primary server for the file system, and the other members of a cluster are configured as secondaries. All servers access shared disks for file data operations. If the primary server fails, one of the secondary servers takes over the file system operations.

See the *Veritas File System Administrator's Guide*.

About Veritas Storage Foundation/High Availability for Sybase (optional)

Veritas Storage Foundation/High Availability (HA) (VCS) lets database administrators implement Sybase dataservers in a high availability configuration that can significantly reduce the down time of Sybase databases caused by a system hardware or software failure.

In addition to the Veritas products included in the base Veritas Storage Foundation for Sybase, Veritas Storage Foundation/HA for Sybase incorporates the following products:

- Veritas Cluster Server ™ (VCS) for Sybase
- Veritas Cluster Server™ (VCS) Enterprise Agent for Sybase

VCS can be configured to perform faster failover using the Adaptive Server Enterprise (ASE) Companion Server. The agent for this configuration is available from Sybase.

Note: Veritas Storage Foundation/HA (VCS) for Sybase is available only for the Enterprise Edition.

Chapter 2

Setting up dataservers

This chapter includes the following topics:

- Tasks for setting up new databases
- About setting up a disk group
- Creating a disk group
- Adding disks to a disk group
- About selecting a volume layout
- Creating a volume
- Creating a volume set
- Adding a volume to a volume set
- File system creation guidelines
- Creating a VxFS file system
- Mounting a file system
- Unmounting a file system
- About fragmentation
- Resizing a file system
- About Quick I/O

Tasks for setting up new databases

If you are using Veritas Storage Foundation for Sybase to set up a new database, complete these tasks in the order listed below:

Determine the number and sizes of file systems you need for the database you want to create.	See the *Veritas File System Administrator's Guide*.
Create volumes to meet your file system needs. You can use disk mirroring as a safeguard against disk failures and striping for better performance.	
Create the VxFS file systems you need on the volumes.	See "File system creation guidelines" on page 40. See "Creating a VxFS file system " on page 41.
Install and configure your database. You must create Quick I/O files before creating the tablespaces.	
If you want to use Database FlashSnap for off-host processing after converting your database files to use Quick I/O or ODM and your volume layout is inconsistent with Database FlashSnap requirements, you will need to "relayout" your volume manager configuration after your database files have been converted.	See the *Veritas Volume Manager Administrator's Guide*.
If you are using Quick I/O, convert all database files to Quick I/O files.	See "Converting Sybase files to Quick I/O files" on page 67.
If you are not currently running on VxVM and VxFS, make sure Veritas Storage Foundation for Sybase is installed and convert your existing database configuration.	See the *Veritas Storage Foundation for Sybase Installation Guide*.

About setting up a disk group

Before creating file systems for a database, set up a disk group for each Sybase dataserver.

A disk group lets you group disks, volumes, file systems, and files that are relevant to a single database into a logical collection for easy administration. Because you can move a disk group and its components as a unit from one machine to another, you can move an entire database when all the configuration objects of the database are in one disk group. This capability is useful in a failover situation.

Disk group configuration guidelines

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 Sybase ASE server.
- The disk group name must be unique. Name each disk group using the Sybase dataserver name this disk group belongs to and a `dg` suffix. The dataserver name is the name of the Sybase server as defined in the Sybase interface file. The `dg` suffix helps identify the object as a disk group. Also, each disk name must be unique within the disk group.
- Never create database devices for a dataserver using file systems or volumes that are not in the same disk group.

In earlier releases of Veritas Volume Manager, a system installed with VxVM was configured with a default disk group, `rootdg`, that had to contain at least one disk. VxVM can now function without any disk group having been configured. Only when the first disk is placed under VxVM control must a disk group be configured.

Note: Most VxVM commands require superuser or equivalent privileges.

See "About tuning VxVM " on page 147.

Creating a disk group

You can use the `vxdg` command to create a new disk group. A disk group must contain at least one disk at the time it is created. You also have the option to create a shared disk group for use in a cluster environment.

Disks must be placed in disk groups before they can be used by VxVM. You can create disk groups to organize your disks into logical sets of disks.

Before creating a disk group, make sure the following conditions have been met:

Prerequisites

- Only disks that are online and do not belong to a disk group can be used to create a disk group.
- The disk group name must be unique in the host or cluster.
- Creating a disk group requires at least one disk.

Usage notes

- Veritas Storage Foundation for Sybase only supports single disk groups.
- New disks must be placed under VxVM control and then added to a dynamic disk group before they can be used for volumes.
- When you place a disk under VxVM control, the disk is either encapsulated or initialized. Encapsulation preserves any existing data on the disk in volumes. Initialization destroys any existing data on the disk.
- If you place the root disk under VxVM control, you must encapsulate the disk. If you want to create an alternate boot disk, you can mirror the encapsulated root disk.
- For information on the `vxdg` command, see the `vxdg`(1M) manual page.

To create a new disk group

- Use the `vxdg` command as follows:

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

The following is an example of creating a disk group using the `vxdg` command:

To create a disk group named `PRODdg` on a raw disk partition `c1t1d0s2`, where the disk name `PRODdg01` references the disk within the disk group:

```
# /opt/VRTS/bin/vxdg init PRODdg PRODdg01=c1t1d0s2
```

Adding disks to a disk group

When a disk group is first created, it can contain only a single disk. You may need to add more disks to the disk group. Before adding disks, make sure the following conditions have been met:

Usage notes

- When you place a disk under VxVM control, the disk is either encapsulated or initialized. Encapsulation preserves any existing data on the disk in volumes. Initialization destroys any existing data on the disk.
- If you place the boot disk under VxVM control, you must encapsulate it. If you want to create an alternate boot disk, you can mirror the encapsulated boot disk.
- Boot disk encapsulation requires a system reboot.
- Disks cannot be added to deported disk groups.
- Disks must be under VxVM control and in a disk group before they can be used to create volumes.
- Disks must be online before they can be added to a disk group.
- Disks that already belong to a disk group cannot be added to another disk group.

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]
```

The following is an example of adding disks to a disk group using the `vxdg` command:

To add disks named `PRODdg02`, `PRODdg03`, and `PRODdg04` to the disk group `PRODdg`:

```
# /opt/VRTS/bin/vxdg -g PRODdg adddisk PRODdg02=c1t2d0s2

# /opt/VRTS/bin/vxdg -g PRODdg adddisk PRODdg03=c1t3d0s2

# /opt/VRTS/bin/vxdg -g PRODdg adddisk PRODdg04=c1t4d0s2
```

About selecting a volume layout

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

An important factor in database performance is the segment placement on the disks.

Disk I/O is one of the most important determining factors of your database's 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.

Sybase maps each physical file or raw device to its database devices. Devices are grouped into segments; tables are created on segments. When deciding which devices to put in a segment, 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 database devices are distributed across the available disks may be sufficient.

Striping also helps sequential table scan performance. When a table is created on a segment containing database devices striped across multiple disks, a high transfer bandwidth can be achieved.

See "About tuning VxVM " on page 147.

How to choose appropriate stripe unit sizes

When creating a striped volume, you need to decide the number of columns to form a striped volume and the stripe unit size. You also need to decide how to stripe the volume. You may stripe a volume across multiple disk drives on the same controller or across multiple disks on multiple controllers. By striping across multiple controllers, disk I/O can be balanced across multiple I/O channels. The decision is based on the disk and controller bandwidth and the database workload. In general, for most OLTP databases, use the default stripe unit size of 64 K or smaller for striped volumes and 16 K for RAID-5 volumes.

How to choose between mirroring and RAID-5

VxVM provides two volume configuration strategies for data redundancy: mirroring and RAID-5. Both strategies allow continuous access to data in the event of disk failure. For most database configurations, we recommend using mirrored, striped volumes. If hardware cost is a significant concern, but having higher data availability is still important, use RAID-5 volumes.

RAID-5 configurations have certain performance implications you must consider. Writes to RAID-5 volumes require parity-bit recalculation, which adds significant I/O and CPU overhead. This overhead can cause considerable performance penalties in online transaction processing (OLTP) workloads. If the database has a high read ratio, however, RAID-5 performance is similar to that of a striped volume.

Volume configuration guidelines

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 segments. Stripe multiple devices to create larger volumes if needed. Use mirroring to improve reliability. Do not use VxVM RAID-5 for transaction logs.
- When normal system availability is acceptable, put the segments on file systems created on striped volumes for most OLTP workloads.
- It is generally good practice to place frequently used databases such as `tempdb`and `sybsystemprocs`on striped devices.
- Put log segment and default segment for each database on different volumes.
- When normal system availability is acceptable, create a segment that contains database devices using Quick I/O files in file systems 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, do not stripe across too many disks or controllers. The single thread that processes sequential scans may not be able to keep up with the disk speed.
- For most workloads, use the default 64 K stripe-unit size for striped volumes and 16 K for RAID-5 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.
- When system availability is critical, use RAID-5 for read-intensive OLTP workloads to improve database performance and availability. Use RAID-5 logs 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.

See “About tuning VxVM ” on page 147.

Creating a volume

Veritas Volume Manager 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.

For databases where the data storage needs to be resilient and the data layout needs to be optimized for maximum performance, we recommend using VxVM. The striping and mirroring capabilities offered by a volume manager will help you achieve your manageability, availability, and performance goals.

After you decide on a volume layout, you can use the `vxassist` command to create the volume.

Before creating a volume, make sure the following conditions are met:

Usage notes

- Creating a volume requires a disk group name, volume name, volume size, and volume layout. You will also need to know subdisk names if you are creating a striped volume.
- Striped or mirrored volumes require at least two disks.
- Striped pro and concatenated pro volumes are mirrored by default, so a striped pro volume requires more disks than an unmirrored striped volume and a concatenated pro volume requires more disks than an unmirrored concatenated volume.
- You cannot use a striped pro or concatenated pro volume for a `root` or `swap` volume.
- A RAID-5 volume requires at least three disks. If RAID-5 logging is enabled, a RAID-5 volume requires at least four disks. RAID-5 mirroring is not supported.

To create a volume

- Use the `vxassist` command as follows:

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

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

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

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

Creating a volume set

Volume Sets enable the use of the Multi-Volume Support feature with Veritas File System (VxFS). It is also possible to use the Veritas Enterprise Administrator (VEA) to create and administer volumes sets. For more information, see the VEA online help.

Before creating a volume set, make sure the following conditions are met:

Usage notes

- Before creating a volume set, you must have at least one volume created.
  See "Creating a volume " on page 37.
- A maximum of 256 volumes may be configured in a volume set.
- Only Veritas File System is supported on a volume set.
- The first volume in a volume set must be larger than 20MB.
- Raw I/O from and to a volume set is not supported.
- Volume sets can be used instead of volumes with the following `vxsnap` operations on instant snapshots: `addmir`, `dis`, `make`, `prepare`, `reattach`, `refresh`, `restore`, `rmmir`, `split`, `syncpause`, `syncresume`, `syncstart`, `syncstop`, `syncwait`, and `unprepare`. The third-mirror break-off usage model for full-sized instant snapshots is supported for volume sets provided that sufficient plexes exist for each volume in the volume set. See the *Veritas Volume Manager Administrator's Guide*.
- Most VxVM commands require superuser or equivalent privileges.
- For details regarding usage of the `vxvset` command, see the `vxvset(1M)` manual page.

To create a volume set for use by Veritas file system (VxFS)

- Use the following command:

  ```
  # /opt/VRTS/bin/vxvset [-g diskgroup] -t vxfs make volset volume
  ```

  where:

  - `volset` is the name of the volume set
  - `volume` is the name of the first volume in the volume set
  - `-t` defines the content handler subdirectory for the application that is to be used with the volume. This subdirectory contains utilities that an application uses to operate on the volume set. The operation of these utilities is determined by the requirements of the application and not by VxVM.

For example, to create a volume set named `db01vset` that contains the volume `db01`, in the disk group `PRODdg`, you would use the following command:

```
# /opt/VRTS/bin/vxvset -g PRODdg -t vxfs make db01vset db01
```

Adding a volume to a volume set

After creating a volume set, you can add additional volumes.

To add a volume to a volume set

- Use the `vxvset` command as follows:

```
# /opt/VRTS/bin/vxvset [-g diskgroup] [-f] addvol volset \
volume
```

Warning: The `-f` (force) option must be specified if the volume being added, or any volume in the volume set, is either a snapshot or the parent of a snapshot. Using this option can potentially cause inconsistencies in a snapshot hierarchy if any of the volumes involved in the operation is already in a snapshot chain.

See the *Veritas Volume Manager Administrator's Guide*.

For example, to add the volume `db02`, to the volume set `db01vset`, use the following command:

```
# /opt/VRTS/bin/vxvset -g PRODdg addvol db01vset db02
```

File system creation guidelines

Follow these guidelines when creating VxFS file systems:

- Specify the maximum block size and log size when creating file systems for databases.
- Except for specifying the maximum log size and support for large files as required, use the VxFS defaults when creating file systems for databases.
- Never disable the intent logging feature of the file system.
- For log segments in user databases, use database devices created on file systems with simple (and mirrored, if necessary) volumes. Put the other database devices on file systems created on striped, striped and mirrored, or mirrored and striped volumes.
- 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.
- If Quick I/O is supported, select `vxfs` as the file system type to take advantage of the Quick I/O feature, online administration, fast recovery of the VxFS file system, and superior reliability features.

Creating a VxFS file system

Always specify `vxfs` as the file system type to take advantage of Quick I/O, Storage Rollback, online administration, fast recovery of the VxFS file system, and superior reliability features.

You can create a file system on a volume, as long as the volume is large enough to accommodate the file system. We recommend creating a VxFS file system and using Quick I/O files as your database devices. Although Sybase ASE 12.0 or later supports regular UNIX files, using Quick I/O provides better performance.

Note: In earlier Sybase ASE releases, Sybase does not guarantee data integrity if you use UNIX files as database devices. Use Quick I/O files to guarantee data integrity.

Before creating a file system, see the following notes:

Usage notes

- See the `mkfs`(1M) and `mkfs_vxfs`(1M) manual pages for more information about the options and variables available for use with the `mkfs` command.
- See the `mount`(1M) and `mount_vxfs`(1M) manual pages for more information about mount settings.

To create a VxFS file system on an existing volume

- Use the `mkfs` command as follows:

```
# /usr/sbin/mkfs -F 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 VxVM volume on which to create the file system (for example, `/dev/vx/rdsk/PRODdg/db01`)
- `size` is the size of the new file system (optional)

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

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 -F vxfs -o largefiles,bsize=8192,\
logsize=2000 /dev/vx/rdsk/PRODdg/db01
```

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

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.

The `mkfs` command displays output similar to the following:

```
version 6 layout

20480 sectors, 10240 blocks of size 1024, log size 1024 blocks
```

You can now mount the newly created file system.

See "Mounting a file system " on page 43.

Large file system and large file support

In conjunction with VxVM, VxFS can support file systems up to 8 exabytes in size. For large database configurations, this eliminates the need to use multiple file systems because of the size limitations of the underlying physical devices.

Changes implemented with the VxFS Version 6 disk layout have greatly expanded file system scalability, including support for large files. You can create or mount file systems with or without large files by specifying either the `largefiles` or `nolargefiles` option in `mkfs` or `mount` commands. If you specify the `nolargefiles` option, a file system cannot contain files 2 GB or larger.

Before creating a VxFS file system, make sure the following conditions are met:

Usage notes

- See the `mount_vxfs` (1M) and `mkfs_vxfs` (1M) manual pages for detailed information on mounting and creating file systems.
- See the `fsadm_vxfs` (1M) manual pages for detailed information about large files.

To enable large files on a file system that was created without the largefiles option

- Use the `fsadm` command as follows:

```
# /opt/VRTS/bin/fsadm -F vxfs -o largefiles \
/mount_point
```

Note: Make sure the applications and tools you use can handle large files before enabling the large file capability. Applications and system administration utilities can experience problems if they are not large file aware.

Multi-volume support

The multi-volume support feature enabled by VxFS Version 6 disk layout allows several volumes to be represented by a single logical object, known as a volume set. The `vxvset` command can be used to create and administer volume sets in Veritas Volume Manager.

VxFS's multi-volume support feature can be used with volume sets. There are two VxFS commands associated with multi-volume support:

- `fsapadm` - VxFS allocation policy administration utility
- `fsvoladm` - VxFS device administration utility

See the *Veritas File System Administrator's Guide*.

Mounting a file system

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

By default, the command tries to enable Quick I/O. If Quick I/O is not installed or licensed, no error messages are displayed unless you explicitly specify the mount option. If necessary, you can turn the Quick I/O option off at mount time or you can remount the file system with the option.

Before mounting a file system, make sure the following conditions are met:

Prerequisites
- A file system must exist in order to be mounted.
- DBAs should log in as the Sybase DBA user.

Usage notes

- The mount point must be an absolute path name (that is, it must begin with /).
- See the mount_vxfs (1M) manual page for more information about mount settings.
- See the mount (1M) manual page for more information about generic mount options.

To mount a file system

- Use the mount command as follows:

```
# /usr/sbin/mount -F vxfs [generic_options] [-r] \
[-o specific_options] special /mount_point
```

where:

- generic_options are the options common to most file systems
- -r mounts the file system as read only
- specific_options are options specific to the VxFS file system
- 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:

```
# mkdir /db01

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

If you would like /db01 to be mounted automatically after rebooting, add an entry for it in /etc/fstab as follows:

```
/dev/vx/dsk/PRODdg/db01  /db01       vxfs  largefiles,qio    0 2
```

If you do not need to use Quick I/O files, set noqio instead of qio as one of the options.

Unmounting a file system

If you no longer need to access the data in a file system, you can unmount the file system using the umount command.

Before unmounting a file system, make sure the following conditions have been met:

Prerequisites	■ A file system must exist and be mounted in order to be unmounted.
Usage notes	■ You cannot unmount a file system that is in use. See the `umount` (1M) manual page for more information on mounting file systems.

To unmount a file system

1 Use the `fuser` command to make sure that the file system is not being used:

```
# fuser -c /mount_point
```

where the `-c` option provides information on file system mount points and any files within mounted file systems.

If the file system is being used and you need to unmount it, use the `fuser -ck` command. See the `fuser`(1M) man page for more information.

2 Unmount the file system using one of the `umount` command options:

- `umount special`
- `umount /mount_point`
- `umount -f /mount_point`

where:

- `special` is a block special device
- `/mount_point` is the location where the file system is mounted
- `-f` forcibly unmounts the mount point

The following is an example of unmounting a file system:

To verify that the file system `/db01` is not in use and then unmount the file system:

```
# fuser -c /db01
/db01:
# umount /db01
```

About fragmentation

When free resources are initially allocated to files in a Veritas file system, they are aligned in the most efficient order possible to provide optimal performance. On an active file system, the original order is lost over time as files are created, removed, or resized. As space is allocated and deallocated from files, the available

free space becomes broken into fragments. This means that space must be assigned to files in smaller and smaller extents. This process is known as fragmentation. Fragmentation leads to degraded performance and availability. The degree of fragmentation depends on file system usage and activity.

How to control fragmentation

It is very rare to have a badly fragmented VxFS file system in an ASE environment. However, fragmentation can occur when many database devices are created and deleted.

VxFS provides online reporting and optimization utilities to enable you to monitor and defragment a mounted file system. These utilities are accessible through the file system administration command, `fsadm`. Using the `fsadm` command, you can track and eliminate fragmentation without interrupting user access to the file system.

Types of fragmentation

VxFS addresses two types of fragmentation:

- Directory Fragmentation
  As files are created and removed, gaps are left in directory inodes. This is known as directory fragmentation. Directory fragmentation causes directory lookups to become slower.
- Extent Fragmentation
  As files are created and removed, the free extent map for an allocation unit changes from having one large free area to having many smaller free areas. Extent fragmentation occurs when files cannot be allocated in contiguous chunks and more extents must be referenced to access a file. In a case of extreme fragmentation, a file system may have free space that cannot be allocated.

How to monitor fragmentation

You can monitor fragmentation in VxFS by running reports that describe fragmentation levels. Use the `fsadm` command to run reports on directory fragmentation and extent fragmentation. The `df` command, which reports on file system free space, also provides information useful in monitoring fragmentation.

Use the following commands to report fragmentation information:

- `fsadm -D`, which reports on directory fragmentation.
- `fsadm -E`, which reports on extent fragmentation.

- `/opt/VRTS/bin/fsadm [-F vxfs] -o s`, which prints the number of free extents of each size.

Defragmenting a file system

You can use the online administration utility `fsadm` to defragment or reorganize file system directories and extents.

The fsadm utility defragments a file system mounted for read/write access by:

- Removing unused space from directories.
- Making all small files contiguous.
- Consolidating free blocks for file system.

The following options are for use with the `fsadm` utility:

`-d`	Reorganizes directories. Directory entries are reordered to place subdirectory entries first, then all other entries in decreasing order of time of last access. The directory is also compacted to remove free space. **Note:** If you specify `-d` and `-e`, directory reorganization is always completed first.
`-a`	Use in conjunction with the `-d` option to consider files not accessed within the specified number of days as “aged” files. Aged files are moved to the end of the directory. The default is 14 days.
`-e`	Reorganizes extents. Files are reorganized to have the minimum number of extents. **Note:** If you specify `-d` and `-e`, directory reorganization is always completed first.
`-D -E`	Produces reports on directory and extent fragmentation, respectively. **Note:** If you use both `-D` and `-E` with the `-d` and `-e` options, the fragmentation reports are produced both before and after reorganization.
`-v`	Specifies verbose mode and reports reorganization activity.
`-l`	Specifies the size of a file that is considered large. The default is 64 blocks.

`-t`	Specifies a maximum length of time to run, in seconds. **Note:** The `-t` and `-p` options control the amount of work performed by fsadm, either in a specified time or by a number of passes. By default, `fsadm` runs five passes. If both `-t` and `-p` are specified, fsadm exits if either of the terminating conditions are reached.
`-p`	Specifies a maximum number of passes to run. The default is five. **Note:** The `-t` and `-p` options control the amount of work performed by fsadm, either in a specified time or by a number of passes. By default, `fsadm` runs five passes. If both `-t` and `-p` are specified, fsadm exits if either of the terminating conditions are reached.
`-s`	Prints a summary of activity at the end of each pass.
`-r`	Specifies the pathname of the raw device to read to determine file layout and fragmentation. This option is used when fsadm cannot determine the raw device.

Note: You must have superuser (`root`) privileges to reorganize a file system using the `fsadm` command.

To defragment a file system

- Run the `fsadm` command followed by the options specifying the type and amount of defragmentation. Complete the command by specifying the mount point or raw device to identify the file system.

  ```
  # /opt/VRTS/bin/fsadm [-d] [-D] [-e] [-E] [-s] [-v] \

  [-l largesize] [-a days] [-t time] [-p pass_number] \
  [-r rawdev_path] mount_point
  ```

  Refer to the *Veritas File System Administrator's Guide* for instructions and information on scheduling defragmentation. Veritas File System Administrator's Guide for instructions and information on scheduling defragmentation.

  For example, to defragment a file system:

  ```
  # /opt/VRTS/bin/fsadm -d -D /sybdata_qiovm

  Directory Fragmentation Report
               Dirs        Total      Immed    Immeds   Dirs to   Blocks
               Searched    Blocks     Dirs     to Add   Reduce    Reduce
    total             5         1         4         0         0 0
    Directory Fragmentation Report
               Dirs        Total      Immed    Immeds   Dirs to   Blocks
               Searched    Blocks     Dirs     to Add   Reduce    Reduce
    total             5         1         4         0         0 0
  ```

Resizing a file system

If you need to extend or shrink a VxFS file system, you can use the `fsadm` command.

If a VxFS file system requires more space, you can use this procedure to extend the size of the file system. If a VxFS File System is too large and you need the space elsewhere, you can use this procedure to shrink the file system.

Remember to increase the size of the underlying device or volume before increasing the size of the file system.

See the *Veritas Volume Manager Administrator's Guide*.Veritas Volume Manager Administrator's Guide.

Before resizing a file system, the following conditions must be met:

Prerequisites	■ This task requires a mounted file system. You must know either the desired size or the amount of space to add to or subtract from the file system size.
Usage notes	■ See the `format`(1M) manual page. See the `fsadm_vxfs`(1M) manual page.

To resize a file system

◆ Use `fsadm` command as follows:

```
# /opt/VRTS/bin/fsadm -F vxfs [-b newsize] \
  [-r rawdev] /mount_point
```

where:

- newsize is the size (in sectors) to which the file system will increase or shrink
- rawdev specifies the name of the raw device if there is no entry in `/etc/vfstab` and `fsadm` cannot determine the raw device
- /mount_point is the location where the file system is mounted

For example, to extend the file system `/db01` to 2 GB:

```
# /opt/VRTS/bin/fsadm -F vxfs -b 2g /db01
```

Note: See the *Veritas File System Administrator's Guide* and `fsadm_vxfs` (1M) manual page for information on how to perform common file system tasks using `fsadm`.

Resizing a file system and the underlying volume

The `fsadm` command resizes the file system only. If you attempt to use `fsadm` to make the file system the same size or larger than the underlying volume, the `fsadm` command will fail. To resize the file system and its underlying volume, use the `vxresize` command instead.

Warning: Resizing a volume with a usage type other than FSGEN or RAID5 can result in data loss. If such an operation is required, use the `-f` option to forcibly resize such a volume.

Before resizing a file system and the underlying volume, the following conditions must be met:

Prerequisites
- You must know the new desired size of the file system.

Usage notes
- `vxresize` works with VxFS and UFS file systems only.
- If the file system is mounted and VxFS, you can grow or shrink the size. If a VxFS file system is unmounted, you cannot grow or shrink the size.
- If the file system is mounted and UFS, you can grow the size only. If the file system is unmounted and UFS, you can grow size only.
- When resizing large volumes, `vxresize` may take a long time to complete.
- Resizing a volume with a usage type other than FSGEN or RAID5 can result in data loss. If such an operation is required, use the `-f` option to forcibly resize such a volume.
- You cannot resize a volume that contains plexes with different layout types.
- See the `vxresize` (1M) manual page for more details.

To resize a file system and the underlying volume

- Use the `vxresize` command as follows:

```
# /etc/vx/bin/vxresize -g disk_group -b -F vxfs -t \
homevolresize homevol volume_size disk_name disk_name
```

For example, to extend a 1-gigabyte volume, `homevol`, that contains a VxFS file system, to 10 gigabytes using the spare disks `disk10` and `disk11`, enter:

```
# /etc/vx/bin/vxresize -b -F vxfs -t homevolresize homevol 10g \
disk10 disk11
```

The `-b` option specifies that this operation runs in the background. Its progress can be monitored by specifying the task tag `homevolresize` to the `vxtask` command.

About Quick I/O

Veritas Quick I/O is a VxFS feature included in Veritas Storage Foundation for Sybase that lets 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.

How Quick I/O works

Veritas Quick I/O supports direct I/O and kernel asynchronous I/O and allows dataservers to access regular files on a VxFS file system as raw character devices.

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 dataservers.

How Quick I/O improves database performance

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

- Supporting kernel asynchronous I/O
- Supporting direct I/O
- Avoiding kernel write locks on database files
- Avoiding double buffering

Supporting kernel asynchronous I/O

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 Solaris provide kernel 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.

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.

Avoiding kernel write locks

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.

About Quick I/O requirements

To use Quick I/O, you must:

- Preallocate files on a VxFS file system
- Use a special file naming convention to access the files

Preallocating files

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.

About Quick I/O naming conventions

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.

How to set up Quick I/O

Quick I/O is included in the VxFS package shipped with Veritas Storage Foundation for Sybase. 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 the Veritas Storage Foundation for Sybase 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
```

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:

- You can use the `qiomkfile` command to preallocate space for database files and make them accessible to the Quick I/O interface.
  See "Creating database files as Quick I/O files using qiomkfile" on page 62.
- 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 61.

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 65.
- You can convert your existing Sybase database files to use the Quick I/O interface using the `qio_getdbfiles` and `qio_convertdbfiles` commands.
  See “Converting Sybase files to Quick I/O files” on page 67.

Chapter 3

Using Veritas Quick I/O

This chapter includes the following topics:

- About Quick I/O
- Preallocating space for Quick I/O files using the setext command
- Creating database files as Quick I/O files using qiomkfile
- Accessing regular VxFS files as Quick I/O files
- Converting Sybase files to Quick I/O files
- Displaying Quick I/O status and file attributes
- Extending a Quick I/O file
- 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 Storage Foundation for Sybase that lets 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.

How Quick I/O works

Veritas Quick I/O supports direct I/O and kernel asynchronous I/O and allows dataservers to access regular files on a VxFS file system as raw character devices.

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 dataservers.

How Quick I/O improves database performance

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

- Supporting kernel asynchronous I/O
- Supporting direct I/O
- Avoiding kernel write locks on database files
- Avoiding double buffering

Supporting kernel asynchronous I/O

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 Solaris provide kernel 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.

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.

Avoiding kernel write locks

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.

About Quick I/O requirements

To use Quick I/O, you must:

- Preallocate files on a VxFS file system
- Use a special file naming convention to access the files

Preallocating files

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.

About Quick I/O naming conventions

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.

How to set up Quick I/O

Quick I/O is included in the VxFS package shipped with Veritas Storage Foundation for Sybase. 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 the Veritas Storage Foundation for Sybase 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
```

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:

- You can use the `qiomkfile` command to preallocate space for database files and make them accessible to the Quick I/O interface.
  See "Creating database files as Quick I/O files using qiomkfile" on page 62.
- 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 61.

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 65.

- You can convert your existing Sybase database files to use the Quick I/O interface using the `qio_getdbfiles` and `qio_convertdbfiles` commands.
  See "Converting Sybase files to Quick I/O files" on page 67.

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:

```
# chown sybase:sybase .filename

# chmod 660 .filename
```

An example to show how to access the mount point `/db01`, create a datafile, 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

# chown sybase:sybase .dbfile

# chmod 660 .dbfile
```

Creating database files as Quick I/O files using qiomkfile

The best way to preallocate space for tablespace containers and to make them accessible using the Quick I/O interface is to use the `qiomkfile`. You can use the `qiomkfile` to create the Quick I/O files for either temprory or permanent tablespaces.

Prerequisites

- You can create Quick I/O files only on VxFS file systems.
- If you are creating device files on an existing file system, run `fsadm` (or similar utility) to report and eliminate fragmentation.
- You must have read/write permissions on the directory in which you intend to create Sybase Quick I/O files.

Usage notes

- The `qiomkfile` command creates two files: a regular file with preallocated, contiguous space, and a file that is a symbolic link pointing to the Quick I/O name extension.
- See the `qiomkfile`(1M) manual page for more information.

`-a`	Creates a symbolic link with an absolute path name for a specified file. Use the `-a` option when absolute path names are required. However, the default is to create a symbolic link with a relative path name.
`-e`	Extends a file by a specified amount to allow Sybase tablespace resizing. See "Extending a Quick I/O file" on page 75.
`-r`	Increases the file to a specified size to allow Sybase tablespace resizing. See "Extending a Quick I/O file" on page 75.
`-s`	Specifies the space to preallocate for a file in bytes, kilobytes, megabytes, gigabytes, or sectors (512 bytes) by adding a `k`, `K`, `m`, `M`, `g`, `G`, `s`, or `S` suffix. The default is bytes—you do not need to attach a suffix to specify the value in bytes. The size of the file that is preallocated is the total size of the file (including the header) rounded to the nearest multiple of the file system block size.

Warning: Exercise caution when using absolute path names. Extra steps may be required during database backup and restore procedures to preserve symbolic links. If you restore files to directories different from the original paths, you must change the symbolic links that use absolute path names to point to the new path names before the database is restarted.

To create a database file as a Quick I/O file using qiomkfile

1 Create a database file using the `qiomkfile` command:

```
$ /opt/VRTS/bin/qiomkfile -s file_size /mount_point/filename
```

2 Add a device to the Sybase dataserver device pool for the Quick I/O file using the `disk init` command:

```
$ isql -Usa -Psa_password -Sdataserver_name
> disk init
> name="device_name",
> physname="/mount_point/filename",
> vdevno="device_number",
> size=51200
> go
> alter database production on new_device=file_size
> go
```

The size is in 2K units. The Enterprise Reference manual contains more information on the `disk init` command.

See the *Sybase Adaptive Server Enterprise Reference Manual.*

3 Use the file to create a new segment or add to an existing segment.

To add a new segment:

```
$ isql -Usa -Psa_password -Sdataserver_name
> sp_addsegment new_segment, db_name, device_name
> go
```

To extend a segment:

```
$ isql -Usa -Psa_password -Sdataserver_name
> sp_extendsegment segment_name, db_name, device_name
> go
```

See the *Sybase Adaptive Server Enterprise Reference Manual.*

An example to show how to create a 100MB database file named `dbfile` on the VxFS file system `/db01` using a relative path name:

```
$ /opt/VRTS/bin/qiomkfile -s 100m /db01/dbfile
$ ls -al
-rw-r--r--   1 sybase   sybase   104857600   Oct 2 13:42    .dbfile
lrwxrwxrwx   1 sybase   sybase         19     Oct 2 13:42    dbfile -> \
.dbfile::cdev:vxfs:
```

In the example, `qiomkfile` creates a regular file named `/db01/.dbfile`, which has the real space allocated. Then, `qiomkfile` creates a symbolic link named `/db01/dbfile`. The symbolic link is a relative link to the Quick I/O interface for `/db01/.dbfile`, that is, to the `.dbfile::cdev:vxfs:` file. The symbolic link allows `.dbfile` to be accessed by any database or application using its Quick I/O interface.

The device size is a multiple of 2K pages. In the example, 51200 times 2K pages is 104857600 bytes. The `qiomkfile` command must use this size.

An example to show how to add a 100MB Quick I/O file named `dbfile` to the list of devices used by database `production`, using the `disk init` command:

```
$ isql -Usa -Psa_password -Sdataserver_name
> disk init
> name="new_device",
> physname="/db01/dbfile",
> vdevno="device_number",
> size=51200
> go
> alter database production on new_device=100
> go
```

An example to show how to create a new segment, named `segment2`, for device `dbfile` on database `production`:

```
$ isql -Usa_password -Sdataserver_name
> sp_addsegment segment2, production, dbfile
> go
```

An example to show how to extend a segment, named `segment1`, for device `dbfile` on database `production`:

```
$ isql -Usa_password -Sdataserver_name
> sp_extendsegment segment1, production, dbfile
> go
```

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).

Note: Sybase requires special prerequisites.

See "Converting Sybase files to Quick I/O files" on page 67.

Usage notes

- When 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

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

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

Converting Sybase files to Quick I/O files

Special commands are provided to assist you in identifying and converting an existing database to use Quick I/O. Use the `qio_getdbfiles` and `qio_convertdbfiles` commands to first extract and then convert Sybase dataserver files to Quick I/O files.

Prerequisites

- Log in as the Database Administrator (typically, the user ID `sybase`) to run the `qio_getdbfiles` and `qio_convertdbfiles` commands.
- Files you want to convert must be regular VxFS files created by Sybase ASE 12.0 or later and have the Sybase `dsync` flag on. Regular VxFS files created without the `dsync` flag on could be sparse. Sparse files should not be converted to Quick I/O files.
- Sybase ASE Server must be installed and running.
- The conversion commands were developed to support localization, so you need to set the `NLSPATH` to obtain the proper message catalog.
- For Sybase ASE 12.0, the `$SYBASE`environment variable must be set to the Sybase home directory
  For Sybase ASE 12.0, the `$DSQUERY` environment variable must be set to the server on which you intend to run the `qio_getdbfiles` command
  For Sybase ASE 12.0, the `$PATH` environment variable must include `$SYBASE/ASE-12_0/bin` and `$SYBASE/OCS-12_0/bin`
  For Sybase ASE 12.0, the`$LD_LIBRARY_PATH` environment variable must include `$SYBASE/OCS-12_0/lib`
  For Sybase ASE 12.5, you must set the following Sybase environment variables:
  `$SYBASE` must be set to the Sybase home directory
  `$DSQUERY` must be set to the server on which you intend to run the `qio_getdbfiles` command
  `$PATH` must include `$SYBASE/ASE-12_5/bin` and `$SYBASE/OCS-12_5/bin`
  `$LD_LIBRARY_PATH` must include `$SYBASE/OCS-12_5/lib`
  Only English is supported in the current release.

For the `qio_getdbfiles` command:

`-d` Lets you specify a specific database name from which to extract a list of dataserver files. If you do not specify the `-d` option, the list of files comes from all databases of the Sybase server. You cannot use this option in conjunction with the `-m` option.

`-m` Lets you specify a master device path. A master device does not have a corresponding physical path name in Sybase's database catalog, but rather has a `d_master` string. When you start an ASE server, you must pass in the full path name of master device. The `qio_getdbfiles` command first attempts to get the master device path for the Sybase in the `RUN_`<server_name> file in the `$SYBASE/ASE-12_0/install` directory for Sybase ASE 12.0 or `$SYBASE/ASE-12_5/install` directory for Sybase ASE 12.5 automatically. However, if you do not have a standard `RUN_`<server_name> file, you can use `-m` flag to pass in the master device path. You cannot use this option in conjunction with the `-d` option.

`-T` Lets you specify the type of database as `syb`. 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`, and `$DSQUERY`, are present on a server).

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.

`-f` Reports on the current fragmentation levels for database files listed in the mkqio.dat file. Fragmentation is reported as not fragmented, slightly fragmented, fragmented, highly fragmented. You must be superuser (`root`) to use this option.

`-h` Displays a help message.

`-T` Lets you specify the type of database as `syb`. 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`, and `$DSQUERY`, 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.

- The `qio_getdbfiles` and `qio_convertdbfiles` commands access the Sybase ASE Server to obtain information. You need to do one of the following to connect to the Sybase ASE Server:
- Supply the Sybase sa password when prompted. (This is the default behavior.)
- Create a file called `sa_password_<dataserver_name>` (where <dataserver_name> is server defined in the `$DSQUERY` environment variable)

in the /opt/VRTSsybed/.private directory that contains the sa password. The .private directory must be owned by the Sybase database administrator user (typically sybase) and carry a file permission mode of 700. The sa_password_<dataserver_name> file must also be owned by the Sybase database administrator user, and carry a file permission mode of 600. Once the sa_password_<dataserver_name> file is correctly set up, users will not be prompted for the sa password, and the convert commands will run in non-interactive mode.

For the qio_getdbfiles command:

- You can use the qio_getdbfiles command to generate lists of files from one or more databases.

For the qio_convertdbfiles command:

- To use the qio_convertdbfiles command with the -f option (the option that lets you report fragmentation), you must be superuser (root).
- Converting existing database files to Quick I/O files may not be the optimal thing to do if these files are fragmented. Use the -f option to determine the fragmentation levels and either:
  - Exclude files that are highly fragmented and do not have sufficient contiguous extents for Quick I/O use.
  - Create new files with the qiomkfile command, rather than converting the files using the qio_convertdbfiles command. The new files will be contiguous. You can then move data from the old files to the new files using the dd(1M) command or a database import facility and the new files defined to the database.

To set the Sybase environment variables and NLSPATH

- Set the required Sybase environment and message catalog variables, as follows:

```
$ SYBASE=/home_directory; export SYBASE
   $ DSQUERY=servername; export DSQUERY
   $ PATH=$SYBASE/ASE-12_5/bin:$SYBASE/OCS-12_5/bin:$PATH; \
      export PATH
   $ LD_LIBRARY_PATH=$SYBASE/OCS-12_5/lib; export LD_LIBRARY_PATH
   $ NLSPATH=/usr/lib/locale/%L/%N:$NLSPATH; export NLSPATH
```

To determine if the Sybase server is up and running

1 Access the `install` directory:

```
$ cd $SYBASE/ASE-12_5/install
```

2 Use the `showserver` and `grep` commands to determine if the Sybase server is running:

```
$ ./showserver | grep servername
```

If the output of these commands displays the server name, the server is running. If the no output is displayed, the server is not running.

To extract a list of Sybase files to convert

1 Supply the sa password when prompted, or create a file called `sa_password_<dataserver_name>` in the `/opt/VRTSsybed/.private` directory that contains the sa password.

2 With the Sybase dataserver up and running, run the `qio_getdbfiles` command without the `-d` option (to extract a list of dataserver files on all databases) from a directory for which you have write permission:

```
$ cd /extract_directory
$ /opt/VRTSsybed/bin/qio_getdbfiles
```

or

With the database instance up and running, run the `qio_getdbfiles` command with the `-d` option (to extract a list of dataserver files on a specific database) from a directory for which you have write permission:

```
$ cd /extract_directory
   $ /opt/VRTSsybed/bin/qio_getdbfiles -d database_name
```

The `qio_getdbfiles` command extracts the list of dataserver files and stores the file names in a file called `mkqio.dat`.

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

To convert the Sybase files to Quick I/O files

1 Shut down the Sybase dataserver.

Caution: 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 Supply the sa password when prompted, or create a file called `sa_password_<dataserver_name>` in the `/opt/VRTSsybed/.private` directory that contains the sa password.

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

```
$ cd /extract_directory
   $ /opt/VRTSsybed/bin/qio_convertdbfiles
```

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

```
  /sybdev/L001/SYS/master.dat
104857600
              /sybdev/L001/USER/qiofile1
209715200
              /sybdev/L001/USER/qiofile2
209715200
              /sybdev/L001/USER/qiofile3
209715200
              /sybdev/L001/USER/qiofile4
209715200
              /sybdev/L001/SYS/sysporcs.dat
83886080
```

The `qio_convertdbfiles` command (with no options specified) renames 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` command and prints an error message if any of the dataserver 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.

4 Start up the database.

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

Examples

- To prepare for and convert Sybase ASE 12.5 dataserver files to Quick I/O files:

```
$ SYBASE=/sybase; export SYBASE
$ DSQUERY=L001; export DSQUERY
$ PATH=$SYBASE/ASE-12_5/bin:$SYBASE/OCS-12_5/bin:$PATH; \
   export PATH
$ LD_LIBRARY_PATH=$SYBASE/OCS-12_5/lib; export LD_LIBRARY_PATH
$ NLSPATH=/usr/lib/locale/%L/%N:$NLSPATH; export NLSPATH
$ cd /sybase/ASE-12_5/install
$ ./showserver | grep L001
$ /opt/VRTSsybed/bin/qio_getdbfiles

Check whether Sybase server L001 is up running...

Attempt to Connect to Server L001...
Enter Sybase SA password for Server L001
Password: pel93
```

You are once again prompted for the `sa` password.

```
Retrieving Database Device Information from L001        Enter Sybase SA
```

- To view the contents of the `mkqio.dat` list file:

```
             $ cat mkqio.dat

            /sybdev/L001/SYS/master.dat
104857600
            /sybdev/L001/USER/qiofile1
209715200
            /sybdev/L001/USER/qiofile2
209715200
            /sybdev/L001/USER/qiofile3
209715200
            /sybdev/L001/USER/qiofile4
209715200
            /sybdev/L001/SYS/sysporcs.dat
83886080
```

- To convert the database files listed in the `mkqio.dat` file to Quick I/O files, shut down the database and enter:

```
$ /opt/VRTSsybed/bin/qio_convertdbfiles

Check whether Sybase server L001 is up running...

Attempt to Connect to Server L001...
Enter Sybase SA password for Server L001:
Password: pel93
CT-LIBRARY error:
 ct_connect(): network packet layer: internal net library error:
 Net-Lib protocol driver call to connect two endpoints failed
```

Note: This error message is displayed because the dataserver is shutdown. This is the correct and expected behavior.

```
master.dat --> .master.dat::cdev:vxfs:

qiofile1 --> .qiofile1::cdev:vxfs:

qiofile2 --> .qiofile2::cdev:vxfs:

qiofile3 --> .qiofile3::cdev:vxfs:

qiofile4 --> .qiofile4::cdev:vxfs:

sysporcs.dat --> .sysporcs.dat::cdev:vxfs:
```

- To undo the previous run of `qio_convertdbfiles`, changing Quick I/O files back to regular VxFS files:

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

.master.dat::cdev:vxfs: --> master.dat

.qiofile1::cdev:vxfs: --> qiofile1

.qiofile2::cdev:vxfs: --> qiofile2

.qiofile3::cdev:vxfs: --> qiofile3

.qiofile4::cdev:vxfs: --> qiofile4

.sysporcs.dat::cdev:vxfs: --> sysporcs.dat
```

Note: If the server is up and running, you will receive an error message stating that you need to shut down before you can run the `qio_convertdbfiles` command.

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*

-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 a 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*

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.
- 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 Sybase resizing.
`-r`	Increases the file to a specified size to allow Sybase 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:

   ```
   # /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 `dbfile` Quick I/O file by 20MB:

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

   $ /opt/VRTS/bin/qiomkfile -e 20M /db01/dbfile
   ```

   An example to show how to grow VxFS file system `/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
   ```

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:

Prerequisites

- Recover your database before attempting to recreate the Quick I/O files.
- Log in as the Database Administrator (typically, the user ID sybase) 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.
- The SYBASE and DSQUERY environment variables must be set.

See "Converting Sybase files to Quick I/O files" on page 67.

Usage notes

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

To recreate Quick I/O files after recovering a database

- Use the `qio_recreate` command as follows:

```
# /opt/VRTSsybed/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.

If...	Then...
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.

To disable Quick I/O

1. If the database is running, shut it down.
2. To change Quick I/O files back to regular VxFS files, run the following command from the directory containing the `mkqio.dat` list:

   ```
   $ /opt/VRTSsybed/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`) renames the files from .filename to filename and removes the symbolic link to .filename that was created along with the Quick I/O files.
3. To remount the file system with Quick I/O disabled, use the `mount -o noqio` command as follows:

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

Chapter 4

Using Veritas Cached Quick I/O

This chapter includes the following topics:

- About Cached Quick I/O
- Enabling Cached Quick I/O on a file system
- Determining candidates for Cached Quick I/O
- Enabling and disabling Cached Quick I/O for individual files

About Cached Quick I/O

Veritas Cached Quick I/O maintains and extends the database performance benefits of Veritas Quick I/O by making more efficient use of large, unused system memory through a selective buffering mechanism. Cached Quick I/O also supports features that support buffering behavior, such as file system read-ahead.

How Cached Quick I/O works

Cached Quick I/O is a specialized external caching mechanism specifically suitable to 32-bit ports of the Sybase server. Cached Quick I/O can be used on 64-bit ports of the Sybase server, but the benefits are not as great. Cached Quick I/O can be selectively applied to datafiles that are suffering an undesirable amount of physical disk I/O due to insufficient dataserver buffer caches. Cached Quick I/O works by taking advantage of the available physical memory that is left over after the operating system reserves the amount it needs and the Sybase dataserver buffer cache has been sized to the maximum capacity allowed within a 32-bit virtual address space. This extra memory serves as a cache to store file data, effectively serving as a second-level cache backing the dataserver buffer caches.

For example, consider a system configured with 12GB of physical memory, an operating system using 1GB, and a total Sybase size of 3.5GB. Unless you have other applications running on your system, the remaining 7.5GB of memory is unused. If you enable Cached Quick I/O, these remaining 7.5GB become available for caching database files.

Note: You cannot allocate specific amounts of the available memory to Cached Quick I/O. When enabled, Cached Quick I/O takes advantage of available memory.

Cached Quick I/O is not beneficial for all device files in a database. Turning on caching for all database device files can degrade performance due to extra memory management overhead (double buffer copying). You must use file I/O statistics to determine which individual database device files benefit from caching, and then enable or disable Cached Quick I/O for individual device files.

If you understand the applications that generate load on your database and how this load changes at different times during the day, you can use Cached Quick I/O to maximize performance. By enabling or disabling Cached Quick I/O on a per-file basis at different times during the day, you are using Cached Quick I/O to dynamically tune the performance of a database.

For example, files that store historical data are not generally used during normal business hours in a transaction processing environment. Reports that make use of this historical data are generally run during off-peak hours when interactive database use is at a minimum. During normal business hours, you can disable Cached Quick I/O for database files that store historical data in order to maximize memory available to other user applications. Then, during off-peak hours, you can enable Cached Quick I/O on the same files when they are used for report generation. This will provide extra memory resources to the database server without changing any database configuration parameters. Enabling file system read-ahead in this manner and buffering read data can provide great performance benefits, especially in large sequential scans.

You can automate the enabling and disabling of Cached Quick I/O on a per-file basis using scripts, allowing the same job that produces reports to tune the file system behavior and make the best use of system resources. You can specify different sets of files for different jobs to maximize file system and database performance.

How Cached Quick I/O improves database performance

Enabling Cached Quick I/O on suitable Quick I/O files improves database performance by using the file system buffer cache to store data. This data storage

speeds up system reads by accessing the system buffer cache and avoiding disk I/O when searching for information.

Having data at the cache level improves database performance in the following ways:

- For read operations, Cached Quick I/O caches database blocks in the system buffer cache, which can reduce the number of physical I/O operations and therefore improve read performance.
- For write operations, Cached Quick I/O uses a direct-write, copy-behind technique to preserve its buffer copy of the data. After the direct I/O is scheduled and while it is waiting for the completion of the I/O, the file system updates its buffer to reflect the changed data being written out. For online transaction processing, Cached Quick I/O achieves better than raw device performance in database throughput on large platforms with very large physical memories.
- For sequential table scans, Cached Quick I/O can significantly reduce the query response time because of the read-ahead algorithm used by Veritas File System. If a user needs to read the same range in the file while the data is still in cache, the system is likely to return an immediate cache hit rather than scan for data on the disk.

How to set up Cached Quick I/O

To set up and use Cached Quick I/O, you should do the following in the order in which they are listed:

- Enable Cached Quick I/O on the underlying file systems used for your database.
- Exercise the system in your production environment to generate file I/O statistics.
- Collect the file I/O statistics while the files are in use.
- Analyze the file I/O statistics to determine which files benefit from Cached Quick I/O.
- Disable Cached Quick I/O on files that do not benefit from caching.

Enabling Cached Quick I/O on a file system

Cached Quick I/O depends on Veritas Quick I/O running as an underlying system enhancement in order to function correctly. Follow the procedures listed here to ensure that you have the correct setup to use Cached Quick I/O successfully.

Prerequisites

- You must have permission to change file system behavior using the `vxtunefs` command to enable or disable Cached Quick I/O. By default, you need superuser (`root`) permissions to run the `vxtunefs` command, but other system users do not. Superuser (`root`) must specifically grant database administrators permission to use this command as follows:

  ```
  # chown root:sybase opt/VRTS/bin/vxtunefs

  # chmod 4550 /opt/VRTS/bin/vxtunefs
  ```

  where users belonging to the `sybase` group are granted permission to run the `vxtunefs` command. We recommend this selective, more secure approach for granting access to powerful commands.
- You must enable Quick I/O on the file system. Quick I/O is enabled automatically at file system mount time.

If you have correctly enabled Quick I/O on your system, you can proceed to enable Cached Quick I/O as follows:

- Set the file system Cached Quick I/O flag, which enables Cached Quick I/O for all files in the file system.
- Setting the file system Cached Quick I/O flag enables caching for all files in the file system. You must disable Cached Quick I/O on individual Quick I/O files that do not benefit from caching to avoid consuming memory unnecessarily. This final task occurs at the end of the enabling process.

Enabling and disabling the qio_cache_enable flag

As superuser (`root`), set the `qio_cache_enable` flag using the `vxtunefs` command after you mount the file system.

To enable the qio_cache_enable flag for a file system

- Use the `vxtunefs` command as follows:

  ```
  # /opt/VRTS/bin/vxtunefs -s -o qio_cache_enable=1 /mount_point
  ```

  For example:

  ```
  # /opt/VRTS/bin/vxtunefs -s -o qio_cache_enable=1 /db02
  ```

  where `/db02` is a VxFS file system containing the Quick I/O files and setting the `qio_cache_enable` flag to “1” enables Cached Quick I/O. This command enables caching for all the Quick I/O files on this file system.

To disable the flag on the same file system

- Use the `vxtunefs` command as follows:

```
# /opt/VRTS/bin/vxtunefs -s -o qio_cache_enable=0 /mount_point
```

For example:

```
# /opt/VRTS/bin/vxtunefs -s -o qio_cache_enable=0 /db02
```

where `/db02` is a VxFS file system containing the Quick I/O files and setting the `qio_cache_enable` flag to "0" disables Cached Quick I/O. This command disables caching for all the Quick I/O files on this file system.

Making Cached Quick I/O settings persistent across reboots and mounts

You can make the Cached Quick I/O system setting persistent across reboots and mounts by adding a file system entry in the `/etc/vx/tunefstab` file.

Note: The `tunefstab` file is a user-created file. For information on how to create the file and add tuning parameters, see the `tunefstab` (4) manual page.

To enable a file system after rebooting

- Put the file system in the `/etc/vx/tunefstab` file and set the flag entry:

```
/dev/vx/dsk/dgname/volname qio_cache_enable=1
```

where:

- `/dev/vx/dsk/dgname/volname` is the name of a block device
- *`dgname`* is the name of the disk group
- *`volname`* is the name of the volume

For example:

```
/dev/vx/dsk/PRODdg/db01 qio_cache_enable=1
/dev/vx/dsk/PRODdg/db02 qio_cache_enable=1
```

where `/dev/vx/dsk/PRODdg/db01` is the block device on which the file system resides.

The `tunefstab` (4) manual pages contain information on how to add tuning parameters.

See the `tunefstab` (4) manual page.

Note: `vxtunefs` can specify a mount point or a block device; `tunefstab` must always specify a block device only.

Using vxtunefs to obtain tuning information

Check the setting of the `qio_cache_enable` flag for each file system using the `vxtunefs` command.

To obtain information on only the qio_cache_enable flag setting

- Use the `grep` command with `vxtunefs`:

  ```
  # /opt/VRTS/bin/vxtunefs /mount_point | grep qio_cache_enable
  ```

  For example:

  ```
  # /opt/VRTS/bin/vxtunefs /db01 | grep qio_cache_enable
  ```

  where `/db01` is the name of the file system. This command displays only the `qio_cache_enable` setting as follows:

  ```
  qio_cache_enable = 0
  ```

  You can also use the `vxtunefs` command to obtain a more complete list of I/O characteristics and tuning statistics.

  See the `vxtunefs` (1) manual page.

To obtain information on all vxtunefs system parameters

- Use the `vxtunefs` command without `grep`:

  ```
  # /opt/VRTS/bin/vxtunefs /mount_point
  ```

For example:

```
# /opt/VRTS/bin/vxtunefs /db01
```

The `vxtunefs` command displays output similar to the following:

```
Filesystem i/o parameters for /db01
read_pref_io = 65536
read_nstream = 1
read_unit_io = 65536
write_pref_io = 65536
write_nstream = 1
write_unit_io = 65536
```

```
pref_strength = 10
buf_breakup_size = 1048576
discovered_direct_iosz = 262144
max_direct_iosz = 1048576
default_indir_size = 8192
qio_cache_enable = 1
write_throttle = 0
max_diskq = 1048576
initial_extent_size = 8
max_seqio_extent_size = 2048
max_buf_data_size = 8192
hsm_write_prealloc = 0
read_ahead = 1
inode_aging_size = 0
inode_aging_count = 0
fcl_maxalloc = 130150400
fcl_keeptime = 0
fcl_winterval = 3600
```

The `vxtunefs`(1) manual pages contain a complete description of `vxtunefs` parameters and the tuning instructions.

See the `vxtunefs`(1) manual page.

Determining candidates for Cached Quick I/O

Determining which files can benefit from Cached Quick I/O is an iterative process that varies with each application. For this reason, you may need to complete the following steps more than once to determine the best possible candidates for Cached Quick I/O.

Before determining candidate files for Quick I/O, make sure the following conditions have been met:

Prerequisites
- You must enable Cached Quick I/O for the file systems.
  See "Enabling Cached Quick I/O on a file system" on page 81.

Usage notes
- See the `qiostat` (1M) manual page for more information.

Collecting I/O statistics

Once you have enabled Cached Quick I/O on a file system, you need to collect statistics to determine and designate the files that can best take advantage of its benefits.

To collect statistics needed to determine files that benefit from Cached Quick I/O

1. Reset the `qiostat` counters by entering:

```
$ /opt/VRTS/bin/qiostat -r /mount_point/filenames
```

2. Run the database under full normal load and through a complete cycle (24 to 48 hours in most cases) to determine your system I/O patterns and database traffic in different usage categories (for example, OLTP, reports, and backups) at different times of the day.
3. While the database is running, run `qiostat -l` to report the caching statistics as follows:

```
$ /opt/VRTS/bin/qiostat -l /mount_point/filenames
```

or, use the `-i` option to see statistic reports at specified intervals:

```
$ /opt/VRTS/bin/qiostat -i n /mount_point/filenames
```

where n is time in seconds

For example:

To collect I/O statistics from all database device files on file system `/db01`:

```
$ /opt/VRTS/bin/qiostat -l /db01/*.dbf
```

About I/O statistics

The output of the `qiostat` command is the primary source of information to use in deciding whether to enable or disable Cached Quick I/O on specific files. Statistics are printed in two lines per object.

The second line of information is defined as follows:

- `CREAD` is the number of reads from the VxFS cache (or total number of reads to Quick I/O files with cache advisory on)
- `PREAD` is the number of reads going to the disk for Quick I/O files with the cache advisory on
- `HIT RATIO` is displayed as a percentage and is the number of `CREADS` minus the number of `PREADS` times 100 divided by the total number of `CREADS`. The formula looks like this:

```
(CREADs - PREADs) * 100/ CREADs
```

The `qiostat -l` command output looks similar to the following:

```
/db01/sysprocs.dbf 17128 9634 68509 38536   24.8  0.4
17124  15728    8.2

/db01/master.dbf     6     1    21    4     10.0  0.0
  6       6     0.0

/db01/user.dbf    62552 38498 250213 153992 21.9  0.4
62567  49060    21.6
```

Analyze the output to find out where the cache-hit ratio is above a given threshold. A cache-hit ratio above 20 percent on a file for a given application may be sufficient to justify caching on that file. For systems with larger loads, the acceptable ratio may be 30 percent or above. Cache-hit-ratio thresholds vary according to the database type and load.

Using the sample output above as an example, the file `/db01/master.dbf` does not benefit from the caching because the cache-hit ratio is zero. In addition, the file receives very little I/O during the sampling duration.

However, the file `/db01/user.dbf` has a cache-hit ratio of 21.6 percent. If you have determined that, for your system and load, this figure is above the acceptable threshold, it means the database can benefit from caching. Also, study the numbers reported for the read and write operations. When you compare the number of reads and writes for the `/db01/user.dbf` file, you see that the number of reads is roughly twice the number of writes. You can achieve the greatest performance gains with Cached Quick I/O when using it for files that have higher read than write activity.

Based on these two factors, `/db01/user.dbf` is a prime candidate for Cached Quick I/O.

See "Enabling and disabling Cached Quick I/O for individual files" on page 88.

Effects of read-aheads on I/O statistics

The number of `CREAD`s in the `qiostat` output is the total number of reads performed, including Cached Quick I/O, and the number of `PREAD`s is the number of physical reads. The difference between `CREAD`s and `PREAD`s (`CREADS - PREADS`) is the number of reads satisfied from the data in the file system cache. Thus, you expect that the number of `PREAD`s would always be equal to or lower than the number of `CREAD`s.

However, the `PREAD`s counter also increases when the file system performs read-aheads. These read-aheads occur when the file system detects sequential reads. In isolated cases where cache hits are extremely low, the output from `qiostat` could show that the number of `CREAD`s is lower than the number of `PREAD`s.

The cache-hit ratio calculated against these CREAD/PREAD values is misleading when used to determine whether Cached Quick I/O should be enabled or disabled.

Under these circumstances, you can make a more accurate decision based on a collective set of statistics by gathering multiple sets of data points. Consequently, you might want to enable Cached Quick I/O for all the device files used by a given database, even if just one of the files exhibited a high cache-hit ratio.

Other tools for analysis

While the output of the qiostat command is the primary source of information to use in deciding whether to enable Cached Quick I/O on specific files, we also recommend using other tools in conjunction with qiostat. For example, benchmarking software that measures database throughput is also helpful. If a benchmark test in which Cached Quick I/O was enabled for a certain set of data files resulted in improved performance, you can also use those results as the basis for enabling Cached Quick I/O.

Enabling and disabling Cached Quick I/O for individual files

After using qiostat or other analysis tools to determine the appropriate files for Cached Quick I/O, you need to disable Cached Quick I/O for those individual files that do not benefit from caching using the qioadmin command.

Prerequisites

- Enable Cached Quick I/O for the file system before enabling or disabling Cached Quick I/O at the individual file level.

Usage notes

- You can enable or disable Cached Quick I/O for individual files while the database is online.
- You should monitor files regularly using qiostat to ensure that a file's cache-hit ratio has not changed enough to reconsider enabling or disabling Cached Quick I/O for the file.
- Enabling or disabling Cached Quick I/O for an individual file is also referred to as setting the cache advisory on or off.
- See the qioadmin (1) manual page.

Setting cache advisories for individual files

You can enable and disable Cached Quick I/O for individual files by changing the cache advisory settings for those files.

To disable Cached Quick I/O for an individual file

- Use the `qioadmin` command to set the cache advisory to `OFF` as follows:

  ```
  $ /opt/VRTS/bin/qioadmin -S filename=OFF /mount_point
  ```

  For example, to disable Cached Quick I/O for the file `/db01/master.dbf`, set the cache advisory to `OFF`:

  ```
  $ /opt/VRTS/bin/qioadmin -S master.dbf=OFF /db01
  ```

To enable Cached Quick I/O for an individual file

- Use the `qioadmin` command to set the cache advisory to `ON` as follows:

  ```
  $ /opt/VRTS/bin/qioadmin -S filename=ON /mount_point
  ```

  For example, running `qiostat`shows the cache hit ratio for the file `/db01/master.dbf`reaches a level that would benefit from caching. To enable Cached Quick I/O for the file `/db01/master.dbf`, set the cache advisory to `ON`:

  ```
  $ /opt/VRTS/bin/qioadmin -S master/dbf=ON /db01
  ```

Making individual file settings for Cached Quick I/O persistent

You can make the enable or disable individual file settings for Cached Quick I/O persistent across reboots and mounts by adding cache advisory entries in the `/etc/vx/qioadmin` file.

Cache advisories set using the `qioadmin` command are stored as extended attributes of the file in the inode. These settings persist across file system remounts and system reboots, but these attributes are not backed up by the usual backup methods, so they cannot be restored. Therefore, always be sure to reset cache advisories after each file restore. This is not necessary if you maintain the cache advisories for Quick I/O files in the `/etc/vx/qioadmin` file.

To enable or disable individual file settings for Cached Quick I/O automatically after a reboot or mount

- Add cache advisory entries in the `/etc/vx/qioadmin` file as follows:

```
device=/dev/vx/dsk/<diskgroup>/<volume>

filename,OFF

filename,OFF

filename,OFF

filename,ON
```

For example, to make the Cached Quick I/O settings for individual files in the `/db01` file system persistent, edit the `/etc/vx/qioadmin` file similar to the following:

```
#
# List of files to cache in /db01 file system
#
device=/dev/vx/dsk/PRODdg/db01

user.dbf,ON
sysprocs.dbf,OFF
master.dbf,OFF
```

Determining individual file settings for Cached Quick I/O using qioadmin

You can determine whether Cached Quick I/O is enabled or disabled for individual files by displaying the file's cache advisory setting using the `qioadmin` command.

Note: To verify caching, always check the setting of the flag `qio_cache_enable` using `vxtunefs`, along with the individual cache advisories for each file.

To display the current cache advisory settings for a file

- Use the `qioadmin` command with the `-P` option as follows:

  ```
  $ /opt/VRTS/bin/qioadmin -P filename /mount_point
  ```

  For example, to display the current cache advisory setting for the file `sysprocs.dbf`in the `/db01`file system:

  ```
  $ /opt/VRTS/bin/qioadmin -P sysprocs.dbf /db01

  sysprocs.dbf,OFF
  ```

Chapter 5

Using Veritas Concurrent I/O

This chapter includes the following topics:

- About Concurrent I/O
- Enabling and disabling Concurrent I/O

About Concurrent I/O

Veritas 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.

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

See "About Quick I/O" on page 57.

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

See "About Cached Quick I/O" on page 79.

How Concurrent I/O works

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 Veritas 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
-

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.

Enabling Concurrent I/O

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

Before enabling Concurrent I/O, make sure the following conditions have been met:

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.

To enable Concurrent I/O on a file system using mount with the -o cio option

- Mount the file system using the `mount`command as follows:

  ```
  # /usr/sbin/mount -F vxfs -o cio special /mount_point
  ```

  where:

  - *`special`* is a block special device

- */mount_point* is the directory where the file system will be mounted.

Disabling Concurrent I/O

If you need to disable Concurrent I/O, unmount the VxFS file system and mount it again without the mount option.

To disable Concurrent I/O on a file system using the mount command

1. Shutdown the Sybase instance.
2. Unmount the file sytem using the `umount` command.
3. Mount the file system again using the `mount` command without using the `-o cio` option.

Chapter 6

Converting existing database configurations to VxFS

This chapter includes the following topics:

- Converting native file systems to VxFS with Quick I/O
- Upgrading from earlier VxFS version layouts
- Converting from raw devices

Converting native file systems to VxFS with Quick I/O

If you are currently using file systems native to your operating system, use the procedure to upgrade each file system used by the database to a VxFS file system with Quick I/O.

To convert a native file system to VxFS with Quick I/O

1. Shut down the database.
2. Create a backup of the UFS file system.
3. Unmount the UFS file system.
4. Remove the entry for the UFS file system from the `/etc/vfstab` directory.
5. Create a VxFS file system of the same size or larger than the original UFS file system, using the mount point where the UFS file system was originally mounted.

   See "Creating a VxFS file system " on page 41.

6 Preallocate Quick I/O files using `qiomkfile`.

See "Creating database files as Quick I/O files using qiomkfile" on page 62.

7 Restore the backup that you created earlier to the Quick I/O files in the new VxFS file system.

8 Restore the database.

Upgrading from earlier VxFS version layouts

Before starting the upgrade process, make sure the following conditions have been met:

Prerequisites
- Perform a full backup of the file system before upgrading to a new disk layout.

Usage notes
- The `vxupgrade` command lets you to upgrade the VxFS file system disk layout while the file system is mounted. See the `vxupgrade`(1M) manual page for more details.
- VxFS supports four file system disk layouts: Versions 4, 5, 6, and 7. New file systems are created with the Version 6 (for large file systems) disk layout by default when the current version of Veritas Storage Foundation for Sybase is installed on a system. You must minimally upgrade to Version 4 disk layout if you want to use the Storage Rollback or Veritas NetBackup BLI Backup features.

To upgrade an existing VxFS file system to a new file system disk layout version

◆ Use the `vxupgrade` command to upgrade to Version 4, 5, 6, or 7 disk layout:

```
# /opt/VRTS/bin/upgrade -n new_version new_version/mount_point
```

where:

- new_version is the version of the file system disk layout you want to upgrade to
- `/mount_point` is the location where the file system is mounted

This is an example of upgrading to disk layout Version 7:

```
# /opt/VRTS/bin/vxupgrade -n 7 /db01
```

To use Quick I/O after upgrading the file system disk layout to version 4, 5, 6, or 7

1. Shut down the database.
2. Make each datafile accessible as a Quick I/O file.

   See "Accessing regular VxFS files as Quick I/O files" on page 65.
3. Restart the database.

Converting from raw devices

If you already have your Sybase dataserver running on raw devices (UNIX raw disk partitions, Solstice DiskSuite 4 (SDS4) metadevices, or VxVM volumes) and would like to convert database devices to use vxfs files with Quick I/O. You can do the upgrade online with Sybase disk mirroring. Due to the overhead used by file systems, make sure the total size of the file systems are about 10 percent larger than the total size of raw devices.

Quick I/O devices fully support aynchronous I/O operations. Therefore, asynchronous I/O will continue to be used by the Sybase server after mirroring a raw device to a Quick I/O file.

Warning: The procedure provided assumes that the database runs on a single file system after the upgrade.

To convert from raw devices to VxFS with Quick I/O

1 Convert a logical database device (proddev) that uses the raw partition to a device of the same logical name but uses a Quick I/O file

/sybdata/proddev_file:

```
$ cd /sybdata
$ qiomkfile -s 1g proddev_file
...
$ isql -Usa -Psa_password -Sdataserver_name
> disk mirror
> name = "proddev",
> mirror = "/sybdata/proddev_file",
> go
...
```

2 After the mirror has been populated, break off the mirror and remove the raw partition from the dataserver:

```
> disk unmirror
> name = "proddev",
> side = "primary", mode = remove
> go
```

3 Repeat this step for all your raw database devices except the master device.

4 Use the same procedure to convert your master device. You should replace the reference to the old raw master device in your `RUN_servername` script in `$SYBASE/install` directory.

5 Once you finish with all the raw partitions in a disk, this disk can be claimed by the Veritas Volume Manager. With the extra space available, you can choose to use it for expanding the volumes and file systems or for mirroring.

6

Chapter 7

Using volume snapshots for dataserver backup and off-host processing

This chapter includes the following topics:

- About snapshot volumes
- Backup and off-host processing applications
- FastResync of snapshot volumes
- Disk group split and join
- Preparing hosts for database backup or off-host processing
- Sybase Adaptive Server Enterprise 12.5 quiesce feature
- How to set up volume snapshots with Sybase ASE 12.5 server
- Implementing online backup or off-host processing
- Creating a warm standby server
- Resynchronizing the snapshot to your ASE dataserver
- Recovering the database from a backup image
- Refreshing a snapshot database image
- Dissociating a snapshot volume
- Removing a snapshot volume

About snapshot volumes

Veritas Volume Manager (VxVM) provides a snapshot facility to create a point-in-time image of a volume to use as a source for taking a backup or for off-host processing. This capability is provided through the `vxsnap` command, and the FastResync and disk group move, split, and join features of VxVM, which are described in this chapter.

A snapshot volume can be used to create a database clone, which can be used on a secondary host for off-host processing, including decision-support analysis and reporting, application development and testing, database backup, and logical error recovery. A snapshot volume can be reysnchronized with the primary database volumes. In addition, in the event of a failure, the primary database can be recovered by resynchronizing it with the snapshot volume.

VxVM's snapshot operation creates a new volume that is an exact point-in-time copy of an existing volume. This is done by creating a mirror of the existing volume using disk space from the pool of free disk space. The mirror is brought up-to-date and a separate snapshot volume is then created. You can use the snapshot volume to make a backup of the original volume at a convenient time without stopping the original volume. A volume can be moved from one disk group to another, provided that there is no disk that is shared between the two disk groups. You can split the snapshot volumes to a new disk group, and import it onto another host to perform off-host backup and decision support operations.

To ensure the mirrors are a consistent and recoverable image of the database, the snapshot functionality must be used in conjunction with Sybase ASE 12.5's database quiesce feature. The Sybase database must be put into the quiescent mode to temporarily suspend I/O to the dataserver before attempting to break off mirrors. Taking a volume snapshot occurs quickly, so the ASE server can be released from the quiescent state immediately after the mirrors are broken off.

Snapshot volumes can also be used in conjunction with the cluster functionality of VxVM.

After the snapshot mirror is synchronized, it continues being updated until it is detached. You can then select a convenient time at which to create snapshot volumes for all the volumes used by the database to represent a valid backup image of the database. You need to either shut down the database for an offline backup or suspend I/O writes to the database for an online backup during the brief time required to detach the snapshot volume (typically less than a minute). In contrast to the brief amount of time that it takes to detach a mirror and create a snapshot volume, the amount of time involved in creating a snapshot mirror is long and directly proportional to the size of the original volume.

Backup and off-host processing applications

The following are typical backup and off-host processing applications made possible using the vxassist command, FastResync, and disk group move, split, and join features of VxVM:

- Database Backup and Restore: Many enterprises require 24/7 online data availability. They cannot afford the downtime involved in backing up critical data offline. By taking a snapshot of the data and then using it to back up your data, your business-critical applications can continue to run without extended down time or impacted performance. After a snapshot volume is created, it can be used as a source to backup the volume.
- 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 clone database that you have created from snapshot volumes, rather than allowing them to compete for access to the primary volume or database. When required, you can quickly resynchronize the clone database with the primary database to get up-to-date information.
- Application Development and Testing: Development or service groups can use a clone database created with snapshot volumes as a test database for new applications. A clone database provides developers, system testers, and quality assurance groups with a realistic basis for testing the robustness, integrity, and performance of new applications.
- Logical Error Recovery: Logical errors caused by an administrator or an application program can compromise the integrity of a database. You can recover a database by restoring the database files from a snapshot volume or by recovering logical objects (such as tables, for example) from a clone database created from snapshot volumes. These solutions are faster than fully restoring database files from tape or other backup media.

FastResync of snapshot volumes

Note: You may need an additional license to use this feature.

FastResync optimizes mirror resynchronization by keeping track of updates to stored data that have been missed by a mirror. If 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. If the data in one mirror is not updated for some reason, it becomes out-of-date, or stale, with respect to the other mirrors

in the volume. The presence of the FastResync map means that only those updates that the mirror has missed need be reapplied to resynchronize it with the volume. A full (and therefore much slower) resynchronization of the mirror from the volume is unnecessary. The FastResync feature increases the efficiency of the VxVM snapshot mechanism to better support operations such as backup and decision support.

The persistent form of FastResync ensures that FastResync maps survive both system crashes and cluster restarts. When snapshot volumes are reattached to their original volumes, FastResync allows the snapshot data to be quickly refreshed and re-used. If Persistent FastResync is enabled on a volume in a private disk group, such incremental resynchronization can happen 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. When the disk groups are rejoined, this allows the snapshot plexes to be quickly resynchronized. Non-Persistent FastResync cannot be used for this purpose.

FastResync allows you to refresh and re-use snapshots rather than discard them. You can quickly resynchronize a snapshot volume with its original volume. This reduces the system overhead required to perform cyclical operations, such as backups, that rely on the snapshot functionality of VxVM. You can also resynchronize the original volume from the snapshot volume. In this case, the database must be shut down and all of the file systems on the original volumes must be unmounted.

Up to 31 snapshot mirrors can be taken and tracked via FastResync.

For more information about FastResync, see the *Veritas Volume Manager Administrator's Guide*.

Disk group split and join

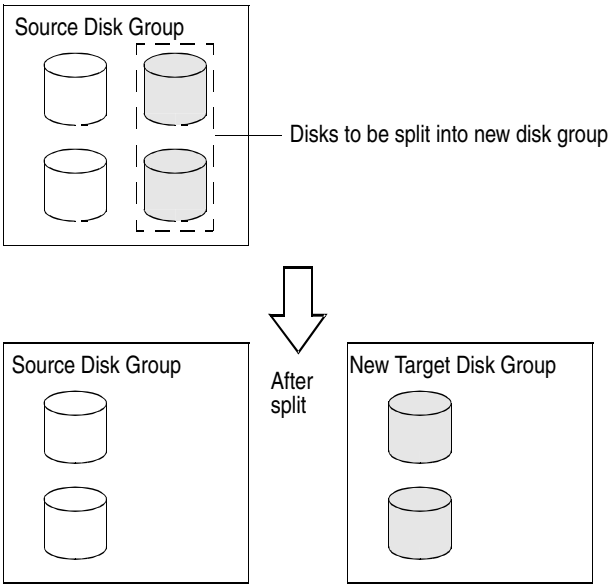
Note: You may need an additional license to use these features.

A snapshot volume can be split off into a separate disk group and deported. It is then ready for importing on another host that is dedicated to off-host processing. At a later stage, the disk group can be deported, re-imported, and joined with the original disk group or with a different disk group.

The `split` and `join` operations allow you to move VxVM objects such as disks or top-level volumes from one disk group to another.

The `split` operation is illustrated in Figure 7-1.

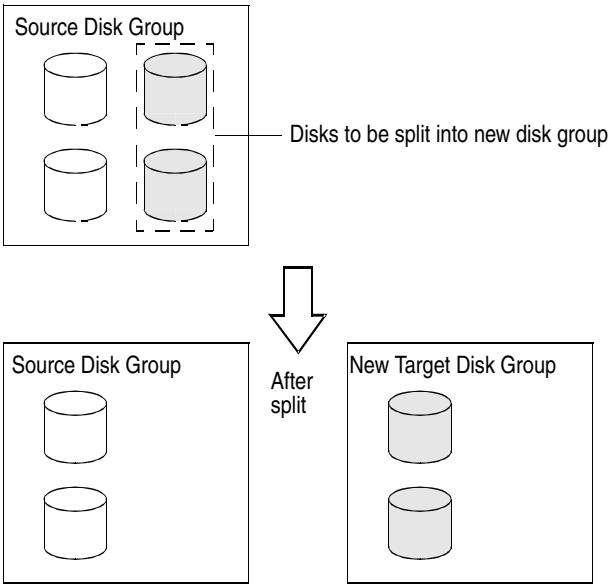
Figure 7-1 Disk group split operation



The `join` operation allows you to remove all VxVM objects from an imported disk group and move them to an imported target disk group. The source disk group is removed when the `join` is complete.

The `join` operation is illustrated in Figure 7-2 .

Figure 7-2 Disk group join operation



Caution: Before moving volumes between disk groups, you must stop all applications that are accessing the volumes and unmount all file systems that are configured in the volumes.

If the system crashes or a hardware subsystem fails, VxVM attempts to complete or reverse an incomplete disk group reconfiguration when the system is restarted or the hardware subsystem is repaired, depending on how far the reconfiguration had progressed. If one of the disk groups is no longer available because it has been imported by another host or because it no longer exists, you must recover the disk group manually.

The disk group `move`, `split` and `join` features have the following limitations:

- Disk groups involved in a `move`, `split` or `join` must be version 90 or greater. If needed, you can upgrade your volume. Refer to the *Veritas Volume Manager Administrator's Guide*.
- The reconfiguration must involve an integral number of physical disks.
- Objects to be moved must not contain open volumes.
- Moved volumes are initially disabled following a disk group `move`, `split` or `join`. If required, use either `vxrecover -m` or `vxvol startall` to restart the volumes.

- Data change objects (DCOs) and snap objects that have been dissociated by persistent FastResync cannot be moved between disk groups.
- Veritas Volume Replicator (VVR) objects cannot be moved between disk groups.
- For a disk group `move` to succeed, the source disk group must contain at least one disk that can store copies of the configuration database after the move.
- For a disk group `split` to succeed, both the source and target disk groups must contain at least one disk that can store copies of the configuration database after the split.
- For a disk group `move` or `join` to succeed, the configuration database in the target disk group must be able to accommodate information about all the objects in the enlarged disk group.
- Splitting or moving a volume into a different disk group changes the volume's record ID.

For more information, see the *Veritas Volume Manager Administrator's Guide*.

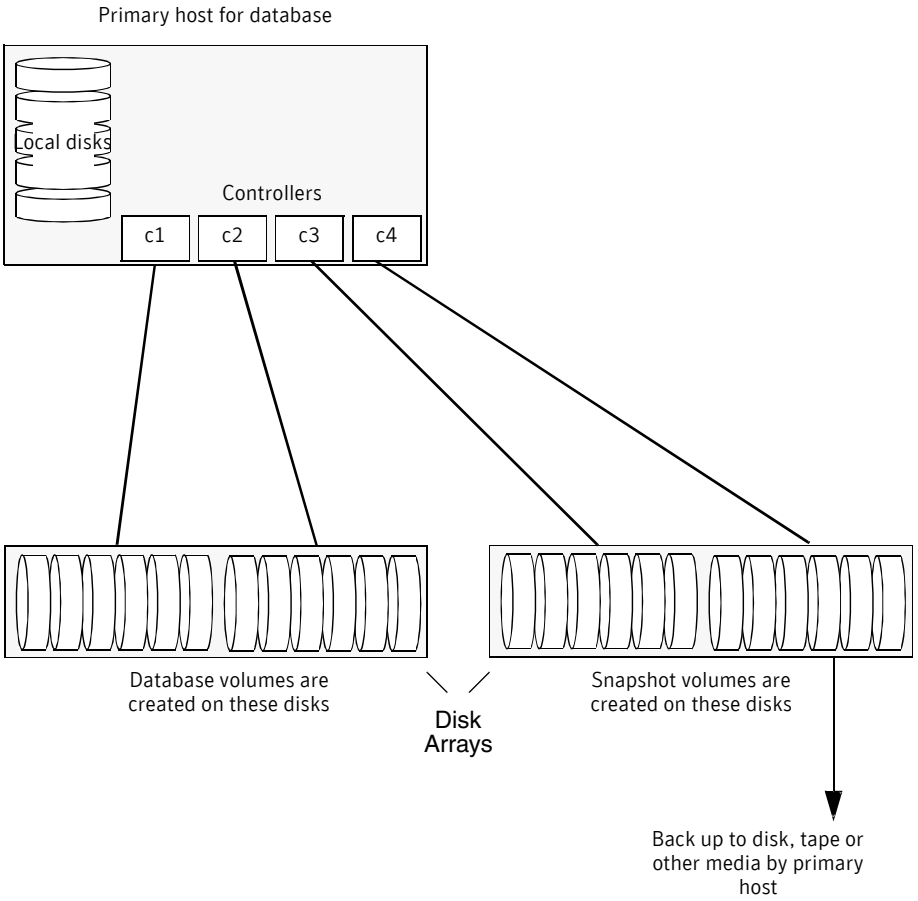
Preparing hosts for database backup or off-host processing

Snapshot volumes can be used on the same host that the database resides on (the primary host) or on a secondary host. On a secondary host, a snapshot volume can be used to implement regular online backup of a volume in a private disk group or set up a clone of the production database for decision support or off-host processing. Snapshot volumes can also be used on the primary host to create a backup image of a database.

Single-host configuration

Figure 7-3 shows the suggested arrangement for using snapshot volumes on the primary host to avoid disk contention.

Figure 7-3 Example of a single-host configuration



Two-host configuration

Figure 7-4 shows the suggested arrangement for using snapshot volumes on a secondary host so that CPU- and I/O-intensive operations can be performed for online backup and decision support without degrading the performance of the primary host running the production database.

A two-host configuration also allows the snapshot volumes to avoid contending for I/O resources on the primary host.

For off-host processing applications, both the primary and secondary hosts need to be able to access the disks containing the snapshot volumes.

Figure 7-4 Example of a two-host configuration

Note: A snapshot volume represents 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 by the overlying file system, or by applications such as databases that have files open in the file system. If the `fsgen` volume usage type is set on a volume that contains a Veritas File System (VxFS), intent logging of the file system metadata ensures the internal consistency of the file system that is backed up. For other file system types, depending on the intent logging capabilities of the file system, there may potentially be inconsistencies between in-memory data and the data in the snapshot image.

For databases, a suitable mechanism must additionally be used to ensure the integrity of segment data when the snapshot volume is taken. The facility to temporarily suspend file system I/O is provided by most modern database software. 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.

Upgrading existing volumes to use VxVM 4.0 features

This section describes how to upgrade a volume created before VxVM 4.0 so that it can take advantage of new features.

Note: The plexes of the DCO volume require persistent storage space on disk to be available. To make room for the DCO plexes, you may need to add extra disks to the disk group, or reconfigure existing volumes to free up space in the disk group. Another way to add disk space is to use the disk group move feature to bring in spare disks from a different disk group.

To upgrade an existing volume

1. Upgrade the disk group that contains the volume to the latest version before performing the remainder of the procedure described in this section. Use the following command to check the version of a disk group:

   ```
   # vxdg list diskgroup
   ```

   To upgrade a disk group to the latest version, use the following command:

   ```
   # vxdg upgrade diskgroup
   ```

2. If the volume to be upgraded has an old-style DRL plex or subdisk, remove it:

   ```
   # vxassist -g diskgroup remove log volume [nlog=n]
   ```

   Use the optional attribute `nlog=` to specify the number, `n`, of logs to be removed. By default, the `vxassist` command removes one log.

3. For a volume that has one or more associated snapshot volumes, reattach and resynchronize each snapshot:

   ```
   # vxassist -g diskgroup snapback snapvol
   ```

   If persistent FastResync was enabled on the volume before the snapshot was taken, the data in the snapshot plexes is quickly resynchronized from the original volume. If persistent FastResync was not enabled, a full resynchronization is performed.

4. Turn off persistent FastResync for the volume:

   ```
   # vxvol -g diskgroup set fastresync=off volume
   ```

5 Dissociate an old-style DCO object, DCO volume, and snap objects from the volume:

```
# vxassist -g diskgroup remove log volume logtype=dco
```

6 Upgrade the volume:

```
# vxsnap [-g diskgroup] prepare volume [ndcomirs=number] \
[regionsize=size] [drl=yes|no|sequential] \
[storage_attribute ...]
```

The `ndcomirs` attribute specifies the number of DCO plexes that are created in the DCO volume. It is recommended that you configure as many DCO plexes as there are data and snapshot plexes in the volume. The DCO plexes are used to set up a DCO volume for any snapshot volume that you subsequently create from the snapshot plexes. For example, specify `ndcomirs=5` for a volume with 3 data plexes and 2 snapshot plexes.

The value of the `regionsize` attribute specifies the size of the tracked regions in the volume. A write to a region is tracked by setting a bit in the change map. The default value is `64k` (64 KB). A smaller value requires more disk space for the change maps, but the finer granularity provides faster resynchronization.

To enable DRL logging on the volume, specify `drl=yes`. If sequential DRL is required, specify `drl=sequential`.

You can also specify `vxassist`-style storage attributes to define the disks that can or cannot be used for the plexes of the DCO volume.

Note: The `vxsnap prepare` command automatically enables persistent FastResync on the volume and on any snapshots that are generated from it. If the volume is a RAID-5 volume, it is converted to a layered volume that can be used with snapshots and FastResync.

Sybase Adaptive Server Enterprise 12.5 quiesce feature

The Sybase Adaptive Server Enterprise (ASE) 12.5 `quiesce database` command allows you to temporarily suspend all writes to one or more databases. Background tasks will also skip any database that is in the suspended state. An ASE database

will stay in the quiesce state until a `quiesce database release` command is issued.

Without shutting down the ASE server, a database administrator (DBA) can take VxVM volume snapshots while the database is in the quiesce state to ensure that break-off mirrors contain a consistent and recoverable database image. Quiescing can be done at the database level and ASE databases that are not in the quiesce state can be updated as usual.

Note: With ASE 12.0, snapshot images taken when the database is in the quiesce state cannot be rolled forward. With ASE 12.5, they can be. Also, the for external dump clause was added to the `quiesce database` command in ASE 12.5 to permit rolling forward a snapshot database.

How to set up volume snapshots with Sybase ASE 12.5 server

Follow these guidelines when setting up the Sybase ASE 12.5 server to use volume snapshots:

- Veritas recommends that you use Quick I/O files instead of raw volumes.
- Do not share file systems and volumes between two ASE servers.
- Do not share the same disk group between two ASE servers.
- Avoid sharing file systems and volumes between user databases if the database administrator intends to create a warm standby for a single database only.
- When taking volume snapshots, create them on disks separate from the original volumes and preferably on the disks that belong to a separate array box. The disk `group split` command cannot move volumes if the move results in two disks sharing the same disk. This rule also applies when creating a DCO log volume for storing FastResync maps.
- Do not put database devices (Quick I/O files, for example) in a Sybase file system. Database files should not be in the Sybase software installation area.
- Do not turn on FastResync tracking for `tempdb` devices because they will be cleared every time the ASE server starts. In addition, the default data and log segments for `tempdb` on the master device should be dropped. `tempdb` should have its own file systems and volumes.

Table 7-1 shows the storage layout for an example ASE server with FastResync running on the primary host. In this storage configuration:

- All volumes are mirrored and belong to the disk group `syb`.
- A secondary host is in the network and has I/O paths to disks in `syb`.
- Primary and secondary hosts access the same set of disks via fibre channel switches. They can also be configured to attach to a dual-hosted RAID box or JBOD.
- The production server is performing I/O in the example.

Table 7-1 Example of storage layout for an ASE server with FastResync

Volume Name	Mount Point	Sybase Device Name	Physical File Name
`sybvol1`	`/sybasedata`	`Master`	`/sybasedata/snaptest/master`
		`Sysprocsdev`	`/sybasedata/snaptest/sysprocs`

Implementing online backup or off-host processing

To implement online backup of an ASE database on a secondary host, you back up its database devices. You can set up a regular backup cycle on another host under Sybase ASE 12.0.0.1 ESD 1 or higher by combining the Persistent FastResync and disk group split and join features of VxVM.

This procedure can also be used to create a second Sybase ASE installation on a secondary host for off-host processing purposes, such as decision-support, reporting, and testing.

Before implementing off-host processing activities, make sure the following conditions have been met:

Prerequisites

- You must be logged in as superuser (root).
- The disk group must be version 90 or later. For more information on disk group versions, see the `vxdg`(1M) manual page.

Usage notes

When creating snapshot mirrors used by databases:

- Ensure that your Sybase instance owner directory is on a volume included in the snapshot.
- Create a separate disk group for Sybase database-related files.
- Do not share file systems and volumes between two ASE servers.
- Do not share the same disk group between two ASE servers.
- If you intend to move a snapshot volume to another host for off-host processing, do not share any disks between the original mirror plex and the snapshot mirror.
- Create snapshot mirrors on a separate controller and separate disk from the primary volume.
- Create snapshot mirrors for datafiles and archive logs so that they do not share any disks with the data of the original volumes. If they are not created in this way, they cannot be split and moved to a secondary host.
- Allocate separate volumes for logs and do not create snapshots on those volumes.
- Resynchronization speed varies based on the amount of data changed in both the primary and secondary volumes during the break-off time.

To back up database devices

1 On the primary host, see if the volume is associated with a version 20 data change object (DCO) and DCO volume that allow instant snapshots and Persistent FastResync to be used with the volume:

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

This command returns `on` if the volume can be used for instant snapshot operations; otherwise, it returns `off`.

If the volume was created under VxVM 4.0, and it is not associated with a new-style DCO object and DCO volume...	Prepare the volume for DRL and instant snapshots. See the *Veritas Volume Manager Administrator's Guide*.
If the volume was created before release 4.0 of VxVM, and it has any attached snapshot plexes or it is associated with any snapshot volumes...	See "Upgrading existing volumes to use VxVM 4.0 features" on page 109.

2 Prepare the volume for being snapshot using the `vxsnap prepare` command:

```
# vxsnap -g diskgroup prepare volume_name [alloc=disk_name]
```

The `vxsnap prepare` command creates a DCO and DCO volumes and associates them with the volume. It also enables persistent FastResync on the volume.

3 On the primary host, verify that FastResync is enabled on the volume:

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

This command returns `on` indicating that FastResync is enabled.

4 Create a snapshot mirror of a volume:

```
# vxsnap -g diskgroup addmir volume_name alloc=disk_name \
[nmirror=N]
```

where *N* specifies the number of mirrors.

Ensure that the mirror is created on a disk with no other volumes.

By default, one snapshot plex is added unless you specify a number using the `nmirror` attribute. For a backup, you should usually only require one plex.

Alternatively, create a new volume for use as the snapshot volume as described in *Veritas Volume Manager Administrator's Guide*.

5 Suspend I/O updates on the primary database by using the `quiesce database` command:

```
$ isql -Usa -Ppassword -Sserver
> quiesce database tag1 hold testdb, sybsystemdb, sybsystemprocs
> go
> quiesce database tag2 hold master
> go
> quit
```

6 If you added a snapshot plex to the volume earlier, create a full-sized instant snapshot volume by running the following command on the primary host:

```
# vxsnap -g diskgroup make \
source=volume/newvol=snapshot_volume/plex=plex_name
```

By default, VxVM attempts to avoid placing snapshot mirrors on a disk that already holds any plexes of a data volume. However, this may be impossible if insufficient space is available in the disk group. In this case, VxVM uses any available space on other disks in the disk group. If the snapshot plexes are placed on disks which are used to hold the plexes of other volumes, this may cause problems when you subsequently attempt to move a snapshot volume into another disk group. To override the default storage allocation policy, you can use storage attributes to specify explicitly which disks to use for the snapshot plexes.

See the *Veritas Volume Manager Administrator's Guide*.

If a database spans more than one volume, you can specify all the volumes and their snapshot volumes using one command, as shown here:

```
# vxsnap -g diskgroup make \
source=vol1/newvol=snapvol1/nmirror=1 \
source=vol2/newvol=snapvol2/nmirror=1 \
source=vol3/newvol=snapvol3/nmirror=1
```

Note: This step sets up the snapshot volumes ready for the backup cycle, and starts tracking changes to the original volumes. When you are ready to make a backup, proceed to the next step.

---

7 On the primary host, resume I/O updates on the primary database. To resume the updates, release the databases from quiesce mode:

```
$ isql -Usa -Ppassword -Sserver
> quiesce database tag2 release
> go
> quiesce database tag1 release
> go
> quit
```

8 On the primary host, use the following command to move the snapshot volume to another disk group from the original disk group:

```
# vxdg split diskgroup new_diskgroup snapshot_volume
```

The `split` (move) command will fail if the move will cause disks to be shared by two disk groups. The `split` (move) command will fail if the result of the move causes disks to be shared by two disk groups.

9 On the primary host, deport the snapshot volume's disk group:

```
# vxdg deport new_diskgroup
```

10 After the split, the snapshot volume is initially disabled. Use the following commands on the secondary host to recover and restart the snapshot volume:

```
# vxrecover -g new_diskgroup
# vxvol -g new_diskgroup start snapshot_volume
```

11 On the secondary host, back up the snapshot volume. If you need to remount the file system in the volume to back it up, first run `fsck` on the volume. The following are the commands for checking and mounting a file system:

```
# fsck -F vxfs /dev/vx/rdsk/new_diskgroup/snapshot_volume
# mount -F vxfs /dev/vx/dsk/new_diskgroup/snapshot_volume \

/mount_point
```

After the file system is mounted, administrators can back up the files using `cp` commands.

12 When the backup is finished, unmount the file system:

```
# umount /mount_point
```

13 On the secondary host, deport the snapshot volume's disk group:

```
# vxdg deport new_diskgroup
```

14 On the primary host, re-import the snapshot volume's disk group:

```
# vxdg import new_diskgroup
```

15 On the primary host, join the snapshot volume's disk group with the original volume's disk group:

```
# vxdg join new_diskgroup diskgroup
```

16 After the join, the snapshot volume is initially disabled. Use the following commands on the primary host to recover and restart the snapshot volume:

```
# vxrecover -g diskgroup -m snapshot_volume
```

The contents of the snapshot volume are now ready to be re-attached and resynchronized with the original volume. VxVM offers a snapprint command for users to check the percentage of volumes that need to be synchronized. In the `vxprint` output, this is referred to as `% DIRTY`.

17 To check the percentage of volumes that need to be synchronized:

```
# vxsnap -g diskgroup print snapshot_volume
```

snapprint<snapshot_volume>

18 To resynchronize with the original volume:

```
# vxsnap -g diskgroup reattach snapvol source=vol
```

Each time you need to back up the volume, repeat this procedure from the `disk split` command to move the snapshot volume to another disk group from the original disk group.

To back up database devices

1 On the primary host, see if the volume is associated with a version 20 data change object (DCO) and DCO volume that allow instant snapshots and Persistent FastResync to be used with the volume:

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

This command returns `on` if the volume can be used for instant snapshot operations; otherwise, it returns `off`.

If the volume was created under VxVM 4.0, and it is not associated with a new-style DCO object and DCO volume...	Prepare the volume for DRL and instant snapshots. See the *Veritas Volume Manager Administrator's Guide*.
If the volume was created before release 4.0 of VxVM, and it has any attached snapshot plexes or it is associated with any snapshot volumes...	See "Upgrading existing volumes to use VxVM 4.0 features" on page 109.

2 Prepare the volume for being snapshot using the `vxsnap prepare` command:

```
# vxsnap -g diskgroup prepare volume_name [alloc=disk_name]
```

The `vxsnap prepare` command creates a DCO and DCO volumes and associates them with the volume. It also enables persistent FastResync on the volume.

3 On the primary host, verify that FastResync is enabled on the volume:

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

This command returns `on` indicating that FastResync is enabled.

4 Create a snapshot mirror of a volume:

```
# vxsnap -g diskgroup addmir volume_name alloc=disk_name \
[nmirror=N]
```

where *N* specifies the number of mirrors.

Ensure that the mirror is created on a disk with no other volumes.

By default, one snapshot plex is added unless you specify a number using the `nmirror` attribute. For a backup, you should usually only require one plex.

Alternatively, create a new volume for use as the snapshot volume as described in *Veritas Volume Manager Administrator's Guide*.

5 Suspend I/O updates on the primary database by using the `quiesce database` command:

```
$ isql -Usa -Ppassword -Sserver
> quiesce database tag1 hold testdb, sybsystemdb, sybsystemprocs
> go
> quiesce database tag2 hold master
> go
> quit
```

6 If you added a snapshot plex to the volume earlier, create a full-sized instant snapshot volume by running the following command on the primary host:

```
# vxsnap -g diskgroup make \
source=volume/newvol=snapshot_volume/plex=plex_name
```

By default, VxVM attempts to avoid placing snapshot mirrors on a disk that already holds any plexes of a data volume. However, this may be impossible if insufficient space is available in the disk group. In this case, VxVM uses any available space on other disks in the disk group. If the snapshot plexes are placed on disks which are used to hold the plexes of other volumes, this may cause problems when you subsequently attempt to move a snapshot volume into another disk group. To override the default storage allocation policy, you can use storage attributes to specify explicitly which disks to use for the snapshot plexes.

See the *Veritas Volume Manager Administrator's Guide*.

If a database spans more than one volume, you can specify all the volumes and their snapshot volumes using one command, as shown here:

```
# vxsnap -g diskgroup make \
source=vol1/newvol=snapvol1/nmirror=1 \
source=vol2/newvol=snapvol2/nmirror=1 \
source=vol3/newvol=snapvol3/nmirror=1
```

Note: This step sets up the snapshot volumes ready for the backup cycle, and starts tracking changes to the original volumes. When you are ready to make a backup, proceed to the next step.

---

7 On the primary host, resume I/O updates on the primary database. To resume the updates, release the databases from quiesce mode:

```
$ isql -Usa -Ppassword -Sserver
> quiesce database tag2 release
> go
> quiesce database tag1 release
> go
> quit
```

8 On the primary host, use the following command to move the snapshot volume to another disk group from the original disk group:

```
# vxdg split diskgroup new_diskgroup snapshot_volume
```

The `split` (move) command will fail if the move will cause disks to be shared by two disk groups. The `split` (move) command will fail if the result of the move causes disks to be shared by two disk groups.

9 On the primary host, deport the snapshot volume's disk group:

```
# vxdg deport new_diskgroup
```

10 After the split, the snapshot volume is initially disabled. Use the following commands on the secondary host to recover and restart the snapshot volume:

```
# vxrecover -g new_diskgroup
# vxvol -g new_diskgroup start snapshot_volume
```

11 On the secondary host, back up the snapshot volume. If you need to remount the file system in the volume to back it up, first run `fsck` on the volume. The following are the commands for checking and mounting a file system:

```
# fsck -F vxfs /dev/vx/rdsk/new_diskgroup/snapshot_volume
# mount -F vxfs /dev/vx/dsk/new_diskgroup/snapshot_volume \

/mount_point
```

After the file system is mounted, administrators can back up the files using `cp` commands.

12 When the backup is finished, unmount the file system:

```
# umount /mount_point
```

13 On the secondary host, deport the snapshot volume's disk group:

```
# vxdg deport new_diskgroup
```

14 On the primary host, re-import the snapshot volume's disk group:

```
# vxdg import new_diskgroup
```

15 On the primary host, join the snapshot volume's disk group with the original volume's disk group:

```
# vxdg join new_diskgroup diskgroup
```

16 After the join, the snapshot volume is initially disabled. Use the following commands on the primary host to recover and restart the snapshot volume:

```
# vxrecover -g diskgroup -m snapshot_volume
```

The contents of the snapshot volume are now ready to be re-attached and resynchronized with the original volume. VxVM offers a snapprint command for users to check the percentage of volumes that need to be synchronized. In the `vxprint` output, this is referred to as `% DIRTY`.

17 To check the percentage of volumes that need to be synchronized:

```
# vxsnap -g diskgroup print snapshot_volume
```

snapprint<snapshot_volume>

18 To resynchronize with the original volume:

```
# vxsnap -g diskgroup reattach snapvol source=vol
```

Creating a warm standby server

With volume snapshots, you can use ASE 12.5 `quiesce database` and `dump`/`load transaction` commands to create a standby database and roll it forward with a transaction dump from the production database.

In some Sybase environments, two installations are maintained: one for production and one for DSS or failover. The DBA creates the initial standby database server by first quiescing the databases on the primary server. Protected by the database quiesce framework, the DBA creates snapshots for all the volumes used by the database server and then releases the databases. Next, the DBA can deport the volume snapshots to another host and start the standby database on that host. After the initial setup, the DBA periodically performs `dump transaction`

commands on the production database and loads them onto the standby database. Between the load transactions, the standby database is available as read-only and users can run queries against the standby database that do not update it. If the production database fails, a DBA can switch the standby database to read-write mode to take over the production workload.

The following is a summary of the steps to create a warm standby server in the ASE 12.5 environment:

- Use VxVM volume snapshots to move the entire ASE server to the secondary host. After users perform the first load transaction step, the failover database will be online standing by.
- Perform periodic `dump transaction` commands for databases to a dedicated disk dump device. This disk dump device can reside on a separate disk group, for example, `dump_diskgroup`, which will be able to deport/import between the primary host and the secondary host. After the dump is finished, deport the disk group `dump_diskgroup` on the primary host.
- Import the disk group `dump_diskgroup` onto the secondary host and perform `load` transaction commands on the failover database.
- Repeat the `dump`/`load transaction` commands (steps 2 and 3 above) in fixed intervals. For example, repeat the process every hour.
- Repeat the entire process in fixed intervals. For example, repeat the process every day.

Before creating a warm standby server, make sure the following conditions have been met:

Prerequisites

- You must be logged in as superuser (root).
- The disk group must be version 90 or later. For more information on disk group versions, see the `vxdg`(1M) manual page.

Usage notes

- Ensure that your Sybase instance owner directory is on a volume included in the snapshot.
- Create a separate disk group for Sybase database-related files.
- Do not share file systems and volumes between two ASE servers.
- Do not share the same disk group between two ASE servers.
- If you intend to move a snapshot volume to another host for off-host processing, do not share any disks between the original mirror plex and the snapshot mirror.
- Create snapshot mirrors on a separate controller and separate disk from the primary volume.
- Create snapshot mirrors for datafiles and archive logs so that they do not share any disks with the data of the original volumes. If they are not created in this way, they cannot be split and moved to a secondary host.
- Allocate separate volumes for logs and do not create snapshots on those volumes.
- Resynchronization speed varies based on the amount of data changed in both the primary and secondary volumes during the break-off time.

To create a warm standby server

1 On the primary host, use the following command to see if the volume is associated with a version 20 data change object (DCO) and DCO volume that allow instant snapshots and Persistent FastResync to be used with the volume:

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

This command returns `on` if the volume can be used for instant snapshot operations; otherwise, it returns `off`.

If the volume was created under VxVM 4.0, and it is not associated with a new-style DCO object and DCO volume...	Prepare the volume for DRL and instant snapshots. See the *Veritas Volume Manager Administrator's Guide*.
If the volume was created before release 4.0 of VxVM, and it has any attached snapshot plexes or it is associated with any snapshot volumes...	See "Upgrading existing volumes to use VxVM 4.0 features" on page 109.

Note: If the volume was created under PRODUCTNAME 4.0, and it is not associated with a new-style DCO object and DCO volume, follow the procedure for preparing a volume for DRL and instant snapshots in the *Veritas Volume Manager Administrator's Guide*.

If the volume was created before release 4.0 of PRODUCTNAME, and it has any attached snapshot plexes or it is associated with any snapshot volumes, follow the procedure given in .

2 Prepare the volume for taking a snapshot using the `vxsnap prepare` command:

```
# vxsnap -g diskgroup prepare volume_name [alloc=disk_name]
```

The `vxsnap prepare` command creates a DCO and DCO volumes and associates them with the volume. It also enables persistent FastResync on a volume.

3 On the primary host, verify that FastResync is enabled on the volume:

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

This command returns `on` indicating that FastResync is enabled.

4 Create a snapshot mirror of the volume.

Note: Ensure that the mirror is created on a disk with no other volumes.

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

5 Prepare the secondary host to receive the snapshot volume that contains the copy of the database tables. This may involve setting up private volumes to contain any redo logs, and configuring any files that are used to initialize the database.

6 List the plexes and determine the name of the snapshot plex. The snapshot plex appears directly after the snapshot volume:

```
# vxprint -g diskgroup
```

7 On the primary host, suspend I/O updates by using the `quiesce database` command:

```
$ isql -Usa -Ppassword -Sserver
> quiesce database tag1 hold userdb, sybsystemdb, \
sybsystemprocs for external dump
> go
> quiesce database tag2 hold master for external dump
> go
> quit
```

Note: The `for external dump` clause is new in ASE 12.5 and it tells the ASE that a physical copy of the database device(s) will be made during the quiesce state, and that the copy will serve as the foundation for a new dump sequence. A dump made with this clause can be used for starting a secondary server and the secondary server can be rolled forward with transaction logs.

The `for external dump` clause has no effect on system databases.

8 Create a full-sized instant snapshot volume on the primary host by specifying the snapshot plex identified in 6:

```
# vxsnap -g diskgroup make \
source=volume_name/newvol=snapshot_volume/plex=plex_name
```

By default, PRODUCTNAME attempts to avoid placing snapshot mirrors on a disk that already holds any plexes of a data volume. However, this may be impossible if insufficient space is available in the disk group. In this case, PRODUCTNAME uses any available space on other disks in the disk group. If the snapshot plexes are placed on disks that are used to hold the plexes of other volumes, this may cause problems when you subsequently attempt to move a snapshot volume into another disk group. To override the default storage allocation policy, you can use storage attributes to specify explicitly which disks to use for the snapshot plexes.

See the *Veritas Volume Manager Administrator's Guide*.

If a database spans more than one volume, you can specify all the volumes and their snapshot volumes using one command, as shown here:

```
# vxsnap -g diskgroup make \
source=volume1/newvol=snapshot_volume1/nmirror=2 \
source=volume2/newvol=snapshot_volume2/nmirror=2 \
source=volume3/newvol=snapshot_volume3/nmirror=2
```

Note: This step sets up the snapshot volumes, and starts tracking changes to the original volumes. When you are ready to create a clone database, proceed to the next step.

9 On the primary host, resume I/O updates on the primary database. To resume the updates, release the databases from quiesce mode:

```
# vxdg split diskgroup new_diskgroup snapshot_volume
```

10 On the primary host, move the snapshot volume to another disk group from the original disk group:

```
# vxdg split diskgroup new_diskgroup snapshot_volume
```

The `split` (move) command will fail if the result of the move causes disks to be shared by two disk groups.

11 On the primary host, deport the snapshot volume's disk group:

```
# vxdg deport new_diskgroup
```

12 On the secondary host, import the snapshot volume's disk group:

```
# vxdg import new_diskgroup
```

13 After the split, the snapshot volume is initially disabled. Use the following commands on the secondary host to recover and restart the snapshot volume:

```
# vxrecover -g new_diskgroup -m snapshot_volume
# vxvol -g new_diskgroup start snapshot_volume
```

14 On the secondary host, check and mount all of the file systems used by the ASE servers. The following are the commands for checking and mounting a file system:

```
# fsck -F vxfs /dev/vx/dsk/new_diskgroup/snapshot_volume
# mount -F vxfs /dev/vx/dsk/new_diskgroup/snapshot_volume \
/mount_point1
```

15 Change the ownership of the mount point specified in step 14 to this instance:

```
# chown -R sybase:sybase mount_point
```

Ensure that redo logs are part of the snapshot volume so that Sybase can perform a crash recovery if needed.

16 Create a server on the secondary host:

```
$ server_name
```

17 Modify the `interfaces` file on the secondary host to change the host name from the primary to the secondary host. The `interfaces` file is located under the /$SYBASE directory.

18 Start the ASE server on the secondary host with the ASE 12.5 `-q` flag:

```
$ /sybase/ASE-12_5/bin/dataserver -Sserver_name \
-d/master_device_path -e/error_log_file_path \
-M/sybase_software_path -q
```

19 After transactions have occurred on the primary host, deport the snapshot volume's disk group on the secondary host:

```
$ isql -Usa -Ppassword -Sserver
> dump tran userdb to dump_device with standby_access
> go
> quit
```

20 On the primary host, import the snapshot volume's disk group:

```
# vxdg split diskgroup dump_diskgroup snapshot_dump_volume
```

21 Join new_diskgroup back to the original disk group:

```
# vxdg join new_diskgroup diskgroup
```

22 Recover the snapshot volume:

```
# vxrecover -g new_diskgroup
```

23 Dump the transactions to the dump device (dump_diskgroup):

```
$ isql -Usa -Ppassword -Sserver
  > dump tran userdb to dump_diskgroup with standby_access
  > go
  > quit
```

24 Deport the disk group dump_diskgroup on the primary host:

```
# vxdg deport dump_diskgroup
```

25 Import the disk group dump_diskgroup on the secondary host:

```
# vxdg import dump_diskgroup
```

26 On the secondary host, recover and restart the snapshot volume:

```
# vxrecover -g dump_diskgroup -m snapshot_volume
  # vxvol -g dump_diskgroup start snapshot_volume
```

27 Load the transaction dumps to the failover database and put the database on the secondary host online:

```
$ isql -Usa -Ppassword -Sserver
> load tran userdb from dump_device
> go
> online database userdb for standby_access
> go
```

28 After the load is finished, deport `dump_diskgroup` and import it back to the primary host disk group. It is recommended that 19 to 27 be repeated regularly (for example, every hour).

29 To refresh all ASE servers, join new_diskgroup back to the original disk group and restart all volume snapshots:

```
# vxdg join new_diskgroup diskgroup
  # vxrecover -g diskgroup
```

30 Resynchronize the snapshot volume with the original volume:

```
# vxsnap -g diskgroup reattach snapshot_volume source=volume
```

The entire cycle can be repeated every day, if desired.

In this example, a warm standby dataserver is created in an ASE 12.5 environment on a secondary host.

Log in as root and create the disk group, `syb`, for database use.

```
# vxdg init syb disk1=c2t130d0s2
# vxdg -g syb adddisk disk2=c2t131d0s2
```

Log in as the instance owner and create a volume and VxFS file system.

```
# vxassist -g syb make sybvol1 5g disk1
# mkfs -F vxfs -o largefiles /dev/vx/rdsk/syb/sybvol1
# mkdir /sybasedata
# mount -F vxfs -o largefiles /dev/vx/dsk/syb/sybvol1 /sybasedata
# chown -R sybase:sybase /sybasedata
```

Create an instance, `inst1`.

```
# /sybase/ASE-12.5/bin/dataserver -d/sybasedata/snaptest/master \
-e/sybasedata/snaptest/errorlog -M/sybase/ASE-12.5/ -s/mysrv -q
```

Create an ASE dataserver on the VxFS file system.

```
# srvbuild -r srvbuild.mysrv.rs
```

Modify the `interfaces` file on the secondary host to change the host name from the primary to the secondary host. The `interfaces` file is located under the `/$SYBASE` directory.

Log in to the dataserver and create a database.

```
$ isql -Usa -P -Smysrv
> disk init
> name=db1
> physname=/sybasedata/testdb/testfile
> vdevno=3
> size=5m
> dsync=true
> go
> disk init
> name=db1log
> physname=/sybasedata/testdb/testlog
```

```
> vdemo=4
> size=3m
> dsync=true
> go
> create database dbtest1 on db1=3m log db1log=1m
> go
> create table dept ( chat# char(2), name char(2), talk decimal(5),
> go
> insert into dept values (ab, cd, 12345);
> go
```

Log in as root and prepare the volume, `sybvol1`, for taking a snapshot.

```
# vxsnap -g syb prepare sybvol1 alloc=syb01
```

Verify that DCO and FastResync are enabled on the volume.

```
# vxprint -g syb -F%instant sybvol1
on

on
```

Create a snapshot mirror.

```
# vxsnap -g syb addmir sybvol1 alloc=syb02
```

Create a full-sized instant snapshot, `snap_vol1`, on the primary host.

```
# vxsnap -g syb make \
source=sybvol1/newvol=snap_vol1/plex=sybvol1-02
```

Move the snapshot volume, `snap_vol1`, into a separate disk group, `new_syb`, from the original disk group, `syb`.

```
# vxdg split syb new_syb snap_vol1
```

Deport the snapshot volume's disk group on the primary host.

```
# vxdg deport new_syb
```

Import the snapshot volume's disk group on the secondary host.

```
# vxdg import new_syb
```

After the split, the snapshot volume is initially disabled. Recover and restart the snapshot volume.

```
# vxrecover -g new_syb -m snap_vol1
# vxvol -g new_syb start snap_vol1
```

On the secondary host, check and mount the snapshot volume.

```
# mkdir /sybasedata
# fsck -F vxfs /dev/vx/rdsk/new_syb/snap_vol1
# mount -F vxfs /dev/vx/dsk/new_syb/snap_vol1 /sybasedata
# chown -R sybase:sybase /sybasedata
```

Log in as the instance owner.

```
# su - sybase
```

Start the ASE server on the secondary host with the ASE 12.5 `-q` flag:

```
$ /sybase/ASE-12.5/bin/dataserver -d/sybasedata/snaptest/maste \
-e/sybasedata/snapshot/errorlog -M/sybasedata/ASE-12.5/ -s mysrv \
-q
> isql -Usa -P -Smysrv
> use dbtest1
> go
```

The warm standby dataserver is now ready to use.

Resynchronizing the snapshot to your ASE dataserver

When you want to refresh a clone database, you can resynchronize it with the original database. This is also known as refreshing the snapshot volume or merging the split snapshot image back to the current database image. After resynchronizing, the snapshot is ready to be used for backup or decision-support purposes.

There are two choices when resynchronizing the data in a volume:

- Resynchronizing the snapshot from the original volume. This option is explained in this section.
- Resynchronizing the original volume from the snapshot. This choice is known as reverse resynchronization. Reverse resynchronization may be necessary to restore a corrupted database or file system, or to implement upgrades to production software, and is usually much quicker than using alternative approaches such as full restoration from backup media.
  See “Recovering the database from a backup image” on page 139.

When you want to resynchronize the snapshot volume's data with the primary database, you can refresh the snapshot plexes from the original volume as described below.

To resynchronize the snapshot image

1 On the secondary host, shut down the clone database.

```
> isql -Usa -P -Sserver
> shutdown
> go
```

2 Log in as root and unmount the file system.

```
# umount mount_point
```

3 On the secondary host, deport the snapshot volume's disk group:

```
# vxdg deport new_diskgroup
```

4 On the primary host, re-import the snapshot volume's disk group:

```
# vxdg import new_diskgroup
```

5 On the primary host, rejoin the snapshot volume's disk group with the original volume's disk group:

```
# vxdg join new_diskgroup diskgroup
```

6 After the join, the snapshot volume is initially disabled. Use the following commands on the primary host to recover and restart the snapshot volume:

```
# vxvol -g diskgroup startall
```

The contents of the snapshot volume are now ready to be refreshed.

7 Reattach the snapshot volume's disk group to the original volume's disk group:

```
# vxsnap -g diskgroup reattach <snapshot_volume> \
source=volume
```

8 Follow step 10 through step 18 to split the snapshot volume from the disk group when synchronization is again complete.

The snapshot image is now ready to be re-used for backup or decision-support applications.

Example

On the secondary host:

Log in as the instance owner and shut down the clone database:

```
> isql -Usa -P -Smysrv
> shutdown
> go
```

Log in as root and unmount the file system:

```
# umount /sybasedata
```

As root, deport the snapshot volume's disk group:

```
# vxdg deport new_syb
```

On the primary host:

Log in as root and import the snapshot volume's disk group:

```
# vxdg import new_syb
```

As root, rejoin the snapshot volume's disk group to the original volume's disk group:

```
# vxdg join new_syb syb
```

As root, recover and restart the snapshot volume:

```
# vxrecover -g syb
# vxvol -g syb start snap_vol1
```

As root, reattach the snapshot:

```
# vxsnap -g syb reattach snap_vol1 source=sybvol1
```

Split the snapshot volume into a separate disk group from the original disk group:

```
# vxdg split syb new_syb snap_vol1
```

Note: The `split` (move) command will fail if the result of the move causes disks to be shared by two disk groups.

Deport the snapshot volume's disk group:

```
# vxdg deport new_syb
```

On the secondary host:

Log in as root and import the snapshot volume's disk group:

```
# vxdg import new_syb
```

After the split, the snapshot volume is initially disabled. As root, use the following commands on to recover and restart the snapshot volume:

```
# vxrecover -g syb -m snap_vol1
```

As root, check and mount all of the file systems used by the ASE servers. The following are the commands for checking and mounting a file system:

```
# fsck -F vxfs /dev/vx/dsk/syb/snap_vol1
# mount -F vxfs /dev/vx/dsk/syb/snap_vol1 /sybasedata
```

Ensure that the dataserver is up and running on the secondary host.

As root, change the ownership of the mount point `sybasedata`, if needed:

```
# chown -R sybase:sybase sybasedata
```

Note: Ensure that redo logs are part of the snapshot volume so that Sybase can perform a crash recovery if needed.

Modify the `interfaces` file on the secondary host to change the host name from the primary to the secondary host. The `interfaces` file is located under the `/$SYBASE` directory.

Start the ASE server on the secondary host with the ASE 12.5 `-q` flag:

```
$ /sybase/ASE-12.5/bin/dataserver -d/sybasedata/snaptest/master \
-e/sybasedata/snapshot/errorlog -M/sybasedata/ASE-12.5/ -s mysrv \
-q
> isql -Usa -P -Smysrv
> use dbtest1
> go
```

Recovering the database from a backup image

A backup image on the primary host can be used to restore the primary database if it becomes corrupted. This section describes the procedure for using snapshot volumes to create a backup image of a database on the primary host. It also explains how to recover the primary database from the backup image.

Prerequisites

- You must be logged in as superuser (root).
- The disk group must be version 90 or later. For more information on disk group versions, see the `vxdg`(1M) manual page.

Usage notes

- Ensure that your Sybase instance owner directory is on a volume included in the snapshot.
- Create a separate disk group for Sybase database-related files.
- Do not share file systems and volumes between two ASE servers.
- Do not share the same disk group between two ASE servers.
- If you intend to move a snapshot volume to another host for off-host processing, do not share any disks between the original mirror plex and the snapshot mirror.
- Create snapshot mirrors on a separate controller and separate disk from the primary volume.
- Create snapshot mirrors for datafiles and archive logs so that they do not share any disks with the data of the original volumes. If they are not created in this way, they cannot be split and moved to a secondary host.
- Allocate separate volumes for logs and do not create snapshots on those volumes.
- Resynchronization speed varies based on the amount of data changed in both the primary and secondary volumes during the break-off time.

To create a backup image on the primary host

1 Follow steps 1 to 4 and then steps 6 to 8 in:

See "Resynchronizing the snapshot to your ASE dataserver" on page 134.

to create a snapshot volume on the primary host.

A failure occurs, requiring you to restore the primary database from the backup image.

2 Shut down the primary database if it is still active.

```
> isql -Usa -P -Sserver
> shutdown
> go
```

3 Unmount the file system for the primary database

```
# umount -f /mount_point
```

4 Restore the volume from the snapshot volume. You must be logged in as root to execute this command.

```
# vxsnap -g diskgroup restore volume_name \
source=snapshot_volume destroy=yes
```

5 Check and mount the file system for the primary database.

```
# fsck -F vxfs /dev/vx/dsk/diskgroup/volume
  # mount -F vxfs /dev/vx/dsk/diskgroup/volume /mount_point
```

6 Ensure that the ASE dataserver has the correct permissions:

```
# chown -R sybase:sybase /mount_point
```

7 As the instance owner, start the primary database.

```
> isql -Usa -P -Sserver
```

8 Bring the backup image of the database online by applying the transaction logs and roll it forward.

The Sybase clients are now ready to reconnect.

Note: The archive logs are needed to roll the database forward. In case the archive logs are corrupted on the primary database, they need to be restored from backup.

In this example the primary database is restored from a backup image on the primary host.

After the corruption occurs, log in as the instance owner and terminate all active connections to the primary database:

```
> isql -Usa -P -Smysrv
> shutdown
> go
```

Log in as root and unmount the primary database's file systems:

```
# umount -f /sybasedata
```

As root, join the snapshot volume's disk group with the primary disk group:

```
# vxdg join new_syb syb
```

After the join, the snapshot volume is initially disabled. Recover and restart the snapshot volume:

```
# vxrecover -g syb -m snap_vol1
```

As root, restore the volume from the snapshot volume:

```
# vxsnap -g syb restore sybvol1 source=snap_vol1 destroy=yes
```

As root, check and mount the file system for the primary database:

```
# fsck -F vxfs /dev/vx/rdsk/syb/sybvol1
# mount -F vxfs /dev/vx/dsk/syb/sybvol1 /sybasedata
```

Ensure that the ASE dataserver has the correct permissions:

```
# chown -R sybase:sybase /sybasedata
```

Log in as the instance owner and start the primary database:

```
# su - sybase
$ /sybase/ASE-12.5/install/startserver -f RUN_mysrv
```

The database has now been restored to where it was when the snapshot was taken or refreshed last.

Refreshing a snapshot database image

Refresh a snapshot database image so that it can later be used to restore the primary database.

Before refreshing a snapshot database image, make sure the following conditions have been met:

Prerequisites

- You must be logged in as superuser (root).
- The disk group must be version 90 or later. For more information on disk group versions, see the `vxdg`(1M) manual page.

Usage notes

When creating snapshot mirrors used by databases refer to the following:

- Ensure that your Sybase instance owner directory is on a volume included in the snapshot.
- Create a separate disk group for Sybase database-related files.
- Do not share file systems and volumes between two ASE servers.
- Do not share the same disk group between two ASE servers.
- If you intend to move a snapshot volume to another host for off-host processing, do not share any disks between the original mirror plex and the snapshot mirror.
- Create snapshot mirrors on a separate controller and separate disk from the primary volume.
- Create snapshot mirrors for datafiles and archive logs so that they do not share any disks with the data of the original volumes. If they are not created in this way, they cannot be split and moved to a secondary host.
- Allocate separate volumes for logs and do not create snapshots on those volumes.
- Resynchronization speed varies based on the amount of data changed in both the primary and secondary volumes during the break-off time.

To refresh a snapshot image of the database on the primary host

1. As root, join the snapshot disk group with the primary disk group:

```
# vxdg join new_diskgroup diskgroup
```

2. After the join, the snapshot volume is initially disabled. Recover and restart the snapshot volume:

```
# vxrecover -g diskgroup -m snapshot_volume
```

3. Suspend I/O updates on the primary database by using the `quiesce database` command:

```
$ isql -Usa -Ppassword -Sserver
> quiesce database tag1 hold testdb, sybsystemdb, \
sybsystemprocs
> go
> quiesce database tag2 hold master
> go
> quit
```

4. Refresh the snapshot image:

```
$ vxsnap -g diskgroup refresh snapshot_volume \
  source=volume_name syncing=yes
```

5. Resume I/O updates on the primary database. To resume the updates, release the databases from quiesce mode:

```
$ isql -Usa -Ppassword -Sserver
> quiesce database tag2 release
> go
> quiesce database tag1 release
> go
> quit
```

6. On the primary host, split the snapshot volume into a separate disk group from the original disk group:

```
# vxdg split diskgroup new_diskgroup snapshot_volume
```

Dissociating a snapshot volume

You can permanently break the link between a snapshot and its original volume so that the snapshot volume becomes an independent volume.

- You must be logged in as superuser (root).
- The snapshot volume must be associated with its original volume.

You can permanently break the link between a snapshot and its original volume so that the snapshot volume becomes an independent volume.

To dissociate a snapshot from its original volume

1 Use the `vxsnap dis` command as follows:

```
# vxsnap -g <diskgroup> [-f] dis <snapshot_volume>
```

2 Remove the DCO logs from the snapshot volume:

```
# vxsnap -g <diskgroup> [-f] unprepare <snapshot_volume>
```

3 Remove the DCO logs from the primary volume, if necessary:

```
# vxsnap -g <diskgroup> [-f] unprepare <volume_name>
```

This example shows how to dissociate a snapshot volume from its original volume:

```
# vxsnap -g syb dis snap_vol
```

This example shows how to remove the DCO logs from the snapshot volume:

```
# vxsnap -g syb -f unprepare snap_vol
```

This example shows how to remove the DCO logs from the primary volume:

```
# vxsnap -g syb -f unprepare sybvol
```

Removing a snapshot volume

If a volume is no longer necessary, you can remove the volume and free up the disk space for other uses by using the `vxedit rm` command.

Note: Removing a volume destroys all of the data in that volume. After a volume is removed, the space it occupied is returned to the free space pool.

Before removing a snapshot volume, make sure the following requirements have been met:

Prerequisites
- You must be logged in as superuser (root).
- You must have an existing snapshot volume.

To remove a snapshot volume using the command line

1 Dissociate the snapshot from its original volume:

```
# vxsnap -g diskgroup [-f] dis snapshot_volume
```

2 Remove the snapshot volume:

```
# vxedit -g diskgroup -rf rm snapshot_volume
```

where `-r` recursively removes all plexes and subdisks.

Warning: If the volume is on a mounted file system, you must unmount it before removing the volume.

This example shows how to remove a snapshot volume from a disk group:

```
# vxsnap -g syb dis snap_vol1
# vxedit -g syb -rf rm snap_vol1
```

Chapter 8

Tuning for performance

This chapter includes the following topics:

- Additional documentation
- About tuning VxVM
- About tuning VxFS
- About tuning Sybase dataservers
- About tuning Solaris for Sybase

Additional documentation

Use the tuning tips and information provided in this chapter in conjunction with other more in-depth publications, such as:

- *Sybase Adaptive Server Enterprise Performance and Tuning Guide*— covers general tuning tips
- Other generic Sybase documentation that deals with Sybase tuning issues
- *Veritas Volume Manager Administrator's Guide*, chapter on "VxVM Performance Monitoring"

About tuning 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.

For more information on tuning VxVM, see the *Veritas Volume Manager 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 *Veritas Volume Manager 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` or the `df -os` commands.

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.

`default_indir_ size` On VxFS, files can have up to ten variably sized direct extents stored in the inode. After these extents are used, the file must use indirect extents that are a fixed size. The size is set when the file first uses indirect extents. These indirect extents are 8K by default. The file system does not use larger indirect extents because it must fail a write and return `ENOSPC` if there are no extents available that are the indirect extent size. For file systems with many large files, the 8K indirect extent size is too small. Large files that require indirect extents use many smaller extents instead of a few larger ones. By using this parameter, the default indirect extent size can be increased so that large files in indirects use fewer large extents.

Be careful using this tunable. If it is too large, then writes fail when they are unable to allocate extents of the indirect extent size to a file. In general, the fewer and the larger the files on a file system, the larger the `default_indir_size` parameter can be. The value of this parameter is generally a multiple of the `read_pref_io` parameter.

This tunable is not applicable on Version 4 disk layouts.

`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 tuning Sybase dataservers

To achieve optimal performance on your Sybase dataserver, the server may need to be tuned to work together with VxFS. This section lists some general suggestions.

Sybase tempdb database

Sybase `tempdb` is used quite frequently so it should be placed on a separate file system mounting on a dedicated volume. The volume should be striped and its disks should not be shared with other high activity volumes. This database should also bind to its own cache space with the Sybase ASE-named cache feature to reduce paging.

The `tempdb` database needs to be large enough to contain all the work tables and temporary tables created by the dataserver. When the Adaptive Server is installed, `tempdb` is created entirely on the master device. The database administrator need to move `tempdb` on to larger, dedicated devices. (The default size is 2 MB only.)

To do so, first alter `tempdb` onto the new device created on the new Quick I/O file. By default, the master device is included in `tempdb`'s logsegment and defaultsegment. To have control on the placement of the log segment and default

segment, you need to drop those segments from the master device as shown in the example below.

See the tempdb performance chapter in the Sybase ASE Performance and Tuning Guide.

To change tempdb to a dedicated 200MB device

1 Create a Quick I/O file:

```
$ qiomkfile -s 200m /new/newtempdb_dev
```

2 Execute these commands on the Sybase Adaptive Server:

```
$ isql -Usa -P<sa_password> -S<dataserver_name>
> disk init
> name="newtempdb",
> physname="/newtempdb_dev",
> vnevno=<next_available_number>,
> size=102400
> go
> alter database tempdb on newtempdb=200
> go
> sp_dropsegment "default", tempdb, master
> go
> sp_dropsegment logsegment, tempdb, master
> go
```

Work tables and other temporary tables in `tempdb` will now be created on the device `newtempdb` instead of on the `tempdb` master device.

Sybase sybsecurity database

If you use auditing on your dataserver, the auditing system performs frequent input and output to the `sysaudits` table in the `sybsecurity` database. Follow the same recommendation on the placement of this database as that for the `tempdb`.

Placement of the transaction logs

You should place the transaction log on a separate volume to reduce contentions. Because the I/O pattern of a transaction log is sequential, the logsegment should consist of devices created on Quick I/O files mounting on simple (non-striped) volumes. Do not put log devices and others database devices on the same file system. For log devices, you should use mirroring instead of RAID-5 for high availability.

Database device layout

Create database devices for user tables from Quick I/O file systems mounted on striped volumes. Stripe across as many disk drives as possible. For heavily updated tables, use mirroring for high availability instead of RAID-5. Use user-defined segments to achieve the exact placements for your database objects.

Nonclustered indexes placement

Data are usually being accessed at the same time the nonclustered indexes are accessed. To reduce contention, you should separate the data and their nonclustered indexes. This means placing them on separate Quick I/O file systems mounted on separate volumes.

About tuning Solaris for Sybase

To achieve optimal performance using Veritas Storage Foundation for Sybase, certain Solaris parameters need to be tuned. Changing these parameters requires modifying the Solaris kernel settings (specified in the `/etc/system` file) and rebooting the system.

You can add or change these tuning parameters in the `/etc/system` file using a text editor. The following example shows the contents of an `/etc/system` file:

```
* start sybase *
set shmsys:shminfo_shmmax=512000000
set shmsys:shminfo_shmmin=1
set shmsys:shminfo_shmmni=100
set shmsys:shminfo_shmseg=200
*
set semsys:seminfo_semmnu=60
* end sybase *
```

Note: The settings for all tunable parameters depend on such factors as the size of your system and database, the database load, and the number of users. In some cases, we make suggestions for setting the parameters; however, you should always consult the Sybase Installation Guide for your system and version, and use the settings recommended by Sybase when provided.

maxuprc

This parameter sets the maximum number of processes that can be run concurrently by any one user. If you anticipate having a large number of users accessing the database concurrently, you may need to increase this parameter.

To increase the maxuprc parameter

1. Check the current setting for `maxuprc` as follows:

   ```
   # echo "maxuprc/D" | adb -k
   ```

2. Modify or add the `maxuprc` setting in the `/etc/system` file as follows:

   ```
   # set maxuprc=some_integer
   ```

shmmax

This parameter sets the maximum size (in bytes) of a single shared memory segment. See your database documentation for the recommended value.

shmmni

This parameter sets the number of shared memory identifiers. See your database documentation for the recommended value.

shmseg

This parameter sets the maximum number of shared memory segments that can be attached by a process. See your database documentation for the recommended value.

semmap

This parameter sets the number of entries in semaphore map. The memory space given to the creation of semaphores is taken from `semmap`, which is initialized with a fixed number of map entries based on the value of `semmap`. The value of `semmap` should never be larger than `semmni`. See your database documentation for the recommended value.

semmni

This parameter sets the number of semaphore set identifiers in the system. The `semmni` parameter determines the number of semaphore sets that can be created

at any one time, and may need to be set higher for a large database. See your database documentation for the recommended value.

semmns

This parameter sets the maximum number of semaphores in the system. The `semmns` parameter may need to be set higher for a large database. See your database documentation for the recommended value.

semmnu

This parameter sets the system-wide maximum number of undo structures. Setting this parameter value equal to `semmni` provides for an undo structure for every semaphore set. Semaphore operations performed using `semop`(2) can be undone if the process terminates, but an undo structure is required to guarantee it. See your database documentation for the recommended value of `semmnu`.

Appendix A

Veritas Storage Foundation for Sybase command line interface

This appendix includes the following topics:

- Overview of commands
- About the command line interface

Overview of commands

Veritas Storage Foundation for Sybase commands supported in the command line interface are located in the `/opt/VRTSsybed/bin` directory. Online manual pages for these commands are located in the `/opt/VRTS/man`directory. Follow the installation instructions provided in the *Veritas Storage Foundation Installation Guide* to ensure you can use these commands and view the online manual pages.

Table A-1 summarizes the commands available to you from the command line.

Table A-1 Veritas Storage Foundation for Sybase commands

Command	Description
`qio_convertdbfiles`	Converts VxFS files to Quick I/O files.
`qio_getdbfiles`	Extracts information on files used by the database and stores the names of these files in `mkqio.dat`. The `mkqio.dat` file is used by the `qio_convertdbfiles` command.

Table A-1 Veritas Storage Foundation for Sybase commands *(continued)*

Command	Description
`qio_recreate`	Automatically recreates Quick I/O files when the database is recovered. The command expects to find a mkqio.dat file in the directory where the `qio_recreate` command is run.
`edgetmsg2`	Manages message log files. You can use this utility to display and list message log files. You can also use this utility to write a message to a log file or to the console, or read the log file and print to the console.

About the command line interface

You can use the Veritas Storage Foundation for Sybase command line interface to perform administrative operations. For more detailed information about the commands and their syntax and available options, see the individual manual pages.

Converting VxFS files to Quick I/O using qio_convertdbfiles

After running `qio_getdbfiles`, you can use the `qio_convertdbfiles` command to convert database files to use Quick I/O. This command is for use with VxFS file systems only.

The `qio_convertdbfiles` command converts regular files or symbolic links that point to regular files on VxFS file systems to Quick I/O. The `qio_convertdbfiles` command converts only those files listed in the `mkqio.dat` file to Quick I/O. The `mkqio.dat` file is created by running `qio_getdbfiles`. It can also be created manually.

Before converting files, the following conditions must be met:

Prerequisites

- To use this command for Sybase, the `SYBASE` and `DSQUERY` environment variables must be set.
- You must be logged in as the database administrator.
- Remove any non-VxFS files from mkqio.dat before running `qio_convertdbfiles`. The `qio_convertdbfiles` command will display an error message if any of the database files in mkqio.dat are not on a VxFS file system.

Usage notes

- The `qio_convertdbfiles` command expects all files to be owned by the database administrator.
- Converting existing database files to Quick I/O is not recommended if the files are fragmented. In this case, it is recommended that you create new files with the `qiomkfile` command (these files are guaranteed not to be fragmented) and then convert the data from the old files (using a command such as `dd`).
- Ensure that the database is shut down before running `qio_convertdbfiles`.
- See the `qio_convertdbfiles`(1M) manual page for more information.

Note: The qio_getdbfiles and qio_convertdbfiles commands connect to the Sybase ASE server via a Sybase sa account. It is important to protect the sa password so that it is not visible to other users.

Table A-2 lists options for the `qio_convertdbfiles` command.

Table A-2 qio_convertdbfiles command options

Option	Description
`-T`	Forces the behavior for a specific database type. The database options that are supported are `ora`, `syb`, and `db2`. Use this option in environments with more than one type of database.
`-a`	Changes regular files to Quick I/O files using absolute pathnames. Use this option when symbolic links need to point to absolute pathnames. By default, relative pathnames are used.
`-f`	Reports on current fragmentation levels for files listed in `mkqio.dat`. Fragmentation is reported at four levels: not fragmented, slightly fragmented, fragmented, and highly fragmented.
`-h`	Displays a help message.
`-i`	Creates extra links for all database files and log files in the /dev directory to support the SAP command.
`-u`	Changes Quick I/O files back to regular files.

To convert VxFS files to Quick I/O files

1 After running the `qio_getdbfiles` command, shut down the database:

Warning: Running `qio_convertdbfiles` with any option except `-f` while the database is up and running can cause severe problems for your database, including data loss and corruption. Make sure the database is shut down before running the `qio_convertdbfiles` command.

---

2 Run the `qio_convertdbfiles` command to convert the list of files in `mkqio.dat` to Quick I/O files:

```
$ /opt/VRTSsybed/bin/qio_convertdbfiles
```

You must remove any non-VxFS files from `mkqio.dat` before running `qio_convertdbfiles`. The `qio_convertdbfiles` command will display an error message if any of the database files in `mkqio.dat` are not on a VxFS file system.

3 Restart the database to access these database files using the Quick I/O interface.

To undo a previous run of qio_convertdbfiles

- Use the `qio_convertdbfiles` as follows:

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

.dbfile::cdev:vxfs: --> dbfile
```

This reverts a previous run of `qio_convertdbfiles` and changes Quick I/O files back to regular VxFS files.

If the database is up and running, an error message will be displayed stating that you need to shut it down before you can run `qio_convertdbfiles`.

Identifying VxFS files to convert to Quick I/O using qio_getdbfiles

You can use the `qio_getdbfiles` command to identify VxFS files before converting them to Quick I/O files. Only VxFS files may be converted to Quick I/O.

The `qio_getdbfiles` command queries the database and gathers a list of datafiles to be converted to Quick I/O. The command requires direct access to the database.

Before using the `qio_getdbfiles` command, the following conditions must be met:

Prerequisites
- The SYBASE and DSQUERY environment variables must be set.
- You must be logged in as the database administrator.

Usage notes
- The -T option forces the behavior for a specific database type. The database options that are supported are ora, syb, and db2. Use this option in environments with more than one type of database.
- The -a option specifies that all datafiles should be included. By default, potential sparse files are excluded.
- See the qio_getdbfiles(1M) manual page for more information.
- See the qio_getdbfiles(1M) manual page for more information.

Note: The qio_getdbfiles command connects to the Sybase ASE server via a Sybase sa account. It is important to protect the sa password so that it is not visible to other users.

To identify the VxFS files to convert to Quick I/O

1 Use the qio_getdbfiles command as follows:

```
$ /opt/VRTSsybed/bin/qio_getdbfiles [-T syb] \
[-d <database_name>] [-m <master_device_pathname>]
```

where -T syb forces behavior for Sybase, <database_name> specifies the database device files, and <master_device_pathname> specifies the full path name of the master device for the Sybase ASE server.

The qio_getdbfiles command stores the filenames and file sizes in bytes in a file called mkqio.dat.

2 View the mkqio.dat file:

```
$ cat mkqio.dat
```

The mkqio.dat file contains the database filenames that can be converted to Quick I/O files. The format of the file is a list of paired file paths and file sizes. For example:

```
/database/dbfiles.001 1024000

/database/dbfiles.002 2048000
```

Recreating Quick I/O files using qio_recreate

You can use the command to automatically recreate Quick I/O files when the database is recovered.

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

Prerequisites

- The `SYBASE` and `DSQUERY` environment variables must be set.
- You must be logged in as the database administrator to use this command.

Usage notes

- The command expects to find a file named in the directory where the command is run. The mkqio.dat file contains a list of the Quick I/O files used by the database and their sizes. If the file is not in the directory, you will be prompted to create it using .
  See "Identifying VxFS files to convert to Quick I/O using qio_getdbfiles" on page 166.
- The `qio_recreate` command supports conventional Quick I/O files only (that is, Quick I/O files in the following form: `file --> .file::cdev:vxfs:`). In creating a Quick I/O file, the `qio_convertdbfiles` command renames the regular VxFS file, `file`, to `.file` with the Quick I/O extension (`:cdev::vxfs:`) and creates a symbolic link to it. By default, the symbolic link uses a relative path name.
- There are no options for the `qio_recreate` command and no output is returned when the command runs successfully.
- See the `qio_recreate`(1M) manual page for more information.

The qio_recreate command follows these rules in recreating Quick I/O files when a database is recovered:

- If a Quick I/O file (`.file::cdev:vxfs:`) is missing, then `qio_recreate` recreates it.
- If both a symbolic link (`file`) and its associated Quick I/O file (`.file::cdev:vxfs:`) are missing, `qio_recreate` recreates both the symbolic link and the Quick I/O file.
- If a symbolic link (`file`) from a regular VxFS file to its associated Quick I/O file (`.file::cdev:vxfs:`) is missing, then `qio_recreate` recreates the symbolic link.
- If a Quick I/O file (`.file::cdev:vxfs:`) is missing and the regular VxFS file that is symbolically linked to it is not the same one that originally created it, then `qio_recreate` issues a warning message and does not recreate the Quick I/O file.
- If a Quick I/O file (`.file::cdev: vxfs:`) is smaller than the size listed in `mkqio.dat`, `qio_recreate` issues a warning message.

To automatically recreate Quick I/O files when the database is recovered

- Use the `qio_recreate` command as follows:

```
$ /opt/VRTSsybed/bin/qio_recreate
```

Managing log files using edgetmsg2

You can use the `edgetmsg2` utility to manage message log files. You can use the `edgetmsg2` utility to write a message to a log file or to the console, read the log file and print to the console, and display the available log files.

Before managing log files with the `edgetmsg2` command, review the following information:

Prerequisites
- Log in as the dataserver administrator or root to use this command.

Usage notes
- The default log file for a database is located in the following directory:
  `/etc/vx/vxdbed/logs/sfua_`*`database`*`.log`
  where *`database`* is the `DSQUERY`.
- Be default, only messages with a severity equal to or greater than `ERROR` will be logged.
- See the `edgetmsg2`(1M) manual page for more information.

Table A-3 lists options for `edgetmsg2`.

Table A-3 edgetmsg2 options

Option	Description
`-s set_num`	Specifies the message catalogue set number. The default is 1.
`-M msgid[:severity]`	Specifies the message ID and severity to be printed.
`-f msg_catalog | logfile | log_directory`	Specifies the message catalogue path, log file, or log directory.
`-v severity | severity`	Overwrites the minimum log severity or creates a severity filter. The severity values are either 0-8 or 100-108.
`-p`	Pauses the cursor at the end of a display message. By default, a line feed is added to each display message. Use the `-p` option to indicate that no line feed is to be added.

Table A-3 edgetmsg2 options *(continued)*

Option	Description
`-o list [,suppress_time]`	Displays the content of a log file. You can specify `,suppress_time` to exclude time information in the utility output.
`-o report[,no_archive]`	Displays the available log files. You can specify ,no_archive to exclude log files from the utility output.
`-t from_time[,to_time]`	Reduces the length of the utility output by specifying the time range to include. This option must be used together with the `-o list` option. Use the following format: `yyy-mm-dd HH:MM:SS`.
`DSQUERY`	Specifies the `DSQUERY` for a Sybase dataserver.
`"default format string"`	Specifies the C language `printf()` format string.
`[args]`	Specifies arguments for the format string conversion characters.

To print a message

- Use the `edgetmsg2` command as follows:

```
$ /opt/VRTS/bin/edgetmsg2 [-s set_num] \

[-M msgid[:severity]] \
[-f msg_catalog] [-v severity] [-p] [-m value] \
["default format string" [args]]
```

To read a message log file

- Use the `edgetmsg2` command as follows:

```
$ /opt/VRTS/bin/edgetmsg2 -o list[,suppress_time] \
DSQUERY | [-f logfile] [-v severity] \
[-t from_time[,to_time]]
```

To list available log files

- Use the `edgetmsg2` command as follows:

```
$ /opt/VRTS/bin/edgetmsg2 -o report[,no_archive] \
[-f log_directory]
```

Glossary

address-length pair	Identifies the starting block address and the length of an extent (in file system or logical blocks).
asynchronous I/O	A format of I/O that performs non-blocking reads and writes. This enables the system to handle multiple I/O requests simultaneously.
atomic operation	An operation that either succeeds completely or fails and leaves everything as it was before the operation was started. If the operation succeeds, all aspects of the operation take effect at once and the intermediate states of change are invisible. If any aspect of the operation fails, then the operation aborts without leaving partial changes.
block map	A file system is divided into fixed-size blocks when it is created. As data is written to a file, unused blocks are allocated in ranges of blocks, called extents. The extents are listed or pointed to from the inode. The term used for the data that represents how to translate an offset in a file to a file system block is the “block map” for the file.
boot disk	A disk used for booting an operating system.
buffered I/O	A mode of I/O operation (where I/O is any operation, program, or device that transfers data to or from a computer) that first transfers data into the Operating System buffer cache.
cache	Any memory used to reduce the time required to respond to an I/O request. The read cache holds data in anticipation that it will be requested by a client. The write cache holds data written until it can be safely stored on non-volatile storage media.
Cached Quick I/O	Cached Quick I/O allows databases to make more efficient use of large system memory while still maintaining the performance benefits of Quick I/O. Cached Quick I/O provides an efficient, selective buffering mechanism to complement asynchronous I/O.
cluster	A set of hosts that share a set of disks.
cluster-shareable disk group	A disk group in which the disks are shared between more than one host.
cold backup	The process of backing up a database that is not in active use.
command launcher	A graphical user interface (GUI) window that displays a list of tasks that can be performed by Veritas Volume Manager or other objects. Each task is listed with the object type, task (action), and a description of the task. A task is launched by

	clicking on the task in the Command Launcher. concatenation A Veritas Volume Manager layout style characterized by subdisks that are arranged sequentially and contiguously.
concurrent I/O	A form of Direct I/O that does not require file-level write locks when writing to a file. Concurrent I/O allows the relational database management system (RDBMS) to write to a given file concurrently.
configuration database	A set of records containing detailed information on existing Veritas Volume Manager objects (such as disk and volume attributes). A single copy of a configuration database is called a configuration copy.
copy-on-write	A technique for preserving the original of some data. As data is modified by a write operation, the original copy of data is copied.
database	A database is a collection of information that is organized in a structured fashion. Two examples of databases are Relational Databases (such as Oracle, Sybase, or DB2), where data is stored in tables and generally accessed by one or more keys and Flat File Databases, where data is not generally broken up into tables and relationships. Databases generally provide tools and/or interfaces to retrieve data.
dataserver	A logical concept of a Sybase instance. A Sybase instance contains databases and daemon processes that manage the data. A Sybase dataserver manages Sybase system databases and user created databases. Each Sybase datasever is uniquely named when it is created.
Decision Support Systems	Decision Support Systems (DSS) are computer-based systems used to model, identify, and solve problems, and make decisions.
defragmentation	The act of reorganizing data to reduce fragmentation. Data in file systems become fragmented over time.
device file	A block- or character-special file located in the /dev directory representing a device.
device name	The name of a device file, which represents a device. AIX syntax is Disk_#; HP-UX syntax is c#t#d#; Linux syntax is sda, where "a" could be any alphabetical letter; Solaris syntax is c#t#d#s#.
direct I/O	An unbuffered form of I/O that bypasses the kernel's buffering of data. With direct I/O, data is transferred directly between the disk and the user application.
Dirty Region Logging	The procedure by which the Veritas Volume Manager monitors and logs modifications to a plex. A bitmap of changed regions is kept in an associated subdisk called a log subdisk.
disk access name	The name used to access a physical disk, such as Disk_1 on an AIX system, c1t1d1 on an HP-UX system, sda on a Linux system, or c0t0d0s0 on a Solaris system. The term device name can also be used to refer to the disk access name.

disk array A collection of disks logically and physically arranged into an object. Arrays provide benefits including data redundancy and improved performance.

disk cache A section of RAM that provides a cache between the disk and the application. Disk cache enables the computer to operate faster. Because retrieving data from hard disk can be slow, a disk caching program helps solve this problem by placing recently accessed data in the disk cache. Next time that data is needed, it may already be available in the disk cache; otherwise a time-consuming operation to the hard disk is necessary.

disk group A collection of disks that share a common configuration.

A disk group configuration is a set of records containing detailed information on existing Veritas Volume Manager objects (such as disk and volume attributes) and their relationships. Each disk group has an administrator-assigned name and an internally defined unique ID. The root disk group (rootdg) is a special private disk group

disk name A Veritas Volume Manager logical or administrative name chosen for the disk, such as disk03. The term disk media name is also used to refer to the disk name.

DMP See “Dynamic Multipathing.”

DSS See “Decision Support Systems.”

Dynamic Multipathing Dynamic Multipathing (DMP) is a Veritas Volume Manager feature that allows the use of multiple paths to the same storage device for load balancing and redundancy.

error handling Routines in a program that respond to errors. The measurement of quality in error handling is based on how the system informs the user of such conditions and what alternatives it provides for dealing with them.

evacuate Moving subdisks from the source disks to target disks.

exabyte A measure of memory or storage. An exabyte is approximately 1,000,000,000,000,000,000 bytes (technically 2 to the 60th power, or 1,152,921,504,606,846,976 bytes). Also EB.

extent A logical database attribute that defines a group of contiguous file system data blocks that are treated as a unit. An extent is defined by a starting block and a length.

extent attributes The extent allocation policies associated with a file and/or file system. For example, see “address-length pair.”

failover The act of moving a service from a failure state back to a running/available state. Services are generally applications running on machines and failover is the process of restarting these applications on a second system when the first has suffered a failure.

file system	A collection of files organized together into a structure. File systems are based on a hierarchical structure consisting of directories and files.
file system block	The fundamental minimum size of allocation in a file system.
fileset	A collection of files within a file system.
fixed extent size	An extent attribute associated with overriding the default allocation policy of the file system.
fragmentation	Storage of data in non-contiguous areas on disk. As files are updated, new data is stored in available free space, which may not be contiguous. Fragmented files cause extra read/write head movement, slowing disk accesses.
gigabyte	A measure of memory or storage. A gigabyte is approximately 1,000,000,000 bytes (technically, 2 to the 30th power, or 1,073,741,824 bytes). Also GB, Gbyte, and G-byte.
high availability (HA)	The ability of a system to perform its function continuously (without significant interruption) for a significantly longer period of time than the combined reliabilities of its individual components. High availability is most often achieved through failure tolerance and inclusion of redundancy; from redundant disk to systems, networks, and entire sites.
hot backup	The process of backing up a database that is online and in active use.
hot pluggable	To pull a component out of a system and plug in a new one while the power is still on and the unit is still operating. Redundant systems can be designed to swap disk drives, circuit boards, power supplies, CPUs, or virtually anything else that is duplexed within the computer. Also known as hot swappable.
hot-relocation	A Veritas Volume Manager technique of automatically restoring redundancy and access to mirrored and RAID-5 volumes when a disk fails. This is done by relocating the affected subdisks to disks designated as spares and/or free space in the same disk group.
inode list	An inode is an on-disk data structure in the file system that defines everything about the file, except its name. Inodes contain information such as user and group ownership, access mode (permissions), access time, file size, file type, and the block map for the data contents of the file. Each inode is identified by a unique inode number in the file system where it resides. The inode number is used to find the inode in the inode list for the file system. The inode list is a series of inodes. There is one inode in the list for every file in the file system.
intent logging	A logging scheme that records pending changes to a file system structure. These changes are recorded in an intent log.
interrupt key	A way to end or break out of any operation and return to the system prompt by pressing Ctrl-C.

kernel asynchronous I/O A form of I/O that performs non-blocking system level reads and writes. This enables the system to handle multiple I/O requests simultaneously.

kilobyte A measure of memory or storage. A kilobyte is approximately a thousand bytes (technically, 2 to the 10th power, or 1,024 bytes). Also KB, Kbyte, kbyte, and K-byte.

large file A file more than two gigabytes in size. An operating system that uses a 32-bit signed integer to address file contents will not support large files; however, the Version 4 disk layout feature of VxFS supports file sizes of up to two terabytes.

large file system A file system more than two gigabytes in size. VxFS, in conjunction with VxVM, supports large file systems.

latency The amount of time it takes for a given piece of work to be completed. For file systems, this typically refers to the amount of time it takes a given file system operation to return to the user. Also commonly used to describe disk seek times.

load balancing The tuning of a computer system, network tuning, or disk subsystem in order to more evenly distribute the data and/or processing across available resources. For example, in clustering, load balancing might distribute the incoming transactions evenly to all servers, or it might redirect them to the next available server.

load sharing The division of a task among several components without any attempt to equalize each component's share of the load. When several components are load sharing, it is possible for some of the shared components to be operating at full capacity and limiting performance, while others components are under utilized.

Logical Unit Number A method of expanding the number of SCSI devices that can be placed on one SCSI bus. Logical Unit Numbers address up to seven devices at each SCSI ID on an 8-bit bus or up to 15 devices at each ID on a 16-bit bus.

logical volume See "volume."

LUN See "Logical Unit Number."

master node A computer which controls another computer or a peripheral.

megabyte A measure of memory or storage. A megabyte is approximately 1,000,000 bytes (technically, 2 to the 20th power, or 1,048,576 bytes). Also MB, Mbyte, mbyte, and K-byte.

metadata Data that describes other data. Data dictionaries and repositories are examples of metadata. The term may also refer to any file or database that holds information about another database's structure, attributes, processing, or changes.

mirror A duplicate copy of a volume and the data therein (in the form of an ordered collection of subdisks). Each mirror is one copy of the volume with which the mirror is associated. The terms mirror and plex can be used synonymously.

mirroring	A layout technique that mirrors the contents of a volume onto multiple plexes. Each plex duplicates the data stored on the volume, but the plexes themselves may have different layouts.
mount point	The directory path name at which a file system attaches to the file system hierarchy.
multithreaded	Having multiple concurrent or pseudo-concurrent execution sequences. Used to describe processes in computer systems. Multithreaded processes are one means by which I/O request-intensive applications can use independent access to volumes and disk arrays to increase I/O performance.
NBU	See "Veritas NetBackup (NBU)."
node	One of the hosts in a cluster.
object (VxVM)	An entity that is defined to and recognized internally by the Veritas Volume Manager. The VxVM objects include volumes, plexes, subdisks, disks, and disk groups. There are two types of VxVM disk objects—one for the physical aspect of the disk and the other for the logical aspect of the disk.
OLTP	See "Online Transaction Processing."
online administration	An administrative feature that allows configuration changes without system or database down time.
Online Transaction Processing	A type of system designed to support transaction-oriented applications. OLTP systems are designed to respond immediately to user requests and each request is considered to be a single transaction. Requests can involve adding, retrieving, updating or removing data.
paging	The transfer of program segments (pages) into and out of memory. Although paging is the primary mechanism for virtual memory, excessive paging is not desirable.
parity	A calculated value that can be used to reconstruct data after a failure. While data is being written to a RAID-5 volume, parity is also calculated by performing an exclusive OR (XOR) procedure on data. The resulting parity is then written to the volume. If a portion of a RAID-5 volume fails, the data that was on that portion of the failed volume can be recreated from the remaining data and the parity.
partition	The logical areas into which a disk is divided.
persistence	Information or state that will survive a system reboot or crash.
petabyte	A measure of memory or storage. A petabyte is approximately 1,000 terabytes (technically, 2 to the 50th power).
plex	A duplicate copy of a volume and its data (in the form of an ordered collection of subdisks). Each plex is one copy of a volume with which the plex is associated. The terms mirror and plex can be used synonymously.

preallocation	Prespecifying space for a file so that disk blocks will physically be part of a file before they are needed. Enabling an application to preallocate space for a file guarantees that a specified amount of space will be available for that file, even if the file system is otherwise out of space.
Quick I/O	Quick I/O presents a regular Veritas File System file to an application as a raw character device. This allows Quick I/O files to take advantage of asynchronous I/O and direct I/O to and from the disk device, as well as bypassing the UNIX single-writer lock behavior for most file system files.
Quick I/O file	A regular UNIX file that is accessed using the Quick I/O naming extension (::cdev:vxfs:).
RAID	A Redundant Array of Independent Disks (RAID) is a disk array set up with part of the combined storage capacity used for storing duplicate information about the data stored in that array. This makes it possible to regenerate the data if a disk failure occurs.
repository	A repository holds the name, type, range of values, source, and authorization for access for each data element in a database. The database maintains a repository for administrative and reporting use.
root disk	The disk containing the root file system.
root disk group	A special private disk group on the system. The root disk group is named rootdg. However, starting with the 4.1 release of Veritas Volume Manager, the root disk group is no longer needed.
root file system	The initial file system mounted as part of the UNIX kernel startup sequence.
script	A file, containing one or more commands that can be run to perform processing.
shared disk group	A disk group in which the disks are shared by multiple hosts (also referred to as a cluster-shareable disk group).
sector	A minimal unit of the disk partitioning. The size of a sector can vary between systems. A sector is commonly 512 bytes.
segment	Any partition, reserved area, partial component, or piece of a larger structure.
SGA	See "System Global Area."
single threading	The processing of one transaction to completion before starting the next.
slave node	A node that is not designated as a master node.
slice	The standard division of a logical disk device. The terms partition and slice can be used synonymously.
snapped file system	A file system whose exact image has been used to create a snapshot file system.

snapped volume	A volume whose exact image has been used to create a snapshot volume.
snapshot	A point-in-time image of a volume or file system that can be used as a backup.
snapshot file system	An exact copy of a mounted file system, at a specific point in time, that is used for online backup. A snapshot file system is not persistent and it will not survive a crash or reboot of the system.
snapshot volume	An exact copy of a volume, at a specific point in time. The snapshot is created based on disk mirroring and is used for online backup purposes.
spanning	A layout technique that permits a volume (and its file system or database) too large to fit on a single disk to distribute its data across multiple disks or volumes.
storage class	Set of volumes with the same volume tag.
stripe	A set of stripe units that occupy the same positions across a series of columns in a multi-disk layout.
stripe unit	Equally sized areas that are allocated alternately on the subdisks (within columns) of each striped plex. In an array, this is a set of logically contiguous blocks that exist on each disk before allocations are made from the next disk in the array.
stripe unit size	The size of each stripe unit. The default stripe unit size for VxVM is 32 sectors (16K). For RAID 0 stripping, the stripe unit size is 128 sectors (64K). For VxVM RAID 5, the stripe unit size is 32 sectors (16K). A stripe unit size has also historically been referred to as a stripe width.
striping	A layout technique that spreads data across several physical disks using stripes. The data is allocated alternately to the stripes within the subdisks of each plex.
subdisk	A consecutive set of contiguous disk blocks that form a logical disk segment. Subdisks can be associated with plexes to form volumes.
superuser	A user with unlimited access privileges who can perform any and all operations on a computer. In UNIX, this user may also be referred to as the “root” user. On Windows/NT, it is the “Administrator.”
tablespace	A tablespace is a storage structure (containing tables, indexes, large objects, and long data) that allows you to assign the location of database and table data directly onto containers. Tablespaces reside in database partition groups.
terabyte	A measure of memory or storage. A terabyte is approximately 1,000,000,000,000 bytes (technically, 2 to the 40th power, or 1,000 GB). Also TB.
throughput	A measure of work accomplished in a given amount of time. For file systems, this typically refers to the number of I/O operations in a given period of time.
UFS	The Solaris name for a file system type derived from the 4.2 Berkeley Fast File System.

unbuffered I/O	I/O that bypasses the file system cache for the purpose of increasing I/O performance (also known as direct I/O).
Veritas Enterprise Administrator	Application that is required to access graphical user interface (GUI) functionality.
Veritas NetBackup (NBU)	A product that lets you back up, archive, and restore files, directories, or raw partitions that reside on your client system.
Veritas Volume Replicator (VVR)	A feature of Veritas Volume Manager, VVR is a data replication tool designed to contribute to an effective disaster recovery plan.
volume	A logical disk device that appears to applications, databases, and file systems as a physical disk partition. A logical disk can encompass multiple or one to many physical volumes.
volume layout	A variety of layouts that allows you to configure your database to meet performance and availability requirements. This includes spanning, striping (RAID-0), mirroring (RAID-1), mirrored stripe volumes (RAID-0+1), striped mirror volumes (RAID-1+0), and RAID 5.
volume manager objects	Volumes and their virtual components. See “object (VxVM).”
VVR	See “Veritas Volume Replicator (VVR).”
vxfs or VxFS	The acronym for Veritas File System.
vxvm or VxVM	The acronym for Veritas Volume Manager.

Index

Numerics

$DSQUERY 67
$LD_LIBRARY_PATH 67
$PATH 67
$SYBASE 67

A

absolute path names
    using with Quick I/O 68
absolute pathnames
    use with symbolic links 66
accessing
    Quick I/O files with symbolic links 66
allocating file space 63
allocation policies
    block-based 25
    UFS 25
analyzing I/O statistics 86
asynchronous I/O 52, 58
availability
    using mirroring for 17

B

backups
    implementing online 113
balancing I/O load 158
benefits of Concurrent I/O 94
benefits of Quick I/O 52, 57

C

cache advisory
    checking setting for 91
cache hit ratio
    calculating 87
Cached Quick I/O
    caching statistics 156
    customizing 88
    determining files to use 86
    disabling individual files 89
    enabling individual files 89
Cached Quick I/O *(continued)*
    making settings persistent 89
    overview 24
    prerequisite for enabling 82
calculating cache hit ratio 87
changing file sizes 63
chgrp command 61
chmod command
    commands
        chmod 82
chown command 61
    commands
        chown 82
Cluster Volume Manager 23
collecting I/O statistics 86
commands
    chgrp 61
    chown 61
    edgetmsg2 164
    fsadm 26, 49
    fsadm command 76
    grep 84
    ls 74
    mkfs 28, 41–42
    mount 28, 42, 54, 60
    overview 163
    qio_convertdbfiles 67, 71, 163–164
    qio_getdbfiles 67, 70, 163, 166
    qio_recreate 164, 167, 169
    qioadmin 88
    qiomkfile 75–76
    qiostat 86, 156–157
    setext 61
    umount 44
    vxtunefs 90
    vxupgrade 98
concatenation 16
Concurrent I/O
    benefits 94
    disabling 95
    enabling 94

converting
Quick I/O files back to regular files Quick I/O
converting back to regular files 68
regular files to Quick I/O files 71
CREADs 87
creating
a volume 37
Quick I/O files 64
symbolic links to access Quick I/O files 63
cron 149
cross-platform data sharing 27
customizing Cached Quick I/O 88

D

data change object 21
data redundancy 17
data warehousing 18
database
specifying type for Quick I/O 68
tuning 160
database performance
using Quick I/O 52, 58
dataserver buffer cache 79
DCO 21
log volume 21
decision support
implementing 139
default_indir_size tunable parameter 152
defragmentation 26
extent 150
scheduling 150
utility 26
determining
if Quick I/O installed and enabled 74
device interface 22
direct I/O 53, 59
direct-write
copy-behind 81
Dirty Region Logging 19
dirty region logging 19, 37
dirty volumes 18
disabling Cached Quick I/O for a file 89
disabling Concurrent I/O 95
disabling qio_cache_enable flag 83
discovered_direct_iosize tunable parameter 152
disk arrays 15
DMP-supported 22
disk group
naming a disk group 33
disk groups
about 15
adding disks 35
configuration guidelines 33
creating
using the command line 34
defined 15
join 106
joining 105
limitations of move 106
recovery from failed reconfiguration 106
split 106
split and join 21
disk space allocation 25
disks
adding to a disk group 35
failure and hot-relocation 21
dissociating snapshot volumes 145
DMP 22
DMP-supported disk arrays 22
double buffering 53, 59, 80
DRL 19, 37. *See* Dirty Region Logging
DSS workloads
guidelines 37
dsync flag 67
dynamic LUN expansion 22
Dynamic Multipathing 22

E

edgetmsg2 command 164
enabling
Quick I/O 54, 60
enabling Cached Quick I/O for a file 89
enabling Concurrent I/O 94
enabling qio_cache_enable flag 82
excessive reads or writes 158
exclusive OR 18
expansion
file system 149
extending a file 63
extending Quick I/O files 75
extent-based allocation 25
extracting file list for Quick I/O conversion 70

F

fast file system 25
fast recovery 37

FastResync
non-persistent 20
operation with off-host processing 104
persistent 20
use with snapshots 20
FastReysnc 20
file
space allocation 63
file fragmentation
reporting on 68
file system locking 53, 59
file systems
configuration guidelines 40
growing to accommodate Quick I/O files 75
increasing the size 49
mounting 43
overview 23
resizing 26, 49
running databases on 24
unmounting 44–45
fragmentation 26, 45
controlling 46
monitoring 46, 149
reorganization facilities 149
reporting 149
types 46
fragmented file system
characteristics 150
free space 21, 149
monitoring 149
fsadm
reporting extent fragmentation 150
scheduling 150
fsadm command 26, 49, 76
largefiles option 43
fsadm utility 26, 50
fsapadm command 43
fsvoladm command 43

G

grep command 84
growing
file systems 75
Quick I/O files 75
guidelines
creating file systems 40
disk groups 33
for DSS workloads 37
for OLTP workloads 37
guidelines *(continued)*
striped volumes 37
volumes 37

H

High Availability (HA)
overview 12
hot-relocation 21

I

I/O
asynchronous 52, 58
Cached Quick I/O 24
direct 53, 59
kernel asynchronous 52, 58
load balancing 158
performance data 157
Quick I/O 23
sequential 25
statistics
obtaining 148
improving
database performance 52, 58
initial_extent_size tunable parameter 153

K

kernel asynchronous I/O 52, 58
kernel settings
modifying 160
kernel write locks 53, 59

L

large file systems 27
support for 42
large files
enabling 43
support for 28, 42
largefiles option 43
list file for Quick I/O conversion 70
ls command 74

M

master device path 68
max_direct_iosize tunable parameter 153
max_diskq tunable parameter 153
max_seqio_extent_size tunable parameter 154
maxuprc 161

memory
persistence of FastResync in 20
mirrored volume snapshots 20
mirrored-stripe volumes 17
mirroring 13, 17
choosing 36
defined 17
mirroring and striping data 17
mkfs command 28, 41–42
mkqio.dat file 70–71, 78
mkqio.sh script options
report on file fragmentation 68
monitoring fragmentation 149
mount command 28, 42, 54, 60
mounting
file systems 28
mounting file systems 43
moving hot files or busy file systems 158
multi-volume support 27, 43
fsvoladm command 43
mutli-volume support
fsapadm command 43

N

naming convention
for Quick I/O files 54, 59
nolargefiles option 28, 42
non-persistent FastResync 20

O

OLTP. *See* online transaction processing
OLTP workloads
guidelines 37
online backups
implementing 113
online relayout 18
online transaction processing 18, 53, 59
options
largefiles and nolargefiles 28
overview
of Quick I/O 23

P

parallel data transfer
using striping for 17
parameters
default 151
tunable 151
parameters *(continued)*
tuning 150
parity 18
performance
obtaining statistics for volumes 148
RAID-5 18
performance data
using 157
performance tuning
for databases 160
list of guides 147
persistence
for Cached Quick I/O settings 89
persistent FastResync 20
PREADs 87
preallocating space for Quick I/O files 53, 59, 61

Q

qio_cache_enable flag
disabling 83
enabling 82
qio_cache_enable tunable parameter 154
qio_convertdbfiles command 67, 71, 163–164
qio_getdbfiles command 67, 70, 163, 166
qio_recreate command 164, 167, 169
qioadmin command 88
qiomkfile command 75–76
options for creating files
symbolic links 63
qiostat
output of 86
qiostat command 86, 156–157
QoSS. *See* quality of storage service
quality of storage service 27
Quick I/O
accessing regular VxFS files as 65
benefits 52, 57
converting files to 71
converting VxFS files to 164, 166
determining file fragmentation before converting 68
determining status 74
enabling 54, 60
environment variable requirements 67
extending files 75
extracting file list for conversion 70
improving database performance with 52, 58
list file for conversion 70
naming convention for files 54, 59

Quick I/O *(continued)*
overview 23
performance improvements 81
preallocating space for files 53, 59, 61
recreating files 167, 169
requirements 53, 59
setting environment variables for 69
showing resolution to a raw device 75
specifying database name fof 67
using relative and absolute pathnames 66
quotas 28

R

RAID 15
RAID-0 16
RAID-0+1 17
RAID-1 17
RAID-1+0 17
RAID-5 18, 37
choosing 36
performance 36
RAID-5 log 37
raw devices
running databases on 24
read-ahead algorithm
for Cached Quick I/O 81
read_nstream tunable parameter 151
read_pref_io tunable parameter 151
recreating
data using RAID-5 18
redo logs
configuration guidelines 40
creating a file system 40
relative pathnames
use with symbolic links 66
relayout 18
reliability
using mirroring for 17
removing non-VxFS files from mkqio.dat file 71
removing snapshot volumes 145
report
extent fragmentation 149
requirements of Quick I/O 53, 59
resizing a file 63
resizing file systems 49
resizing utility 26
resynchronization
using DRL logs 37
using RAID-5 logs 37
resynchronizing
volumes 18
RUN_SERVER file 68

S

sa_password 68
SCSI devices 22
selecting volume layouts 35
semmap 161
semmni 161
semmns 162
semmnu 162
sequential I/O
using extent-based allocation 25
sequential read/writes
using spanning for 16
setext command 61
setting environment variables for Quick I/O 69
settings
making Cached Quick I/O persistent 83
shmmax 161
shmmni 161
shmseg 161
showing
Quick I/O file resolved to raw device 75
snapshot volumes
dissociating 145
using the command line 145
removing 145
snapshots
and FastResync 20
spanning 13, 16
defined 16
spare disks 21
specifying
master device path 68
specifying database name for Quick I/O 67
statistics
volume I/O 148
Storage Checkpoints
file system restores 28
Storage Expert 22
stripe unit sizes
choosing 36
stripe units 16
striped volumes 37
configuration guidelines 37
striping 13, 16
defined 16

striping and mirroring data 17
support for large files 28
Sybase
    environment variables for Quick I/O 67
symbolic links
    advantages and disadvantages 66
    to access Quick I/O files 66
system buffer cache 81

T

tunable I/O parameters 151
    default_indir_size 152
    discovered_direct_iosize 152
    initial_extent_size 153
    max_direct_iosize 153
    max_diskq 153
    max_seqio_extent_size 154
    qio_cache_enable 154
    read_nstream 151
    read_pref_io 151
    write_nstream 152
    write_pref_io 151
    write_throttle 155
tunefstab file
    adding tuning parameters to 83
Tuning
    file I/O statistics 156
    VxFS 149
    VxFS I/O parameters 151
tuning
    for database performance 160
    vxfs 149
    VxVM 147
tuning I/O parameters 150
tuning parameters
    adding to tunefstab file 83

U

umount command 44
unmounting
    a file system 45
unmounting file systems 44
upgrade
    from raw devices 99
upgrading
    from earlier VxFS layout versions 98
    from UFS 97
using performance data 157
utilities. *See* commands
    fsadm 26, 50
        See also commands
    online administration 26

V

verifying caching using vxfstune parameters 84
verifying vxtunefs system parameters 84
Veritas FastResync. *See* FastResync
Veritas File System
    cluster functionality 29
    cross-platfrom data sharing 27
    defragmentation utility 26
    fast file system 25
    multi-volume support 27
    online administration utilities 26
    overview 23
    quality of storage service 27
    quotas 28
    resizing utility 26
    support for large file systems 27
Veritas Volume Manager 13
    and RAID 15
    objects 15
    overview 13
Veritas Volume Replicator 23
volume layouts 15
    changing 18
    concatenation 16
    mirrored-stripe 17
    mirroring 17
    RAID-5 18
    selecting 35
    spanning 16
    striping 16
volume resynchronization 18
volume snapshots. *See* snapshots
volumes
    about 14
    configuration guidelines 37
    creating 37
        using the command line 38
    definition 15
    layouts 15
    marked as dirty 18
    obtaining performance statistics 148
    resynchronizing 18
vxassist
    used to remove DCOs from volumes 111

VxFS
    resizing utility 26
    tuning 149
VxFS files
    converting to Quick I/O 164
VxFS files,converting to Quick I/O 166
VxFS.. *See* Veritas File System
vxstat
    used to obtain volume performance
        statistics 148
vxtunefs command 90
    commands
        vxtunefs 84
vxupgrade command 98
VxVM . *See* Veritas Volume Manager
    overview 13
    tuning 147

W

workloads
    write-intensive 37
write_nstream tunable parameter 152
write_pref_io tunable parameter 151
write_throttle tunable parameter 155

X

XOR . *See* exclusive OR