VERITAS

VERITAS Volume Manager™

Getting Started Guide

Release 3.0.1

Solaris

May, 1999

P/N 100-001123

Contents

Preface

The *VERITAS Volume Manager*™ *Getting Started Guide* is an overview of the Volume Manager components, features, and interfaces. It also has information about how to setup Volume Manager to help you plan your storage management system.

Audience

This guide is intended for system administrators responsible for configuring and maintaining systems using the VERITAS Volume Manager.

This guide assumes:

- a basic understanding of system administration
- a working knowledge of the operating system being used
- experience with the window system being used

Organization

This guide provides an overview to help you understand how the Volume Manager works.

This guide is organized as follows:

Chapter 1, “Understanding Volume Manager**,”** is an overview of the VERITAS Volume Manager and related concepts.

Chapter 2, “Volume Manager Operations,” is an overview of Volume Manager features.

Chapter 3, “Volume Manager Initialization and Setup,” describes how to initialize Volume Manager and includes guidelines for setup.

Related Documents

The following documents provide related information:

- The *VERITAS Volume Manager Command Line Interface Administrator’s Guide* provides instructions for performing administration tasks by using Volume Manager commands.
- The *VERITAS Volume Manager Storage Administrator Administrator’s Guide* provides administrators with information on how to perform various Volume Manager tasks through the graphical user interface.
- The *VERITAS Volume Manager Administrator’s Reference Guide* provides administrators with information on Volume Manager recovery procedures.

Conventions

The following table describes the typographic conventions used in this guide.

Typeface	Usage	Examples

`courier`	Computer output; user input; names of commands, files, and directories	`$You have mail.`
		The `cat` command displays files.
		`$ls -a`

italics	New terms; document titles; words to be emphasized; variables to be substituted with a real name or value	`$cat` *filename* Refer to the *User’s Guide* for details.
bold	Glossary terms	

Understanding Volume Manager

1

Introduction

This “Getting Started Guide” is an overview of the VERITAS Volume Manager (VxVM) that describes what Volume Manager is, how it works, how you can communicate with it through the user interfaces, and Volume Manager concepts. Related documents that provide specific information about Volume Manager are listed in the *Preface*.

VERITAS Volume Manager provides easy-to-use online disk storage management for computing environments. Traditional disk storage management often requires that machines be taken offline at a major inconvenience to users. In the distributed client/server environment, databases and other resources must maintain high availability, be easy to access, and be safe from damage caused by hardware malfunction.

VERITAS Volume Manager provides the tools to improve performance and ensure data availability and integrity. Volume Manager also *dynamically configures disk storage while the system is active*.

This chapter introduces the VERITAS Volume Manager concepts and describes the tools that Volume Manager uses to perform storage management.

The following topics are covered in this chapter:

- How Data is Stored
- Volume Manager Overview
- Physical Objects
    - Physical Disks and Disk Naming

    - Partitions
- Volumes and Virtual Objects
    - Volume Manager Disks
    - Disk Groups
    - Subdisks
    - Plexes
    - Volumes
    - Connection Between Volume Manager Virtual Objects
- Virtual Object Data Organization (Volume Layouts)
    - Concatenation
    - Striping (RAID-0)
    - RAID-5
    - Mirroring (RAID-1)
    - Mirroring Plus Striping (RAID-1 + RAID-0)
    - Striping Plus Mirroring (RAID-0 + RAID-1)
- Volume Manager and RAID-5
    - Logging
- Layered Volumes
- Volume Manager User Interfaces
    - User Interface Overview
- Volume Manager Conceptual Overview
    - Why You Should Use Volume Manager
    - Volume Manager Objects
    - Volume Manager and the Operating System
    - Volume Manager Layouts

How Data is Stored

There are several methods used to store data on physical disks. These methods organize data on the disk so the data can be stored and retrieved efficiently. The basic method of disk organization is called *formatting*. Formatting prepares the hard disk so that files can be written to and retrieved from the disk by using a prearranged storage pattern.

Hard disks are formatted, and information stored, in two ways: physical-storage layout and logical-storage layout. Volume Manager uses the *logical-storage layout* method. The types of storage layout supported by Volume Manager are introduced in this chapter and described in detail in the *Volume Manager Administrator's Reference Guide*.

Volume Manager Overview

The Volume Manager uses *objects* to do storage management. The two types of objects used by Volume Manager are *physical objects* and *virtual objects*.

- physical objects

  Volume Manager uses two *physical objects*: physical disks and partitions. Partitions are created on the physical disks (on systems that use partitions).

- virtual objects

  Volume Manager creates *virtual objects*, called *volumes*. Each volume records and retrieves data from one or more physical disks. Volumes are accessed by a file system, a database, or other applications in the same way that physical disks are accessed. Volumes are also composed of other virtual objects that are used to change the volume configuration. Volumes and their virtual components are called *virtual objects*.

Physical Objects

This section describes the physical objects (physical disks and partitions) used by Volume Manager.

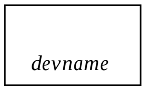
Physical Disks and Disk Naming

A *physical disk* is the basic storage device (media) where the data is ultimately stored. You can access the data on a physical disk by using a *device name* to locate the disk. The physical disk device name varies with the computer system you use. Not all parameters are used on all systems. Typical device names can include: `c#t#d#`, where:

- `c#` is the controller
- `t#` is the target ID
- `d#` is the disk number

Figure 1 shows how a physical disk and device name (*devname*) are illustrated in this document. For example, device name `c0t0d0` is connected to controller number `0` in the system, with a target ID of `0`, and physical disk number `0`.

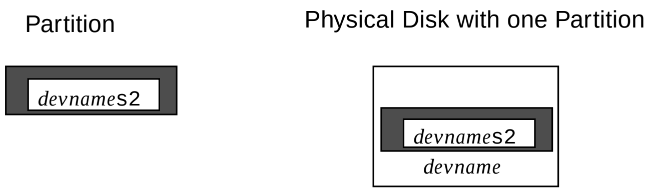
Figure 1 Example of a Physical Disk



Partitions

On some computer systems, a physical disk can be divided into one or more *partitions*. The *partition number*, or `s#`, is added at the end of the device name (*devname*). Note that a partition can be an entire physical disk, such as the partition shown in Figure 2.

Figure 2 Example of a Partition



Volumes and Virtual Objects

The connection between physical objects and Volume Manager objects is made when you place a physical disk under Volume Manager control.

Volume Manager creates *virtual objects* and makes logical connections between the objects. The virtual objects are then used by Volume Manager to do storage management tasks.

A *volume* is a virtual disk device that appears to applications, databases, and file systems as a physical disk. However, a volume does not have the limitations of a physical disk. When you use Volume Manager, applications access volumes created on Volume Manager disks (VM Disks) rather than physical disks.

Volumes contain other virtual objects that you can use to manipulate data within a volume. The virtual objects contained in volumes are: VM disks, disk groups, subdisks, and plexes. Details of the virtual objects are described in the following sections. The combination of virtual objects and how they are manipulated by volumes is described in the “Volumes” section.

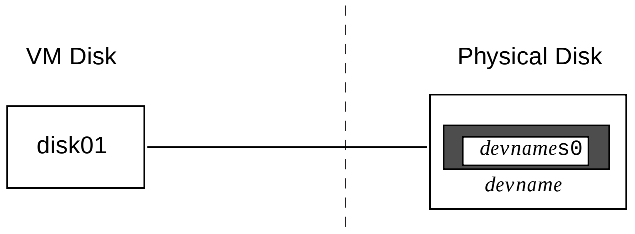
Volume Manager Disks

When you place a physical disk under Volume Manager control, a Volume Manager disk (or VM Disk) is assigned to the physical disk. A VM Disk is under Volume Manager control and is usually in a disk group. Each VM disk corresponds to at least one physical disk. Volume Manager allocates storage from a contiguous area of Volume Manager disk space.

A VM disk typically includes a *public region* (allocated storage) and a *private region* where Volume Manager internal configuration information is stored.

Each VM Disk has a unique *disk media name* (a virtual disk name). You can supply the disk name or allow Volume Manager to assign a default name that typically takes the form `disk##`. Figure 3 shows a VM disk with a media name of `disk01` that is assigned to the disk *devname*`s0`.

Figure 3 Example of a VM Disk



Disk Groups

A *disk group* is a collection of VM disks that share a common configuration. A disk group configuration is a set of records with detailed information about related Volume Manager objects, their attributes, and their connections. The default disk group is `rootdg` (the root disk group).

You can create additional disk groups as necessary. Disk groups allow the administrator to group disks into logical collections. A disk group and its components can be moved as a unit from one host machine to another.

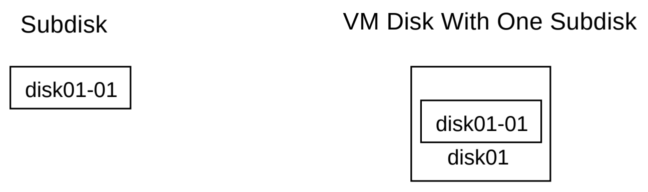
Volumes are created within a disk group. A given volume must be configured from disks in the same disk group.

Subdisks

A *subdisk* is a set of contiguous disk blocks. A block is a unit of space on the disk. Volume Manager allocates disk space using subdisks. A VM disk can be divided into one or more subdisks. Each subdisk represents a specific portion of a VM disk, which is mapped to a specific region of a physical disk.

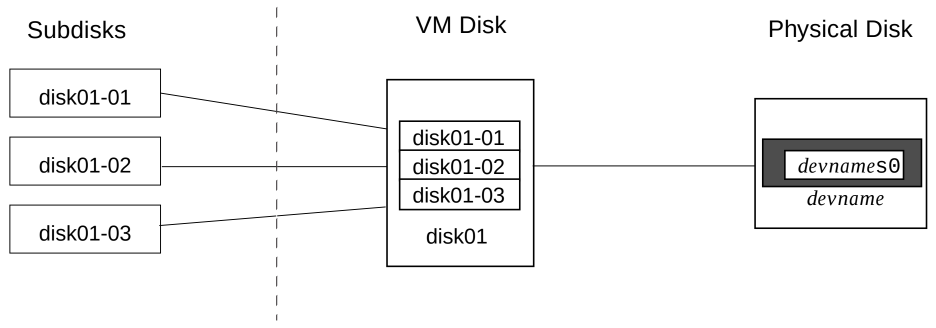
The default name for a VM disk is `disk##` (such as `disk01`) and the default name for a subdisk is `disk##-##`. As shown in Figure 4, `disk01-01` is the name of the first subdisk on the VM disk named `disk01`.

Figure 4 Example of a Subdisk



A VM disk can contain multiple subdisks, but subdisks cannot overlap or share the same portions of a VM disk. Figure 5 shows a VM disk with three subdisks. The VM disk is assigned to one physical disk.

Figure 5 Example of Three Subdisks Assigned to One VM Disk



Any VM disk space that is not part of a subdisk is free space. You can use free space to create new subdisks.

Volume Manager release 3.0 or higher allows subdisks to contain volumes. For previous versions of Volume Manager, subdisks cannot contain volumes. For more information, see “Layered Volumes.”

Plexes

The Volume Manager uses subdisks to build virtual objects called *plexes*. A plex consists of one or more subdisks located on one or more physical disks. You can organize data on the subdisks to form a plex by using these methods:

- concatenation
- striping (RAID-0)

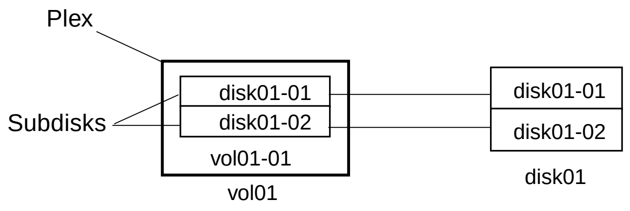
- striping with parity (RAID-5)
- mirroring (RAID-1)

Note: A Redundant Array of Independent Disks (RAID) is a disk array where part of the combined storage capacity is used to store duplicate information about the data in the array, allowing you to regenerate the data in case of a disk failure.

Figure 6 shows a plex with two subdisks.

Concatenation, striping (RAID-0), RAID-5, and mirroring (RAID-1) are described in "Virtual Object Data Organization (Volume Layouts)".

Figure 6 Example Plex With Two Subdisks



Volumes

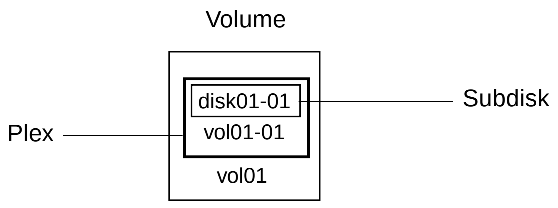
A volume consists of one or more plexes, each holding a copy of the selected data in the volume. Due to its virtual nature, a volume is not restricted to a particular disk or a specific area of a disk. The configuration of a volume can be changed by using the Volume Manager user interfaces. Configuration changes can be done without causing disruption to applications or file systems that are using the volume. For example, a volume can be mirrored on separate disks or moved to use different disk storage.

A volume can consist of up to 32 plexes, each of which contains one or more subdisks. A volume must have at least one associated plex that has a complete set of the data in the volume with at least one associated subdisk. Note that all subdisks within a volume must belong to the same disk group.

A volume with two or more data plexes is “mirrored” and contains mirror images of the data. Each plex contains an identical copy of the volume data. Refer to “Mirroring (RAID-1)” for more information on mirrored volumes.

The Volume Manager uses the default naming conventions of `vol`*##* for volumes and `vol`*##-##* for plexes in a volume. You should select meaningful names for your volumes. Figure 7 shows a volume with a single plex.

Figure 7 Example of a Volume with One Plex

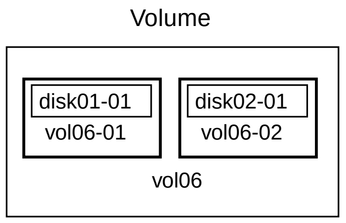


Volume `vol01` in Figure 7 has the following characteristics:

- it contains one plex named `vol01-01`
- the plex contains one subdisk named `disk01-01`
- the subdisk `disk01-01` is allocated from VM disk `disk01`

Figure 8 shows a mirror volume with two plexes.

Figure 8 Example of a Volume with Two Plexes



Volume `vol06` in Figure 8 has the following characteristics:

- it contains two plexes named `vol06-01` and `vol06-02`
- each plex contains one subdisk
- each subdisk is allocated from a different VM disk (`disk01` and `disk02`)

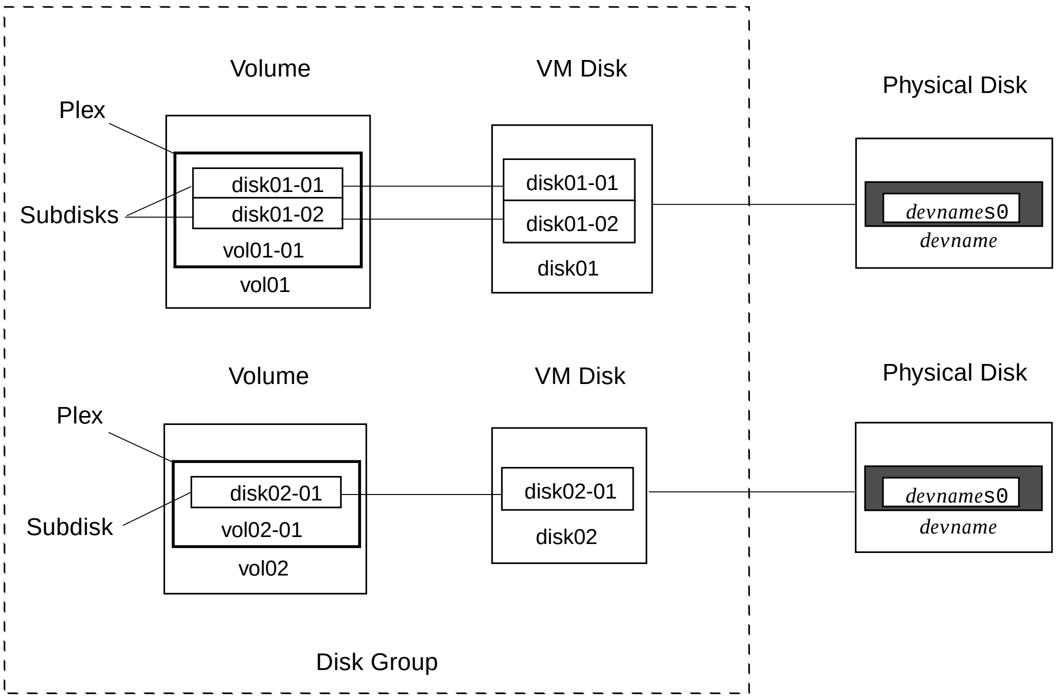
Connection Between Volume Manager Virtual Objects

Volume Manager virtual objects are combined to build volumes. The virtual objects contained in volumes are: VM disks, disk groups, subdisks, and plexes. Volume Manager objects have the following connections:

- Volume Manager disks are grouped into disk groups
- one or more subdisks (each representing a specific region of a disk) are combined to form plexes
- a volume is composed of one or more plexes

The example in Figure 9 shows the connections between Volume Manager virtual objects and how they relate to physical disks. Figure 9 shows a disk group with two VM disks (`disk01` and `disk02`). `disk01` has a volume with one plex and two subdisks. `disk02` has a volume with one plex and a single subdisk.

Figure 9 Connection Between Volume Manager Objects



Virtual Object Data Organization (Volume Layouts)

You can organize data in virtual objects to create volumes by using these layout methods:

- concatenation
- striping (RAID-0)
- RAID-5 (striping with parity)
- mirroring (RAID-1)
- mirroring plus striping
- striping plus mirroring

The following sections describe each layout method.

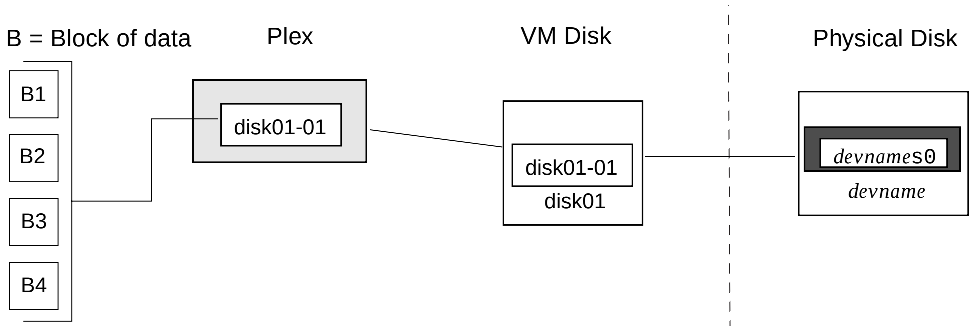
Concatenation

Concatenation maps data in a linear manner onto one or more subdisks in a plex. To access all 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 VM disk. Concatenation using subdisks that reside on more than one VM disk is called *spanning*.

Figure 10 shows concatenation with one subdisk.

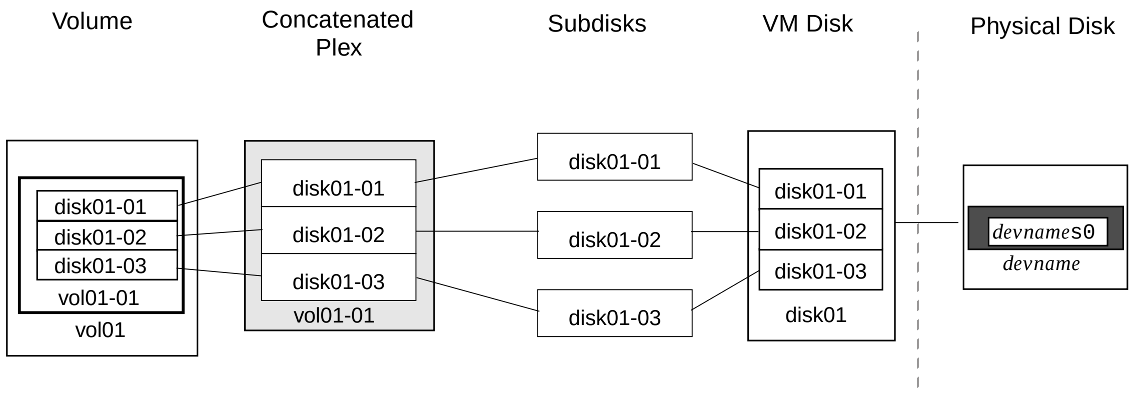
Figure 10 Example of Concatenation



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

Figure 11 shows a volume in a concatenated configuration.

Figure 11 Example of a Volume in a Concatenated Configuration

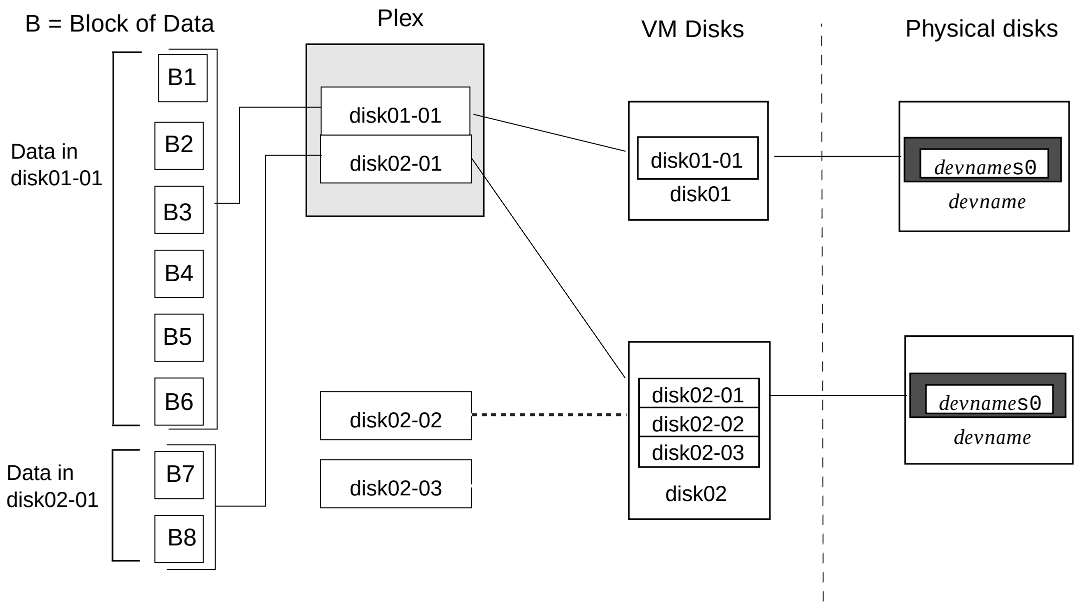


In the example shown in Figure 12, the first six blocks of data (`B1` through `B6`) use most of the space on the disk that VM disk `disk01` is assigned to. This requires space only on subdisk `disk01-01` on VM disk `disk01`. However, the last two blocks of data, `B7` and `B8`, use only a portion of the space on the disk that VM disk `disk02` is assigned to.

The remaining free space on VM disk `disk02` can be put to other uses. In this example, subdisks `disk02-02` and `disk02-03` are available for other disk management tasks.

Figure 12 shows data spread over two subdisks in a spanned plex.

Figure 12 Example of Spanning



CAUTION! Spanning a plex across multiple disks increases the chance that a disk failure results in failure of the assigned volume. Use mirroring or RAID-5 (both described later) to reduce the risk that a single disk failure results in a volume failure.

Striping (RAID-0)

Striping (RAID-0) maps data so that the data is interleaved among two or more physical disks. A striped plex contains two or more subdisks, spread out over two or more physical disks. Data is allocated alternately and evenly to the subdisks of a striped plex.

The subdisks are grouped into “columns,” with each physical disk limited to one column. Each column contains one or more subdisks and can be derived from one or more physical disks. The number and sizes of subdisks per column can vary. Additional subdisks can be added to columns, as necessary.

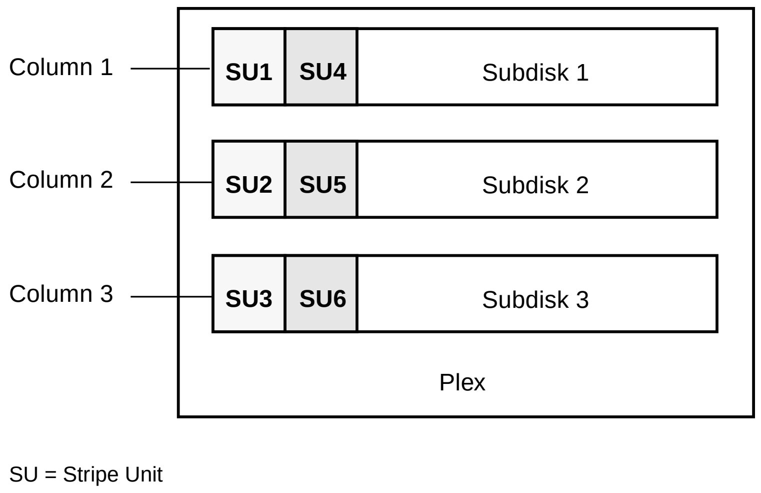
CAUTION! Striping a volume, or splitting a volume across multiple disks, increases the chance that a disk failure will result in failure of that volume. For example, if five volumes are striped across the same five disks, then failure of any one of the five disks will require that all five volumes be restored from a backup. If each volume is on a separate disk, only one volume has to be restored. Use mirroring or RAID-5 to substantially reduce the chance that a single disk failure results in failure of a large number of volumes.

Data is allocated in equal-sized units (*stripe units of size* called *stripe unit size*) that are interleaved between the columns. Each stripe unit is a set of contiguous blocks on a disk. The default stripe unit size is 64 kilobytes.

For example, if there are three columns in a striped plex and six stripe units, data is striped over three physical disks, as shown in Figure 13:

- the first and fourth stripe units are allocated in column 1
- the second and fifth stripe units are allocated in column 2
- the third and sixth stripe units are allocated in column 3

Figure 13 Striping Across Three Disks (Columns)



A *stripe* consists of the set of stripe units at the same positions across all columns. In Figure 13, stripe units 1, 2, and 3 constitute a single stripe.

Viewed in sequence, the first stripe consists of:

- stripe unit 1 in column 1
- stripe unit 2 in column 2
- stripe unit 3 in column 3

The second stripe consists of:

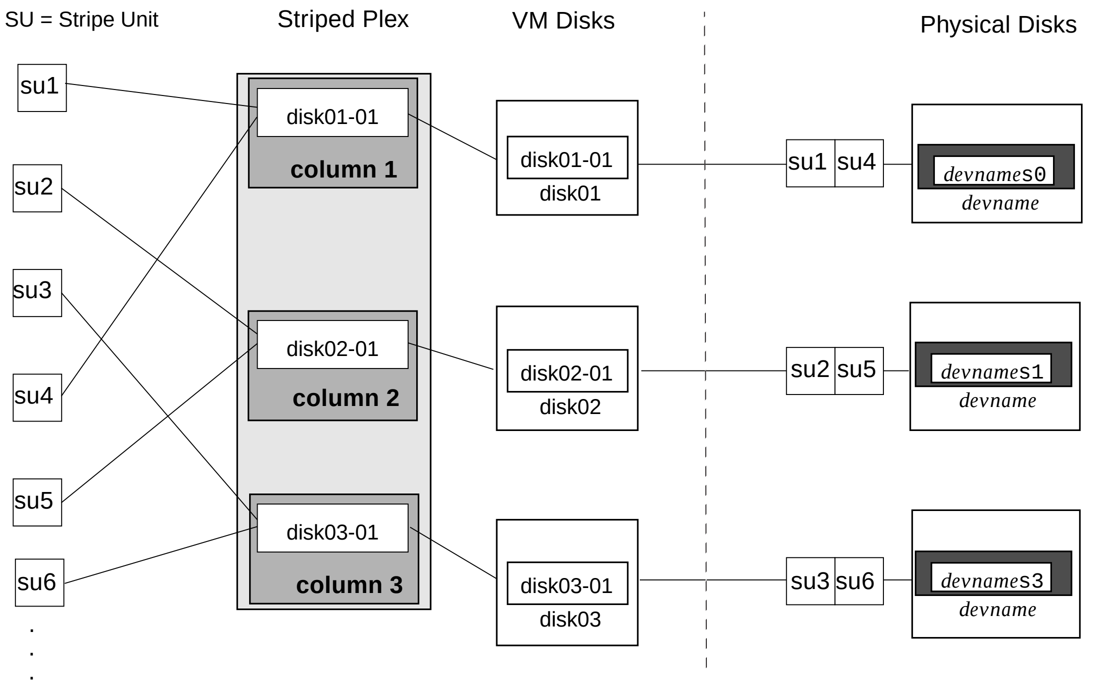
- stripe unit 4 in column 1
- stripe unit 5 in column 2
- stripe unit 6 in column 3

Striping continues for the length of the columns (if all columns are the same length) or until the end of the shortest column is reached. Any space remaining at the end of subdisks in longer columns becomes unused space.

Striping is useful if you need large amounts of data written to or read from the physical disks quickly by using parallel data transfer to multiple disks. Striping is also helpful in balancing the I/O load from multi-user applications across multiple disks.

Figure 14 shows a striped plex with three equal sized, single-subdisk columns. There is one column per physical disk.

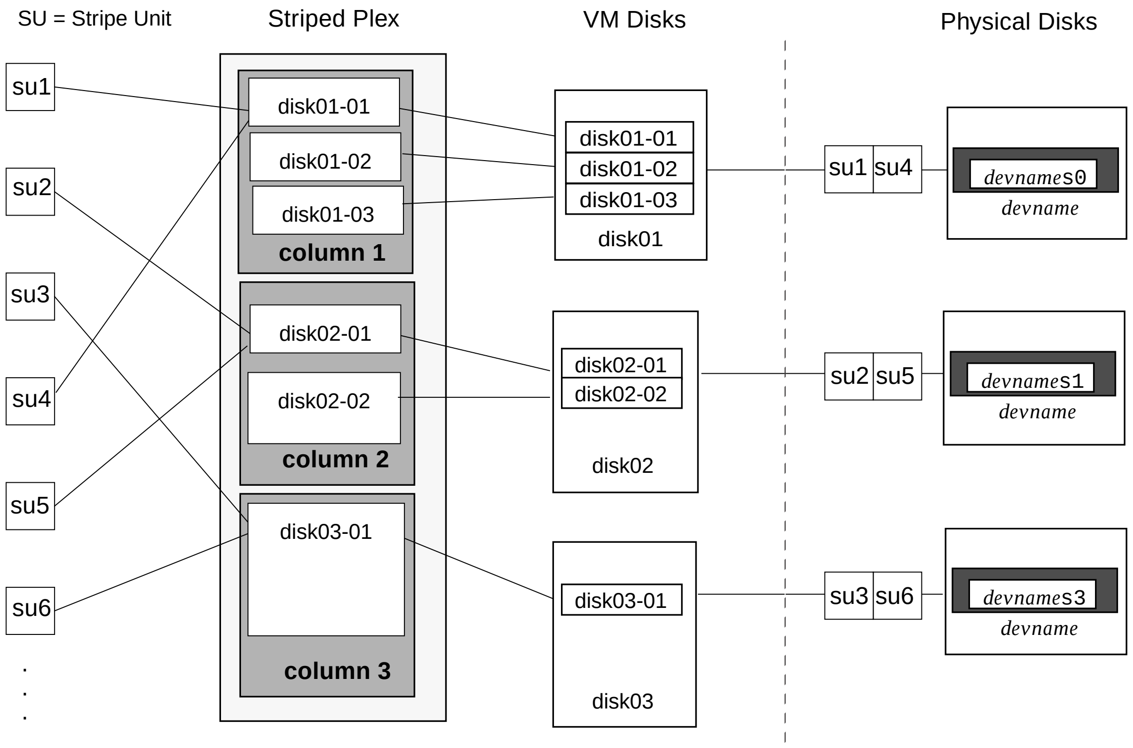
Figure 14 Example of a Striped Plex with One Subdisk per Column



The example in Figure 14 shows three subdisks that occupy all of the space on the VM disks. It is also possible for each subdisk in a striped plex to occupy only a portion of the VM disk, which leaves free space for other disk management tasks.

Figure 15 shows a striped plex with three columns containing subdisks of different sizes. Each column contains a different number of subdisks. There is one column per physical disk. Striped plexes can be created by using a single subdisk from each of the VM disks being striped across. It is also possible to allocate space from different regions of the same disk or from another disk (for example, if the plex is grown). Columns can contain subdisks from different VM disks if necessary.

Figure 15 Example of a Striped Plex with Concatenated Subdisks per Column



RAID-5

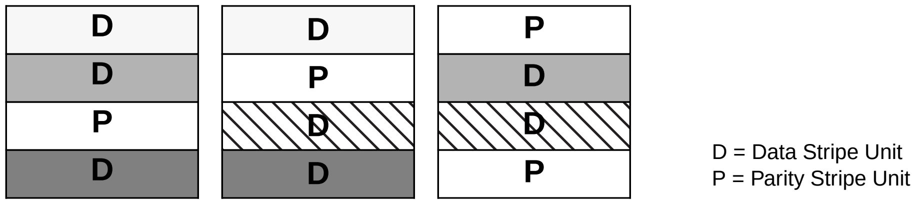
RAID-5 provides data redundancy by using *parity*. Parity is a calculated value used to reconstruct data after a failure. While data is being written to a RAID-5 volume, parity is calculated by doing an *exclusive OR* (XOR) procedure on the data. The resulting parity is then written to the volume. If 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 parity information.

RAID-5 volumes maintain redundancy of the data within a volume. RAID-5 volumes keep a copy of the data and calculated parity in a plex that is “striped” across multiple disks. In the event of a disk failure, a RAID-5 volume uses parity to reconstruct the data. It is possible to mix concatenation and striping in the layout.

RAID-5 volumes can do logging to minimize recovery time. RAID-5 volumes use RAID-5 logs to keep a copy of the data and parity currently being written. RAID-5 logging is optional and can be created along with RAID-5 volumes or added later.

Figure 16 shows parity locations in a RAID-5 array configuration. Every stripe has a column containing a parity stripe unit and columns containing data. The parity is spread over all of the disks in the array, reducing the write time for large independent writes because the writes do not have to wait until a single parity disk can accept the data.

Figure 16 Parity Locations in a RAID-5 Model



For more information on RAID-5 and how it is implemented by the Volume Manager, refer to “Volume Manager and RAID-5.”

Mirroring (RAID-1)

Mirroring uses multiple mirrors (plexes) to duplicate the information contained in a volume. In the event of a physical disk failure, the plex on the failed disk becomes unavailable, but the system continues to operate using the unaffected mirrors. Although a volume can have a single plex, at least two plexes are required to provide redundancy of data. Each of these plexes must contain disk space from different disks to achieve redundancy.

When striping or spanning across a large number of disks, failure of any one of those disks can make the entire plex unusable. The chance of one out of several disks failing is sufficient to make it worthwhile to consider mirroring in order to improve the reliability (and availability) of a striped or spanned volume.

Mirroring Plus Striping (RAID-1 + RAID-0)

The Volume Manager supports the combinations of mirroring plus striping. When used together on the same volume, mirroring plus striping offers the benefits of spreading data across multiple disks (striping) while providing redundancy (mirror) of data.

For mirroring plus striping to be effective when used together, the mirror and its striped plex must be allocated from separate disks. The layout type of the mirror can be concatenated or striped.

Striping Plus Mirroring (RAID-0 + RAID-1)

The Volume Manager supports the combination of striping with mirroring. In previous releases, whenever mirroring was used, the mirroring had to happen above striping. Now there can be mirroring both above and below striping.

Putting mirroring below striping mirrors each column of the stripe. If the stripe is large enough to have multiple subdisks per column, each subdisk can be individually mirrored. This layout enhances redundancy and reduces recovery time in case of an error.

In a mirror- stripe layout, if a disk fails, the entire plex is detached, thereby losing redundancy on the entire volume. When the disk is replaced, the entire plex must be brought up to date. Recovering the entire plex can take a substantial amount of time. If a disk fails in a stripe-mirror layout, only the failing subdisk must be detached, and only that portion of the volume loses redundancy. When the disk is replaced, only a portion of the volume needs to be recovered.

Compared to mirroring plus striping, striping plus mirroring offers a volume more tolerant to disk failure. If a disk failure occurs, the recovery time is shorter for striping plus mirroring. See “Layered Volumes” for more information.

Volume Manager and RAID-5

This section describes how Volume Manager implements RAID-5. For general information on RAID-5, refer to “RAID-5”.

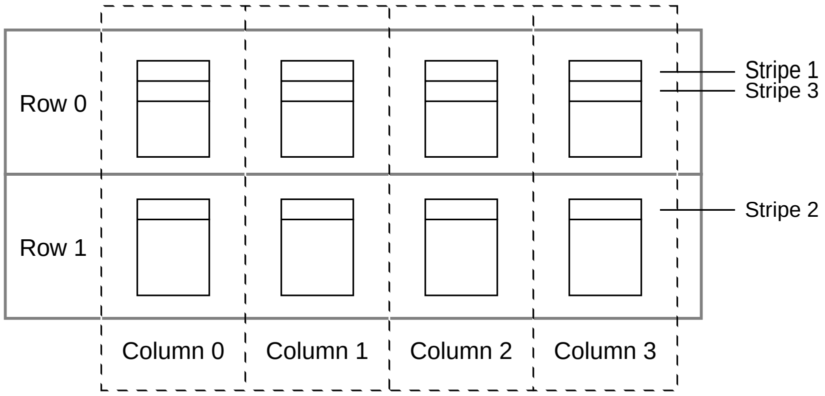
Although both mirroring (RAID-1) and RAID-5 provide redundancy of data, they use different methods. Mirroring provides data redundancy by maintaining multiple complete copies of the data in a volume. Data being written to a mirrored volume is reflected in all copies. If a portion of a mirrored volume fails, the system continues to use the other copies of the data.

RAID-5 provides data redundancy by using *parity*. Parity is a calculated value used to reconstruct data after a failure. While data is being written to a RAID-5 volume, parity is calculated by doing an *exclusive OR* (XOR) procedure on the 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 parity information.

Traditional RAID-5 Arrays

A *traditional* RAID-5 array is several disks organized in rows and columns. A *column* is a number of disks located in the same ordinal position in the array. A *row* is the minimal number of disks necessary to support the full width of a parity stripe. Figure 17 shows the row and column arrangement of a traditional RAID-5 array.

Figure 17 Traditional RAID-5 Array

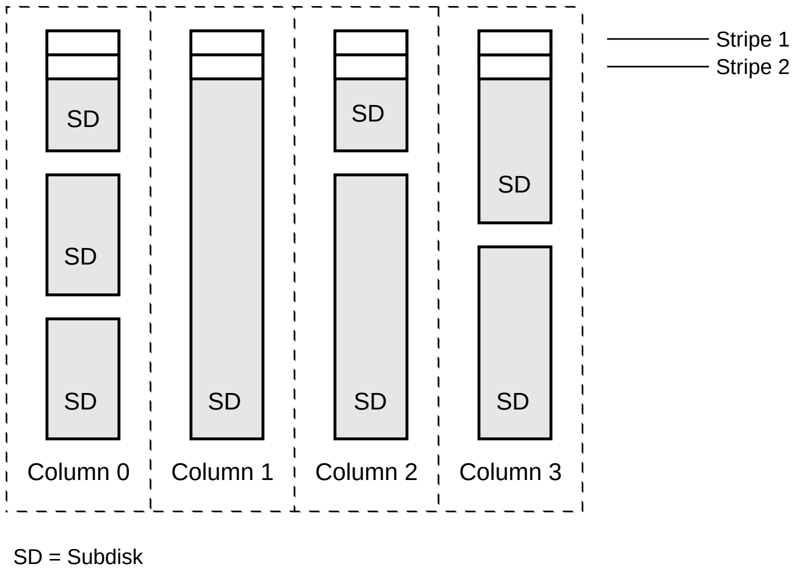


This traditional array structure supports growth by adding more rows per column. Striping is accomplished by applying the first stripe across the disks in Row 0, then the second stripe across the disks in Row 1, then the third stripe across the Row 0 disks, and so on. This type of array requires all disks (partitions), columns, and rows to be of equal size.

Volume Manager RAID-5 Arrays

The Volume Manager RAID-5 array structure differs from the traditional structure. Due to the virtual nature of its disks and other objects, the Volume Manager does not use rows. Instead, the Volume Manager uses columns consisting of variable length subdisks (as shown in Figure 18). Each subdisk represents a specific area of a disk.

Figure 18 Volume Manager RAID-5 Array



With the Volume Manager RAID-5 array structure, each column can consist of a different number of subdisks. The subdisks in a given column can be derived from different physical disks. Additional subdisks can be added to the columns as necessary. Striping (see “Striping (RAID-0),” is done by applying the first stripe across each subdisk at the top of each column, then another stripe below that, and so on for the length of the columns. For each stripe, an equal-sized stripe unit is placed in each column. With RAID-5, the default stripe unit size is 16 kilobytes.

Note: Mirroring of RAID-5 volumes is not currently supported.

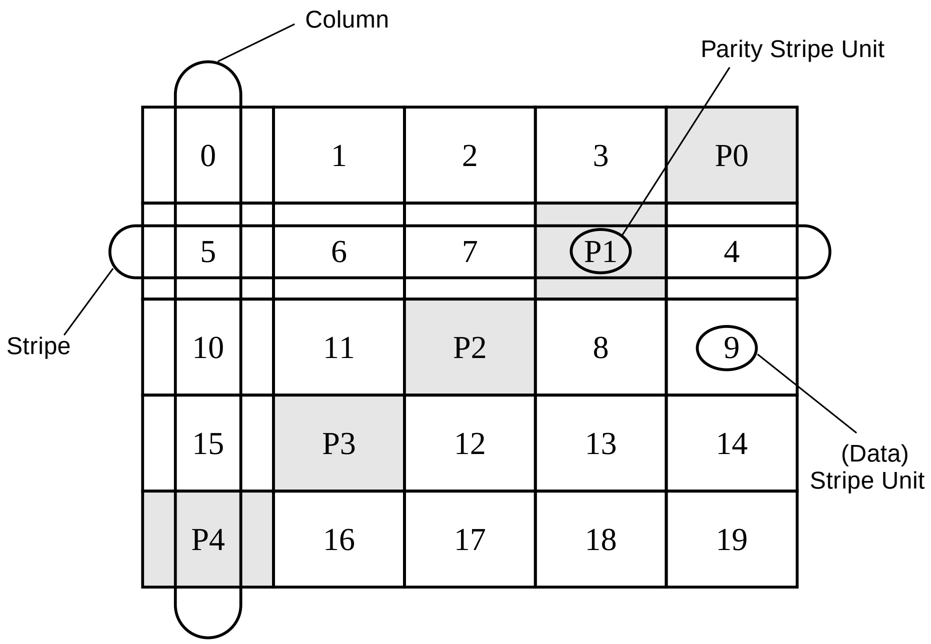
Left-Symmetric Layout

There are several layouts for data and parity that can be used in the setup of a RAID-5 array. The Volume Manager implementation of RAID-5 is the left-symmetric layout. The left-symmetric parity layout provides optimal performance for both random I/O operations and large sequential I/O operations. In terms of performance, the layout selection is not as critical as the number of columns and the stripe unit size selection.

The left-symmetric layout stripes both data and parity across columns, placing the parity in a different column for every stripe of data. The first parity stripe unit is located in the rightmost column of the first stripe. Each successive parity stripe unit is located in the next stripe, shifted left one column from the previous parity stripe unit location. If there are more stripes than columns, the parity stripe unit placement begins in the rightmost column again.

Figure 19 shows a left-symmetric parity layout with five disks (one per column).

Figure 19 Left-Symmetric Layout



For each stripe, data is organized starting to the right of the parity stripe unit. In Figure 19, data organization for the first stripe begins at P0 and continues to stripe units 0-3. Data organization for the second stripe begins at P1, then continues to stripe unit 4, and on to stripe units 5-7. Data organization proceeds in this manner for the remaining stripes.

Each parity stripe unit contains the result of an exclusive OR (XOR) procedure done on the data in the data stripe units within the same stripe. If data on a disk corresponding to one column is inaccessible due to hardware or software failure, data can be restored. Data is restored by XORing the contents of the remaining columns data stripe units against their respective parity stripe units (for each stripe).

For example, if the disk corresponding to the far left column in Figure 19 fails, the volume is placed in a degraded mode. While in degraded mode, the data from the failed column can be recreated by XORing stripe units 1-3 against parity stripe unit P0 to recreate stripe unit 0, then XORing stripe units 4, 6, and 7 against parity stripe unit P1 to recreate stripe unit 5, and so on.

Note: Failure of multiple columns in a plex with a RAID-5 layout detaches the volume. The volume is no longer allowed to satisfy read or write requests. Once the failed columns have been recovered, it may be necessary to recover the user data from backups.

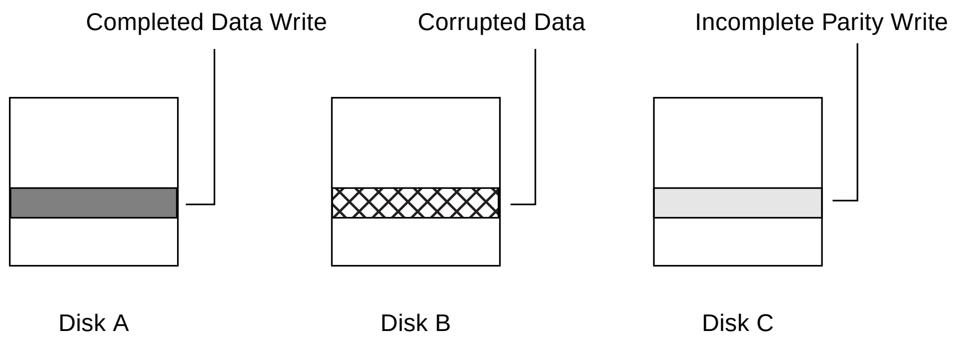
Logging

Logging (recording) is used to prevent corruption of recovery data. A log of the new data and parity is made on a persistent device (such as a disk-resident volume or non-volatile RAM). The new data and parity are then written to the disks.

Without logging, it is possible for data not involved in any active writes to be lost or silently corrupted if a disk fails and the system also fails. If this double-failure occurs, there is no way of knowing if the data being written to the data portions of the disks or the parity being written to the parity portions have actually been written. Therefore, the recovery of the corrupted disk may be corrupted itself.

In Figure 20, the recovery of Disk B is dependent on the data on Disk A and the parity on Disk C having both been completed. The diagram shows a completed data write and an incomplete parity write causing an incorrect data reconstruction for the data on Disk B.

Figure 20 Incomplete Write



This failure case can be avoided by logging all data writes before committing them to the array. In this way, the log can be replayed, causing the data and parity updates to be completed before the reconstruction of the failed drive takes place.

Logs are associated with a RAID-5 volume by being attached as additional, non-RAID-5 layout plexes. More than one log plex can exist for each RAID-5 volume, in which case the log areas are mirrored.

Layered Volumes

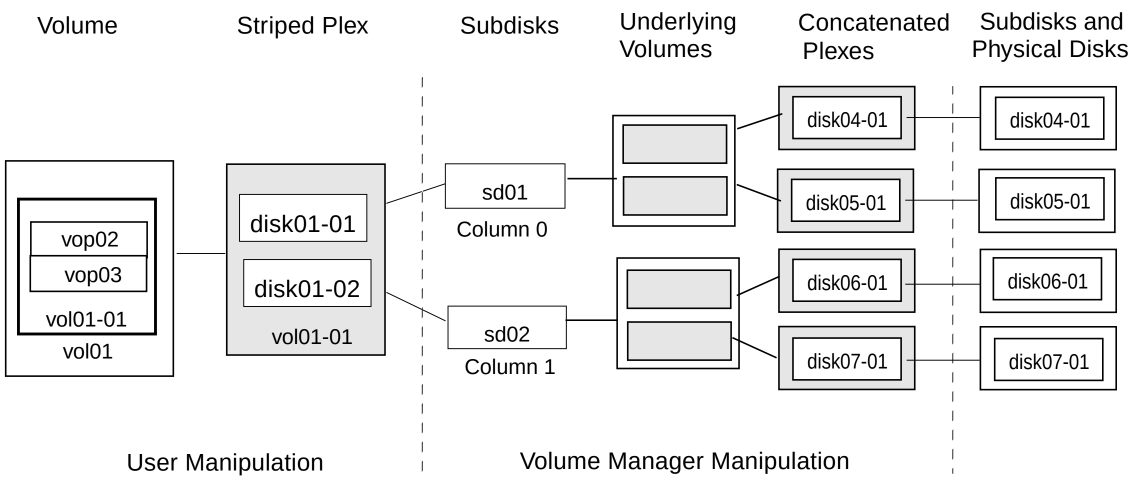
Another Volume Manager virtual object is the *layered volume*. A layered volume is built on top of volume(s). The layered volume structure tolerates failure better and has greater redundancy than the standard volume structure. For example, in a striped and mirrored layered volume, each mirror (plex) covers a smaller area of storage space, so recovery is quicker than with a standard mirror volume. Figure 21 shows an example layered volume design.

Note: Volume Manager 3.0 supports layered volumes and previous versions do not support layered volumes. Volume Manager 3.0 or higher allows subdisks to be built on volumes (storage volumes). For previous versions of Volume Manager, subdisks cannot be built on volumes.

As shown in Figure 21, the volume and striped plex in the "User Manipulation" area allow you to perform normal Volume Manager tasks. User tasks can be performed only on the top-level volume of a layered volume. You cannot detach a layered volume or perform any other operation on the underlying volumes by manipulating the internal structure. You can perform all necessary operations from the user manipulation area that includes the volume and striped plex (for example, to change the column width, or to add a column).

The "Volume Manager Manipulation" area shows subdisks with two columns, built on underlying volumes with each volume internally mirrored. Layered volumes are an infrastructure within Volume Manager and they allow the addition of certain features to be added to Volume Manager. Underlying volumes are used exclusively by the Volume Manager and are not designed for user manipulation. The underlying volume structure is described here to help you understand how layered volumes work and why they are used by Volume Manager.

Figure 21 Example of a Striped-Mirrored Layered Volume



System administrators may need to manipulate the layered volume structure for troubleshooting or other operations (for example, to place data on specific disks). Layered volumes are used by Volume Manager to perform these tasks and operations:

- striped-mirrors (see the `vxassist` manual page)
- concatenated mirrors (see the `vxassist` manual page)
- Online Relayout (see the `vxrelayout` and `vxassist` manual pages)
- RAID-5 subdisk moves (see the `vxsd` manual page)
- RAID-5 snapshot (see the `vxassist` manual page)

Volume Manager User Interfaces

This section briefly describes the VERITAS Volume Manager user interfaces.

User Interface Overview

The Volume Manager supports the following user interfaces:

- Volume Manager Storage Administrator

  The Storage Administrator is a graphical user interface to the Volume Manager. The Storage Administrator provides visual elements such as icons, menus, and dialog boxes to manipulate Volume Manager objects. The Storage Administrator also acts as an interface to some common file system operations. The Storage Administrator is described in the *Storage Administrator Administrator's Guide.*

- Command Line Interface

  Volume Manager commands range from simple commands to complex commands requiring detailed user input. Many Volume Manager commands require you to have an understanding of Volume Manager concepts. Volume Manager concepts are described in this chapter. Most Volume Manager commands require superuser or other appropriate privileges. The command line interface is described in the *Command Line Interface Administrator's Guide.*

- Volume Manager Support Operations

  Volume Manager Support Operations interface (`vxdiskadm`) is a menu-driven interface for disk and volume administration functions. vxdiskadm uses a command line main menu where you can select storage management tasks to be performed. `vxdiskadm` is described in Chapter 4, "Menu Interface Operations," of the *Command Line Interface Administrator's Guide*.

Volume Manager objects created by one interface are compatible with those created by the other interfaces.

Volume Manager Conceptual Overview

This section describes key terms and relationships between Volume Manager concepts. Figure 22 is a summary of the terms and concepts discussed in this section.

Why You Should Use Volume Manager

Volume Manager provides enhanced data storage service by separating the physical and logical aspects of data management. Volume Manager enhances data storage by controlling these aspects of storage:

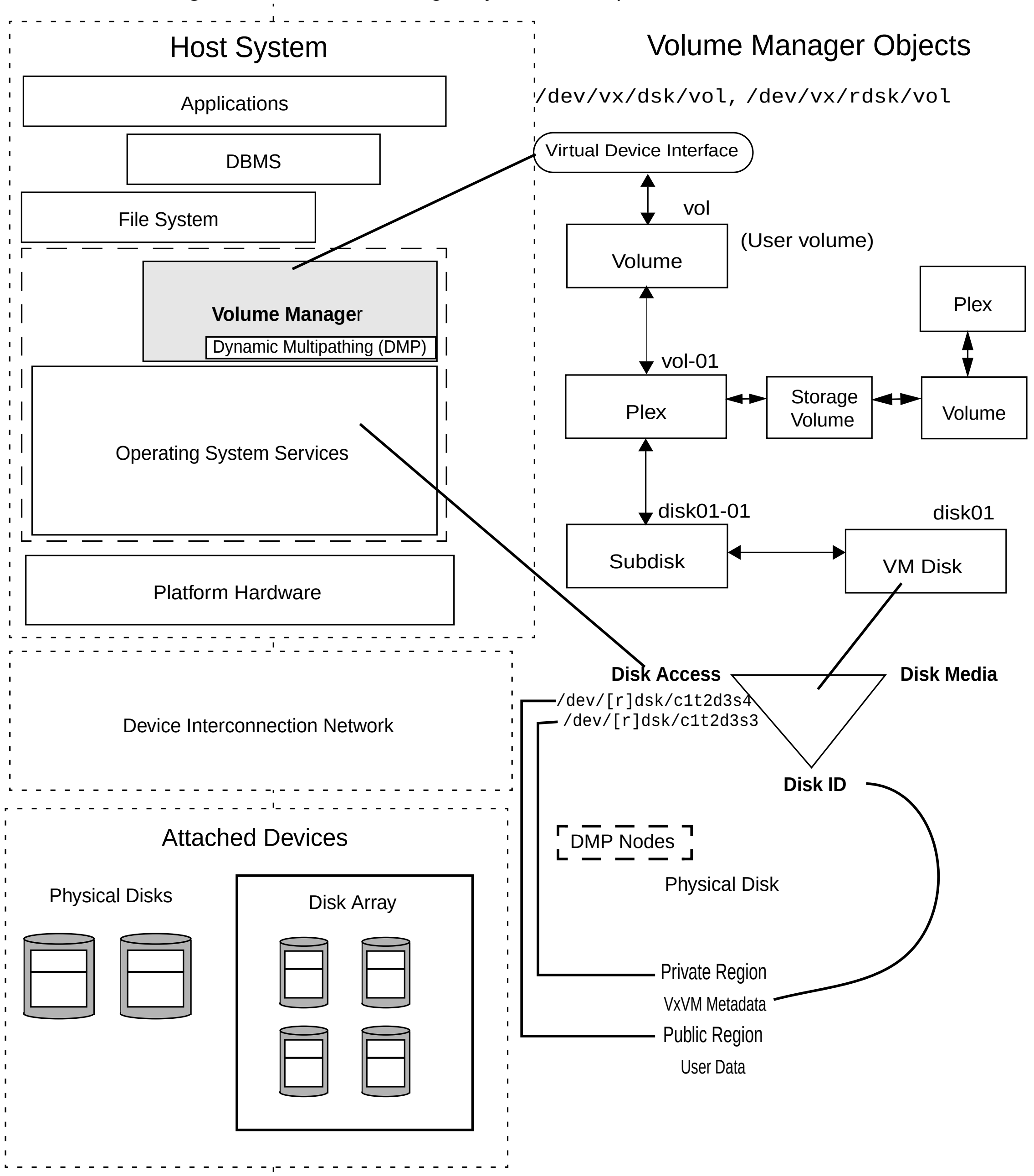
- space—allocation and use
- performance—by enhancing data delivery
- data availability—continuous operation and multisystem access
- device installation—centralized and optimized support
- system—multisystem support and monitoring of private/shared systems

Volume Manager Objects

Volume Manager is a storage management subsystem that allows you to manage physical disks as *logical* devices called *volumes*. The Volume Manager interfaces provide enhanced data access and storage management by using volumes. A volume is a logical device that appears to data management systems as a physical disk partition device. Volumes provide enhanced recovery, data availability, performance, and storage configuration options.

A Volume Manager volume is a logical *object*. Volume Manager creates other objects that you can operate, control, monitor, and query to optimize storage management. To configure and maintain a volume for use, Volume Manager places physical disks under its control and collects the disk space into *disk groups*. A disk group is a collection of claimed disks organized into logical volumes. Volume Manager then allocates the space on those disks to logical volumes.

Figure 22 Volume Manager System Concepts



For example, after installing Volume Manager on a host system, you need to do the following steps before you can configure and use Volume Manager objects:

- bring the contents of physical disks under Volume Manager control
- collect the Volume Manager disks into disk groups
- allocate the disk group space to create logical volumes

Bringing the contents of physical disks under Volume Manager control is done only if:

- you allow Volume Manager to take control of the physical disks
- the disk is not under control of another storage manager

Volume Manager writes identification information on physical disks under Volume Manager control (claimed disks). Claimed disks can be identified even after physical disk disconnection or system outages. Volume Manager can then re-form disk groups and logical objects to provide failure detection and to speed system recovery.

Volume Manager and the Operating System

Volume Manager operates as a subsystem between your operating system and your data management systems, such as file systems and database management systems.

Before a disk can be brought under Volume Manager control, the disk must be accessible through the operating system device interface. Volume Manager is a subsystem layered on top of the operating system interface services. Therefore, Volume Manager is dependent upon how the operating system accesses physical disks.

Volume Manager is dependent upon the operating system for the following.

- operating system (disk) devices
- device handles
- VM disks
- Volume Manager dynamic multipathing (DMP) metadevice

Dynamic Multipathing (DMP)

A multipathing condition can exist when a physical disk can be accessed by more than one operating system device handle. Each multipath operating system device handle permits data access and control through alternate host-to-device pathways.

Volume Manager can be configured with its own DMP system to organize access to multipathed devices. Volume Manager detects multipath systems by using the Universal World-Wide-Device Identifiers (WWD IDs). The physical disk must provide unambiguous identification through its WWD ID for DMP to access the device.

If DMP cannot identify the physical disk through its WWD ID, identification is left to the Volume Manager device detection methods. Device detection depends on Volume Manager recognizing on-disk metadata identifiers.

Volume Manager DMP creates metanodes representing metadevices for each multipath target that it has detected. Each metanode is mapped to a set of operating system device handles and configured with an appropriate multipathing policy. Volume Manager DMP creates metanodes for all attached physical disks accessible through an operating system device handle.

Volume Manager DMP manages multipath targets, such as disk arrays, which define policies for using more than one path. Some disk arrays permit more than one path to be concurrently active (Active / Active). Some disk arrays permit only one path to be active, holding an alternate path as a spare in case of failure on the existing path (Active / Passive). Some disk arrays have more elaborate policies.

In general, Volume Manager is designed so that the VM disk is mapped to one Volume Manager DMP metanode. To simplify VxVM logical operations, each VM disk is mapped to a unique Volume Manager DMP metanode. The mapping occurs whether or not the physical disk device is connected in a multipathing configuration.

When using Volume Manager DMP, you should be aware of the layering of device recognition:

- How does the operating system view the paths?
- How does Volume Manager DMP view the paths?
- How does the Multipathing target deal with their paths?

Additional information about DMP can be found in the following documents:

- Getting Started Guide (this document), chapter 2
- Command Line Interface Administrator's Guide:
    - Chapter 2, Displaying Multipaths Under a VM Disk
    - Chapter 4, Volume Configuration Daemon `vxdctl`
    - Chapter 5, Disk Tasks
- Administrator's Reference Guide:
    - Chapter 2, Recovering the Volume Manager Configuration
    - Chapter 3, Using vxdisk To Display Multipathing Information
    - Appendix A, DMP Error Messages

Volume Manager Layouts

A Volume Manager virtual device is defined by a volume. A volume has a layout defined by the association of a volume to one or more plexes, which in turn, each map to subdisks. The volume then presents a virtual device interface exposed to Volume Manager clients for data access. These logical building blocks re-map the volume address space through which I/O is re-directed at run-time.

Different volume layouts each provide different levels of storage service. A volume layout can be configured and re-configured to match particular levels of desired storage service.

In previous releases of Volume Manager, the subdisk was restricted to mapping directly to a VM disk. This allowed the subdisk to define a contiguous extent of storage space backed by the public region of a VM disk. When active, the VM disk is associated with an underlying physical disk, this is how Volume Manager logical objects map to physical objects, and stores data on stable storage.

The combination of a volume layout and the physical disks which provide backing store, therefore determine the storage service available from a given virtual device.

In the 3.0 or higher release of Volume Manager “layered volumes” can be constructed by permitting the subdisk to map either to a VM disk as before, or, to a new logical object called a *storage volume*. A storage volume provides a recursive level of mapping with layouts similar to the top-level volume. Eventually, the “bottom” of the mapping requires an association to a VM disk, and hence to attached physical storage.

Layered volumes allow for more combinations of logical compositions, some of which may be desirable for configuring a virtual device. Because permitting free use of layered volumes throughout the command level would have resulted in unwieldy administration, some ready-made layered volume configurations have been designed into the 3.0 release of the Volume Manager.

These ready-made configurations operate with built-in rules to automatically match desired levels of service within specified constraints. The automatic configuration is done on a “best-effort” basis for the current command invocation working against the current configuration.

To achieve the desired storage service from a set of virtual devices, it may be necessary to include an appropriate set of VM disks into a disk group, and to execute multiple configuration commands.

To the extent that it can, the 3.0 release of Volume Manager handles initial configuration and on-line re-configuration with its set of layouts and administration interface to make this job easier and more deterministic.ystems

Volume Manager Operations 2

Introduction

This chapter provides detailed information about the VERITAS Volume Manager features.

The following topics are covered in this chapter:

- Online Relayout
- Hot-Relocation
- Volume Resynchronization
- Dirty Region Logging
- Volume Manager Rootability
  - Booting With Root Volumes
  - Boot-time Volume Restrictions
- Dynamic Multipathing (DMP)
  - Path Failover Mechanism
  - Load Balancing
  - Booting From DMP Devices
  - Enabling and Disabling of Controllers
  - Displaying DMP Database Information
- VxSmartSync Recovery Accelerator

    - Data Volume Configuration
    - Redo Log Volume Configuration
- Volume Manager Task Monitor
- Volume Manager Cluster Functionality

Online Relayout

Online Relayout allows you to convert any supported storage layout in the Volume Manager to any other, in place, with *uninterrupted data access*.

You usually change the storage layout in the Volume Manager to change the redundancy or performance characteristics of the storage. The Volume Manager adds redundancy to storage either by duplicating the address space (mirroring) or by adding parity (RAID-5). Performance characteristics of storage in the Volume Manager can be changed by changing the striping parameters which are the number of columns and the stripe width.

Layout changes can be classified into these types:

- RAID-5 to mirroring and mirroring to RAID-5
- adding or removing parity
- adding or removing columns
- changing stripe width

Storage Layout

Online Relayout currently supports these storage layouts:

- concatenation
- striped
- RAID-5
- mirroring (also supported where data is duplicated across different storage devices)
- striped-mirror
- concatenated-mirror

How Online Relayout Works

The VERITAS Online Relayout feature allows you to change storage layouts that you have already created in place, without disturbing data access. You can change the performance characteristics of a particular layout to suit changed requirements.

For example, you can have a striped layout with a 128K stripe unit size that may not be providing optimal performance. You can change the stripe unit size of the layout by using the Relayout feature. You can transform one layout to another by invoking a single command.

File systems mounted on the volumes do not need to be unmounted to achieve this transformation as long as the file system provides online shrink and grow operations.

Online Relayout reuses the existing storage space and has space allocation policies to address the needs of the new layout. The layout transformation process converts a given volume to the destination layout by using minimal temporary space.

The transformation is done by moving a portion-at-a-time of data in the source layout to the destination layout. Data is copied from the source volume to the temporary space, which removes data from the source volume storage area in portions. The source volume storage area is then transformed to the new layout and data saved in the temporary space is written back to the new layout. This operation is repeated until all the storage and data in the source volume has been transformed to the new layout.

You can use Online Relayout to change the number of columns, change stripe width, remove and add parity, and change RAID-5 to mirroring.

Types of Transformation

At least one of the following criteria must be met for an Online Relayout operation to be effective. You must perform one or more of the following operations:

- RAID-5 to or from mirroring
- change the number of columns
- change the stripe width

- remove or add parity

To be eligible for layout transformation, mirrored volume plexes must be identical in layout, with the same stripe width and number of columns.

In Table 1:

- Yes indicates that an Online Relayout operation is possible.
- No indicates that the operation *may* be possible but you cannot use Relayout.
- The numbers in the table point to a brief description of the possible changes in that particular layout transformation.
- The operation described can be performed in both directions.

Table 1 shows the layout transformations supported.

Table 1 Layout Transformations Supported

From/To	**Striped Mirrored**	**Concatenated Mirrored**	**Regular Mirrored**	**RAID-5**	**Concatenated**	**Striped**
Striped Mirrored	Yes 1	Yes 2	No 3	Yes 4	Yes 5	Yes 6
Concatenated Mirrored	Yes 7	No 8	No 9	Yes 10	No 11	Yes 12
Regular Mirrored	Yes 13	Yes 14	No 15	Yes 16	No 17	No 18
RAID-5	Yes 4	Yes 10	No 19	Yes 20	Yes 21	Yes 22
Concatenated	Yes 5	No 11	No 17	Yes 21	No 23	Yes 24
Striped	Yes 6	Yes 12	No 18	Yes 22	Yes 24	Yes 25

The numbers in Table 1 describe the Relayout operation as follows:

1. This can be used to change the stripe width or number of columns.
2. All of the columns are removed.
3. This is not a Relayout, but a convert operation.

4. Change mirroring to RAID-5 and/or stripe width/column changes.
5. This changes mirroring to RAID-5 and/or stripe width/column changes.
6. You use this to change stripe width/column and remove a mirror.
7. Add columns.
8. This is not a Relayout operation.
9. This is a convert operation.
10. Mirroring to RAID-5. See `vxconvert`.
11. Remove a mirror, not a Relayout operation.
12. Remove a mirror and add striping.
13. An old mirrored volume changed into a Stripe Mirror, Relayout is valid only if there are changes to columns/stripe width, otherwise this is a convert operation. See `vxconvert`.
14. An old mirrored volume changed into a Concatenated-Mirror. Relayout is valid only if there are changes to columns, otherwise this is a convert operation.
15. No change. Not a Relayout operation.
16. Change an old mirrored volume to RAID-5. You have to choose a plex in the old mirrored volume to use Relayout. The other plex is removed at the end of the Relayout operation.
17. Unless you choose a plex in the mirrored volume and change the column/stripe width, this is not a Relayout operation.
18. Unless you choose a plex in the mirrored volume and change the column/stripe width, this is not a Relayout operation.
19. This is not a Relayout operation.
20. Change stripe width/column.
21. Remove parity and all columns.
22. Remove parity.
23. No change. Not a Relayout operation.

24. Remove columns.

25. Change Stripe width / number of columns.

A striped mirror plex is a striped plex on top of a mirrored volume, resulting in a single plex that has both mirroring and striping. This combination forms a plex called a striped-mirror plex. A concatenated plex can be created in the same way. Online Relayout supports transformations to and from striped-mirror and concatenated-mirror plexes. *Changing the number of mirrors during a transformation is not currently supported.*

Transformation Characteristics

Transformation of data from one layout to another involves rearrangement of data in the existing layout to the new layout.

During the transformation, Online Relayout retains data redundancy by mirroring any temporary space used.

Read / write access to data is not interrupted during the transformation.

Data is not corrupted if the system fails during a transformation. The transformation continues after the system is restored and read / write access is maintained.

You can reverse the layout transformation process at any time, but the data may not be returned to the exact previous storage location. Any existing transformation in the volume should be stopped before doing a reversal.

You can determine the transformation direction by using the `vxrelayout status` command.

These transformations eliminate I / O failures as long as there is sufficient redundancy to move the data.

Transformations and Volume Length

Some layout transformations can cause the volume length to increase or decrease. If the layout transformation causes the volume length to increase or decrease, Online Relayout uses `vxresize` to shrink or grow a file system.

Sparse plexes are not transformed by Online Relayout, and no plex can be rendered sparse by Online Relayout.

Online Relayout can be used only with volumes created with the `vxassist` command.

Transformations Not Supported

Transformation of log plexes is not supported.

A snapshot of a volume when there is an Online Relayout operation running in the volume is not supported.

Hot-Relocation

Hot-relocation allows a system to automatically react to I/O failures on redundant (mirrored or RAID-5) Volume Manager objects and restore redundancy and access to those objects. The Volume Manager detects I/O failures on objects and relocates the affected subdisks. The subdisks are relocated to disks designated as *spare disks* and/or free space within the disk group. The Volume Manager then reconstructs the objects that existed before the failure and makes them redundant and accessible again.

When a partial disk failure occurs (that is, a failure affecting only some subdisks on a disk), redundant data on the failed portion of the disk is relocated. Existing volumes on the unaffected portions of the disk remain accessible.

Note: Hot-relocation is only performed for redundant (mirrored or RAID-5) subdisks on a failed disk. Nonredundant subdisks on a failed disk are not relocated, but the system administrator is notified of the failure.

How Hot-Relocation Works

The hot-relocation feature is enabled by default. No system administrator action is needed to start hot-relocation when a failure occurs.

The hot-relocation daemon, `vxrelocd`, monitors Volume Manager for events that affect redundancy and performs hot-relocation to restore redundancy. `vxrelocd` also notifies the system administrator (via electronic mail) of failures and any relocation and recovery actions. See the `vxrelocd`(1M) manual page for more information on `vxrelocd`.

The `vxrelocd` daemon starts during system startup and monitors the Volume Manager for failures involving disks, plexes, or RAID-5 subdisks. When a failure occurs, it triggers a hot-relocation attempt.

A successful hot-relocation process involves:

1. Detecting Volume Manager events resulting from the failure of a disk, plex, or RAID-5 subdisk.
2. Notifying the system administrator (and other designated users) of the failure and identifying the affected Volume Manager objects. This is done through electronic mail.
3. Determining which subdisks can be relocated, finding space for those subdisks in the disk group, and relocating the subdisks. Notifying the system administrator of these actions and their success or failure.
4. Initiating any recovery procedures necessary to restore the volumes and data. Notifying the system administrator of the outcome of the recovery attempt.

Note: Hot-relocation does not guarantee the same layout of data or the same performance after relocation. The system administrator may make some configuration changes after hot-relocation occurs.

How Space is Chosen for Relocation

A spare disk must be initialized and placed in a disk group as a spare *before* it can be used for replacement purposes. If no disks have been designated as spares when a failure occurs, Volume Manager automatically uses any available free space in the disk group in which the failure occurs. If there is not enough spare disk space, a combination of spare space and free space is used.

The system administrator can designate one or more disks as hot-relocation spares within each disk group. Disks can be designated as spares by using the Storage Administrator interface, `vxdiskadm`, or `vxedit` (as described in the *VERITAS Volume Manager Administrator's Reference Guide*). Disks designated as spares do not participate in the free space model and should not have storage space allocated on them.

When selecting space for relocation, hot-relocation preserves the redundancy characteristics of the Volume Manager object that the relocated subdisk belongs to. For example, hot-relocation ensures that subdisks from a failed plex are not relocated to a disk containing a mirror of the failed plex. If redundancy cannot be preserved using any available spare disks and/or free space, hot-relocation does not take place. If relocation is not possible, the system administrator is notified and no further action is taken.

When hot-relocation takes place, the failed subdisk is removed from the configuration database and Volume Manager ensures that the disk space used by the failed subdisk is not recycled as free space.

For information on how to take advantage of hot-relocation, refer to Chapter 3, “Volume Manager Initialization and Setup.” Refer to the *VERITAS Volume Manager Installation Guide* for information on how to disable hot-relocation.

Volume Resynchronization

When storing data redundantly, using mirrored or RAID-5 volumes, the Volume Manager ensures that all copies of the data match exactly. However, under certain conditions (usually due to complete system failures), some redundant data on a volume can become inconsistent or *unsynchronized*. The mirrored data is not exactly the same as the original data. Except for normal configuration changes (such as detaching and reattaching a plex), this can only occur when a system crashes while data is being written to a volume.

Data is written to the mirrors of a volume in parallel, as is the data and parity in a RAID-5 volume. If a system crash occurs before all the individual writes complete, it is possible for some writes to complete while others do not. This can result in the data becoming unsynchronized. For mirrored volumes, it 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, it can lead to parity corruption and incorrect data reconstruction.

The Volume Manager needs to ensure 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*. For volumes that are part of disk groups that are automatically imported at boot time (such as `rootdg`), the resynchronization process takes place when the system reboots.

Not all volumes require resynchronization after a system failure. Volumes that were never written or that were quiescent (that is, had no active I/O) when the system failure occurred could not have had outstanding writes and do not require resynchronization.

The Volume Manager records when a volume is first written to and marks it as *dirty*. When a volume is closed by all processes or stopped cleanly by the administrator, all writes have been completed and the Volume Manager removes the dirty flag for the volume. Only volumes that are marked dirty when the system reboots require resynchronization.

The process of resynchronization depends on the type of volume. RAID-5 volumes that contain RAID-5 logs can “replay” those logs. If no logs are available, the volume is placed in reconstruct-recovery mode and all parity is regenerated. For mirrored volumes, resynchronization is done by placing the volume in recovery mode (also called *read-writeback recovery mode*). Resynchronizing of data in the volume is done in the background. This allows the volume to be available for use while recovery is taking place.

The process of resynchronization can be expensive and can impact system performance. The recovery process reduces some of this impact by spreading the recoveries to avoid stressing a specific disk or controller.

For large volumes or for a large number of volumes, the resynchronization process can take time. These effects can be addressed by using Dirty Region Logging for mirrored volumes, or by ensuring that RAID-5 volumes have valid RAID-5 logs. For volumes used by database applications, the VxSmartSync™ Recovery Accelerator can be used (see “VxSmartSync Recovery Accelerator”).

Dirty Region Logging

Dirty Region Logging (DRL) is an optional property of a volume, used to provide a speedy recovery of mirrored volumes after a system failure. 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 the portions of the volume that need to be recovered.

If DRL is not used and a system failure occurs, all mirrors of the volumes must be restored to a consistent state. Restoration is done by copying the full contents of the volume between its mirrors. This process can be lengthy and I/O intensive. It may also be necessary to recover the areas of volumes that are already consistent.

DRL logically divides a volume into a set of consecutive regions. It keeps track of volume regions that are being written to. A dirty region log is maintained that contains a status bit representing each region of the volume. For any write operation to the volume, the regions being written are marked dirty in the log before the data is written. If a write causes a log region to become dirty when it was previously clean, the log is synchronously written to disk before the write operation can occur. On system restart, the Volume Manager recovers only those regions of the volume that are marked as dirty in the dirty region log.

Log subdisks are used to store the dirty region log of a volume that has DRL enabled. A volume with DRL has at least one log subdisk; multiple log subdisks can be used to mirror the dirty region log. Each log subdisk is associated with one plex of the volume. Only one log subdisk can exist per plex. If the plex contains only a log subdisk and no data subdisks, that plex can be referred to as a *log plex*.

The log subdisk can also be associated with a regular plex containing data subdisks. In that case, the log subdisk risks becoming unavailable if the plex must be detached due to the failure of one of its data subdisks.

If the `vxassist` command is used to create a dirty region log, it creates a log plex containing a single log subdisk by default. A dirty region log can also be created manually by creating a log subdisk and associating it with a plex. Then the plex can contain both a log subdisk and data subdisks.

Only a limited number of bits can be marked dirty in the log at any time. The dirty bit for a region is not cleared immediately after writing the data to the region. Instead, it remains marked as dirty until the corresponding volume region becomes the least recently used. If a bit for a given region is already marked dirty and another write to the same region occurs, it is not necessary to write the log to the disk before the write operation can occur.

Note: DRL adds a small I/O overhead for most write access patterns.

Volume Manager Rootability

The Volume Manager can place various files from different systems (for example, the root file system, `swap` device, `usr` file system, and `stand` file system) under Volume Manager control. This is called *rootability*. The *root disk* (that is, the disk containing the root file system) can be put under Volume Manager control through the process of *encapsulation*.

Encapsulation converts existing partitions on that disk to volumes. Once under Volume Manager control, the `root` and `swap` devices appear as volumes and provide the same characteristics as other Volume Manager volumes. A volume that is configured for use as a swap area is referred to as a *swap volume*; a volume that contains the root file system is referred to as a *root volume*; and a volume that contains the stand file system is referred to as a *stand volume*.

It is possible to mirror the `rootvol`, `swapvol,` and `standvol` volumes, as well as other parts of the root disk that are required for a successful boot of the system (for example, `/usr`). This provides complete redundancy and recovery capability in the event of disk failure. Without Volume Manager rootability, the loss of the `root`, `swap`, `usr`root, or stand partition prevents the system from being booted from surviving disks.

Mirroring disk drives critical to booting ensures that no single disk failure renders the system unusable. A suggested configuration is to mirror the critical disk onto another available disk (using the `vxdiskadm` command). If the disk containing the `root`, `stand`, and `swap` partitions fails, the system can be rebooted from the disk containing the root mirror. For more information on mirroring the boot (root) disk and system recovery procedures, see Chapter 1, “Recovery,” in the *Administrator’s Reference Guide*.

Booting With Root Volumes

Ordinarily, when the operating system is booted, the `root` file system, `stand` file system, and `swap` area are available for use early in the boot procedure. This is before user processes can be run to load the Volume Manager configuration and start volumes. The `root`, `stand`, and `swap` device configurations must be completed prior to starting the Volume Manager. Starting the Volume Manager `vxconfigd` daemon as part of the `init` process is too late to configure volumes for use as a `root` or `swap` device.

To avoid this restriction, the mirrors of the `rootvol`, `standvol`, and `swapvol` volumes are accessed by the system during startup. During startup, the system sees the `rootvol`, `standvol`, and `swapvol` volumes as regular partitions and accesses them using standard partition numbering. `rootvol`, `standvol`, and `swapvol` volumes are created from contiguous disk space that is also mapped by a single partition for each. Due to this restriction, it is not possible to stripe or span the primary plex (that is, the plex used for booting) of a `rootvol`, `standvol`, or `swapvol` volume. Any mirrors of these volumes needed for booting cannot be striped or spanned.

Boot-time Volume Restrictions

The `rootvol`, `standvol`, `swapvol`, and `usr` volumes differ from other volumes in that they have very specific restrictions on the configuration of the volumes:

- The root volume (`rootvol`) must exist in the default disk group, `rootdg`. Although other volumes named `rootvol` can be created in disk groups other than `rootdg`, only the `rootvol` in `rootdg` can be used to boot the system.
- A `rootvol` volume has a specific minor device number: minor device 0. Also, `swapvol` has minor device number 1. The `usr` volume does not have a specific minor device number. See “Reserving Minor Numbers for Disk Groups,” in the *Administrator’s Reference Guide*.
- Restricted mirrors of `rootvol`, `var`, `usrrootvol`, `standvol`, and `swapvol` devices have “overlay” partitions created for them. An overlay partition is one that exactly includes the disk space occupied by the restricted mirror. During boot, before the `rootvol`, `var`, `usrrootvol`, `standvol`, and `swapvol` volumes are fully configured, the default volume configuration uses the overlay partition to access the data on the disk. (See “Booting With Root Volumes.”)
- Although it is possible to add a striped mirror to a `rootvol` device for performance reasons, you cannot stripe the primary plex or any mirrors of `rootvol` that may be needed for system recovery or booting purposes if the primary plex fails.
- `rootvol`, `standvol`, and `swapvol` cannot be spanned or contain a primary plex with multiple noncontiguous subdisks.

- When mirroring parts of the boot disk, the disk being mirrored to must be large enough to hold the data on the original plex, or mirroring may not work.
- `rootvol`, `standvol`, `swapvol`, and `usr` cannot be Dirty Region Logging volumes.

In addition to these requirements, it is a good idea to have at least one contiguous, (cylinder-aligned if appropriate) mirror for each of the volumes for `root`, `usr`, `var`, `opt`, `varadm`, `usrkvm`, and `swap`. This makes it easier to convert these from volumes back to regular disk partitions (during an operating system upgrade, for example).

Dynamic Multipathing (DMP)

On some systems, the Volume Manager supports multiported disk arrays. It automatically recognizes multiple I/O paths to a particular disk device within the disk array. The Dynamic Multipathing feature of the Volume Manager provides greater reliability by providing a path failover mechanism. In the event of a loss of one connection to a disk, the system continues to access the critical data over the other sound connections to the disk. DMP also provides greater I/O throughput by balancing the I/O load uniformly across multiple I/O paths to the disk device.

In the Volume Manager, all the physical disks connected to the system are represented as metadevices with one or more physical access paths. A single physical disk connected to the system is represented by a metadevice with one path. A disk that is part of a disk array is represented by a metadevice that has two physical access paths. You can use the Volume Manager administrative utilities such as `vxdisk` to display all the paths of a metadevice and status information of the various paths.

Path Failover Mechanism

DMP enhances system reliability when used with multiported disk arrays. In the event of the loss of one connection to the disk array, DMP automatically selects the next I/O paths for the I/O requests dynamically without action from the administrator.

DMP allows the administrator to indicate to the DMP subsystem in the Volume Manager whether the connection is repaired or restored. This is called DMP reconfiguration.The reconfiguration procedure also allows the detection of newly added devices, as well as devices that are removed after the system is fully booted (only if the operating system sees them properly).

Load Balancing

To provide load balancing across paths, DMP follows the *balanced path mechanism* for active/active disk arrays. Load balancing makes sure that I/O throughput can be increased by utilizing the full bandwidth of all paths to the maximum. However, sequential I/Os to a disk are sent down the same path to optimize I/O throughput. This is done in order to utilize the effect of disk track caches.

For active/passive disk arrays, I/Os are sent down the primary path until it fails. Once the primary path fails, I/Os are then switched over to the other available primary paths or secondary paths. Load balancing across paths is not done for active/passive disk arrays to avoid the continuous transfer of ownership of LUNs from one controller to another, thus resulting in severe I/O slowdown.

Booting From DMP Devices

When the root disk is placed under Volume Manager control, it is automatically accessed as a DMP device with one path if it is a single disk, or with multiple paths if the disk is part of a multiported disk array. By encapsulating the root disk, system reliability is enhanced against loss of one or more of the existing physical paths to a disk.

Enabling and Disabling of Controllers

This feature allows the administrator to turn off I/Os to a host I/O controller to perform administrative operations. It can be used for maintenance of controllers attached to the host or a disk array supported by Volume Manager. I/O operations to the host I/O controller can be turned on after the maintenance task is completed. This once again enables I/Os to go through this controller. This operation can be accomplished using the `vxdmpadm(1M)` utility provided with Volume Manager.

For example, if the system has a StorEdge A5000(TM) array and the user needs to change an A5000 Interface Board connected to this disk array, the `vxdmpadm(1M)` command should be used to get a list of host I/O controllers connected on this A5000 interface board. These should be disabled. Once this is done, further I/Os to the disks which are accessed through these controller(s) is stopped.

The Interface Board can then be replaced without causing any disruption to ongoing I/Os to disks present on this disk array. This is required because normally, for Active/Active type disk arrays (as in this example), Volume Manager uses the *Balanced Path mechanism* to schedule I/Os to a disk with multiple paths to it. As a result, I/Os may go through any path at any given point in time.

For Active/Passive type disk arrays, I/Os will be scheduled by Volume Manager to the primary path until a failure is encountered. Therefore, to change an interface card on the disk array or a card on the host (when possible) that is connected to the disk array, the I/O operations to the host I/O controller(s) should be disabled. This allows all I/Os to be shifted over to an active secondary path or an active primary path on another I/O controller before the hardware is changed.

After the operation is over, the paths through these controller(s) can be put back into action by using the enabled option of the `vxdmpadm(1M)` command.

Volume Manager does not allow you to disable the last active path to the rootdisk.

Displaying DMP Database Information

The `vxdmpadm(1M)` utility can be used to list DMP database information and perform other administrative tasks. This command allows the user to list all the controllers on the systems (connected to disks) and other related information stored in the DMP database. This information can be used to locate system hardware and also make a decision regarding which controllers to enable/disable.

It also provides you with other useful information such as disk array serial number and the list of DMP devices (disks) that are connected to the disk array, the list of paths that go through a particular controllers, etc.

VxSmartSync Recovery Accelerator

The VxSmartSync™ Recovery Accelerator is available for some systems. VxSmartSync for Mirrored Oracle® Databases is a collection of features that speed up the resynchronization process (known as *resilvering*) for volumes used in with the Oracle Universal Database™. These features use an extended interface between Volume Manager volumes and the database software so they can avoid unnecessary work during mirror resynchronization. These extensions can result in an order of magnitude improvement in volume recovery times.

To enable the VxSmartSync feature, the `_ENABLE_ORACLE_RESILVERING` parameter must be set to `TRUE` in the Oracle instance startup parameter file. To do this, add the following line to the `init`*SID*`.ora` file:

```
_ENABLE_ORACLE_RESILVERING        = TRUE
```

where *SID* is the system identifier of the Oracle database instance. This file is usually located in the directory `${ORACLE_HOME}/dbs`.

The system administrator must configure the volumes correctly to use VxSmartSync. For Volume Manager, there are two types of volumes used by the database:

- *redo log volumes* contain redo logs of the database.
- *data volumes* are all other volumes used by the database (control files and tablespace files).

VxSmartSync works with these two types of volumes differently, and they must be configured correctly to take full advantage of the extended interfaces. The only difference between the two types of volumes is that redo log volumes should have dirty region logs, while data volumes should not.

Data Volume Configuration

The improvement in recovery time for data volumes is achieved by letting the database software decide which portions of the volume require recovery. The database keeps logs of changes to the data in the database and can determine which portions of the volume require recovery. By reducing the amount of space that requires recovery and allowing the database to control the recovery process, the overall recovery time is reduced.

Also, the recovery takes place when the database software is started, not at system startup. This reduces the overall impact of recovery when the system reboots. Because the recovery is controlled by the database, the recovery time for the volume is the resilvering time for the database (that is, the time required to replay the redo logs).

Because the database keeps its own logs, it is not necessary for Volume Manager to do logging. Data volumes should therefore be configured as mirrored volumes *without* dirty region logs. In addition to improving recovery time, this avoids any run-time I/O overhead due to DRL, which improves normal database write access.

Redo Log Volume Configuration

A *redo log* is a log of changes to the database data. No logs of the changes to the redo logs are kept by the database, so the database itself cannot provide information about which sections require resilvering. Redo logs are also written sequentially, and since traditional dirty region logs are most useful with randomly-written data, they are of minimal use for reducing recovery time for redo logs. However, Volume Manager can reduce the number of dirty regions by modifying the behavior of its Dirty Region Logging feature to take advantage of sequential access patterns. This decreases the amount of data needing recovery and reduces recovery time impact on the system.

The enhanced interfaces for redo logs allow the database software to inform Volume Manager when a volume is to be used as a redo log. This allows Volume Manager to modify the DRL behavior of the volume to take advantage of the access patterns. Since the improved recovery time depends on dirty region logs, redo log volumes should be configured as mirrored volumes *with* dirty region logs.

Volume Manager Task Monitor

The Volume Manager Task Monitor tracks the progress of system recovery by monitoring task creation, maintenance, and completion. The Task Monitor allows you to monitor task progress and to modify characteristics of tasks, such as pausing and recovery rate (for example, to reduce the impact on system performance). See “Volume Manager Task Monitor Operations,” for

Task Monitor command line operations. You can also monitor and modify the progress of the Online Relayout feature. See “Online Relayout,” for more information.

Volume Manager Cluster Functionality

The Volume Manager includes an *optional* cluster feature that enables VxVM to be used in a cluster environment. The cluster functionality in the Volume Manager allows multiple hosts to simultaneously access and manage a given set of disks under Volume Manager control (*VM disks*).

A *cluster* is a set of hosts sharing a set of disks; each host is referred to as a *node* in the cluster. The nodes are connected across a network. If one node fails, the other node(s) can still access the disks. The Volume Manager cluster feature presents the same logical view of the disk configurations (including changes) on all nodes. When the cluster feature is enabled, Volume Manager objects are capable of being shared by all of the nodes in a cluster.

For more information about the cluster functionality in the Volume Manager, refer to the chapter on cluster functionality in the *VERITAS Volume Manager Administrator’s Reference Guide*.

Note: The cluster functionality in Volume Manager is a separately licensable feature.

Volume Manager Initialization and Setup

3

Introduction

This chapter briefly describes the steps needed to initialize Volume Manager and the daemons that must be running for Volume Manager to operate. This chapter also provides guidelines to help you set up a system with storage management.

Refer to the *VERITAS Volume Manager Installation Guide* for detailed information on how to install and set up the Volume Manager and the Storage Administrator.

The following topics are covered in this chapter:

- Volume Manager Initialization
  - Reboot After `vxinstall`
- Volume Manager Daemons
  - Configuration Daemon `vxconfigd`
  - Volume I/O Daemon `vxiod`
- System Setup
- Example System Setup Sequence
- System Setup Guidelines
  - Hot-Relocation Guidelines
  - Striping Guidelines
  - Mirroring Guidelines

- Dirty Region Logging (DRL) Guidelines
- Mirroring and Striping Guidelines
- Striping and Mirroring Guidelines
- RAID-5 Guidelines
- Protecting Your System

Volume Manager Initialization

You initialize the Volume Manager by using the `vxinstall` program. `vxinstall` places specified disks under Volume Manager control. By default, these disks are placed in the `rootdg` disk group. You must use `vxinstall` to initialize at least one disk into `rootdg.` You can then use `vxdiskadm` or the Storage Administrator to initialize or encapsulate additional disks into other disk groups.

Once you have completed the package installation, follow these steps to initialize the Volume Manager.

1. Log in as superuser (or appropriate access level).
2. Create a `disks.exclude` file if you want to exclude disks from Volume Manager control. `vxinstall` ignores any disks listed in this file. Place this file in: `/etc/vx/disks.exclude.`
3. Create a `cntrls.exclude` file if you want to exclude all disks on a controller from Volume Manager control. Place this file in: `/etc/vx/cntrls.exclude`.
4. Start `vxinstall` by entering this command: `vxinstall.`

`vxinstall` then does the following:

- runs and displays the license information and prompts for a key
- examines and lists all controllers attached to the system
- describes the installation process: Quick or Custom installation

Quick installation gives you the option of initializing or encapsulating all disks. To encapsulate some disks on a given controller and initialize others, use the Custom installation process.

Custom installation allows you to control which disks are added to Volume Manager control and how they are added. You can initialize or encapsulate all disks on a controller, or initialize some disks on a controller and encapsulate others.

For details on how to use the Quick or Custom installation option, refer to the *VERITAS Volume Manager Installation Guide.*

Reboot After `vxinstall`

The way you choose to configure your system determines whether or not a shutdown and reboot is required. If you choose to encapsulate any disks, a reboot is necessary. `vxinstall` informs you if a reboot is required.

You can use this command to confirm that key Volume Manager processes are running (`vxconfigd`, `vxnotify`, and `vxrelocd`):

```
ps -ef | grep vx
```

Volume Manager Daemons

Two daemons must be running for the Volume Manager to operate properly:

- `vxconfigd`
- `vxiod`

Configuration Daemon `vxconfigd`

The Volume Manager configuration daemon (`vxconfigd`) maintains Volume Manager disk and disk group configurations. `vxconfigd` communicates configuration changes to the kernel and modifies configuration information stored on disk.

Starting the Volume Manager Configuration Daemon

`vxconfigd` is invoked by startup scripts during the boot procedure.

To determine whether the volume daemon is enabled, enter this command:

```
vxdctl mode
```

This message is displayed if vxconfigd is running and enabled:

```
mode: enabled
```

This message is displayed if vxconfigd is running, but not enabled:

```
mode: disabled
```

To enable the volume daemon, enter this command:

```
vxdctl enable
```

This message is displayed if vxconfigd is not running:

```
mode: not-running
```

To start vxconfigd enter this command:

```
vxconfigd
```

Once started, vxconfigd automatically becomes a background process.

By default, vxconfigd issues errors to the console. However, vxconfigd can be configured to issue errors to a log file.

For more information on the vxconfigd daemon, refer to the vxconfigd(1M) and vxdctl(1M) manual pages.

Volume I/O Daemon vxiod

The volume extended I/O daemon (vxiod) allows for extended I/O operations without blocking calling processes.

For more detailed information about vxiod, refer to the vxiod (1M) manual page.

Starting the Volume I/O Daemon

vxiod daemons are started at system boot time. There are typically several vxiod daemons running at all times. Rebooting after your initial installation starts vxiod.

Verify that vxiod daemons are running by entering this command:

```
vxiod
```

Because `vxiod` is a kernel thread and is not visible to you through the `ps command`, this is the only way to see if any `vxiod` daemons are running.

If any `vxiod` daemons are running, the following message is displayed:

```
10 volume I/O daemons running
```

where `10` is the number of `vxiod` daemons currently running.

If no `vxiod` daemons are currently running, start some by entering this command:

```
vxiod set 10
```

where `10` can be substituted by the desired number of `vxiod` daemons. It is recommended that at least one `vxiod` daemon exist for each CPU in the system.

System Setup

This section has information to help you set up your system for efficient storage management. For specific setup tasks, see the *Command Line Interface Administrator's Guide* and the *Volume Manager Storage Administrator Administrator's Guide.*

The following system setup sequence is typical and should be used as an example. Your system requirements may differ. The system setup guidelines provide helpful information for specific setup configurations.

Example System Setup Sequence

The following list describes typical activities you can use when setting up your storage management system.

▼ Initial Setup

1. Place disks under Volume Manager control.
2. Create new disk groups (if you do not want to use `rootdg` or you want other disk groups).
3. Create volumes.
4. Put file system(s) in volumes.

▼ Options

- Encapsulate the `boot/root` disk and mirror it to create alternate boot disk.
- Designate hot-relocation spare disks.
- Add mirrors to volumes if necessary.

▼ Maintenance

- Resize volumes and file systems.
- Add more disks/disk groups.
- Create snapshots.

System Setup Guidelines

These general guidelines can help you to understand and plan an efficient storage management system. You can use the cross references in each section to get more information about the featured guideline.

Hot-Relocation Guidelines

You can follow these general guidelines when using hot-relocation. See “Hot-Relocation,” in Chapter 2, “Volume Manager Operations,” for more information.

- The hot-relocation feature is enabled by default. Although it is possible to disable hot-relocation, it is advisable to leave it enabled.
- Although hot-relocation does not require you to designate disks as spares, you can designate at least one disk as a spare within each disk group. This gives you some control over which disks are used for relocation. If no spares exist, Volume Manager uses any available free space within the disk group. When free space is used for relocation purposes, it is possible to have performance degradation after the relocation.
- After hot-relocation occurs, you can designate one or more additional disks as spares to augment the spare space (some of the original spare space may be occupied by relocated subdisks).

- If a given disk group spans multiple controllers and has more than one spare disk, you can set up the spare disks on different controllers (in case one of the controllers fails).
- For a mirrored volume, the disk group must have at least one disk that does not already contain a mirror of the volume. This disk should either be a spare disk with some available space or a regular disk with some free space.
- For a mirrored and striped volume, the disk group must have at least one disk that does not already contain one of the mirrors of the volume or another subdisk in the striped plex. This disk should either be a spare disk with some available space or a regular disk with some free space.
- For a RAID-5 volume, the disk group must have at least one disk that does not already contain the RAID-5 plex (or one of its log plexes) of the volume. This disk should either be a spare disk with some available space or a regular disk with some free space.
- If a mirrored volume has a DRL log subdisk as part of its data plex, that plex cannot be relocated. You can place log subdisks in plexes that contain no data (log plexes).
- Hot-relocation does not guarantee that it preserves the original performance characteristics or data layout. You can examine the location(s) of the newly-relocated subdisk(s) and determine whether they should be relocated to more suitable disks to regain the original performance benefits.
- Hot-relocation is capable of creating a new mirror of the root disk if the root disk is mirrored and it fails. The `rootdg` disk group should therefore contain sufficient contiguous spare or free space to accommodate the volumes on the root disk (`rootvol` and `swapvol` require contiguous disk space).
- Although it is possible to build VxVM objects on spare disks (using `vxmake` or the Storage Administrator interface), it is preferable to use spare disks for hot-relocation only.

Striping Guidelines

You can follow these general guidelines when using striping. See "Striping (RAID-0)," in Chapter 1, "Understanding Volume Manager," for more information.

- Do not place more than one column of a striped plex on the same physical disk.
- Calculate stripe unit sizes carefully. In general, a moderate stripe unit size (for example, 64 kilobytes, which is also the default used by `vxassist`) is recommended. If it is not feasible to set the stripe unit size to the track size, and you do not know the application I/O pattern, it is recommended that you use 64 kilobytes for the stripe unit size.

Note: Many modern disk drives have "variable geometry," which means that the track size differs between cylinders (i.e., outer disk tracks have more sectors than inner tracks). It is therefore not always appropriate to use the track size as the stripe unit size. For these drives, use a moderate stripe unit size (such as 64 kilobytes), unless you know the I/O pattern of the application.

- Volumes with small stripe unit sizes can exhibit poor sequential I/O latency if the disks do not have synchronized spindles. Generally, striping over non-spindle-synched disks performs better if used with larger stripe unit sizes and multi-threaded, or largely asynchronous, random I/O streams.
- Typically, the greater the number of physical disks in the stripe, the greater the improvement in I/O performance; however, this reduces the effective mean time between failures of the volume. If this is an issue, striping can be combined with mirroring to provide a high-performance volume with improved reliability.
- If only one plex of a mirrored volume is striped, be sure to set the policy of the volume to `prefer` for the striped plex. (The default read policy, `select`, does this automatically.)
- If more than one plex of a mirrored volume is striped, make sure the stripe unit size is the same for each striped plex.
- Where possible, distribute the subdisks of a striped volume across drives connected to different controllers and buses.
- Avoid the use of controllers that do not support overlapped seeks (these are rare).

The `vxassist` command automatically applies and enforces many of these rules when it allocates space for striped plexes in a volume.

Mirroring Guidelines

You can follow these general guidelines when using mirroring. See “Mirroring (RAID-1),” in Chapter 1, “Understanding Volume Manager,” for more information.

- Do not place subdisks from different plexes of a mirrored volume on the same physical disk. This action compromises the availability benefits of mirroring and degrades performance. Use of `vxassist` precludes this from happening.
- To provide optimum performance improvements through the use of mirroring, at least 70 percent of the physical I/O operations should be read operations. A higher percentage of read operations results in a higher benefit of performance. Mirroring may not provide a performance increase or result in performance decrease in a write-intensive workload environment.

Note: The UNIX operating system implements a file system cache. Read requests can frequently be satisfied from the cache. This can cause the read/write ratio for physical I/O operations through the file system to be biased toward writing (when compared to the read/write ratio at the application level).

- Where feasible, use disks attached to different controllers when mirroring or striping. Most disk controllers support overlapped seeks that allow seeks to begin on two disks at once. Do not configure two plexes of the same volume on disks attached to a controller that does not support overlapped seeks. This is important for older controllers or SCSI disks that do not cache on the drive. It is less important for many newer SCSI disks and controllers used in most modern workstations and server machines. Mirroring across controllers can be of benefit because the system can survive a controller failure. The other controller can continue to provide data from the other mirror.

- A plex can exhibit superior performance due to being striped or concatenated across multiple disks, or because it is located on a much faster device. The read policy can then be set to prefer the “faster” plex. By default, a volume with one striped plex is configured with preferred reading of the striped plex.

Dirty Region Logging (DRL) Guidelines

You can follow these general guidelines when using dirty region logging. See “Dirty Region Logging,” in Chapter 2, “Volume Manager Operations,” for more information.

Dirty Region Logging (DRL) can speed up recovery of mirrored volumes following a system crash. When DRL is enabled, Volume Manager keeps track of the regions within a volume that have changed as a result of writes to a plex. Volume Manager maintains a bitmap and stores this information in a *log subdis*k. Log subdisks are defined for and added to a volume to provide DRL. Log subdisks are independent of plexes, are ignored by plex policies, and are only used to hold the DRL information.

Note: Using Dirty Region Logging can impact system performance in a write-intensive environment.

Follow these guidelines when using DRL:

- For Dirty Region Logging to be in effect, the volume must be mirrored.
- At least one log subdisk must exist on the volume for DRL to work. However, only one log subdisk can exist per plex.
- The subdisk that is used as the log subdisk should not contain necessary data.
- It is possible to “mirror” log subdisks by having more than one log subdisk (but only one per plex) in the volume. This ensures that logging can continue, even if a disk failure causes one log subdisk to become inaccessible.
- Log subdisks must be configured with two or more sectors (preferably an even number, as the last sector in a log subdisk with an odd number of sectors is not used). The log subdisk size is normally proportional to the volume size. If a volume is less than 2 gigabytes, a log subdisk of 2 sectors

is sufficient. The log subdisk size should then increase by 2 sectors for every additional 2 gigabytes of volume size. However, `vxassist` chooses reasonable sizes by default. In general, use of the default log subdisk length provided by `vxassist` is recommended.

- The log subdisk should not be placed on a heavily-used disk, if possible.
- Persistent (non-volatile) storage disks must be used for log subdisks.

Mirroring and Striping Guidelines

You can follow these general guidelines when using mirroring and striping together. See "Mirroring Plus Striping (RAID-1 + RAID-0)," in Chapter 1, "Understanding Volume Manager," for more information.

- Make sure that there are enough disks available for the striped and mirrored configuration. At least two disks are required for the striped plex and one or more *other* disks are needed for the mirror.
- Never place subdisks from one plex on the same physical disk as subdisks from the other plex. Follow the striping guidelines described in "Striping Guidelines."
- Follow the mirroring guidelines described in "Mirroring Guidelines."

Striping and Mirroring Guidelines

You can follow these general guidelines when using striping and mirroring together. See "Striping Plus Mirroring (RAID-0 + RAID-1)," in Chapter 1, "Understanding Volume Manager," for more information.

- Make sure that there are enough disks available for the striped and mirrored configuration. At least two disks are required for the striped plex and one or more *other* disks are needed for the mirror.
- Never place subdisks from one plex on the same physical disk as subdisks from the other plex. Follow the striping guidelines described in "Striping Guidelines."
- Follow the mirroring guidelines described in "Mirroring Guidelines."

RAID-5 Guidelines

You can follow these general guidelines when using RAID-5. See “RAID-5,” in Chapter 1, “Understanding Volume Manager,” for more information.

In general, the guidelines for mirroring and striping together also apply to RAID-5. The following guidelines should also be observed with RAID-5:

- Only one RAID-5 plex can exist per RAID-5 volume (but there can be multiple log plexes).
- The RAID-5 plex must be derived from at least two subdisks on two or more physical disks. If any log plexes exist, they must belong to disks other than those used for the RAID-5 plex.
- RAID-5 logs can be mirrored and striped.
- If the volume length is not explicitly specified, it is set to the length of any RAID-5 plex associated with the volume; otherwise, it is set to zero. If the volume length is set explicitly, it must be a multiple of the stripe unit size of the associated RAID-5 plex, if any.
- If the log length is not explicitly specified, it is set to the length of the smallest RAID-5 log plex that is associated, if any. If no RAID-5 log plexes are associated, it is set to zero.
- Sparse RAID-5 log plexes are not valid.

Protecting Your System

Disk failures can cause two types of problems: loss of data on the failed disk and loss of access to your system. Loss of access can be due to the failure of a key disk (a disk used for system operations). The VERITAS Volume Manager can protect your system from these problems.

To maintain system availability, data important to running and booting your system must be mirrored. The data must be preserved so it can be used in case of failure.

Here are suggestions on how to protect your system and data:

- Place the disk containing the root file system (the *root* or *boot* disk) under Volume Manager control through encapsulation. This converts the `root` and `swap` devices to volumes (`rootvol` and `swapvol`). Then mirror the root disk so that an alternate root disk exists for booting purposes. By mirroring disks critical to booting, you ensure that no single disk failure leaves your system unbootable and unusable.

For maximum availability of the system, create mirrors for the `rootvol`, `swapvol`, `usr`, and `var` volumes. See “Possible Root (/), swap, and usr Configurations,” in the “Administrator’s Reference Guide,” for more information.

- Use mirroring to protect data. By mirroring your data, you prevent data loss from a disk failure. To preserve data, create and use mirrored volumes that have at least two data plexes. The plexes must be on different disks. If a disk failure causes a plex to fail, the data in the mirrored volume still exists on the other disk.

  If you use `vxassist mirror` to create mirrors, it locates the mirrors so the loss of one disk does not result in a loss of data. By default, `vxassist` does not create mirrored volumes; you can edit the file `/etc/default/vxassist` to set the default layout to mirrored. Refer to Chapter 4, in the “Command Line Interface Administrator’s Guide”, for information on the `vxassist` defaults file.

- Leave the Volume Manager hot-relocation feature enabled to automatically detect failures, notify you of the nature of the failures, attempt to relocate any affected subdisks that are redundant, and initiate recovery procedures. Provide at least one hot-relocation spare disk per disk group so sufficient space is available for relocation in the event of a failure.

  If the `root` disk is mirrored, hot-relocation can automatically create another mirror of the `root` disk if the original `root` disk fails. The `rootdg` disk group should contain enough contiguous spare or free space for the volumes on the root disk (`rootvol` and `swapvol` volumes require contiguous disk space).

- For mirrored volumes, take advantage of the Dirty Region Logging feature to speed up recovery of your mirrored volumes after a system crash. Make sure that each mirrored volume has at least one log subdisk. (Note that `rootvol`, `swapvol`, and `usr` volumes cannot be Dirty Region Logging volumes.)

- For RAID-5 volumes, take advantage of logging to prevent corruption of recovery data. Make sure that each RAID-5 volume has at least one log plex.

- Perform regular backups to protect your data. Backups are necessary if all copies of a volume are lost or corrupted in some way. For example, a power surge could damage several (or all) disks on your system. Also, typing a command in error can remove critical files or damage a file system directly.

Index