VERITAS™ Storage Foundation 4.0 for Sybase

Database Administrator’s Guide

Solaris

N09986F

January 2004
Disclaimer
The information contained in this publication is subject to change without notice. VERITAS Software Corporation makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. VERITAS Software Corporation shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this manual.

VERITAS Copyright
Copyright © 2000-2004 VERITAS Software Corporation. All rights reserved. VERITAS, the VERITAS logo, and all other VERITAS product names and slogans are trademarks or registered trademarks of VERITAS Software Corporation. VERITAS and the VERITAS Logo Reg. U.S. Pat & Tm. Off. Other product names and/or slogans mentioned herein may be trademarks or registered trademarks of their respective companies.

VERITAS Software Corporation
350 Ellis St.
Mountain View, CA 94043
Phone 650–527–8000
Fax 650–527–8050
www.veritas.com

Third-Party Copyrights

Data Encryption Standard (DES)
Copyright© 1990 Dennis Ferguson. All rights reserved. Commercial use is permitted only if products which are derived from or include this software are made available for purchase and/or use in Canada. Otherwise, redistribution and use in source and binary forms are permitted. Copyright© 1985, 1986, 1987, 1988, 1990 by the Massachusetts Institute of Technology. All Rights Reserved.

Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting. WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.
Contents

Preface ................................................................. ix
  Scope and Audience ........................................... ix
  Organization ...................................................... x
  Related Documents ........................................... x
  Conventions ...................................................... xiii
    Typographic and Symbolic .................................... xiii
    Notes and Cautions ........................................... xiv
  Getting Help .................................................... xiv

Chapter 1. Introducing VERITAS Storage Foundation for Sybase ............... 1
  VERITAS Storage Foundation for Sybase ........................... 2
    Standard Edition/Enterprise Edition ............................ 2
    Features ......................................................... 2
  VERITAS Volume Manager ....................................... 4
    Volumes ......................................................... 5
    Disk Groups .................................................... 5
    Volume Layouts ................................................ 6
    Online Relayout ............................................... 8
    Volume Resynchronization .................................... 8
    Dirty Region Logging ......................................... 9
    Volume Snapshots ........................................... 9
    VERITAS FastResync (Optional) ............................... 9
    Disk Group Split and Join (Optional) .......................... 10
    Hot-Relocation ............................................... 11
Chapter 2. Setting Up Dataservers ................................. 19
Setting Up a New Database ........................................ 20
Setting Up a Disk Group ............................................ 21
Disk Group Configuration Guidelines ............................. 21
Creating a Disk Group for a Dataserver ......................... 22
Adding Disks to a Dataserver Disk Group ....................... 25
Selecting a Volume Layout ......................................... 27
Choosing Appropriate Stripe Unit Sizes ......................... 27
Choosing Between Mirroring and RAID-5 ........................ 27
Preparing Hosts for Database Backup or Off-Host Processing ........................................... 105
  Single-Host Configuration .................................................................................................. 105
  Two-Host Configuration .................................................................................................... 105
  Upgrading Existing Volumes to Use VxVM 4.0 Features .................................................... 107
Sybase Adaptive Server Enterprise 12.5 Quiesce Feature ..................................................... 109
Setting Up Volume Snapshots with Sybase ASE 12.5 Server .............................................. 110
Implementing Online Backup or Off-host Processing ......................................................... 111
Creating a Warm Standby Server .......................................................................................... 116
Resynchronizing the Snapshot to Your ASE Dataserver .................................................... 125
Recovering the Database from a Backup Image ................................................................... 128
Refreshing a Snapshot Database Image ............................................................................... 131
Dissociating a Snapshot Volume .......................................................................................... 133
Removing a Snapshot Volume .............................................................................................. 134

Chapter 7. Tuning for Performance ....................................................................................... 135
  Tuning VxVM ...................................................................................................................... 136
    Obtaining Volume I/O Statistics ......................................................................................... 136
  Tuning VxFS ....................................................................................................................... 138
    Monitoring Free Space ..................................................................................................... 138
    Tuning VxFS I/O Parameters ........................................................................................... 139
    Tunable VxFS I/O Parameters ......................................................................................... 140
    Obtaining File I/O Statistics using the Quick I/O Interface ............................................. 144
    Using I/O Statistics Data .................................................................................................. 145
    Interpreting I/O Statistics ............................................................................................... 146
  Tuning Sybase Dataservers ................................................................................................. 147
    Tuning Sybase tempdb Database ..................................................................................... 147
    Tuning Sybase sybsecurity Database .............................................................................. 148
    Placement of the Transaction Logs ................................................................................ 148
    Database Device Layout .................................................................................................. 148
    Nonclustered Indexes Placement .................................................................................... 148
Preface

VERITAS Storage Foundation for Sybase, formerly known as VERITAS Database Edition, is an integrated set of system software enhancements and configuration guidelines that combine to help Sybase database administrators configure a database system with high performance, availability, manageability, and reliability.

Scope and Audience

The Database Administrator’s Guide is intended for database and system administrators responsible for configuring and maintaining Sybase databases with VERITAS Storage Foundation for Sybase, which includes:

- VERITAS Volume Manager™ (VxVM)
- VERITAS File System™ (VxFS) with Quick I/O™ for Databases
- VERITAS Enterprise Administrator™ (VEA)

This guide assumes that the administrator has a:

- Basic understanding of system and database administration
- Working knowledge of the operating system
- General understanding of file systems
Organization

This guide is organized as follows:

◆ **Chapter 1, “Introducing VERITAS Storage Foundation for Sybase” on page 1** introduces the features and characteristics of VERITAS Storage Foundation for Sybase.

◆ **Chapter 2, “Setting Up Dataservers” on page 19** discusses how to select volume layouts and create optimal file system and database configurations.

◆ **Chapter 3, “Using VERITAS Quick I/O” on page 49** describes how to set up and use VERITAS Quick I/O.

◆ **Chapter 4, “Using VERITAS Cached Quick I/O” on page 77** describes how to set up and use VERITAS Cached Quick I/O.

◆ **Chapter 5, “Converting Existing Database Configurations to VxFS” on page 93** discusses how to migrate existing databases to VxFS file systems.

◆ **Chapter 6, “Using Volume Snapshots for Dataserver Backup and Off-Host Processing” on page 99** describes the online snapshot facilities provided with VxFS and VxVM.

◆ **Chapter 7, “Tuning for Performance” on page 135** provides tuning tips and describes commands you can use to monitor and tune dataserver performance.

◆ **Appendix A, “VERITAS Storage Foundation for Sybase Command Line Interface” on page 153** describes the command line interface to key Quick I/O operations.

◆ The “Glossary” on page 161 provides terminology and general background information for Sybase database administration.

Related Documents

The following documents provide related information:

◆ **Release Notes** provide important, up-to-date, and release-specific information for the various products bundled with this product. Reading all of the Release Notes before installing or using any VERITAS products is recommended. Portable Document Format (.pdf) versions of these notes are provided on the product disc.

◆ The **VERITAS Storage Foundation for Sybase Installation Guide** provides instructions on how to install the product, which includes VERITAS File System with Quick I/O and QuickLog, and VERITAS Volume Manager.

◆ The **VERITAS File System Administrator’s Guide** provides conceptual information about the VERITAS File System, describes how to use associated commands and utilities, and provides file system error message information and performance tuning.
guidelines. This guide also includes an appendix that describes how to perform common file system tasks with the VERITAS File System and provides examples of typical VERITAS File System operations.

◆ The VERITAS Volume Manager Administrator’s Guide describes the procedures and concepts involved with volume management and system administration using the VERITAS Volume Manager. This guide includes how to take advantage of various VERITAS Volume Manager disk administration features, how to use VERITAS Volume Manager commands to create and manipulate objects, how to recover from disk failures, and how to monitor VERITAS Volume Manager performance.

◆ The VERITAS Volume Manager User’s Guide - VERITAS Enterprise Administrator provides administrators with information on how to perform various VERITAS Volume Manager tasks through the VERITAS Enterprise Administrator (VEA) graphical user interface.

◆ VERITAS Volume Manager Intelligent Storage Provisioning Administrator’s Guide provides information on how to use the VERITAS Intelligent Storage Provisioning (ISP) feature of VERITAS Volume Manager (VxVM).

◆ The VERITAS FlashSnap Point-In-Time Copy Solutions Administrator’s Guide provides information about how to implement solutions for online backup of databases and cluster-shareable file systems, for decision support on enterprise systems, and for Storage Rollback of databases to implement fast database recovery.

◆ The VERITAS Volume Manager Hardware Notes provides important VERITAS Volume Manager hardware compatibility and support information.

◆ The VERITAS Volume Manager Troubleshooting Guide provides information about how to recover from hardware failure, and how to understand and deal with VxVM error messages.
While not shipped with VERITAS Storage Foundation for Sybase, the following documents provide related information if you plan to use VERITAS NetBackup™ to back up your databases:

◆ The VERITAS NetBackup Release Notes provide important, up-to-date, and release-specific information for VERITAS NetBackup. Reading all of the Release Notes before installing or using any VERITAS products is recommended.


◆ The VERITAS NetBackup User’s Guide explains how to use VERITAS NetBackup to back up, archive, and restore files and directories.


◆ The VERITAS NetBackup BusinessServer Media Manager System Administrator’s Guide and VERITAS NetBackup DataCenter Media Manager System Administrator’s Guide describe how to use the extensive media management capabilities of VERITAS NetBackup.
## Conventions

### Typographic and Symbolic

The following tables explain the typographic and symbolic conventions used in this guide.

#### Typeface Conventions

<table>
<thead>
<tr>
<th>Typeface</th>
<th>Usage</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>monospace</strong></td>
<td>Computer output, files, directories, software elements such as command options, function names, and parameters</td>
<td>Read tunables from the <code>/etc/vx/tunefstab</code> file. See the <code>ls(1)</code> manual page for more information.</td>
</tr>
<tr>
<td><strong>monospace</strong> (bold)</td>
<td>User input</td>
<td><code># mount -F vxfs /h/filesys</code></td>
</tr>
<tr>
<td><strong>italic</strong></td>
<td>New terms, book titles, emphasis, variables replaced with a name or value</td>
<td>See the User's Guide for details. The variable <code>ncsize</code> determines the value of...</td>
</tr>
</tbody>
</table>

#### Symbolic Conventions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Usage</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>C shell prompt</td>
<td></td>
</tr>
<tr>
<td>$</td>
<td>Bourne/Korn shell prompt</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Superuser prompt (all shells)</td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td>Sybase isql prompt</td>
<td><code>&gt; quiesce database tag-1 release</code></td>
</tr>
<tr>
<td>\</td>
<td>Continued input on the following line; you do not type this character</td>
<td><code># mkfs -F vxfs -o largefiles \ /dev/vx/rdsk/PRODdg/db01</code></td>
</tr>
<tr>
<td>[ ]</td>
<td>In a command synopsis, brackets indicates an optional argument</td>
<td><code>ls [-a]</code></td>
</tr>
</tbody>
</table>
Getting Help

Symbolic Conventions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Usage</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>\</td>
<td>In a command synopsis, a vertical bar separates mutually exclusive arguments</td>
<td>mount [ suid</td>
</tr>
<tr>
<td>blue text</td>
<td>In PDF and HTML files, click on these active hyperlinks to move to the specified location</td>
<td>See “Using Snapshots for Database Backup” on page 97 for more information.</td>
</tr>
</tbody>
</table>

Notes and Cautions

**Note** A Note provides information that makes it easier to use the product or helps you avoid problems.

**Caution** A Caution warns you about situations that can cause data loss.

Getting Help

For assistance with any of the VERITAS products, contact VERITAS Technical Support:

- U.S. and Canadian Customers: 1-800-342-0652
- International: +1-650-527-8555
- Email: support@veritas.com

For license information:

- Phone: 1-650-527-0300
- Email: license@veritas.com
- Fax: 1-650-527-0952

For software updates:

- Email: swupdate@veritas.com
For additional technical support information, such as TechNotes, product alerts, and hardware compatibility lists, visit the VERITAS Technical Support Web site at:

http://support.veritas.com

For additional information about VERITAS and VERITAS products, visit the Web site at:

http://www.veritas.com
Introducing VERITAS Storage Foundation for Sybase

This chapter provides an overview of the features and component products of VERITAS Storage Foundation for Sybase.

Topics covered in this chapter include:

◆ “VERITAS Storage Foundation for Sybase” on page 2
◆ “VERITAS Volume Manager” on page 4
◆ “VERITAS File System” on page 13
◆ “VERITAS Storage Foundation/High Availability for Sybase (Optional)” on page 18
VERITAS Storage Foundation for Sybase

**Standard Edition/Enterprise Edition**

There are two versions of VERITAS Storage Foundation for Sybase:

- 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 for the Enterprise Edition.

---

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

**Features**

VERITAS Storage Foundation for Sybase combines the strengths of the core VERITAS technology products with database-specific enhancements to offer unrivaled performance, availability, and manageability for Sybase Adaptive Server Enterprise (ASE) 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 superior 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 (up to 8 exabytes.)
VERITAS File System offers 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 VERITAS 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 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.

You can also use the VxDBA menu or command line interface to perform these tasks.

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.

The following is an example of the VERITAS Storage Foundation for Sybase storage stack:

<table>
<thead>
<tr>
<th>Disks</th>
<th>RAID</th>
</tr>
</thead>
<tbody>
<tr>
<td>VxVM</td>
<td></td>
</tr>
<tr>
<td>VxFS</td>
<td>Quick I/O</td>
</tr>
<tr>
<td></td>
<td>Sybase</td>
</tr>
</tbody>
</table>

The rest of this chapter summarizes the key features of these products.
VERITAS Volume Manager

Databases typically 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:

- Spanning of multiple disks—eliminates media size limitations.
- Striping—increases throughput and bandwidth.
- Mirroring or RAID-5—increases data availability.
- Online relayout—allows online volume layout changes to improve database performance.
- 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. With this release, VxVM introduces full-sized and space-optimized instant snapshots, the preferred way to implement 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.
- 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.
- 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 SCSI devices.
- 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 planning.

Free space pool management—simplifies administration and provides flexible use of available hardware.

Online administration—allows configuration changes without system or database down time.

The following sections provide brief overviews of VxVM concepts and features that are relevant to database administration. The information and examples presented in the remainder of this guide assume that you are using VERITAS Volume Manager. For a more detailed description of VxVM and its features, refer to the VERITAS Volume Manager Administrator’s Guide.

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. Due to its virtual nature, a volume is not restricted to a particular disk or a specific area. For example, a volume can span multiple disks and can be used to create a large file system.

Volumes are composed 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.

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.
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.

Spanning and Concatenation

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.

The subdisks in a concatenated plex do not have to be physically contiguous and can belong to more than one disk under VERITAS Volume Manager control. Concatenation using subdisks that reside on more than one VM disk is called spanning.

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

Striping (RAID-0)

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 generally make the entire volume unusable.

**Mirroring (RAID-1)**

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 in order for the redundancy to be effective.

**Mirrored-Stripe Volumes (RAID-0+1)**

VxVM supports the combination of mirroring with striping. When used together on the same volume, mirroring plus striping offers the benefits of spreading data across multiple disks while providing redundancy of data.

Mirrored-stripe volumes have multiple plexes as mirrors, each constructed as a striped plex. Allocate subdisks used in the same striped plex from separate disks, and use a disk in only one mirror of a volume.

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

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

VxVM supports the combination of striping with mirroring. When used together on the same volume, striping plus mirroring offers the benefits of spreading data across multiple disks while providing redundancy of data.

Striped-mirror volumes combine striping and mirroring, but the mirroring is done at stripe column level. In case of failure, this type of volume recovers faster than RAID-0+1 volumes and the tolerance for disk failure is greater.

For databases that support online transaction processing (OLTP) workloads, we recommend either mirrored-stripe or striped-mirror volumes to improve database performance, reliability, and availability.
RAID-5

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.

Online Relayout

As databases grow and usage patterns change, online relayout lets you change volumes to a different layout. 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.

Volume Resynchronization

When storing data redundantly, using mirrored or RAID-5 volumes, VERITAS Volume Manager takes necessary measures to ensure 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 be computationally expensive and can have a significant impact on 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.

**Dirty Region Logging**

Dirty Region Logging (DRL) 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. In VxVM 4.0, if a version 20 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.

**Volume Snapshots**

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

- Full-sized instant snapshots
- Space-optimized instant snapshots
- Snappoints (based on instant snapshots)
- Traditional third-mirror snapshots

For detailed information on performing database backups using volume snapshots, see “Using Volume Snapshots for Dataserver Backup and Off-Host Processing” on page 99.

**VERITAS FastResync (Optional)**

VERITAS FastResync (previously called Fast Mirror Resynchronization or FMR) is included with the Enterprise 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.

**Non-Persistent FastResync**

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.

**Persistent FastResync**

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. If persistent FastResync is enabled on a volume or on a snapshot volume, a **data change object** (DCO) and a **DCO log volume** are associated with the volume.

In VxVM 3.2 and 3.5, the DCO object only managed information about the FastResync maps. These maps track writes to the original volume (and to each of up to 32 snapshot volumes) since the last snapshot operation. The DCO log volume on disk holds the 33 maps, each of which is 4 blocks in size by default.

In VxVM 4.0, 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.

**Disk Group Split and Join (Optional)**

Disk group split and join is included with the Enterprise 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.

**Hot-Relocation**

In addition to providing volume layouts that help improve database performance and availability, VxVM offers additional features that you can use to further improve system availability in the event of a disk failure. *Hot-relocation* is the ability of 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.

**DMP-Supported Disk Arrays**

VxVM provides administrative utilities and driver support for disk arrays that can take advantage of its Dynamic Multipathing (DMP) feature. The DMP driver is a pseudo driver that logically loads on top of SCSI target drivers. Its purpose is to present a single device interface to a SCSI disk or device that can be accessed through multiple hardware paths. The DMP driver manages these multiple paths and allows for transparent failover, load sharing, and hot plugging of SCSI devices. For detailed information, see the *VERITAS Volume Manager Hardware Notes*.

**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* for more information.
Veritas Volume Manager

**Hitachi Extended Copy Facility**

The Hitachi Extended Copy facility allows a host server to request data movement from disk to tape or to another disk without the overhead of server CPU time I/O during the actual movement of data.

**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* for more information.

**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* for more information.

**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. In the event that 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 for more information.
VERITAS File System

VERITAS File System (referred to as VxFS or vxfs) is an extent-based, intent logging file system intended for use in 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. For a more detailed description of VxFS and its complete feature set, see the VERITAS File System Administrator’s Guide.

VERITAS Quick I/O

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 “Using VERITAS Quick I/O” on page 49 for more information.

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 “Using VERITAS Cached Quick I/O” on page 77 for more information.
**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.

The VxFS file system addresses this performance issue by using a different allocation scheme that is extent-based. An *extent* is defined as one or more adjacent blocks of data within the file system. An extent is presented as an *address-length* pair that identifies the starting block address and the length of the extent (in file system or logical blocks). 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 is done 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.

For information on how to create preallocated database files using extent attributes, see “Preallocating Space for Quick I/O Files Using the setext Command” on page 58.

**Fast File System and Database Recovery**

After a system crash, database recovery cannot start until after the completion of file system recovery. By default, the UFS file system relies on a full structural verification by the *fsck* utility as the only means to recover from a system failure. This verification involves checking the entire structure, verifying that the file system is intact, and correcting any inconsistencies that are found. For large file system and database configurations, this process is often very time consuming because the entire file system must be scanned.

The VxFS file system provides recovery only *seconds* after a system failure by using a tracking feature called *intent logging*. Intent logging is a scheme that records pending changes to the file system structure in a transaction log. During system failure recovery, the VxFS *fsck* utility replays the intent log to complete or terminate any pending file system operations. The file system can then be mounted immediately after the log replay, without requiring a full structural check of the file system. Except for the noticeable speed at which file system recovery occurs, the intent log recovery feature is transparent to the administrator.
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.

Defragmentation Utility

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 further and further along 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.

Resizing Utility

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 matter of stopping applications, off-loading 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.

Cross-platform Data Sharing

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. See the VERITAS Volume Manager Cross-platform Data Sharing Administrator’s Guide for more information.
Multi-volume Support

The multi-volume support 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. This feature can be used only in conjunction with VERITAS Volume Manager.

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.

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 4.0 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 mksfs or mount commands. See “Creating a VxFS File System” on page 34 for more information.
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.

See the *VERITAS File System Administrator’s Guide* for details on using VxFS quotas.

Storage Checkpoint File System Restore

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* for more information.

Cluster Functionality (Optional)

File system clustering is an optional, separately licensable 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* for more information.
VERITAS Storage Foundation/High Availability for Sybase (Optional)

VERITAS Storage Foundation/High Availability (HA) (VCS) for Sybase 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.
Setting Up Dataservers

This chapter describes how to use VERITAS Volume Manager and VERITAS File System to set up optimal system configurations for Sybase dataservers.

Topics covered in this chapter include:

- “Setting Up a Disk Group” on page 21
- “Creating a Disk Group for a Dataserver” on page 22
- “Adding Disks to a Dataserver Disk Group” on page 25
- “Selecting a Volume Layout” on page 27
- “Creating a Volume” on page 29
- “File System Creation Guidelines” on page 33
- “Creating a VxFS File System” on page 34
- “Mounting a File System” on page 38
- “Unmounting a File System” on page 41
- “Understanding Fragmentation” on page 43
- “Resizing a File System” on page 47
Setting Up a New Database

VERITAS Storage Foundation for Sybase contains a combination of performance, management, and high availability features. This section provides an overview of the steps to follow if you are using VERITAS Storage Foundation for Sybase to set up a new database.

To set up a new database using VERITAS Storage Foundation for Sybase

1. Determine the number and sizes of file systems you need for the database you want to create. For detailed information, see the VERITAS File System Administrator’s Guide.

2. Create volumes to meet your file system needs. You can use disk mirroring as a safeguard against disk failures and striping for better performance. For more information see the “Selecting a Volume Layout” on page 27 and “Creating a Volume” on page 29. For information about performance tuning to improve database performance, see “Tuning for Performance” on page 135.

3. Create the VxFS file systems you need on the volumes. See “File System Creation Guidelines” on page 33 and “Creating a VxFS File System” on page 34.

4. Install and configure your database. For best OLTP performance, use Quick I/O For information on creating and using Quick I/O files, and converting existing files to Quick I/O, see “Using VERITAS Quick I/O” on page 49.

If you are not currently running on VxVM and VxFS, see the VERITAS Storage Foundation for Sybase Installation Guide and “Converting Existing Database Configurations to VxFS” on page 93 for information about how to convert your existing database storage configuration to VERITAS Storage Foundation for Sybase.
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 dataserver 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 dataserver when all the configuration objects of the dataserver 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 for use with dataservers:

◆ 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.

Note 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.
Creating a Disk Group for a Dataserver

You can use the `vxdg` command or the graphical user interface (GUI) 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.

**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**

- For information on the `vxdg` command, see the `vxdg(1M)` manual page.
- In the GUI, if multiple disks are specified in the **Disk Device(s)** field and only one disk name is specified in the **Disk Name(s)** field, VxVM appends numbers to the disk name so that each disk name is unique within its disk group.
- New disks must be placed under VxVM control and then added to a dynamic disk group before they can be used for volumes. The **Actions > Rescan** command performs these tasks to prepare new disks for VxVM use.
- 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.
- Disks are automatically assigned a default name. Once a disk is under VxVM control, you can select **Actions > Rename Disk** in the GUI to change the disk name.

**To create a new disk group using the command line**

Use the `vxdg` command as follows:

```
# /opt/VRTS/bin/vxdg init disk_group [disk_name=disk_device]
```
Creating a Disk Group for a Dataserver

Example

To create a disk group named PRODdg on a raw disk partition c1t0d0s2, where the disk name PRODdg01 references the disk within the disk group:

```
# /opt/VRTS/bin/vxdg init PRODdg PRODdg01=c1t0d0s2
```

▼ To create a new disk group using the GUI

1. Select the Disk Groups folder.

2. Select Actions > New Disk Group.
   
The New Disk Group wizard welcome appears.

3. Click Next to continue. The New Disk Group wizard appears. Enter the following information:

   | Group name: | Enter a new disk group name in the Group Name field. |
   | Create cluster group | If the cluster feature is available, you can select the Create cluster group checkbox if the new disk group is to be used with clusters. |
   | Available disks: Selected disks: | Select which disks (from Available disks) you want to include in the group. Make sure the disks you want to include are in the right pane of the window (Selected disks).
   | Disk names: | Type in the names of any disks to be added that do not appear under Selected disks. |

4. When you have provided all the necessary information in the dialog box, click Next.

5. Click Yes when the Add the following disks? confirmation message appears.
6. If the disks are not empty, indicate whether the disk should be initialized or encapsulated. If you initialize the disk, any existing data on the disk will be destroyed. If you encapsulate the disk, any existing data will be preserved in volumes.

If you choose to encapsulate the root disk, the disk needs to be rebooted before encapsulation can take effect. You have the option of rebooting the system now or later.

The Organization Principle window appears.

7. Click None, if you do not want to organize the disk group. Otherwise, click Organize Disk Group By to create an ISP disk group.

8. The next screen confirms the disks you have selected. Choose Finish to continue if you are satisfied with the disk selection. If you are not satisfied, you can click the Back button to go back to the previous screen in order to modify your selections.

After clicking Finish, the new disk group will appear under the Disk Groups node. Normally, you would add all the disks you want in the group at this point. You can always add more disks later with the Add Disk to Disk Group command.

9. If you chose to encapsulate the root disk, reboot the system.
Adding Disks to a Dataserver Disk Group

When a disk group is first created, it contains only a single disk. You may need to add more disks to the disk group. This section describes how to add disks to a disk group using the `vxdg` command or the GUI. If you have many disks to add to the disk group, it is easier to use the `vxdg` command.

**Usage Notes**

- By default, the GUI assigns default disk names by appending numbers to the disk group name so that each disk name is unique within its disk group. After the disk is under VxVM control, you can rename it.
- 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.
- 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 using the command line**

Use the `vxdg` command as follows:

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

**Example**

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

```
# /opt/VRTS/bin/vxdg -g PRODdg adddisk PRODdg02=c1t1d0s2
# /opt/VRTS/bin/vxdg -g PRODdg adddisk PRODdg03=c1t2d0s2
# /opt/VRTS/bin/vxdg -g PRODdg adddisk PRODdg04=c1t3d0s2
```
Adding Disks to a Dataserver Disk Group

▼ To add a disk to a disk group using the GUI

1. Select the uninitialized disk to be placed under VxVM control.

2. Choose Actions > Add Disk to Disk Group. The Add Disk to Disk Group wizard appears. Click Next to continue.

3. Complete the Add Disk to Dynamic Group wizard as follows:

   | Disk group name: | From the pull-down menu, select the group you want to add the disk to.  
   |                 | To add the disk to a new disk group, click the New disk group button and enter the name of the new disk group in the dialog box.  
   | Available disks:  | Move the disk to be added from Available disks to Selected disks.  
   | Selected disks:   |  

4. When you have provided all the necessary information in the dialog box, click Next.

5. When the confirmation window appears, click Yes to confirm your selection.

6. Click Finish to add the disk to the selected disk group.

7. If your system supports encapsulation, indicate whether the root disk should be initialized or encapsulated. If you initialize the disk, any existing data on the disk is destroyed. If you encapsulate the disk, any existing data is preserved in volumes.

8. If you chose to encapsulate the root disk, reboot the system.
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.

Choosing 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.

Choosing 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.
Selecting a Volume Layout

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 redo 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.
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 highly 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 or the GUI to create the volume.

If you choose to use the GUI, the *VERITAS Volume Manager Administrator’s Guide* provides a detailed comparison of the layout choices and more detailed procedures on creating each volume layout type. The GUI gives you the option of placing a file system on the new volume or mirroring during volume creation.

**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 using the command line**

Use the `vxassist` command as follows:

```
# /opt/VRTS/bin/vxassist -g disk_group make volume_name size \
layout=layout_type
```
Creating a Volume

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

```
# /opt/VRTS/bin/vxassist -g PRODdg make db01 1g layout=mirror
```

▼ To create a volume using the GUI

1. Under Disk Groups under the selected host in the left pane, click on the disk group to be used to create the volume.

2. Choose Actions > New Volume. You are now asked if you want Volume Manager to select the disks to use or if you want to select them manually.

3. Select Manually select disks to use with this volume, then click Next.

4. The Select Disks to use for Volume screen appears. You can now:
   - Select disks to be used by VxVM when creating the volume. (Move to the Included: area.)
   - Select disks not to be used by VxVM when creating the volume. (Move to Excluded: area.)
   - Specify that the volume is to be mirrored across (Mirror Across:) or striped across (Stripe Across:) controllers, trays, targets, or enclosures.

Note For ISP volumes, the layouts Mirror Across: and Stripe Across: cannot be specified.

   - Specify ordered allocation (Ordered). Ordered allocation uses the specified storage to first concatenate disks, then to form columns, and finally to form mirrors.

5. Click Next. The Select the attributes for this volume screen appears.

6. Type in the Volume Name. This is a Volume Manager-specific name that is used in some Volume Manager commands. It is different from the volume label for the file system.

7. Specify the volume size, or select Maxsize.
   As you have already selected a disk, a size is shown in the volume size box that represents the maximum concatenated (simple or spanned) volume size on the disk. If you then click the Maxsize button, a new number appears in the volume size box that represents the maximum size for a spanned volume that spans all the disks in the
Creating a Volume

group. You can also click on another volume layout and then click the Maxsize button to get the maximum size for that layout that involves unallocated space on all disks in the group.

If you choose Striped or RAID-5, Number of Columns and Stripe Unit Size need to have an entry. Defaults are provided.

8. Select the required layout:
   - Concatenated (Concatenated and Concatenated Mirrored)
   - Striped (Striped and Striped Mirrored)
   - RAID-5
   - Mirrored

9. Select the required options

<table>
<thead>
<tr>
<th>Option</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mirror Info:</td>
<td>• To mirror the volume, select Mirrored. In the Total Number of Mirrors field, enter the total number of mirrors for the volume.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> Concatenated mirrored volumes are mirrored by default.</td>
</tr>
<tr>
<td>Initialize zero:</td>
<td>• To clear the volume before enabling it for general use, select Initialize Zero.</td>
</tr>
<tr>
<td>No layered volumes:</td>
<td>• To prevent the creation of a layered volume, select No Layered Volumes. In cases where a layered volume layout is appropriate, VxVM can create a layered volume when a non-layered layout is specified. This option ensures that the volume has a non-layered layout. If a layered (Pro) layout is selected, this option is ignored.</td>
</tr>
<tr>
<td>Enable FastResync</td>
<td>• If Enable FastResync is checked, then Enable Logging is an option only if the volume is mirrored. (The reason it is only enabled when Enable FastResync is checked is to ensure that a new style DRL is created.)</td>
</tr>
<tr>
<td>Enable logging</td>
<td></td>
</tr>
</tbody>
</table>
10. After you provide all the necessary information in the dialog box, click Next. You are now asked if you want to create a file system.

11. If you want to create a file system, select Create a File System and follow the instructions. If you do not want to create a file system, select No File System, then click Next.

12. A summary of your selections appears. Click Back to make changes; otherwise, click Finish.
File System Creation Guidelines

Follow these guidelines when creating VxFS file systems:

- To take advantage of Quick I/O, online administration, fast recovery of the VxFS file system, and superior reliability features, select \texttt{vxfs} as the file system type.
- Specify the maximum block size and log size 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 \texttt{db01} and the mirrors \texttt{db01-01} and \texttt{db01-02} to relate to the configuration objects. If you are using the \texttt{vxassist} command or the GUI, this is transparent.
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.

The GUI lets you add a file system on a new volume during the volume creation process. See the `VERITAS Volume Manager User's Guide - VERITAS Enterprise Administrator` for a detailed comparison of the layout choices and more detailed procedures on creating each volume layout type.

**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.
- In the GUI, you must specify a file system mount point if the file system is to be mounted at startup.
- If you select the Add to file system table checkbox in the GUI, the file system table file will be automatically updated when the file system is mounted.
- When specifying a mount point in the GUI, you must use an absolute pathname (that is, it must begin with `/`).
To create a VxFS file system on an existing volume using the command line

Use the `mkfs` command to create a VxFS file system on an existing volume 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 pathname 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.

**Example**

To create a VxFS file system that 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
largefiles not supported
```

You can now mount the newly created file system. See “Mounting a File System” on page 38.
Creating a VxFS File System

▼ To add a new file system to an existing volume using the GUI

1. Select the volume to contain the file system.


3. Complete the New File System dialog box as follows:

<table>
<thead>
<tr>
<th>File System Type</th>
<th>Select the file system type from the pull-down menu.</th>
</tr>
</thead>
</table>
| Create Options:  | • Specify the allocation size and block size if you do not want to use the default.  
|                  | • To specify whether large files (files greater than or equal to 2 GB) will be supported, click New File System Details. Select either largefiles or nolargefiles. |
| Compress (checkbox): | • If your platform supports file compression, then you can select Compress to compress the files on your file system |
| Mount Options:   | • Enter the mount point for the file system, if you want the file system mounted at system startup.  
|                  | • Select the Create mount point checkbox if you want the system to create the mount point if it does not already exist.  
|                  | • Select the Read only and Honor setuid checkboxes, as required.  
|                  | • Select the Add to file system table and Mount at boot checkboxes to update the system table file and mount the file system at system startup.  
|                  | • To update the system table file and not mount the file system at system startup, select Add to file system table checkbox and leave the Mount at Boot checkbox unselected.  
|                  | • To specify mount options, click Mount File System Details and specify the appropriate options in the Mount Details dialog box. |

4. After you provide all necessary information in the dialog box, click OK.
Support for Large File Systems and Large Files

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.

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
```

Caution 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 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

For more information about volume sets and multi-volume support, 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 `mount` 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 `-o qio` 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 `-o noqio` option.

Prerequisites

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

Usage Notes

- 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.
- If you use the GUI, the file system table file is automatically updated.
- The mount point must be an absolute pathname (that is, it must begin with `/`).
- If you use the GUI, the path specified for the mount point will be created if it does not already exist.

▼ To mount a file system using the command line

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
Mounting a File System

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.

▼ To mount a file system on an existing volume using the GUI
1. Select the volume that contains the file system to be mounted.
2. Choose Actions > File System > Mount File System.
3. Complete the Mount File System dialog box (see step 3 on page 39 in the procedure below).
4. After you have provided all the necessary information in the dialog box, click OK.

▼ To mount any file system using the GUI
1. Select the file system to be mounted.
2. Choose Actions > Mount File System.
3. Complete the Mount File System dialog box as follows:

| FS Type: | Select the file system type. |
Mounting a File System

<table>
<thead>
<tr>
<th>Mount Options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• If you want the system to use the mount options defined in the system table, check <strong>Mount options in the file system table</strong>.</td>
</tr>
<tr>
<td>• Enter the mount point for the file system, if you want the file system mounted at system startup.</td>
</tr>
<tr>
<td>• Select the <strong>Create Mount Point</strong> checkbox if you want the system to create the mount point if it does not already exist.</td>
</tr>
<tr>
<td>• Select the <strong>Read Only</strong> and <strong>Honor setuid</strong> checkboxes, as required.</td>
</tr>
<tr>
<td>• To specify mount options, click <strong>Mount File System Details</strong> and specify the appropriate options in the <strong>Mount Details</strong> dialog box.</td>
</tr>
</tbody>
</table>

4. After you have provided all the necessary information in the dialog box, click **OK**.
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.

Prerequisites
◆ A file system must exist and be mounted in order to be unmounted.

Usage Notes
◆ See the `umount(1M)` manual page for more information on mounting file systems.
◆ You cannot unmount a file system that is in use.

▼ To unmount a file system using the command line

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.

   **Note** 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 the `umount` command:

   ```
   # umount special
   ```

   or

   ```
   # umount /mount_point
   ```

   or

   ```
   # 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
Example

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

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

▼ To unmount a file system on a volume using the GUI

1. Select the volume containing the file system to be unmounted.

2. Choose Actions > File System > Unmount File System.

3. Click Yes in the Unmount File System dialog box to confirm that you want to unmount the file system.

4. If an entry exists for a file system in the file system table, a Remove File System dialog appears. Click Yes in the Remove File System dialog if you want the file system table entry removed.
Understanding 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.

Controlling 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.
Monitoring 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/df -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:

Options

- `-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.
- `-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.
- `-D` - `-E` Produces reports on directory and extent fragmentation, respectively.
- `-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.
Understanding Fragmentation

-p Specifies a maximum number of passes to run. The default is five.
-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.

Usage Notes
◆ If you specify -d and -e, directory reorganization is always completed first.
◆ If you use both -D and -E with the -d and -e options, the fragmentation reports are produced both before and after reorganization.
◆ 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.

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

To defragment a file system using the command line
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.

```
```

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

Example
To defragment a file system:

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

<p>| Directory Fragmentation Report |
|-----------------------------|-----------------|-----------------|-----------------|-----------------|</p>
<table>
<thead>
<tr>
<th></th>
<th>Dirs</th>
<th>Total</th>
<th>Immed</th>
<th>Immeds</th>
<th>Dirs to Add</th>
<th>Blocks to Reduce</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Directory Fragmentation Report

<table>
<thead>
<tr>
<th>Dirs</th>
<th>Total</th>
<th>Immed</th>
<th>Immeds</th>
<th>Dirs to Add</th>
<th>Blocks to Reduce</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Chapter 2, Setting Up Dataservers 45
Understanding Fragmentation

▼ To defragment a file system on a volume using the GUI

1. Select the volume containing the file system to be defragmented.


3. Select Yes in the displayed dialog box.
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.

**Note** If you are using the command line, 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 for more information.

**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)` and `fsadm_vxfs(1M)` manual pages for more details.

**To resize a file system using the command line**

Use `fsadm` command as follows:

```
/usr/lib/fs/vxfs/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
Resizing a File System

Example
To extend the file system /db01 to 2 GB:

```
# 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.

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

Usage Notes
◆ If you use the GUI, the underlying volume is resized when the file system is resized.
◆ `vxresize` works with VxFS file systems 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.

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.
Using VERITAS Quick I/O

VERITAS Quick I/O is a licensable 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 performance degradation typically associated with running databases on file systems or raw devices. This chapter describes how to set up and use Quick I/O.

Topics covered in this chapter include:
◆ “Understanding Quick I/O” on page 50
◆ “Creating Database Files as Quick I/O Files Using qiomkfile” on page 54
◆ “Preallocating Space for Quick I/O Files Using the setext Command” on page 58
◆ “Accessing Regular VxFS Files as Quick I/O Files” on page 60
◆ “Converting Sybase Files to Quick I/O Files” on page 62
◆ “Displaying Quick I/O Status and File Attributes” on page 69
◆ “Extending a Quick I/O File” on page 71
◆ “Recreating Quick I/O Files After Recovering a Database” on page 73
◆ “Disabling Quick I/O” on page 75
Understanding Quick I/O

How Quick I/O Works

VERITAS Quick I/O supports direct I/O and 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 for Sybase ASE dataservers 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

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

Preallocation

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. To extend Quick I/O files, see “Extending a Quick I/O File” on page 71.

Naming Convention

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 Quick I/O Improves Database Performance

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

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

### Supporting 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 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 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 the extra copying.
Understanding Quick I/O

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.

How to Set Up Quick I/O

Quick I/O is part of the VxFS binaries 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 by default without Quick I/O, 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 54 for more information.
  ● 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 58.
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 60 for more information.

- 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 62 for more information.
Creating Database Files as Quick I/O Files Using qiomkfile

The best way to preallocate space for device files and to make them accessible using the Quick I/O interface is to use the qiomkfile command. You can use the qiomkfile command to create Quick I/O files for either temporary 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.

Options

-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 71 for more information.
-r Increases the file to a specified size to allow Sybase tablespace resizing. See “Extending a Quick I/O File” on page 71 for more information.
-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.

Caution 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.
Creating Database Files as Quick I/O Files Using qiomkfile

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.

▼ 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. For more information on the disk init command, see the Sybase Adaptive Server Enterprise Reference Manual.

3. You can now 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:
   $$ isql -Usa -Psa_password -Sdataserver_name > sp_extendsegment segment_name, db_name, device_name > go $$
   For more information, see the Sybase Adaptive Server Enterprise Reference Manual.
Creating Database Files as Quick I/O Files Using qiomkfile

Example

◆ 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 this example, qiomkfile creates a regular file named /db01/.dbfile, which has the real space allocated. Then, qiomkfile creates a symbolic link named /db01/dbfile. This symbolic link is a relative link to the Quick I/O interface for /db01/.dbfile, that is, to .dbfile::cdev:vxfs:. 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 this example, 51200 2K pages is 104857600 bytes. The qiomkfile command must use this size.

Example

◆ To add a 100MB Quick I/O file named dbfile to the list of devices used by database production, use the disk init command as follows:

```
$ 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
```

Example

◆ To create a new segment, named segment2, for device dbfile on database production:

```
$ isql -Usa_password -Sdataserver_name
  > sp_addsegment segment2, production, dbfile
  > go
```
Creating Database Files as Quick I/O Files Using qiomkfile

Example

To extend a segment, named segment1, for device dbfile on database production:

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

Prerequisites
◆ The setext command requires superuser (root) privileges.

Usage Notes
◆ You need to use the chown and chgrp commands 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 .filename
   # chgrp sybase .filename
   # chmod 660 .filename
   ```
Example

To access the mount point /db01, create a datafile, preallocate the space, and change the permissions:

```bash
# cd /db01
# touch .dbfile
# /opt/VRTS/bin/setext -r 100M -f noreserve -f chgsize .dbfile
# ln -s .dbfile::cdev:vxfs: dbfile
# chown sybase .dbfile
# chgrp sybase .dbfile
# chmod 660 .dbfile
```
Accessing Regular VxFS Files as Quick I/O Files

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

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

**Note** For prerequisite information specific to Sybase, see the “Prerequisites” section of “Converting Sybase Files to Quick I/O Files” on page 62.

**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
   ```
**Example**

To access the VxFS file `dbfile` as a Quick I/O file:

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

To show the symbolic link created:

```bash
$ 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.

**Note** Only English is supported in the current release.

- You must set the following Sybase environment variables:
  
  For Sybase ASE 12.0
  
  - `$SYBASE` must be set to the Sybase home directory
  - `$DSQUERY` must be set to the server on which you intend to run the `qio_getdbfiles` command
  - `$PATH` must include `$SYBASE/ASE-12_0/bin` and `$SYBASE/OCS-12_0/bin`
  - `$LD_LIBRARY_PATH` must include `$SYBASE/OCS-12_0/lib`

  For Sybase ASE 12.5
  
  - `$SYBASE` must be set to the Sybase home directory
  - `$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`
Options

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.

- **-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.
Converting Sybase Files to Quick I/O Files

Usage Notes

◆ 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 be 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 database instance 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
   ```
Converting Sybase Files to Quick I/O Files

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 `qio_getdbfiles`, 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 writable directory where the `mkqio.dat` list resides:

   ```
   $ 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:

```bash
$ 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/VRTS/managed/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 password for Server L001
Password: pel93

◆ To view the contents of the mkqio.dat list file:

```bash
$ 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
```
Converting Sybase Files to Quick I/O Files

◆ 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.

**Options**

- `-al` Lists all files on a file system, including Quick I/O files and their links.
- `-lL` Shows if Quick I/O was successfully installed and enabled.
- `-alL` 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
```

**Example**

To show the absolute path name created using `qiomkfile` with the `-a` option:

```
$ 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
```

**Example**

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
```

where the first character, `c`, indicates it is a raw character device file, and the size field is replaced with a pair of (major, minor) numbers. If you see a `No such file or directory` message, Quick I/O did not install properly or does not have a valid license key.
To show a Quick I/O file resolved to a raw device

Use the `ls` command with the filenames as follows:

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

**Example**

To show 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.

Prerequisites

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

Options

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

Usage Notes

◆ You can also grow VxFS file systems online (provided the underlying disk or volume can be extended) using the fsadm command.

Note 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.

For more information, see “Creating Database Files as Quick I/O Files Using qiomkfile” on page 54.

◆ See the fsadm_vxfs(1M) and qiomkfile(1M) manual pages for more information.
### 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:

   ```bash
   # /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:

   ```bash
   $ /opt/VRTS/bin/qiomkfile -e extend_amount filename
   or
   $ /opt/VRTS/bin/qiomkfile -r newsize filename
   ```

**Example**

- To grow VxFS file system `/db01` to 500MB and extend the `dbfile` Quick I/O file by 20MB:

  ```bash
  # /opt/VRTS/bin/fsadm -b 500M /db01
  $ /opt/VRTS/bin/qiomkfile -e 20M dbfile
  ```

- To grow VxFS file system `/db01` to 500MB and resize the `dbfile` Quick I/O file to 300MB:

  ```bash
  # /opt/VRTS/bin/fsadm -b 500M /db01
  $ /opt/VRTS/bin/qiomkfile -r 300M dbfile
  ```
Recreating Quick I/O Files After Recovering a Database

If you need to recover 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.

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. If you do not have a `mkqio.dat` file, see “Converting Sybase Files to Quick I/O Files” on page 62.
- The `SYBASE` and `DSQUERY` environment variables must be set.

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:

<table>
<thead>
<tr>
<th>If...</th>
<th>Then...</th>
</tr>
</thead>
<tbody>
<tr>
<td>a Quick I/O file is missing</td>
<td>the Quick I/O file is recreated.</td>
</tr>
<tr>
<td>a symbolic link from a regular VxFS file to a Quick I/O file is missing</td>
<td>the symbolic link is recreated.</td>
</tr>
</tbody>
</table>
### Recreating Quick I/O Files After Recovering a Database

<table>
<thead>
<tr>
<th>If...</th>
<th>Then...</th>
</tr>
</thead>
<tbody>
<tr>
<td>a symbolic link and its associated Quick I/O file are missing</td>
<td>both the link and the Quick I/O file are recreated.</td>
</tr>
<tr>
<td>a Quick I/O file is missing and the regular VxFS file that it is symbolically linked to is not the original VxFS file</td>
<td>the Quick I/O file is not recreated and you will see a warning message.</td>
</tr>
<tr>
<td>a Quick I/O file is smaller than the size listed in the mkqio.dat file</td>
<td>the Quick I/O file is not recreated and you will see a warning message.</td>
</tr>
</tbody>
</table>
Disabling Quick I/O

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

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 writable directory where the mkqio.dat list resides:

   `$ /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:

   `# /usr/sbin/mount -F vxfs -o remount,noqio /mount_point`
Disabling Quick I/O
Using VERITAS 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.

This chapter describes how to enable and use Cached Quick I/O for enhanced performance.

Topics covered in this chapter include:

- “Understanding Cached Quick I/O” on page 78
- “Enabling Cached Quick I/O on the File System” on page 81
- “Determining Candidates for Cached Quick I/O” on page 86
- “Enabling and Disabling Cached Quick I/O for Individual Files” on page 89
Understanding Cached Quick I/O

How Cached Quick I/O Works

Cached Quick I/O is a specialized external caching mechanism specifically suitable to 32- and 64-bit ports of the Sybase server. 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 effectively 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 simply takes advantage of available memory.

Cached Quick I/O is not, however, 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 also 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 must:

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

The rest of this chapter discusses how to set up Cached Quick I/O in more detail.
Enabling Cached Quick I/O on the 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.

Prerequisite

♦ 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:sysbase /opt/VRTSvxfs/sbin/vxtunefs
# chmod 4550 /opt/VRTSvxfs/sbin/vxtunefs
```

where users belonging to the `sysbase` 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
```
Enabling Cached Quick I/O on the File System

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. 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
```

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. This command disables caching for all the Quick I/O files on this file system.

Changing System Permissions for Database Administrators

The system administrator can grant database administrators permission to change default file system behavior in order to enable and disable Cached Quick I/O.

▼ To change the vxtunefs execute permissions

Use the chown and chmod commands:

```
# chown root:sybase /opt/VRTS/bin/vxtunefs
# chmod 4550 /opt/VRTS/bin/vxtunefs
```

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` is the name of a block device
- `dgname` is the name of the disk group
- `volname` is the name of the volume

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.

For 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
```
Enabling Cached Quick I/O on the File System

Example

```bash
# /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:

```bash
qio_cache_enable = 0
```

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

▼ To obtain information on all vxtunefs system parameters

Use the vxtunefs command without grep:

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

Example

```bash
# /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_iuosz = 262144
max_direct_iuosz = 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_keepetime = 0
fcl_winterval = 3600

For a complete description of `vxtuneefs` parameters and the tuning instructions, refer to the `vxtuneefs(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.

Prerequisites

◆ You must enable Cached Quick I/O for the file systems. See “Enabling and Disabling Cached Quick I/O for Individual Files” on page 89.

Usage Notes

◆ See the qiostat(IM) manual page for more information.

Collecting I/O Statistics

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

1. Reset the qiostat counters by entering:

```bash
$ /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:

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

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

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

where n is time in seconds

Example

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

```bash
$ /opt/VRTS/bin/qiostat -l /db01/*.dbf
```
Determining Candidates for Cached Quick I/O

Analyzing 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.

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

```
OPERATIONS FILE BLOCKS AVG TIME(ms)
CACHE STATISTICS
FILE NAME	 READ WRITE READ WRITE READ WRITE
CREAD PREAD HIT RATIO
```

Analyze the output to find out where the cache-hit ratio is above a given threshold. The cache-hit ratio is calculated as:

\[
\frac{\text{CREAD} - \text{PREAD}}{\text{CREAD}} \times 100
\]

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. For more information on enabling and disabling Cached Quick I/O at the file level, see “Enabling and Disabling Cached Quick I/O for Individual Files” on page 89.
Determining Candidates for Cached Quick I/O

**Effects of Read-Aheads on I/O Statistics**

The number of CREADs in the qiostat output is the total number of reads performed, including Cached Quick I/O, and the number of PREADs is the number of physical reads. The difference between CREADs and PREADs is the number of reads satisfied from the data in the file system cache. Thus, you expect that the number of PREADs would always be equal to or lower than the number of CREADs.

However, the PREADs 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 CREADs is lower than the number of PREADs. 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.

**Using 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 for more information.

Setting Cache Advisories for Individual 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
```
Enabling and Disabling Cached Quick I/O for Individual Files

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

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
Enabling and Disabling Cached Quick I/O for Individual Files

Example
To make the Cached Quick I/O settings for individual files in the /db01 file system persistent:

```
# List of files to cache in /db01 file system
device=/dev/vx/dsk/PRODdg/db01
sysprocs.dbf,OFF
master.dbf,OFF
user.dbf,ON
```

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
```

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
```
Enabling and Disabling Cached Quick I/O for Individual Files
You can convert existing database configurations to VERITAS File System. This chapter describes how to migrate UFS file systems, earlier version layouts, and raw devices to current VxFS file systems.

Topics covered in this chapter include:

- “Converting From UFS to VxFS With Quick I/O” on page 94
- “Upgrading From Earlier VxFS Version Layouts” on page 95
- “Converting From Raw Devices” on page 96
Converting From UFS to VxFS With Quick I/O

Converting From UFS to VxFS With Quick I/O

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

**Caution** Do not upgrade your root file system to VxFS.

To convert a UFS 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 UFS entry in the `/etc/filesystems` directory.
5. Create a VxFS file system of the same size as the original UFS file system, using the mount point where the UFS file system was originally mounted. Use the procedure described in “Creating a VxFS File System” on page 34 to create a VxFS file system.
6. Preallocate Quick I/O files using `qiomkfile`. Use the procedure described in “Creating Database Files as Quick I/O Files Using qiomkfile” on page 54.
7. Restore the backup created in step 2 to the Quick I/O files in the new VxFS file system.
8. Restart the database.
Upgrading From Earlier VxFS Version Layouts

Prerequisites

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

Usage Notes

- The `vxupgrade` command lets you upgrade the VxFS file system disk layout while the file system is mounted. See the `vxupgrade(1M)` manual page for more details.

- VxFS supports three file system disk layouts: Versions 4, 5, and 6. 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.

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, or 6 disk layout:

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

Example

This is an example of upgrading to disk layout Version 6.

```
# /opt/VRTS/bin/vxupgrade -n 6 /db01
```

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

1. Shut down the database.
2. Make each database file accessible as a Quick I/O file.
   - See “Accessing Regular VxFS Files as Quick I/O Files” on page 60 for more information.
3. Restart the database.
Converting From Raw Devices

If the database is currently using raw disks or volumes, use one of the following procedures to use VxFS with the Quick I/O feature.

If you already have your Sybase dataserver running on raw devices (UNIX raw disk partitions, 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% larger than the total size of raw devices.

Sybase does not recommend mirroring from a raw device to a regular file because it causes the server to lose asynchronous I/O ability to the logical device. However, a Quick I/O file presents itself to the server as an asynchronous I/O raw device so you do not need to be concerned about that.

**Note** For simplicity sake, the procedure provided assumes 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*:
   
   ```bash
   $ 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:
   
   ```bash
   > 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.
Converting From Raw Devices
Using Volume Snapshots for Dataserver Backup and Off-Host Processing

This chapter describes how to use VERITAS Volume Manager (VxVM) snapshot volumes to create a copy of a database for backup and off-host processing in a Sybase ASE 12.5 environment.

Topics covered in this chapter include:

- “Using Snapshot Volumes” on page 100
- “Backup and Off-host Processing Applications“ on page 101
- “FastResync of Snapshot Volumes” on page 102
- “Disk Group Split and Join“ on page 103
- “Preparing Hosts for Database Backup or Off-Host Processing” on page 105
- “Sybase Adaptive Server Enterprise 12.5 Quiesce Feature” on page 109
- “Setting Up Volume Snapshots with Sybase ASE 12.5 Server” on page 110
- “Implementing Online Backup or Off-host Processing“ on page 111
- “Creating a Warm Standby Server” on page 116
- “Resynchronizing the Snapshot to Your ASE Dataserver” on page 125
- “Recovering the Database from a Backup Image” on page 128
- “Dissociating a Snapshot Volume” on page 133
- “Removing a Snapshot Volume“ on page 134
Using 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 \texttt{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/Os 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 “Disk Group Split Operation” below.

![Disk Group Split Operation](image)

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 “Disk Group Join Operation” below.

![Disk Group Join Operation](image)

**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

The following figure, “Example of a Single-Host Configuration,” shows the suggested arrangement for using snapshot volumes on the primary host to avoid disk contention.

![Example of a Single-Host Configuration](image)

Two-Host Configuration

The figure below, “Example of a Two-Host Configuration,” 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.
Preparing Hosts for Database Backup or Off-Host Processing

Example of a Two-Host Configuration

Primary Host

Secondary Host

Network

1 2

SCSI or Fibre Channel Connectivity

Disks containing primary volumes used to hold production databases or file systems

Disks containing snapshot volumes

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.
Preparing Hosts for Database Backup or Off-Host Processing

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=n` 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
   ```
Preparing Hosts for Database Backup or Off-Host Processing

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.
Setting 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.

Example

The table below 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>
<thead>
<tr>
<th>Volume Name</th>
<th>Mount Point</th>
<th>Sybase Device Name</th>
<th>Physical File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>sybvoll</td>
<td>/sybasedata</td>
<td>Master</td>
<td>/sybasedata/nlaptest/master</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sysprocsdev</td>
<td>/sybasedata/nlaptest/sysprocs</td>
</tr>
</tbody>
</table>
Implementing Online Backup or Off-host Processing

This section describes how to implement online backup of an ASE database on a secondary host by backing up database devices. It shows you how to 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.

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.
◆ Resynchronization speed varies based on the amount of data changed in both the primary and secondary volumes during the break-off time.
Implementing Online Backup or Off-host Processing

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`.

   **Note** If the volume was created under VxVM 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 VxVM, and it has any attached snapshot plexes or it is associated with any snapshot volumes, follow the procedure given in “Upgrading Existing Volumes to Use VxVM 4.0 Features” on page 107.

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.

   **Note** 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:

```bash
$ 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 in step 4 on page 112, create a full-sized instant snapshot volume by running the following command on the primary host:

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

**Note** 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 for more information.

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

```bash
# vxsnap -g diskgroup make
source=vol1/newvol=snapvol1/nmirror=1 \nsource=vol2/newvol=snapvol2/nmirror=1 \nsource=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:

```bash
$ isql -Usa -Ppassword -Sserver
> quiesce database tag2 release
> go
> quiesce database tag1 release
> go
> quit
```
Implementing Online Backup or Off-host Processing

8. On the primary host, use the following command to move the snapshot volume to another disk group from the original disk group:
   
   ```bash
   # vxdg split diskgroup new_diskgroup snapshot_volume
   ```

   **Note** 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:
   
   ```bash
   # vxdg deport new_diskgroup
   ```

10. On the secondary host where the backup is to be performed, import the snapshot volume’s disk group:

    ```bash
    # vxdg import new_diskgroup
    ```

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

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

12. 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:

    ```bash
    # 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.

13. When the backup is finished, unmount the file system:

    ```bash
    # umount /mount_point
    ```

14. On the secondary host, deport the snapshot volume’s disk group:

    ```bash
    # vxdg deport new_diskgroup
    ```

15. On the primary host, re-import the snapshot volume’s disk group:

    ```bash
    # vxdg import new_diskgroup
    ```
16. On the primary host, join the snapshot volume’s disk group with the original volume’s disk group:

# vxdg join new_diskgroup diskgroup

17. 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 vxsnap print command for users to check the percentage of volumes that need to be synchronized. In the command output, this is referred to as % DIRTY.

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

# vxsnap -g diskgroup print snapshot_volume

19. To resynchronize with the original volume:

# vxsnap -g diskgroup reattach snapvol source=vol

Repeat step 8 on page 114 through step 19 on page 115 each time you need to back up the volume.
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:

1. 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.

2. 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.

3. Import the disk group dump_diskgroup onto the secondary host and perform load transaction commands on the failover database.

4. Repeat the dump/load transaction commands (steps 2 and 3 above) in fixed intervals. For example, repeat the process every hour.

5. Repeat the entire process in fixed intervals. For example, repeat the process every day.

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.
Creating a Warm Standby Server

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.
- 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:

   ```bash
   # 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.

   **Note** If the volume was created under VxVM 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 VxVM, and it has any attached snapshot plexes or it is associated with any snapshot volumes, follow the procedure given in “Upgrading Existing Volumes to Use VxVM 4.0 Features” on page 107.

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

   ```bash
   # 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.

**Note** 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 step 6 on page 118:

```
# vxsnap -g diskgroup make \\
source=volume/newvol=snapshot_volume/plex=plex_name
```
Note 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 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 for more information.

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

```bash
# 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:

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

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

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

Note 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:

```bash
# vxdg deport new_diskgroup
```

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

```bash
# 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 on page 120 to this instance:

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

**Note** 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:

```
$ srvbuild -r srvbuild.server_name.rs
```

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 -server_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:

```
# vxdg deport new_diskgroup
```

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

```
# vxdg deport new_diskgroup
```

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_diskgroup
   > 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 step 19 on page 120 to step 27 on page 121 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.
Creating a Warm Standby Server

Example

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
> vde
```
Creating a Warm Standby Server

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

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

Verify that DCO and FastResync are enabled on the volume.

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

Create a snapshot mirror.

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

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

```bash
# 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.

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

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

```bash
# vxdg deport new_syb
```

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

```bash
# vxdg import new_syb
```

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

```bash
# vxrecover -g new_syb -m snap_vol1
# vxvol -g new_syb start snap_vol1
```

On the secondary host, check and mount the snapshot volume.

```bash
# mkdir /sybasedata
# fsck -F vxfs /dev/vx/rdsyk/new_syb/snap_vol1 \\
# mount -F vxfs /dev/vx/dsk/new_syb/snap_vol1 /sybasedata \\
# chown -R sybase:sysbase /sybasedata
```

Log in as the instance owner.

```bash
# su - sybase
```
Creating a Warm Standby Server

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

```bash
$/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. For more information, see “Recovering the Database from a Backup Image” on page 128.

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 -PP -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
Resynchronizing the Snapshot to Your ASE Dataserver

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:

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

   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 new_diskgroup source=volumedg
   ```

8. Follow step 10 on page 119 through step 18 on page 120 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
   ```
Resynchronizing the Snapshot to Your ASE Dataserver

As root, reattach the snapshot:

```
# vxsnap -g syb reattach snap_vol1 source=sybvoll
```

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.
◆ 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 “Resynchronizing the Snapshot to Your ASE Dataserver” on page 125 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/rdsk/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.
Recovering the Database from a Backup Image

Example

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.
Refresh a Snapshot Database Image

This section describes the procedure for refreshing a snapshot database image so that it can later be used to restore the primary database.

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.
◆ 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.

Prerequisites

◆ 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
   ```

Example

To dissociate a snapshot volume from its original volume:

```
# vxsnap -g syb dis snap_vol1
```

To remove the DCO logs from the snapshot volume:

```
# vxsnap -g syb -f unprepare snap_vol1
```

To remove the DCO logs from the primary volume, if necessary:

```
# vxsnap -g syb -f unprepare sybvol
```
Removing a Snapshot Volume

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.

**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.

**Caution** If the volume a mounted file system, you must unmount it before removing the volume.

**Example**

To remove a snapshot volume from a disk group:

```
# vxsnap -g syb dis snap_vol1
# vxedit -g syb -rf rm snap_vol1
```
Tuning for Performance

This chapter provides tuning tips that you can use to improve database performance. Topics covered in this chapter include:

- “Tuning VxVM” on page 136
- “Tuning VxFS” on page 138
- “Tuning Sybase Dataservers” on page 147
- “Tuning Solaris” on page 149

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
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 information on tuning VERITAS Volume Manager, refer to the “Tuning VxVM” section of the “Performance Monitoring and Tuning” chapter in the VERITAS Volume Manager Administrator’s Guide.

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 with no arguments:

<table>
<thead>
<tr>
<th>TYP</th>
<th>NAME</th>
<th>READ</th>
<th>WRITE</th>
<th>READ</th>
<th>WRITE</th>
<th>READ</th>
<th>WRITE</th>
</tr>
</thead>
<tbody>
<tr>
<td>vol</td>
<td>log2</td>
<td>0</td>
<td>6312</td>
<td>0</td>
<td>79836</td>
<td>.0</td>
<td>0.2</td>
</tr>
<tr>
<td>vol</td>
<td>db02</td>
<td>2892318</td>
<td>3399730</td>
<td>283759</td>
<td>7852514</td>
<td>20.6</td>
<td>25.5</td>
</tr>
</tbody>
</table>

The “Performance Monitoring” section of the “Performance Monitoring and Tuning” chapter in the *VERITAS Volume Manager Administrator’s Guide* provides detailed information 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. Refer to the `vxstat(1M)` manual page for more information.
**Tuning VxFS**

VERITAS File System provides a rich 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.

**Monitoring Free Space**

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 for a description of online file system expansion.

**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` runs. If the degree of fragmentation is low, increase the interval between `fsadm` runs.

### Tuning VxFS I/O Parameters

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. 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. For more details, see the `vxtunefs(1M)` and `tunefstab(4)` manual pages. The `vxtunefs` command can be used to print the current values of the I/O parameters.
If the default alignment from `mkfs` is not acceptable, the `-o align=n` option can be used to override alignment information obtained from VxVM.

**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 the size of the stripe unit, which is usually 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 the size of the stripe unit, which is usually 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 the number of columns.

- **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 the number of columns.
### 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 indirections 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.
Tuning VxFS

**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.
write_throttle

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 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.
Tuning VxFS

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.

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:

<table>
<thead>
<tr>
<th>FILENAME</th>
<th>OPERATIONS</th>
<th>FILE BLOCKS</th>
<th>AVG TIME (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>READ WRITE</td>
<td>READ WRITE</td>
<td>READ WRITE</td>
</tr>
<tr>
<td>/db01/file1</td>
<td>0</td>
<td>0</td>
<td>0.0 0.0</td>
</tr>
<tr>
<td>/db01/file2</td>
<td>0</td>
<td>0</td>
<td>0.0 0.0</td>
</tr>
<tr>
<td>/db01/file3</td>
<td>73017 181735</td>
<td>718528 1114227</td>
<td>26.8 27.9</td>
</tr>
<tr>
<td>/db01/file4</td>
<td>13197 20252</td>
<td>105569 162009</td>
<td>25.8 397.0</td>
</tr>
<tr>
<td>/db01/file5</td>
<td>0</td>
<td>0</td>
<td>0.0 0.0</td>
</tr>
</tbody>
</table>

For detailed information on available options, see the qiostat(1M) manual page.
Using 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:

<table>
<thead>
<tr>
<th>Filename</th>
<th>Operations</th>
<th>File Blocks</th>
<th>Avg Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>READ  WRITE</td>
<td>READ  WRITE</td>
<td>READ  WRITE</td>
</tr>
<tr>
<td>/db01/cust1</td>
<td>218</td>
<td>36</td>
<td>22.8</td>
</tr>
<tr>
<td>/db01/hist1</td>
<td>0</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>/db01/nord1</td>
<td>10</td>
<td>14</td>
<td>21.0</td>
</tr>
<tr>
<td>/db01/ord1</td>
<td>19</td>
<td>16</td>
<td>17.4</td>
</tr>
<tr>
<td>/db01/ord2</td>
<td>189</td>
<td>41</td>
<td>21.1</td>
</tr>
<tr>
<td>/db01/roll1</td>
<td>0</td>
<td>50</td>
<td>0.0</td>
</tr>
<tr>
<td>/db01/stk1</td>
<td>1614</td>
<td>238</td>
<td>19.3</td>
</tr>
<tr>
<td>/db01/sys1</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>/db01/templ</td>
<td>1</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>/logs/ware1</td>
<td>3</td>
<td>14</td>
<td>23.3</td>
</tr>
<tr>
<td>/logs/log1</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>/logs/log2</td>
<td>0</td>
<td>217</td>
<td>0.0</td>
</tr>
</tbody>
</table>

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.
Interpreting I/O Statistics

When running your database through the file system, the read-write lock on each file allows only one active write per file. When you look at the disk statistics using `iostat`, the disk reports queueing time and service time. The service time is the time that I/O spends on the disk, and the queueing time is how long it waits for all of the other I/Os ahead of it. At the volume level or the file system level, there is no queueing, so `vxstat` and `qiostat` do not show queueing time.

For example, if you send 100 I/Os at the same time and each takes 10ms, the disk reports an average of 10ms service and 490ms of queueing time. The `vxstat`, `odmstat`, and `qiostat` report an average of 500ms service time.
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.

Tuning 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. For details please refer to the tempdb performance chapter in the Sybase ASE Performance and Tuning Guide.

Example

To change tempdb to a dedicated 200 MB 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
   ```
Tuning Sybase Dataservers

```sql
> 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.

**Tuning 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.
Tuning Solaris

To achieve optimal performance using VERITAS Storage Foundation for Sybase and 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.

The rest of this section describes the important tuning parameters that Sybase depends on for optimal performance. 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 Sybase concurrently, you may need to increase this 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
   ```
Tuning Solaris

**shmmxax**

This parameter sets the maximum size (in bytes) of a single shared memory segment. See your Sybase documentation for the recommended value.

**shmmni**

This parameter sets the number of shared memory identifiers. See your Sybase 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 Sybase 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 Sybase 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 Sybase 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 Sybase documentation for the recommended value.
semnmu

This parameter sets the system-wide maximum number of undo structures. Setting this parameter value equal to semni 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 Sybase documentation for the recommended value of semnmu.
VERITAS Storage Foundation for Sybase
Command Line Interface

VERITAS Storage Foundation for Sybase provides a command line interface (CLI) to many key operations. The command line interface lets you incorporate command operations into scripts and other administrative processes.

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 Database Edition for Sybase Installation Guide to ensure you can use these commands and view the online manual pages.

The following table summarizes the commands available to you from the command line:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qio_getdbfiles</td>
<td>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.</td>
</tr>
<tr>
<td>qio_convertdbfiles</td>
<td>Converts VxFS files to Quick I/O files.</td>
</tr>
<tr>
<td>qio_recreate</td>
<td>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.</td>
</tr>
</tbody>
</table>
Examples of Using the Command Line Interface

This section provides examples for using 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.

Prerequisites

- You must log in as the database administrator to use the following CLI commands.
  - qio_getdbfiles
  - qio_convertdbfiles
  - qio_recreate

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.

Prerequisites

- The SYBASE and DSQUERY environment variables must be set.
- You must be logged in as the Database Administrator.

Usage Notes

- qio_getdbfiles 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.
- 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.
- See the qio_getdbfiles(1M) manual page for more information.
Examples of Using the Command Line Interface

▼ To identify the VxFS files to convert to Quick I/O

1. While the database instance is up and running, extract information on the files used by the database:

```
$ /opt/VRTSybed/bin/qio_getdbfiles [-T syb] [-d database_name] [-m master_device_pathname]
```

where -T syb forces behavior for Sybase. Use this option in environments where the database type is ambiguous. 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 default behavior is to include all database device files that are part of the ASE server. Only Sybase database device files with the dsync flag set to true will be converted to Quick I/O.

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
```

Converting VxFS Files to Quick I/O Files 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.
Examples of Using the Command Line Interface

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 to use this command.

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 qio_convertdbfiles command expects all files to be owned by the database administrator.
- qio_getdbfiles and qio_convertdbfiles 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.
- Ensure that the database is shut down before running qio_convertdbfiles.
- See the qio_convertdbfiles(1M) manual page for more information.

To convert VxFS files to Quick I/O files

1. After running the qio_getdbfiles command, shut down the database.

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 [-T syb]

   **Note** 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`

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

```
$ /opt/VRTSsybed/bin/qio_convertdbfiles -u
.dbfile::cdev:vxfs: --> dbfile
```

**Note** 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`.

Example

In this example a regular VxFS file named `dbfile` is converted to Quick I/O and then converted back to a regular VxFS file:

Get information about the file:

```
$ /opt/VRTSsybed/bin/qio_getdbfiles
$ cat mkqio.dat
dbfile 104800000
```

Shut down the database instance and convert the file to Quick I/O:

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

```
$ ls -alL d* .d*
-rw-r--r-- 1 admin dbgrp 104857600 May 2 13:42 .dbfile
crw-r--r-- 1 admin dbgrp 45, May 3 12:18 dbfile
```

```
$ ls -al d* .d*
-rw-r--r-- 1 admin dbgrp 104857600 May 2 14:42 .dbfile
```

```
lrwxrwxrwx 1 admin dbgrp 17 May 3 12:18 dbfile -> .dbfile::cdev:vxfs:
```

The `qio_convertdbfiles` command renames the file `dbfile` to `.dbfile` and creates a symbolic link to `.dbfile` with the Quick I/O extension. By default, the symbolic link uses a relative path name.

Start up the database.

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

```
$ /opt/VRTSsybed/bin/qio_convertdbfiles -u
.dbfile::cdev:vxfs: --> dbfile
```
Recreating Quick I/O Files Using qio_recreate

You can use the qio_recreate command to automatically recreate Quick I/O files when the database is recovered.

Prerequisites
◆ You must be logged on as the database administrator.
◆ The SYBASE and DSQUERY environment variables must be set.

Usage Notes
◆ The qio_recreate command expects to find a file named mkqio.dat 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 mkqio.dat file is not in the directory, you will be prompted to create it using qio_getdbfiles. For more information, see “Identifying VxFS Files to Convert to Quick I/O Using qio_getdbfiles” on page 154.
◆ 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.
◆ 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) is missing, then qio_recreate recreates it.
  ● If both a symbolic link (file) and it is 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.
Examples of Using the Command Line Interface

- If a Quick I/O file (.file::cdev: vxfs:) is smaller than the size listed in mkqio.dat, qio_recreate issues a warning message and does not recreate the Quick I/O file.

- See the qio_recreate(1M) manual page for more information.

▼ To automatically recreate Quick I/O files when the database is recovered

Use the qio_recreate command as follows:

$ /opt/VRTSsybed/bin/qio_recreate
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 purposes.

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. For a read, it is then copied to the application buffer; for a write, it is written to the storage system.
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 of 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.

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 in some fashion.
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 dataserver 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. It represents a device. The c#t#d#s# syntax identifies the controller, target address, disk, and partition.

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 c0t0d0s2. The c#t#d#s# syntax identifies the controller, target address, partition, and disk. 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 that always exists.

**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

**DSS**

See

**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.

**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

**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 some form of 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.

**files**

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**

Approximately one billion bytes. Also GB, Gbyte, 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.

kilobyte

One thousand bytes. For technical specifications, it refers to 1,024 bytes. In general usage, it sometimes refers to an even one thousand bytes. Also KB, Kbyte and K-byte.

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.

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 fifteen 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
One million bytes, or more precisely 1,048,576 bytes. Also MB, Mbyte and M-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 pathname 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

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 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.

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 kernel-supported 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 that always exists on the system. The root disk group is named rootdg.

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.

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.

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 VERITAS 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.”

terabyte

Shorthand for 1,000,000,000,000 (10^{12}) bytes (or approximately 1000 GB).

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 File Replicator (VFR)
An enterprise data replication solution used to distribute Web or file server data. It enables multi-host processing and protects against critical data loss.

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).”

VFR
See “VERITAS File Replicator (VFR).”

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

Symbols
$DSQUERY 62
$LD_LIBRARY_PATH 62
$PATH 62
$SYBASE 62

A
absolute path names
using with Quick I/O 63

absolute pathnames
use with symbolic links 60

accessing
Quick I/O files with symbolic links 60

adding disks to a disk group 25

address-length 14

agent
Cluster Server 18

allocating file space 54

allocation policies
block-based 14
UFS 14

allocation, extent-based 14

analyzing I/O statistics 87

asynchronous I/O 51

availability, using mirroring for 7

B

backups
implementing online 111

balancing I/O load 145

benefits of Quick I/O 50

buffered I/O 161

C
cache advisory
checking setting for 91

cache hit ratio
calculating 87

Cached Quick I/O

caching statistics 144

customizing 89

determining files to use 86

disabling individual files 89

enabling individual files 90

making settings persistent 90

prerequisite for enabling 81

Cached Quick I/O, overview 13

calculating cache hit ratio 87

changing file sizes 54

changing permissions 82

changing volume layouts 8

chgrp command 58

chmod command 81, 82

chown command 58, 81, 82

Cluster Server 18

collecting I/O statistics 86

command line interface 153

commands
chgrp 58

chmod 81, 82

chown 58, 81, 82

fsadm command 72

fsck 14

grep 83

ls 69

mkfs 16, 35, 37

mount 16, 37, 52

qio_convertdbfiles 62, 66

qio_getdbfiles 62, 65

qioadmin 89–??

qiomkfile 54, 71, 72

qistat 86, 144, 145

setext 58

umount 41

vxassist 29

vxdiq 22, 25

vxtune 82–84, 91

vxupgrade 95

175
concatenation 6
correcting 6
conventions xiii
coversion
   Quick I/O files back to regular files 63
   converting back to regular files 63
   regular files to Quick I/O files 66
CREADS 88
creating
   a volume 29
   Quick I/O files 55
   symbolic links to access Quick I/O files 54
creating file systems 35
cron 138
customer support xiv
customizing Cached Quick I/O 89

data
   redundancy 7
data change object 10
   DCO 10
   data warehousing 8
database
   backup using volume snapshots 9
   recovery, fast 14
   specifying type for Quick I/O 63
   tuning 149
   database performance
   using Quick I/O 51
datasync for cachep 78
DCO
   data change object 10
   log volume 10
decision support
   implementing 128
default_indir_size tunable parameter 141
defragmentation 15, 163
   extent 138
   scheduling 138
determining
   if Quick I/O installed and enabled 69
device interface 11
direct I/O 52, 162, 163
direct-write, copy-behind 79
Dirty Region Logging 9
dirty region logging 28
dirty volumes 8
disabling Cached Quick I/O for a file 89
disabling qio_cache_enable flag 82
disabling Quick I/O 75
discovered_direct_ksize tunable parameter 141
disk arrays 6
disk group
   naming a disk group 21
disk group, defined 5
disk groups
   adding disks 25
   adding disks using the command line 25
   adding disks using the GUI 26
   configuration guidelines 21
   joining 103
   limitations of move, split, and join 104
   recovery from failed reconfiguration 103
   setting up 20, 21
disk space allocation 14
disks
   encapsulating 26
   failure and hot-relocation 11
   initializing 26
dissociating snapshot volumes 133
DMP 11
double buffering 52, 78
DRL 28
DRL. See Dirty Region Logging
DSS workloads
   guidelines 28
dsync flag 62
Dynamic Multipathing 11
enabling
   Quick I/O 52
   enabling Cached Quick I/O for a file 90
   enabling qio_cache_enable flag 81
   encapsulating disks 26
   excessive reads or writes 145
   exclusive OR 8
   expansion
      file system 138
      extending a file 54
      extending Quick I/O files 71
      extent 14, 161, 164, 165
      extent-based allocation 14
      extracting file list for Quick I/O
conversion 65

fast file system recovery 14
fast recovery 28
FastResync
  Non-Persistent 10
  operation with off-host processing 102
  use with snapshots 9
file
  space allocation 54
file fragmentation, reporting on 63
file system
  overview 13
file system block 165
file system locking 52
file systems 47
  configuration guidelines 33
  creating
    using the command line 35
  fast recovery 14
  growing to accommodate Quick I/O
    files 71
  mounting 38
    using the command line 38
  recovery of 14
  resizing 15
  unmounting 41
    using the command line 41
fileset 165
fixed extent size 165
fragmentation 15, 43
  defragmenting a file system 44
  monitoring 138
  reorganization facilities 138
  reporting 138
  types 43
fragmented file system
  characteristics 139
free space 11, 138
monitoring 138
fsadm
  reporting extent fragmentation 139
  scheduling 139
fsadm command 72
fsadm utility 15, 47
fsck command 14

grep command 83
growing
  file systems 71
  Quick I/O files 71
guidelines
  creating file systems 33
  disk groups 21
  for DSS workloads 28
  for OLTP workloads 28
  striped volumes 28
  volumes 28
High Availability (HA), overview of 3
hot-relocation 11
I/O
  asynchronous 51
  Cached Quick I/O 13
  direct 52
  load balancing 145
  performance data 145
  Quick I/O 13
  sequential 14
  statistics
    obtaining 136
    unbuffered 172
I/O, kernel asynchronous 51
improving
  database performance 51
increasing the size of a file system 47
initial_extent_size tunable parameter 142
initializing disks 26
intent logging 14, 166
kernel asynchronous I/O 51
kernel settings, modifying 149
kernel write locks 52
large files, support for 16
latency 167
list file for Quick I/O conversion 66
log replay 14
ls command 69
master device path 63
max_direct_iosize tunable parameter 142
max_diskq tunable parameter 142
max_seqio_extentsize tunable parameter 142
maxuprc 149
memory persistence of FastResync in 10
mirrored volumes
snapshots 9
mirroring 4
choosing 27
mirroring and striping data 7
mirroring, defined 7
mkfs command 16, 35, 37
mkqio.dat file 66, 75
mkqio.sh script options
report on file fragmentation 63
monitoring fragmentation 138
mount command 16, 37, 52
mounting
file systems 16
mounting file systems 38
moving hot files or busy file systems 145

N
naming convention, for Quick I/O files 50
nolargefiles option 16, 37
Non-Persistent FastResync 10

O
OLTP See online transaction processing
OLTP workloads
guidelines 28
OLTP. See online transaction processing
online backups
implementing 111
online relayout 8
online transaction processing 7, 50
options
largefiles and nolargefiles 16
overview
of Quick I/O 13
of the file system 13

P
parallel data transfer, using striping for 7
parameters
default 139
tunable 140
tuning 139
parity 8
performance
obtaining statistics for volumes 136
RAID-5 8
performance data
using 145
performance tuning, for databases 149
performance tuning, list of guides 135
permissions
changing 82
persistence
for Cached Quick I/O settings 90
PREADs 88
preallocating space for Quick I/O files 50, 58
preallocation 169

Q
qio_cache_enable flag
disabling 82
enabling 81
qio_convertdbfiles command 62, 66
qio_getdbfiles command 62, 65
qioadmin command 89–??
qiomkfile command 54, 71, 72
options for creating files
symbolic links 54
qiostat command 86, 144, 145
qiostat, output of 87
Quick I/O
accessing regular VxFS files as 60
benefits 50
converting files to 66
determining file fragmentation before converting 63
determining status 69
disabling 75
enabling 52
environment variable requirements 62
extending files 71
extracting file list for conversion 66
improving database performance with 51
list file for conversion 66
naming convention for files 50
overview 13
performance improvements 79
preallocating space for files 50, 58
requirements 50
setting environment variables for 65
showing resolution to a raw device 70
specifying database name for Quick I/O 63
using relative and absolute pathnames 60

R
RAID 6
RAID-0 6
RAID-0+1 7
RAID-1 7
RAID-1+0 7
RAID-5 8, 28
    choosing 27
    performance 28
RAID-5 log 28
read_nstream tunable parameter 140
read_preference tunable parameter 140
read-ahead algorithm, for Cached Quick I/O 79
recreating data using RAID-5 8
redo logs
    configuration guidelines 33
    creating a file system 33
relative pathnames
    use with symbolic links 60
relayout 8
reliability, using mirroring for 7
removing non-VxFS files from mkqio.dat file 66
removing snapshot volumes 134
replaying the intent log 14
report
    extent fragmentation 138
requirements of Quick I/O 50
resizing 47
resizing a file 54
resizing utility 15
resynchronization
    using DRL logs 28
    using RAID-5 logs 28
resynchronization, of volumes 8
RUN_SERVER file 63

S
sa_password 64
SCSI devices 11
selecting volume layouts 27
semmap 150
semmini 150
semms 150
semmsnu 151
semmsl 151
sequential I/O
    using extent-based allocation 14
    sequential read/writes, using spanning for 6
setext command 58
setting environment variables for Quick I/O 65
setting up disk groups 20, 21
settings
    making Cached Quick I/O persistent 82
shmemx 150
shmmni 150
shmseg 150
showing
    Quick I/O file resolved to raw device 70
    snapped file systems 171
snapshot file systems 171
snapshot volumes
    database backup 9
    dissociating 133
    using the command line 133
    removing 134
    snapshots
    and FastResync 9
    spanning 4
    spanning, defined 6
    spare disks 11
    specifying
    master device path 63
    specifying database name for Quick I/O 63
statistics
    volume I/O 136
stripe unit sizes
    choosing 27
stripe units 6
striped volumes 28
    configuration guidelines 28
striping 4
striping and mirroring data 7
striping, defined 6
support for large files 16
Sybase
    environment variables for Quick I/O 62
symbolic links
    advantages and disadvantages 60
    to access Quick I/O files 60
system buffer cache 79
system recovery 14
technical support xiv
throughput 172
tunable I/O parameters 140
default_indir_size 141
discovered_direct_iosize 141
initial_extent_size 142
max_direct_iosize 142
max_diskq 142
max_seqio_extent_size 142
read_nstream 140
read_pref_io 140
write_nstream 140
write_pref_io 140
write_throttle 143
tunefstab file
adding tuning parameters to 83
Tuning
file I/O statistics 144
VxFS 138
VxFS I/O parameters 139
tuning
vxfs 138
VxVM 136
tuning I/O parameters 139
tuning parameters
adding to tunefstab file 83
tuning, for database performance 149
typographic conventions xiii
U
umount command 41
unmounting
a file system 41
unmounting a file system 41
upgrade
from raw devices 96
from UFS 94
upgrading
from earlier VxFS layout versions 95
using performance data 145
using volume snapshots for database backup 9
utilities
fsadm 15, 47
online administration 15
resizing file systems 15
utilities See commands
V
verifying caching using vxfstune
parameters 83
verifying vxtunefs system parameters 84
VERITAS Cluster Server 18
VERITAS Cluster Server Enterprise
Agent 18
VERITAS File System, overview 13
VERITAS Volume Manager
overview 4
volume layouts
selecting 27
Volume Manager 4
and RAID 6
objects 5
volume snapshots, defined 9
volumes
configuration guidelines 28
creating 29
using the command line 29
using the GUI 30
definition 5
marked as dirty 8
obtaining performance statistics 136
resynchronization 8
tvassist
used to remove DCOs from volumes 107
tvassist command 29
txdg command 22, 25
VxFS
online administration utilities 15
overview 13
tuning 138
tvstat
used to obtain volume performance statistics 136
tvxtunefs command 82–84, 91
tv_upgrade command 95
VxVM
overview 4
tuning 136
W
workloads
write-intensive 28
write_nstream tunable parameter 140
write_pref_io tunable parameter 140
write_throttle tunable parameter 143
X

XOR. See exclusive OR