

Storage System Bottlenecks and Their Solutions

A Whitepaper

July 21, 2005



Copyright © 2005 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

This distribution may include materials developed by third parties.

Sun, Sun Microsystems, the Sun logo, Java, Solaris, Sun StorEdge, Sun StorEdge™ Workload Analysis Tool and Sun StorEdge™ vdbench are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Veritas™ vxbench. Netscape™ Mail. Oracle©. Sybase©. Microsoft® Windows®. Windows® Tracelog.

Table of Contents

- 1.Scenario #1:.....8
 - Understanding Storage System Performance Corner Cases:.....8
 - Corner Case Benchmark Results include:.....9
 - Workload Characterization Tools:.....10
 - Sun StorEdge™ Workload Analysis Tool (SWAT)10
 - Sun StorEdge™ Vdbench (vdbench).....11
 - Supply vs. Demand:.....12
 - Single-Threaded versus Multi-Threaded:.....12
 - I/O Queue Depth:.....14
 - Statistics Produced by sar/iostat:.....15
 - Solution Summary for Scenario #1:.....17
- 2.Scenario #2:.....18
 - Controller Striping:.....18
 - Customer Transfer Sizes:.....19
 - Transfer and Drive Access:.....20
 - Mechanical Latency:.....21
 - Impact of Stripe Size:23
 - Solution Summary for Scenario #2:.....25
- 3.Scenario #3:.....25
 - Skew:.....26
 - Throughput Curve:.....27
 - Host-Based Volume Manager Usage:.....28
 - Beware of host-based stripe sizes that are too small.....28
 - Beware of host-based volume stripe sizes that are too large.....31
 - Volume Configuration Recommendations.....33
 - Solution Summary for Scenario #3:.....35
- 4.Conclusion.....36
- 5.Authors.....37
- 6.Editor:.....37

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 Adapter, 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:

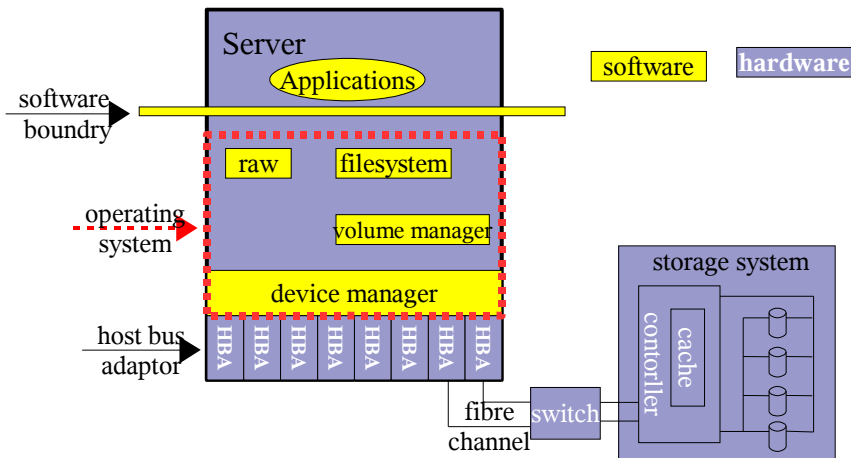


Illustration 1 Typical Storage System

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, 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 LUN, where a LUN (Logical Unit) is a combination of disks or storage systems visible to the server. A LUN is created by an OS command and 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, thereby increasing 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 Adapter (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 and response time.

- **Throughput:** The amount of work that can be done by the storage system per unit of time. For storage performance, the measurement unit is MegaBytes (MB) per second. Throughput can be described as how effectively the storage controller is able to supply the application with data. Generally, batch jobs and scientific applications such as seismic evaluation are most affected by throughput.
- **Response Time:** How long it takes to complete a storage operation, defined to include reads, writes, searches, etc. Response Time describes how efficiently the storage controller processes transactions. Generally, on-line transaction types of applications are most concerned with IO/Second (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 (Rotational 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.
- **Write Coalescing:** Combining several or many small blocks into one large block then writing that single large block to disk, typically used on the storage array controller for sequential or skip sequential write operations.

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, lab and theoretical data and have been used to analyze these issues and demonstrate the steps to resolve

them.

1. Scenario #1:

A company (for purposes of this paper 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 an OLTP-type application, 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 pure cache and pure disk performance capabilities.

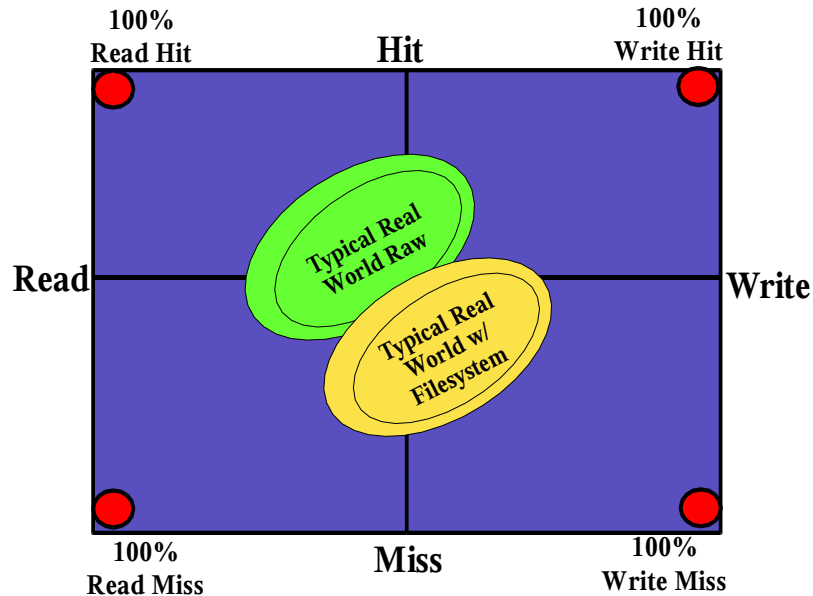


Illustration 2 - Corner Case Illustration

- **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 Miss or 100% Read Miss in Illustration 2 above. This demonstrates the slowest (and 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 strategy to truly understand the workload of the application(s) that will be using the new storage, and it is also extraordinarily important 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 issues.
- 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 recurring 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 can be obtained directly from I/O trace data. The information gathered is stored on the system where the data has been collected.

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

- **Sun StorEdge™ Vdbench (vdbench)**

Sun StorEdge™ vdbench, 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.

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

- I/O rate can be set to MAX, exponential, or fixed inter-arrival time.
- Percent read controls 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 to 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 affect 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. 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 per second is requested of a storage system, it will supply only one I/O per second, 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.

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 storage system I/O demand is to request multiple simultaneous I/Os; this is commonly referred to as multi-threading. Thus, the storage system is allowed to supply more I/Os to the server, 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 multi-threaded format. Single-threaded theoretically takes 8 ms to complete the request, while the multi-threaded request might require only 1 ms to complete.

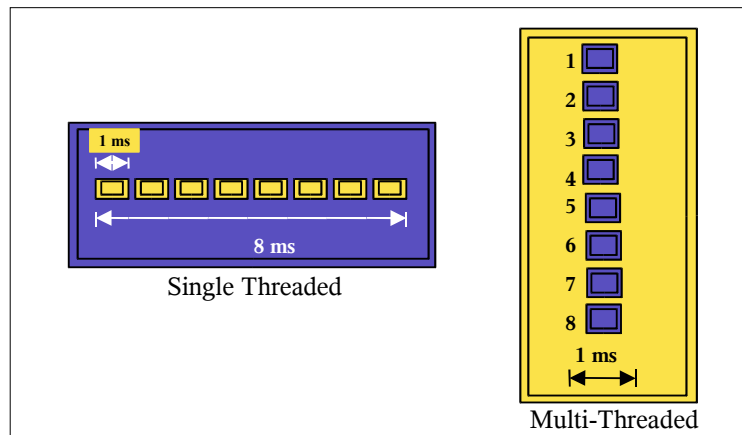


Illustration 3 Single-Threaded vs. Multi-Threaded

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, 8 threads produced 707 IOPs, and 16 threads produced 967 IOPs – a 562% performance improvement! This demonstrates that increasing the thread count, within reason, can dramatically improve storage system performance.

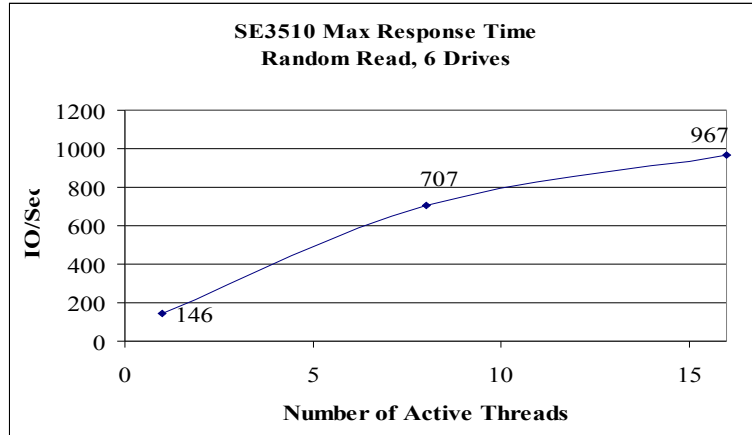


Illustration 4 Response Time vs Thread Count

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

- **I/O Queue Depth:**

If the application such as 'dd' is single threaded, the end-to-end system response time determines the volume of I/Os processed by the storage system per unit 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 the server I/O activity is single-threaded like the Solaris 'dd' command, then performance is gated by each individual I/O request, for example if the response time is 1 ms; 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, Sun StorEdge™ vdbench is an I/O generation application which has many options and can be programmed to drive a storage system with

almost endless variations and load combinations.

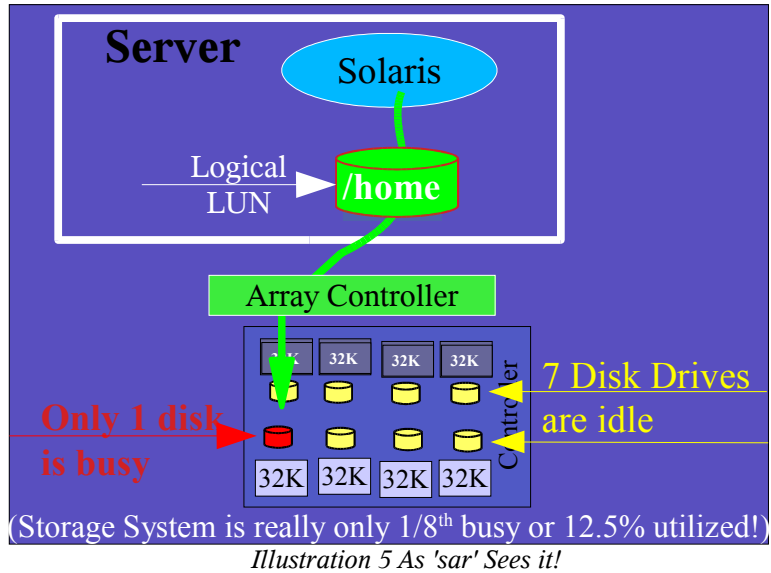
An application such as database redo logs is an excellent example where response time is VERY important. In this case, the entire application is gated by the redo log or a specific I/O because the database application must wait for a response back from each I/O before it may proceed with its next operation.

- **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 iostat command was likely created when storage systems were directly connected to servers, with no volume management involved. At that time, it was quite a useful tool 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 iostat 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, iostat reported that the storage system was 100% busy while running the 'dd' command. As shown in Illustration 5 below, iostat was reporting the behavior of only one drive, not the entire LUN consisting of 8 drives. As 'dd' requested its single I/O, 1 of the 8 drives satisfied the request. But since iostat was reporting the behavior of 8 disk drives, it appeared to the system administrators that the 8 drives were 100% busy, when in fact that was not the case. 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 1/8th of its true potential, as Illustration 5 below shows. These system administrators required a much more detailed reporting tool to view the true behavior of their storage system.



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 shown in Illustration 6 below.

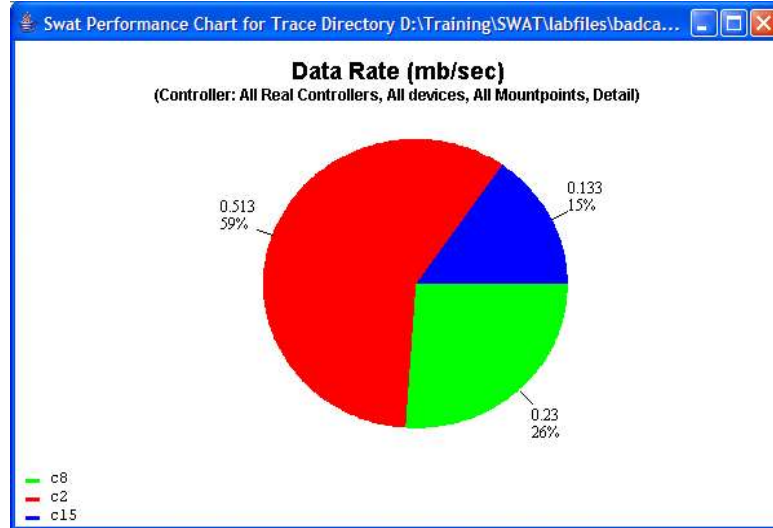


Illustration 6 SWAT IOP's Pie Chart of 3 Storage Controllers

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 better performance.

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 of 1 second or more several times an hour.

Controller Striping:

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.

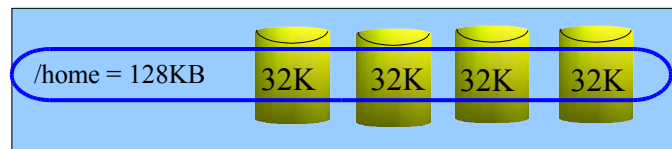


Illustration 7 Disk Striping at the Controller Level

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 is how striping improves data transfer parallelism. Striping is also helpful in 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 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 involves

striping across a set of drives, with a parity stripe interleaved into the data stripes. This parity is used to provide for write error correction as well as recoverability in the event of one disk drive failure in the volume. However, RAID 5 has an overhead penalty that other typically used RAID options do not have. This penalty is incurred by parity computation, 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 reveals this significant difference in overhead. RAID 1 requires only two disk drive accesses for every write request: one drive access to write to the data drive, and 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 periods of extraordinarily long response times, SWAT, discussed in Scenario 1 above, was used to trace workload behavior and storage system performance.

Review of the data collected by SWAT revealed that during periods of long response times, Netscape Mail was demanding many large write 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. Consequently, the IO queue depth in the storage system grew to between 100 and 200 I/Os. Unfortunately, the storage system as it was currently configured could not efficiently handle the large block size write requests.

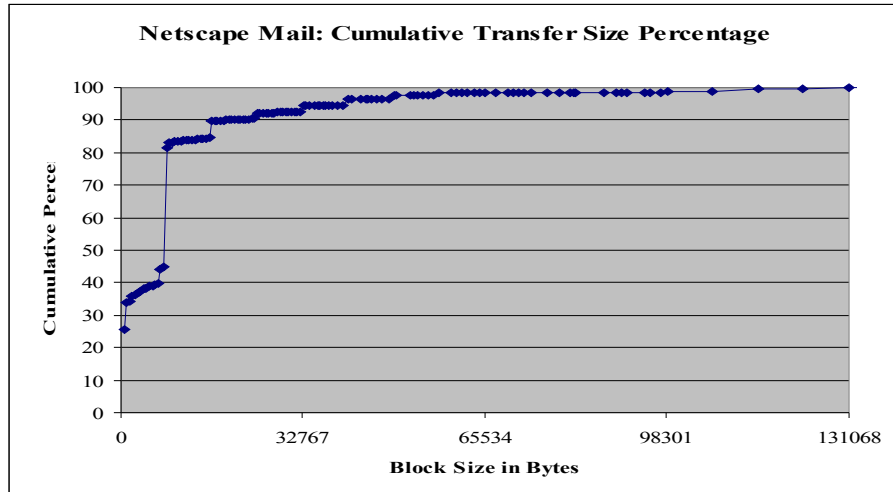


Illustration 8 Netscape Mail Cumulative Transfer Size Percentage

Given that the storage controller volume stripe size was set to 16 KB, and the majority of the burst of 100 to 200 I/O requests were 130 KB or larger, the storage system had to access at least 9 individual disk drives to write that 130 KB worth of data; i.e., $130 \div 16 = 9$ (rounded up). 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 drive accesses to complete all I/O requests; i.e., $9 \times (100, 200) = (900, 1800)$.

In this scenario, 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 16 KB, and a write request of 128 KB is made, 8 drives must be accessed to complete that write, i.e. $8 \times 16 = 128$. 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.

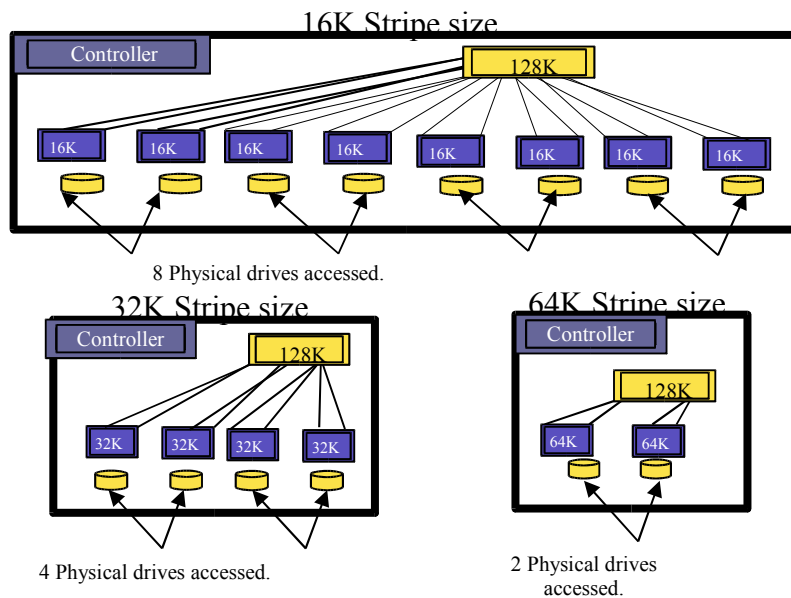


Illustration 9 Controller Stripe Size vs Block Write Size

Mechanical Latency:

Drive transfer rate is another variable impacting the performance of a storage system. Fortunately, the newer 15K RPM disk drives have been designed to improve mechanical latency which has improved their 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 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 at that point (seek + latency) no matter the size of the data being transferred. This is depicted in Illustration 10 below, which represents a simple case of the effect of drive stripe size with large data transfers and their performance impact on the 15K RPM disk drives.

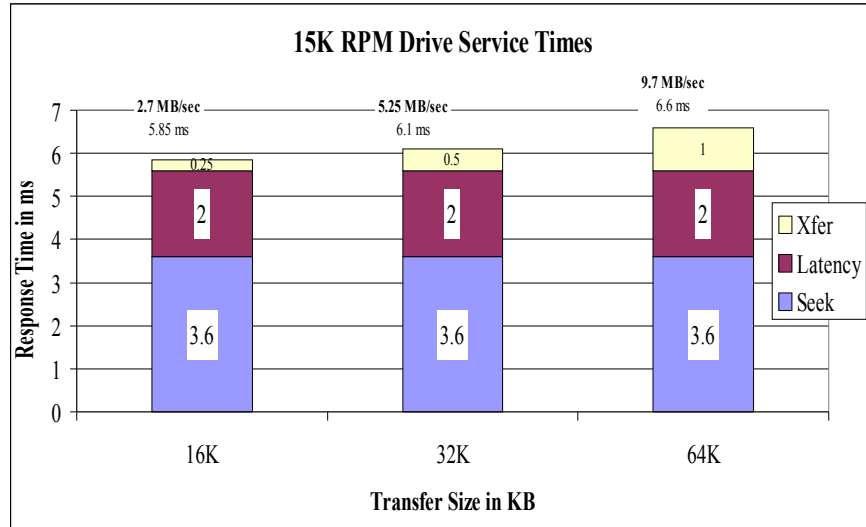


Illustration 10 Drive Service Time

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 128 KB and above. Using the vendor specifications for the transfer speed of the 15K RPM drives, we can theoretically calculate the accumulative service times of drives verses stripe sizes.

In our example, the accumulative service times is equal to the productive disk service time + non-productive disk service time. Here, productive disk service time is when the disk is actually transferring data, while non-productive service time is when the disk is *positioning* to transfer data, i.e. not transferring any data. For different stripe sizes, the theoretical calculations are as follows:

- 16K Stripe Size:** With the storage configuration of a 16K stripe size on the disk, the storage system would have to access 8 drives to accommodate a write request of 128KB. Theoretically, calculating the total time for that RAID set to complete this write request, shows that it will take 8 drives x 5.85 ms = 46.8 ms. Of the total 46.8 ms, only 2 ms is used for transferring the data or spent in productive disk service time. The remaining 44.8 ms is spent in non-productive disk service time! This translates to 96% of non-productive activity of the disk drives for this particular example!
- 32K Stripe Size:** With the storage configuration of a 32K stripe

size on the disk, the storage system would have to access 4 drives to accommodate a write request of 128KB. Theoretically, calculating the total time for that RAID set to complete this write request, shows that it will take 4 drives x 6.1 ms = 24.4 ms. Of the total 24.4 ms, only 2 ms is used for transferring the data or spent in productive disk service time. The remaining 22.4 ms is spent in non-productive disk service time! This translates to 92% of non-productive activity of the disk drives for this particular example!

- **64K Stripe Size:** With the storage configuration of a 64K stripe size on the disk, the storage system would have to access 2 drives to accommodate a write request of 128KB. Theoretically, calculating the total time for that RAID set to complete this write request, shows that it will take 2 drives x 6.6 ms = 13.2 ms. Of the total 13.2 ms, only 2 ms is used for transferring the data or spent in productive disk service time. The remaining 11.2 ms is spent in non-productive disk service time! This translates to 84% of non-productive activity of the disk drives for this particular example!

Thus, to complete the same size write request a:

- 16K stripe spends only 4% of it's total time performing the actual transfer of data.
- 32K stripe spends only 8% of it's total time performing the actual transfer of data.
- 64K stripe spends only 16% of it's total time performing the actual transfer of data.

Clearly, a 64KB stripe is much more *efficient* for transferring large block IO's than smaller stripe sizes. In other words, a higher percentage of the disk activity is spent performing productive transfer rather than non-productive disk movement.

Impact of Stripe Size:

To substantiate the impact of disk stripe size, a simple model was created where a storage system was configured with an 8 drive RAID5 set. The configuration started with a disk drive stripe size of 16 KB. Next, a workload of reads was modeled. The workload started with reads of 16 KB blocks or smaller. It was then increased to between 16 KB blocks and 32 KB blocks and so on, until the workload demand was above 500 KB 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.

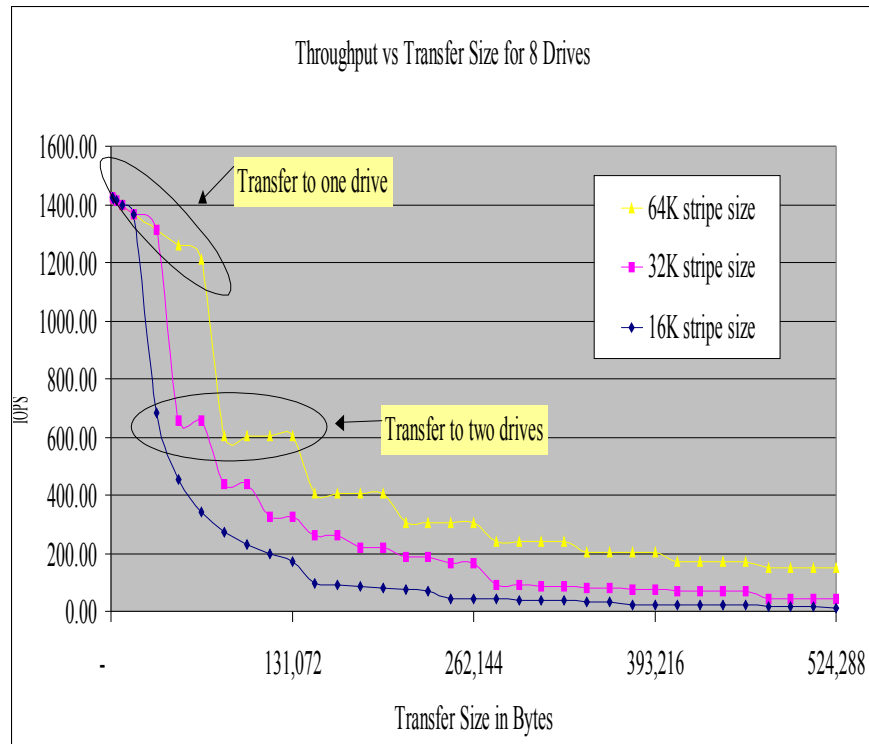


Illustration 11 Throughput vs. Transfer Size

The first execution, where the storage block stripe size was 16KB, of the 16KB block reads 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! This is due to the need to access a second disk in the stripe to complete the transfer.

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 model showed that the performance of a larger physical stripe size was the most efficient configuration to use regardless of the I/O request size. 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 16 KB 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!

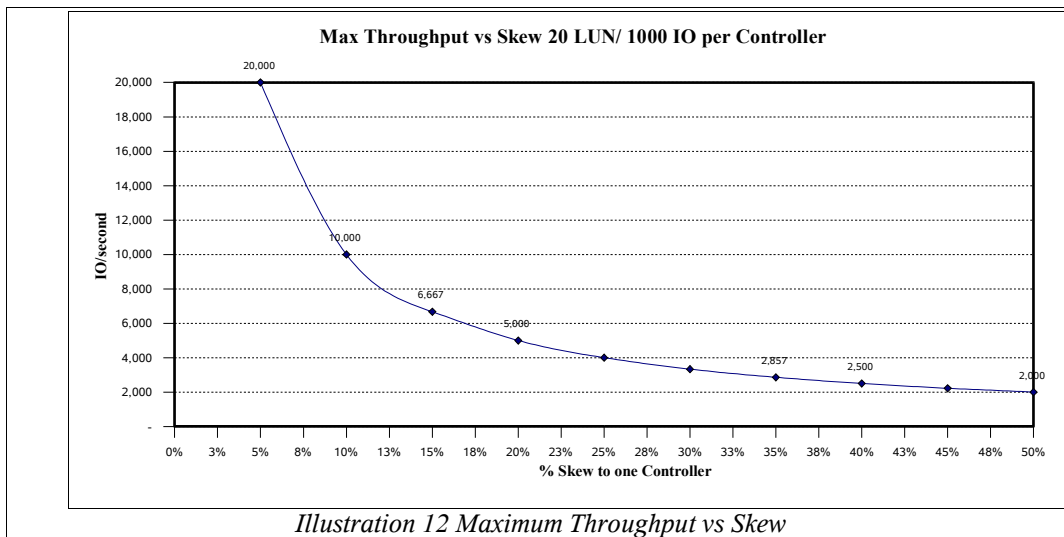
Solution Summary for Scenario #2:

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. Robust, powerful tools such as the Sun StorEdge™ Workload Analyzer Tool and the Sun StorEdge™ vdbench I/O generator gather and analyze workload data and provide invaluable assistance making the right decisions to configure any storage system for the best possible performance.

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 large bursts of I/O demands placed 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 primary question in this situation is: why does 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 uneven distribution is known 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 approaches 100% utilization, 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, use of a host-based volume manager is suggested to distribute 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. For example, 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 the impact of skew on throughput, let's 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 potentially 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 12 below. Based on this example, it is clear that it is advantages to balance the workload across all available devices, controllers and disk drives, regardless of how few or many are available to the server. This keeps the workload evenly distributed and works all available devices, reducing idle controllers and disk drives.

However, under real conditions, 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 distributions 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.

Host-Based Volume Manager Usage:

As discussed in the previous section, a volume manager was used to uniformly distribute the workload across many controllers. How a volume manager is used to configure a storage system can have a significant effect on the overall performance of the end-to-end solution. Through real experience, 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 host-based 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 storage system controllers. With today's improved disk and controller technologies however, that approach is no longer necessary. It is now more important to make better use of the new technology capabilities rather than to spread across many devices.

As such, creating small stripe sizes across *many* controllers may stop the storage controllers ability to perform some of its technology improvements. One of these improvements that is typically negated by many small stripe sizes is 'parity-on-the-fly'.

Parity-on-the-fly is a specially designed performance optimization of the storage controllers RAID 5 algorithm used for write operations, which simply means in certain scenarios it can calculate parity with out having to access the disk drives. More specifically, parity-on-the-fly activates for a write operation when all the data in a RAID 5 group is stored in the storage controllers cache. In this scenario, the cache algorithm collects the full RAID 5 group data from the cache, then calculates the new parity from that data rather than wasting valuable time accessing the disk drives to retrieve the old data and old parity to calculate the new parity.

The example depicted in Illustration 13 below, shows a **host** LUN or volume, which was created by the host volume manager using 128K chunks out of a storage LUN or volume. The **storage** LUN was created by the storage controller as a 512K

RAID 5 stripe set using 64K chunk sizes on the disks to make up the 512K stripe size.

When a 512K file is written to the storage controller, the **host** volume manager will write this 512K file in 4 separate 128K chunks, ($512K / 128K = 4$). In addition, these 128K chunks are written alternately between the two controllers as shown in this example by the different color boxes of green, pink, purple, and yellow.

Therefore, the 1st 128K (green box) of the hosts 512K file, is written to storage controller 1, the 2nd 128K chunk (pink box) is written to storage controller 2, the 3rd 128K chunk (purple box) is written to storage controller 1, and finally the 4th 128K chunk (yellow box) of the 512K file is written to the 2nd storage controller.

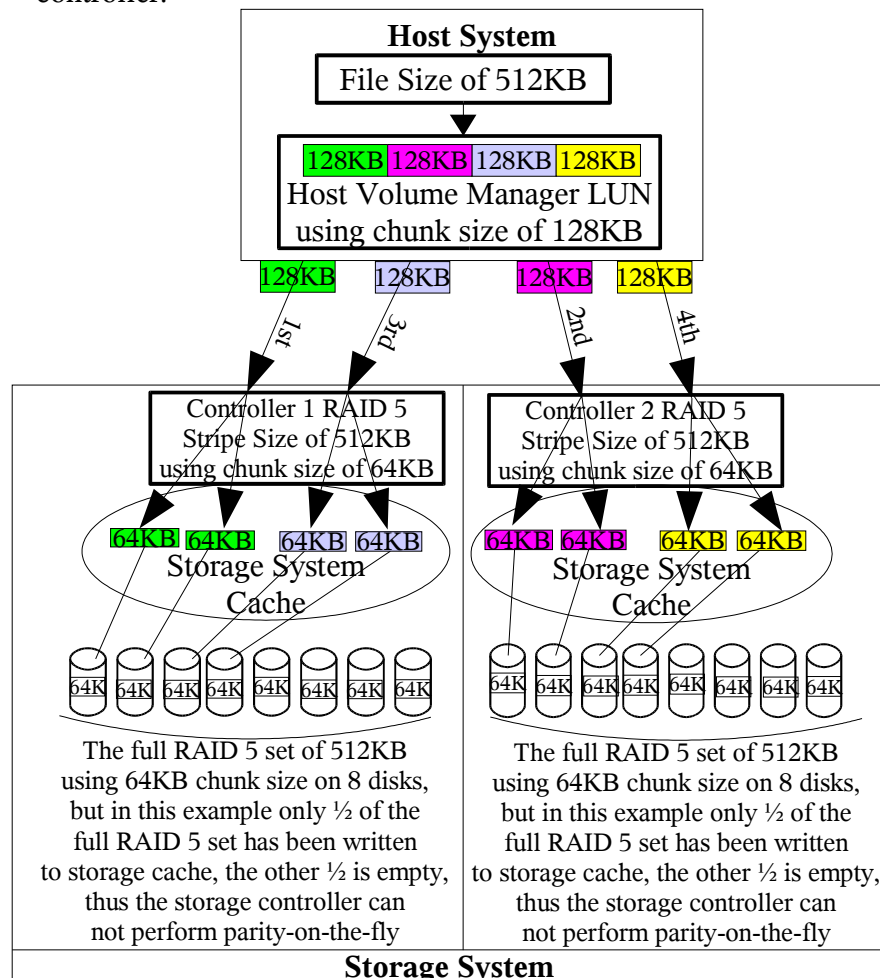


Illustration 13 - Host Stripe Chunk Size Is Too Small

In this particular example of writing a 512K file to the storage system, the **host** LUN chunk size of 128K being written to disk does not fill up the entire storage controller stripe of 8 disks, again where each disk has a 64K chunk size. Further, because the full storage controller stripe size of 512K is not present on the storage controller at the time of the write request parity-on-the-fly can not occur, which can improve storage system performance.

Using small host volume manager striping, it is difficult to get a full storage controller stripe in the controller cache. If a workload is sequential or skip sequential, small host striping will reduce the chances of invoking the controller RAID 5 “parity on the fly”, which improves performance. Our recommendation is to configure the host volume manager stripe size to 1 or 2 times larger than that of the controller stripe size. This approach assists with enabling the controller to have a full controller stripe in its cache, enabling it to perform the optimization of parity-on-the-fly, which assists with improving storage system performance.

This specific configuration also limits the ability for the array controller to do any intelligent data pre-staging. Intelligent data pre-staging can also help with improving storage system performance.

- **Beware of host-based volume stripe sizes that are too large.**

When host volume manager stripe sizes are too large, all of the data is written through one controller. This forces the storage configuration into an unbalanced or skewed situation. In other words, one controller is doing all the work while the other storage controllers sit idle.

If too large a stripe size is chosen for a host-based volume, then the potential for 100% skew can occur. Specifically, if there is a very active file it is most likely that it will reside on a single controller, thus forcing all its I/O's to funnel through that controller causing an unbalanced load, as discussed in Scenario #3 above. For example, if a file of 16MB in 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 does all the work while all other controllers

remain inactive, as shown in Illustration 14 below.

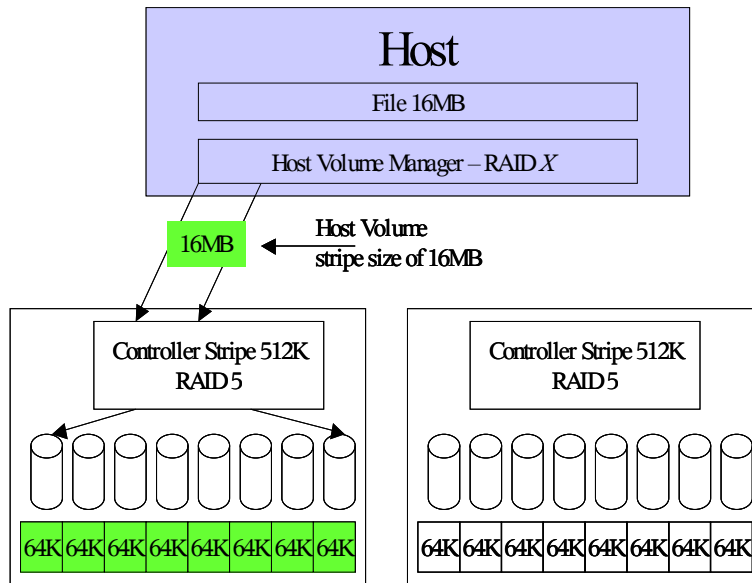


Illustration 14 Host Volume Manger Configured With a Host Stripe Size that is Too Large

- **Volume Configuration Recommendations**

In order to increase the efficiency of how the host and storage controller volume mangers work together, it is recommended to configure them using the following guidelines:

- Storage Controller RAID 5: The host volume manager stripe size, regardless of the RAID level, should be configured using sizes between 2 - 4 times larger than the storage controller stripe size as shown in Illustration 15 below.

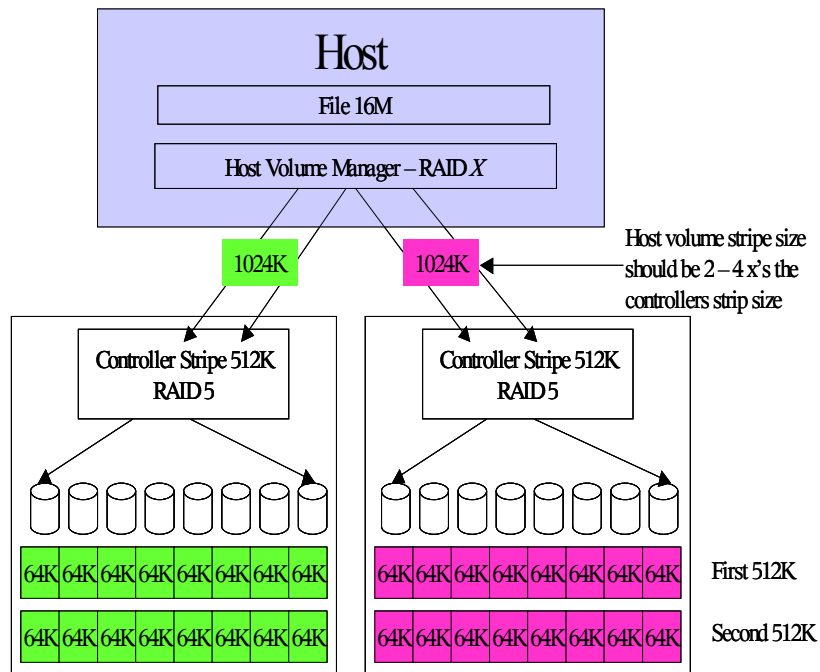


Illustration 15 RAID 5 Configuration Recommendation

With small host volume manager striping, it is difficult to get a full host volume manager stripe in the storage controller cache. If a workload is sequential or skip sequential, small host striping will reduce the chances of invoking the controller RAID 5 “parity on the fly”, which improves performance. In this specific example, host stripe size should have been configured to between 1-2 MBs, or 2 - 4 times larger than the 512K controller stripe size.

This specific configuration also limits the ability for the array controller to do any intelligent data pre-staging, which can also improve performance.

- Storage Controller RAID 0 and 1: The host volume manager stripe size, regardless of the RAID level, should be configured using sizes between 1 - 2 times larger than the storage controller stripe size as show in Illustration 16 below.

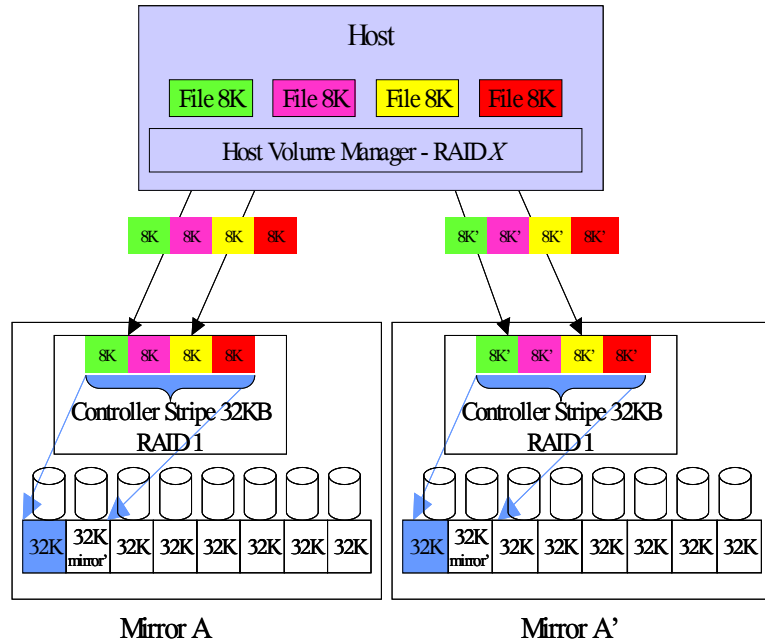


Illustration 16 RAID 0 and 1 Configuration Recommendation

To help improve RAID 1 storage performance, it is recommended to configure the host volume manager with sizes that are 1 – 2 times larger than the storage controller stripe size. In the example above, as 32KB worth of sequential data is collected on the storage controller, the controller coalesces the data into 32KB chunks. Then writes the 32KB of data to disk. This type of write approach is more efficient than trying to write each 8KB worth of data separately, and is why this RAID 1 configuration is suggested.

Solution Summary for Scenario #3:

To improve the performance, 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 lies.

Their workload skew can be evaluated by tracing I/O activity with SWAT as discussed in Scenario #2, and evaluating which controllers are handling the majority of 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 optimal end-to-end performance from its computing solutions. To this end, it is essential to configure solutions to meet business application workload requirements.

Diminishing storage system bottlenecks should help improve overall end-to-end performance, which can assist in meeting 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. 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 workload requirements.
3. Increase storage system throughput by distributing a workload across the storage system.

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!