RAID Chunk Size

xyratex

Notices

The information in this document is subject to change without notice.

While every effort has been made to ensure that all information in this document is accurate, Xyratex accepts no liability for any errors that may arise.

Acknowledgements

Jerry Hoetger, Director, Product Management, Xyratex International Ltd.

Nigel Hart, Director CPD Engineering, Xyratex International Ltd.

Stuart Campbell, Lead Firmware Architect and author of the Xyratex RAID firmware, Cork, Ireland, Xyratex International Ltd.

Xyratex CPD Engineering, Xyratex Lake Mary, FL and Xyratex Naperville, IL

Issue 1.0 | January, 2008

xyratex

Contents

xyratex

Executive Summary

In general, application workloads benefit from large chunk sizes. The most significant reason for using large chunk sizes today in configuring RAID arrays for high throughput rates and performance is the recent advancements in linear transfer rates of hard disk drives. The most common of drive sizes, 3 ½" SCSI and Serial ATA have scaled from 36GB to up to 1TB in today's SATA drives. This phenomenal increase in capacity came about by increasing the amount of data transferred while the platter rotated under the head/actuator assembly. RAID systems can now transfer much more data in the single rotation of a platter without the response penalty of a seek operation. Configuration of RAID arrays for high throughput rates and performance need to consider the increased performance of large chunk (or what some call "strip" sizes) RAID arrays.

Other contributing RAID features support larger chunk sizes: Write-back cache capabilities, coalescing of stripe writes, segmentation of data transfer requests into stripes, and the deployment of read-modify-write operations, optimize RAID performance with large chunk (and large stripe) sizes.

Some applications use wide arrays with between 12 and 16 drives in an array, and therefore have extra large stripes. These arrays may benefit from using the smaller chunk size options available in most controllers.

Benchmarks show that commands to RAID arrays with the smaller chunk size (and stripe size) exhibited slower command response times (latency) and lower throughput (IOPS) than the same commands with larger chunk size and larger stripe size. This effect was noted in almost all workloads and on several different RAID offerings, regardless of drive type (SAS or SATA). Increased linear transfer rate (aerial density), RAID firmware write-back cache and the ability to coalesce write commands contribute to the benefits of larger chunks and larger stripes.

Abstract

Considering the recent advances in disk drive technology, configuring RAID systems for optimum performance presents new challenges. Over the last five years, drive manufacturers have been steadily increasing the capacity of disk drives by increasing the density and linear transfer rates.

Although seek times have not improved in relation to linear transfer rates, to optimize performance it is necessary to transfer as much data as possible while the disk drive heads are in one position. We accomplish this by increasing the chunk size of the disk array. The calculations are based on the number of drives, RAID level of the array, and the type of use is intended for the array.

For example, database servers would have a different chunk size requirement, then one for a general use server or mail server.

Throughout this paper the core concepts of chuck size and full stripe writes are examined, as well as benchmark test results.

xyratex

Chunk Size

Striping and Chunks

All RAID levels used in today's storage rely on the concept of striping data across multiple disk drives in an array. The data in a stripe is usually made up of multiple records or segments of information in accordance to a layout by an application running on the host. The particular record layout is of no significance to the RAID processing. Host applications access the logical data records or segments with varying access patterns. Access patterns and configuration options defined as part of the RAID array, will impact the performance of the array.

RAID 0: Striping

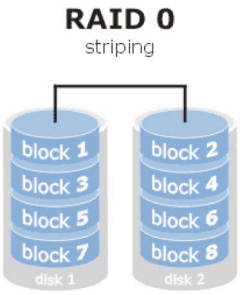

Figure 1: RAID 0 Array

In a RAID 0 system, data is split up into blocks that will get written across all the drives in the array. By using multiple disks (at least 2) at the same time, RAID 0 offers superior I/O performance. This performance can be enhanced further by using multiple controllers, ideally one controller per disk.

Advantages

- RAID 0 offers great performance to both read and write operations.
- There is no overhead caused by parity controls.
- All storage capacity can be used, there is no disk overhead. The technology is easy to implement.

Disadvantages

- RAID 0 is not fault-tolerant. If one disk fails, all data in the RAID 0 array are lost.
- It should not be used on mission-critical systems.

Ideal Use

RAID 0 is ideal for non-critical storage of data that has to be read from or written to a workstation; for example: a workstation using Adobe Photoshop for image processing.

RAID 1: Mirroring

Data is stored twice by writing to both the data disk (or sets of data disks) and a mirror disk (or sets of disks). If a disk fails, the controller uses either the data disk or the mirror disk for data recovery, and continues operation. You need at least two disks for a RAID 1 array.

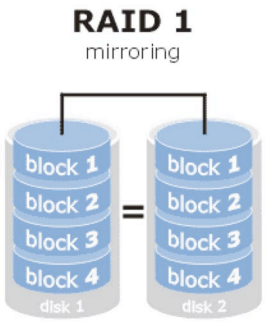

Figure 2: RAID 1 Array

RAID level 1 is often combined with RAID level 0 to improve performance. This RAID level is sometimes referred to as RAID level 10.

Advantages

- RAID 1 offers excellent read speed and a write speed, that is comparable to that of a single disk.
- In case a disk fails, data does not have to be rebuilt, it just has to be copied to the replacement disk.
- RAID 1 is a very simple technology.

Disadvantages

- The main disadvantage is that the effective storage capacity is only half of the total disk capacity because all data gets written twice.
- Software RAID 1 solutions do not always allow a hot swap of a failed disk (meaning it cannot be replaced while the server continues to operate). Ideally a hardware controller is used.

Ideal use

RAID 1 is ideal for mission critical storage, for instance with accounting systems. It is also suitable for small servers, in which only two disks will be used.

RAID 5

RAID 5 is the most common secure RAID level. It is similar to RAID 3 except that data is transferred to disks by independent read and write operations (not in parallel). The data chunks that are written are also larger. Instead of a dedicated parity disk, parity information is spread across all the drives. You need at least 3 disks for a RAID 5 array.

A RAID 5 array can withstand a single disk failure without losing data or access to data. Although RAID 5 can be achieved in software, a hardware controller is recommended. Often extra cache memory is used on these controllers to improve the write performance.

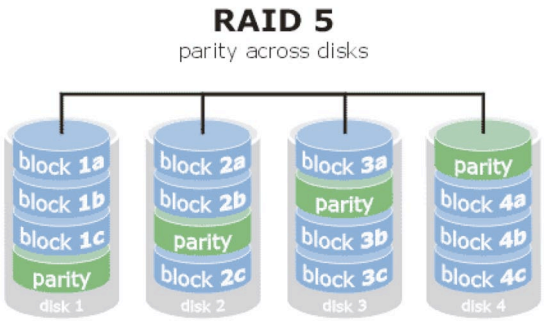

Figure 3: RAID 5 Array

Advantages

Read data transactions are very fast, while write data transaction are somewhat slower (due to the parity that has to be calculated).

Disadvantages

- Disk failures have an effect on throughput, although this is still acceptable.
- Like RAID 3, this is complex technology.

Ideal use

RAID 5 is a good all-round system that combines efficient storage with excellent security and decent performance. It is ideal for file and application servers.

RAID 10: A Mix of RAID 0 & RAID 1

RAID 10 combines the advantages (and disadvantages) of RAID 0 and RAID 1 in a single system. It provides security by mirroring all data on a secondary set of disks (disk 3 and 4 in the Figure below) while using striping across each set of disks to speed up data transfers.

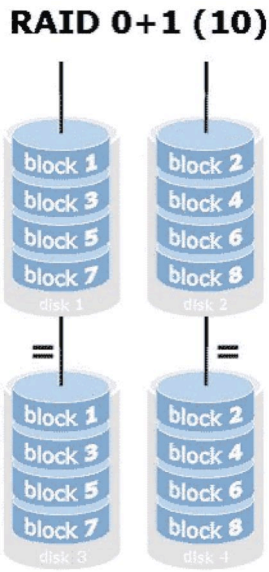

Figure 4: RAID 10 Array

Stripe

The aggregation of records or segments is commonly called a stripe and represents a common object in the array that is maintained by the firmware in the controller, based on commands from the host application. For each stripe read to or written (full or partial) from the RAID firmware, multiple I/O requests are made to each member drive in the array depending on the RAID type and number of drives in the array.

For RAID 0 and 10, no read-modify-write is performed, so there is a single I/O request to a member drive for each host I/O, which can be reduced by coalescing. For RAID 5/50/6, a host read command translates to a single I/O to a single drive, not multiple I/Os to the same drives. For a updating (read-modify-write), there are multiple I/Os to the data and parity drive(s). For a full stripe write, there is a single I/O for each drive. These requests are made in parallel. This particular paralleling aspect of full stripe writes in RAID striping reduces command response time (latency) and increases I/O commands per second (IOPS) for hosts writing to the array sequentially.

Adjusting for Performance and Chunk Size

System administrators have options when defining the RAID systems used for storing and retrieving data by applications. Each application has a unique pattern for storing and retrieving the data based on how the application works, and how users of the application make requests. This pattern of reading and writing data along with the system administrator specified options affect the performance of the application.

Chunk size is one of the configuration options, where the size of the stripe is apportioned to the individual disk drive. Some RAID offerings let the system administrator define the size of the stripe across all of the member drives (and derive the size of the chunk written to individual drives based on the number of member drives). Other RAID offerings have the system administrator configure the chunk or strip size (and derive the larger stripe size based on the number of member drives in the array). In either case the chunk size has performance implications.

Writing and Reading Chunks and Stripes

RAID controller firmware reads and writes stripes to member drives in an array. Writes can be either full stripe or partial stripe write operations (See Read-Modify-Write Section for more information). System Administrator's stipulate a stripe size (and therefore derive chunk size), or stipulate the chunk size (and therefore derive stripe size). Full stripe write operations stripe across all member drives.

Partial stripe operations read and write only to the affected portions of the stripe (usually one data drive and one parity drive). Read operations read only the amount of data associated with the host command. The presence of Read-Ahead capabilities in a RAID controller may modify the amount of data read beyond that which is required to satisfy the host command, in preparation of subsequent read commands.

For RAID levels with parity and parity chunks are maintained on writes. Different RAID firmware handles reads and writes somewhat differently, depending on the vendor and the RAID firmware strategy on full stripe reads and writes.

Writes can be handled in slightly different ways, depending on the vendor RAID strategy. Common in early RAID deployments, vendor RAID strategies stipulated that with writes and updates, the complete stripe was first read from all member drives in an array, locked for update with appropriate data and parity, verified, then completely striped across all member drives in the array with updated data and parity, at which time the lock for that stripe would be released. Xyratex RAID deploys a "read-modify-write" strategy where only the portion of the stripe written to and the portion of the parity chunk updated are transmitted to member disk drives as part of the write or update operation. This mitigates the performance implications of alternative full stripe writes.

Read-Modify-Write for Writing Stripes [1]

Xyratex has implemented "read-modify-write" for writing stripes. Only that portion of the stripe (and parity chunks) is written to the array for servicing writes. This mitigates the performance implication of full stripe writes for random workloads, and is one of the reasons systems administrators can use larger chunks and stripes for all workloads. [2]

The write operation is responsible for generating and maintaining parity data. This function is typically referred to as a read-modify-write operation. Consider a stripe composed of four chunks of data and one chunk of parity. Suppose the host wants to change just a small amount of data that takes up the space on only one chunk within the stripe. The RAID firmware cannot simply write that small portion of data and consider the request complete. It also must update the parity data, which is calculated by performing XOR operations on every chunk within the stripe. So parity must be recalculated when one or more stripes change.

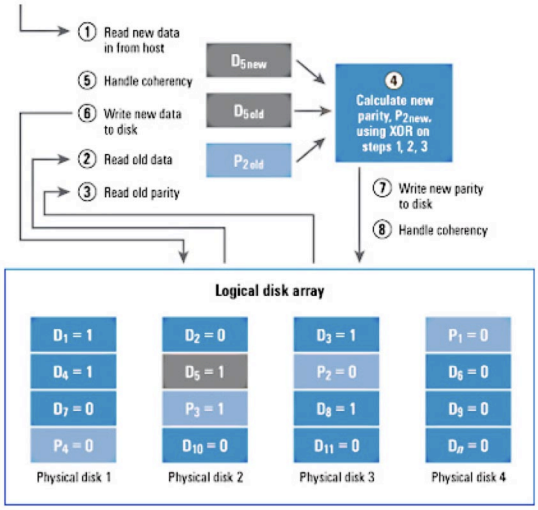

Figure 5: Step-by-Step: Read-Modify-Write Operation to a Four-Disk RAID Array

---

1 Dell Power Solutions, "Understanding RAID-5 and I/O Processors", Paul Luse, May 2003

2 Full stripe writes for sequential write workloads is faster than Read-Modify-Write commands. If the application can fill up a stripe completely, overall full stripe writes perform faster.

Figure 5 shows a typical read-modify-write operation in which the data that the host is writing to disk is contained within just one strip, in position D5. The read-modify-write operation consists of the following steps:

1. **Read new data from host:** The host operating system requests that the RAID subsystem write a piece of data to location D5 on disk 2.
2. **Read old data from target disk for new data:** Reading only the data in the location that is about to be written to eliminates the need to read all the other disks. The number of steps involved in the read-modify-write operation is the same regardless of the number of disks in the array.
3. **Read old parity from target stripe for new data:** A read operation retrieves the old parity. This function is independent of the number of physical disks in the array.
4. **Calculate new parity by performing an XOR operation on the data from steps 1, 2, and 3:** The XOR calculation of steps 2 and 3 provides the resultant parity of the stripe, minus the contribution of the data that is about to be overwritten. To determine new parity for the D5 stripe that contains the new data, an XOR calculation is performed on the new data read from the host in step 1 with the result of the XOR procedure performed in steps 2 and 3.
5. **Handle coherency:** This process is not detailed in Figure 5 because its implementation varies greatly from vendor to vendor. Ensuring coherency involves monitoring the write operations that occur from the start of step 6 to the end of step 7. For the disk array to be considered coherent, or "clean," the subsystem must ensure that the parity data block is always current for the data on the stripe. Because it is not possible to guarantee that the new target data and the new parity will be written to separate disks at exactly the same instant, the RAID subsystem must identify the stripe being processed as inconsistent, or "dirty," in RAID vernacular.
6. **Write new data to target location:** The new data was received from the host in step 1; now the RAID mappings determine on which physical disk, and where on the disk, the data will be written. With Xyratex RAID only the portion of the chunk updated will be written, not the complete chunk.
7. **Write new parity:** The new parity was calculated in step 4; now the RAID subsystem writes it to disk.
8. **Handle coherency:** Once the RAID subsystem verifies that steps 6 and 7 have been completed successfully, and the data and parity are both on disk—the stripe is considered coherent.

This optimized method is fully scalable. The number of read, write, and XOR operations is independent of the number of disks in the array. Because the parity disk is involved in every write operation (steps 6 and 7), parity is rotated to a different disk with each stripe. If all the parity were stored on the same disk all the time, that disk could become a performance bottleneck.

Coalesce

Coalescing is the process of aggregating several command requests together to form a single and larger sequential access on the member drives in an array. Coalesce represents a level of buffering commands to the member drives of an array. In this way, multiple write commands and associated data can be aggregated into a stripe. A single large sequential access reduces the number of disk commands and data transfers significantly, thus greatly increasing the amount of host I/O that can be processed in any specific time interval.

Drive level coalescing by the RAID controller significantly contributes to the overall performance of an array and is very dependent on the access patterns of the application, as well as the algorithms of the RAID offering. Sequential applications with high queue depths leverage the improved performance of write coalescing more than applications with low queue depths and are more random. With a queue depth of one and in write-through mode, only one write command will be issued at a time. The host will wait to receive an acknowledgement that an initial write completes before issuing subsequent write commands. The RAID firmware will not have an opportunity to coalesce, issuing an individual write operation for each write command received. The use of writeback cache settings will allow the controller to acknowledge the writeback to the host applications while the data associated from the command to be coalesced into a common stripe.

As the application (and the host) exhibit writing multiple outstanding write commands, coalescing can increasingly improve performance of the array, regardless of write-through or writeback cache setting. Each data transfer write request potentially requires a corresponding write operation of the stripe, with individual reads and writes to each member drive of the array (read-modify-write), or two I/O operations per drive required for each write. Some write operations that span stripes might require three (not aligned with stripe boundaries). With each extra outstanding write operations per drive (queue depth), the RAID firmware has an increasing chance of grouping such commands together in a single sequential data transfer.

The size of the logical drive will also impact the affect of coalescing random workloads. Larger logical drives that span the full range of the disk platters of member drives in the array will require longer command response times for coalesced random commands than smaller logical drives that are limited to just several cylinders of each set of drives' disk platters. The smaller range of cylinders will limit the effect of the randomness of the commands, reporting an almost sequential like application performance.

RAID Cache Options

Many RAID offerings today offer the option of battery backed cache and an associated "writeback" mode of operation where, for write commands, the host is acknowledged immediately after the data transfer is received in to the cache of the RAID system. In the rare event of a power failure, a hardware failure in the controller, switch, attached cables, or a failure with the RAID firmware, the record stays in cache until it can be completely externalized as part of a recovery process. This mode of operation reduces the response time (or latency) associated with the command. This mode of operation also reduces the throughput of the controller

due to the mirroring of cache operations between the two controllers in a "dual controller configuration." This mirroring operation is usually done through dedicated channels on the base plane of the controller.

Another option found in most of today's RAID systems allows the system administrator to turn off the RAID system cache for writes. Many large data transfer applications can get higher throughput (MB/s) in this mode as records are externalized directly to the member drives in an array, without relying on the RAID cache. This is called "write-through" mode.

Some RAID offerings allow the system administrator to partition the cache for writes protecting performance of other applications sharing the RAID system cache and to turn off the cache mirroring, while in "writeback" mode. The RAID system needs to revert to write-through when cache capacity is unavailable to commands. Partitioning of cache defines the threshold for transitioning, as well as conserves cache for servicing commands to other arrays. Turning off cache mirroring provides the command response time benefits of cached writes and fast response times, while sacrificing fault-tolerance.

Performance Implications of Configuration Options

Many factors impact the performance of an array and thus the performance of host based applications that access one or more arrays. Significant factors include:

1. Data transfer size to and from the host. This is usually set by the application.
2. Access pattern to the array by the host or hosts (concurrency of multiple hosts, sequential, random, random-sequential, queue depth).
3. RAID level.
4. RAID cache for writes (Writeback Mode or Write-Through Mode, amount of cache allocated for writes).
5. Chunk size and therefore Stripe size.
6. Number of member drives in the array.
7. Drive performance.

The system administrator usually has options for configuring RAID level, RAID cache operations, chunk size, number of member drives and drive type (affecting drive performance). The application usually dictates data transfer size and access pattern, although this can be tuned with some operating systems.

Furthermore, vendor RAID strategies will provide differing options to choose from for configuring arrays. Some RAID vendors let system administrators set chunk size, others stripe size. With either approach, the vendor offering will provide a list of available options ranging from as small as 4K to up to 256K chunk sizes. Each vendor strategy includes a selection of RAID levels and has slightly differing strategies for implementing each. The number of drives can vary, but usually ranges from 1 for simple RAID 0 to 16. RAID vendors support a variety of different drive types today ranging from Serial Attached SCSI (SAS), Serial ATA (SATA), Fibre Channel (FC), and SCSI, each with their own performance characteristics and capacities. Usually an array is limited to include a single drive type and capacity.

Given the available options to the system administrator, several general recommendations can be made relative to the required performance of host commands to the array based on the access patterns and data transfer size dictated by the host application.

Large Data Transfer - Sequential Reading and Writing Access Pattern

This workload will have data transfer sizes of at least 32K and more typically 1MB or larger. A special case can be made for workloads that read and write larger than 1MB to the array. Some vendor RAID strategies may be limited to 1MB or perform at less than optimally when support greater host transfer sizes. The access pattern is predominantly sequential, read, write or alternating between read and write for very long intervals of time.

Especially important for sequential write workloads is the ability to fill up stripes with data to maximize performance (the effect of full stripe writes and stripe coalesce). Maintaining enough application data to fill up stripes with write sequential access patterns can happen in several ways:

1. Large data transfer sizes from the host. Some applications such as video streaming allow for configurable data transfer size. Ideally the applications data transfer size should be aligned with the stripe size.
2. The application can issue multiple outstanding commands to build queues of commands that can be sequentially executed to fill up stripes by the RAID controller across the array.
3. Writeback cache setting can fill up stripes from applications that maintain low or non-existent queue depths of outstanding commands.

When the application fills up the stripe during write sequences, large chunks result in maximum throughput capabilities. Larger data transfer sizes and large stripes facilitate segmenting of small records into larger data transfers and multiple data transfers into larger stripes and even larger coalesced data transfers to member disk drives in the array. If the application cannot fill up the stripe, the system administrator should consider smaller stripes either via fewer drives in the array or choosing smaller chunk size.

Many large data transfer applications perform better with write cache disabled, or when the controller is in Write-through mode. Write-through mode optimizes the RAID performance for high throughput rates (MB/s) by bypassing cache coherency operations and writing aggregated streams of data to member drives in an array. Writeback cache should only be considered for small data transfer applications for filling up stripe writes.

Command response time (latency) or a high number of commands per second (IOPS) are typically not important to these workloads. The data transfers are large or very large, and will by nature take longer to service as they are striped across many drives in an array. Typical workloads in this category include disk-to-disk backup, Virtual tape library, archiving applications, and some network file system deployments.

Small Data Transfer – Random Reading and Writing Access Pattern

This work load will typically have data transfer sizes of less than 32K and more typically 4K or 8K in size. The access pattern is predominantly random reads or writes, or intermixed with updates. These workloads typically do not maintain much of a queue of outstanding commands to the array, usually less than 6. The depth of the queue is very application dependent and can very depending on the number of threads the application can generate, number of processors, the number of clients supported or other programming constructs used.

In this case, Writeback cache operations result in minimum response time and maximum number of commands per second (IOPS) from the array, regardless of the chunk size and corresponding stripe size. In Writeback mode, larger cache sizes will increase the likelihood of filling up stripes to leverage the effect of full stripe writes. Throughput is typically not important to these workloads. Typical workloads in this category include database, transactional systems, web servers, and many file servers.

ESG Defined Workloads and Associated Access Patterns

Here is a very detailed description of access patterns by workload used in benchmarking array performance by ESG. The queue depth is fixed at 16K, which may not be typical for most data base or transaction workloads.

Application Workloads					
4K OLTP					
4k transfer size	100%	Access Specification	33% writes	67% reads	100% random
File Server					
512B transfersize	10%	Access Specification	20% writes	80% reads	100% random
1k transfer size	5%	Access Specification	20% writes	80% reads	100% random
2k transfer size	5%	Access Specification	20% writes	80% reads	100% random
4k transfer size	60%	Access Specification	20% writes	80% reads	100% random
8k transfer size	2%	Access Specification	20% writes	80% reads	100% random
16k transfer size	4%	Access Specification	20% writes	80% reads	100% random
32k transfer size	4%	Access Specification	20% writes	80% reads	100% random
64k transfer size	10%	Access Specification	20% writes	80% reads	100% random
Web Server					
512B transfersize	22%	Access Specification	0% writes	100% reads	100% random
1k transfer size	15%	Access Specification	0% writes	100% reads	100% random
2k transfer size	8%	Access Specification	0% writes	100% reads	100% random
4k transfer size	23%	Access Specification	0% writes	100% reads	100% random
8k transfer size	15%	Access Specification	0% writes	100% reads	100% random
16k transfer size	2%	Access Specification	0% writes	100% reads	100% random
32k transfer size	6%	Access Specification	0% writes	100% reads	100% random
64k transfer size	7%	Access Specification	0% writes	100% reads	100% random
128k transfer size	1%	Access Specification	0% writes	100% reads	100% random
512k transfer size	1%	Access Specification	0% writes	100% reads	100% random
Media Server					
32k transfer size	100%	Access Specification	0% writes	100% reads	0% random	100% Sequential
Exchange data					
4k transfer size	90%	Access Specification	27% writes	73% reads	100% random	0% Sequential
4k transfer size	7%	Access Specification	0% writes	100% reads	0% random	100% Sequential
4k transfer size	3%	Access Specification	100% writes	0% reads	0% random	100% Sequential
Exchange logs					
64k transfer size	100%	Access Specification	100% writes	0% reads	0% random	100% Sequential
Windows Media Player					
32k transfer size	100%	Access Specification	0% writes	100% reads	0% random	100% Sequential

Figure 6: ESG Defined Workloads

Performance of Arrays and Varying Chunk Sizes [3]

Xyratex CDP Engineering used ESG specified workloads, specially modified versions of the latest Xyratex RAID offerings, and a recent DotHill RAID offering to test the implications of varying chunk sizes on fixed and repeatable workloads (Model RS-1220-F4-5412E and Model RS-1220-F4-5402E with 1Gb of cache per controller from Xyratex and the Model 2730 from DotHill).

Xyratex used Intel's IOMeter to simulate the workload patterns as specified by ESG as typical for various applications. See the table below for a detailed description of each workload. The configuration consisted of one system using a Microsoft Windows 2003 host with fibre channel interface connected (at 4Gb/s) to the Xyratex RAID system configured with a single RAID 5 (5+1) array, and single LUN externalized to the host. Seagate X15-5 SAS drives and Seagate Barracuda SATA-II drives were used. The RAID systems were configured with dual controllers and cache mirroring in effect, even though only one controller was active.

The tests were run with IOMeter against this common single array/single LUN configuration, against each of the ESG workload categories, and with the following different combinations of platform and chunk size.

RAID Offering	Chunk Size	Drive Type
Xyratex RS-1220-F4-5402E	16K	SAS
Xyratex RS-1220-F4-5402E	64K	SAS
Xyratex RS-1220-F4-5412E	16K	SAS
Xyratex RS-1220-F4-5412E	64K	SAS
DotHill 2730	16K	SAS
DotHill 2730	64K	SAS
Xyratex RS-1220-F4-5402E	16K	SATA
Xyratex RS-1220-F4-5402E	64K	SATA
Xyratex RS-1220-F4-5412E	16K	SATA
Xyratex RS-1220-F4-5412E	64K	SATA
DotHill 2730	16K	SATA
DotHill 2730	64K	SATA

Both 16K and 64K chunk sizes where common to both DotHill and Xyratex, and therefore used for comparison purposes. Xyratex RAID systems offer chunk size options of 64K, 128K, and 256K, and derives the stripe size depending on the number of drives in the array multiplied times the selected chunk size.

DotHill offers 16K, 32K, and 64K chunk sizes. Since these are lower then the chunk sizes available on Xyratex RAID systems, Xyratex created a special set of RAID firmware to enable identical configuration options for one-to-one comparisons.

---

3 Nigel Hart. Xyratex CDP Engineering Manager. "ESG Chunk Effects.xls". dated 3 July 2007.

For a 16K chunk size the derived stripe size is 96K. For a 32K chunk size the derived stripe size is 192K. If chunk size made a difference, it would be reflected in lower command response times (latency) and higher command response throughput (IOPS) in these tests.

The RS-1220-F4-5412E is the current shipping version of RAID system from Xyratex. The previous model, the RS-1220-F4-5402E, used an older version of the LSI 1064 SAS controller and Intel RAID ASIC (IOP331). That model was included only for comparison purposes.

Test Results

Both Xyratex and DotHill RAID with the smaller chunk size (16K) and stripe size (96K) exhibited slower command response times (latency) and lower throughput (IOPS) than the same RAID offering with larger chunk size (64K) and stripe size (192K). This effect was noted in almost all workloads and RAID offerings except one DotHill test case, regardless of drive type (SAS or SATA).

SAS Drive Tests

SAS 64K chunk to 16K	5402	5412	2730
4K OLTP	-11%	-13%	-5%
File Server	-19%	-19%	-14%
Web Server	-20%	-20%	-20%
4 KB random Read	-10%	-10%	-10%
4 KB random Write	-15%	-9%	2%
512 KB Seq Write	-37%	-37%	-51%
512 KB Seq Read	-21%	-21%	-11%
Media Server	-26%	-26%	-35%
Exchange Data	-10%	-10%	-5%
Exchange Logs	-41%	-42%	-33%
File Server	-19%	-20%	-15%
Web Server	-20%	-20%	-21%
Windows Media Player	-26%	-26%	-36%
Exchange Data (EBD)	-10%	-10%	-4%
Exchange Logs (LOG)	-41%	-42%	-31%

Relative percent calculations based the difference in total IOPS between 64K and 16K chunk sizes for each workload. Negative numbers indicate that the 16K chunk size performed with less IOPS than the 64K chunk size test case, in relative terms.

Sequential workloads performed significantly less with smaller chunk sizes (and smaller stripe sizes) than random workloads. Write workloads performed significantly less with smaller chunk sizes (and smaller stripe sizes) than read workloads. This affect is attributed to the lack of 1st level segmentation within a stripe with random and

write workloads. Where as sequential workloads benefit from both 1st level segmentation within the stripe (with larger stripes) and 2nd level coalescing of multiple stripes at a queue depth of 16.

SATA Drive Tests

SATA 64K chunk to 16K	5402	5412	2730
4K OLTP	-10%	-10%	-7%
File Server	-21%	-22%	-18%
Web Server	-22%	-21%	-23%
4 KB random read	-12%	-10%	-11%
4 KB random write	-13%	-11%	-3%
512 KB Seq Write	-82%	-58%	13%
512 KB Seq Read	-10%	-10%	-1%
Media Server	-15%	-16%	-32%
Exchange data	-10%	-8%	-19%
Exchange Logs	-49%	-76%	-43%
File Server	-20%	-22%	-19%
Web Server	-22%	-21%	-23%
Windows Media Player	-15%	-16%	-33%
Exchange data (EBD)	-10%	-8%	-8%
Exchange Logs (LOG)	-49%	-75%	-7%

Relative percent calculations based the difference in total IOPS between 64K and 16K chunk sizes for each workload. Negative numbers indicate that the 16K chunk size performed with less IOPS than the 64K chunk size test case, in relative terms.

Similar differences between random and sequential workloads, and SAS drives are seen with SATA drives. In general, the relative differences with sequential workloads are exacerbated with slower SATA drive performance.

Test Conclusions

Smaller chunk size does not improve latency or increase IOPS for any of the ESG workloads, at the specified queue depth (16) and for the specified RAID offerings, other than one test case with the DotHill RAID controller. The ESG defined workloads are mostly random and small data transfer size workloads, where system administrators would typically assign small chunk sizes or stripe sizes to improve performance and increase IOPS. For the sequential workloads (Media server, Exchange data, Exchange logs, and Media player), smaller chunk size did impact command response time (latency) and reduce IOPS.

The affect of the very fast linear transfer rates with today's disk drives translates to larger chunks for high RAID performance. The use of a small chunk size, 16K, did not improve performance when compared to a larger 64K chunk.

System administrators need to configure external RAID systems such that full stripe reads and writes leverage these fast linear transfer rates by handling as much data transfer as possible without incurring the penalty of slow seek times. Today's hard disk drives have very large track capacities. The Barracuda ES family of Serial ATA drives used in this study from Seagate has an average track size of 32K. [4] The largest capacity Barracuda ES drive has 8 tracks per cylinder, translating to 256K per data transfer between seeks. The smallest Barracuda ES drive has 3 tracks per cylinder, translating to 96K per data transfer between seeks.

Coalescing of data transfers with the writing of data in to stripes also helps produce higher performance and IOPS of the larger chunk size. Coalescing is the process of aggregating several command requests together to form a single and larger sequential access on the member drives in an array. A single large sequential access reduces the number of disk commands and data transfers significantly, thus greatly increasing the amount of host I/O that can be processed in any specific time interval. RAID in this mode of operation, becomes constrained by speed of a chunk update alone. A high queue depth used in the sixteen test cases contributed to the affect of coalescing on the command response times (latency) and throughput (IOPS).

Expect application workloads with lower queue depths (from 4 to 16) to also benefit from larger chunk sizes and stripe sizes, but to a lesser extent than reported with a queue depth of 16. Applications with queue depth approaching 1 or 2 may benefit from smaller chunk sizes (and smaller stripes) as the total data transfer operation is reduced in size with fewer commands being aggregated into larger single sequential accesses.

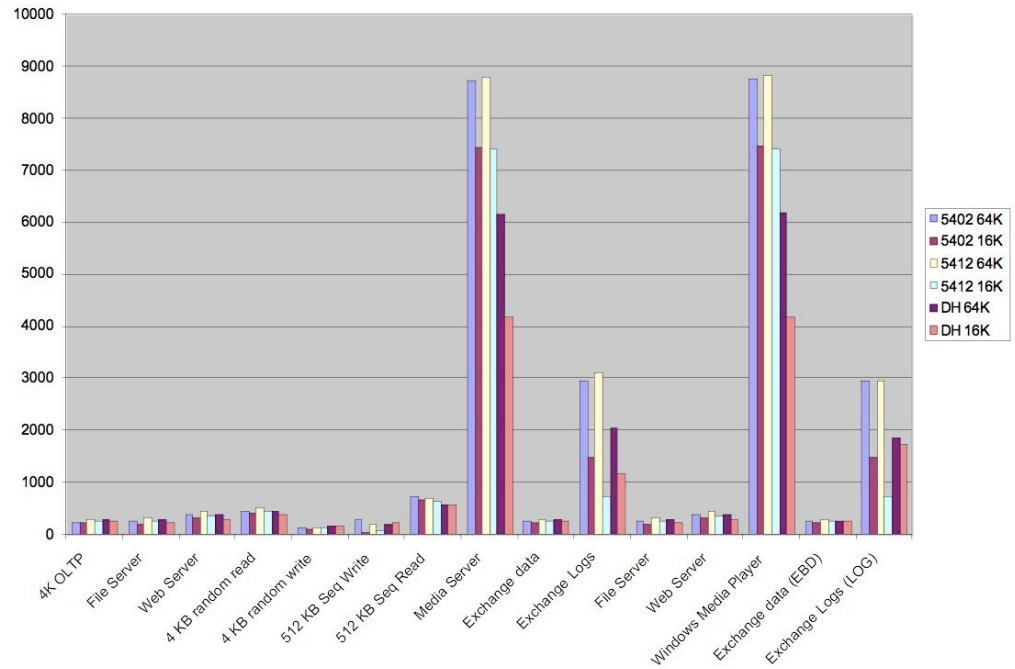

Figure 7: Number of I/O Operations/Sec by Application Workload for 16K and 64K Chunk Sizes

4 Track size will vary, depending on the location of the cylinder. Cylinders towards the outside of the drive will have larger tracks because of the size of the cylinder, versus cylinders towards the inside of the drive, which are smaller. Seagate Barracuda ES Serial ATA Product Manual, Revision A, 6/26/06

A Case for Smaller Chunk Size with Wide Arrays

For performance reasons systems administrators may choose to configure a large number of drives in a single array. This configuration maximizes the parallelism of RAID by breaking up a large single data transfer into multiple and parallel disk commands with smaller data transfers to each drive. Some vendor RAID offerings allow the system administrator to choose from a range of chunk size. In some cases a chunk size less than the maximum available may make since to reduce the overall stripe size of the array.

For example: a 12 drive RAID 5 array with 256K chunks will result in an array with a 3MB stripe. Applications with typically small data transfer requests, especially when made in a random fashion, may perform better with a smaller stripe. The small data transfer size or randomness of the write requests may limit the affect of writeback cache, coalescing, read-modify-write and segmentation at the stripe level. Performance may improve by specify a 64K chunk (providing a 768K stripe) or a 128K chunk (providing a 1.5MB stripe).

Smaller chunks and smaller stripes should definitely be considered for workloads against wide arrays that require disabling writeback cache (write-through mode).

Glossary

RAID: (Redundant Array of Independent Disks) A disk subsystem that is used to increase performance or provide fault tolerance or both. RAID uses two or more ordinary hard disks and a RAID disk controller. In the past, RAID has also been implemented via software only.

In the late 1980's, the term stood for "redundant array of inexpensive disks," being compared to large, expensive disks at the time. As hard disks became cheaper, the RAID Advisory Board changed "inexpensive" to "independent."

Small and Large: RAID subsystems come in all sizes from desktop units to floor-standing models (see NAS and SAN). Stand-alone units may include large amounts of cache as well as redundant power supplies. Initially used with servers, desktop PCs are increasingly being retrofitted by adding a RAID controller and extra IDE or SCSI disks. Newer motherboards often have RAID controllers.

Disk Striping: RAID improves performance by disk striping, which interleaves bytes or groups of bytes across multiple drives, so more than one disk is reading and writing simultaneously.

Mirroring and Parity: Fault tolerance is achieved by mirroring or parity. Mirroring is 100% duplication of the data on two drives (RAID 1). Parity is used to calculate the data in two drives and store the results on a third (RAID 3 or 5). After a failed drive is replaced, the RAID controller automatically rebuilds the lost data from the other two. RAID systems may have a spare drive (hot spare) ready and waiting to be the replacement for a drive that fails.

The parity calculation is performed in the following manner: a bit from drive 1 is XOR'd with a bit from drive 2, and the result bit is stored on drive 3 (see OR for an explanation of XOR).

RAID 0: Speed (Widely Used)

RAID level 0 is disk striping only, which interleaves data across multiple disks for performance. Widely used for gaming, RAID 0 has no safeguards against failure.

RAID 1: Fault Tolerance (Widely Used)

Uses disk mirroring, which provides 100% duplication of data. Offers highest reliability, but doubles storage cost. RAID 1 is widely used in business applications.

RAID 2: Speed

Instead of single bytes or groups of bytes (blocks), bits are interleaved (striped) across many disks. The Connection Machine used this technique, but this is rarely used because 39 disks are required.

RAID 3: Speed and Fault Tolerance

Data are striped across three or more drives. Used to achieve the highest data transfer, because all drives operate in parallel. Using byte level striping, parity bits are stored on separate, dedicated drives.

RAID 4: Speed and Fault Tolerance

Similar to RAID 3, but uses block level striping. Not often used.

RAID 5: Speed and Fault Tolerance (Widely Used)

Data are striped across three or more drives for performance, and parity bits are used for fault tolerance. The parity bits from two drives are stored on a third drive and are interspersed with user data. RAID 5 is widely used in servers.

RAID 6: Speed and Fault Tolerance

Highest reliability because it can recover from a failure of two disks, but not widely used. Similar to RAID 5, but performs two different parity computations or the same computation on overlapping subsets of the data.

RAID 10, RAID 100: Speed and Fault Tolerance

RAID 10 is RAID 1 + 0. The drives are striped for performance (RAID 0), and all striped drives are duplicated (RAID 1) for fault tolerance.

RAID 100 is RAID 10 + 0. It adds a layer of striping on top of two or more RAID 10 configurations for even more speed.

External/GUI	Other similar names	Description
Drive Array	Storage Pool, Storage Pool Group, Device Group, Sub-device Group	A group of drives that all have the same array function place on them. The array functions of the controller are RAID 0 and RAID 5 with an appropriate strip or chunk size. The array is constructed over the minimum useable capacity of all drives (smallest capacity drive extrapolated to all remaining drives in the array). The array can be broken in to logical drives that are constructed from 1 or many parts of the array. All logical drives allocated from the same drive array will have the same RAID characteristics.
Full Stripe Write		The amount of data written by the host, before parity, that will result
Logical Drive, Host Logical Drive	Volume	The logical construction of a "direct access block device" as defined by the ANSII SCSI specification. Each logical drive has a number of host logical blocks (512 bytes), resulting in the capacity of the drive. Blocks can be written to or read from. The controller enforces precedence to the order of commands as received at the controller from connected hosts. Logical drives will either be in a normal state or a failed state depending on physical state of the drive array. Task management is carried out at the logical drive level.
LUN		The protocol end point of SCSI communication. There can be multiple LUNs presented for a given logical drive, thus giving multiple paths to that logical drive. For any host port, a given logical drive can be reported only once as a LUN. The logical drive the LUN is assigned to may be a host logical drive including source or virtual logical drive (associated with a snapshot).
Overwrite Data Area, ODA	Delta area	Protected storage area on an array that has been dedicated to be used for storing data from the snapped logical drive that is overwritten (before images). In itself, the ODA is a logical drive, though not externalized.
Snap Back		The process of restoring a logical drive from a selected snapshot. This process takes place internally in the RAID controller firmware and needs no support from any backup utility or other third party software.
Snapshot	Snap, Snapshot Volume	A method for producing a point-in-time virtual copy of a logical drive. The state of all data on the virtual logical drive remains frozen in time and can be retrieved at a later point in time. In the process of initiating the snapshot, no data is actually copied from the snapped logical drive. As updates are made to the snapped logical drives, update blocks are copied to the ODA.
Snapshot Identifier	Snapshot Number	A unique number identifying the snap relationship to the source logical drive. Snapshot number is not necessarily unique to the controller, only unique to the host logical drive. Snapshots are sequentially numbered for an individual host logical drive starting at 0 through N.
Number of Snapshots		Number of total snapshots on the same host logical drive.
Source Logical Drive	Snapshotted Logical Drive	A logical drive that has a snapshot initiated on it.
Strip, Chunk		The minimum unit of storage managed on a given drive as a contiguous storage unit (usually a number of drive blocks) by the controller. The RAID controller writes data to drives in a drive array in this size.
Stripe Size		The amount of host data plus any parity at which an array function, such as a host read or write, rotates across all members of the array – the optimum unit of access of the array. Chunk size * Stripe Width = Stripe Size.
Stripe Width		The number of members in the array.
Virtual Logical Drive, Snapshot Logical Drive		A special logical drive created from the source logical drive and the data contained in the overwrite data area implementing by the snapshot function.

UK HQ
Langstone Road
Havant
Hampshire PO9 1SA
United Kingdom

T +44(0)23 9249 6000
F +44(0)23 9249 2284

www.xyratex.com

ISO 14001: 2004 Cert No. EMS91560