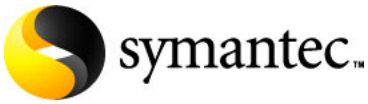

Veritas Storage Foundation™ 5.0
Dynamic Multi-pathing

Optimizing Availability and Performance in Multi-Vendor Environments

May 2007

Table of Contents

Table of Figures

Scope

This paper describes the Dynamic Multi-pathing (DMP) feature of Veritas Storage Foundation™. The product architecture described herein was introduced with DMP release 5.0. It was subsequently back-ported to the Solaris 4.1 code base in *SxRT 4.1 MP2*, the AIX 4.0 code base in *AxRT 4.0 MP4* and the Linux 4.1 code base in *LxRT 4.1 MP4*. These three releases and up are collectively referred to as *DMP Backport* releases throughout this document.

The paper should be used as a guide to understanding Dynamic Multi-pathing. For up-to-date information on features and coverage, readers are advised to consult Symantec documentation and support sources.

1 The Importance of Multiple Storage I/O Paths

The basic techniques for keeping business-critical computer applications and digital data available to users despite hardware and software failures are well-known:

- ***Applications***. Applications can be protected against server failures by interconnecting two or more servers to form a cooperative *cluster* controlled by software that enables an application running on any of the servers to *fail over* and restart on another, should its own server fail.
- ***Data***. Data can be preserved despite storage device failures by techniques such as *mirroring* identical copies on two or more disks[1] and writing all updates to both simultaneously. Mirroring, sometimes called *RAID-1*, keeps data available if a disk fails, and also improves I/O performance by making two or more disks available to satisfy each application read request.

In enterprise data centers, there is another increasingly important link in the information access chain—the *I/O path* that connects servers with the data they process. The I/O path, represented in Figure 1, is a complex chain consisting of host bus adapter, cables, storage network switch, storage device adapter port, and, in disk arrays, a disk controller.

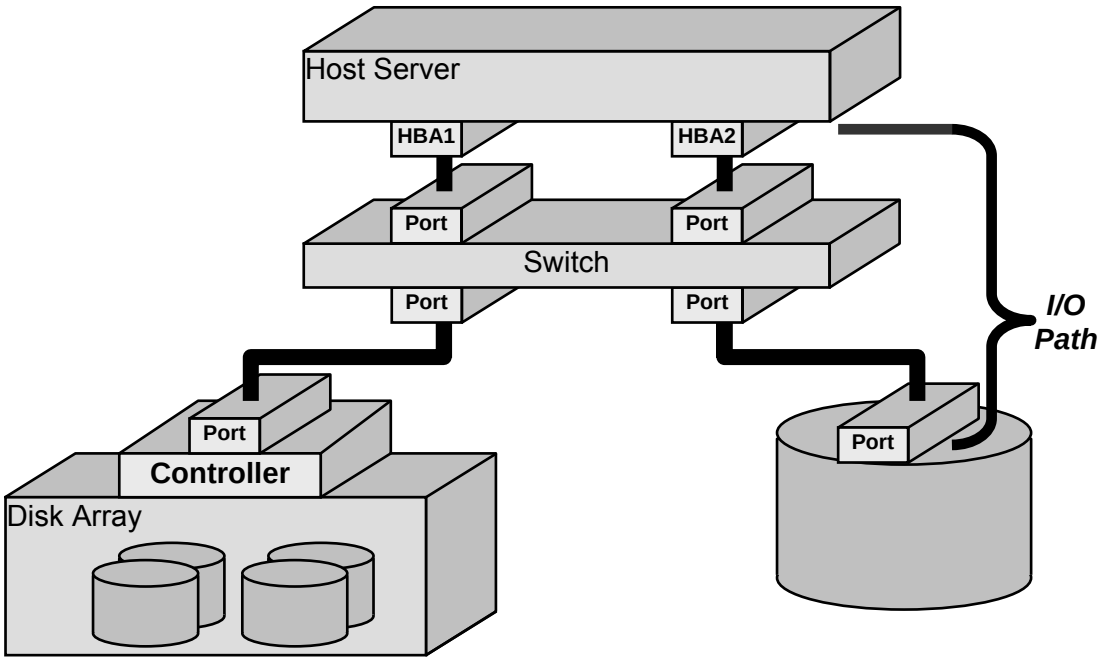

Figure 1: General I/O Path Model

[1] In this paper, the term *disk* refers both to actual disk drives and to the logical units (LUNs) presented to storage network ports by disk arrays.

The I/O path shown in Figure 1 begins at a host bus adapter (HBA)[2] that connects an I/O cable to a server's internal memory access bus. The cable connects the HBA to a corresponding port in a storage network switch. As Figure 1 suggests, the switch manages logical connections between HBAs and ports within disk array controllers, or between HBAs and disk drives. Disk array controllers, which typically have more than one port, virtualize disks within the array and present them to the storage network as logical units, or LUNs. [3]

> **Usage Note**
>
> Each unique combination of these elements that can be used to communicate between a host server and a LUN within a disk array or a disk connected directly to the network is a distinct *I/O path*.

1.1 Multiple I/O Paths Enhance Availability

With increasing deployment of storage networks, IT managers are becoming conscious of the important role that I/O paths play in keeping data available. For example, two disks mirrored by a host-based volume manager may be connected to their hosting server either by the same I/O path, as shown on the left side of Figure 2, or by different paths, as shown on the right. If multiple paths are available, mirroring not only protects against data loss due to disk failure, it also protects against loss of *access* to data if an I/O path element fails, as Figure 2 illustrates.

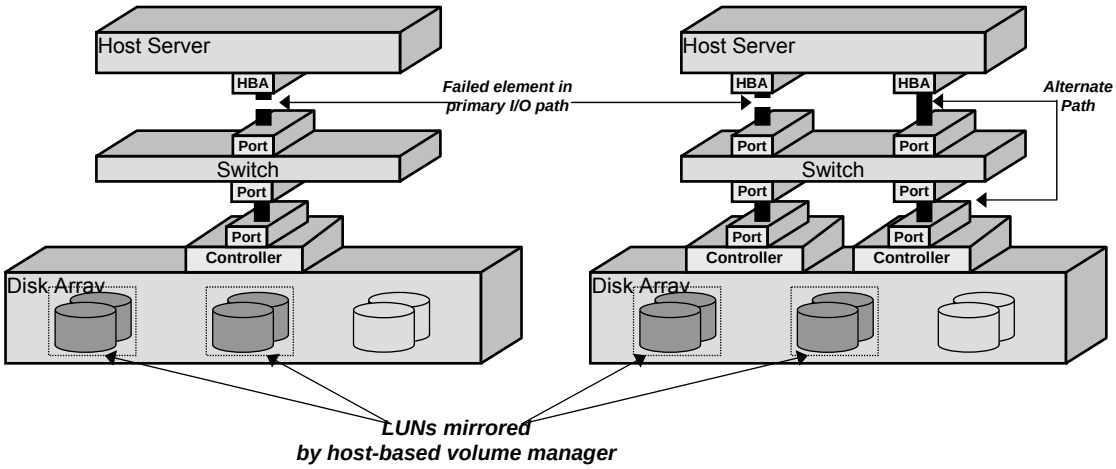

Figure 2: Multiple I/O Paths Improve Data Availability

---

2 Some HBAs have multiple ports, each of which is the starting point of a separate path through the storage network. Since each port is effectively a separate HBA, the model is simplified by treating an HBA as a port.

3 In addition to disk array virtualization, both disks and LUNs are sometimes virtualized by *appliances* or switches within the storage network, and by host-based volume managers such as VxVM. The virtual devices that result from disk array virtualization are universally referred to as LUNs. Virtual devices that result from switch and host-based virtualization are called *virtual disks* or *volumes*.

The server on the left in Figure 2 cannot access its data when the cable between its HBA and the network switch port fails, even though the storage itself remains completely functional, because the cable is a *single point of failure*. The server on the right, on the other hand, can continue to access data if one of its HBAs fails, if a cable fails, or even if one of the disk array's controllers fails, because in each case there is an alternate path that does not include the failed element.

Thus, a second independent path between server and storage increases the number of component failures an I/O subsystem can withstand without loss of function. But even with an alternate path, I/O path failure can still be tantamount to storage device failure unless the system recognizes that it *has* an alternate path and reroutes I/O requests to it. If a server does not recognize an alternate path to a storage device, the device may as well have failed. Even with failure-tolerant mirrored devices, for example, only devices on still-functioning paths are updated after a path failure. Data redundancy is diminished, even though the unreachable device is still functional. Moreover, I/O performance decreases because one less device is available to satisfy read requests.

Thus, an ability to recognize and utilize alternate I/O paths to storage devices would clearly be preferable. If a path failed, I/O requests would be re-routed to the alternate. Mirrored data would remain fully protected, and the effect on I/O performance would be smaller.

1.2 Multiple I/O Paths Enhance I/O Performance

Multiple I/O paths between server and storage device can also improve I/O performance. In many applications, disk arrays satisfy a significant percentage of I/O requests from cache. For example, most disk arrays recognize sequential read patterns, and begin to read data into cache in advance of host I/O requests. In this scenario, I/O path bandwidth can actually limit LUN performance. With multiple I/O paths to a LUN however, all can be delivered concurrently as fast as applications request it. Similarly, if an I/O path that provides access to multiple LUNs becomes momentarily overloaded due to activity on one LUN, other LUNs' I/O requests can be routed to less-busy paths.

2 Different Forms of Multi-Path Access

Disks and disk arrays support multi-path access to LUNs in several different ways. Fundamentally, there is a distinction between:

- ***Active-active (A/A)***. If a disk array accepts and executes I/O requests to a single LUN on any port simultaneously, it is called an *active-active* (A/A) array. If a path to an active-active array fails, I/O requests can simply be re-routed to other paths, maintaining continuous access to data stored on the array's LUNs.

  EMC's Symmetrix and DMX arrays, Hitachi Data Systems' 9900 Series (Lightning), and IBM's ESS series (Shark) are active-active arrays.

- ***Active-passive (A/P)***. If a disk array accepts and executes I/O requests to a LUN on one or more ports to one array controller (the *primary*), but is able to switch, or "fail over," access to the LUN to alternate ports on another array controller (the *secondary)*, it is called *active-passive* (A/P). A simple A/P disk array triggers failover for a LUN based on where I/O for that LUN is received. Since a LUN failover (also called *trespass*) is a slow operation that impacts performance, all I/Os should only be flowing to only one of the available controllers for a LUN at a given point in time. Efficiently managing LUN *trespass* on an A/P array is critical to provide high performance data access.

  EMC's Clariion Cx600 and Cx700, Hitachi Data Systems' 95xx and 9200 series, IBM FASt-T, and Sun's T3 and T4 are active-passive arrays.

In addition to this broad classification, active-passive disk arrays capabilities differ in other ways that affect availability and I/O performance:

- ***Multiple primary & secondary paths (A/PC)***. If an active-passive array accepts and executes simultaneous I/O requests to a LUN on two or more ports of the same array controller, it is called an *active-passive concurrent* (A/P-C) array. Active-passive concurrent array LUNs fail over to secondary paths on alternate array controllers only when all primary paths have failed. Note that DMP's ability to do load balancing over multiple primary or secondary paths to a LUN in an active passive array is fully governed by the I/O policy configured for that enclosure (see Section 4.6.1 for details).

- ***Explicit failover (A/PF).*** A basic active-passive array fails over from primary I/O paths to secondary ones automatically when it receives an I/O request to a LUN on a secondary path. An *Explicit failover active passive* (A/PF) array fails over only when it receives special array model-specific SCSI commands from their hosts. Explicit failover provides the control required to achieve high performance with active-passive arrays in clusters, where multiple hosts can issue I/O requests directly to LUNs. Without explicit failover capability, cluster software must carefully synchronize all hosts' access to a LUN before initiating implicit failover so that I/O requests from multiple hosts do not result in continuous failovers.

  Note: It is generally recommended to configure arrays capable of A/PF as such in cluster configurations to ensure optimum performance and minimize system boot times. A good example is the EMC Clariion which should be set to Failovermode 1 (explicit)' when used with DMP.
  EMC's CLARiiON and Sun Microsystems T3 and T4 arrays are A/PF arrays.

- ***LUN group failover (A/PG).*** In general, LUNs fail over from one array controller to another individually. Some active-passive arrays, however, can fail administratively defined *groups* of LUNs over together. Arrays with this capability are called *active-passive with group failover capability* (A/PG). If all primary paths to a LUN in an A/PG array fail, all the LUNs in its group fail over to secondary paths. LUN group failover is faster than failover of individual LUNs, and can therefore reduce the application impact of array controller failure, particularly in disk arrays that present large numbers of LUNs.
  Hitachi Data Systems 9200 series arrays and Fujitsu ETERNUS 3000 are A/PG arrays.
- ***Active-active asymmetric (A/A-A).*** A/A-A arrays increasingly comply with the Asymmetric Logical Unit Access (ALUA) method specified in SCSI-3 standards. While a LUN in an *active-passive* array can only be accessed through the controller that owns it at a given point in time (accessing that LUN through the other controller will either result in a LUN trespass or an I/O failure), a LUN in an active-active asymmetric array can be accessed through both controllers without dramatic consequences. The only limitation is that I/O serviced through a LUN's secondary controller will experience slower performance than I/O serviced through the primary controller. One can think of ALUA arrays as a more forgiving version of *active-passive* arrays using standard SCSI-3 commands to control LUN/array controller ownership.
  HP EVA and Hitachi Data Systems TagmaStore AMS/WMS series are examples of A/A-A arrays.

As discussed in later sections, the *Dynamic Multi-pathing* (DMP) feature of Storage Foundation has a modular architecture that allows it to support new and different types of multi-path access control quickly and easily.

2.1 Discovering Multiple I/O Paths

UNIX operating systems "discover" the storage devices that are accessible to them automatically when they start up. Operating system device discovery consists of:

- ***Scanning*** I/O buses or querying storage network fabrics to determine which bus or network addresses connect to actual disks or LUNs
- ***Creating*** in-memory data structures in the operating system device tree that identify and describe discovered devices
- ***Loading*** any specialized drivers required to utilize the devices

At the end of device discovery, an operating system has an in-memory database, or *device tree*, that represents the storage devices with which it can communicate, and has loaded the drivers required to control them.

To an operating system, a storage device is an address on a network that responds appropriately to SCSI storage device commands. UNIX operating systems are not inherently multi-path aware. They view a storage device accessible on two or more paths as two devices at different network addresses. Path management software, such as Storage Foundation's Dynamic Multi-pathing , is required to analyze the device tree and identify multi-path devices. DMP's discovery process and the modifications it makes to the operating system device tree are described in Section 4.3.

3 Common Multi-Path Hardware Configurations

The hardware elements that comprise I/O paths can be configured in a variety of ways that affect both system resiliency and I/O performance. This section describes the most commonly encountered multi-path hardware configurations.

3.1 Directly Connected Disk Arrays

Although it is not often encountered in practice, the simplest multi-path hardware configuration consists of a disk or disk array that can present LUNs on two or more ports, each of which is connected directly to a host bus adapter (HBA) on a hosting server. Figure 3 illustrates this configuration.

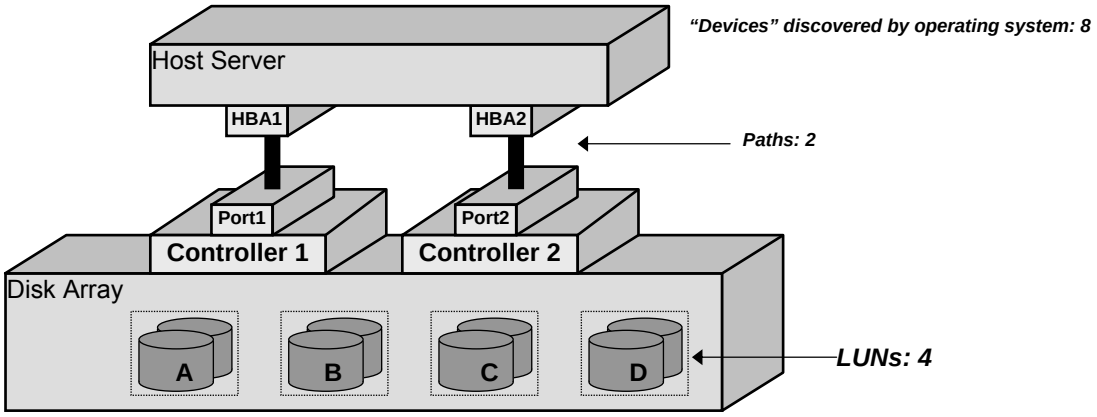

Figure 3: Directly Attached LUNs[4]

The array illustrated in Figure 3 contains four LUNs, each of which is accessible on both of its controller ports. UNIX operating systems would discover the same four LUNs on both paths, so an operating system device tree would contain a total of eight device entries (two for each LUN).

This array might be active-active (able to present LUNs on both ports simultaneously), or active-passive (able to present a LUN on either port, but not on both). If it were active-passive, the array might or might not be capable of explicit failover and LUN group failover. With only one port per controller however, the array could not provide active-passive concurrent LUN access.

[4] For simplicity, Figure 3 and the figures that follow show artificially small numbers of LUNs.

3.2 One Storage Network Switch

A more common multi-path configuration, especially in large data centers, uses a storage network to connect host computers and disk arrays. Figure 4 illustrates this configuration. Assuming open zoning, each HBA can connect through the switch to each of the disk array's controller ports. There are therefore four unique paths between server and disk array:

HBA1↔Port1↔Port3↔Port5
HBA1↔Port1↔Port4↔Port6
HBA2↔Port2↔Port3↔Port5
HBA2↔Port2↔Port4↔Port6

In this configuration, operating system discovery would report a total of 16 devices (four LUNs on each of the four paths).

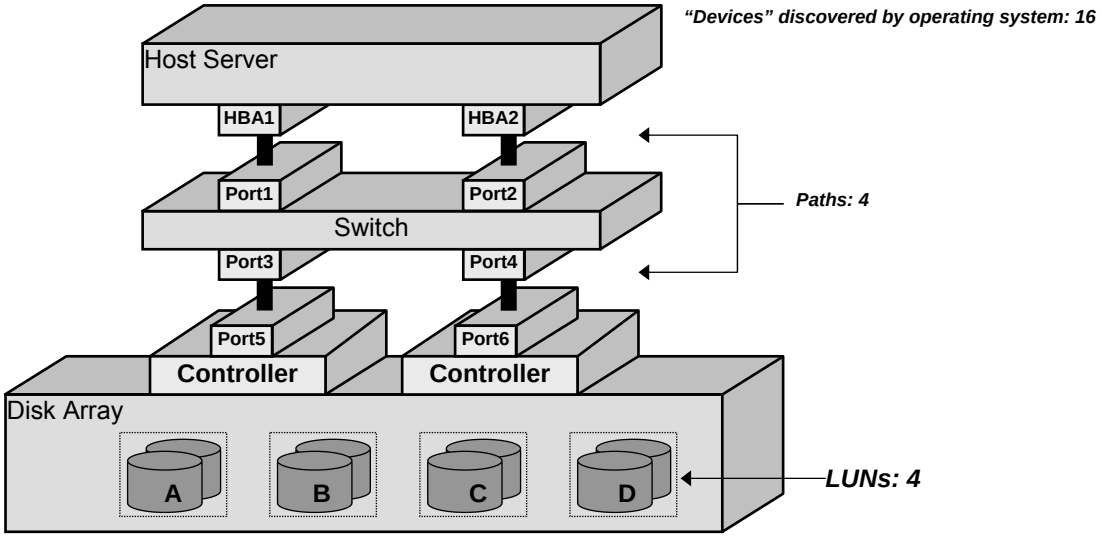

Figure 4: I/O Paths Through a Non-Redundant Storage Network

As with the configuration in Figure 3, the array illustrated in Figure 4 might be active-active or active-passive, with or without explicit and LUN group failover capability. Again, with only one port per controller, active-passive concurrent operation would not be possible. Even if this array were active-passive, concurrent execution of I/O requests to a LUN from both HBAs might be possible, although both would access the same controller port, e.g.:

HBA1↔Port1↔Port3↔Port5
HBA2↔Port2↔Port3↔Port5

This might be slightly advantageous from an availability point of view, since it eliminates failover time for failures of path elements on the host side of the switch, but there is no performance benefit, because access to any given LUN is limited by the performance of the single controller port on which it is presented.

3.3 Redundant Storage Networks

A common (and good) storage network design practice, illustrated in Figure 5, is the configuration of identical parallel fabrics connected to the same storage devices and servers, but not to each other. With this configuration, even a complete storage network outage (e.g., a total switch or director failure) leaves all host servers still able to communicate with all storage devices.

In the configuration illustrated in Figure 5, each HBA is connected to a different *fabric* (represented in the figure by a single switch for simplicity). Similarly, each disk array port is connected to a different fabric, creating two paths between any LUN and the host computer:

HBA1↔Port1↔Port3↔Port5
HBA2↔Port2↔Port4↔Port6

Operating system discovery would report a total of eight devices (four on each of the two paths).

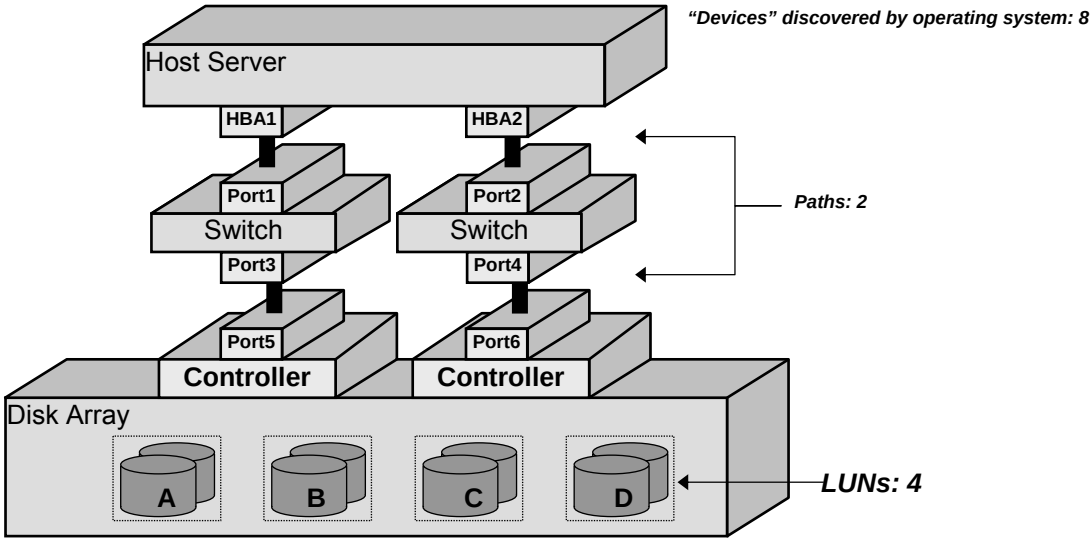

Figure 5: Multiple I/O Paths in a Storage Network with Redundant Fabrics

As in the preceding configurations, this array might be capable of either active-active or active-passive operation. If active-passive, it might be capable of explicit and LUN group failover. But with only one port per controller, active-passive concurrent LUN access would not be possible.

3.4 Dual Port Array Controllers & Redundant Storage Network Fabrics

The configuration illustrated in Figure 6 is identical to that of Figure 5 in that it includes two parallel fabrics. It differs, however in that each disk array controller has two ports and is connected to both fabrics. Assuming that each LUN can be accessed on any of the four ports (again, for simplicity, Figure 6 illustrates only the LUN B and C port connections.) and open zoning in the fabric, the operating system discovery would report each LUN on four paths:

HBA1↔Port1↔Port3↔Port5
HBA1↔Port1↔Port7↔Port6
HBA2↔Port2↔Port4↔Port9
HBA2↔Port2↔Port8↔Port0

The disk array in Figure 6 might be active-active or active-passive, and if active-passive, might be capable of explicit failover and LUN group failover. Because each array controller has two ports, active-passive concurrent operation is possible. LUNs might be presented on primary ports 5 and 9, for example, with ports 6 and 0 designated as secondary ports. EMC Clariion Cx700 arrays can be configured in this fashion. This configuration offers enhanced availability since a controller failure would still leave both fabrics usable. Similarly, failure of a fabric would leave the disk array able to use both of its controllers.

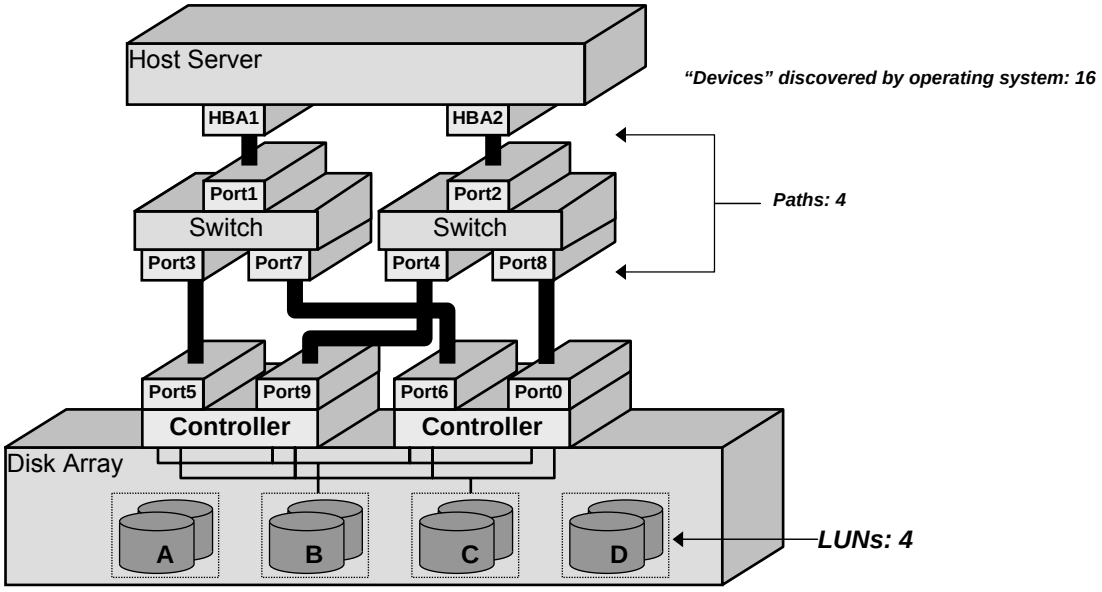

Figure 6: Multi-Port Controllers Cross-Connected to Redundant Fabrics

4 Veritas Storage Foundation 5.0 Dynamic Multi-pathing (DMP)

DMP is a leading multi-pathing solution on the market for maximum availability and performance in large heterogeneous SANs. As such, DMP allows users to confidently standardize on Storage Foundation and take full advantage of the agility that results from no longer being locked in to a single storage hardware vendor.

From a technical perspective, with Storage Foundation 5.0, DMP improves on the architecture of previous versions in order to scale to the largest data centers. It incorporates a number of innovations to ensure that it can not only work *harder* but that it also works *smarter*.

DMP provides the following benefits:

- ***Storage Connectivity Virtualization.*** By fully virtualizing connectivity from the host to storage, DMP increases data center agility. A storage administrator benefits by being able to choose the type of storage hardware that best suits his/her needs, knowing that the multi-pathing driver on the hosts either already support that storage hardware or can easily be enhanced to support it.
- ***Data availability***. If an I/O path to a multi-path storage subsystem fails, DMP automatically re-routes I/O requests to an alternate path transparently to applications and without administrator intervention. When a failed path returns to service, DMP restores the original path configuration automatically and transparently as well.
- ***Optimized I/O performance***. For storage subsystems that support simultaneous access to a single storage device on multiple paths, DMP enhances I/O performance by distributing I/O requests across all available paths according to pre-defined load balancing policies.
- ***Cluster configuration support***. DMP provides full, integrated support of SCSI-3 Persistent Reservations and I/O fencing. These features are required for Storage Foundation Oracle RAC and are strongly recommended for protecting against the risk of data corruption in cluster configurations.
- ***Reduced complexity and increased efficiency.*** By being an integral part of Veritas Storage Foundation, all DMP paths in a data center can be centrally managed and monitored from Veritas Storage Foundation Manager (SFM).

4.1 DMP and the UNIX Storage I/O Software Stack

DMP is a layer in the UNIX storage I/O software stack. While different platform implementations differ in detail, UNIX I/O software stacks share a common overall structure, simply because all perform the same basic functions to provide I/O services. Figure 7 shows a simplified model of a generic UNIX storage I/O software stack that includes VxVM and DMP.

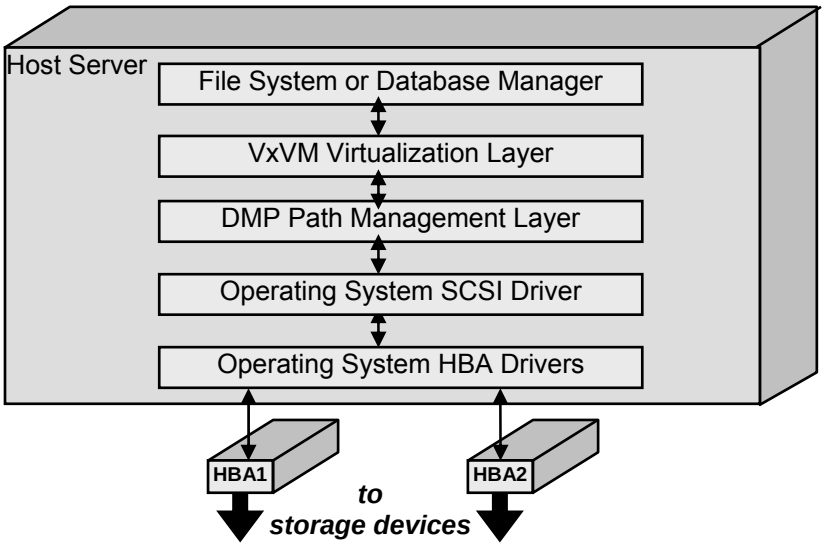

Figure 7: Generic Model of the UNIX Storage I/O Software Stack

In a typical server, almost all I/O requests to a server's I/O subsystem are issued by a file system (in some cases, database managers issue I/O requests to "raw" storage). File systems issue their I/O requests to VxVM virtual volumes (e.g., **`/dev/vx/rdsk/diskgroup/volume`**). The VxVM virtualization layer converts them into equivalent requests to physical disks or LUNs. For example, if a file system issues a write request to a mirrored volume, the VxVM virtualization layer converts it into write requests to corresponding block ranges of each of the mirrors that comprise the volume.

Path management software like DMP necessarily occupies a position below virtualization in the I/O stack. It receives I/O requests from the VxVM virtualization layer, determines which path should carry each one, and issues it to the operating SCSI system driver on that path.

UNIX operating systems have two layers of storage I/O drivers—a SCSI layer that converts operating system I/O request structures into SCSI command data blocks (CDBs) and one that sends and receives messages containing CDBs and data on the storage network or I/O bus.

Sitting above the operating system SCSI layer provides a clean consistent interface to storage devices across multiple UNIX operating systems. This is turn ensures consistency of DMP features and functionality across all platforms on which it is supported. It also allows DMP to benefit from the design and performance improvements that are incorporated in the OS SCSI layers from one UNIX release to the next.

4.2 DMP Multi-Path Devices in the Operating System Device Tree

For each disk or LUN it detects, a UNIX operating system creates data structures sometimes called *nodes* or *device handles*, in its device tree. For example, the Solaris operating system creates nodes in both the **/dev/rdsk** and **/dev/dsk** paths for each device it detects. If a device is accessible on two or more paths, operating systems treat each path as a separate device, and create nodes corresponding to each path.

During its discovery process, VxVM's **vxconfigd** daemon creates similar structures called *metanodes* in the **/dev/vx/rdmp** and **/dev/vx/dmp** trees for each storage device it detects. Each metanode represents a *metadevice,* a VxVM abstraction that corresponds to a disk or LUN and all the I/O paths on which it can be accessed. The VxVM virtualization layer issues its I/O requests to these metadevices.

The **vxconfigd** daemon identifies multiple paths to a device by issuing a SCSI inquiry command to each operating system device. A disk or LUN responds to a SCSI inquiry command with information about itself, including vendor and product identifiers and a unique serial number. An administrator can use the command **/etc/vx/diag.d/vxdmpinq** to issue a SCSI inquiry to a device and display the response, as Dialog 1 illustrates.

```
# /etc/vx/diag.d/vxdmpinq /dev/vx/rdmp/HDS9970V0_4s2
Inquiry for /dev/vx/rdmp/HDS9970V0_4s2, evpd 0x0, page code 0x0, flags 0x4
        Vendor id                       : HITACHI
        Product id                      : OPEN-9      -SUN
        Revision                        : 2106
        Serial Number                   : 045175F30009
```

Dialog 1: Information Returned by SCSI Inquiry Command

If two operating system devices respond to SCSI inquiry commands with the same serial number, they are assumed to be the same physical disk or LUN responding on two different paths.[5]

If VxVM discovery encounters only one instance of a particular serial number, the device can only be accessed on a single path. DMP links its metanode for each single-path device to the corresponding node in the operating system tree, as Figure 8 illustrates, and marks the device for "fast path" access by the VxVM virtualization layer. During system operation, the VxVM virtualization layer sends I/O requests to fast-path devices directly to the operating system's SCSI driver without passing them to DMP. This optimizes usage of system resources.

[5] A consequence of this method of detecting multiple paths to a device is that DMP can only support disks and LUNs that return the same unique disk identifier in response to SCSI inquiry commands on all paths. This is generally true for all path management software.

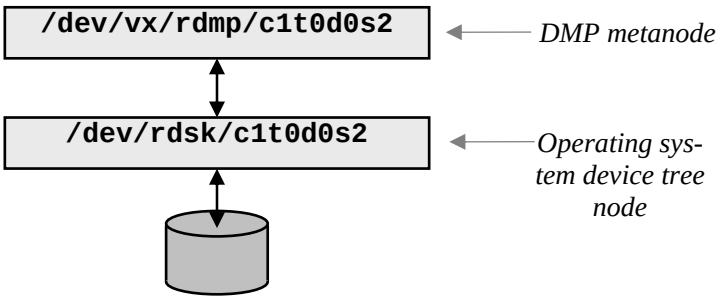

Figure 8: VxVM Subtree for a Single-Path Device (Solaris)

A device that is accessible on multiple paths returns the same serial number to inquiry commands on all paths. When DMP encounters the same serial number on different paths, it creates a metanode and links it to all operating system nodes that represent paths to the device, as Figure 9 illustrates.

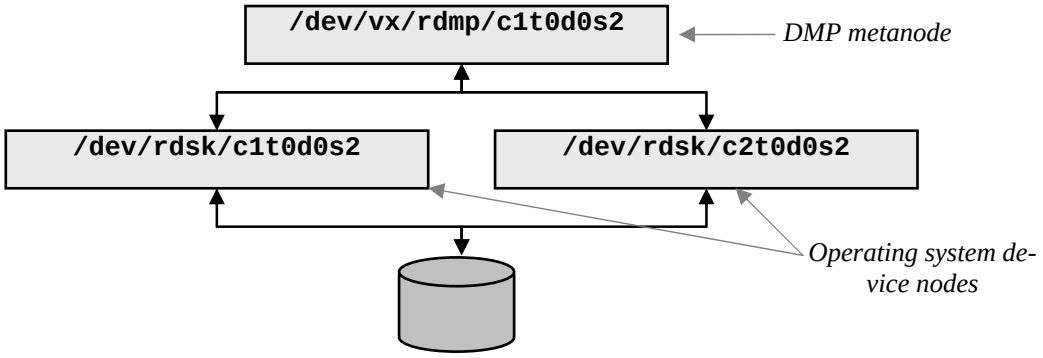

Figure 9: VxVM Subtree for a Dual-Path Device (Solaris)

An administrator can use either the **`vxdmpadm`** or the **`vxdisk path`** command to display information about VxVM metadevices and the paths to which they correspond. Dialog 2 illustrates the use of the **`vxdisk path`** command.

```
# vxdisk path
SUBPATH     DANAME      DMNAME      GROUP  STATE
c1t0d0s2    c1t0d0s2    mydg01      mydg   ENABLED
c2t0d0s2    c1t0d0s2    mydg01      mydg   ENABLED
c1t1d0s2    c1t1d0s2    mydg02      mydg   ENABLED
c2t1d0s2    c1t1d0s2    mydg02      mydg   ENABLED
```

*Dialog 2: **`vxdisk path`** Command for Multi-Path Disks*

For the dual-path device illustrated in Figure 9, the **`vxdisk path`** command in Dialog 2 shows the metanode **`c1t0d0s2`** (in the **`/dev/vx/rdmp`** subtree) as corresponding to operating system nodes **`c1t0d0s2`** and **`c2t0d0s2`**. Information displayed by the VxVM **`vxdisk path`** command includes:

- The *disk access name* (**DANAME**, or VxVM metanode name, e.g., **`c1t0d0s2`** in Dialog 2) of each metadevice. The **DANAME** is the name used by the operating system to manage the LUN.
- The *disk media name* (**DMNAME**, or VxVM user-friendly device name, e.g., **`mydg01`** in Dialog 2), of each metadevice. The **DMNAME** is used in VxVM management operations.

- The operating system device nodes (**SUBPATH**s) corresponding to each metadevice (e.g., **c1t0d0s2** and **c2t0d0s2** corresponding to VxVM metanode **c1t0d0s2** in Dialog 2)
- The VxVM disk group membership of metadevices (**mydg** in Dialog 2)
- The operational state of each metadevice on all access paths. Dialog 2 indicates that all paths are **ENABLED,** or eligible to handle I/O requests. Paths may also be **DISABLED,** either by administrative command, or by DMP itself it fails to recover from an I/O error.

4.3 DMP Device Discovery during System Operation

If a system's storage device configuration changes, for example because a device fails, or because additional disks or arrays are added, these must be discovered as well. Rebooting an operating system after a storage configuration change causes discovery, but rebooting is almost never desirable, especially for enterprise-class systems. UNIX operating systems therefore provide commands that an administrator can invoke to discover storage devices on demand. Table 1 lists the device discovery commands in each of the UNIX operating systems supported by DMP.

Operating System	Storage Device Discovery Commands
Solaris	**devfsadm** command performs subsystem scan, updates the device tree and loads drivers as necessary
AIX	**cfgmgr** command performs subsystem scan, updates the device tree and loads drivers as necessary
HPUX	Administrators should use the **ioscan** command to survey the old configuration, followed by the **insf -e** command to update the device tree and load drivers as necessary.
Linux	**makedev** command can be used to update the device tree, but I/O subsystem scan and driver loading are only done at boot time.

Table 1: UNIX Operating System Commands for Run-Time Storage Device Discovery

Whenever an operating system rediscovers its storage configuration, VxVM must also discover any effects of the change on virtualization and multi-path access. Administrators can use one of two VxVM commands to cause rediscovery by the **vxconfigd** daemon:

vxdctl enable. This command causes **vxconfigd** to scan all storage devices and reconstruct DMP metanodes and other structures to reflect the current device configuration.

vxdisk scandisks. This command may specify complete discovery, or it may be constrained to scan only newly added devices, or designated enclosures, array controllers or device address ranges. A limited scan can be considerably faster than a full one if a small number of changes have been made to a large storage configuration.

Both commands use the **vxconfigd** daemon to re-scan the storage configuration and update in-memory data structures to reflect changes since the previous scan. VxVM on-demand discovery does not interrupt system or application operation.

4.4 DMP's Modular Architecture for Advanced Storage Subsystems Support

Because its value is greatest in large enterprise data centers, DMP is more often used with disk array LUNs than with directly attached disks. Although disks and disk arrays adhere to standards for data transfer (SCSI, Fibre Channel and iSCSI), each disk array model has its own unique way of controlling multi-path LUN and disk access. To support a particular disk array model, DMP can be customized to handle the array's multi-path access capabilities and to interact properly with its interface protocols. The need to support new disk array models as they reach the market rather then on VxVM release cycles prompted the introduction of a then unique modular architecture in Version 3.2 of VxVM. This architecture has been enhanced with every subsequent release of DMP and remains a key attribute.

DMP is able to provide basic multi-pathing and failover functionality to most disk arrays without any customization by treating that disk array's LUNs as disks, provided that the array has the following properties:

- Multi-path access to LUNs is active-active
- LUNs respond to SCSI inquiry commands with unique serial numbers, and each LUN's serial number is reported identically on all paths
- LUNs' unique serial numbers can be read from the SCSI standard mode page location

If an array has these properties, the **`vxddladm`** command with the **`addjbod`** option can be used to add its LUNs (identified by the vendor ID and product ID reported in response to SCSI inquiry commands) to DMP's list of JBOD (physical disk) devices.

For fully optimized support of any array and for support of more complicated array types (as described in Section 8), DMP requires the use of array-specific *array support libraries* (ASLs), possibly coupled with an *array policy modules* (APMs). ASL and APMs effectively are *array specific plugins* that allow close tie-in of DMP with any specific array model.

An ASL contains the set of instructions that allows DMP to properly claim devices during device discovery, allowing DMP to correlate paths to the same device, gather device attributes, identify the array the device is located in and identify the set of commands that DMP must use to efficiently manage multiple paths to that device.

An APM contains the set of commands that DMP uses to efficiently manage multiple paths to a device. The base DMP packages come with a default set of generic APMs to manage *active-active* arrays, basic *active-passive* arrays and *active-active asymmetric* arrays. But the ASL/APM framework allows the creation of an array model specific APM to fully customize DMP for that array model if and when that is needed.

4.4.1 Array Support Libraries

Figure 10 illustrates how ASLs and APMs fit into VxVM's configuration facilities and I/O path, with emphasis on the relationship to the **`vxconfigd`** configuration daemon.

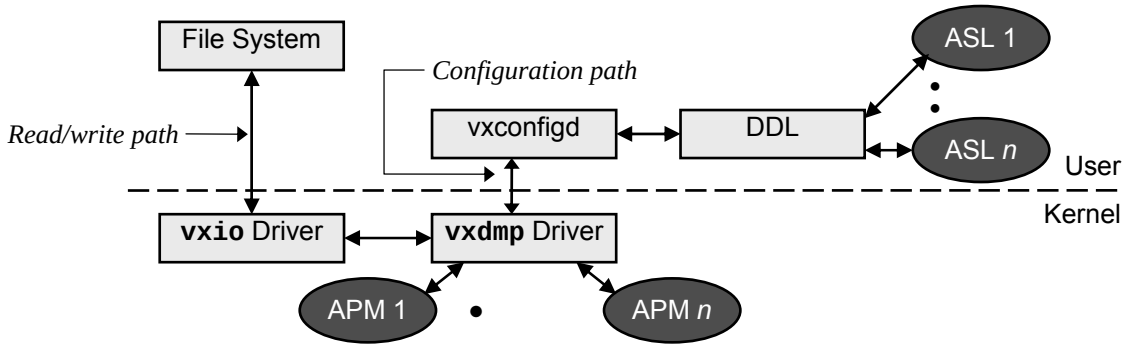

Figure 10: The DMP Device Discovery Layer (DDL) Architecture

After operating system device discovery, VxVM's **`vxconfigd`** daemon executes its own discovery process to elicit the information it requires to operate, and builds its own device tree of nodes similar to those illustrated in Section 17. For each device in its tree, VxVM's *Device Discovery Layer (DDL)* calls each installed ASL in turn until an ASL "claims" the device based on its vendor and product identifiers. The claim associates an *array model* with the device, which in turn determines the APM that **`vxdmp`** invokes to perform such functions as I/O path selection, path failover, and SCSI reservation and release.

All ASLs that ship with VxVM are installed by default during VxVM installation. Dialog 3 lists the ASLs installed on a typical Solaris system, and the types of storage devices they support.

```
# vxddladm listsupport
LIBNAME              VID
==============================================================================
libvxCLARiiON.so     DGC
libvxcscovrts.so     CSCOVRTS
libvxemc.so          EMC
libvxengenio.so      SUN
libvxhds9980.so      HITACHI
libvxhdsalua.so      HITACHI
libvxhdsusp.so       HITACHI
libvxhpalua.so       HP, COMPAQ
libvxibmds4k.so      IBM
libvxibmds6k.so      IBM
libvxibmds8k.so      IBM
libvxsena.so         SENA
libvxshark.so        IBM
libvxsunse3k.so      SUN
libvxsunset4.so      SUN
libvxvpath.so        IBM
libvxxp1281024.so    HP
libvxxp12k.so        HP
libvxibmsvc.so       IBM

# vxddladm listsupport libname=libvxCLARiiON.so
ATTR_NAME            ATTR_VALUE
======================================================================
LIBNAME              libvxCLARiiON.so
VID                  DGC
PID                  CLARiiON
ARRAY_TYPE           CLR-A/P, CLR-A/PF
ARRAY_NAME           EMC_CLARiiON
```

Dialog 3: Partial Listing of DMP Array Support Libraries

ASLs can be installed dynamically while VxVM is running. This makes it possible to add multi-path access control for new disk array models without stopping VxVM or rebooting the system. Installing an ASL does not automatically cause VxVM to recognize LUNs presented by new arrays, however. After ASL installation, the **`vxdctl enable`** VxVM command must be run to cause VxVM to discover new devices and their multi-path capabilities. Alternatively, if the locations of newly added devices are known, the **`vxdisk scandisks`** command can be issued with constraints to cause a (faster) partial device scan.

4.4.2 ASL Tuning

In releases prior to 5.0, the **vxconfigd** daemon calls each installed ASL for each device during discovery. As a result, deactivating ASLs that are not required (e.g., because no storage devices of the types they support are connected) can dramatically improve the speed with which a system running pre-5.0 code starts up. Dialog 4 illustrates the sequence of VxVM commands for deactivating an unused ASL (the ASL remains installed on the system and can be reactivated).

```
# vxddladm excludearray libname=libvxibmsvc.so
# vxdctl enable
# vxddladm listexclude all

The Diskarrays excluded
-----------------------

Based on Library names:
-----------------------
                libvxibmsvc.so

Based on VID, PID Combination:
------------------------------
```

Dialog 4: Deactivating an Unused ASL

When an ASL is deactivated in this way, the multi-path properties of the LUNs it controls do not change until VxVM discovery runs. During VxVM discovery, LUNs that had been controlled by a deactivated ASL are classified as generic disks. After the **vxddladm** command in Dialog 4 deactivates an ASL, the **vxdctl enable** command causes DMP discovery and reconstruction of its metanodes to reflect changes in device multi-path capabilities.

The process of deactivating unnecessary ASL is referred to as *ASL Tuning*. ASL Tuning is strongly recommended on all pre 5.0 releases of DMP (including those containing the *DMP Backport*). With Storage Foundation 5.0, the DMP device discovery algorithms have been enhanced so that optimum boot time performance is achieved out of the box, without any *ASL Tuning*.

4.4.3 Array Policy Modules

Array Policy Modules (APMs) are dynamically loadable kernel modules invoked by the **vxdmp** driver to perform disk array-specific path selection and failover, error processing, and SCSI reservation and release. While DMP contains default procedures for these common functions; installing an APM overrides the default procedure for all arrays whose array models are associated with that APM.

The default DMP APMs support generic A/A, A/P, A/PG and A/A-A array types in both single-host and multi-host configurations. Each array model includes a set of vectors that point to functions which implement policies such as:

- ***I/O request routing***, using one of the six built-in load balancing policies.
- ***Error handling***, including analysis, recovery, and DMP state changes. Built-in error handling policies include inquiry (the most common policy, described later), read-only path (for certain active-active array conditions such as EMC Symmetrix non-disruptive upgrade), and coordinated failover and failback for active-passive arrays in clusters
- ***Get Path State***, for obtaining information about current path and device configuration for use in error handling and elsewhere
- ***LUN group failover***, for active-passive arrays that support concurrent failover of entire LUN groups triggered a single event
- ***Explicit failover***, for arrays that support explicit failover functionality such as the EMC Clariion.
- ***Failover path selection***, using first available path, primary path preferred, or other alternate path selection algorithms

DMP includes one or more default procedures for each of these policies. Custom APMs that implement array-specific procedures can be substituted by creating array models that vector to the procedures that implement custom functions.

4.5 DMP Enhancements to Device Discovery

With Storage Foundation 5.0, DMP device discovery enhancements mainly come in two areas:

- Optimum boot time performance without the need for *ASL Tuning.* As was mentioned earlier, DMP 5.0 contains enhanced ASL selection logic in order to provide optimum boot time performance out of the box.
- Fabric topology discovery through the use of the SNIA HBA API. DMP 5.0 introduces an event source daemon: **vxesd**. This daemon monitors events on the system to trigger appropriate DMP configuration updates. It also gathers fabric topology information through the SNIA HBA API whenever that API is available. This allows DMP to build and maintain a map of the fabric topology, correlating port World Wide Names and array port ids with LUN paths information.

4.6 Maximizing Throughput Performance

DMP maximizes I/O throughput to a given device by efficiently distributing the I/O load over the set of paths available to that device. Throughput can be measured in terms of raw IOPS or Kbps, as well as in terms of Kbps per CPU time used. DMP has historically been a top performer on both of those metrics.

For a device in an *active-active* array, DMP distributes the I/O load over all the paths to the device. For a device in an *active-passive* array, DMP distributes the I/O load over the set of primary paths to the device. The secondary paths to a device in an *active-passive* array are used only when all primary paths have failed. If all the primary paths have failed and there are multiple secondary paths available to a device, DMP will perform load balancing over those too.

4.6.1 DMP I/O policies

The "optimal" path to a device can change over time based on I/O load, but path selection can also be a matter of system policy. DMP offers six different I/O policies that can be applied to multi-path storage devices:

Balanced Path

DMP's balanced path policy routes I/O requests to paths based on the starting block addresses they specify. Effectively, this policy divides a device's block address space into as many disjoint regions as there are active paths, and assigns each I/O request to a path that corresponds to the region in which the data it transfers falls.

For LUNs using the balanced path policy, DMP divides the starting data address specified each I/O request by the system-wide parameter **`DMP_PATHSWITCH_BLKS_SHIFT`** and discards the remainder. The quotient of the division modulo the number of active paths is used to index the active path used to issue the I/O command.

As an example, Figure 11 illustrates the balanced path I/O policy for an active-active device with two paths. For graphic simplicity, **`DMP_PATHSWITCH_BLKS_SHIFT`** has an artificially low value of 4. In this example, DMP would route read and write requests that specify a starting block addresses between 00 and 03 to path **`c1t0d0s0`**, those that specify one of blocks 04-07 to path **`c2t0d0s0`**, those that specify one of blocks 08-11 to path **`c1t0d0s0`**, and so forth.

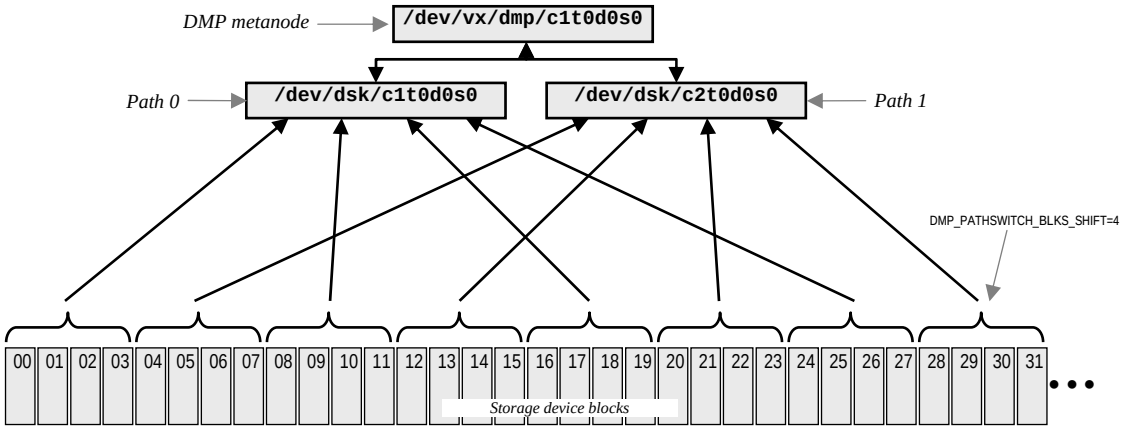

Figure 11: Balanced I/O Policy Path Selection

To illustrate the general algorithm, for a read or write request specifying a starting address of block 13, DMP would divide the address by **`DMP_PATHSWITCH_BLKS_SHIFT`** (13/4), giving an integer quotient of three. Three modulo the number of active paths (two) is one, so DMP would issue an I/O request to the operating system SCSI driver on path 1 (operating system device **`c2t0d0s0`**).

The balanced path policy is DMP's default policy for active-active arrays' LUNs (in earlier versions of DMP, it was the *only* available policy). It is particularly useful for high-speed sequential reading from active-active disk arrays and dual-port disk drives with read-ahead cache. Aligning the value of **`DMP_PATHSWITCH_BLKS_SHIFT`** with the sequential I/O request size causes DMP to route successive requests to alternate paths, which frequently allows data for two or more requests to transfer concurrently.

The default value for **`DMP_PATHSWITCH_BLKS_SHIFT`** is 2048 blocks, or 1 megabyte. The value can be overridden for individual arrays by using the **`setattr`** option of the **`vxdmpadm`** command. Overriding the global **`PATHSWITCH_BLKS_SHIFT`** value is useful in systems connected to two or more different types of arrays.

Round-Robin

The round-robin I/O policy evenly distributes I/O requests on each active I/O path to a device. For each request, DMP computes a pseudo-random number and assigns a path based on the computed number modulo the number of active paths.

The round-robin policy is useful when most I/O requests to a LUN specify approximately the same amount of data transfer, and in storage networks whose loading is relatively evenly distributed. In Storage Foundation 5.0, DMP , round robin is the default I/O policy for *active-passive* arrays with multiple primary paths enabled.

Minimum Queue Length

The minimum queue policy routes each I/O request to the active path that is supported by the HBA port (SCSI controller) with the smallest number of outstanding requests. Each time DMP assigns a request to a path, it increments the controller's outstanding request counter. Each time a request completes, the controller's request counter is decremented. For each new request, DMP selects the path with the smallest outstanding request counter value. This policy tends to counteract momentary load imbalance automatically, as for example, when a path bottlenecks because of error retries or overload from other LUNs. In DMP 5.0, minimum queue is the default I/O policy for *active-active* arrays and is the recommended I/O policy for all array types.

Adaptive

The adaptive routing policy allows DMP to dynamically route I/O requests based on calculated path priorities. When this policy is in effect, DMP records the service time and amount of data transferred for each request, and periodically calculates a priority for each path based on its recent throughput (bytes per second). The priority calculation algorithm produces higher priorities for paths that have recently delivered higher throughput. Incoming I/O requests are routed to paths in proportion to the paths' relative priorities. For example, if there are three active paths whose priorities are calculated as 3, 2, and 1 respectively, half of incoming requests are routed to path 1, a third to path 2, and the remaining sixth to path 3. As total I/O load on higher priority paths increases, the paths tend to deliver lower throughput, resulting in lower priorities on the next recalculation cycle.

The adaptive policy is useful with rapidly varying I/O loads, such as database applications that include both transactions (short transfers) and periodic table scans (long transfers). It is also useful in storage networks where different paths have discernibly different average performance, such as paths with different numbers of network "hops" or individual links of different speeds.

Priority

With the priority routing policy, DMP routes requests based on path priority as with the adaptive policy. Path priorities are assigned by administrators rather than calculated by DMP, however, and do not change without administrative action. The priority routing policy allows administrators to assign path priorities based on considerations other than performance, such as applications' relative importance to an enterprise.

Single Active Path (Preferred Path)

As its name implies, the single active path policy causes DMP to route all I/O requests to one path (called the *preferred* path). Only if the preferred path fails does DMP route I/O to a secondary one. If this policy is configured for an array, DMP routes all I/O requests to the single active path; all other paths are not used unless the active one fails.

Usage Note

I/O performance of active-passive arrays can be influenced by the assignment of different LUNs' preferred paths to different controllers. For example, in an array with two controllers, odd numbered LUNs might be assigned to one controller and even numbered LUNs to the other. If certain LUNs are known a priori to be heavily loaded, their primary path assignments can be distributed across controllers.

4.6.2 Determining the Effect of DMP Load Balancing Policies

Administrators can monitor the effect of any of these load balancing policies by using the **`vxdmpadm iostat`** command, as Dialog 5 illustrates.

```
# vxdmpadm iostat show all
                     cpu usage = 19733393us    per cpu memory = 32768b
                       OPERATIONS            MBYTES            AVG TIME(ms)
PATHNAME              READS     WRITES      READS     WRITES     READS   WRITES
c0t1d0s2                159          0         79          0 10.670886 0.000000
c0t0d0s2                 20          7       1220        162  0.042623 0.203704
c2t3d13s2               870        236      17426       2158  0.205153 0.101946
c3t3d13s2               334          4      15684         56  0.173871 0.017857
c2t3d12s2              9127        507       9236      18365  0.251299 0.076940
c3t3d12s2                45        649        255      18632  0.152941 0.074281
c2t3d11s2              1311         11       2068        185  0.133946 0.021622
c3t3d11s2                 0          1          0          8  0.000000 0.000000
c2t3d10s2           1241887    1200897   19851284   19849637  0.306276 0.213964
c3t3d10s2           1241300    1285848   19850538   19968007  0.274636 0.190663
c2t3d9s2            1240586    1200829   19849347   19850382  0.288218 0.215717
c3t3d9s2            1240839    1204491   19852391   19886690  0.255909 0.190155
c2t3d8s2            1240585    1200814   19849319   19850598  0.272347 0.241037
c3t3d8s2            1241296    1199258   19850357   19878681  0.241508 0.213279
c2t3d7s2            1240584    1201024   19849315   19850304  0.293359 0.242801
c3t3d7s2            1246021    1201910   19846281   19881291  0.262374 0.215197
c2t3d6s2               1311         12       2068        161  0.132495 0.031056
c3t3d6s2                  0          1          0          8  0.000000 0.000000
c2t3d5s2                 10         18        131        800  0.473282 0.013750
c3t3d5s2                  0          1          0          8  0.000000 0.000000
c2t3d4s2           17059295   11930299  137101333   76064118  0.102189 0.158222
c3t3d4s2               6703    2242672     625389    2373697  0.034997 0.899039
c2t3d3s2              82888    1923459     652854    8340732  0.291318 0.180282
c3t3d3s2              82119     913373     549421    4490416  0.357065 0.189884
c2t3d2s2                 21          0         10          0  5.600000 0.000000
c3t3d2s2                  0          0          0          0  0.000000 0.000000
c2t3d1s2                 21          0         10          0  6.400000 0.000000
c3t3d1s2                  0          0          0          0  0.000000 0.000000
c2t3d0s2                 21          0         10          0  5.400000 0.000000
c3t3d0s2                  0          0          0          0  0.000000 0.000000
```

Dialog 5: DMP Collection of I/O Statistics

The output of the **`vxdmpadm iostat`** command displays both the number of read and write operations, the amount of data read and written, and the average execution time for reads and writes for each path since VxVM's **`iostat`** daemon was started, or since its counters were last reset (using a variant of the same command). The command can be executed on a specific dmpnode and at intervals to determine the efficacy of a given load balancing algorithm under actual system I/O loads.

5 I/O Path Failover with DMP

Typically, the primary motivation for installing I/O path management software is to keep data accessible when an I/O path fails.

One challenge of I/O path management is distinguishing between failure of a path to a device and failure of the device itself. If a failed device is able to report its status (e.g., hard read error, unrecoverable positioning error, or write to a write-locked device), the determination is easy. More difficult to diagnose is a device that simply does not respond to an I/O request:

- Has the device failed?
- Has the path to the device failed, leaving the device unable to communicate the status of outstanding requests or accept new ones, even though it is functioning?
- Is the device simply too busy to respond within its timeout period?

When an I/O request fails, DMP must determine whether to re-route future I/O requests to alternate paths or disconnect the device.

Another increasing challenge of I/O path management is being able to manage scaled up environments. A host connected to an array with 500 LUNs and 2 paths to each LUN effectively requires the multi-pathing driver to manage 1000 LUN paths. A host connected to an array with 2000 LUNs and 8 paths to each LUN spread over 2 separate fabrics effectively results in 16,000 LUN paths for the multi-pathing driver to manage. A failure of one of the fabrics translates in the concurrent failure of 8,000 LUN paths (see Figure 12).

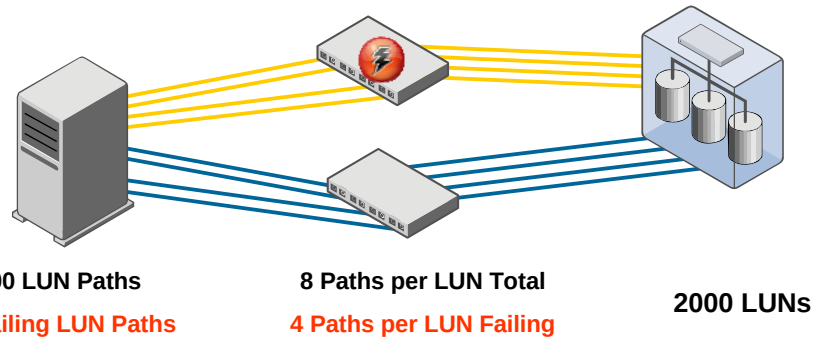

Figure 12: Consequence of a Switch Failure in a Large SAN

With Storage Foundation 5.0, these are the types of environments that DMP has been designed to efficiently manage. DMP has been specifically architected to make failover time independent of the number of LUNs actually affected by the failure. As such, DMP is able to failover thousands of LUN paths in a matter of seconds to a couple of minutes depending on the OS platform and the location of the failure in the SAN.

5.1 DMP Multi-threaded Core Design

In order to support scaled up configurations, in Storage Foundation 5.0, DMP relies on a fully multi-threaded, concurrent, non blocking core design. DMP comprises of a pool of generic worker threads in the kernel (10 by default) that can be invoked to handle specific tasks, as those tasks come up. Tasks can be any number of things, such as 'issuing SCSI inquiries', 'analyzing SCSI inquiry responses', 'disabling paths', 'restoring paths', etc.

To maximize the effectiveness of those multiple threads, DMP's position in the OS stack has also been enhanced, as is depicted in Figure 13.

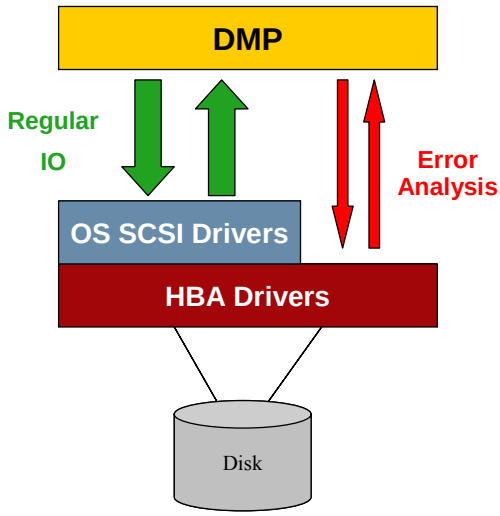

Figure 13: DMP and SCSI Bypass for Error Analysis

For regular application I/O, DMP effectively sits on top of the OS SCSI driver. This allows DMP to benefit from all the improvements that are made by OS vendors in their HBA and SCSI driver stacks. This also gives the HBA and SCSI layer a chance to recover without triggering a full failover for small transient errors.

Once the HBA and SCSI layers give up on an I/O and return the error to DMP, DMP performs all its error analysis by bypassing the SCSI layer and initiating SCSI commands and retrying the I/O directly to the HBA. SCSI Bypass relies on OS APIs and as such remains independent of the specific HBA and HBA driver on the system. This API is available on all OS platforms and provides the following benefits:

- Contrary to the regular SCSI interface, the SCSI bypass interface is an asynchronous interface: no thread ever has to wait for the response to an I/O that was issued. This means that if DMP must analyze 100 I/O errors, a single thread can issue 100 SCSI inquiries to get path states information concurrently. As responses return from the HBA, new tasks will be created to be picked up by DMP threads for further processing. These enables DMP to take roughly the same amount of time to analyze thousands of errors as it takes to analyze one error since the longest part of the analysis (issuing SCSI inquiries and waiting for responses) is now fully parallelized.
- Using this interface means all error analysis can be performed without wasting time in SCSI retries and timeouts. The only layer doing retries during error analysis is the HBA driver.

- Finally, the interface used for SCSI bypass is much richer than the regular SCSI interface in terms of error codes exchanged with DMP. This provides DMP with finer grained information during error analysis, allowing it to make better decisions.

5.2 Subpath Failover Groups

In addition to the increase in raw processing resources of the DMP driver, key algorithm enhancements have been made in Storage Foundation 5.0 to ensure DMP leverages its knowledge of the SAN topology when it comes to efficiently handling large failures.

We discussed earlier that DMP discovered the fabric topology. DMP uses this topology information to build a map of the SAN correlating all LUN paths managed with all ports in the SAN. With this information, DMP automatically configures *Subpath Failover Groups* (SFGs). An SFG is defined as the set of LUN paths that are supported by the same two HBA port and array controller port. Looking at the configuration depicted in Figure 12, one can see that the 16,000 LUN paths effectively translate in no more than 8 SFGs.

The concept of SFG is particularly relevant in modern Storage Area Networks. When the storage is provided by modern arrays, all the LUNs seen by a host are virtualized by the disk array. As such, a host seldom is exposed to single device failures. Instead, most failures originate in the network and translate into multiple devices going off line at the same time. Given that insight, all the members of an SFG will usually share the same state.

5.3 Suspect Paths and Pro-active Failure Handling

By default, an I/O error must reach DMP for error analysis to be triggered, a path to be disabled and I/O failover to occur. In Storage Foundation 5.0, DMP contains logic to pro-actively stop using paths it deems *Suspect*. A path that is deemed suspect is not used for new I/O. It is instead marked for background analysis to confirm its actual state. Pro-actively stopping using a path that may be failing has the benefit of reducing the amount of I/O that will have to be rerouted once the path actually fails.

There are three triggers that can get a path to be treated as suspect:

- The reception of a fabric event indicating that a port failed in the fabric. DMP registers on the SNIA HBA API provided by modern HBA drivers and will receive Registered State Change Notifications emitted by fabric switches following port logins, port failures, etc. Receiving a port failure event immediately gets all the members of SFGs affected by that port to be marked suspect.
- The reception of an I/O error on a LUN path will cause all other members of the SFG to which that LUN path belongs to be marked suspect.
- If the I/O response time on a path is seen to exceed a given threshold (10 seconds by default), then the path is ‘throttled’ and effectively treated as suspect.

The *suspect* path state is an internal state for DMP. A path is in the suspect state for the time it takes (milliseconds to seconds) DMP to confirm its actual state: enabled or disabled.

5.4 Path Analysis

Path analysis is triggered following the reception of an I/O error, or as a result the need to confirm the actual state of a suspect path, or as a time driven event that is part of path restoration policies.

DMP diagnoses path analysis requests by issuing one or more SCSI inquiry commands to the target device on the suspect path. (Most storage devices will respond to SCSI inquiry commands, even if they are unable to transfer data). If DMP receives a valid response within the timeout interval, it concludes that the device was simply too busy to respond, and re-queues the I/O request on the original path.

5.4.1 Path Failover

If a SCSI inquiry fails within a timeout interval, DMP fails I/O over to an alternate path. For active-active arrays, path failover consists simply of adjusting the load balancing algorithm in effect to account for the failed path. For active-passive arrays, a path failure may cause DMP initiates failover, either *implicitly*, by re-queuing outstanding I/O commands to an alternate path, or *explicitly* by invoking services of the APM that corresponds to the LUN being failed over. APMs for disk arrays that support LUN group failover can fail over entire groups of LUNs in response to a single event.

5.4.2 Automated Path Restoration

Every **`dmp_restore_interval`** seconds (300 by default), a restore task is created. This task is picked up by one of the DMP kernel threads and causes it to check I/O path state. By having kernel threads handle paths restoration, DMP ensures that it can reliably restore previously failed paths in a short amount of time without requiring any action on a host. In large datacenters where a single failure can affect hundreds of hosts, knowing that the multi-pathing driver will recover failed paths automatically in a matter of seconds is a significant operational efficiency gain.

In addition to time driver path restoration, DMP also leverages its ability to receive fabric events to reduce the time it takes to restore previously failed path. If a fabric event indicating that a port in the fabric is repaired is received, DMP immediately triggers a check of the state of the paths that are supported by this port without waiting for the next time based trigger.

The **`dmp_restore_policy`** tunable can be used to specify one of four path checking policies:

- **`check_all.`** This policy causes DMP to issue an inquiry command on every path to every device controlled by DMP, irrespective of the path's state. This policy provides timely notification of path failures, whether a path is idle or not. However, the large number of inquiry commands it issues may impact application performance in systems with a large number of storage devices.
- **`check_disabled.`** This policy causes DMP to issue inquiry commands to failed paths (but not administratively disabled ones). While it does not provide proactive notification of path failure, the low overhead of this policy makes it attractive for some data center operations.

- **check_periodic.** This policy causes DMP to execute the **check_disabled** policy except during every **dmp_restore_cycles**th **dmp_restore_interval,** when it executes the **check_all** policy. For example, with a **dmp_restore_interval** of 300 seconds and a **dmp_restore_cycles** of 10, the **restored** daemon would issue inquiries to **FAILED** paths every 300 seconds (5 minutes), and to all devices on all paths every 3000 seconds (50 minutes). This policy represents a compromise between overhead and timely discovery of failed paths.
- **check_alternate.** This policy is similar to the **check_all** policy, except that the restore threads stops issuing inquiry commands to a device when it finds two operational paths. This policy is useful with complex storage network topologies, where checking all paths might result in a large number of inquiries, possibly impacting application performance. Because it checks for at least one functional alternate path, this policy also protects against sudden path failure.

Administrators use the **vxdmpadm** utility to adjust the three tunables that control the restore kernel threads behavior, **dmp_restore_interval**, **dmp_restore_policy**, and, for the **check_periodic** policy only, **dmp_restore_cycles**.

5.4.3 Idle Lun Path Checking

By default, DMP includes idle LUN paths of imported disk groups in the set of paths that are checked every time a restore task is created. This ensures that a system administrator is notified that an idle LUN path failed before that path actually is needed for application I/O. By default, a LUN path is considered idle by DMP if it has not carried I/O over the past one second interval.

6 DMP Configuration and Tuning Considerations

6.1 Recommended DMP 5.0 Tuning

A lot of effort was specifically put into release 5.0 to ensure optimum out of the box performance. Still, the following are considered best practices:

- Use Minimum Queue I/O policy for all arrays. Extensive performance testing has demonstrated that minimum queue matches or beats the best alternative in terms of overall throughput for most workloads, while outperforming all alternatives in terms of availability.
- Increase the number of DMP threads on systems with 20 or more CPU cores. Best practice is to set the `dmp_daemon_count` tunable to half the number of available CPU cores on the system.
- To reduce failover time in unusual scenarios, limit the number of retry attempts that DMP performs before giving up on a path. The default behavior is for DMP to retry an I/O 5 times on paths that are deemed recoverable. DMP can be configured such that it will not spend more than 10 seconds retrying I/O on such path with the following:

  ```
  vxdmpadm setattr {enclosure enc-name} recoveryoption=timebound iotimeout=10
  ```

6.2 Recommended DMP Backport Tuning

All recommended DMP 5.0 tuning applies to 4.x DMP Backport releases. In addition:

- DMP Backport releases also benefit from ASL Tuning, as described in Section 4.4.2.
- On systems that are running with up to date HBA firmware and driver, DMP Backport releases can benefit from Subpath Failover Groups as described in Section 5.2 by setting the `monitor_fabric` tunable to `on`. This tunable controls the usage of the SNIA HBA API by DMP and is set to `off` by default in the DMP Backport releases.

6.3 DMP Tuning

All DMP 5.0 tuning can be done online as long as all tuning is performed using the **`vxdmpadm set-tune`** command. The set of tunables that can be configured online in DMP 5.0 MP1 are:

```
# vxdmpadm gettune all
            Tunable              Current Value  Default Value
------------------------------    -------------  -------------
dmp_failed_io_threshold               57600            57600
dmp_retry_count                           5                5
dmp_pathswitch_blks_shift                11               11
dmp_queue_depth                          32               32
dmp_cache_open                           on               on
dmp_daemon_count                         10               10
dmp_scsi_timeout                         30               30
dmp_delayq_interval                      15               15
dmp_path_age                            300              300
dmp_stat_interval                         1                1
dmp_health_time                          60               60
dmp_probe_idle_lun                       on               on
dmp_log_level                             1                1
dmp_retry_timeout                         0                0
dmp_fast_recovery                        on               on
```

Dialog 6: List of DMP Tunables Available DMP 5.0 MP1

There are three key DMP 'tunable' parameters that administrators should understand and actively manage, because they can affect system performance and availability significantly. The paragraphs that follow describe the functions of a few key parameters.

6.3.1 DMP_DAEMON_COUNT

This tunable controls the number of available kernel threads maintained by DMP to handle I/O errors, paths recovery, etc. The default is 10. Recommended tuning guideline is to set this to half the total number of CPU cores in a system.

6.3.2 DMP_FAILED_IO_THRESHOLD

The **`dmp_failed_io_threshold`** parameter represents the amount of time beyond which DMP considers an I/O request failure to represent a storage device failure. As Figure 7suggests, the VxVM virtualization layer issues I/O requests to DMP metadevices. Before forwarding a request to the operating system SCSI driver DMP saves the current system time. If the SCSI driver signals that an I/O request has failed, DMP samples the time at the moment of failure notification, and computes how long the request was outstanding. If the request was outstanding longer than **`dmp_failed_io_threshold`** seconds, DMP considers the device to have failed, and does not perform error recovery or path failover. If the request fails within **`dmp_failed_io_threshold`** seconds, DMP considers path failure as a possible cause, and initiates the error analysis and recovery procedures described earlier.

By default, the **dmp_failed_io_threshold** parameter is set to ten minutes. For non-redundant volumes, this is typically adequate; too low a value can result in I/O request failures that are actually due to transient storage network errors. For failure tolerant (mirrored) volumes, however, a lower value is usually appropriate, because the VxVM virtualization layer retries a failed I/O request on another of the volume's plexes. For failure tolerant volumes, therefore, the **dmp_failed_io_threshold** parameter should typically be set to a few tens of seconds. Of course, appropriate settings depend on steady-state storage network and device loading as well as individual disk or LUN performance characteristics.

The **dmp_failed_io_threshold** parameter is not used on the HP-UX platform. The HP-UX operating system provides an I/O request timing capability, called **pfto** that causes the SCSI driver to abort I/O requests that remain outstanding for too long. DMP treats I/O requests timed out by the HP-UX SCSI driver identically to failed requests on other platforms for which the **dmp_failed_io_threshold** time is exceeded. The **pfto** parameter can be set separately for each VxVM disk group using the **vxpfto** utility.

6.3.3 DMP_RETRY_COUNT

When a DMP I/O request to a SCSI driver fails within the **dmp_failed_io_threshold** interval, DMP begins error recovery by issuing SCSI inquiry commands to the target device on the suspect path. If the SCSI inquiry fails, then DMP disables the path and fails over the I/O. But if the SCSI inquiry succeeds, DMP will retry the I/O on the same path. If that I/O fails again, DMP will issue another SCSI inquiry, going through the logic outlined above.

For the same application I/O request, DMP will issue as many as **dmp_retry_count** inquiries before giving up on that path. The default value for **dmp_retry_count** is 5.[6] To reduce failover times in storage networks with extensive multi-path connectivity, this value can be lowered. A value of 2 is usually appropriate for storage networks with two or more paths to each device.

Upper layers of the I/O software stack must ultimately interpret I/O failures. Success of a DMP inquiry command followed by repeated failure of an I/O request can occur for a variety of reasons, some of them application-related. For example, write commands to write-protected devices always fail immediately. DMP's SCSI inquiry will succeed, but the application's command will continue to fail. Behavior like this usually indicates an application or administrative error.

As another example, if a disk that is part of a non-redundant LUN is failing intermittently, an application I/O request to the LUN might fail, but DMP's SCSI inquiry to the LUN may succeed. In this case, the hardware failure must be detected and diagnosed by means external to VxVM (e.g., by the disk array's error reporting mechanisms) and acted upon.

---

[6] On the HP-UX platform, the default value for **dmp_retry_count** is 30 because the HP-UX SCSI driver does fewer retries than those of other platforms.

6.3.4 DMP_PATHSWITCH_BLKS_SHIFT

For LUNs configured to use the balanced path I/O policy, DMP divides the starting data address specified in each I/O request by the parameter **dmp_pathswitch_blks_shift** and discards the remainder. The quotient of the division modulo the number of active paths becomes an index to the active path on which the I/O request is issued.

The **dmp_pathswitch_blks_shift** parameter is system-wide; it applies to all storage devices using the balanced path load balancing policy. Its default value is one megabyte (2048 blocks) on all supported platforms, but can be changed to match application requirements. For example, if an application reads large files sequentially using 2-megabyte requests, setting the value of **dmp_pathswitch_blks_shift** to 2 megabytes tends to alternate successive read requests among active paths. If the disk array recognizes sequential access patterns and reads ahead, application requests may be satisfied from data pre-read into cache.

6.3.5 An Additional AIX-Specific DMP Tuning Consideration

In addition to the three parameters that affect DMP availability and performance on all platforms, the **dmp_queue_depth** parameter is unique to the AIX platform. The **dmp_queue_depth** parameter limits the number of I/O requests to a single device that DMP will forward to the operating system driver in order to limit the time required to abort and redirect outstanding requests when a path fails.

While DMP is analyzing a possible path failure, application I/O requests may continue to arrive. If error analysis ultimately determines that the path has failed, DMP must abort all I/O requests outstanding to the operating system driver and reissue them on alternate paths. The operating system driver may take a long time to abort a large number of I/O requests, so DMP *throttles*, or limits the number of I/O requests that it allows to be outstanding to the operating system driver to the value of the **dmp_queue_depth** parameter.

DMP relays only **dmp_queue_depth** requests for a single device to the operating system driver; it retains any additional ones in its own queue. Each time a request completes, the number outstanding drops below **dmp_queue_depth**. If there are additional requests for the device in DMP's queue, a DMP kernel thread issues another one to the operating system driver.

The default value of the **dmp_queue_depth** parameter is 32.

On AIX, the value of **dmp_queue_depth** can also be changed by running the **smit** utility and selecting the **vxvm** subcommand.

6.4 Storage Network Hardware Settings

In addition to VxVM parameters, parameters that control the operation of host bus adapters and storage network switches can affect the performance and function of DMP. These parameters differ from vendor to vendor, but since similar components perform similar functions, conceptual similarity is found between HBAs and switches designed by different vendors. The paragraphs that follow describe HBA and storage network settings that may affect DMP operation and that are typically adjustable by system administrators.

6.4.1 Host Bus Adapter Settings

Host bus adapters (HBAs) send and receive user data, I/O requests, and control messages between host server memory and storage devices. To accomplish this, they transform data between host memory format and I/O bus or storage network format, and add protocol information (for transmission) or remove it (upon reception). The driver software that controls HBAs is at the lowest level of the storage I/O stack (Figure 7).

Because HBA drivers control access to I/O paths, they are often the point at which host servers first detect path failures. HBA designers build mechanisms into their drivers to help detect and analyze path failures. These mechanisms are typically controlled by adjustable parameters. These parameters should generally be set according to the storage hardware vendor recommendations.

The types of HBA parameters that affect DMP performance include:

- ***Link-down timeouts***. Operating system driver I/O request timeouts occur either because a device is unable to respond to a host message in time, or because the path on which a message was sent has failed. To detect path failure, HBAs typically start a timer each time a message is received, and report link failure if the timer lapses without receipt of a message. Most HBAs provide for user adjustment of a *link-down timeout* period. The link-down timeout interval should approximately equal DMP's **`dmp_failed_io_threshold`** parameter value. If the link-down timeout period is significantly shorter than **`dmp_failed_io_threshold`**, DMP will not detect path failures as quickly it should. If the HBA link-down timeout is significantly longer than **`dmp_failed_io_threshold`**, DMP may time I/O requests out when in fact nothing is wrong.
- ***Link retry count***. Links that are inherently noisy may experience more frequent transmission errors than links in less noisy environments. Most HBAs include some type of adjustable link retry count that can be adjusted upward if necessary to prevent premature failovers on links that experience periodic noise bursts.

6.4.2 Storage Network Switch Settings

Storage network switches also expose settable configuration parameters that can affect the operation of DMP. The two most common ones are:

- ***Interoperability mode***. Switch and director vendors often build proprietary enhancements to standard protocols into their products. While these enhancements are useful in homogeneous networks (those that include only one vendor's switches), they may not function correctly or optimally in heterogeneous networks, or with disks and disk arrays that have not been certified for operation with the switch vendor's products. Switches typically have an interoperability mode in which they adhere strictly to standard protocols for I/O and for inter-switch communications. Since DMP assumes standards-compliant storage device behavior, it is usually advisable to operate a storage network in this interoperability mode.

- ***Buffer credits***. Buffer credits are a throttling mechanism used by HBAs and storage devices to avoid being swamped by incoming message and user data frames. According to Fibre Channel protocols, an originator is only permitted to send a frame to a receiver when the receiver has granted it a buffer credit. For long paths, particularly those with multiple “hops” between intermediate switches, buffer credit shortages can increase latency (the elapsed time for sending a message from originator to receiver) well beyond what would be expected given the bandwidth and loading of the link between the two. Alternate paths with different numbers of “hops” might have significantly different average latencies, and if timeout parameters are set for the shorter path, “false” timeouts might occur frequently on the longer one. The best way to avoid timeouts caused by a lack of buffer credits is to increase the number of buffer credits that HBAs and disk array ports grant for multi-hop links. If this cannot be done, DMP and operating system driver timeouts should be raised to accommodate typical latencies actually encountered on longer paths.

7 Conclusion

Multi-pathing is the critical software layer that manages the set of connections between a server and its storage in a datacenter. As such, an effective multi-pathing driver must meet technical requirements (such as throughput performance, failover performance, scalability requirements) as well as business requirements (total cost of operation, contribution to datacenter agility).

DMP is a leading multi-pathing solution for maximum availability and performance in large heterogeneous SANs. DMP is the only truly hardware independent multi-pathing solution in the industry, it offers the broadest Hardware Compatibility List on the market, and is specifically architected to enable close tie in with any type of storage hardware.

DMP provides the following benefits:

- ***Storage connectivity virtualization.*** By fully virtualizing connectivity from the host to storage, DMP increases datacenter agility. A storage administrator benefits by being able to choose the type of storage hardware that best suits his needs knowing that the multi-pathing driver on hosts either already supports that storage hardware or can easily be enhanced to support it.
- ***Data availability***. If an I/O path to a multi-path storage subsystem fails, DMP automatically reroutes I/O requests to an alternate path transparently to applications and without administrator intervention. When a failed path returns to service, DMP restores the original path configuration automatically and transparently as well.
- ***Optimized I/O performance***. For storage subsystems that support simultaneous access to a single storage device on multiple paths, DMP enhances I/O performance by distributing I/O requests across all available paths according to pre-defined load balancing policies.
- ***Cluster configuration support***. DMP provides full, integrated support of SCSI-3 Persistent Reservations and I/O fencing. These features are required for Storage Foundation Oracle RAC and are strongly recommended for protecting against the risk of data corruption in cluster configurations.
- ***Reduced complexity and increased efficiency.*** By being an integral part of Veritas Storage Foundation, all DMPpaths in a datacenter can be centrally managed and monitored from Veritas Storage Foundation Manager.

DMP is specifically designed to scale to the largest data centers, and incorporates numerous innovations to ensure that it not only works harder but that it also works smarter. DMP allows Veritas Storage Foundation users to confidently standardize on Storage Foundation and take full advantage of the agility that comes from no longer being locked in by any hardware vendor.