Storage System Bottlenecks and Their Solutions

A Whitepaper

Table of Contents

Introduction:

In today's demanding and competitive business world, it is critical for Information Technology (IT) departments to obtain optimum end-to-end performance from their companies' computing solutions (with end-to-end defined as involving the servers, the storage, and the applications in between). To achieve optimal performance, it is essential to configure each computing resource in the end-to-end solution to perform optimally to meet application workload needs and specific business performance requirements. To accomplish this, however, they must first understand one of the most critical pieces of the end-to-end computing solution puzzle: the storage system.

There are many storage vendors today, each offering a wide range of storage system options. This includes Data Center storage systems, Mid-Range storage, Workgroup Storage, NAS and Backup, (tape and software.) Though there are many choices available, this paper focuses on disk-based storage systems.

As noted earlier, storage system performance is a crucial piece of the IT solution, for it typically has a significant impact on the overall performance. Consequently, the specific purpose of this paper is to educate the reader concerning typical storage system performance criteria (e.g., input and output (I/O) rate, response time, and data rate) determined by the storage system workload. Additionally, it will discuss how to resolve storage system bottlenecks. A bottleneck is any limiting factor or component in the storage system which negatively impacts its performance; i.e., a storage system will only perform as fast as its slowest component.

Diminishing storage system bottlenecks should help improve storage system performance, which can assist in meeting, and hopefully exceeding, overall business performance requirements.

This paper includes the following topics:

- An overview of key components of a typical storage system (controller, cache, drive, connectivity, Host Bus Adaptor, pathing software, volume manager, etc.) and their roles in determining storage system performance.
- The impact of four major factors that limit storage system performance: queue depth, skew, cache hit rate, and workload.
- Actual deployment examples and their associated data.

- Various tools available to characterize workload (read/write ratio, block size, random vs. sequential) and performance against workload for particular storage system configurations.
- Approaches to resolving common storage performance problems.

Typical Storage System:

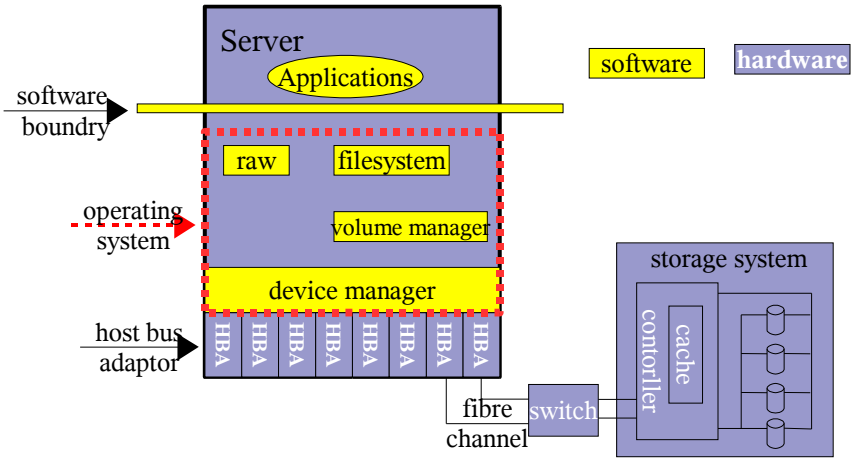

Illustration 1 Typical Storage System

A typical storage system is not just a set of disk drives – it is a hierarchical layering of software and hardware as depicted in Illustration 1 above.

The hierarchy starts at the server (hardware) level where the applications (software) reside. Applications must access data, and have a multitude of ways of doing so, which is enabled by the elements in the server Operating System (OS). These elements typically consist of a file system and/or raw device files.

A file system is a physical partition on a disk drive, which is created by an OS command. It includes metadata which the OS uses to identify where to write files and other important file information -- specifically, inodes, blocks, and super blocks. An additional benefit of a file system is that it contains a file system cache that helps speed up access to data, as the cache is employed to access frequently used data, thus bypassing the slower physical disk drive.

A raw device file is a physical partition on the disk drive that has no file system. However, the data on a raw device cannot be viewed or accessed by users as can be done with a file system. A raw device file

is most commonly implemented to improve I/O performance and generally used by databases such as Oracle© and Sybase©.

Files systems can become very large, and the best way to manage them is to implement a volume manager. A volume manager is an application that manages disks by combining disk sectors into "pools" of storage space typically referred to as volumes. These volumes can then be subdivided and/or combined into RAID (Redundant Array of Inexpensive Disks) sets for redundancy and performance. Additionally, the volume manager is typically utilized to help spread the data across many devices, which again, increases performance and redundancy.

No matter whether the application is using a file system or a raw device file, it must use an interpreter to access the storage system. This interpreter is called a device driver, which is a small program that controls a device and acts as a translator for programs that use a physical device. For a storage system, this device is called a Host Bus Adaptor (HBA). The HBA translates and transmits the I/O request from the application to the storage controller via a connection, which today is most commonly a combination of fibre channel interconnect and storage switch. The fibre interconnects and storage switch are the conduit used to move the I/O from the server to the storage. However, these are not discussed at length in this paper.

The storage controller resides inside the storage array and consists of a mixture of hardware (e.g. storage cache, cpu) and software, (RAID software, RAID algorithms). The controller is the interface into the actual physical disk drives. The controller contains a storage system cache, which is composed basically of memory chips that enable the storage system to access frequently used data directly from memory rather than from the slower mechanical disk drives.

These are the basic components, both hardware and software, that make up a typical storage system. The combined configuration of these hierarchical components can have a significant impact on the performance of the workload as it demands I/O from the storage controller, as this paper will discuss.

Typical Storage Performance Terminology:

- **Storage performance:** The amount of work completed per unit of time, which refers to the number of I/O requests (work) completed by the storage system per second. There are two measurements typically used to define the performance of a storage system:

- **Throughput:** The volume of requests that can be processed by the storage system per unit of time. For storage performance, this is usually measured in megabytes per second or MB/sec. Throughput can be described as how effectively the storage controller is able to supply the application with a lot of data. Generally, batch jobs and scientific applications such as seismic evaluation are most affected by throughput.
- **Response Time:** The number of I/Os processed per second by the storage system, (number of IOPS) is another metric used to define the performance of a storage system. This describes how efficiently the storage controller processes transactions. Generally, on-line transaction types of applications are most concerned with IOPS.
- **Workload:** The specific I/O sequence of reads and writes, cache hits and misses, and sequential or random activity that an application demands of a storage system.
- **Seek:** The initial operation a disk performs to place the read head on the right track of a disk drive.
- **Latency:** The secondary operation that occurs after the "seek", which is the time it takes for the data to reach the read/write head of a disk drive.
- **Transfer time:** The time it takes for data to be read from or written to the host after seek and latency.
- **Disk Drive Service Time:** The sum of seek + latency + transfer time.

Given this base of knowledge concerning a storage system, the next step is to find and resolve three typical storage performance issues experienced in the IT industry.

Typical Problems:

This paper discusses three typical poor storage performance scenarios found in today's IT industry. Real and lab data have been used to analyze these issues and demonstrate the steps to resolve them.

1. Scenario #1:

A company (for purposes of this paper will be called "Fast Growing, Inc.,") invented a great new product, which customers were buying at

an extraordinary rate. This overwhelming increase in demand for their product caused their storage and CPU requirements to increase exceedingly fast. Overnight, computing requirements had gone from a small four processor server with a small storage solution of JBODs, (Just-a-Bunch-Of-Disks with no controller or storage cache), to requiring at least 30 processors with racks and racks of storage with hardware controllers to manage the large I/O demand.

The IT department responded by purchasing a new server and a large amount of storage. Based on the performance numbers published by the storage vendor, they assumed that application performance was going to increase significantly in the new configuration.

When they received the new storage, they ran their standard benchmarks using the UNIX command “dd,” or “disk to disk copy,” to show the IT Vice President how efficiently the new storage system was going to perform. Much to their dismay, the storage system did not perform even close to what the vendor’s published numbers had promised and the server was reporting that the storage was 100% busy!

Understanding Storage System Performance Corner Cases:

Fast Growing, Inc.'s system administrators did not understand what the storage vendor’s published performance numbers truly represent. Storage vendors typically publish the best possible performance numbers for their storage systems. This means that the vendors benchmarks access data that primarily (or) only resides in the storage system cache; i.e., the I/O requests never actually need to access the slower mechanical disk drive. This can be described as a Corner Case for storage performance benchmarks. It does not represent real world application performance, but instead represents the potential pure cache and pure disk performance capabilities of a storage system.

In some situations a customer may need to understand the pure performance or raw performance capabilities of a storage system as portrayed by the Corner Case performance numbers. For example, a customer's workload may require that the storage system has a very fast cache algorithm, such as OLTP-type applications, or the workload may require very fast disk drives, as is the case with data streaming applications.

Corner Case Benchmarks, as depicted in Illustration 2 below, demonstrate the pure cache and pure disk performance capabilities.

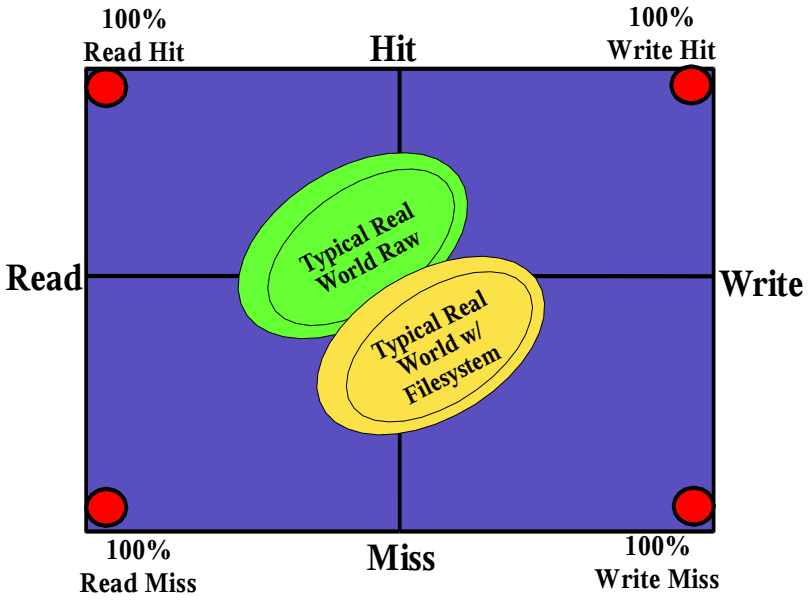

Illustration 2 - Corner Case Illustration

- **Corner Case Benchmark Results include:**

  - *Cache Performance:*
    The cache performance of a storage system is demonstrated by having all read and write I/O activity come from the storage controller cache; i.e., no slow mechanical disk drives are involved, which is represented as 100% Read Hit in Illustration 2 above. This demonstrates the fastest and somewhat unrealistic performance that can be expected of the storage system.
  - *Disk Performance:*
    The disk performance of a storage system is demonstrated by having all read and write activity come from the slower mechanical disk drives; i.e., no storage controller cache is involved, which is represented as 100% Write or Read Miss in Illustration 2 abve. This demonstrates the slowest but uncommon performance that can be expected of the storage system.

  - *Combination Performance:*
    Combination performance of a storage system is a combination of both cache and disk performance as described in the previous two paragraphs. This is the case for scenario number 1. Fast

Growing, Inc.'s application required access to both the disks and the cache. Therefore, relying on the vendor's published performance numbers can be - and most always is – misleading when compared to real application behavior.

Though the Corner Case Benchmark numbers represent pure cache and disk performance, it is quite uncommon that a real world application would perform at the same level as these numbers suggest; thus, it is important to use the storage vendor's published performance numbers as a guide to the vendor's storage systems' true performance.

- **Workload Characterization Tools:**

It is always an excellent and recommended strategy to truly understand the workload of the application(s) that will be using the new storage, and it is also extraordinarily import to understand workloads in current environments. This can assist the implementation of changes to improve current IT solution performance, as well as the configuration of new storage solutions.

Sun offers several powerful tools - for free - that can assist system administrators in understanding workloads in their current environments. By understanding their workloads, they will be able to incorporate the appropriate configuration changes to do the following:

- Improve current solution performance, by understanding if the workload is using only a portion of the configuration, (skewed or unbalanced) versus using all components evenly (balanced).
- Determine the source of potential performance issue(s).
- Plan upgrades to new systems based on knowledgeable configuration choices.
- Characterize storage workloads.

- **Sun StorEdge™ Workload Analysis Tool (SWAT)**

SWAT is a graphical user interface (GUI) JAVA application that collects, processes, and reports storage performance information about disk I/O workloads. In the Solaris operating system, the tool uses the Solaris trace normal form (TNF) utility. In the Microsoft Windows operating system, the tool uses the Tracelog utility. The information can be displayed at a variety of detail levels via many different graphs and charts to identify reoccurring performance problems. In the Solaris operating environment, the tool can also capture iostat/kstat level

storage performance data that can be used for a less detailed analysis than what can be obtained from I/O trace data. The information gathered is stored on the system where the data has been gathered.

This data, which includes transfer sizes, queue depths, cache hit rates, and skew across devices, helps administrators understand the behavior of the storage system workload(s). In turn, this provides valuable insight to how to best configure the storage system to meet the demands of the workload.

- **Sun StorEdge™ Vdbench (vdbench)**

Sun StorEdge™ vdbench, I/O driver, is a command line, Java-based, synthetic I/O driver that is portable across multiple platforms, and which can be used to replicate and approximate workload performance and throughput. vdbench replaces tools such as Veritas™ vxbench.

Vdbench can be used to:

- Validate software package installation and verify connections by generating I/O through those connections.
- Benchmark by varying variables in a workload.
- Play back a real customer workload captured using SWAT.
- Validate performance and throughput capabilities of a storage system in question.

Vdbench can control many aspects of a workload. The following is a sampling of options:

- I/O rate can be set to MAX, exponential, or fixed inter-arrival time.
- Percent read can control workload read to write ratio.
- Transfer size can be fixed or varied.
- Queue depth controls the number of active threads for a given workload.
- Percent read hit controls the number of reads taken from the storage system's controller cache versus the slower mechanical disk drives.
- Percent write hit controls the number of writes that overwrite previous writes in the storage system's controller cache, versus writing to the slower mechanical disk drives.
- Percent random determines the percentage of I/O requests

involving mechanical disk overheads known as “seek” and “latency”, and which therefore significantly system performance.

As mentioned previously, vdbench can “replay” a customer workload which has been previously captured using SWAT. Another benefit of vdbench is its ability to provide real-time feedback during benchmark execution. This enables the system under test to provide immediate benchmark progress. Vdbench displays the following information:

- I/O rate
- Response time
- Data rate
- Read-to-write ratio
- Maximum service time
- Service time standard deviation

Finally, vdbench provides other valuable data as text and HTML files which may be easily imported into spreadsheet packages, allowing graphing of virtually any parameter versus any other parameter. This permits further analysis and facilitates the sharing of performance information with others.

Supply vs. Demand:

Simply put, a storage system will usually supply the number of IOPS demanded of it. If only one I/O is requested of a storage system it will supply only one, even if it is capable of much more. It is impossible for a storage system to supply large amounts of data if a high demand for I/O was never requested of it.

The test situation in Scenario 1, as described previously, did not demonstrate the new storage system’s true potential because the demand for I/O was not very high. This was because the UNIX command 'dd' is single threaded; specifically, it asks for only one I/O at a time, defined as synchronous I/O. Given that 'dd' only requests one I/O at a time, the storage system will only supply one I/O at a time, even though it has the ability to supply much more.

Single Threaded versus Multi Threaded:

As mentioned above, the UNIX command, 'dd', is a single threaded application because it only requests one I/O at a time. A way to increase the demand for I/O of a storage system is to demand multiple I/Os from it simultaneously; this is commonly referred to as multi-

threading.

Multithreading enables the storage system to perform many I/Os simultaneously. As such, the storage system is allowed to supply more I/Os to the server, thus increasing its overall performance.

Illustration 3 below shows a theoretical example of an application requesting 8 chunks of data in single threaded format, versus requesting 8 chunks of data in multithreaded format. Single threaded theoretically takes 8 ms to complete the request, while the multi-threaded request theoretically might only 1 ms to complete.

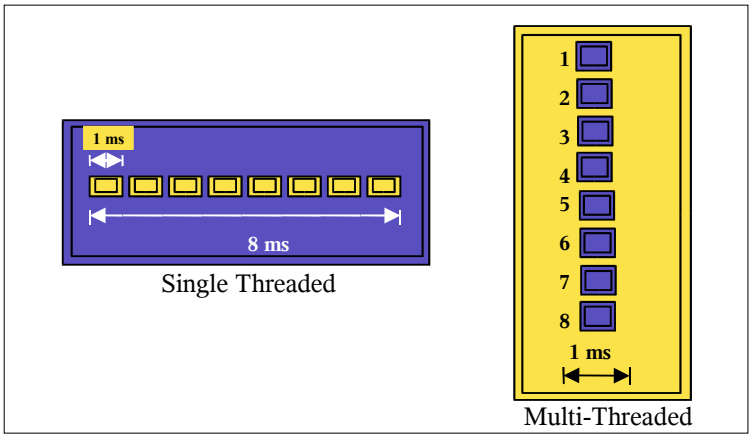

Illustration 3 Single Threaded vs. Multi-Threaded

To prove this theory and truly demonstrate the effects of multithreading I/O requests, an experiment was run using Sun's I/O generator tool, the Sun StorEdge™ vdbench and a Sun StorEdge™ 3510 (SE3510) with a 6 drive LUN configuration. There were several simple benchmarks run, starting with demanding only a single thread, (I/O), of the storage system. This was subsequently increased to 8 threads, and finally to 16 threads. As shown in Illustration 4 below, requesting a single thread produced 146 IOPs, at 8 threads it produced 707 IOPs, and at 16 threads it produced 967 IOPs – a 562% performance improvement! This demonstrates that increasing the thread count, within reason, can dramatically improve storage system performance.

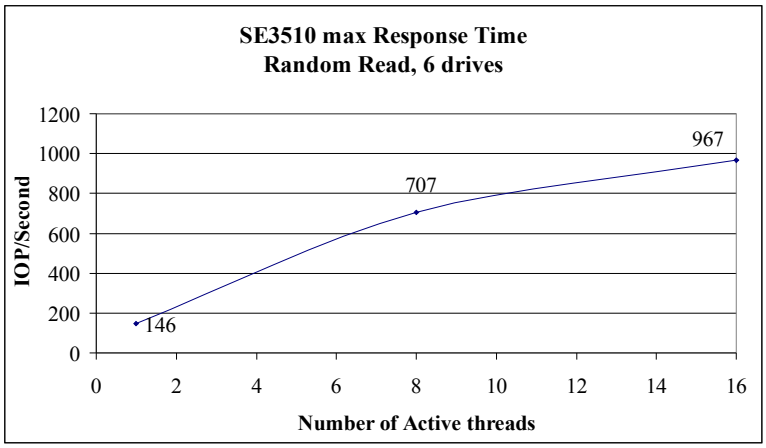

Illustration 4 Response Time vs Thread Count

NOTE: Though this experiment only requested a maximum of 16 threads, producing 967 IOPs, the SE3510 is capable of much more!

- **IO Queue Depth:**

If the application such as 'dd' is single threaded, the end-to-end system response time determines the volume of I/Os that are processed by the storage system per unit of time. While in the case of scenario #1, above, the performance of the storage system using the 'dd' command was an accurate picture for a single threaded application, it does not reflect the true throughput potential of the new storage system.

One of the ways that a host's demand for I/Os can be limited is the length of its I/O queue depth, where queue depth is the number of I/O requests waiting to be completed (also known as outstanding I/Os). There can be a significant increase in overall storage performance with a higher queue depth as opposed to zero queue depth. This is due to a storage system's ability to "hide" queued I/O activity from the application. The application in this case does not have to wait for each I/O to complete before continuing on with its next operation.

Further, if server I/O activity is single threaded like 'dd', then performance is gated by each individual I/O request, which typically has a response time of 1 ms; thus, the storage system will only produce 1000 I/Os per second.

To truly exercise a storage system to its potential, a much more robust I/O generator tool should be used instead of 'dd'. As discussed above, the Sun StorEdge™ vdbench tool is an I/O generation tool which has many options and can be programmed to drive a storage system with almost endless variations and load combinations.

An application such as database redo logs is an excellent example where response time is VERY important. In this case, the entire application is gated by the redo log or that one I/O because the database application must wait for a response back from this I/O before it may proceed with its next operation.

- **Statistics Produced by sar/iostat:**

Another unfortunate choice of Fast Growing, Inc.'s system administrators was their use of the UNIX reporting commands sar and iostat to assess the performance of their new storage system.

The sar command was likely created when storage systems were directly connected to servers, with no volume management involved. It was quite a useful tool at that time to show the performance of a specific disk. However, with the inclusion of volume managers, current configurations may consist of multiple disks and controllers, as well as multiple storage systems. The unfortunate aspect of using sar is that it reports the activity of a *logical* storage unit (LUN) from the server's perspective, rather than according to the actual *physical* layout.

In scenario 1, sar reported that the storage system was 100% busy while running the 'dd' command. As shown in Illustration 5 below , sar was reporting the behavior of only one drive, not the entire LUN which consisted of 8 drives. As 'dd' requested its I/O, the disk that stored that data

responded as it should, but since sar was reporting the behavior of 8 disk drives, it appeared to the system administrators that 8 drives together could not even handle the one I/O that was requested. The truth of the matter here is that of the 8 drives, only 1 was busy, thus the storage system was actually only 12.5% busy – not 100% busy; in other words, the storage system was only showing $1/8^{th}$ of its true potential. These system administrators required a much more detailed reporting tool to view the true behavior of their system.

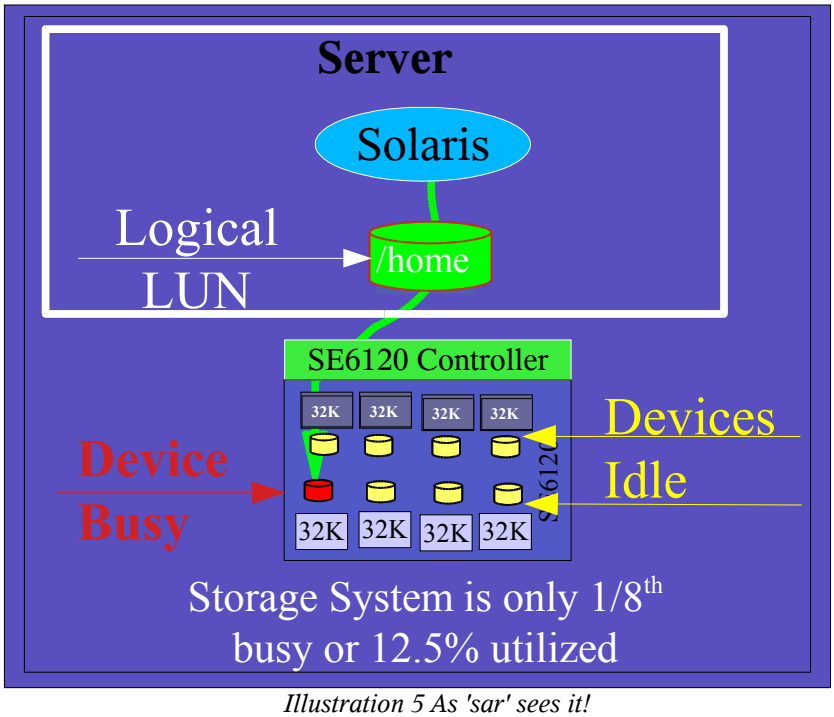

Illustration 5 As 'sar' sees it!

When a detailed level of analysis and display of data is required of a storage system, it is highly recommended to use Sun's StorEdge™ Workload Analysis Tool (SWAT) rather than the UNIX command 'sar'. SWAT has the ability, among *many* others, to display pie charts of controller activity as show in Illustration 6 below.

It is clear from this pie chart that though there are three controllers available, one controller is handling 59% of the workload! The other two are handling the remaining 41%. Thus, this scenario cannot demonstrate the full potential of this particular storage system.

Solution Summary for Scenario #1:

Had the Fast Growing, Inc. system administrators significantly increased the number of I/O requests to their new storage system using a more robust I/O generating tool such as Sun's vdbench, they would have observed considerably superior performance from their new storage system.

2. Scenario #2:

A large university was in the midst of upgrading its storage system to

accommodate an increase in the student body's use of the Netscape Mail application. The system administrators wanted to ensure that the new storage was configured to meet Netscape Mail workload demands. Therefore, they evaluated the behavior of the I/O that makes up the mail workload. Their findings showed that the average I/O was 8KB. Having determined the average I/O size of their workload, they configured the new storage system using the default storage controller volume stripe size of 16K. After upgrading to faster CPUs and storage, the system frequently felt slower to the users. Administrators were observing response times in the 1 second or more several times an hour.

Controller Striping:

As a level set, striping is a technique of mapping data so that the data is interleaved among two or more physical disks. More specifically, data is allocated in equal-sized units (called stripe units) that are interleaved between the disks. Each stripe unit is a set of contiguous blocks on a single physical disk as shown in Illustration 7 below.

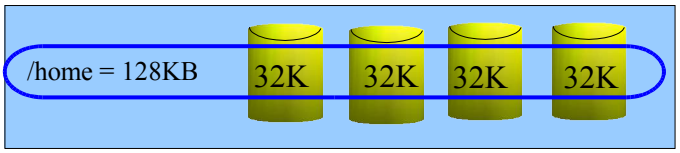

Illustration 7 Disk Striping at the Controller Level

Striping is useful if large amounts of data must be written to or read from the physical disks quickly; using many parallel data transfers to multiple disks does this. Striping is also helpful with balancing the I/O demands of multithreaded workloads across multiple disks.

Implementing the default storage controller stripe size of 16KB to handle the average I/O of 8KB seemed to be the appropriate decision. However, once the storage system was put into production, it appeared to nearly stop responding at multiple times throughout the day. These very slow response times from the storage system caused the application, Netscape Mail, to also nearly stop. Clearly, this type of application behavior is unacceptable to any computing end user.

The performance problem in this case was exacerbated by the fact that the storage volumes were configured as RAID 5. RAID 5 is striping across a set of drives, with a parity stripe interleaved into the data stripes. The parity is used to provide for write error correction as well as recoverability in the event of one disk drive failure in that volume. RAID 5 has an overhead penalty that the other typically used RAID

options do not have. This penalty is incurred by the parity operation, which is required for every single write request. Each write requires two reads and two writes - one read to obtain the old data and another to obtain the old parity, followed by calculation of the new parity, and finally one write to store the new data and another write to store the new parity.

Comparing RAID 5 to RAID 1 shows the significant difference in overhead of RAID 5. RAID 1 requires only two disk drive accesses for every write request: one drive access to write to the data drive, another drive access to write to the mirror drive, clearly much more streamlined, and usually much faster, than a RAID 5 operation.

Customer Transfer Sizes:

As noted previously, the university's application average transfer size was 8KB, and the system administrators used the default 16KB controller storage stripe size, which to their disappointment did not perform as they would have logically expected. To resolve this, a thorough analysis of how their Netscape Mail workload behaved in the new storage configuration was required.

For further analysis during the periods when the system had extraordinarily slow response times, SWAT, which is discussed in Scenario 1 above, was used to trace the workload behavior and the storage system performance.

Reviewing the data collected by SWAT revealed that during the slow response times from the storage system, Netscape Mail was demanding many large I/Os from its RAID 5 volumes. These very long latencies were triggered by many write requests of very large block sizes as shown in Illustration 8 below, which were typically above 130KB. As this happened, the IO queue depth in the storage system grew to between 100 and 200 I/Os waiting to be processed. Unfortunately, the storage system as it was currently configured could not handle this queue depth with these large block size write requests very efficiently.

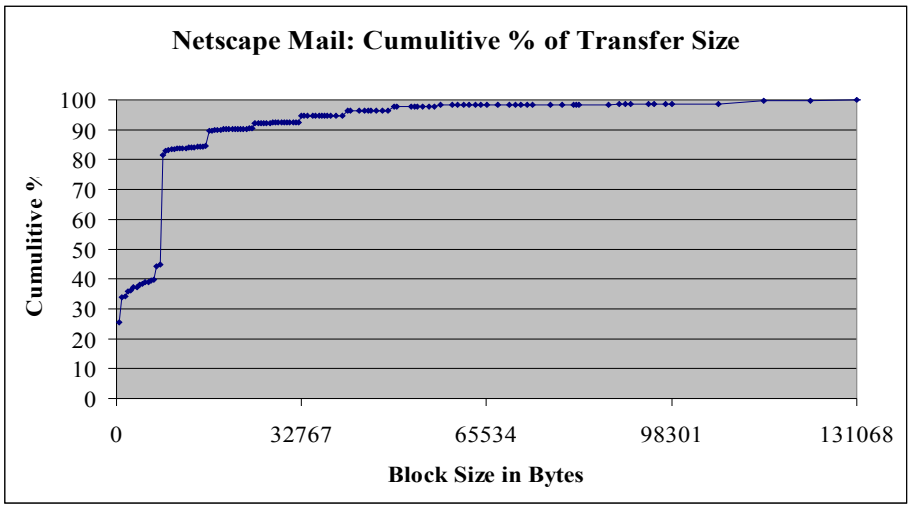

Illustration 8 Netscape Mail Cumulative % of Transfer Size

Given that the storage controller volume stripe size was set to 16K, and the majority of the burst of 100 to 200 I/O requests were 130KB or larger, the storage system had to access at least 9 individual disk drives to write that 130KB worth of data, i.e. 130 ÷ 16 = 8.125 rounded up to 9 drives. During these bursts of high volume large writes, there were between 100 to 200 I/Os queued at one time, which translates to between 900 and 1800 disk drives accesses to complete all of those I/O requests, i.e. 9 x (100 and 200) = 900 and 1800.

In this scenario, i.e. a workload consisting of large block writes and a storage system configured with small stripe sizes, the storage system must work furiously to complete all I/O requests. While servicing these I/Os, the storage system must access an exceedingly high number of drives, and the application must wait for a response from the storage system indicating that each I/O request has completed successfully. In the storage industry this is referred to as storage thrashing; i.e., a lot of storage activity, but very little response back to the requesting application.

Transfer and Drive Access:

In order to understand the effects that small storage controller stripe sizes have on large block write requests, it is essential to understand the disk drive configuration itself. As an example, if the disk drive is configured with small stripe sizes such as 16KB, and a write request of 128KB is made, 8 drives must be accessed to complete that write, i.e. 8 x 16 = 128 . If the stripe size is set to 32KB, then 4 drives must be accessed. Similarly, if the stripe size is set to 64KB, then only 2 drives

need be accessed to complete the 128KB write, as depicted in Illustration 9 below.

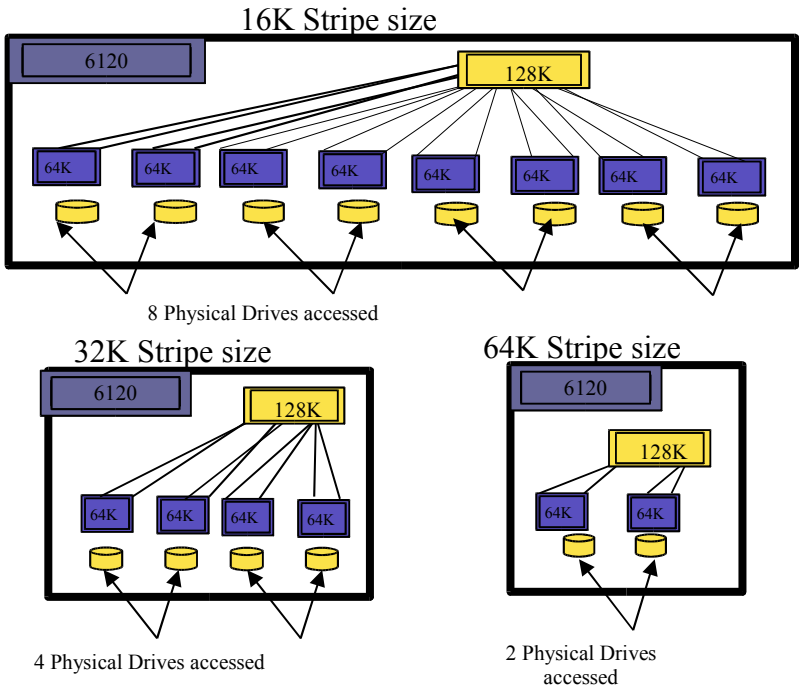

Illustration 9 Controller Stripe Stripe vs Block Write Size

Mechanical Latency:

Extending the the theory of transfer size and number of drives being accessed, is the actual transfer rate of a disk drive itself. Transfer rate of a drive is another element that affects the performance of a storage system. Fortunately, the newer 15K RPM disk drives have been designed to improve mechanical latency which has improved its data transfer rate. These drives have transfer speeds that range from 57-85 MB/second, as published by the disk vendors. However, while the mechanical speed to access this data (usually referred to as seek and latency) has improved over the years, mechanical access times have not quite kept up with the data transfer rates.

Further investigation shows that the larger the transfer size, or chunk of data being transferred to these 15K RPM drives, the more efficient

the transfer process itself is. This is because it takes the same amount of time to get to the data and put the head of the drive on that data point (Seek + Latency) - no mater the size of the data being transferred as depicted in Illustration 10 below.

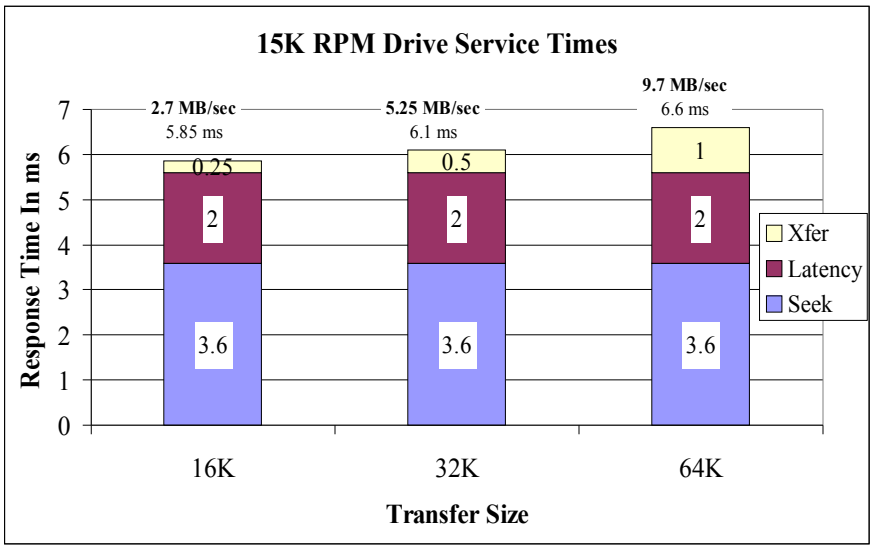

Illustration 10 Drive Service Time

Additionally, it is more efficient to transfer data from one disk drive versus many drives simultaneously, as the following example shows. As discussed above in the 'Transfer and Disk Access' section, when the Netscape Mail application slowed almost to a halt, there were bursts of large write requests of 130 KB and above. Using the vendors specifications of the transfer speed of the 15K RPM drives, we can calculate the total response time for different stripe size configurations as follows:

- **16K Stripe Size:** With the storage configuration of a 16K stripe size on the disk, the storage system must access 8 drives to accommodate that write request. Calculating the total response time shows that it will take 8 x 5.85 = 46.8 ms to complete the write.
- **32K Stripe Size:**With the storage configuration of a 32K stripe size on the disk, the storage system must access 4 drives to accommodate that write request. Calculating the total response time shows that it will take 4 x 5.85 = 23.4 ms to complete the write.
- **64K Stripe Size:**With the storage configuration of a 64K stripe

size on the disk, the storage system need only access 2 drives to accommodate that write request. Calculating the total response time shows that it will take 2 x 5.85 = 11.7 ms to complete the write.

Thus, to complete the same size write request a:

- 16K stripe should take 48.6 ms to complete
- 32K stripe should take 23.4 ms to complete
- 64K stripe should take 11.7ms to complete

Clearly, a 64k stripe is much more ***efficient*** at completing large blocks writes as opposed to 32K and 16K stripe sizes.

Impact of Stripe Size:

To substantiate this theory of the impact of disk stripe size, a lab exercise was performed where a storage system was configured with an 8 drive RAID5. The configuration started with a disk drive stripe size of 16KB. Next, a workload of writes was applied to the storage system. The workload started with writes of 16KB blocks and under. It was then increased to between 16KB blocks and 32KB blocks and so on, until the workload demand was above 500KB block writes. The exercise was repeated for 32K stripe sizes and 64K stripe sizes. The output of the exercise is shown in Illustration 11 below.

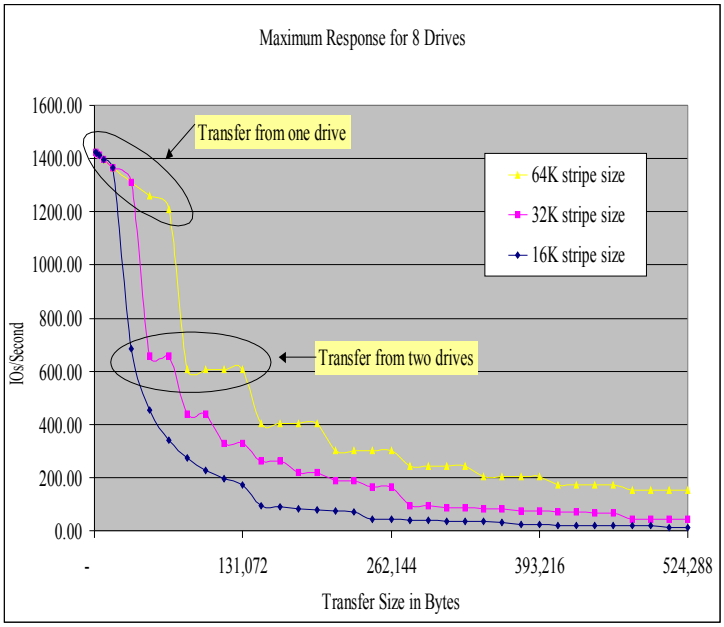

Illustration 11 Maximum Response Time

The first execution, where the storage block stripe size was 16KB, of the lab exercise of the 16KB block writes resulted in 1,400 I/Os per second! However, the instant the workload exceeded 16KB (which, remember, is the physical stripe size on the disk) the performance was decreased by more than one half to only 670 I/Os per second than what was achieved when the write request was 16K or less! This is due to the need to access a second disk in the RAID5 LUN to complete the write.

This same observation proved true for the larger stripe sizes as well, i.e. as soon as the physical stripe size was exceeded and a second drive was required to be accessed, the number of I/Os per second decreased dramatically.

This experiment showed that the performance of a larger physical stripe size was the most efficient configuration to use regardless of the I/O request size, i.e. smaller I/O requests performed just as fast with larger stripe sizes as with smaller stripe sizes. Further, a large I/O request response time was much better with large stripe sizes than with small stripe sizes, as fewer drives were required to be accessed to complete the I/O request(s).

With this level of understanding, it made an approach to a resolution to the university's overall end-to-end solution performance problem manageable. The first step was to take a trace of the university's solution to gather an hour of I/O activity during the slow response times. The data gathered was then analyzed, and it was determined that there were 15 periods where response time was an abominable 800 ms or more!

In order to take appropriate steps toward making recommended configuration changes to the university's production environment, their Netscape Mail production configuration was recreated in the lab, specifically using 16K physical disk stripe sizes in a RAID 5 configuration. A replay of the data gathered by SWAT was performed using vdbench and it demonstrated the precise performance problems observed in the production environment.

The Netscape mail configuration was then reconfigured using 32K physical disk stripe sizes and RAID 5. The replay of the workload was run, and the very slow response times were reduced from 15 periods to 5 periods, which is a 66% performance improvement! Next the physical disk stripe size was adjusted to 64K and the replay of the workload was executed. However, this time there were ***no*** periods of slow response time – a 100% improvement over the initial production environment performance!

Solution Summary for Scenario #2:

A thorough understanding of the application workload characterization enabled the system administrators to make proper changes to obtain best performance to meet their application and business needs. The use of robust powerful tools such as Sun StorEdge™ Workload Analyzer Tool and Sun StorEdge™ vdbench I/O generator tool to help gather and analyze the workload are also very important tools to assist them to gain a thorough understanding of the workload and in making the right decision to configure for the best performance of any storage system.

In this specific case, the right configuration decision was to implement a large physical disk stripe size to improve performance and decrease the severe I/O bottlenecks triggered by the large bursts of I/O demands places upon the storage system.

3. Scenario #3:

An IT department was experiencing slow storage response time due to the age of storage systems and amount of data. Consequently, they purchased new storage with faster controllers, faster disks, and more storage capacity, which was to be added to their current configuration.

After installing the new storage in the current configuration, the system administrators expected to gain significant overall system performance - end-to-end. Much to their chagrin however, the performance was the same as before the new storage system was added.

The primary question in this situation is: why dose the overall performance not scale significantly having added the new (faster) storage to the current configuration? The problem lies in the fact that the system administrators merely added the storage but did not distribute the application workload(s) across all the storage now available to the server(s). This can be characterized as skew.

Skew:

Skew refers to the asymmetry of a distribution about its mean, or, in our present context, to the non-uniform distribution of data or I/O activity across storage devices.

There are two types of storage system skew: disk skew and controller skew.

- *Disk Skew:*

Disk skew occurs when an area on a disk has a higher amount of activity than the rest of the disk. This is simply because some data is accessed more frequently than other data. For example, an index to a database might have 8-12 accesses before the required set of data from the database is accessed. This can cause application bottlenecks and reduced performance.

- *Controller Skew:*
  Controller skew occurs when one controller has a higher amount of activity compared to the rest of the controllers in a storage system. For example, if one controller out of 20 is receiving 50% of the total I/O activity or workload, then this particular workload exhibits significant skew.

In the case of disk or controller skew, if either reaches a saturation point, the potential performance of the entire end-to-end system decreases. In other words, if the workload is not evenly distributed, the application cannot scale to its full potential.

When skew is present, it is suggested to employ a host-based volume manager to distribute the I/O activity as evenly as possible across all available storage controllers. In short, a volume manager is a host-based application that manages the storage system(s) as if it(they) were disks to the server. As an example of employing a volume manager, consider an application which requires 200K IOPs to perform efficiently, given that a single storage controller can produce 20K IOPS. In this case, 200K IOPs may be attained by using a volume manager to distribute the workload evenly across ten controllers.

Throughput Curve:

To explain skews impact on throughtput lets examine a simple system with twenty storage controllers. Each controller is capable of 1,000 IOs per Second for a specific workload. With the load evenly distributed across 20 controllers, the system is capable of 20,000 I/Os. Now in an extreme case, if we had 50% of the workload going to one controller, the overall throughput would be 1,000 IOs from the controller with 50% of the load, and we would have another 1,000 IOPS from the remainder of the controllers. The theoretical results are shown in Illustration 12below. It is very advantageous to have skew numbers of 15% or lower. This keeps the workload evenly spread across several devices.

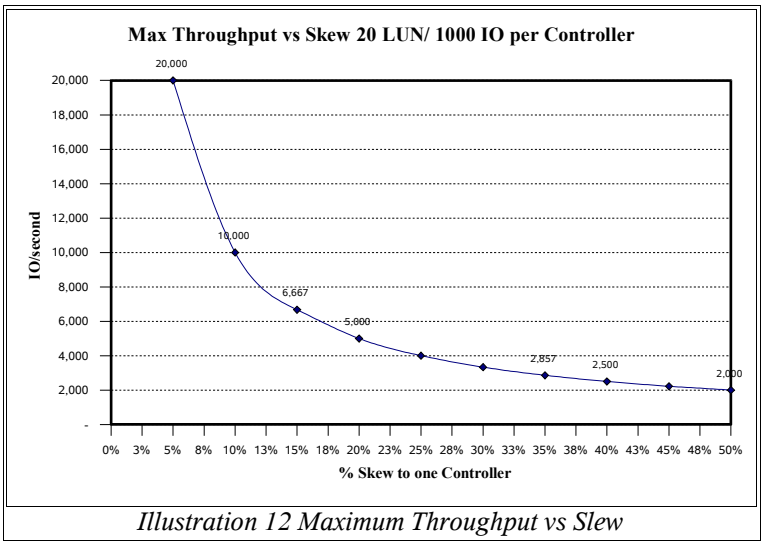

Illustration 12 Maximum Throughput vs Slew

With real application conditions however, the variability of the demand for I/O rarely - if ever - will scale linearly as this experiment was designed to do. As such, real world distribution will not likely be additive, i.e. if one controller is capable of supplying 2,000 IOPs, this does not automatically mean adding a second controller will result in the combination of the two producing 4,000 IOPs. Because of such variability in real world application workload behavior, it is highly recommended to evaluate the performance of the workload after employing a volume manager to understand how much skew still exists and make appropriate adjustments as necessary.

Volume Manager Usage:

As discussed in the previous section, a volume manager was used to manipulate the workload across many controllers. How the volume manager is used to configure the storage system(s) can have a significant effect on the overall performance of the end-to-end solution. Through real experiences, benchmarking, and lab exercises, much has been learned regarding the most effective implementation of host volume management stripe sizes. Here are several recommendations:

- **Beware of stripe sizes that are too small**

Historically, and particularly with large databases, to increase performance it was common to create many small stripes across as many devices as possible, both disks and controllers. However, with today's improved storage technologies, this is no longer necessary. It is more important to make better use of the storage technology than to spread across many controllers.

Creating many small stripe sizes across many controllers may render the storage system's ability to perform certain performance enhancements such as 'parity-on-the-fly'. Where 'parity-on-the-fly' is an optimization of the storage controller parity calculation algorithm which does not require finding and reading the old data from the slow physical hard drives to create the Raid 5 parity, if the entire data stripe is in cache (e.g., if all of the data of 6 of the drives in a 6+1 RAID 5 group are in the storage cache) then the controller can calculate the parity from the cache and need not read from the slower hard drives.

For example, if a file of 16MB in size needs to be written to disk, but the volume it belongs to has been created with stripe sizes of 128KB, it would require 125 block writes to complete a full write to the disk system; i.e., 16MB ÷ 128KB = 125. As such, it is unlikely that all of the 125 writes would reside in one storage controller cache as depicted in the Illustration 13 below. This unfortunately disables the storage controller's ability to use parity-on-the fly.

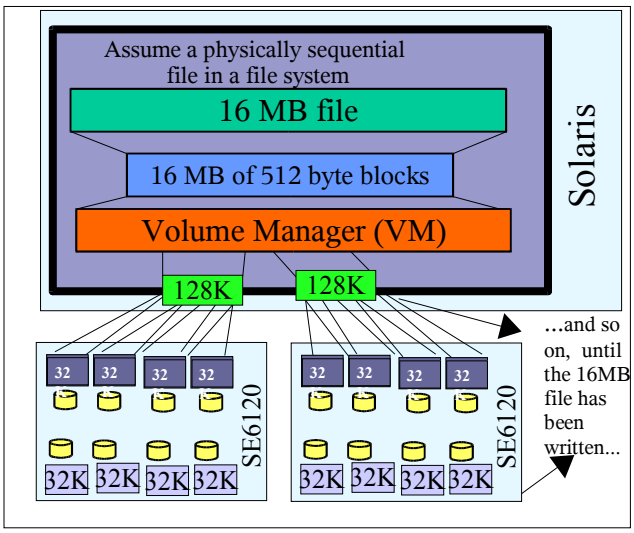

Illustration 13 Stripe Size is Too Small

- **Beware of stripe sizes that are too large**

  If too large a stripe size is chosen for a host based volume, then all I/Os are forced to funnel through a single controller causing an unbalanced load, or a load that is skewed as discussed here in Scenario #3 above. For example, if a file of 16MB size needs to be written to disk, and the volume it belongs to has been created with stripe sizes of 16MB, then all of the I/O activity goes through one - and only one – controller, causing 100% skew; i.e., a single controller doing all the work while all other controllers remain idle, as shown in Illustration 14 below.

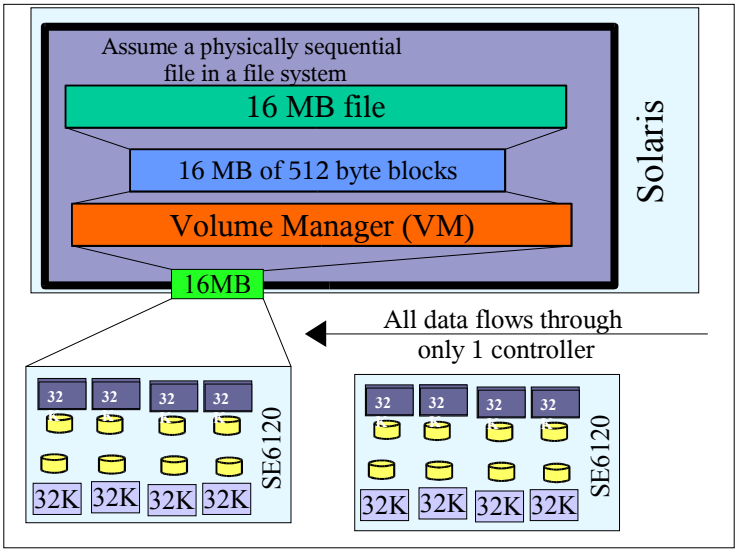

Illustration 14 Stripe Size is Too Large

- **Several Host Volume Manager Configuration Recommendations**

  As previously mentioned, through many benchmarks and real life examples, the following recommendations have come to light regarding the harmony of a host volume manager stripe size and the storage controller stripe size:

  - RAID 5: The host volume manager stripe size should be between 2 - 4 times larger than the storage controller stripe size
  - RAID 0 and 1: The host volume manager stripe size should be between 1 - 2 times larger than the storage controller stripe

size

Solution Summary for Scenario #3:

To improve the performance of the situation in scenario #3, the system administrators must distribute the workload as evenly as possible across all available storage controllers using a host-based volume manager. In order to do so properly, the administrators must understand how much workload skew exists and where it is focused.

The evaluation of skew in their workload can be performed by tracing the I/O activity using SWAT as discussed in Scenario #2, and evaluating which controllers are handling the majority of the I/O activity. Then, a reconfiguration plan can be established to redistribute the workload more evenly. Ideally, a test environment should be used to examine the effects of the reconfiguration plan before applying it to a production environment. This process may require several iterations to obtain the most performance benefits.

4. Conclusion

As previously discussed, in today's demanding and competitive business world, it is paramount for any IT department to obtain optimum end-to-end performance from their companies' computing solutions. To achieve optimal performance, it is essential to configure their solutions to meet their business application workload requirements.

Diminishing storage system bottlenecks should help improve overall end-to-end performance, which can assist in meeting, and hopefully exceeding, overall business performance requirements.

Having read this paper, it should be much easier to address several of the common storage system bottlenecks in the industry today, thus improving end-to-end performance. The reader should now have the ability to:

1. Increase end-to-end performance by taking advantage of multi-threaded I/Os to the storage system controller and its drives
2. Deliver predictable storage system performance by matching the storage system configuration to the workload requirements
3. Increase storage system through put by distributing the

workload across the storage system(s)

5. Authors

Amanda Scharmer (Hudson), Storage Specialist and Storage Performance Evangelist, Sun Microsystems, Inc.

Steven Johnson, Storage Performance Scientist, Sun Microsystems, Inc.

6. Editor:

A HUGE and humble thanks to Jeff Shafer, Ph.D., Performance Analyst,Sun Microsystems, Inc. for completing yet another outstanding editing job of this and many of our other papers!