

HP and Symantec: Enabling ILM solutions with Dynamic Storage Tiering white paper



Executive overview	3
Solving business problems with HP and Symantec	3
Veritas Storage Foundation's Dynamic Storage Tiering (DST) facility from Symantec	4
HP StorageWorks XP External Storage.....	4
HP StorageWorks family overview	5
Case study: Dynamic Storage Tiering in a database environment	6
Overview	6
Dynamic Storage Tiering policies exercised.....	6
General operations flow	7
Preparation and initialization	7
Hardware and software components	7
Set up Benchmark Factory software	11
Create volume sets and MVFS	12
Create VxVM disk groups (vxdg init).....	12
Create standard VxVM volumes for /oracle/u01 and /oracle/u02 (vxassist)	12
Create MVFS volumes for /oracle/u03 through /oracle/u08 (vxassist)	12
Import the disk groups just created (import).....	12
Initialize the volumes (vxvol).....	13
Create the Volume Set oracle_ts (vxvset).....	13
Create file systems (mkfs).....	13
Mount the file systems (mount).....	13
Add volumes to volume set under existing file system to expand the MVFS (fsvoladm add)	14
Illustrate volume usage and mapping under MVFS (fsvoladm list)	14
Define placement classes by tagging the volumes	15
Use placement classes to manage file locations.....	15
Tag the volumes	16
Check that volumes are correctly tagged	16
Create file placement policies	17
Specifics on file placement recommendations	17
DST policy definition (HPDST.XML)	18
Validate that the current DST policy has no syntax errors (fspadm validate)	20
Activate file change log	20
Assign the XML policy to the MVFS mount point (fspadm assign).....	20

Preview results of policies before move (what-if analysis).....	21
Enforce policies to move files (sweep and move).....	21
Initial sweep and move of files (procedures and results)	22
Enforce the active policy assigned to the file system (fspadm enforce)	22
DST report as captured during the "enforce" step	22
Schedule policies for ongoing file migrations.....	23
Resize volumes for LUN reclamation.....	23
Conclusion.....	24
For more information.....	25

Executive overview

Three distinct issues face IT managers as they strive to manage, store, and disseminate business information that users need to access quickly, reliably, and easily:

- Information volume (accelerated by government regulations regarding information retention)
- Budget constraints (related to capital and operational expenditure)
- Service Level Agreement (SLA) requirements (related to performance and availability)

HP and Symantec jointly present a tiered storage solution to help IT managers meet the preceding requirements. The solution combines the Dynamic Storage Tiering (DST) facility of Veritas Storage Foundation with the HP StorageWorks product family. The tiered storage solution enables customers to:

- Lower operational costs by automating data movement between storage infrastructure
- Reduce infrastructure costs by optimizing the use of existing devices through the alignment of data with appropriate storage resources
- Improve application performance by moving less frequently accessed data off high-performance storage
- Cost effectively and efficiently integrate large elements of storage and data (such as that derived from a business merger or acquisition)
- Provide a foundation for an Information Lifecycle Management (ILM) strategy

In an effort to help customers understand how they can deploy this solution, the two companies jointly tested and documented the procedural steps and results in this white paper as a case study. This white paper:

- Helps organizations understand how to implement DST effectively in an HP StorageWorks environment
- Outlines the use of DST in an Oracle® database environment
- Profiles the steps required to implement setup recommendations
- Serves as a reference guide for IT managers planning their own DST projects
- Outlines DST policy definitions and the actual movement of data across storage tiers based on the policies

Solving business problems with HP and Symantec

HP and Symantec have collaborated to deliver a tiered storage solution using newer, more flexible storage technologies. Symantec's DST facility, combined with HP StorageWorks XP External Storage, optimizes storage resources, dynamically streamlining operations and saving costs.

Veritas Storage Foundation's Dynamic Storage Tiering (DST) facility from Symantec

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

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

The DST alternative is most appropriate for applications with aggressive online requirements, a large number of files, and a business justification for periodically, or temporarily, migrating data between storage classes. Application examples include data warehousing, email (Critical Path), and online archives (media, engineering design documents, scanned images).

DST leverages two key technologies included with Veritas Storage Foundation: support for multi-volume file systems (MVFS) and automatic policy-based placement of files within the storage managed by a file system.

An MVFS occupies two or more virtual storage volumes and thereby enables a single file system to span across multiple physical storage devices. By presenting a single name space, multi-volumes are transparent to users and applications. This MVFS remains aware of each volume's identity, making it possible to control the locations at which individual files are stored.

When combined with the automatic policy-based placement of files, the MVFS provides an ideal storage tiering facility, which moves data automatically without any downtime requirements for applications and users alike.

HP StorageWorks XP External Storage

HP StorageWorks XP External Storage allows customers to host HP StorageWorks XP24000/12000/XP10000 Disk Array datasets on select external storage subsystems, including HP StorageWorks Modular Smart Array (MSA) storage systems, HP StorageWorks Enterprise Virtual Array (EVA) storage systems, legacy XP arrays, as well as current and legacy arrays from other storage providers.

The XP24000/12000/XP10000 Disk Arrays are an excellent choice for large-scale database applications, such as the Oracle environment chosen for the case study. HP XP External Storage allows provisioning of an XP24000/12000/XP10000 solution to optimize the return on IT investment. This action enables customers to focus high-performance/high-availability native XP24000/12000/XP10000 Disk Array capacity against most mission-critical data while hosting less critical datasets on cost-optimized MSA, EVA, or legacy XP Disk Array subsystems.

In addition, HP XP External Storage enables administrators to virtualize a number of storage systems into a single pool, regardless of the vendor. The combination of a variety of arrays connected to an XP12000 Disk Array will achieve a single high-performance (up to 1.9 million IOPS; 68 GB/sec), high-capacity (up to 32 PB) virtual pool.

HP XP External Storage virtualization has storage-based and network-based attributes. The XP24000 or the XP12000 Disk Array replaces the need for administrators to manage a collection of individual arrays. Business-user applications receive the data by accessing the XP array. Administrators find this capability useful for migrating data from one array to another easily, enabling “retiring” the old array when the data transfer is complete. Thus, a single pool of storage can contain multiple tiers—a straightforward way to deploy tiered storage without the overhead associated with multiple management interfaces and other complexities.

Also, the data on arrays that belong to an XP pool remains intact and usable, even when an administrator chooses to remove the array from the pool. This contrasts with typical network-based virtualization approaches that deposit data on constituent arrays using unique and proprietary schemes, resulting in unusable data (if the element is removed from the pool).

HP StorageWorks family overview

HP uses open, industry-standard technologies that leverage existing infrastructure investments, enabling tiered storage solutions that encompass the complete portfolio of storage devices, media, and solutions. As customer IT and business requirements change, the HP storage solution aligns and evolves, efficiently and cost effectively—from the desktop to the data center to archiving.

StorageWorks arrays	Features
XP24000/12000/10000	<ul style="list-style-type: none"> Always-on availability Data center consolidation and disaster recovery Large-scale Oracle/SAP applications HP-UX, Microsoft® Windows®, +20 more, including mainframe
HP StorageWorks EVA4100/6100/8100	<ul style="list-style-type: none"> Outstanding TCO ≤ 120 TB Storage consolidation and disaster recovery Simplification through virtualization Windows, HP-UX, Linux, OVMS, Tru64 + more
EVA4100 Starter Kits	<ul style="list-style-type: none"> Powerfully simple, affordable SAN deployment Affordable, SAN made easy Excellent for fast, small SAN deployment Scales to 28 TB Leverages EVA Available with SAN components
MSA	<ul style="list-style-type: none"> Low-cost consolidation Simple (ProLiant management) Affordable and simple DAS-to-SAN (ProLiant) Windows, HP-UX, Linux, Netware, OVMS, Tru64 + more

Case study: Dynamic Storage Tiering in a database environment

Overview

The following case study profiles:

- Steps required to implement DST in an HP StorageWorks environment with an Oracle database
- DST policy definition
- Observations and results of policy-based data migration activities

Database workloads represent an important use case for a tiered storage solution based on DST and the HP StorageWorks platform. The Oracle database was chosen for this case study because databases represent a core component of application environments and can consume a significant portion of high-performance storage capacity within the data center. Databases can contain hundreds or thousands of files with different access profiles and file types, making a database an excellent target for the reclamation of critical storage resources. Additionally, databases often support critical applications that cannot tolerate the downtime associated with traditional data migration, further highlighting the DST benefits.

In this case study, a single instance Oracle database (10g) was tested with a simulated 1,000-user workload running on HP-UX 11.23. A total of 574 GB of data (approximately 40% of the 1.4 TB of top tier XP storage associated with the Oracle database) was moved from top tier XP storage to lower storage tiers. DST policies based on file type and activity enabled the data movement while the application was online.

Dynamic Storage Tiering policies exercised

The following DST functions were exercised:

- File allocation based on directory
- File allocation based on type
- File relocation based on type
- File relocation based on I/O temperature
- Extent Balanced File System¹ (equal distribution of file extents and I/O across volumes)

¹ Also referred to as Load Balanced File System or Load Balancing File System in Symantec documentation

General operations flow

1. Preparation and initialization
2. Create VxVM volume sets and MVFS
3. Define placement classes by tagging the volumes
4. Create file placement policies using XML
5. Validate XML
6. Activate VxFS File Change Log
7. Assign the XML Policy to the MVFS
8. Preview results of policies before move
9. Enforce policies to move the files
10. Schedule policies for ongoing enforcement
11. Resize volumes for LUN reclamation

Preparation and initialization

Hardware and software components

Operating system:

- HP-UX operating system 11.23 with September 2006 patch bundle

Storage:

- HP EVA8000 (10 EVA LUNs)
- HP XP12000 (10 XP LUNs)
- HP EVA4000 (10 EVA E-LUNs presented through the XP12000 Disk Array)

Server:

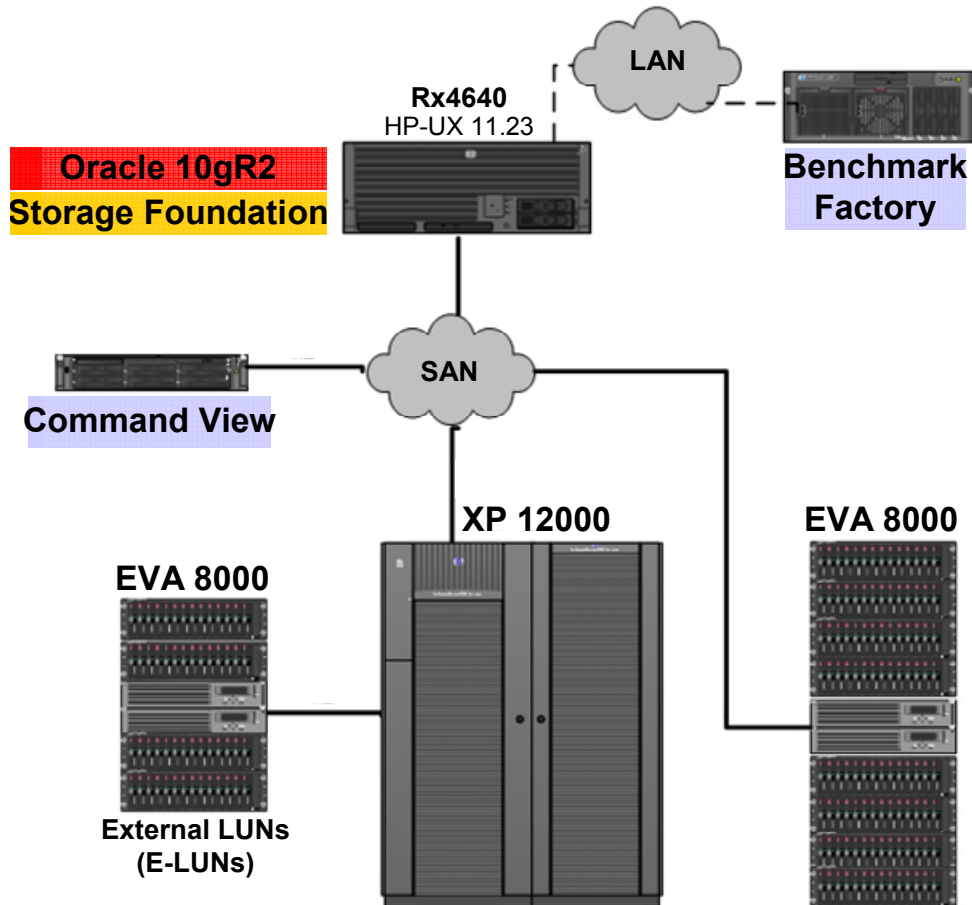
- HP ProLiant rx4640 server with 8-GB RAM (used to host the Oracle database)
- HP ProLiant 380 G4 server (used to host HP StorageWorks Command View EVA)

Software:

- Veritas 5.0 MP1 Storage Foundation for Oracle by Symantec for HP-UX
- HP StorageWorks XP External Storage Software
- Oracle 10g Release 2 (10.2)
- Quest Software Benchmark Factory for Databases

Figure 1 illustrates the configuration used in the case study.

Figure 1. Hardware configuration



The case study used an HP-UX Itanium®-based server running 11.23 with an operating system patched to equal or greater patch levels as required in the document "Oracle Database, Quick Installation Guide, 10g Release 2 (10.2) for HP-UX Itanium."

File system and Volume Manager versions

The DST facility requires that file systems to which it is applied be formatted with Veritas File System (VxFS) layout Version 7 or a newer one, and that volumes use Veritas Volume Manager (VxVM) layout Version 110 or a newer one. The following commands are used to validate and upgrade versions.

Determine the version of the volume's diskgroup:

```
# vxdg list dg1
```

If the version is less than 110, upgrade the diskgroup:

```
# vxdg upgrade dg1
```

Determine the disk layout version of the file system:

```
# vxupgrade /mnt1
```

If the disk layout version is less than 6, upgrade to Version 7:

```
vxupgrade -n 7 /mnt1
```

Create LUNs from XP, EVA, and HP XP External Storage

Using HP StorageWorks XP Command View, three LUNs were created for the **oracle_db** file system and two LUNs for the **oracle_redo** file system. The LUNs created for **oracle_db** were added to **dg1** Disk Group. The LUNs created for the **oracle_redo** were added to **dg2** Disk Group. Table 1.1 is a summary of the devices created for **oracle_db** and **oracle_redo**.

The XP12000 Disk Array was chosen to store the Oracle installation binaries and redo logs because it was the highest storage tier in our configuration. HP best practices indicate that if redo logs are necessary, it is best to have them available on the highest performing storage tier. A 25-GB file system was created called **oracle_db** and a 10-GB file system was created for **oracle_redo**.

As a best practice for implementing DST, it is important to clearly document or utilize software to keep track of device configurations. Table 1.1 illustrates easy device tracking as well as how each device maps to each storage tier. There were two device files per device because of the redundant paths. Only the Veritas DMP metanode device file is listed for each device.

Table 1.1. Device summary for oracle_db and oracle_redo

Mount	DG	Volume	Device File	LDEV*	LUN Size	Tier
oracle_db	dg1	lvol_db	c12t2d6	04:3c	8 GB	XP
oracle_db	dg1	lvol_db	c12t2d7	04:3d	8 GB	XP
oracle_db	dg1	lvol_db	c13t2d5	0a:1a	100 GB ²	XP
oracle_redo	dg2	lvol_redo	c12t3d0	04:3e	8 GB	XP
oracle_redo	dg2	lvol_redo	c12t3d1	04:3f	8 GB	XP

*Logical Device (LDEV) is term used for an XP LUN.

Three terabytes of storage were allocated for the Oracle table space and data files—1 TB each from the XP12000 Disk Array, EVA 8000, and the XP external storage. Each storage tier was placed into a different logical volume and added to a volume set **oracle_ts**, which in turn was used to host an MVFS called **oracle_tsfs**. DST policies can enact across logical volumes but not across disk groups. All of the devices listed in Table 1.2 were placed into the same **dg3** Disk Group.

² Used 100-GB LUN as it was the only one available but could have used a smaller LUN

Using HP StorageWorks XP Command View, 10 LDEVs of 100 GB in size were created for **oracle_tsfs** and presented to the server. The LUNs were added to the **dg3** Disk Group, and then added to the logical volume **lvol-xp**. Table 1.2 is the XP12000 Disk Array device table for oracle_tsfs.

Table 1.2. XP12000 Disk Array Device Table for oracle_tsfs

Mount	DG	Volume	Device File	LDEV	Size	Tier
oracle_tsfs	dg3	lvol_xp	c12t1d2	0a:07	100 GB	XP
oracle_tsfs	dg3	lvol_xp	c12t1d4	0a:00	100 GB	XP
oracle_tsfs	dg3	lvol_xp	c12t1d5	0a:01	100 GB	XP
oracle_tsfs	dg3	lvol_xp	c12t1d7	0a:0c	100 GB	XP
oracle_tsfs	dg3	lvol_xp	c12t2d0	0a:0d	100 GB	XP
oracle_tsfs	dg3	lvol_xp	c12t2d1	0a:0e	100 GB	XP
oracle_tsfs	dg3	lvol_xp	c12t2d2	0a:17	100 GB	XP
oracle_tsfs	dg3	lvol_xp	c12t2d3	0a:18	100 GB	XP
oracle_tsfs	dg3	lvol_xp	c12t2d4	0a:20	100 GB	XP
oracle_tsfs	dg3	lvol_xp	c13t1d6	0a:02	100 GB	XP

Using HP StorageWorks Command View EVA, 10 Virtual Disks of 100 GB in size were created for the Oracle table space and presented to the server. The LUNs were added to the **dg3** Disk Group, and then added to the logical volume **lvol-eva**. Table 1.3 is the EVA device table for oracle_tsfs.

Table 1.3. EVA8000 devices for oracle_tsfs

Mount	DG	Volume	Device File	Virtual Disk*	Size	Tier
oracle_tsfs	dg3	lvol_eva	c12t0d0	Vdisk051	100 GB	EVA
oracle_tsfs	dg3	lvol_eva	c12t0d1	Vdisk052	100 GB	EVA
oracle_tsfs	dg3	lvol_eva	c13t0d2	Vdisk053	100 GB	EVA
oracle_tsfs	dg3	lvol_eva	c13t0d3	Vdisk054	100 GB	EVA
oracle_tsfs	dg3	lvol_eva	c13t0d4	Vdisk055	100 GB	EVA
oracle_tsfs	dg3	lvol_eva	c13t0d5	Vdisk056	100 GB	EVA
oracle_tsfs	dg3	lvol_eva	c13t0d6	Vdisk057	100 GB	EVA
oracle_tsfs	dg3	lvol_eva	c13t0d7	Vdisk058	100 GB	EVA
oracle_tsfs	dg3	lvol_eva	c12t1d0	Vdisk059	100 GB	EVA
oracle_tsfs	dg3	lvol_eva	c13t1d1	Vdisk060	100 GB	EVA

*Virtual Disk (Vdisk) is the term used for an EVA LUN.

Using HP StorageWorks Command View EVA, 10 virtual disks of 100 GB each were created for **oracle_tsfs** and presented to the XP12000 Disk Array. HP StorageWorks XP Command View was used to discover the EVA devices and assign XP LDEVs to the EVA Virtual Disks. These devices are called E-LUNs. The XP12000 Disk Array E-LUNs were presented to the server, added to **dg3** Disk Group, and added to the logical volume **lvol-xpeva**. Table 1.4 is the EVA External LUNs device table for oracle_tsfs.

Table 1.4. External XP-EVA E-LUNs for oracle_tsfs

Mount	DG	Volume	Device File	LDEV*	Size	Tier
oracle_tsfs	dg3	lvol_xpeva	c12f0d0	19:00	100 GB	External -EVA
oracle_tsfs	dg3	lvol_xpeva	c12f0d1	19:01	100 GB	External -EVA
oracle_tsfs	dg3	lvol_xpeva	c13f0d2	19:02	100 GB	External -EVA
oracle_tsfs	dg3	lvol_xpeva	c13f0d3	19:03	100 GB	External -EVA
oracle_tsfs	dg3	lvol_xpeva	c13f0d4	19:04	100 GB	External -EVA
oracle_tsfs	dg3	lvol_xpeva	c13f0d5	19:05	100 GB	External -EVA
oracle_tsfs	dg3	lvol_xpeva	c13f0d6	19:06	100 GB	External -EVA
oracle_tsfs	dg3	lvol_xpeva	c13f0d7	19:07	100 GB	External -EVA
oracle_tsfs	dg3	lvol_xpeva	c12f1d0	19:08	100 GB	External -EVA
oracle_tsfs	dg3	lvol_xpeva	c13f1d1	19:09	100 GB	External -EVA

*Logical Device (LDEV) is term the used for an XP LUN. Because the EVA LUN is virtualized through the XP, an LDEV is assigned to the server.

To demonstrate the Extent Based File System, four logical volumes of one LUN each were created from the XP12000 Disk Array. Table 1.5 is the device table for the devices used to demonstrate the Extent Based File System.

Table 1.5. XP12000 Disk Array LUNs for Extent Based File System

Mount	DG	Volume	Device File	LDEV	Size	Tier
oracle_tsfs	dg3	lvol_xp1	c12f3d2	0a:05	100 GB	XP
oracle_tsfs	dg3	lvol_xp3	c13f2d3	0a:0b	100 GB	XP
oracle_tsfs	dg3	lvol_xp4	c13f2d1	0a:13	100 GB	XP
oracle_tsfs	dg3	lvol_xp4	c12f3d5	0a:1f	100 GB	XP

Set up Benchmark Factory software

Benchmark Factory software, developed by Quest Software, was used to populate the Oracle database and generate load on the database. The load simulated an OLTP-type of workload with 1,000 users. The OLTP tables were all in one table space, which consisted of several data files totaling 650 GB. All of the data files resided under the /oracle_tsfs mount point.

As updates were committed, redo information was generated and written to the Oracle redo logs under the mount point /oracle_redo. The redo logs were under constant IO. When a redo log is full, Oracle archives it before reusing it. In this environment, the archived log files were copied to /oracle_tsfs/archive directory.

Archived log files are backups and are not accessed again until a recovery is needed. Because of the nature of the Oracle log files, a policy was created to sweep the old archive files off the XP12000 Disk Array, and move them to the EVA External Storage LUNs. This is discussed in more detail later in the paper.

Create volume sets and MVFS

Create VxVM disk groups (vxdg init)

Disk groups are named collections of disks that share a common configuration. Volumes are created within a disk group and are restricted to using disks within that disk group. The disks are specified by their device names.

Create disk groups dg1, dg2, and dg3.

```
/usr/sbin/vxdg init dg1 c12t2d6 c12t2d7 ...
```

```
/usr/sbin/vxdg init dg2 c12t3d0 c12t3d1 ...
```

```
/usr/sbin/vxdg init dg3 c10t0d1 c10t0d2 ...
```

A new disk can be added to the disk group at any time in the future.

```
/usr/sbin/vxdg -g dg3 adddisk c13t1d6
```

Create standard VxVM volumes for /oracle/u01 and /oracle/u02 (vxassist)

Create volumes lvol_db and lvol_redo in dg1 and dg2 Disk Groups, respectively.

```
/usr/sbin/vxassist -g dg1 make lvol_db 25G
```

```
/usr/sbin/vxassist -g dg2 make lvol_redo 10G
```

/oracle/u01 resides under the mount point /oracle_db.

dg1: lvol_db (25 GB; 3 LUNs XP OPEN-V, 2 OPEN-9)

/oracle/u02 resides under the mount point /oracle_redo.

dg2: lvol_redo (10 GB; 2 LUNs XP, OPEN-9)

Create MVFS volumes for /oracle/u03 through /oracle/u08 (vxassist)

Create volumes lvol_xp, lvol_eva, and lvol_xpeva in dg3 Disk Group.

```
/usr/sbin/vxassist -g dg3 make lvol_xp 950G
```

```
/usr/sbin/vxassist -g dg3 make lvol_eva 950G
```

```
/usr/sbin/vxassist -g dg3 make lvol_xpeva 950G
```

/oracle/u03... /oracle/u08 = /oracle_tsfs (MVFS mount point)

dg3: lvol_eva (950GB - 10LUNs (OPEN-V))

lvol_xpeva (950GB- 10 LUNs (OPEN-V))

lvol_xp (950GB - 11 LUNs (OPEN-V))

Import the disk groups just created (import)

```
/usr/sbin/vxdg import dg1
```

```
/usr/sbin/vxdg import dg2
```

```
/usr/sbin/vxdg import dg3
```

Initialize the volumes (vxvol)

```
/usr/sbin/vxvol -g dg1 startall
```

```
/usr/sbin/vxvol -g dg2 startall
```

```
/usr/sbin/vxvol -g dg3 startall
```

Create the Volume Set oracle_ts (vxvset)

Note:

Volume Set cannot span across multiple disk groups. All volumes in the following volume set definition are in the same disk group (dg3)

Start by creating the volume set with only one volume lvol_eva. Add other volumes to the volume set as next steps.

```
/usr/sbin/vxvset -g dg3 make oracle_ts lvol_eva
```

```
/usr/sbin/vxvset -g dg3 addvol oracle_ts lvol_xpeva
```

```
/usr/sbin/vxvset -g dg3 addvol oracle_ts lvol_xp
```

Create file systems (mkfs)

```
/usr/sbin/mkfs -F vxfs -o largefiles /dev/vx/rdisk/dg1/lvol_db
```

```
/usr/sbin/mkfs -F vxfs -o largefiles /dev/vx/rdisk/dg2/lvol_redo
```

```
/usr/sbin/mkfs -F vxfs -o largefiles /dev/vx/rdisk/dg3/oracle_ts
```

/oracle_ts is the MVFS. /lvol_db and /lvol_redo are regular file systems.

Mount the file systems (mount)

```
mkdir /oracle_db
```

```
mount -F vxfs /dev/vx/dsk/dg1/lvol_db /oracle_db
```

```
mkdir /oracle_redo
```

```
mount -F vxfs /dev/vx/dsk/dg2/lvol_redo /oracle_redo
```

```
mkdir /oracle_tsfs
```

```
mount -F vxfs /dev/vx/dsk/dg3/oracle_ts /oracle_tsfs
```

Add volumes to volume set under existing file system to expand the MVFS (fsvoladm add)

Note

The “fsvoladm” command allows new volumes to be added to the volume set of the MVFS while the file system is online and mounted (/oracle_tsfs). It should be noted that this was done while the Oracle table spaces were being populated.

```
/opt/VRTS/bin/fsvoladm add /oracle_tsfs lvol_xp1 95G
/opt/VRTS/bin/fsvoladm add /oracle_tsfs lvol_xp2 95G
/opt/VRTS/bin/fsvoladm add /oracle_tsfs lvol_xp3 95G
/opt/VRTS/bin/fsvoladm add /oracle_tsfs lvol_xp4 95G
```

Four additional XP volumes were added to the MVFS to demonstrate an Extent Balanced File System.

The final composition of the MVFS (/oracle_tsfs) is as follows:

```
/oracle/u03 thru /oracle/u08 = /oracle_tsfs (vset name = oracle_ts)
```

dg3:

```
lvol_eva (950GB - 10LUNs (OPEN-V))
lvol_xpeva (950GB- 10 LUNs (OPEN-V))
lvol_xp (950GB - 11 LUNs (OPEN-V))
lvol_xp1 (95GB - 1 LUN OPEN-V)
lvol_xp2 (95GB - 1 LUN OPEN-V)
lvol_xp3 (95GB - 1 LUN OPEN-V)
lvol_xp4 (95GB - 1 LUN OPEN-V)
```

Illustrate volume usage and mapping under MVFS (fsvoladm list)

Note that volumes lvol_xp1, lvol_xp2, lvol_xp3, and lvol_xp4 are empty. The Oracle table space files were generated and populated on vols lvol_eva, lvol_xp, and lvol_xpeva.

```
/opt/VRTS/bin/fsvoladm list /oracle_tsfs
```

devid	Size	Used	Avail	Name
0	996147200	373832	995773368	lvol_eva
1	996147200	671918448	324228752	lvol_xp
2	996147200	16	996147184	lvol_xpeva
3	99614720	16	99614704	lvol_xp1
4	99614720	16	99614704	lvol_xp2
5	99614720	16	99614704	lvol_xp3
6	99614720	16	99614704	lvol_xp4

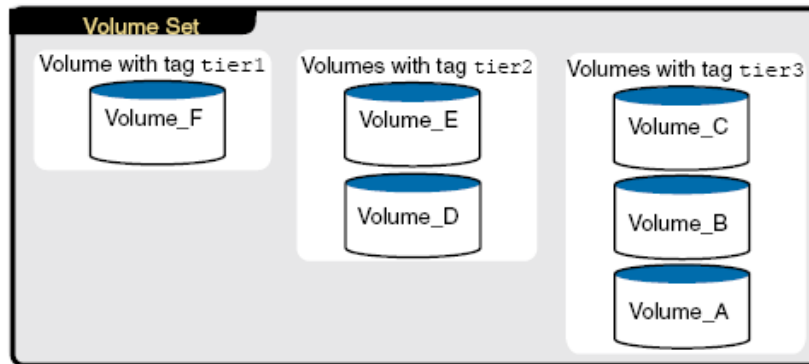
Define placement classes by tagging the volumes

Use placement classes to manage file locations

Administrators of multi-volume VxFS file systems can control the locations of files within volume sets by defining file placement policies that control both initial file location and the circumstances under which existing files are relocated. A VxFS file placement policy consists of rules that restrict the locations of files to administrator-defined subsets of the volumes in a file system's volume set. These subsets are called placement classes. A placement class is typically identified with a storage tier. Policy rules cause files to be created and extended within specified placement classes, and to be relocated to other placement classes when they meet certain naming, activity, access rate, and size-related qualifications.

As an example, Figure 2 represents a VxFS MVFS whose volume set consists of three placement classes called tier 1, tier 2, and tier 3.

Figure 2. Example volume set for MVFS



Such a volume set might be suitable for a file system containing a few critical files (tier1), a larger number of files of average importance (tier2), and a still larger number of inactive files (tier3).

An administrator assigns VxVM volumes to placement classes by associating character strings called volume tags with them. In Figure 2, Volume_F is tagged tier1, Volume_D and Volume_E are tagged tier2, and Volume_A, Volume_B, and Volume_C are tagged tier3. For file placement purposes, VxFS treats all of the volumes in a single placement class as equivalent, and balances space allocation approximately equally across them.

To the VxFS file system, a volume tag is a character string used to classify a volume. Any storage tier may be tagged with any convenient name. For example, some enterprises name their storage tiers after precious metals—gold (uppermost tier), silver (middle tier), and bronze (lowest tier).

VxFS imposes no capacity, performance, availability, or other constraints on placement classes. Any volume may be added to any placement class by assigning to it the tag that identifies the class, no matter what its type or the types of other volumes in the class. HP best practices suggest placing only volumes with identical, or at least very similar, I/O performance and availability characteristics in a single placement class—in other words, to identify a placement class with a physical storage tier.

Tag the volumes

Because Veritas Storage Foundation can create logical volumes in any number of storage array configurations, it is necessary to tag the volumes to associate them with a particular class or tier of storage. In this fashion, administrators can designate volumes within a particular array as tier 1 storage and volumes associated with another storage array as tier 2. It is even possible to classify volumes within a single array as different tiers, which is necessary when a single storage device presents up multiple types of physical storage. Or, as in the case of HP XP External Storage, presents up LUNs from multiple storage arrays through a single device.

For example, the following commands tag the `lvol_eva` volume as placement class `eva`, `lvol_xpeva` as placement class `xpeva`, and `lvol_xp` as placement class `xp`.

```
/usr/sbin/vxassist -g dg3 settag lvol_eva vxfs.placement_class.eva
/usr/sbin/vxassist -g dg3 settag lvol_xpeva vxfs.placement_class.xpeva
/usr/sbin/vxassist -g dg3 settag lvol_xp vxfs.placement_class.xp
/usr/sbin/vxassist -g dg3 settag lvol_xp1 vxfs.placement_class.xp
/usr/sbin/vxassist -g dg3 settag lvol_xp2 vxfs.placement_class.xp
/usr/sbin/vxassist -g dg3 settag lvol_xp3 vxfs.placement_class.xp
/usr/sbin/vxassist -g dg3 settag lvol_xp4 vxfs.placement_class.xp
```

The first volume in a VxFS file system's volume set will be used to store metadata by default (property known as `metadataok`). VxFS sets all other volumes' `dataonly` flags at file system creation time, or as volumes are added to a volume set. If necessary, the `metadataok` and `dataonly` flag settings can be changed at the time of creation or at a later point, using one of the following commands:

```
# fsvoladm setflags dataonly <file system>_<diskgroup>_<volume>
# fsvoladm add -f metadataok <file system> <volume> <size>
# fsvoladm clearflags dataonly <file system> <volume>
# fsvoladm setflags dataonly <file system> <volume>
```

Check that volumes are correctly tagged

```
/usr/sbin/vxvoladm -g dg3 listtag
```

TY NAME	DISKGROUP	POOL	TAG
v lvol_eva	dg3	-	vxfs.placement_class.eva
v lvol_xpeva	dg3	-	vxfs.placement_class.xpeva
v lvol_xp	dg3	-	vxfs.placement_class.xp
v lvol_xp1	dg3	-	vxfs.placement_class.xp
v lvol_xp2	dg3	-	vxfs.placement_class.xp
v lvol_xp3	dg3	-	vxfs.placement_class.xp
v lvol_xp4	dg3	-	vxfs.placement_class.xp

Create file placement policies

VxFS places files among the volumes of a file system's volume set, in accordance with the file system's active file placement policy. A file placement policy consists of rules that govern the initial location and subsequent relocation of designated sets of files. A rule may designate the files to which it applies by name, directory, ownership, or combination of the three.

Policy rules specify where files should be placed in terms of placement classes rather than specific volumes. This makes it unnecessary to change a file system's active placement policy when volumes are added to or removed from its volume set. Moreover, because the volume tags that define placement classes need not be unique, one placement policy can be used for any number of file systems with similar requirements and storage complements.

Policy rules specify both initial allocation destinations and relocation destinations as priority-ordered lists of placement classes. Files are allocated in the first placement class in the list if free space permits; in the second class if no free space is available in the first; and so forth.

In addition to the <CLASS> sub-element, a <DESTINATION> element may contain a <BALANCE_SIZE> sub-element. When an <ON> clause contains a <BALANCE_SIZE> sub-element, VxFS allocates extents of the indicated size (rounded up to the nearest multiple of the file system block size) for files selected by the rule, distributing its allocations approximately uniformly across the volumes in the placement class. Distributing a large file across multiple volumes may provide performance benefits for both transactional and streaming access.

Specifics on file placement recommendations

An important application of VxFS DST is automating the relocation of inactive files to lower cost storage. If a file has not been accessed for the period of time specified in the <ACCAGE> element, a scan of the file system should schedule the file for relocation to a lower storage.

DST implements the concept of I/O temperature and access temperature to overcome the deficiencies of a binary measure such as access age. For example, in a rule specifying that files on tier 2 volumes that have been accessed within the last three days should be relocated to tier 1 volumes, no distinction is made between a file that was browsed by a single user and a file that actually was used intensively by applications. A file's I/O temperature remedies this by quantifying relative activity. I/O temperature is equal to the number of bytes transferred to or from it over a specified period of time divided by the size of the file. For example, if a file occupies 1 MB of storage at the time of an fsppadm enforce operation and the data in the file has been completely read or written 15 times within the last three days, VxFS calculates its 3-day average I/O temperature to be 5 (15 MB of I/O ÷ 1-MB file size ÷ 3 days). In other words, one "degree" of I/O temperature is equivalent to one file-size-worth of I/O per day. For example, a 50-MB file that gets 150 MB of I/O transfer per day has a temperature of 3.

Similarly, a file's average access temperature is the number of read or write requests made to it over a specified number of 24-hour periods divided by the number of periods. One "degree" of access temperature is equivalent to one file access per day. For example, a file that is accessed 150 times per day has a temperature of 150. Unlike I/O temperature, access temperature is unrelated to file size. A large file to which 20 I/O requests are made over a 2-day period has the same average access temperature as a small file accessed 20 times over a 2-day period.

DST policy definition (HPDST.XML)

The following XML snippets illustrate placement policies and relocation rules:

- All archive files are created in the xp tier and eventually moved to xpeva tier.
- Dbf (*.dbf) files are created in the xp tier.
- Dbf (*.dbf) files are balanced in equal 8-MB chunks across volumes in the xp tier.
- Control (*.ctl) files are created on the eva tier and relocated to the xp tier based on I/O temperature (when they heat up).
- All other files are created first on the eva tier, then the xpeva tier (when eva is full), then the xp tier (when xpeva is full).

```
<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM "/opt/VRTSfspro/config/placement_policy.dtd">
<PLACEMENT_POLICY Version="5.0" Name="HPDST_Policy">
```

```
<RULE Flags = "data" Name = "archive_files">
<SELECT>
  <DIRECTORY Flags = "recursive"> archive </DIRECTORY>
</SELECT>
```

```
<CREATE>
  <ON>
    <DESTINATION>
      <CLASS> xp </CLASS>
    </DESTINATION>
  </ON>
</CREATE>
```

```
<RELOCATE>
  <TO>
    <DESTINATION>
      <CLASS> xpeva </CLASS>
    </DESTINATION>
  </TO>
</RELOCATE>
```

```
<RULE Flags = "data" Name = "dbf_files">
<SELECT>
  <PATTERN> *.dbf </PATTERN>
  <PATTERN> test1* </PATTERN>
<CREATE>
  <ON>
    <DESTINATION>
      <CLASS> xp </CLASS>
      <BALANCE_SIZE Units="MB"> 8 </BALANCE_SIZE>
    </DESTINATION>
  </ON>
</CREATE>
<RELOCATE>
```

```

    <TO>
        <DESTINATION>
            <CLASS> xp </CLASS>
            <BALANCE_SIZE> Units= "MB"> 8 </BALANCE_SIZE>
        </DESTINATION>
    </TO>
</RELOCATE>

```

```

<RULE Flags = "data" Name = "control_files">
<SELECT>
    <PATTERN> *.ctl </PATTERN>
<CREATE>
    <ON>
        <DESTINATION>
            <CLASS> eva </CLASS>
        </DESTINATION>
    </ON>
</CREATE>
<RELOCATE>
    <TO>
        <DESTINATION>
            <CLASS> xp </CLASS>
        </DESTINATION>
    </TO>
    <WHEN>
        <IOTEMP Type = "nrwbytes">
            <MIN Flags = "gteq"> 2 </MIN>)
            <PERIOD Units = "days"> 2 </PERIOD>)
        </IOTEMP>
    </WHEN>
</RELOCATE>

```

```

<RULE Flags = "data" Name= "all_the_rest">
<SELECT>
    <PATTERN> * </PATTERN>
<CREATE>
    <ON>
        <DESTINATION>
            <CLASS> eva </CLASS>
        </DESTINATION>
    <DESTINATION>
        <CLASS> xpeva </CLASS>
    </DESTINATION>
    <DESTINATION>
        <CLASS> xp </CLASS>
    </DESTINATION>
    </ON>
</CREATE>

```

```

<RELOCATE>
  <TO>
    <DESTINATION>
      <CLASS> eva </CLASS>
    </DESTINATION>
  </TO>
</RELOCATE>

</RULE>
</PLACEMENT POLICY>

```

Validate that the current DST policy has no syntax errors (fsppadm validate)

```
/opt/VRTS/bin/fsppadm validate /oracle_tsfs hpdst.xml
```

Activate file change log

If a file system's active placement policy includes any <IOTEMP> or <ACCESSTEMP> clauses, VxFS begins policy enforcement by using information in the file system's File Change Log (FCL) file to calculate average I/O activity against all files in the file system during the longest <PERIOD> specified in the policy. Shorter specified periods are ignored. VxFS uses these calculations to qualify files for I/O temperature-based relocation and deletion.

If a policy contains any <IOTEMP> or <ACCESSTEMP> relocation qualifiers, no deletions or relocations based on those qualifiers will occur unless the FCL is running, filestats storage is enabled, and the amount of data saved in the FCL represents a period at least as large as the largest <PERIOD> specified in the policy.

DST simplifies this by ensuring that when a file placement policy containing IOTEMP or ACCESSTEMP qualifiers is assigned to a file system, VxFS automatically starts the FCL and causes statistics to be stored in it³. The FCL_keeptime (which determines how long FCL records will be retained) should be set at or slightly larger than the largest value specified in any <PERIOD> element in the policy. `vxtunefs -o fcl_keeptime=<seconds> <file-system>` can be used to adjust the fcl_keeptime.

A file system's metadata storage capacity should be increased if necessary when a policy containing <IOTEMP> or <ACCESSTEMP> qualifiers is assigned. As its name implies, the FCL records information about changes made to files in a VxFS file system. In addition to recording creations, deletions, and extensions, the FCL periodically captures the cumulative amount of I/O activity (number of bytes read and written) on a file-by-file basis. File I/O activity is recorded in the FCL each time a file is opened or closed, as well as at timed intervals to capture information about files that remain open for long periods.

Assign the XML policy to the MVFS mount point (fsppadm assign)

The following command assigns policy defined in hpdst.xml file (as previously listed) to the oracle_tsfs MVFS.

```
/opt/VRTS/bin/fsppadm assign /oracle_tsfs /oracle_tsfs/hpdst.xml
```

³ FCL and filestats can be turned on manually by using any of the following commands:

```

# /opt/VRTS/bin/fcladm on <file system>
# /opt/VRTS/bin/fcladm set filestats <file system>
# /opt/VRTS/bin/vxtunefs -o fcl_keeptime=<seconds> <file system>

```

Preview results of policies before move (what-if analysis)

The `fsppadm query` command can be used to determine the locations of some or all of the files in an MVFS and the placement policy rules that resulted in those locations. Note that for better readability, we used the following `-l` option that provides a summarized version of the output (list of files and their expected destination tiers). The `-a` option will provide more information including file size.

```
/opt/VRTS/bin/fsppadm query -l /oracle_tsfs | more

"/oracle_tsfs/flashrecovery/HPSYM/flashback/o1_mf_2x3pn4rl_.flb" eva
"/oracle_tsfs/flashrecovery/HPSYM/flashback/o1_mf_2x3r5spy_.flb" eva
"/oracle_tsfs/flashrecovery/HPSYM/flashback/o1_mf_2x3rpp8y_.flb" eva
.....
"/oracle_tsfs/archive/1_1_613986588.dbf" xpeva
"/oracle_tsfs/archive/1_2_613986588.dbf" xpeva
"/oracle_tsfs/archive/1_3_613986588.dbf" xpeva
.....
"/oracle_tsfs/archive/1_5276_613986588.dbf" xpeva
"/oracle_tsfs/archive/1_5277_613986588.dbf" xpeva
"/oracle_tsfs/archive/1_5278_613986588.dbf" xpeva
.....
"/oracle_tsfs/controlfile/control03.ctl" eva
"/oracle_tsfs/test2/test2_file1" eva
"/oracle_tsfs/test2/test2_file2" eva
"/oracle_tsfs/test2/test2_file3" eva
.....
"/oracle_tsfs/test1/test1_file1" xp
"/oracle_tsfs/test1/test1_file2" xp
"/oracle_tsfs/test1/test1_file3" xp
.....
"/oracle_tsfs/test3/test3_file1" eva
"/oracle_tsfs/test3/test3_file2" eva
"/oracle_tsfs/test3/test3_file3" eva
```

Enforce policies to move files (sweep and move)

File relocation is performed when a policy is enforced, either on-demand or periodically, and is similar to initial allocation. Files are relocated to the first placement class listed in the rule that selects them if space is available, to the second class if no space is available in the first, and so forth.

File relocation may be unconditional, or it may be based on qualifications such as time, since most recent access or modification, intensity of access by applications (I/O temperature), and file size. A file system's policy for allocating and relocating files is expressed in a set of internal data structures called its active file placement policy.

Administrators write file placement policies in the XML language, according to a Document Type Description (DTD) supplied with VxFS. The Storage Foundation graphical management console includes wizards that create four popular types of policies in accordance with user-supplied parameters.

File placement policies are not inherently bound to specific file systems. An administrator assigns a policy to a file system, making it the file system's active policy. A file system may have only one active policy at a time. When assigned to a file system, a file placement policy allocates and relocates files among the placement classes that are named in the policy and represented by tags assigned to the volumes.

Initial sweep and move of files (procedures and results)

Placement policy enforcement is typically scheduled to run from an authorized account at regular intervals, but it can also be run by an authorized administrator whenever necessary to cause immediate relocation and deletion according to the rules of the active file placement policy.

In addition to entire file systems, the command can be run against specific files or sets of files. Unscheduled policy enforcement against a specific set of files can be particularly useful after an administrative action such as renaming a group of files changes conditions in such a way that identifiable sets of files are no longer located in accordance with the active policy.

The first “sweep” is likely to move the substantial amount of data from existing tier locations to the intended destination tier locations.

Enforce the active policy assigned to the file system (fspadm enforce)

The following command will cause a scan/sweep of data (files) and generate a summary report:

```
/opt/VRTS/bin/fspadm enforce /oracle_tsfs -r /tmp/dst-report
```

Note that the very first scan/sweep following the assignment of a new policy is typically the most intensive in unstructured environments. In this example, 574 GB (5,534 files) of data was relocated over a 13-hour window of time. Note that a majority of the data files were active database files (*.dbf) with WorkBench Factory creating a simulated user load (1,000 users) during file relocation.

The -r switch in the fspadm enforce command causes the detailed relocation report to be written to the file /tmp/dst-report. This report lists files that were relocated as a result of policy enforcement.

Sweep path : /oracle_tsfs

Files enforced : 5534

KB moved : 574810804

Tier Name	Size (KB)	Free Before (KB)	Free After (KB)
eva	996147200	9949222616	6561511152
xp	1394606080	719778184	1288187544
xpeva	996147200	996062440	753621656

DST report as captured during the “enforce” step

Following are details about which files were selected and relocated by the assigned policy rule (as captured in the /tmp/dst-report file). Only a sample of the data is provided for illustration.

Current Class	Current Volume	Relocated Class	Relocated Volume	Rule	File
xp	lvol_xp	eva	lvol_eva	all_the_rest	/oracle_tsfs/flashrecovery/HPSYM/flashback/o1_mf_2x3zft9o_.flb
xp	lvol_xp	eva	lvol_eva	all_the_rest	/oracle_tsfs/flashrecovery/HPSYM/flashback/o1_mf_2x3rpp8y_.flb
xp	lvol_xp	eva	lvol_eva	all_the_rest	/oracle_tsfs/flashrecovery/HPSYM/flashback/o1_mf_2x40960k_.flb
.....					
xp	lvol_xp	xpeva	lvol_xpeva	archive_files	/oracle_tsfs/archive/1_15_613986588.dbf
xp	lvol_xp	xpeva	lvol_xpeva	archive_files	/oracle_tsfs/archive/1_17_613986588.dbf
xp	lvol_xp	xpeva	lvol_xpeva	archive_files	/oracle_tsfs/archive/1_18_613986588.dbf

It should be noted that the xp-class (vxfs.placement_class.xp) that contains volumes lvol_xp, lvol_xp1, lvol_xp2, lvol_xp3, and lvol_xp4 had the extent balanced rule applied (see HPDST.XML policy). The extent balancing after enforcement of the DST policy is clearly illustrated by the nearly equal amounts of used volume space as illustrated.

devid	Size	Used	Avail	Name
0	996147200	340338912	655808288	lvol_eva
1	996147200	21090736	975056464	lvol_xp
2	996147200	242525544	753621656	lvol_xpeva
3	99614720	21275984	78338736	lvol_xp1
4	99614720	21352848	78261872	lvol_xp2
5	99614720	21365296	78249424	lvol_xp3
6	99614720	21375992	78238728	lvol_xp4

Schedule policies for ongoing file migrations

A VxFS MVFS enforces the new file allocation part of its active file placement policy when files are created. Existing files are not relocated automatically, however. To relocate existing files, the active policy must be enforced by an authorized administrator using the `fsppadm enforce` command.

File placement policy enforcement is typically scheduled as a cron job to run at regular intervals (daily, for example) to keep actual file locations in reasonable accord with the administrative intent for the file system as expressed in its active file placement policy. To minimize the effect of file relocation on application performance, the command should be scheduled to run at times when application activity against the file system is low. The frequency with which it should be scheduled depends on the type of activity a file system experiences over time. File systems in which events that can make relocation desirable are frequent should have policy enforcement scheduled more frequently than less dynamic file systems.

An alternative to cron as scheduling mechanism for file placement policy enforcement is the inbuilt scheduler provided in the database extension package of Storage Foundation called Storage Foundation for Database

Resize volumes for LUN reclamation

While this paper does not include specific steps for reclamation of storage resources, it is recommended that any implementation of a tiered storage strategy be followed with steps to reclaim and repurpose critical storage resources.

In many instances, retrofitting DST to existing file systems and relocating inactive files to low-cost storage makes it possible to reclaim premium-quality storage capacity for other uses. If a premium volume is made up of disks whose capacity is not shared with other volumes, it can be shrunk, and the freed capacity can be made available for other purposes.

Standard VxVM commands can be used to resize and relay out existing volumes to return the disks to free pool. Use `fsvoladm list <filesystem>` to determine the used versus available space per volume. Use `vxresize -F vxfs -g <diskgroup> <volume> <size>` to resize (shrink) the volume by size. Depending on the layout of the volume (concatenated, striped, and so on), the resize operation will free the subdisks/LUNs directly or reduce the sizes of the existing subdisks/LUNs. In the latter case, a "volume relay out" operation can be used to convert the freed space in the subdisks to create "reusable" subdisks. The "reusable" subdisks can be further removed from the disk group to truly "free" the subdisk and be able to reuse the space for practically any other purpose/application.

Conclusion

By combining their technologies and expertise, HP and Symantec provide customers with the business value and technical answers to help solve their storage management needs with tiered storage. IT managers can take advantage of:

- Improved storage utilization across heterogeneous operating systems and storage arrays
- Higher performance through optimized usage of storage assets
- Operational efficiencies through automated and streamlined data migration capabilities

This paper presents a concrete implementation of a tiered storage solution using the best-of-breed components from both Symantec and HP. The paper outlines how to set up an MVFS, defining DST policies, and enforcing the policies in an HP storage environment. These actions enable IT managers to relocate data based on age and type from tier 1 HP XP storage to tier 2 HP XP External Storage and tier 3 HP EVA storage. The freed up tier 1 HP XP storage can be reclaimed and repurposed for consumption by other applications. Further, the paper illustrates how to improve performance by distributing the I/O load across spindles using a Load Balanced File System to evenly distribute file extents across volumes in a tier.

For more information

- HP and Symantec Alliance
www.hp.com/solutions/symantec
- HP StorageWorks
www.hp.com/go/storageworks
- HP StorageWorks XP24000 Disk Array
www.hp.com/go/xp
- ESG Analyst White Paper on DST
http://www.symantec.com/content/en/us/about/media/industryanalysts/esg_brief_symc_dst_sep_06.pdf
- Symantec Yellow Book: Using Dynamic Storage Tiering
<http://www.symantec.com/enterprise/yellowbooks>
- XML Guide for Dynamic Storage Tiering
http://www4.symantec.com/van/articles/DST_XML_VAN_mcl.jsp
- Veritas™ File System Administrator's Guide HP-UX 5.0
http://ftp.support.veritas.com/pub/support/products/FileSystem_UNIX/283704.pdf
- Veritas™ Volume Manager Administrator's Guide HP-UX 5.0
http://ftp.support.veritas.com/pub/support/products/VolumeManager_UNIX/283742.pdf

© 2007 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Microsoft and Windows are U.S. registered trademarks of Microsoft Corporation. Oracle is a registered U.S. trademark of Oracle Corporation, Redwood City, California. Itanium is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

4AA1-2792ENW, June 2007

