The Performance of Non–Redundant Striping in a SSA Disk Array

Cynthia Childers
Vanderbilt University*
Elizabeth Varki
University of New Hampshire

February 8, 1999

Abstract

An increasingly popular method of improving I/O performance is by distributing data across multiple disks in parallel. This organization of data is called striping, and a group of disks organized in this manner is called a disk array. This work investigates the performance of striping data in an array of Serial Storage Architecture (SSA) disks connected in a loop topology. A synthetic I/O workload that allows many parameters to be varied and is representative of many real workloads is used. Performance metrics of this architecture are computed from measurements of the experimental data. This study: 1) presents experimental measurements on the performance of striping data in a loop of SSA disks under different workload scenarios, 2) identifies workload parameters that significantly affect I/O performance in this particular architecture, 3) demonstrates the optimal amount of I/O parallelism given a specific workload, and 4) compares these results to previous disk performance studies.

Index Terms: parallel input/output, RAID disks, performance evaluation, I/O performance, disk striping.

1 Introduction

Magnetic disk performance has long been of concern to computer users and computer designers. The performance differences between the CPU/memory and secondary storage devices requires improvements in the way data is stored and retrieved. Disk arrays provide improved I/O performance by distributing data across multiple disks in parallel. This organization of data that results in parallelism of I/O requests is called **striping**, and an array of disks organized in this format is called a **disk array**, or a Redundant Array of Inexpensive Disks (RAID) [7].

In this work, the performance of non–redundant striping on a loop of Serial Storage Architecture (SSA) disks is evaluated. Synthetic workloads with many parameters that may be varied execute and issue I/O requests to the disks. These workloads represent a wide variety of real workloads. Actual measurements of service and response times are taken. Based on measured performance and the workload parameters varied, specific parameters that most significantly affect performance in this architecture are identified. These results are compared to previous disk performance studies.

*This work done while a co-op student at IBM T.J. Watson Research Lab, Yorktown Heights, NY.

One of the primary goals of this study is to determine the *"amount of parallelism"* required to produce optimal performance for any particular I/O workload executed on this architecture. The "amount of parallelism" refers to the number of disks across which each I/O request is striped. The workload parameter that defines the parallelism of an I/O request is the **basic striping unit (BSU)** size. The BSU size is the amount of contiguous data written to or read from a single disk. A *logical disk request* issued by an application is divided (i.e., striped) into equally sized *physical* requests to the disk array. Each logical request is one **stripe** of data. Each smaller physical request accesses a different disk, thereby reducing the amount of time to transfer the data.

In related work, Chen and Patterson[1] use simulations of a non–redundant, disk array to conclude that the optimal BSU size depends most significantly on the number of outstanding I/O requests or **multiprogramming level (MPL)** and only minimally on the request size. The optimal striping unit is further examined by Chen and Lee[3] in a RAID level 5 via simulation. Parity is modeled and a read/write ratio is included to investigate the effect of workloads composed of reads and writes. They conclude that the optimal striping unit for read requests varies inversely with the number of disks in the array and vice versa for write requests. Merchant and Yu[6] investigate striping strategies in RAID level 1 architectures utilized by on–line transaction processing workloads. Weikum and Zabback[9] have proposed file–specific BSU sizes based on file access characteristics and workload throughput requirements in a non–redundant disk array.

The results of this work both verify and expand upon previous work. Previous work in this area is primarily based on simulations. The results of this work are derived from experimental measurements on an interesting disk technology of a loop of Serial Storage Architecture (SSA) disks (see Figure 1, discussed in more detail in Section 2). The optimal BSU size is determined on this new disk architecture for a wide variety of workloads. The I/O workload incorporates the scenario where processes alternate between computation and I/O, whereas previous work has concentrated on workloads that continually issue I/O requests. It is determined via experimentation that the MPL and I/O request size are significant workload parameters that affect performance on this architecture. More generally, it is shown that disk array utilization affects the choice of optimal BSU size. For a given workload, as the disk array utilization increases, optimal BSU size increases. Other workload parameters such as the amount of sequentiality of requests and the request type (read or write) are shown to have little affect on the choice of optimal BSU.

The remaining part of this paper is organized as follows: Section 2 describes the hardware and software platform, Section 3 describes the experimental results, and conclusions and future work are given in Section 4.

2 Experimental Platform

2.1 Hardware Description

All experiments are executed on an IBM RS/6000 uniprocessor running an AIX operating system. Besides the executing experiments, no other applications are executing on the CPU and no other applications are accessing the disk drives or using the lines connecting them.

The measured experiments are performed on a disk array consisting of four SSA disks connected in a loop topology with the CPU (see Figure 1). SSA is a serial interface for connecting devices and is one of the serial storage options of the SCSI-3 standard. It supports SCSI commands such

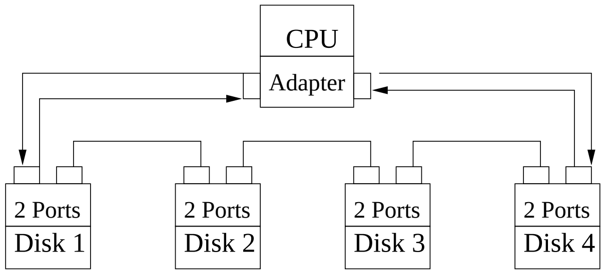

Figure 1: SSA Disk Architecture

Disk capacity	4 GB
Average rotational latency	4.17 ms
Average seek	8.8 ms
Cylinders	4390
Tracks per cylinder	8
Sectors per track	135
Disk transfer rate	7.9 MB per second
Bus communication rate	20 MB per second per route

Table 1: SSA Disk Specifications

as read, write and sense data. It also returns SCSI status codes. Frames are used to transmit data and are multiplexed; more than one data communication may be sent simultaneously over a single line. Each disk and the CPU has an adapter with two ports. Each port is capable of two 20 MB/sec communications, one inbound line and one outbound line, for a total of 40 MB/sec per port. A disk could be simultaneously sending and receiving data at each port, over each in and outbound line, resulting in a maximum throughput of 80 MB/sec. The SSA interface uses store and forward routing and each line is independent of the others resulting in alternate paths to each disk and no single point of failure. If one line fails, all disks can still be accessed via another route. Table 1 summarizes the hardware specifications.

2.2 Workload Description

A synthetic workload is used to read and write data to the disks. It is composed of a varying number $(1, \cdots, 4)$ of processes issuing requests to the disk array. This MPL is one of the workloads' varied parameters. Each process in the experiment has the following variable parameters: 1) I/O request size, 2) number of disks across which the data that the I/O request is accessing is striped, 3) average delay at a multiprocessor server where each process executes on exactly one CPU and there is never a queue (i.e., computation time between I/O requests), 4) the percent of I/O requests from this process that are sequential, and 5) the percent of I/O requests from this process that are read requests. Table 2 summarizes these parameters and lists their value ranges.

Read probability, r	0, ..., 1.0
Multiprogramming level, m	1, ..., 4
Delay time at infinite server, $1/\lambda$	0, ..., 5000 ms
Sequential probability, s	0, ..., 1.0
Request size, q	2 KB, ..., 4 MB
Number of disks striped across, n	1, 2, 4

Table 2: Workload Parameters and their Value Ranges

Each experiment is comprised of 100 measurement cycles. During a measurement cycle, each process reads or writes one stripe of data, waits for its completion, experiences a delay at the delay server (unless the mean delay is set to 0), and then issues the next request The number of processes issuing parallel I/O requests in each experiment is fixed, and hence the system is modeled as a closed queuing model with multiprogramming level fixed at m.

For each experiment, the values of the parameters given in Table 2 are fixed. Each process in an experiment experiences an exponentially distributed delay at the delay server, which models a multiprocessor capable of dedicating a single CPU to each of its processes and represents computation between I/O requests. There is no wait time at the delay server for any of its processes. After the delay, an I/O request of size q is issued to the disk array. The amount of parallelism of this I/O request is represented by n, the number of disks across which it is striped. Thus, the BSU, the amount of contiguous data accessed on a single disk, is q/n. The process accessing the stripe of data *forks* n threads, each responsible for one BSU of q/n bytes on one disk. A thread waits in the queue of its respective disk (queuing discipline is first-come-first-served). When the thread gets to the head of the disk queue, it is serviced by the disk. An I/O request is complete when all of its threads *join*, that is, when all threads have completed accessing their BSU. Note that for each iteration of the experiment, the first disk across which the stripe is accessed is chosen at random. If the request will access more than one disk (i.e., $n > 1$), the second disk is adjacent to the randomly chosen first disk. For example, if the request is striped across two disks and disk 3 is the first disk chosen randomly, then the I/O request will access disks 3 and 4. If disk 4 is the first randomly chosen disk, disk 1 is considered to be the next disk adjacent to it. For each iteration of the experiment, the request type (read or write) is determined by the read probability, r, and the beginning sector of a stripe (the same for each BSU on each disk) is determined based upon the sequential probability, s. If an I/O request is sequential, its beginning sector is adjacent to the ending sector of the previous request. If it is not sequential, it's beginning sector is randomly chosen.

Each BSU is written by one or more consecutive SCSI read or write commands (SCSI_READ_EXTENDED or SCSI_WRITE_EXTENDED). The maximum SCSI request size is 128 KB, so requests larger than this are broken into multiple SCSI commands, resulting in increased overhead. SCSI commands are issued to the raw device avoiding data buffering by the file system. In the SCSI command, a flag is set to avoid reading ahead by the disk.

Figure 2 is a high level view of the algorithm used. The parent process forks MPL children and waits. Each child process is responsible for writing one entire stripe of data. A child process uses

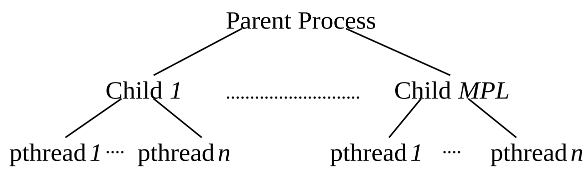

Figure 2: High-Level Workload Algorithm

a pthread to write a single BSU of the data, spawning n pthreads, one per disk in that process' stripe. The following is the main program algorithm that forks each child process which issues I/O requests.

1. From command line input, initialize workload parameters of MPL, number of disks striped across, BSU size, mean delay at delay server, the percent of requests that are sequential, the percent of requests that are read requests.

2. Dynamically allocate and align space for data (entire stripe) to be written to the disk array.

3. Open devices for raw reads and writes to the disks, thus avoiding any buffering by the file system.

4. Mark all disk queues as empty using an array in a shared memory segment.

5. Mark all children not yet finished with their 100 test iterations.

6. For each BSU, fill SSA command structure with parameters that are static for this experiment such as a pointer to portion of buffer to be written from/read to and its SCSI command structure with the same pointer and length of the buffer.

7. Each child that executes uses pthreads to access it's data stripe, one pthread per disk (i.e., a pthread accesses one BSU). A pthread issues SCSI commands to each disk until the BSU has been accessed. Set values such that pthread stack information remains intact and a join of all pthreads of a particular child will complete successfully.

8. Fork MPL processes and wait for all of them to complete. After all children have completed execution, the parent process closes all devices and exits.

While the parent process waits, each child process executes the following algorithm.

1. Initialize those portions of the SSA command structure that may change with each iteration of this experiment and/or are different for each child process such as the first disk striped across and beginning sector location on each disk (potentially different for each child process) and type of request (read or write).

2. Begin the 100 test iterations.

3. If there's a mean delay time, wait a randomly generated, exponentially distributed amount of time.

4. Child process puts itself in shared memory disk queue at first available position. A semaphore is used to control access to this shared memory segment.

5. Read the beginning time.

6. For all disks needed in this I/O, wait until each disk is acquired. When a single disk in the stripe is acquired, that BSU access begins by creating a pthread that sends consecutive SCSI commands to the disk until this BSU access is complete. If it is the first disk acquired, the time is read in order to later compute wait time. When the pthread has completed the BSU access, it takes itself off the disk queue and exits.

7. The child process, parent to its pthreads, waits until all pthreads have completed then reads the time.

8. For each test iteration, wait time and response time are computed and stored in an array in memory.

9. Non-static test parameters such as request type (read/write) and whether the next stripe is sequential or not is recomputed and the appropriate parameters in the SSA and SCSI structures are set and the next test iteration begins.

10. After every child has completed 100 test iterations, the child continues issuing I/O requests until every child has finished. Then each child writes its response times and wait times to a separate file identified by its process id and exits.

3 Experimental Results

The performance metric used in this work is average response time. Throughput can then be computed using Little's Law[1]. Because the experiments model a closed queueing system, throughput is inversely proportional to response time. As average response time increases, average throughput decreases.

Experimental results are presented in four sets depending on the three workload parameters r, read probability, s, sequential probability, and $1/\lambda$, mean delay. These four experimental sets are: 1) random read requests, no delay ($r = 1, s = 0,\ 1/\lambda = 0$), 2) random write requests, no delay ($r = 0, s = 0,\ 1/\lambda = 0$), 3) sequential write requests, no delay ($r = 0, s = 1,\ 1/\lambda = 0$), 4) and random read requests with varying delays ($r = 1, s = 0$). For each of these sets of experiments the remaining workload parameters, of multiprogramming level, request size, and number of disks striped across are varied. The rest of this section focuses on the results for each of the experimental sets.

3.1 Random, Read and Write Requests with no Delay

In this experiment, stripes are read from or written to the disk array with the beginning sector of the BSU's randomly chosen. The flag set in the SCSI command that prohibits the disk from

[1] Little's Law implies that Throughput = $\frac{Number\ of\ processes}{Response\ time} = \frac{Request\ size * MPL}{Response\ time}$.

reading ahead does not affect random reads. It is left to future work to incorporate the effects of reading ahead for sequential read requests, thereby determining the exact performance impact of this buffering. Each BSU at a disk has the same beginning sector position. There is no delay at the delay server. An example of real workloads represented by these synthetic workloads include on–line transaction processing where I/O requests are randomly distributed throughout the file system with little or no computation incurred between requests.

Figure 3 plots the average of measured read request response times for varying request sizes striped across $n = 1, 2, 4$ disks. Each separate graph plots response times for a differing number of requests in the system. When the MPL is 2, there are two processes issuing I/O requests to the disk array, each with its own beginning BSU sector per experiment iteration. Since the BSU size is q/n, for each point in the graph the BSU size changes. For example, in the first graph with MPL = 1, the BSU size for a 4 MB stripe distributed across 4 disks is 1 MB which, in this case, results in lowest response time compared to a BSU size of 2 MB or 4 MB when this stripe is distributed across 2 and 1 disk(s), respectively. While it is advantageous to stripe a 4 MB request, the performance of the smallest I/O request investigated, 2KB, suffers from striping, from a decrease in BSU size. Note, however, that in the 2 KB case, from no striping response time increases at 2 disks (a BSU size of 1 KB) then decreases at 4 disks (a BSU size of 512 bytes). The increase at 2 disks results from an increase in seek[2] and rotational delay[3]. The disk arms and spindles are not synchronized and at each iteration of the experiment a new set of 2 disks is chosen so that the arms of the 2 selected disks may never be synchronized. When the arms are not synchronized, overall seek time, the time for both arms to seek to the desired sector, increases. It is equal to the maximum of both seek times. At MPL = 1 when the 2 KB request is striped across all four disks in the array, the arms are always synchronized and the increase in seek time is not experienced. If the disk array were composed of more than four disks, there would be no decrease in response time at four disks. However, this anomaly would become evident for some request sizes as the stripe accessed an increasing number of disks in the array. For request sizes larger than 2 KB, the same reduction in the maximum seek time is experienced, but is not as clearly evident because a reduction in transfer time[4] outweighs it. At higher MPL's, response time is also significantly affected by the request size. At MPL = 4, a 4 MB request size gains a decrease of 17% from no striping to striping across all four disks whereas at MPL = 1, the decrease is 65%. Each request size experiences a different speed up (or slow down) in response time as the request is striped across an increasing number of disks. The graph clearly indicates that request size significantly affects performance.

Performance of striped I/O requests is also significantly affected by the MPL. In Figure 4, a different view of the data in Figure 3, note the performance of the 8 KB request in the first graph as the MPL increases. When MPL = 1, as the request is striped across more disks, response time changes little. An 8 KB request experiences a slight decrease in response time when it is distributed across two disks and its BSU size is 4 KB. As the MPL increases, response time from no striping to striping across four disks increases until at MPL = 4, the increase in response time is almost 300%. The performance degradation of the 8 KB request increases as the MPL increases. For the 4 MB request, at MPL = 1, response time decreases dramatically from no striping to striping across four disks, For all request sizes, as the MPL increases the benefit from striping decreases. It is better

[2]Seek time is the time for the read/write head to be positioned at the appropriate cylinder.
[3]Rotational latency is the time for the requested sector to rotate under the read/write head.
[4]Transfer time is the time taken to transfer data from the disk to the I/O bus and vise-versa.

to reduce the amount of data striping via an increase in the BSU size so that more I/O requests may be serviced concurrently.

Figure 5 graphs the optimal BSU size for each request size at each MPL. The optimal BSU size is dependent upon the MPL and the request size. For 2 KB, the optimal BSU size is 2 KB regardless of MPL. The same is true for 8 KB, ..., 128 KB. The optimal BSU size for these request sizes is the request size itself. For this hardware platform, it is not beneficial to stripe data less than or equal to 128 KB in size. At MPL = 1, request sizes of 512 KB and 1 MB achieve their best response times with BSU sizes of 256 KB, across 2 disks and 4 disks, respectively. As the MPL increases, the optimal BSU size increases to 512 KB where the 512 KB request is not striped and the 1 MB request is striped across 2 disks. This optimal BSU size remains unchanged through MPL = 4. A request size of 2 MB has an optimal BSU size of 512 KB until the MPL = 4 where the optimal BSU size changes to 1 MB. Thus, it is better to stripe a 2 MB request across 4 disks until MPL = 4 where the best response time is reached when it is striped across only two disks. The optimal BSU size for a 2 MB request is 512 KB for MPL = 1, 2 and 1 MB for MPL = 3, 4. For a request size of 4 MB, the optimal BSU is 1 MB, regardless of MPL. Note that several response times were almost identical as in the case of a 4 MB request with MPL = 2, striped across 2 disks or 4. In these cases, the optimal BSU size was chosen as the larger of the two. The addition of more disks accessed doesn't result in any performance gain but does utilize disks that could be used by another I/O request. These trends verify and expand upon Chen and Patterson's[1] work. Chen and Patterson show, via simulations, that the selection of BSU for a workload is based on the multiprogramming level of the system. The previous three graphs in Figures 3, 4, and 5 show that both MPL and request sizes are important factors in the choice of BSU sizes.

As in other hardware platforms, performance improvements resulting from striping occur when request sizes increase to the point where a major component of disk service time is the transfer time. In general, smaller I/O requests do not benefit from striping. The media transfer time is negligible compared to the seek and rotate times. Therefore, the overhead involved in increased seek and rotate times when striping across more disks is greater than the reduction in disk transfer time. For large request sizes, where disk transfer time is the major component of disk service time, the reduction in transfer time due to striping across more disks is significant and outweighs the increase in seek/rotate times.

Response times for write requests whose beginning sector is randomly generated for MPL = 1, 2 are shown in Figure 6. Measurements are the result of writing to the disks themselves since delayed write buffering has been avoided by opening the raw device. Since the array has no redundant data as in a RAID level 5, write requests do not generate additional data to be written. Redundancy and write buffering are left for future work. The response time curves of this experimental set follow the response time curves of the random, read requests. Write requests result in slightly higher response times because of the additional time needed to more accurately position the read/write head in order to write data to the device. Although performance is affected by the request type, the optimal BSU size is not. Whether a request is a read or write request in a non–redundant disk array, the optimal BSU size is the same.

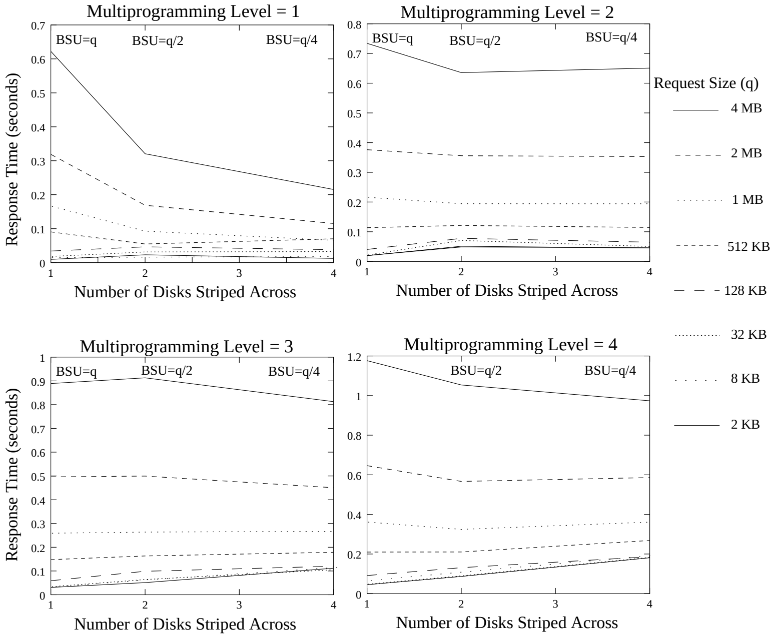

Figure 3: Response Times for Workload - Random Reads for varying Request Sizes

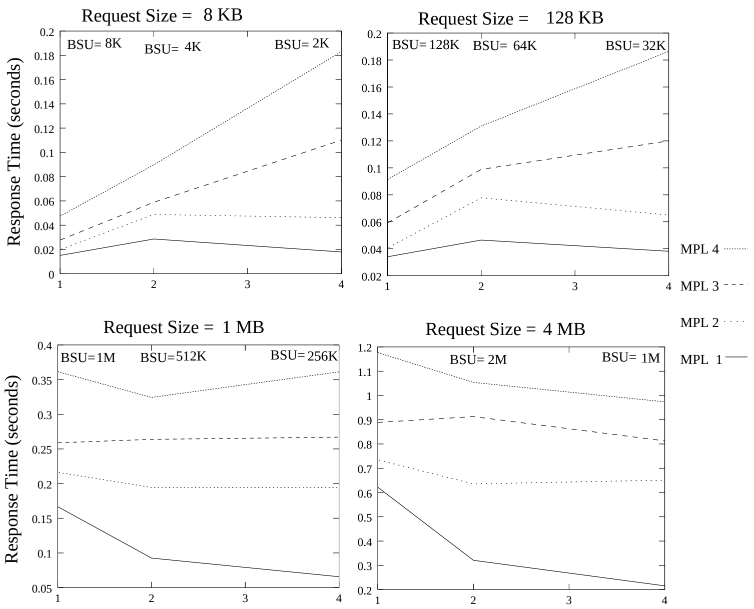

Figure 4: Response Times for Workload - Random Reads for varying MPLs

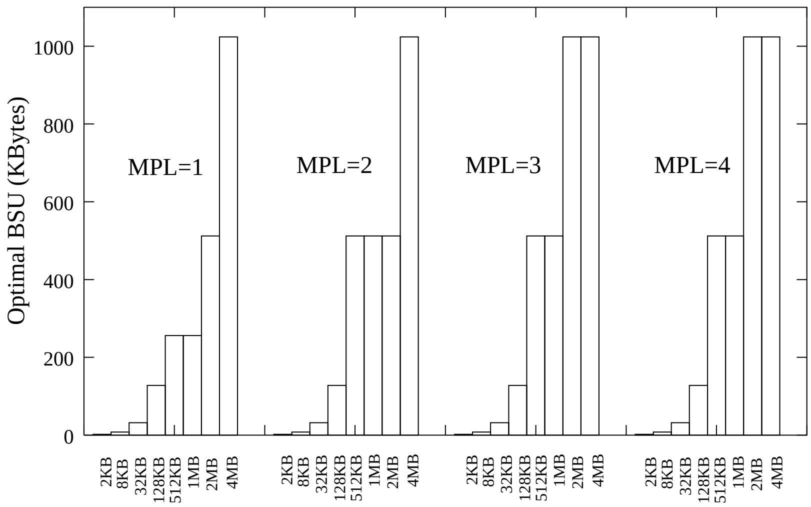

Figure 5: Optimal BSU sizes - Random Reads for varying Request Sizes

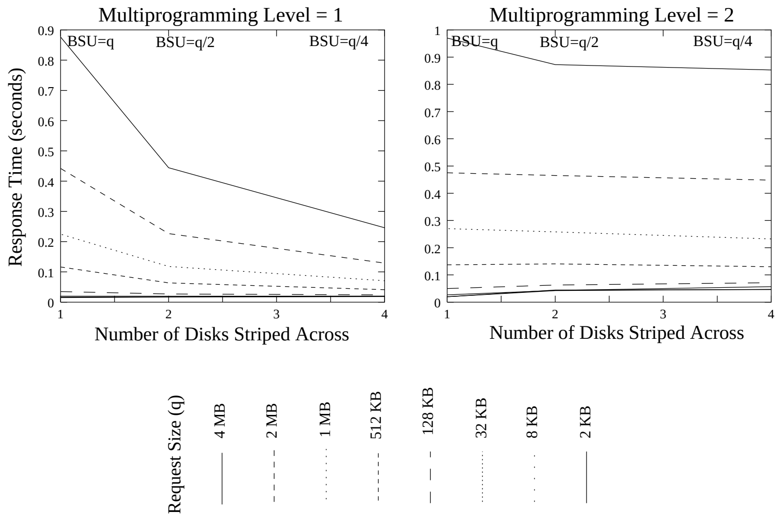

Figure 6: Response Times for Workload - Random Writes for varying Request Sizes

3.2 Sequential Accesses with No Delay

This experiment involves processes repeatedly issuing 100 percent sequential requests to the disk array (as opposed to random in the previous cases). Similar real workloads include multimedia video and audio and data back up and restore. For each process, the beginning sector of subsequent I/O requests begins at the sector adjacent to the last sector of the previous request. The first sector of each request is generated randomly. Read ahead buffering has been disabled, therefore read request results are similar to write request results excepting the additional overhead time needed to write data to the device. Future work will include incorporating the effects of buffering data that has been read ahead.

Figure 7 graphs response time for this experimental suite. When the MPL = 1, I/O requests experience almost no seek time since the beginning sector is adjacent to the end of the previously read BSU. More generally, this is true for any MPL in the particular case where a disk has been idle since the last I/O serviced from the same current process. At MPL = 1, sequential write requests have a slightly lower response time, more evident for smaller requests where seek time makes up a large portion of service time, than random write requests. For large I/O requests at any MPL, the decrease in seek time is an insignificant portion of service time and the gain of sequential requests is negligible. At higher MPL's, the decrease in seek time is most often lost because a disk arm has moved as a result of a separate I/O request. Because the gain from sequential accesses is either lost (at MPL's higher than 1) or insignificant (for larger request sizes), the response time curves of sequential write requests follows the response time curves of random read and write requests. Thus, the amount of parallelism (i.e., optimal BSU size) is insensitive to the percentage of sequentiality in the workload.

3.3 Accesses with Delay

In this experimental suite, every process issues I/O requests, waits for its completion, experiences a delay at the delay server, and then issues the next request. This workload might represent some scientific computing applications such as matrix manipulations. The delay is exponentially distributed with a mean given as an input parameter to the experiment. The delay server models a multiprocessor server with each process allocated exactly one CPU. Thus, there is never a queue of processes waiting for a CPU.

Figure 8 graphs the response times of 2 MB random read requests for different mean delay times. The graph plotting no delay is shown for comparison purposes. These differing delay times result in a change in the utilization of the disk array. As the average delay increases, a process spends a higher percentage of time at the delay server and a smaller percentage of time at the disk array, leaving it idle more frequently. Disk array utilization decreases as the mean delay increases. The addition of delay between requests has the same affect as decreasing the MPL at the disk array. For example, response times curves at MPL = 2 when the mean delay is 345 ms is similar to the response time curve when mean delay is 0 and the MPL is set to 1. There is enough delay between requests that each separate process has almost sole access to the disk array. When the MPL = 3 with a mean delay of 345 ms, response times follow the same trend as with no delay and MPL = 1, but are slightly higher indicating that the disk array is slightly more utilized than the base case when MPL = 1. This higher utilization results in some queueing time which leads to a slight increase in response time. When the MPL = 4 and the mean delay is 1000 ms, the response

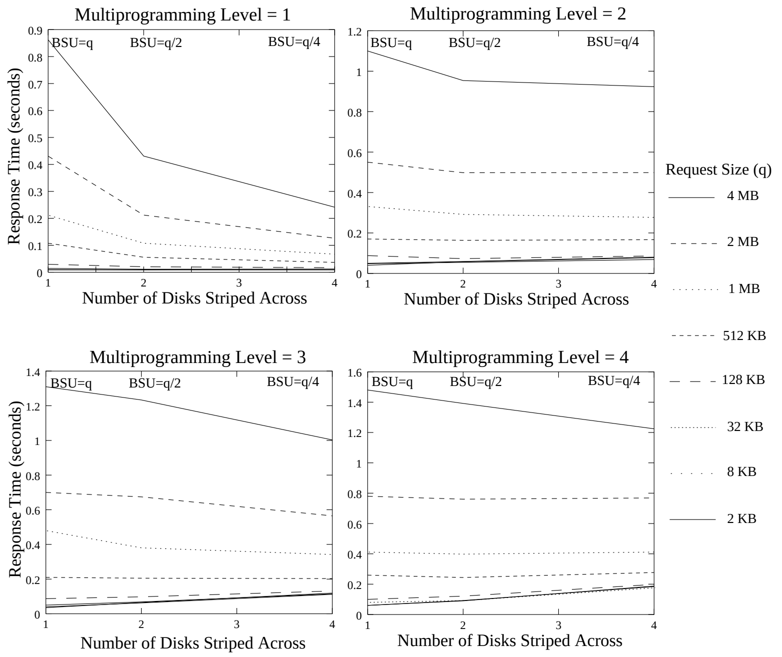

Figure 7: Response Times for Workload - Sequential Writes for varying Request Sizes

time curve is similar to no delay and an MPL = 1. For MPL = 4 with a mean delay of 345 ms, the response time curve follows the same trend as with MPL = 1 and no delay but is from 18% to 125% higher from the addition of queueing time waiting for one or more disks that are not free. Compared to MPL = 2 with no delay, random reads at MPL = 4 and a mean delay of 345 ms have response times differing by 2% to 26% but the response time curve is not as flat. Thus, there is not enough queueing time to negate the benefits of striping, but enough to increase response time substantially. In general, as mean delay decreases, the disk array utilization increases and the benefits of striping data decreases. An increase in the mean delay has the same affect as lowering the MPL. Just as MPL significantly effects the choice of optimal BSU size, more generally, so does the utilization of the disk array.

3.4 Comparison with Previous Disk Performance Studies

The performance predictions of analytic models introduced by Chen and Patterson[1] and Lee and Katz[5] are compared with the experimental results pertaining to SSA disks. Chen and Patterson[1] use simulations of a non-redundant 16-disk array system to derive an equation for the BSU as: $S * (mean\ disk\ positioning\ time^{5}) * (disk\ transfer\ rate) * (MPL - 1) + sector\ size$, where S refers to the MPL-slope coefficient and is found to be approximately 1/4 for a 16-disk array system. In their formula, the request size is not a factor in the computation of the BSU. These experimental results clearly show that both request size and MPL are significant workload parameters in the computation of BSU size. The value of S given in [1] is particular to a 16-disk array system and

[5] Positioning time refers to the sum of seek time and rotational latency.

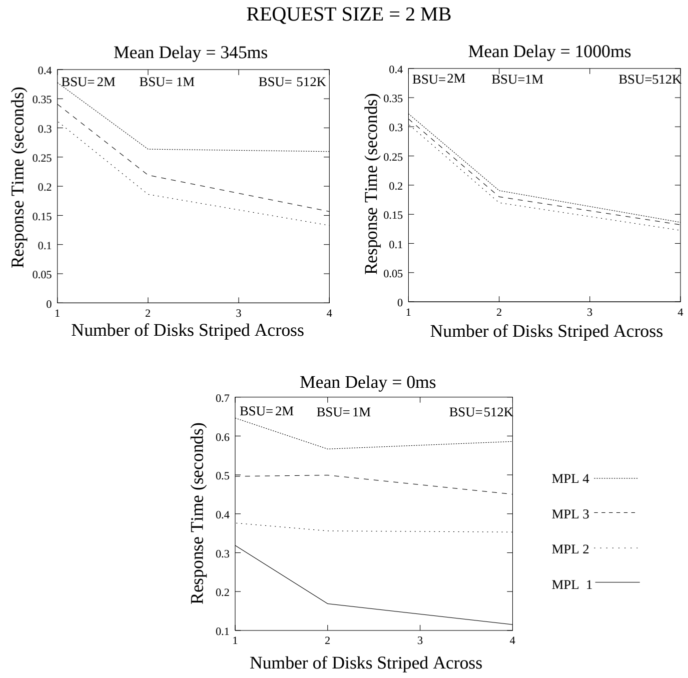

Figure 8: Response Times for Workload - Random Reads with Delay for varying MPLs

cannot be used for the 4-disk array SSA system used in the experimental platform. Hence, it is not applicable.

Lee and Katz[5] develop an analytic model of non-redundant disk arrays and use simulations to validate their model. They derive the following equation for BSU:

$$\sqrt{\frac{\text{positioning time * transfer rate * (MPL -1) * request size}}{\text{number of disks}}}$$

They conclude that both the request size and the MPL are important workload parameters in determining the BSU and this is consistent with the experimental results shown here. Table 3 compares the BSU sizes determined by the Lee and Katz model with the experimental results. The table shows that the BSU sizes computed by the Lee and Katz model are, in general, much smaller than that shown by the experimental results. This indicates that a better model is required and the working paper[8] deals with this aspect.

4 Conclusions

Previous work has used simulations of classical disk arrays and analytical models to identify workload parameters that determine the optimal amount of parallelism for I/O requests. This work verifies previous results and expands upon it. Experimental data on a SSA loop of disks is used to identify workload parameters that significantly affect the choice of BSU size. It is verified that for this architecture the request size and multiprogramming level affect the optimal amount of parallelism in the disk array. More generally, it is shown that disk array utilization has a significant

Request Size (KB)	BSU (KB) when MPL = 2		BSU (KB) when MPL = 4	
	Experimental	Lee/Katz	Experimental	Lee/Katz
8	8	14	8	25
128	128	58	128	100
1024	512	164	512	284
2048	512	232	2048	401
4096	1024	328	1024	568

Table 3: Comparison with the Lee and Katz Model

impact on the choice of BSU size. It is also found that the workload parameters of I/O sequentiality and request type do not affect the optimal amount of parallelism when read ahead and write buffering have been disabled.

Future work includes enabling read ahead and write buffering to determining the performance significance of these parameters by comparing the data to measurements without this buffering. Redundancy as in a RAID Level 5 disk array will also be incorporated into the disk array and the read and write workloads. Finally, this work is leading to a file system that attempts to dynamically determine a "good" BSU size for a file based upon such parameters as average request size and average measured disk array utilization.

References

[1] Chen, P. M., Patterson, D. A. "Maximizing performance in a striped disk array", Proceedings of the 1990 International Symposium on Computer Architecture. IEEE, New York, 1990, pp. 322 – 331.

[2] Chen, S., Towsley, D. "The design and evaluation of RAID 5 and parity striping disk array architectures", Journal of Parallel and Distributed Computing, 17, 1993, pp. 58 – 74.

[3] Chen, P. M., Lee, E. K. "Striping in a RAID level 5 disk array", Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, May, 1995.

[4] Kim, M. Y., Tantawi, A. N. "Asynchronous disk interleaving: approximating access delays", *IEEE Transactions on Computers*, vol. 40, no.7, July 1991 , pp. 801 – 810.

[5] Lee, E. K., Katz, R. H. "An analytic performance model of disk arrays" Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, May, 1993.

[6] Merchant, A., Yu, P. S. "Analytic modeling and comparisons of striping strategies for replicated disk arrays", IEEE Transaction on Computers, vol. 44, no. 3, March 1995, pp. 419 – 433.

[7] Patterson, D., Gibson, G., Katz, R.H., "A case for redundant arrays of inexpensive disks (RAID)", Proceedings ACM SIGMOD Conference Management of Data, June 1988.

[8] Varki, E., Childers, C., "Analytic modeling of non-redundant disk arrays", Work in progress.

[9] G. Weikum, P. Zabback, "Tuning of striping units in disk–array–based file systems", Proceedings of the 2nd International Workshop on Research Issues on Data Engineering: Transaction and Query Processing, IEEE, Washington, DC, 1992, pp. 80–87.