WHITE PAPER: TECHNICAL

symantec.
Confidence in the connected world.

Using Solid State Drive Storage with Veritas Storage Foundation

Anindya Banerjee, Principal Software Engineer
Murthy Mamidi, Advisory Engineer
Shashikumar Paul, Development Manager
Niranjan Pendharkar, Distinguished Engineer
Michael Wahl, Sr. Principal Software Engineer
Oscar Wahlberg, Sr. Principal Technical Product Manager

Storage and Availability Management Group

December, 2009

Symantec Technical White Paper

Symantec Technical White Paper

Using Solid State Drive Storage with Veritas Storage Foundation

Table of Contents

Using Solid State Drive Storage with Veritas Storage Foundation

Executive Summary

The performance gap between computing devices and storage devices has been growing for several years. Currently, CPU and DRAM operations are measured in nanoseconds while hard disk drive (HDD) operations are measured in milliseconds – that is several orders of magnitude difference. Solid-state storage has been identified as a technology to fill this performance gap between computing and storage devices. Of particular interest is solid-state disk (SSD) technology which couples solid-state storage with hardware which provides a disk device interface through a physical host interface such as SATA, SAS, FC, and PCI-E.

While the performance benefits of solid-state storage make it extremely attractive for most high-end enterprise SANs, deploying and efficiently managing mixed solid state disk and regular hard disk storage environments comes with a unique set of challenges:

- **Visibility**: It is increasingly common to offset the cost of solid state disks by replacing a given amount of FC drives with a mix of SSD and SATA drives. Efficiently managing multiple types of storage devices provisioned to a host at minimum requires being able to differentiate between them. If all storage types look identical to the system administrator, the risk of errors during provisioning and routine management operations dramatically increases. Given the big differences in cost and performance between solid state disks and SATA disks, confusing the two can be catastrophic to applications.
- **Manageability**: Day to day management of the storage consumed by a particular host involves monitoring utilization, adding storage, removing storage, replacing storage, etc. Doing any of these operations in a mixed storage environment is complex. Having the right level of visibility on the host helps minimize errors. Automating these operations so that growing a host volume based on solid state disks automatically happens on solid state storage only (and never on hard disks) and vice versa is the only way to provide 100% protection against allocation errors.
- **Data placement policies**: The reality is that very few applications deserve to have all their data on SSD. On the other hand, most applications can benefit greatly from having a subset of their data on SSD. So it makes sense for users to be selective in the type of data that should be placed on solid state disk and the type of data that should be left on regular hard disk. Those decisions have to be granular. Doing this at a full LUN level is too coarse.
- **Automation**: Placing the right data on SSD is important. However, in real production environments, the data that should be on SSD and not on HDD changes over time: today's hot data is tomorrow's cold data. Being able to simply place the right amount of data on the right type of storage is not enough. Users will need to be able to continuously optimize their mixed SSD and regular disk storage environment and ensure that data that is consuming SSD storage indeed deserves to be there based on its current needs.

Veritas Storage Foundation provides advanced host based storage management functionality that allows system administrators to effectively tackle all of these challenges. This paper walks through these benefits and provides best practices as well as detailed 'how to' information regarding configuring and using Veritas Storage Foundation in a storage environment containing a mixture of SSD and HDD storage. The operating systems covered are AIX, HP-UX, Linux and Solaris. It is assumed that the reader has some familiarity with Veritas Storage Foundation.

SSD Overview

SSD Performance Considerations

Solid-state storage is a nonvolatile storage medium that employs integrated circuits rather than magnetic or optical media. There are two basic categories of solid-state storage: RAM-based and flash-based. RAM-based solid-state storage uses DRAM or SRAM chips similar to those used as volatile memory in a computer system. Flash-based storage is a collection of flash memory chips; there are several ways of organizing the cells into a flash memory chip. Most of the solid-state disks (SSDs) available today are based on NAND (From the "Not And" logical operation used within integrated circuits) flash technology because of the technology's overall price-performance ratios in comparison to other technologies. Therefore the rest of this paper will focus on NAND flash-based SSD.

Flash memory has the concept of an erased cell. An erased cell represents a '1' state. A cell programmed with the appropriate charge represents a '0' state. An important point in this regards is that changing the cell from a '0' to a '1' state (often referred to as *erasing* the cell) is different than changing the cell from a '1' to a '0' state (referred to as *writing* the cell). Because of the semiconductor properties involved in flash memory, writes (and erasures) are slower than reads. In NAND flash, cells are organized into pages which are the smallest unit that is readable and writable. In the current generation of SSD's 4KB pages are the norm. These pages are grouped together into blocks. Currently, it's common to have 128 pages (512KB) or more (e.g., 4MB) in a block. These blocks are the smallest structure that can be erased in a NAND flash. Erasing a block can therefore be a more costly operation than a normal data write.

A flash-based SSD is normally comprised of several integrated circuits (IC). From a performance perspective, these ICs can be accessed in parallel. So, if each IC can provide 20 MB/sec of read bandwidth and an SSD has 10 ICs, the SSD can provide about 200 MB/sec of read bandwidth. Write performance, however, is more problematic. Normally, an entire page must be written to a completely erased area. So, if less than a page is written, the entire page needs to be copied from flash to DRAM memory, the new bits of data must be inserted, and the entire page written back to the flash memory. To make matters worse, the copying may need to involve an entire (erase) block depending on the level of remapping (page versus block) being maintained by the SSD. Also, once a (physical) page has been written, it can't be overwritten; it must be erased first before it can be written to again. Because a flash-based SSD can have write performance that is much worse than its read performance, the overall performance mix with reads and writes can be an area for confusion. If doing pure reads is about 10 times faster than doing pure writes, it might be expected that a 50-50 mix of reads and writes would be 5 times faster than doing all writes; however, that is incorrect -- it will be less than twice as fast. Note that even though write performance of an SSD does not match its read performance, it is still orders of magnitude better than an HDD.

SSD Reliability Concerns

There are a couple of reliability issues with flash memory that impact the design and operation of flash-based SSDs. One of particular interest is that flash memory cells have a finite lifecycle, meaning each cell can be written/erased a limited number of times before the cell wears outs and becomes unprogrammable. This is also referred to as the *write endurance* of the flash memory. It is common for cells to wear out between 10,000 to 1 million erase/write cycles. Even at 1 million cycles, commonly used parts of a file system and other software will easily exceed that many cycles within a relatively short amount of time. So, in order to maximize the life-time of a flashed-based SSD, the flash memory cells need to wear out at roughly the same pace. This can be achieved by moving the contents of erase units around. This technique is called *wear-leveling*. In its simplest form, a controller within the SSD runs a wear leveling algorithm that ensures writes are evenly distributed amongst all blocks within a flash chip and across all the flash chips in the SSD. The SSD must maintain a (persistent) logical-to-physical block mapping of its storage along with the wear-leveling data. This logical-to-physical block mapping also allows the SSD to get better write performance since "out-of-place" writes can be performed to an already erased block. However, if the SSD were to "fill up", write performance would go down drastically since every write would require an erase. The approach SSD vendors use to avoid filling up the SSD is to under provision the drive. For example, an SSD with 80GB of flash memory might only advertise 73GB of space available to the end user. While an SSD does not suffer the mechanical failures of HDD's they can fail. Users requiring high availability and reliability will need external forms of redundancy for SSDs, similar to HDDs.

SSD versus HDD

The following lists some of the advantages of a flash-based SSD over an HDD:

- Faster random access for reading. Since an SSD has no read/write head to move, it has access times that are much lower than an HDD.
- SSD access times are independent of the physical location of the data thus file fragmentation has almost no impact on performance.
- SSD storage tend to support more diverse operating environment characteristics:
    - Lower power consumption
    - Minimized heat production
    - Ability to endure more shock and vibration
    - Operates over a larger range of operating temperatures.

The following lists some of the advantages of an HDD over a flash-based SSD:

- HDD prices are currently considerably lower per gigabyte
- HDD capacities are currently considerably higher
- The (random) write performance of HDDs is typically closer to the read performance
- HDD's may demonstrate a longer life cycle as SSD's have limited write endurance and require wear leveling

Because the cost per bit of SSDs is much higher than HDDs, SSDs are still not cost effective for bulk data storage for all classes of data. Clearly their deployment will likely be focused where the added cost can be justified, such as the most performance sensitive workloads, or where other benefits justify the increased cost. SSDs will therefore be most useful in environments where a subset of data or metadata would benefit greatly from very high IOPS and/or very low latency. Initially users who are currently using enterprise HDDs, but intentionally configure them with less 100% utilization in order to get the highest possible random access performance by minimizing head movements will benefit the most from enterprise SSDs at the least additional cost. As SSD costs come down, SSDs will become cost effective for an increasingly wide range of users, but again predominantly for random access workloads. Because magnetic drives (HDDs) are quite good at sequential workloads, the benefit of SSDs for those workloads is much less for their higher cost. Thus, SSDs will initially be used for specific applications, such as OLTP or paging/swap space, and they will be used as the "tier zero" storage within tiered storage systems.

Flash-based SSDs are starting to appear in the enterprise in a variety of forms:

- integrated into a storage array
- as a stand-alone external disk (or JBOD offering)
- integrated into the server (e.g., as an internal disk drive or PCI-E device)

Storage arrays will initially take advantage of SSDs by allowing them to be used with existing array enclosures and controllers, usually allowing a mixture of SSDs and HDDs within the same enclosure. Because of the extremely high random I/O performance of SSDs, an array designed around the performance of many HDDs may encounter internal bottlenecks earlier when using even a much smaller number of SSDs. Attempts to use more SSDs than the array controller can support may not translate into the expected performance gain, thereby wasting the additional SSDs and the cost paid for them. This necessitates software solutions that allow applications to optimally layout their data on a mixture of SSDs and HDDs. This paper will explain how the Veritas Storage Foundation product can be used to leverage SSDs in mixed storage configurations.

Veritas Storage Foundation Product Overview

The Veritas Storage Foundation suite of products, available from Symantec, delivers heterogeneous, OS-independent, host-based storage virtualization for all types of enterprise workloads. The component products of interest to this paper are the Veritas Volume Manager and the Veritas File System. Specific support for solid-state storage is available as of version 5.1 of the Storage Foundation products.

Veritas Volume Manager

The Veritas Volume Manager (VxVM) is a storage management subsystem that allows users to manage physical disks as logical devices called volumes. A VxVM volume appears to applications and the operating system as a physical device on which file systems, databases and other managed data objects can be configured. By supporting the Redundant Array of Independent Disks (RAID) model, VxVM can be configured to protect against disk and hardware failure, and to increase I/O throughput. Additionally, VxVM provides features that enhance fault tolerance and fast recovery from disk failures. VxVM overcomes physical restrictions imposed by hardware disk devices by providing a logical volume management layer. This allows volumes to span multiple disks. VxVM also has a dynamic multi-pathing module which achieves high availability and high performance by efficiently using multiple access paths to a device

Some of the features of VxVM that are pertinent to this paper are:

- VxVM contains a device discovery module which scans storage devices attached to a server and discovers critical device attributes, including device type (e.g. SSD or HDD), a unique identifier for the device, vendor and enclosure information, RAID level and device capabilities such as the device being a *replicated* device or a *thin* device.
- The disk group construct in VxVM allows multiple devices to be grouped together for storage pooling.
- VxVM's storage provisioning tool allows a user to create VxVM volumes that can span more than one device within a disk group. A volume can either be a concatenation of multiple devices or it can have a combination of RAID configurations to achieve desired performance and redundancy characteristics.
- Additionally, a volume:
    - Can be recovered in the event of device failures if it has appropriate redundancy.
    - Can have its performance and redundancy characteristics modified online whenever necessary.
    - Can be configured to have one or more snapshots (point in time images).
    - Can be configured to have its contents replicated to a remote site for disaster recovery.
- The volume set (VSET) construct in VxVM allows multiple volumes to be grouped together and utilized as a single unit.
- A storage cache construct that is used to store data for space-optimized snapshots. When a volume that has a space-optimized snapshot is written to, the old contents of the volume for that region are copied to the cache object to preserve point in time image consistency.
- Replication constructs, including a storage replication log, for use with Veritas Volume Replicator to replicate volumes across sites. An SRL is a regular VxVM volume that is used as a log for the replication of volumes using Veritas Volume Replicator. Refer to the Veritas Volume Replicator Administrator's Guide for more details on setting up and using replication.

For further details on the capabilities of VxVM and VVR, refer to the VxVM Administrators' Guide and Volume Replicator Administrator's Guide (available at **http://sfdoccentral.symantec.com**).

Veritas File System

A file system is a method for storing and organizing computer files and the data they contain to make it easy to find and access them. More formally, a file system is a set of abstract data types (such as metadata) that are implemented for the storage, hierarchical organization, manipulation, navigation, access, and retrieval of data. The Veritas File System (VxFS) was the first commercial journaling file system. With journaling, metadata changes are first written to a log (or journal) then to disk. Since changes do not need to be to be written in multiple places, throughput is much faster as the metadata is written asynchronously. VxFS is also an extent-based, intent logging file system. VxFS is designed for use in operating environments that require high performance and availability and deal with large amounts of data.

VxFS is particularly well suited for high performance database configurations. Veritas File System provides a number of built-in database accelerators allowing users to get performance equivalent (if not better) to that of raw storage, with all the management benefit of a file system. Examples are Quick I/O, Cached Quick I/O, Concurrent I/O (CIO), Oracle Disk Manager (ODM) and Cached Oracle Disk Manager (CODM).

VxFS supports a rich set of extended mount options to specify enhanced data integrity modes, enhanced performance modes, temporary file system modes, improved synchronous writes, and large file sizes to enable a wide range of applications to utilize VxFS in an optimal fashion. For example, backup and restore applications can leverage the storage checkpoint and snapshot features of VxFS to obtain a consistent point-in-time image of the file system. VxFS allows online reorganization and resizing of the file system. VxFS also provides a File Change Log feature to track changes to files and directories in a file system. The File Change Log can be used to improve the performance of applications such as backup products, web crawlers, search and indexing engines, and replication software that typically scan an entire file system searching for modified files since a previous scan.

VxFS provides support for multi-volume file systems when used in conjunction with the Veritas Volume Manager. With multi-volume support (MVS), a single file system can be created over multiple volumes, each volume having its own properties. For example, it is possible to place metadata on mirrored storage while placing file data on better-performing volume types such as RAID-1+0 (striped and mirrored). The MVS feature also allows file systems to reside on different classes of devices, so a file system can be comprised of both inexpensive disks as well as disks from expensive arrays. Via the MVS administrative interface, a user can control which data goes on which volume types.

The Veritas Storage Foundation Cluster File System (CFS) allows clustered severs to mount and use a file system simultaneously as if all applications using the file system were running on the same server. CFS uses a symmetric architecture in which all nodes in the cluster can simultaneously function as metadata severs. Applications access the user data in files directly from the server on which they are running. Each CFS node has its own intent log. File system operations, such as allocating or deleting files, can originate from any node in the cluster.

For further details on the capabilities of VxFS, refer to the VxFS Administrator's Guide (available at **http://sfdoccentral.symantec.com**).

Dynamic Storage Tiering

Dynamic Storage Tiering (DST) is a feature of VxFS that is of particular interest with respect to solid-state storage. DST is built on the aforementioned multi-volume support (MVS) technology. Using DST, the user can determine on which volume the files are located, which can improve application performance. Files can be dynamically relocated from one volume to another either by configuring policies for automatic relocation or by manually running file relocation commands. As such, DST solves the problem of matching large numbers of files to appropriate storage tiers without administrative intervention. DST can completely automate the relocation of any or all of the files in a file system between storage tiers according to a flexible range of administrator-defined policies. DST is a "set and forget" facility. Once an administrator has defined a file system's placement parameters and a relocation schedule, file movement is both automatic and transparent to applications.

When creating a DST file system, an administrator specifies:

- ***Storage***. The volumes that comprise each of the file system's storage tiers
- ***Tiering policy***. Optional specification of placement and relocation rules for classes of files

From that point on, DST automatically manages file locations based on I/O activity levels or other criteria specified in the placement rules, making proper file placement completely transparent to applications and administrators.

While other technologies for automating data movement between storage tiers exist in the industry, DST is unique in four respects:

- ***Configuration flexibility***. DST storage tier definitions are completely at the discretion of the administrator and provide complete storage heterogeneity. There is no artificial limit to the number of tiers that may be defined, nor to the number or type of volumes that may be assigned to a tier.
- ***Managed objects***. DST operates at the file level. DST policies act on files rather than volumes, placing the entities that correspond most closely to business objects on whichever type of storage is deemed most appropriate by the administrator.
- ***Policy flexibility***. DST includes a wide range of file placement and relocation options. While the most frequent application of DST is I/O activity-based relocation, file placement policies can be based on other criteria, such as location in the name space, size, or ownership.
- ***Application transparency***. DST file relocation is transparent to users and applications. DST moves files between storage tiers as indicated by the policy in effect, but leaves them at their original logical positions in the file system's name space. No adjustment of applications or scripts is necessary

Utilizing Solid State Drives with Storage Foundation

Let us explore how Veritas Storage Foundation and SSD's can be leveraged to provide a high performance storage environment. Given the high cost per gigabyte of Solid State Drives, there is a need to ensure that SSD storage is used in the most efficient way. To meet this need, Storage Foundation provides easy management of SSD storage that is connected to a server and will ensure that the performance characteristic of the SSD storage is leveraged automatically in the best possible way. The following sections describe the various capabilities offered by Storage Foundation to ensure optimal use of SSD storage.

Automated Discovery of SSD Attributes of Devices

VxVM's intelligent device discovery module has the ability to automatically determine the media type of devices from certain supported vendors. So, if a device is identified as an SSD, VxVM discovers this property automatically and flags the device as having a media type of SSD. In case a device connected to the server is not in the list of devices supported for automatic discovery of the device's media type, and the user knows that it is an SSD, the user can explicitly flag the devices as having a media type of SSD. This allows VxVM to apply all features related to SSD storage to these, manually flagged, devices as well.

Once a device has been flagged with a certain media type, all VxVM objects (such as volumes and plexes) that are created on the device derive their *mediatype* property from the device's *mediatype* property. Whenever an object (such as volume) consists of devices of different media types, the *mediatype* property of such objects is considered to have a *mixed* media type.

For details on displaying and manipulating the *mediatype* of devices and other VxVM objects, please see "Appendix A: Details on how to use Storage Foundation with SSD storage" for further details.

Storage Provisioning Using SSD storage

To ensure the best utilization of SSD storage, SSD and HDD devices should not get mixed together in unintended ways while provisioning storage for VxVM volumes. Inappropriate mixing of SSD and HDD devices within the same VxVM volume will result in suboptimal usage of the SSD storage. To address this issue, VxVM's storage provisioning tool allows the specification of the desired media type of the storage to use. For example, the VxVM storage provisioning tool provides the ability to create a volume using only SSD (or without using SSD) storage without having to specify a list of disks (or enclosures).

For details on using storage media types with the VxVM storage provisioning tool, please see "Appendix A: Details on how to use Storage Foundation with SSD storage" for further details.

Leveraging Performance Characteristics of SSD Storage with VxVM

Solid State Drive devices exhibit excellent performance on read requests. To benefit from this characteristic, while reading a volume that has one or more plexes (active data copies, aka. mirrors) on SSD storage, VxVM automatically gives read-preference to these SSD plexes. As a result, whenever a volume consists of a mix of SSD and HDD plexes, the read performance of a volume hugely benefits from SSD storage. When a volume contains multiple SSD plexes, VxVM intelligently utilizes all SSD plexes to get even better read performance.

Placing VxVM cache objects on SSD storage can improve performance of a volume that has space-optimized snapshots. A VxVM cache object is a storage cache that is used to store data for space-optimized snapshots. When a volume that has a space-optimized snapshot is written to, the old contents of the volume for that region are copied to the cache object to preserve point in time image consistency.

Placing Veritas Volume Replicator storage replication log (SRL) volumes on SSD storage can improve performance of volumes that are being replicated; it also improves the performance of replication itself. An SRL is a regular VxVM volume that is used as a log for the replication of volumes using Veritas Volume Replicator.

For more information on how to configure SSD storage for the various uses described in this section, please see "Appendix A: Details on how to use Storage Foundation with SSD storage" for further details.

Leveraging Performance Characteristics of SSD Storage with VxFS

As previously mentioned, VxFS provides support for multi-volume file systems. Given the high cost of SSD storage compared to HDD devices, the MVS feature of VxFS can be used to get optimum utilization of SSD storage. Most of the file system metadata is accessed in a random fashion; this metadata includes extents allocated to structural files, blocks reserved for the super block, the volume label, indirect extents, extents belonging to the File Change Log file, the history log file, extended attributes, directories, access control lists, and so on. SSD storage provides huge advantages over HDD devices for random I/O, especially reads. Hence, storing the file system metadata on SSD storage can improve the overall file system performance to a great extent.

With the MVS feature, VxFS differentiates between two types of volumes, one that can contain only data, referred to as *dataonly*, and the other which can contain metadata or data, referred to as *metadataok*. Here, data refers to direct extents of regular files, which contain user data, and named data streams in a file system. A user can run a command to specify if a volume is *dataonly* or *metadataok*. Using this command, the volumes residing on SSD storage can be flagged as *metadataok*.

Leveraging Performance Characteristics of SSD Storage with DST

Just as file system metadata can benefit from the performance advantages of SSD storage, so can user data. The DST feature of VxFS allows a user to easy manage the placement of user data on SSD storage in a multi-volume file system (MVFS), as well as the movement of data between SSD and HDD devices. VxFS provides an easy to use tool that facilitates DST deployment using SSD storage by providing multiple ways to configure VxVM volumes residing on SSD storage into a MVFS. By default, the tool uses one SSD based VxVM volume for metadata and the rest of the volumes for data only. The tool can determine the media type of the disks comprising the volumes automatically. The DST feature of VxFS has several features that allow a user to fully take advantage of SSD storage, including:

- Allowing relocations based on I/O statistics collected over a period as short as one hour. This is useful in the context of SSD storage since the activity levels of different files can change within a business day, thereby allowing different files to take advantage of the SSD storage at different times during the day.

- The ability to specify a preference for files with higher or lower temperatures (ratio of I/O activity to file size) to be relocated when a large number of files meet some defined threshold criteria. This is useful in the case of SSD storage where it is desirable to put the most active files on SSD storage, thereby allowing the most effective utilization of the limited and expensive SSD storage.

- The ability to specify I/O temperature thresholds as a ratio of per-file activity over a period of time to the average file system activity instead of absolute temperatures. This eases policy specification since the administrator does not have to arrive at absolute values for temperature thresholds, which would otherwise need experimentation and trail runs.

- Providing quick identification of hot and cold files. Since the population of hot and cold files could change within a business day, relocation scan performance to identify hot and cold files has to be very optimal.

For details on the tools for rapid deployment of DST and using DST to take advantage of SSD storage, please see ”Appendix A: Details on how to use Storage Foundation with SSD storage”.

Using DST to Leverage the Performance Characteristics of SSD Storage in Database Environments

A substantial improvement in database performance can be achieved by leveraging the DST features of Storage Foundation together with SSD storage. As different database files will have different operational and access patterns, utilizing DST together with a limited amount of SSD storage can significantly improve the overall database performance without having to place large portions of the database on SSD storage.

The characteristics of SSD storage are particularly well suit suited for the random access workload of most database applications. Data and index files with high I/O activity, transaction rollback storage and temporary files are ideal candidates for SSD storage in a database environment. The data and index files are part of the primary data repository of the database and in addition to these any temporary files used for sorting data in SQL queries are good candidates for SSD storage. Placing temporary files on SSD storage can have a remarkable effect on the performance of SQL operations.
In order to determine the ideal candidates for SSD storage it may be necessary to study the I/O statistics reports provided by the database.

For instance, if database I/O statistics reports indicate that the transaction log writes are a bottleneck, moving the transaction logs to SSD storage would boost performance. By creating a suitable the placement policy for DST a DBA can place hot files with high I/O access rate on SSD storage and derive significant performance benefits with a minimum amount of SSD storage. As access patterns change over time the DBA can use DST to relocate files in and out SSD storage without impacting the availability of the database.
The selection of data could be in terms of database objects such as table spaces, partitions etc. or file level I/O statistics provided by VxFS. Based on these criteria the DBA can choose the ideal candidates for SSD storage.
One could go further and also implement an ILM (Information Lifecycle Management) strategy by keeping hot or current information on SSD storage while moving cold data onto HDDs.

The database specific DST policy templates in “Appendix A: Details on how to use Storage Foundation with SSD storage” provide specific examples on how to apply some these principles.

SSD Performance Studies

As described in earlier sections, there are various unique performance characteristics of SSD storage that can be utilized in Storage Foundation products. This section details some of the performance studies that were done to determine how to best utilize SSD storage with respect to Storage Foundation metadata.

For detailed information on the benchmarks and testing methodology employed, please see "Appendix B: SSD Performance Study Test Configurations".

VxFS Test Scenarios

To investigate the benefits of using SSD storage with VxFS, various scenarios where the file system metadata was placed on SSD storage compared to HDD storage were explored. The benchmark used for this investigation was SPECsfs2008. The following was observed regarding the maximum I/O Operations per Second (IOPS) achieved with the benchmark when using SSD storage compared to using HDD devices:

- When the I/O response time for the devices containing the VxFS metadata were small (< 2 ms), marginal (< 4%) improvement was observed.
- When the I/O response time for the devices containing the VxFS metadata were large (> 5 ms), simulated by reducing the amount of available array cache, improvements up to 200% were observed.
- When all of the VxFS metadata was cached in host memory (cache hit ratio > 99%), no significant gain (<3%) was observed.
- When the host system faced significant memory pressure, simulated by reducing the available system memory, thus reducing the probability of finding the VxFS metadata cached in memory, improvements up to 12% were observed.

VxVM Test Scenarios

To explore the benefits of SSD storage with VxVM, the following scenarios were investigated:

A) Placing VxVM cache volumes on SSD

1) Impact on space-optimized (SO) snapshot overhead:
   Tests with TPC-C benchmark in an OLTP environment showed that
   a) When I/O response times for the LUNs comprising a cache volume is small (~1-1.5 ms), no noticeable improvement in application throughput was observed by utilizing SSD storage for the cache volume.
   b) When I/O response times for the LUNs comprising a cache volume is large (> 6 ms), simulated by switching off the array write cache associated with the LUNs, up to 2000% improvement was observed by utilizing SSD storage for the cache volume.

   Similar results were observed for tests with vxbench.

2) Performance of restore operations using a SO snapshot:
   Tests with TPC-C benchmark in an OLTP environment showed a considerable decrease in time needed to perform a restore operation when the SO snapshot's cache volume was placed on SSD. A performance gain of ~55% was observed for this scenario. This was consistently seen for various I/O response time scenarios ranging from 1 to 10 millisecond latency.

B) Placing VxVM DRL map on SSD

For small I/O response time environment (~1msec), it was observed that placing the dirty region log (DRL) maps on HDD LUNs was more effective than placing DRL map on SSD LUNs. However in environments where the I/O response times of the HDD LUNs housing the DRL maps were large (>5 ms), substantial performance improvements were observed by placing the DRL maps on SSD LUNs. Such an environment was simulated by switching off the array cache associated with the LUNs housing the DRL maps. Performance gains up to 500% were observed for write intensive environment (> 80% writes) while the gains reduced to less than 200% for read intensive environments (e.g., random read=70%, random write=30%).

With an OLTP workload generated by the TPC-C benchmark, no noticeable performance difference was observed when the DRL maps were placed on SSD LUNs compared to HDD LUNs.

Best Practices for using Storage Foundation with SSD storage

Based on the performance study results and the capabilities of the Storage Foundation product, the following constitute the best practices for utilizing SSD devices in place of HDD devices in disk arrays:

- When the host computer's system memory and/or the array cache memory is sufficient to mask the I/O accesses to the disk devices (thereby providing small I/O response times), there is little to no performance benefit to using SSD devices instead of HDD devices.
- When the host and array memory cannot mask the I/O accesses, significant performance gains can be had by using SSD storage. That is, in environments where the array cache is saturated (e.g., many hosts sharing a single array), the performance gains achieved by utilizing SSD storage can be quite noticeable.

  To take advantage of these gains with the Storage Foundation product:
  - Always ensure that the Solid State Drives have been discovered correctly or manually tagged with the correct media type.
  - Use the multi-volume support feature of VxFS to place the file system metadata on SSD storage while allowing user data files to reside on either SSD or HDD storage. For additional performance, use the dynamic storage tiering feature to set policies which allow the SSD storage to comprise a tier-0 storage level to contain (cache) the "hot" user files.
  - Use SSD storage for VxVM cache volumes to increase the performance of space-optimized snapshots.
  - Use SSD storage for VxVM DRL maps when the application environment is write intensive (>80% writes).
  - Use SSD storage for VVR SRL (Synchronous Replication Logs) to improve replication performance

Summary

This paper has provided a brief overview of the current state of SSD technology, an overview of Veritas Storage Foundation, and mechanisms for utilizing these technologies together. The performance gap between CPU and DRAM operations and hard disk drive operations have been growing for several years. Solid State storage has been identified as a technology to fill the performance gap between computing and storage devices. Of particular interest is Solid State Drives which couples solid state storage with hardware which provide a disk device interface through a physical host interface as SATA, SAS, FC and PCI-E.

The performance studies presented in this paper compare the performance of SSD devices against HDD devices as they are deployed in storage arrays. These studies highlight the scenarios where utilizing SSD storage yielded significant performance gains. In environments where the array cache is saturated (e.g., many hosts are sharing a single array), the performance gains achieved by utilizing SSD storage can be quite noticeable.

The results of these studies have been distilled into a set of best practices for leveraging SSD storage in storage arrays with Veritas Storage Foundation in order to achieve the most optimal storage environment for user applications:

- Using the multi-volume support feature of Veritas File System to place metadata on SSD storage increase performance and by using the Dynamic Storage Tiering capabilities to place “hot” data on SSD storage as a tier-0 (cache) can further improve performance.
- Using SSD storage for VxVM cache volumes increases the performance of space-optimized snapshots.
- Using SSD storage for VxVM DRL maps increases throughput when in environment that are write intensive (>80% writes).

While the performance benefits of solid-state storage make it extremely attractive for most high-end enterprise SANs, deploying and efficiently managing mixed solid state disk and regular hard disk storage environments comes with a unique set of challenges:

- **Visibility**: It is increasingly common to offset the cost of solid state disks by replacing a given amount of FC drives with a mix of SSD and SATA drives. Efficiently managing multiple types of storage devices provisioned to a host at minimum requires being able to differentiate between them. If all storage types look identical to the system administrator, the risk of errors during provisioning and routine management operations dramatically increases. Given the big differences in cost and performance between solid state disks and SATA disks, confusing the two can be catastrophic to applications.
- **Manageability**: Day to day management of the storage consumed by a particular host involves monitoring utilization, adding storage, removing storage, replacing storage, etc. Doing any of these operations in a mixed storage environment is complex. Having the right level of visibility on the host helps minimize errors. Automating these operations so that growing a host volume based on solid state disks automatically happens on solid state storage only (and never on hard disks) and vice versa is the only way to provide 100% protection against allocation errors.
- **Data placement policies**: The reality is that very few applications deserve to have all their data on SSD. On the other hand, most applications can benefit greatly from having a subset of their data on SSD. So it makes sense for users to be selective in the type of data that should be placed on solid state disk and the type of data that should be left on regular hard disk. Those decisions have to be granular. Doing this at a full LUN level is too coarse.
- **Automation**: Placing the right data on SSD is important. However, in real production environments, the data that should be on SSD and not on HDD changes over time: today’s hot data is tomorrow’s cold data. Being able to simply place the right amount of data on the right type of storage is not enough. Users will need to be able to continuously optimize their mixed

SSD and regular disk storage environment and ensure that data that is consuming SSD storage indeed deserves to be there based on its current needs.

Veritas Storage Foundation provides advanced host based storage management functionality that allows system administrators to effectively tackle all of these challenges. This paper has shown some of these benefits and provided a set of best practices as well as detailed 'how to' information regarding configuring and using Veritas Storage Foundation in a storage environment containing a mixture of SSD and HDD storage.
Veritas Storage Foundation is the only host based storage management solution to provide this visibly and manageability of mixed environments across all major Unix/Linux operating systems (AIX, HP-UX, Solaris as well as RedHat and SuSE Linux).

For scale out management across the data center, Veritas Storage Foundation Manager, a no-charge add-on to Veritas Storage Foundation, provides centralized application, server and storage management capabilities across a heterogeneous infrastructure. SFM offers a simple graphical interface that allows administrators to centrally monitor and manage thousands of Storage Foundation instances across complex data center environments rather than relying on a range of solutions that don't function across different platforms.

Appendix A: Details on how to use Storage Foundation with SSD storage

This section contains more detailed information on how to use SSD storage with the Storage Foundation products. It contains detailed examples of how to use the command line tools of Storage Foundation to display and administrate Storage Foundation objects in SSD storage environments.

Utilizing Disk Media Types with VxVM

Listing Known Solid State Drives

The intelligent device discovery module of VxVM has the ability to determine the media type of devices from certain supported vendors. Thus, an SSD device discovered in this manner will automatically be flagged, via a property called *mediatype*, as having a media type of SSD. An administrator can use the ***vxdisk*** command to display the attributes of VxVM disk objects. For example:

- To list all the disks that have a certain media type:

```
# vxdisk -o mediatype=ssd list
DEVICE       TYPE            DISK         GROUP        STATUS
sdc          auto:cdsdisk    sdc          testdg       online
sdf          auto:cdsdisk    sdc          testdg       online
```

- To list all of the disks and see which, if any, have an attribute of SSD (use the **–e** option):

```
# vxdisk -e list
DEVICE   TYPE     DISK     GROUP      STATUS    OS_NATIVE_NAME   ATTR
sdc      auto     disk1    testdg     online    sdm              std,ssd
sdd      auto     disk2    testdg     online    sdn              std
sde      auto     disk1    testdg     online    sdm              std
sdf      auto     disk2    testdg     online    sdn              std,ssd
```

- To list the VxVM media type tag for a disk as part of the listing for a disk (use the **–v** option):

```
# vxdisk -v list sdd
Device:    sdd
devicetag: sdd
type:      auto
...
Annotations:
tag      udid_asl=HITACHI%5FDF600F%5F77011207%5F0002
tag      vxmediatype=hdd
Multipathing information:
numpaths:   2
...
```

Adjusting Media Type for Unknown Solid State Drives

In case a device connected to a server is not in the list of devices supported for automatic discovery of their media type attribute, but the administrator knows that the device is an SSD device, the administrator can explicitly flag the device as being an SSD. VxVM automatically applies all features related to SSD devices to these manually flagged SSD devices as well. It is also possible to override the automatically discovered media type; the listing of the VxVM disk object will flag a *mismatch* as illustrated below. For example:

- To set the media type of a device whose media type was not automatically discovered to SSD (note that valid values of *mediatype* attribute supported in VxVM 5.1 are `ssd` and *`hdd`*):

```
# vxdisk set sdc mediatype=ssd
```

- To override the automatically discovered media type of a device requires the **–f** option:

```
# vxdisk –f set sdc mediatype=ssd
```

- To determine if the media type of a disk has been overridden (indicated by the *media_mismatch* flag):

```
# vxdisk list sdc

Device:    sdc
devicetag: sdc
type:      auto
...
info:      format=cdsdisk,privoffset=256,pubslice=3,privslice=3
flags:     online ready private autoconfig prcapable media_mismatch
...
```

Displaying VxVM Information Based on Media Type

Once the media type of a device has been flagged, all VxVM objects (such as volumes and plexes) that are created on the device derive their *mediatype* property from the device's *mediatype* property. Whenever an object (such as volume) consists of devices of different media types, the *mediatype* property of such objects is displayed as *mixed*.

- To list the media type of some particular disk media:

```
# vxprint –g testdg –F %mediatype disk1 disk2
ssd
hdd
```

or

```
# vxprint –g testdg –l disk1 disk2
Disk:      disk1
...
mediatype: ssd
Disk:      disk2
...
mediatype: hdd
```

- To list the media type of some particular plexes:

```
# vxprint -g testdg -F %mediatype tvol-02 tvol-01 tvol2-01
ssd
hdd
mixed
```

or

```
# vxprint -g testdg -l tvol-02 tvol-01 tvol2-01
Plex:      tvol-02
...
mediatype: ssd
Plex:      tvol-01
...
mediatype: hdd
Plex:      tvo12-01
...
mediatype: mixed
```

- To list the media type of some particular volumes:

```
# vxprint -g testdg -F %mediatype tvol1 tvol3 tvol
ssd
hdd
mixed
```

or

```
# vxprint -g testdg -l tvol1 tvol3 tvol
Volume:      tvol1
...
mediatype: ssd
Volume:      tvol3
...
mediatype: hdd
Volume:      tvo1
...
mediatype: mixed
```

- To list which disk media is on SSD storage:

```
# vxprint -g testdg -se '(sd_mediatype=ssd)' -F%disk
disk1
```

- To get a list of plexes which are utilizing only SSD storage for a particular volume (*tvol*):

```
# vxprint -g testdg -pe '(pl_mediatype=ssd && assoc.name="tvol")'
tvol-01
```

- To get a list of volumes that are utilizing only SSD storage:

```
# vxprint -g testdg -ve '(v_mediatype=ssd)'
tvol1
```

- To list all attributes including the media type of a volume (and similarly for other objects):

```
# vxprint -g testdg -m vol
...
Mediatype=ssd
...
```

Using Mediatype to Provision Storage

To ensure better utilization of SSD devices, the VxVM storage provisioning tool ***vxassist*** allows the administrator to indicate the type of media that is desired via the use of a mediatype:*value* specifier as part of a ***vxassist*** storage specification. This allows an administrator to easily create a volume using only SSD (or without using SSD) devices without having to specify a list of disks (and/or enclosures) on the command line. The following illustrates some of the expected usages of the *mediatype* specification:

- To create a volume using HDD storage:

```
# vxassist -g testdg make tvol 10g
or
# vxassist -g testdg make tvol 10g mediatype:hdd
```

- To create a volume using SSD storage:

```
# vxassist -g testdg make tvol 10g mediatype:ssd
```

- To create a volume not using SSD storage:

```
# vxassist -g testdg make tvol 10g !mediatype:ssd
```

- To add a mirror to an existing volume using SSD storage:

```
# vxassist -g testdg mirror tvol mediatype:ssd
```

- To change the stripe width of a volume and use SSD storage for the temporary storage space:

```
# vxassist -g testdg relayout tvol stwidth=<newsz> tmpalloc=mediatype:ssd
```

Note: some other VxVM commands directly pass storage specifications through to ***vxassist***. The following illustrates some of the expected usages of the **mediatype** specification with these commands:

- To prepare a volume for snapshot operations by creating a DCO volume on SSD storage:

```
# vxsnap -g testdg prepare tvol alloc=mediatype:ssd
```

- To grow a cache volume using SSD storage:

```
# vxcache -g testdg growcacheby cobj 10g alloc=mediatype:ssd
```

Placing VxVM cache volumes on SSD devices can improve the performance of a volume that has space optimized instant snapshots. The following illustrates some of the expected usages:

- To create a cache volume using SSD storage:

```
# vxassist -g testdg make cachevol 1g layout=mirror init=active mediatype:ssd
or
# vxassist -g testdg make cachevol 1g layout=mirror init=active ssd_dev1 ssd_dev2
```

- To create a cache object using a cache volume:

```
# vxmake -g testdg cache cobjmydg cachevolname=cachevol regionsize=64k autogrow=on
```

- To create a spaced optimized snapshot:

```
# vxsnap -g testdg make source=vol/newvol=snapvol/cache=cobjmydg
```

- To create a spaced optimized snapshot on a dedicated cache object using SSD storage:

```
# vxsnap -g mydg make source=myvol/new=snapvol/cachesize=1g/autogrow=yes
  alloc=mediatype:ssd
```

  The above command creates a space optimized instant snapshot *snapvol* of volume *myvol* that has a cache object of size 1GB with the cache *autogrow* attribute as *yes*. The *mediatype* attribute ensures that the cache volume is created on SSD devices.

Deploying DST Using mkdstfs

VxFS provides a simple data placement tool, ***mkdstfs***, which facilitates easy DST deployment. The syntax for invoking the tool is:

```
mkdstfs [-f] [-h] [-M mkfsopts] [-m mountopts] [-v] <mount> <dg> <vset> <vol>[:<placement_class>] ...

    -f    Create DST file system even if no SSD or mixed volumes are specified
    -h    Prints this usage
    -M    mkfs options for VxFS
    -m    mount options for VxFS
    -v    Verbose option prints all the commands executed
```

- After the options, a mount point (<*mount*>), a disk group (<*dg*>), a volume set (<*vset*>), and at least one volume (<*vol*>) must be passed, in that order. The VxVM disk group must already be imported. The volume set will be created by the tool if it does not already exist in the disk group. The specified volumes must already exist in the specified disk group.
- By default, the placement class of each of the specified volumes will be identical to the media type of the volume. Currently, the media types can be 'ssd', 'hdd', or 'mixed'. If it is not possible to detect a volume's media type or if it is desired to place a volume in a different placement class than its media type, the placement class can be specified on the command-line.
- If the volume set already exists, the specified volumes will be added to it.
- If the volume set does not already exist:
    - the volume set will be created in the specified disk group
    - if one or more ssd volumes are specified, the first ssd volume will become the first volume in the volume set
    - if no ssd volumes are specified but one or more mixed volumes are specified, the first mixed volume will become the first volume in the volume set
    - if an ssd or a mixed volume is not specified, the operation will fail unless it is forced to proceed by using the **-f** option
    - the first volume in the volume set will be marked as *metadataok* and additional volumes will be marked as *dataonly*
    - a VxFS file system will be created on the volume set
    - a placement policy is set up for the file system
- If the volume already exists, the utility will add the specified volumes to a pre-existing file system based on the same rules.

The following example shows the verbose output produced for creating a new volume set (*ssdvset*), and file system, using a volume on HDD storage (*hddvol*) and another on SSD storage (*ssdvol*):

```
# mkdstfs -v /dst ssdg ssdvset hddvol ssdvol
/opt/VRTS/bin/vxvset -g ssdg list ssdvset 2>/dev/null|/bin/grep -v '^VOLUME'
/opt/VRTS/bin/vxprint -g ssdg ssdvset >/dev/null 2>&1
/opt/VRTS/bin/mkfs -m /dev/vx/rdsk/ssdg/ssdvset 2>/dev/null
/opt/VRTS/bin/mkfs -m /dev/vx/rdsk/ssdg/hddvol 2>/dev/null
/opt/VRTS/bin/mkfs -m /dev/vx/rdsk/ssdg/ssdvol 2>/dev/null
/opt/VRTS/bin/vxprint -g ssdg -F \%mediatype hddvol
/opt/VRTS/bin/vxprint -g ssdg -F \%mediatype ssdvol
/opt/VRTS/bin/vxvset -g ssdg make ssdvset ssdvol
/opt/VRTS/bin/vxvset -g ssdg addvol ssdvset hddvol
    version 7 layout
    524288 sectors, 262144 blocks of size 1024, log size 1024 blocks
    largefiles supported
/opt/VRTS/bin/mkfs /dev/vx/rdsk/ssdg/ssdvset
mount
/bin/mkdir -p /dst
/opt/VRTS/bin/mount /dev/vx/dsk/ssdg/ssdvset /dst
/opt/VRTS/bin/vxassist -g ssdg settag ssdvol vxfs.placement_class.ssdmetadata
/opt/VRTS/bin/vxassist -g ssdg settag hddvol vxfs.placement_class.hdd
/opt/VRTS/bin/fsppadm assign /dst /tmp/mkdstfs.xml
```

The following example shows adding another volume, *hddvol2*, to the above volume set (file system):

```
# mkdstfs -f -v /dst ssdg ssdvset hddvol2
/opt/VRTS/bin/vxvset -g ssdg list ssdvset 2>/dev/null|/bin/grep -v '^VOLUME'
/opt/VRTS/bin/mkfs -m /dev/vx/rdsk/ssdg/ssdvset 2>/dev/null
/opt/VRTS/bin/mkfs -m /dev/vx/rdsk/ssdg/hddvol2 2>/dev/null
/opt/VRTS/bin/vxprint -g ssdg -F \%mediatype hddvol2
/opt/VRTS/bin/vxvset -g ssdg addvol ssdvset hddvol2
mount
/opt/VRTS/bin/vxprint -g ssdg -F \%len  hddvol2
/opt/VRTS/bin/fsvoladm add -f dataonly /dst hddvol2 262144
/opt/VRTS/bin/vxprint -g ssdg -F \%mediatype ssdvol
/opt/VRTS/bin/vxassist -g ssdg settag ssdvol vxfs.placement_class.ssdmetadata
/opt/VRTS/bin/vxprint -g ssdg -F \%mediatype hddvol
/opt/VRTS/bin/vxassist -g ssdg settag hddvol vxfs.placement_class.hdd
/opt/VRTS/bin/vxassist -g ssdg settag hddvol2 vxfs.placement_class.hdd
/opt/VRTS/bin/fsppadm assign /dst /tmp/mkdstfs.xml
```

The DST placement policy file created by the **mkdstfs** tool is as follows:

```
# cat /tmp/mkdstfs.xml
<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM "/opt/VRTSvxfs/etc/placement_policy.dtd">
<PLACEMENT_POLICY Version="5.1" Name="IOTEMP_Policy">
        <RULE Flags="data" Name="ssdfs_policy">
                <SELECT>
                        <PATTERN> * </PATTERN>
                </SELECT>
                <CREATE>
                        <ON>
                                <DESTINATION>
                                                <CLASS> hdd </CLASS>
                                </DESTINATION>
                                <DESTINATION>
                                                <CLASS> ssdmetadata </CLASS>
                                </DESTINATION>
                        </ON>
                </CREATE>
        </RULE>
</PLACEMENT_POLICY>
```

With this placement policy, all newly created files are placed in the “hdd” tier unless that tier is full. If the “hdd” tier is full, new files will be placed in the “ssdmetadata” tier. Thus, if all of the SSD-based volumes in the volume set are tagged with the “ssdmetadata” tier name, SSD storage will be used as the last resort for storing user files while the file system metadata will be confined to using SSD storage.

The above is the default policy that is generated by **mkdstfs**. The policy basically places all files on HDD storage when they are created and leaves them there. If fancier control over data placement and relocation is desired, section 8.3 documents the details of the XML grammar for creating more elaborate placement policies.

Relocating Files Using DST

An important concept in DST is that of temperature. Roughly, temperature is the I/O activity over a period of time. For example, the read I/O temperature of a given file is computed by the following formula:

```
Read IOTEMP = (number of bytes read from the file in a time period) /
              (length of the time period in hours * the size of the file in bytes)
```

Similarly, the average read I/O temperature of a given file is computed by the following formula:

```
Average Read IOTEMP = (number of bytes read from all active files in a time period) /
                      (length of the time period in hours *
                       the size of all of the active files in bytes)
```

The following sections illustrate how to utilize DST to relocate files between HDD and SSD devices (user selectable values are highlighted in bold-italics).

Hourly I/O Temperatures

The DST feature computes temperature values periodically. By default, the *<PERIOD>* element, for the *<IOTEMP>* and *<ACCESSTEMP>* criteria, is in days. With SSD devices, relocation decisions might need to happen within the day itself, based on the I/O activity measured over a few hours. As such, "hours" can be specified for the *units* attribute value within a *<PERIOD>* element. The following placement policy snippet gives an example of specifying a period of 4 hours.

```
<RELOCATE>
   <WHEN>
      <IOTEMP Type="nwbytes">
         <MIN Flags="gteq"> 2 </MIN>
            <PERIOD Units="hours"> 4 </PERIOD>
      </IOTEMP>
   </WHEN>
</RELOCATE>
```

Different *<PERIOD>* elements may be used in the *<IOTEMP>* and *<ACCESSTEMP>* criteria of different *<RELOCATE>* statements within the same policy. In case of an SSD storage tier, it may be desirable to relocate a file to SSD as soon as there is a marked pick up in the I/O activity for the file. However having brought the file into SSD, it may be beneficial to keep it on SSD as long as the activity remains high and thus avoid frequent thrashing. So once the file is on SSD, a longer time period may be desirable before deciding to move the file out of SSD compared to the length of time used earlier to decide to bring the file into SSD.

Quick identification of hot and cold files

When specifying "hours" for the *Units* attribute of a *<PERIOD>* element, DST must scan more frequently, which leads to a higher scan load on the host computers. So the following conflicting requirements have to be satisfied simultaneously:

- Bring down the temperature collection windows to hourly levels when using SSD devices
- At the same time reduce the impact of more frequent scans on resources, such as CPU, I/O, and memory etc.
- And complete the scans quickly

To reduce the impact of frequent scans, the following two items can be done:

- Confine the scans to just the active files in the immediate PERIOD preceding the scan (by focusing only on the files which showed any activity in the FCL). Please refer to the **–C** option of the **fsppadm** command detailed below.
- Limit the scope of scanning to specific tiers, like the SSD tier. Please refer to the **–T** option of the **fsppadm** command.

With the first item, the I/O statistics collected in the FCL file since the last scan provide details on fewer files but cover files with higher activity. Thus active files are quickly identified. However, cold files cannot be detected with the help of the FCL, and they will remain where they are located. Cold files residing in the SSD tier can result in a lack of room for active files should they need to be moved into the SSD tier, thereby resulting in ineffective use of the SSD storage.

This leads to the second item. DST maintains a file location map that associates storage devices with the files residing on them. This map is updated during (a) DST's own file relocations, and (b) upon examination of the file system's FCL for changes made outside of DST's scope. The map is updated during DST's relocation scans and can also be updated anytime with the **fsppadm** command. With the help of the map, instead of scanning the full file system, a scan can be confined to files on the SSD tiers, besides the active files recorded in the FCL.

Using these two items together, the scan duration as well as its resource consumption can be substantially reduced, facilitating multiple DST scans within a day. The full file system scans can still be scheduled at a coarser frequency as before.

Recapping, the **fsppadm** command can be used as follows:

```
fsppadm enforce [–C]  [ -T .[ <class>. ] [<path> ... } ]
fsppadm query [–C]  [ -T { <class>. } ... ] [<path> ... } ]
fsppadm analyze [–C ] [ -T { <class>. } ... ]  [<path> ... } ]
```

The -C option will confine the scan to just the active files in the immediate PERIOD preceding the scan (i.e., files which have their I/O stats collected in the FCL). Thus this option will quickly identify the hot files. The -T option will confine the scan to a specified set of tiers, for instance SSD tiers. This option can be used to identify cold files living in SSD tiers but with no I/O activity registered in the FCL.

Preferential placement

A *Prefer* attribute can be specified for the *<IOTEMP>* and *<ACCESSTEMP>* criteria, and it can take one of two values: *low* or *high*. If *low* is specified, DST relocates files with a lower I/O temperature before relocating files with a higher I/O temperature, even though all such files may satisfy the threshold criteria. Similarly, and if *high* is specified, DST relocates the files with the higher I/O temperature before relocating the files with the lower I/O temperature. This is particularly useful for relocating to SSD devices, which are expensive and limited in size, allowing them to be used more effectively (i.e., for the most active files).

The following placement policy snippet gives an example of using the *Prefer* attribute. In this example, if there are a number of files whose I/O temperature is greater than the given minimum value, the files with the higher temperature are first subject to the *<RELOCATE>* operation before the files with the lower temperature.

```
<RELOCATE>
   <WHEN>
      <IOTEMP Type="nrbytes" Prefer="high">
         <MIN Flags="gteq"> 3.4 </MIN>
         <PERIOD Units="hours"> 6 </PERIOD>
      </IOTEMP>
   </WHEN>
</RELOCATE>
```

Average I/O activity

By default, an absolute value is specified for the I/O temperature when using the *<IOTEMP>* and *<ACCESSTEMP>* criteria. However arriving at such absolute numbers is often difficult and may require experimentation and observing data access patterns over a period of time. Moreover over time, this value may need to be changed due to changing access patterns; thereby, requiring the experimentation to be repeated. In order to ease constructing *<IOTEMP>* and *<ACCESSTEMP>* criteria, the *Average* attribute can be used. The *<PERIOD>* element in the *RELOCATE* criteria specifies the length of time immediately prior to the time of scan, during which the I/O statistics that are collected are used to process the files being scanned. Since I/O activity can change over time, it is desirable to collect the average I/O activity over a longer duration than the *<PERIOD>* itself. Additionally, an average temperature of the whole file system is computed. With the *Average* attribute, the temperature is calculated as a ratio of the per-file activity over the *<PERIOD>* to the overall file system activity over a longer period of time (default is 24 hours). Keeping this averaging period longer than the *<PERIOD>* value normalizes the effects of any spikes and lulls in the file activity. Such criteria are more intuitive and easier to specify than absolute values. The following snippet moves any files whose read *<IOTEMP>* over the last 6 hours is 1.5 times that of the average of all the active files (whose activity is still available in the VxFS File Change Log (FCL) at the time of the scan) in the whole file system over the last 24 hours:

```
<RELOCATE>
   <WHEN>
      <IOTEMP Type="nrbytes" Prefer="high" Average="*">
         <MIN Flags="gteq"> 1.5 </MIN>
         <PERIOD Units="hours"> 6 </PERIOD>
      </IOTEMP>
   </WHEN>
</RELOCATE>
```

It is to be noted that the meaning (i.e., the type) of *Average* will be determined by the context of the *Type* of *<IOTEMP>* or *<ACCESSTEMP>*. For example, if the *<IOTEMP>* Type is for reads, the *Average* is the average read *<IOTEMP>* while a Type pertaining to writes for *<ACCESSTEMP>* implies the average write *<ACCESSTEMP>* . The default for *Average* is a 24 hour average temperature, which is the total of all of the temperatures available up to the last 24 hours in the FCL, divided by the number of files for which such I/O statistics still exist in the FCL. The number of hours the average is based on can be overriden by specifying the *AveragePeriod* attribute in the *<PLACEMENT_POLICY>* element. The following example specifies an averaging period of 30 hours:

```
<PLACEMENT_POLICY Name="Policy1" Version="5.1" AveragePeriod="30">
```

Template DST Policy Files Provided with Storage Foundation

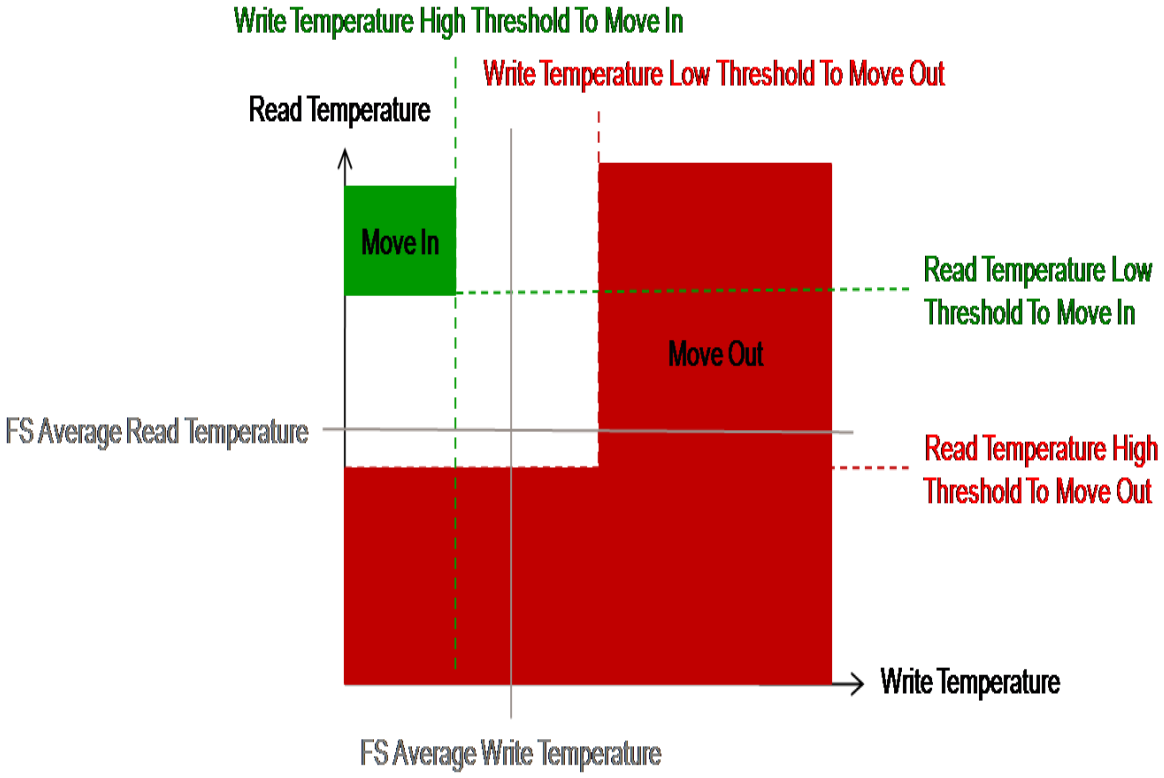
The Veritas Storage Foundation product includes template DST policy files. These template files contain more elaborate DST policies than the default policy file created by the **mkdstfs** utility.

DST Policy for Read Intensive Workloads

This policy template can be used with any file system in the context of SSD storage. The objectives of this placement policy, based on the ability of SSD devices to provide much higher throughput compared to other kinds of storage for randomly read files and the potential for a heavy write load over a long period of time to impact the longevity of the SSD storage, are:

- Move files out of SSD storage as soon as any pick up noticed in the write activity
- Move more intensely read files into SSD storage as soon as the read activity picks up while the write load remains below acceptable thresholds.
- After moving files into SSD storage, leave them there for sometime after a falloff in read activity before moving them out, to avoid thrashing.

The following figure illustrates the above objectives:



The files whose I/O activity falls in the green ("move in") area are good candidates for SSD placement. These files have lower write activity such that they have less impact on wear leveling. These files have intense read activity, which also makes the files ideal for placement on SSD storage since reads are faster from SSD storage.

The XML DST placement policy **Error! Reference source not found.** shown below can be used to achieve the above objectives. In this DST placement policy, new files are created on the SSD tier if space is available, or elsewhere if space is not available. When DST enforcement is performed, the files that are currently in the SSD tier whose write activity has increased above a threshold or whose read activity has fallen below a threshold over a given period of time are moved out of the SSD tier. The first two *RELOCATE*-statements capture this intent. Finally, the files whose read activity has intensified above a threshold and whose write activity does not exceed a threshold over the given *period of time* are moved into the SSD tier, while giving preference to files with higher read activity. This is captured by the third and final *RELOCATE-statement*.

ssd_read_intensive_template.xml:

```
<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM "/opt/VRTSvxfs/etc/placement_policy.dtd">
<PLACEMENT_POLICY Version="5.0" Name="ssd_read_intensive_template">
     <COMMENT>
          The policy has two tiers: 'ssdtier' and 'nonssdtier'. This policy can be used for any file system
          in the context of SSD storage.

          This policy tries to reduce SSD wear-out by placing read intensive files on SSD storage while
          evacuating write intensive files to HDD storage.
```

```
</COMMENT>
<RULE Flags="data" Name="all_files">
    <COMMENT>
          The first two RELOCATEs will do the evacuation out of SSD storage tier to create room for any
          relocation into the SSD tier by the third RELOCATE. The parameters that can be tuned are the
          values for PERIOD, MIN and/or MAX. The values for MIN and MAX are treated as multiples of
          average activity over past 24 hour period.
    </COMMENT>
    <SELECT>
          <PATTERN> * </PATTERN>
    </SELECT>
    <CREATE>
          <COMMENT>
               Create files on the ssd tier; failing that create them on another tier.
          </COMMENT>
          <ON>
               <DESTINATION Flags="any">
               <CLASS> ssdtier </CLASS>
          </DESTINATION>
          </ON>
    </CREATE>
    <RELOCATE>
           <COMMENT>
                Move the files out of the SSD tier if their last 3 hour write IOTEMP is more than 1.5
                times the last 24 hour average write IOTEMP. The PERIOD is purposely shorter than the
                other RELOCATEs because we want to move it out as soon as write activity starts peaking.
           </COMMENT>
           <FROM>
                <SOURCE>
                     <CLASS> ssdtier </CLASS>
                </SOURCE>
           </FROM>
           <TO>
                <DESTINATION>
                     <CLASS> nonssdtier </CLASS>
                </DESTINATION>
           </TO>
           <WHEN>
                <IOTEMP Type="nwbytes" Average="*">
                     <MIN Flags="gt"> 1.5 </MIN>
                     <PERIOD Units="hours"> 3 </PERIOD>
                </IOTEMP>
           </WHEN>
    </RELOCATE>
    <RELOCATE>
           <COMMENT>
                 Move files out of the SSD tier if their last 6 hour read IOTEMP is less than half the
                 last 24 hour average read IOTEMP. The PERIOD is longer because we want to observe for a
                 longer PERIOD once the file is brought in. This avoids moving the file out of SSD
                 storage too quickly.
           </COMMENT>
           <FROM>
                <SOURCE>
                     <CLASS> ssdtier </CLASS>
                </SOURCE>
           </FROM>
           <TO>
                <DESTINATION>
                     <CLASS> nonssdtier </CLASS>
                </DESTINATION>
           </TO>
           <WHEN>
                <IOTEMP Type="nrbytes" Average="*">
                     <MAX Flags="lt"> 0.5 </MAX>
                     <PERIOD Units="hours"> 6 </PERIOD>
                </IOTEMP>
           </WHEN>
    </RELOCATE>
    <RELOCATE>
```

```
                <COMMENT>
                     Move the files into the SSD tier if their last 3 hour read IOTEMP is more than or equal
                     to 1.5 times the last 24 hour average read IOTEMP AND their last 6 hour write IOTEMP is
                     less than half of the last 24 hour average write IOTEMP.
                </COMMENT>
                <TO>
                     <DESTINATION>
                          <CLASS> ssdtier </CLASS>
                     </DESTINATION>
                </TO>
                <WHEN>
                     <IOTEMP Type="nrbytes" Prefer="high" Average="*">
                          <MIN Flags="gteq"> 1.5 </MIN>
                          <PERIOD Units="hours"> 3 </PERIOD>
                     </IOTEMP>
                     <IOTEMP Type="nwbytes" Average="*">
                          <MAX Flags="lt"> 0.5 </MAX>
                          <PERIOD Units="hours"> 3 </PERIOD>
                     </IOTEMP>
                </WHEN>
          </RELOCATE>
     </RULE>
</PLACEMENT_POLICY>
```

DST Policy for Database Data Files and Indexes

The DST *ssd_dbdata_and_index_template.xml* policy template can be used for setting up databases in the context of SSD storage. This template assumes a common practice of keeping logs (undo and temp files) in one file system and data (data and index files) on a different file system. This template is targeted for the file system containing the data (data and index files) of the data base and can be used in conjunction with the *ssd_dblog_and_tmp_template.xml* policy template described in the next section.

The *ssd_dbdata_and_index_template.xml* policy has two tiers “ssdtier” and “nonssdtier”, and it has two *<RULE>* clauses; the first rule is for the database index files and is based on *<ACCESSTEMP>* while the second rules is for the rest of the data files and is based on *<IOTEMP>*.

The XML DST placement policy *ssd_dbdata_and_index_template.xml* is show below:

```
<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM "/opt/VRTSvxfs/etc/placement_policy.dtd">
<PLACEMENT_POLICY Version="5.1" Name="ssd_dbdata_and_index_template">
     <COMMENT>
          This template can be used for setting up databases in the context of SSD storage. This template
          assumes a common practice of keeping logs (undo and temp files) in one file system and data (data
          and index files) in a different file system.

          This template can be used for the file system hosting the data (data and index files) of the data
          base. The policy has two tiers 'ssdtier' and 'nonssdtier'

          There are two RULEs; the first one is for the index files and is based on ACCESSTEMP while the
          second one is for rest of the files and is based on IOTEMP. The first RULE could be eliminated,
          should one decide to place all the files including indices based on IOTEMP itself.

          This template can be used in conjunction with ssd_dblog_and_tmp_template.xml
     </COMMENT>
     <RULE Flags="data" Name="index_files">
          <COMMENT>
```

```
        The first RELOCATE will do the evacuation out of SSD storage to create room for any relocation
        into SSD storage by the second RELOCATE. The parameters that can be tuned are the values for
        PERIOD, MIN and/or MAX. The values for MIN and MAX are treated as multiples of average IO
        activity over the past 24 hour period.
    </COMMENT>
    <SELECT>
        <COMMENT>
            A tentative PATTERN is specified as a template filler that needs to be replaced by a more
            appropriate pattern (for instance *.idx) to SELECT the index files. Data bases use their
            own naming convention for index files. This template needs to be updated with those names
            in the PATTERN below. Multiple filename patterns can be used by having as many PATTERN
            elements as needed to specify all of the desired patterns.
        </COMMENT>
        <PATTERN> file_name_pattern_of_index_files </PATTERN>
    </SELECT>
    <CREATE>
        <COMMENT>
            Create index files on ssdtier; failing that create them on another tier.
        </COMMENT>
        <ON>
            <DESTINATION Flags="any">
                <CLASS> ssdtier </CLASS>
            </DESTINATION>
        </ON>
    </CREATE>
    <RELOCATE>
        <COMMENT>
            Move the indices out of SSD if their last 6 hour read-write ACCESSTEMP is less than 0.5
            times the last 24 hour average read-write ACCESSTEMP. The value of PERIOD is larger than
            the 2nd RELOCATE so that once the files have been brought into the ssdtier they do not
            get moved out too quickly.
        </COMMENT>
        <TO>
            <DESTINATION>
                <CLASS> nonssdtier </CLASS>
            </DESTINATION>
        </TO>
        <WHEN>
            <ACCESSTEMP Type="nrws" Average="*">
                <MAX Flags="lt"> 0.5 </MAX>
                <PERIOD Units="hours"> 6 </PERIOD>
            </ACCESSTEMP>
        </WHEN>
    </RELOCATE>
    <RELOCATE>
        <COMMENT>
            Move the indices into SSD if their last 3 hours read-write ACCESSTEMP is more than or
            equal to 1.5 times the last 24 hour average read-write IOTEMP. The PERIOD is smaller than
            the earlier RELOCATE. The idea is to move the files into SSD at the earliest detection of
            high ACCESS activity. If multiple indices satisfy the ACCESSTEMP thresholds, give
            preference to files with higher ACCESSTEMP.
        </COMMENT>
        <TO>
            <DESTINATION>
                <CLASS> ssdtier </CLASS>
            </DESTINATION>
        </TO>
        <WHEN>
            <ACCESSTEMP Type="nrws" Prefer="high" Average="*">
                <MIN Flags="gteq"> 1.5 </MIN>
                <PERIOD Units="hours"> 3 </PERIOD>
            </ACCESSTEMP>
        </WHEN>
    </RELOCATE>
</RULE>
<RULE Flags="data" Name="rest_of_the_files_including_data_files">
    <COMMENT>
```

```
            The first RELOCATE will do the evacuation out of SSD storage to create room for any relocation
            into the SSD storage by the second RELOCATE. The parameters that can be tuned are the values
            for PERIOD, MIN and/or MAX. The values for MIN and MAX are treated as multiples of average IO
            activity over past 24 hour period.
        </COMMENT>
        <SELECT>
            <PATTERN> * </PATTERN>
        </SELECT>
        <CREATE>
            <COMMENT>
                Create files on ssdtier; failing that, create them on another tier.
            </COMMENT>
            <ON>
                <DESTINATION Flags="any">
                    <CLASS> ssdtier </CLASS>
                </DESTINATION>
            </ON>
        </CREATE>
        <RELOCATE>
            <COMMENT>
                Move the files out of SSD if their last 6 hour read-write IOTEMP is less than 0.5 times
                the last 24 hour average read-write IOTEMP. The value of PERIOD is larger than the 2nd
                RELOCATE so that once the files have been brought into the ssdtier they do not get moved
                out too quickly.
            </COMMENT>
            <TO>
                <DESTINATION>
                    <CLASS> nonssdtier </CLASS>
                </DESTINATION>
            </TO>
            <WHEN>
                <IOTEMP Type="nrwbytes" Average="*">
                    <MAX Flags="lt"> 0.5 </MAX>
                    <PERIOD Units="hours"> 6 </PERIOD>
                </IOTEMP>
            </WHEN>
        </RELOCATE>
        <RELOCATE>
            <COMMENT>
                Move the files into SSD if thier last 3 hour read-write IOTEMP is more than or equal to
                1.5 times the last 24 hour average read-write IOTEMP. The PERIOD is smaller than the
                earlier RELOCATE. The idea is to move the files into SSD at the earliest detection of
                high IO activity. If multiple files satisfy the IOTEMP thresholds, give preference to
                files with higher IOTEMP.
            </COMMENT>
            <TO>
                <DESTINATION>
                    <CLASS> ssdtier </CLASS>
                </DESTINATION>
            </TO>
            <WHEN>
                <IOTEMP Type="nrwbytes" Prefer="high" Average="*">
                    <MIN Flags="gteq"> 1.5 </MIN>
                    <PERIOD Units="hours"> 3 </PERIOD>
                </IOTEMP>
            </WHEN>
        </RELOCATE>
    </RULE>
</PLACEMENT_POLICY>
```

DST Policy for Database Transaction Logs and Temporary Files

The DST *ssd_dblog_and_tmp_template.xml* policy template can be used for setting up databases in the context of SSD storage. This template assumes a common practice of keeping logs (undo and temp files) in one file system and data (data and index files) on a different file system. This template is targeted for the file system containing the logs (undo and temp files) of the data base and can be used in conjunction with the *ssd_dbdata_and_indexp_template.xml* policy template described in the previous section.

The *ssd_dblog_and_tmp_template.xml* policy has two tiers "ssdtier" and "nonssdtier". The policy has a single *<RULE>* whose *<SELECT>* clause should be modified to specify the names of the log/temp files and/or their directory locations. If control over the placement of additional files in the same file system is desired, another *<RULE>* can be added to select those files.

The XML DST placement policy *ssd_dblog_and_tmp_template.xml* is shown below:

```
<?xml version="1.0"?>
<!DOCTYPE PLACEMENT_POLICY SYSTEM "/opt/VRTSvxfs/etc/placement_policy.dtd">
<PLACEMENT_POLICY Version="5.1" Name="ssd_dblog_and_tmp_template">
     <COMMENT>
          This template can be used for setting up databases in the context of SSD storage. This template
          assumes a common practice of keeping logs (undo and temp files) in one file system and data (data
          and index files) on a different file system.

          This template can be used for the file system containing the logs (undo and temp files) of the data
          base. The policy has two tiers: 'ssdtier' and 'nonssdtier'.

          Appropriate modifications in the below SELECT clause is required to specify the names of logs/temp
          files and/or their DIRECTORY locations.

          Also note that this policy covers only the specific files in the SELECT. If the placement of any
          other files on the same file system is desired, another RULE can be added with SELECT's PATTERN of
          * to cover rest of the files.

          This template could be used in conjunction with ssd_dbdata_and_index_template.xml
     </COMMENT>
     <RULE Flags="data" Name="redo_or_undo_or_temp_files">
          <COMMENT>
               Note this RULE has no CREATE, because of which the frequently accessed files such as logs and
               temporary files permanently reside on ssdtier. This template could be used for existing
               databases which are coming under DST afresh or a new database deployment under DST, both with
               SSD storage.
          </COMMENT>
          <SELECT>
               <COMMENT>
                    A tentative PATTERN is specified as a template filler that needs to be replaced by the
                    appropriate pattern (like *.tmp) to SELECT the desired files. Databases use their own
                    naming convention for log files. This template needs to be updated with those names in
                    the PATTERN below. Multiple filename patterns can be used by having as many PATTERN
                    elements as needed to specify all of the desired patterns.
               </COMMENT>
               <PATTERN> file_name_pattern_of_relevant_files </PATTERN>
          </SELECT>
          <RELOCATE>
               <COMMENT>
                    Relocate all the SELECTed files unconditionally into SSD tiers FROM whereever they may
                    be, at the earliest opportunity.
               </COMMENT>
               <TO>
                    <DESTINATION>
                         <CLASS> ssdtier </CLASS>
                    </DESTINATION>
               </TO>
          </RELOCATE>
     </RULE>
</PLACEMENT_POLICY>
```

Appendix B: SSD Performance Study Test Configurations

This appendix contains detailed information on the various test configurations used to characterize the performance gains achieved using SSD devices with the Veritas Storaage Foundation product.

Test Setup 1

The following setup was mainly used for Veritas File System (VxFS) benchmarking.

- Server: SUNW,UltraSPARC-T2+
  - CPU: 64X1165 MHz
  - Memory: 16 GB RAM
  - OS: Solaris 10 U6
- Storage Foundation 5.0MP3 release
- Storage Array: EMC CX4-960
  - Cache: 16GB array Cache (5.6GB used by SP)
  - Disks: 85X HDD; 30XSSD
  - Each LUN exported as a RAID5 LUN consisting of (4+1) disks

Test Setup 2

The following setup was mainly used for Veritas Volume Manager (VxVM) benchmarking.

- Server: SUN SPARC Enterprise M4000
  - CPU: 8X 2.15 GHz
  - Memory: 16GB RAM
  - OS: Solaris 10 U6
- Storage Foundation 5.0MP3 release
- Storage Arrays:
  - Array I: EMC CX4-960
    - Cache: 16GB array Cache (5.6GB used by SP)
    - Disks: 30X HDD; 30XSSD
    - Each LUN exported as a RAID5 LUN consisting of (4+1) disks
  - Array II: Hitachi AMS500
    - Cache: 4GB array Cache (5.6GB used by SP)
    - Disks: 160 X HDD
    - Each LUN exported as a RAID5 LUN consisting of (4+1) disks

Benchmark Tools

During the performance study, multiple benchmarks were used. Below is brief description of each of them.

- **Vxbench**

  Vxbench is an I/O intensive benchmark. It is well suited to testing different types of I/O operations. It can generate read, write, random read, random write and read-write mixed random operations with varying I/O sizes and for different file sizes. It also supports various options for performing synchronous, asynchronous, direct, and memory-mapped I/O operations. It also supports simultaneous multi-process I/O on files and multi-threaded I/O on a file.

- **TPC-C**

  The TPC Benchmark C is an on-line transaction processing (OLTP) benchmark. TPC-C involves a mix of five concurrent transactions of different types and complexity either executed on-line or queued for deferred execution. The database is comprised of nine types of tables with a wide range of record and population sizes. TPC-C is measured in transactions per minute (tpmC). TPC-C simulates a complete computing environment where a population of users executes transactions against a database. The benchmark is centered on the principal activities (transactions) of an order-entry environment. These transactions include entering and delivering orders, recording payments, checking the status of orders, and monitoring the level of stock at warehouses. While the benchmark portrays the activity of a wholesale supplier, TPC-C is not limited to representing the activity of any particular business segment, but, rather represents any industry that must manage, sell, or distribute a product or service.

- **SPECsfs2008**

  SPECsfs is a licensed benchmark from the Standard Performance Evaluation Corporation (SPEC) that measures network file serving throughput (in ops/sec) and response time (in msec/op) and provides a standardized way of comparing performance across platforms. The objective of the benchmark is to generate a throughput response time curve and to identify the peak throughput and the response time at the peak. SPECsfs is also colloquially known as LADDIS. The current version of SPECsfs is SPECsfs2008; this version of the benchmark supports both NFS and CIFS, but not NFSv2 or the UDP transport.

  SPECsfs2008 generates a load on the NFS/CIFS server and measure the ops/sec and the msec/op. This benchmark is run on a group of clients and uses RPC in order to bypass the client's NFS/CIFS stack (to bypass the client-side differences and caching). SPECsfs is a synthetic benchmark consisting of operation percentages derived from use studies performed at many different sites with many types of use cases. SPECsfs puts a heavy load on the disk I/O and network I/O subsystems. SPECsfs uses a "Prime Client" as a director and numerous "Load Clients" to generate the benchmark's load. Ideally, the entire SPECsfs environment should be on a private network.

  There are five main phases of the SPECsfs benchmark; each one of these phases is repeated for each load point. The phases are: Mount, Initialization, Warm-up, Measurement, and Data Collection. Some of major operations which form part of the operation matrix for SPECsfs2008 are read, write, lookup, getattr, setattr, etc. A complete list of operations and their percentage in the total operation mix can be found at **http://www.spec.org/sfs2008/**

Testing Methodology

To simulate different types of workloads and I/O patterns, the various benchmarks, described earlier, were used. During each investigation, various system statistics were gathered such as I/O statistics, memory usage statistics and CPU usage information. Also care was taken to reach a stable state during benchmark tests before taking measurements. Wherever possible various logs and metadata (e.g., redo logs in TPC-C or intent logs in case of LADDIS) were placed on separate LUNs to avoid possible effects of I/O contention (e.g. with specsf2008 benchmark, a MVFS was used to separate metadata and user data). For measurements with TPC-C benchmark, the 2000W TPC-C kit from Oracle was used as the load generator. Various parameters were adjusted to reach maximum throughput as per set norms for TPC-C benchmarking.

For all performance tests described in this paper, SSD devices were used without array cache fronting them. For tests with HDD only, the amount of write cache was varied (between 0% and 100%) to simulate various latency scenarios observed in customer environments and to characterize performance gains due to SSD in these scenarios. Tests done with TEST SETUP II, as described above, had the EMC array cache (which housed both HDD and SSD LUNs) switched off to simulate a high I/O response time environments. That is, one where the array cache is saturated by 100's of hosts sharing a single array.

About Symantec

Symantec is a global leader in infrastructure software, enabling businesses and consumers to have confidence in a connected world. The company helps customers protect their infrastructure, information, and interactions by delivering software and services that address risks to security, availability, compliance, and performance. Headquartered in Mountain View, Calif., Symantec has operations in 40 countries. More information is available at www.symantec.com.

For specific country offices and contact numbers, please visit our Web site. For product information in the U.S., call toll-free 1 (800) 745 6054.

Symantec Corporation
World Headquarters
350 Ellis Street
Mountain View, CA 94043 USA
+1 (650) 527 8000
+1 (800) 721 3934
www.symantec.com