CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2006; **18**:1929–1950
Published online 4 April 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.1038

Experimental analysis of a mass storage system

Shahid Bokhari[1,2,*,†], Benjamin Rutt[2], Pete Wyckoff[3]
and Paul Buerger[3]

[1]*Department of Electrical Engineering, University of Engineering and Technology, Lahore, Pakistan*
[2]*Department of Biomedical Informatics, Ohio State University, Columbus, OH 43210, U.S.A.*
[3]*Ohio Supercomputer Center, Columbus, OH 43212, U.S.A.*

SUMMARY

Mass storage systems (MSSs) play a key role in data-intensive parallel computing. Most contemporary MSSs are implemented as redundant arrays of independent/inexpensive disks (RAID) in which commodity disks are tied together with proprietary controller hardware. The performance of such systems can be difficult to predict because most internal details of the controller behavior are not public. We present a systematic method for empirically evaluating MSS performance by obtaining measurements on a series of RAID configurations of increasing size and complexity. We apply this methodology to a large MSS at Ohio Supercomputer Center that has 16 input/output processors, each connected to four $8 + 1$ RAID5 units and provides 128 TB of storage (of which 116.8 TB are usable when formatted). Our methodology permits storage-system designers to evaluate empirically the performance of their systems with considerable confidence. Although we have carried out our experiments in the context of a specific system, our methodology is applicable to all large MSSs. The measurements obtained using our methods permit application programmers to be aware of the limits to the performance of their codes. Copyright © 2006 John Wiley & Sons, Ltd.

Received 9 March 2005; Revised 29 August 2005; Accepted 29 August 2005

KEY WORDS: RAID controller; disk; IBM DS4300; IBM FAStT600; mass storage systems; parallel computing; performance evaluation; RAID

*Correspondence to: Shahid Bokhari, Department of Electrical Engineering, University of Engineering and Technology, Lahore 54890, Pakistan.
†E-mail: shb@acm.org

Contract/grant sponsor: National Science Foundation; contract/grant numbers: ACI-9619020 (UC Subcontract 10152408); EIA-0121177; ACI-0203846; ACI-0130437; ANI-0330612; ACI-9982087
Contract/grant sponsor: Lawrence Livermore National Laboratory; contract/grant number: B517095 (UC Subcontract 10184497)
Contract/grant sponsor: NIH NIBIB BISTI; contract/grant number: P20EB000591
Contract/grant sponsor: Ohio Board of Regents BRTTC; contract/grant number: BRTT02-0003

1. INTRODUCTION

Current and proposed high-performance computing applications in the fields of physics, astronomy and, increasingly, bioinformatics need sustained access to datasets in the petabyte range. Such massive storage requirements can only be satisfied with large disk farms organized as redundant arrays of independent/inexpensive disks (RAID). Most high-performance, high-capacity RAID arrays employ proprietary controllers and their performance can be difficult to predict as little information about the internal details of these controllers is made available by the manufacturers. Users of such storage systems cannot be sure whether the performance obtained by them is indeed the maximum that their hardware can deliver and, if not, where the bottleneck lies.

This paper presents a methodology for experimentally evaluating mass storage systems (MSSs). Our research is motivated by the need to understand the performance of a MSS at the Ohio Supercomputer Center (OSC). This system is based on a pool of IBM FAStT600/DS4300[‡] storage controllers that, in conjunction with 16 dedicated Intel P4 input/output ('xio') processors and 576 Maxtor disks, provide 116.8 TB of scratch storage. However, the methodology we describe is applicable to all RAID arrays.

Our methodology involves identifying the potential bottlenecks in the system and then carrying out a series of measurements on a range of RAID systems, starting with a single disk and working up to the largest array possible with the available hardware. The read/write performance of the system should evolve smoothly as the system size increases, with any discontinuities in performance clearly attributable to specific changes in resources. Saturation of performance should be attributable to one of the potential bottlenecks.

In the OSC MSS, for example, we carried out measurements on six RAID configurations, ranging from a one-disk, single-controller system all the way to a 36-disk, two-controller system (the latter corresponds to the configuration actually in use at the OSC). The measured performance increased predictably with system size and resources, giving us confidence in our data for the full configuration.

The methodology developed here will be valuable to designers of MSSs who, like us, will rarely have access to proprietary internal details of disk controller architecture and will have to resort to empirical measurements to ensure that their systems are performing at or near optimal levels. The experimental observations obtained using our methodology should be useful to application programmers developing high-performance computing applications with massive out-of-core storage requirements.

This paper is organized as follows. Section 2 describes the problems of predicting RAID performance and outlines our proposed three-stage methodology. Sections 3, 4 and 5 describe how we applied the three stages of our methodology to the OSC FAStT600 pool. We present our conclusions in Section 6.

2. THE METHODOLOGY

2.1. Performance of RAID arrays

RAID technology has developed very rapidly since the early 1990s as a means for utilizing inexpensive hard disks to provide high-volume, high-data-rate, reliable secondary storage. The seminal reference

[‡]The FAStT600 was rebranded as the DS4300 shortly after we carried out this research.

for this technology is [1]. Modeling of disk drive performance is discussed in [2] while recent developments in this area are surveyed in [3]. The effective read/write rates of RAID units of various types and sizes are clearly discussed in [4].

Most contemporary high-performance, high-capacity RAID arrays incorporate proprietary controller hardware. The capabilities of these controllers can often be the crucial factor that determines the overall performance of the system. Unfortunately, the internal details of these controllers are seldom clearly indicated in the associated documentation. As a result, administrators and users of large RAID arrays are often unsure if they are obtaining the maximum performance possible with the resources available to them.

2.2. Proposed methodology

We propose a three-stage process for experimentally evaluating a MSS.

(1) Examine the architecture of the system to identify the components whose performance can be studied separately.
(2) Identify the *potential* bottlenecks in the system. If possible, simple experiments on individual components can be conducted to identify the bottlenecks. For example, a single disk's raw I/O rates are easy to measure and yield important theoretical barriers to performance[§]. This investigation can identify the one or two components very likely to be the *true* bottleneck.
(3) Carry out a series of measurements on RAID arrays of various sizes, configured around the hardware used in the target system.
    (a) These measurements should ideally start with a single disk and work up to the full system.
    (b) For each of the test systems, measure the read and write rates for files of different sizes, utilizing different buffer sizes and different numbers of threads.
    (c) The measured performance should increase smoothly with increasing system size, with any discontinuities clearly attributable to changes in hardware (as RAID configurations change). Saturation of performance should be attributable to one of the bottlenecks identified in stage (2), above.

In the following sections we describe how we executed the above methodology during our investigation of the OSC MSS.

3. STAGE 1: STUDY OF ARCHITECTURE

An application programmer's view of the OSC Intel Pentium 4 Xeon cluster is given in Figure 1. There are 256 compute nodes connected to 16 xio nodes and 112 of these processors are connected to the 16 xio nodes through a switched Infiniband network with a data rate of 8 Gbit s^{-1}. All 256 compute nodes are also connected to the 16 xio nodes through a gigabit Ethernet switch. Each xio node sees a four-way RAID0 disk unit. In fact, each of the imagined disks in this RAID0 is mapped onto an $8 + 1$ RAID5 unit, as discussed below.

[§]The theoretical maximum performance of the system cannot exceed the product of a single disk's raw performance multiplied by the total number of disks.

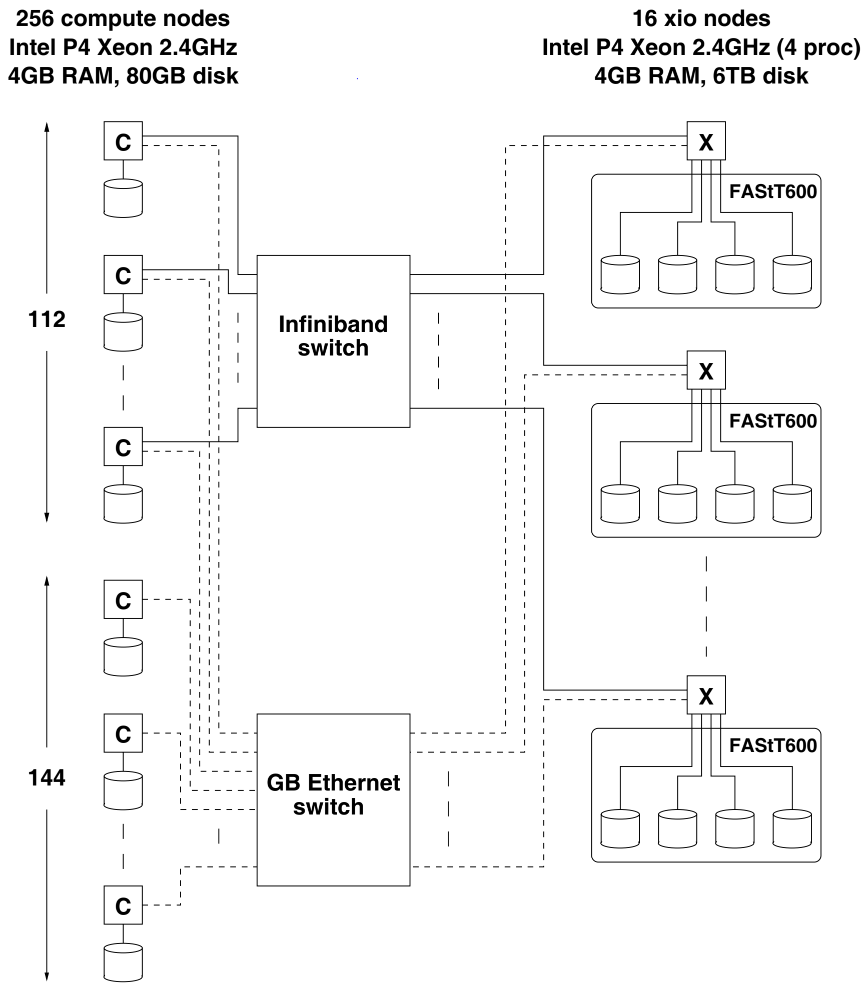

Figure 1. A programmer's view of the FAStT600 system.

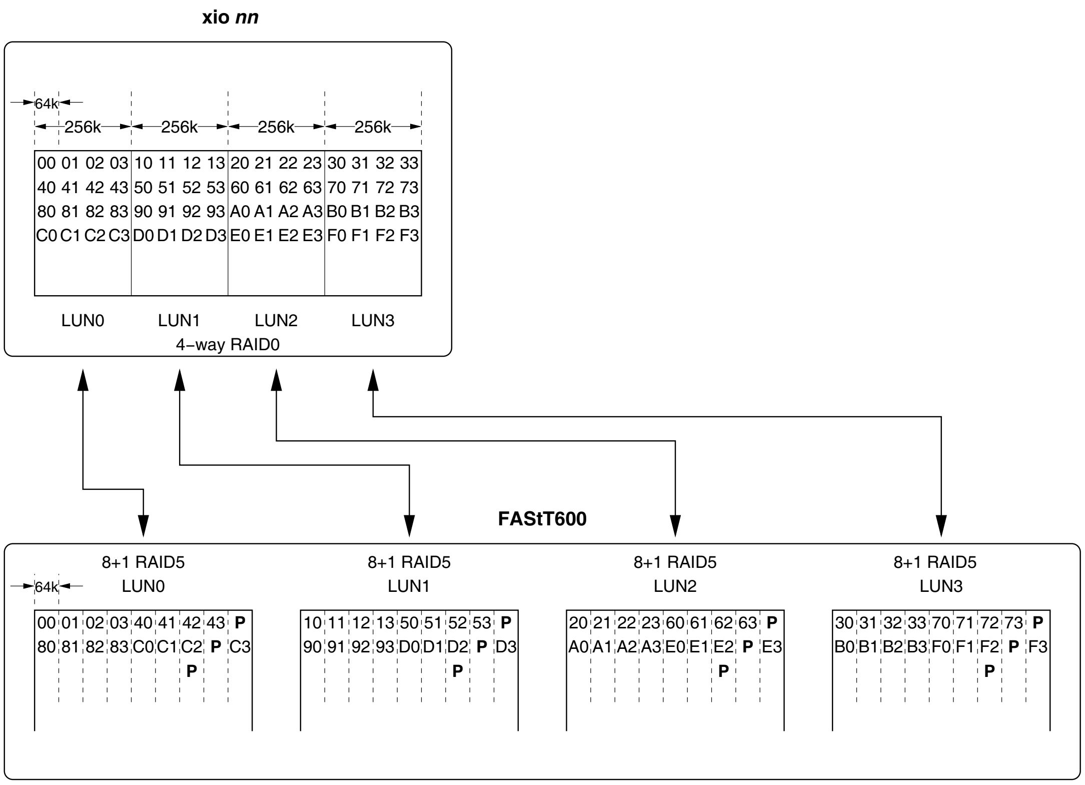
xio nn
64k
256k
256k
256k
256k
00 01 02 03 10 11 12 13 20 21 22 23 30 31 32 33
40 41 42 43 50 51 52 53 60 61 62 63 70 71 72 73
80 81 82 83 90 91 92 93 A0 A1 A2 A3 B0 B1 B2 B3
C0 C1 C2 C3 D0 D1 D2 D3 E0 E1 E2 E3 F0 F1 F2 F3
LUN0
LUN1
LUN2
LUN3
4–way RAID0
FAStT600
8+1 RAID5
LUN0
00 01 02 03 40 41 42 43 P
80 81 82 83 C0 C1 C2 P C3
P
8+1 RAID5
LUN1
10 11 12 13 50 51 52 53 P
90 91 92 93 D0 D1 D2 P D3
P
8+1 RAID5
LUN2
20 21 22 23 60 61 62 63 P
A0 A1 A2 A3 E0 E1 E2 P E3
P
8+1 RAID5
LUN3
30 31 32 33 70 71 72 73 P
B0 B1 B2 B3 F0 F1 F2 P F3
P

Figure 2. Each xio node sees a four-way RAID0 with a block size of 256 kB. Each of these RAID0s is mapped on an 8 + 1 RAID5 with block size 64 kB. Parity blocks are labeled with 'P'.

The OSC system under examination uses two types of RAID[¶]. As shown in Figure 2, each xio node sees a software implemented RAID0 with four logical units (LUNs). These RAID0s have a block size of 256 kB. Each of these LUNs is mapped onto an 8 + 1 RAID5 implemented in a FAStT600 unit. The four-way RAID0 permits the xio processor to stripe data across multiple units and thus improve its transfer rate [4]. No error correction is provided at this level. The xio itself is a two-processor Pentium 4 with 'hyperthreading' and effectively behaves like four processors. Thus, it is capable of handling multiple threads of access with significant speedup. The RAID5s in the FAStT600 are made up of eight data disks and one parity disk each, as shown in Figure 2. The block size in this case is 64 kB.

[¶]While there is a large number of RAID types, those of immediate interest to us are RAID0 and RAID5 [1, p. 153]. RAID0 uses multiple disk units simply to improve transfer rates. RAID5, on the other hand, uses block interleaved distributed-parity for the purposes of error detection and correction. RAID5 also provides high-data-transfer rates for large reads and writes [1, p. 155].

Table I. Basic parameters of Maxtor 7Y250M0 [9].

Formatted capacity	250 GB
Rotation speed	7200 rpm
Data zones per surface	16
Bytes per sector	512
Transfer rate to/from interface	150 MB s^{-1}

These RAID5s provide error detection and correction [1,5]. The read (write) rate from these RAID5s should, in principle, be nine (eight) times the rate of each individual disk [4]. In fact, the overheads in the controller hardware of the FAStT600 result in cumulative rates far below these theoretical figures.

3.1. Components of the system

3.1.1. FAStT600

The IBM FAStT600 (turbo) storage server is the key element of the OSC storage pool. This server is capable of supporting up to 112 hard drives with a total of 16.4 TB capacity. Each FAStT600 has dual controllers and can be configured to present different RAID configurations to the outside world. For example, Figure 5 later shows a FAStT600 configured in six different ways for the purpose of performance experiments (described later in this paper). In normal usage at the OSC, the configuration shown at the bottom of this figure is used. In this case 36 disks are organized as four RAID5 units of nine disks each, with each controller assigned to two LUNs. A different installation might use, for example, two RAID5 units of 14 disks each. The architecture and performance of the FAStT600 controller are discussed in [6,7].

3.1.2. Cisco Director 9509

Two Cisco Director 9509 fibre channel switches [8] are used in the OSC system to connect the FAStT600 units to their respective xio processors. These connections are static and under the control of system administrators.

3.1.3. Maxtor disks

The disks used in the FAStT600 storage pool are Maxtor 7Y250M0 SATA (serial advanced technology attachment) with 250 GB capacity. Table I lists the basic parameters of this disk that are relevant to our discussion. These are taken from [9]. It is noteworthy that ‘zoned recording’ is used in this disk [3]. That is, the number of sectors per track varies with the diameter. This is to compensate for the increase in circumference in the tracks as the diameter increases. There are 16 zones on each disk surface and the number of 512-B sectors varies from 610 in the innermost track to 1102 on the outermost (Table II).

Table II. Parameters that vary with diameter [9].

Parameter	Inside diameter	Outside diameter
Data sectors per track	610	1102
Sustained transfer rate	37 MB s^{-1}	67 MB s^{-1}

We were able to verify the variation in read rates for files written at different locations by connecting a single disk to a Linux PC and then creating 16 large files on it. The file sizes were chosen to be 13 GB each so that these 16 files nearly filled the disk (the remaining capacity was taken up by the filesystem, scratch space, etc.). We argued that files written earlier would be located in the outermost tracks and *vice versa*. Timings proved us correct, as shown in Figure 3. As the disk fills up, the read rate drops to precisely the 37 MB s^{-1} figure given in Table II[‖]. The trend of the file access time is towards 60 MB s^{-1} at the outermost tracks. This is less than the 67 MB s^{-1} claimed by Maxtor. Independent test results on this disk are available in [10,11]. These tests were presumably carried out using low-level operations so that the entire surface of the disk could be evaluated. The plots given in [11] are in general agreement with Figure 3. In particular, the maximum and minimum data rates in these plots are 60.5 and 34.5 MB s^{-1} compared with our 60 (extrapolated) and 37 MB s^{-1}.

3.2. xio nodes

There are 16 xio processors in the FAStT600 storage pool, each connected to its private four-way $8 + 1$ RAID5 array. These processors are dual processor Intel P4 Xeons with 2.4 GHz clocks and 4 GB RAM, running Linux 2.6.6. ‘Hyperthreading’ is used in these machines, so the dual processors appear to be four processors. This is important for the programmer to appreciate, as it means that code written with four to eight threads is likely to perform well on these machines.

4. STAGE 2: IDENTIFICATION OF POTENTIAL BOTTLENECKS

There are a large number of *potential* bottlenecks between the xio processor and the magnetic medium in the disks. These are shown in Figure 4. The following sections present arguments about the importance of each potential bottleneck. The motivation for this discussion is the fact that, although there are 36 disks (each with a measured minimum sustained data-read rate of 34.5 MB s^{-1}) connected to each xio processor, the sustained data-read rate seen at the xio processor is no more than approximately 300 MB s^{-1}, far below the theoretical upper limit of $(36 \times 34.5 =)$ 1.242 GB s^{-1}. Some element in the chain between the magnetic medium and the xio processor is acting as a bottleneck. Let us discuss each possibility.

[‖]The sustained data rates in Table II can be obtained from (sectors per track) × (Bytes per sector) × (rotation speed).

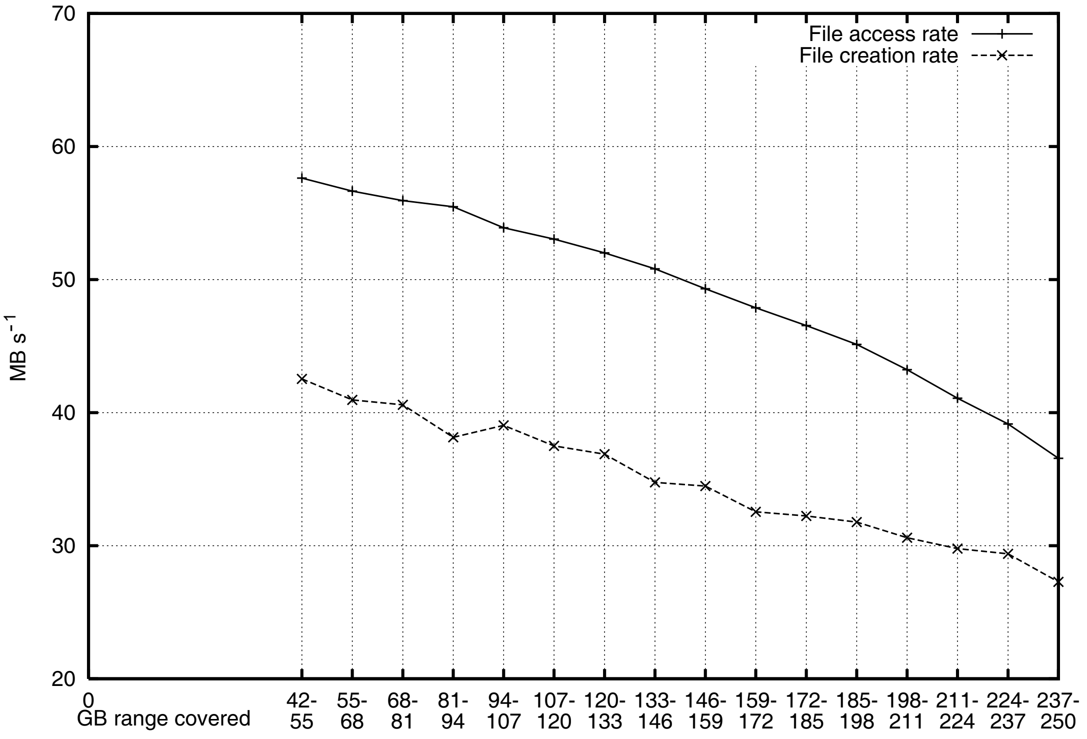

Figure 3. Sustained read and write performance of the Maxtor 7Y250M0 disk across tracks, where 0 GB is the outermost and 250 GB is the innermost track. The first 42 GB of the disk were taken up by the filesystem, scratch space, etc., and could not be evaluated. Accessing a full disk (ext3/1 MB appl. buffer/Maxtor MaXLine Plus II 250 GB SATA disk).

4.1. Filesystem

The ext3 filesystem on the xio nodes undoubtedly impacts the overall performance of the system. However, the experiment described in Figure 3 demonstrates that this filesystem can support transfer rates close to the raw data transfer rates of the disk when the FAStT600 system is bypassed. Thus, the ext3 filesystem is not a major factor impacting on the system performance.

4.2. Bus bandwidth

The internal buses of the xio processors are designed to support 2.4 GHz Pentium 4 processors and have bandwidths of the order of GB s^{-1}. These cannot cause bottlenecks as far as disk I/O is concerned.

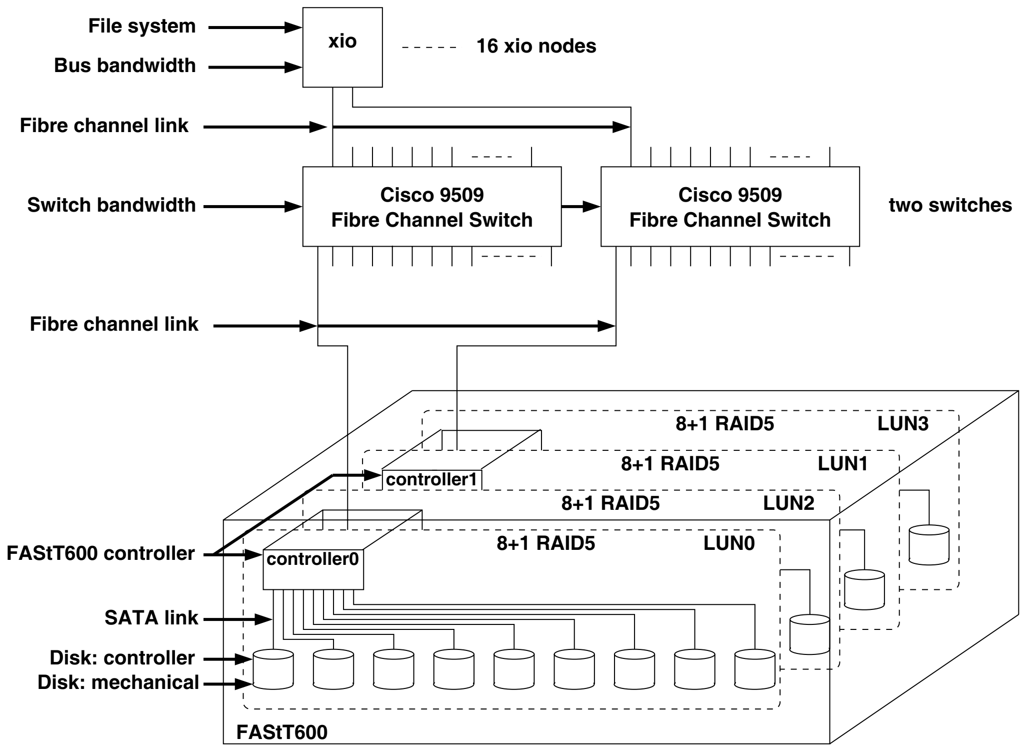

Figure 4. Potential bottlenecks in the system.

4.3. Fibre channel links

There are two fibre channel links between the FAStT600 controllers and each xio processor. Each link has a bandwidth of 212.5 MB s^{-1}, giving a combined bandwidth of 425 MB s^{-1} per xio processor. This is considerably greater than the maximum data rates we have measured to/from the disks when the FAStT600 is bypassed.

4.4. Cisco switch

This is a non-blocking switch fully capable of supporting 2 GB s^{-1} traffic at each port [8]. Furthermore, it is used to *statically* connect FAStT600s to xio processors, so there is no occurrence of switching delays. This switch can therefore support the 425 MB s^{-1} traffic offered by the fibre channel links.

4.5. Disk: SATA link, mechanical and controller

The peak SATA data rate is 150 MB s^{-1} (Table I). The sustained data rate is likely to be much lower. The Maxtor 7Y250M0 disk specifications shown in Table II claim 37–67 MB s^{-1} sustained transfer rates. Experiments carried out on directly connected disks by us (Figure 3) and others [11] have confirmed sustained transfer rates of 34.5–60.5 MB s^{-1}. Since at least 36 disks are connected to each controller, we have a cumulative offered rate of at least $(34.5 \times 36 =)$ 1.242 GB s^{-1} to the FAStT600 controller. Thus, the disk unit (comprising the SATA link, disk controller and magnetic media) cannot be the bottleneck.

4.6. FAStT600 controller

The unit labeled as the FAStT600 controller is in fact made up of two host bus adapters connected to two RAID controllers [6,7]. The internal behavior of the RAID controllers is unknown and we speculate that these are the major bottlenecks in the system.

5. STAGE 3: SERIES OF MEASUREMENTS

5.1. The six experimental configurations

In line with the methodology described earlier in the paper, we performed a series of six experiments on RAIDs varying in size from 1–36 disks, as shown in Figure 5. For convenience we use the notation nC:$m(r)$ to represent a setup with n controllers and m LUNs and (r) denotes the RAID5 configuration within a LUN. For example, configuration 1C:$2 \times (8+1)$ is a single-controller setup with two $(8+1)$ RAIDs. The objective of this series of experiments is to demonstrate that the performance of successively larger RAIDs increases smoothly and that the maximum performance (of the full 2C:$4 \times (8+1)$ RAID) is predicted by the smaller RAIDs.

5.2. Read experiments

To test read performance, we first faced the obstacle of how to clear out various kinds of cache. For large enough files, this is a non-issue as there will eventually be enough cache turnover, but we wanted to test the performance of smaller files as well. Since we could not *reboot* the system or *re-mount* the filesystems, we needed another option. To clear the kernel cache, we first wrote out a file twice the RAM size of the machine, flushed it to disk via the 'sync' command, and then deleted it. To further ensure that the FAStT600 controller cache was cleared, we then read through a 'dummy' file that was twice the size of the controller RAM (which is 1 GB). After doing all of this before each run, we did not observe the effects of the cache distorting the results. To carry out the reads, we spawned between one and an arbitrary number of threads, dividing up the data to be accessed as evenly as possible among threads. Each thread would access its designated contiguous region of a file and would then exit. Every byte of the target file was accessed only once. A detailed discussion of how we divided the file data among threads is given in [12]. We varied file size and application buffer size along with the number of threads to try to find the peak performance.

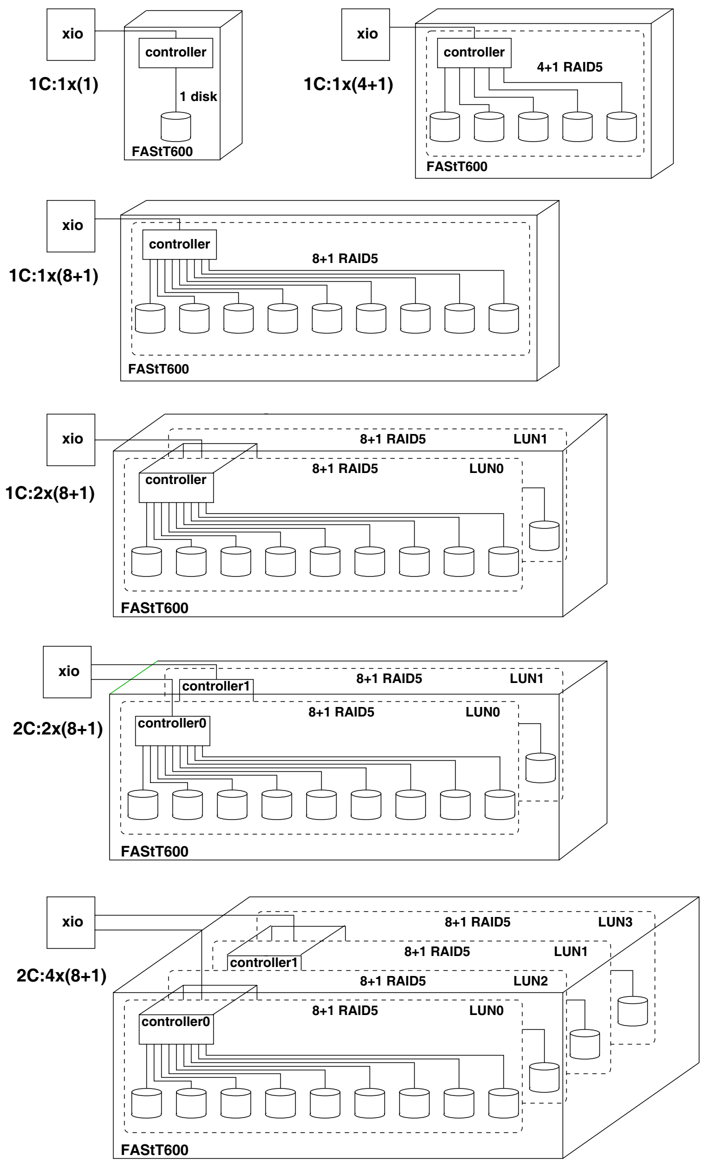

Figure 5. The six experimental configurations. nC:$m \times (r)$ denotes a setup with n controllers, m LUNs and RAID5 configuration (r). For example, 1C:$2 \times (8 + 1)$ is a single-controller setup with two $(8 + 1)$ RAID5s.

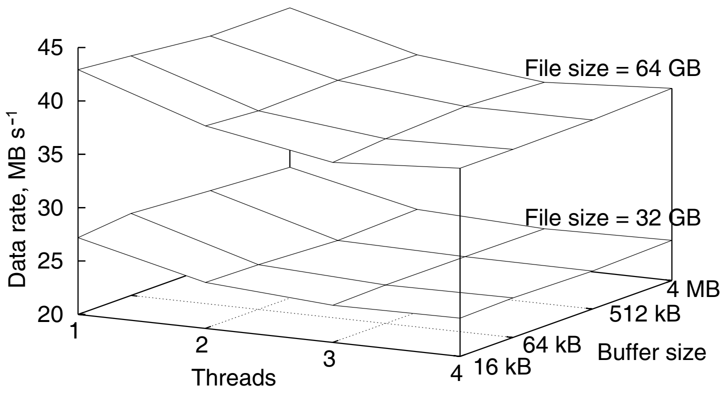

Figure 6. Read performance of 1C:1 × (1), a single disk controlled by a FAStT600 unit.

5.2.1. 1C:1 × (1)*: a single disk*

A single disk connected to a FAStT600 does not constitute a RAID as such. However, the measured performance (Figure 6) gives valuable insight into the overheads in the FAStT600 controllers. In this case, there is no advantage to multi-threading as multiple threads attempting to read from the same disk will only lead to contention. Buffer size has little impact. The delivered data rate increases significantly with file size and, interestingly, the maximum obtained is 43 MB s^{-1}, which is greater than the *minimum* direct read data rate of 34.5 MB s^{-1} (from inner-track reads) reported in Figure 3. This is because a 64 GB file makes up about 25% of the capacity of a single disk and thus cannot be confined to the (low-data-rate) inner tracks.

5.2.2. 1C:1 × (4 + 1)*: a five-disk RAID5*

In this experiment (Figure 7), a five-disk RAID5 was set up (four data disks and one parity disk). There is a small positive impact of increasing buffer size and a *decrease* in performance with increased multi-threading. The latter is most likely because the system cannot support multiple requests. The maximum data read rate achieved is about 85 MB s^{-1}. This is half the theoretical figure (based on the inner-track, direct read rate) of $[(4 + 1) \times 34.5] = 172.5$ MB s^{-1}.

5.2.3. Data zones and the FAStT600

On the 4 + 1 RAID5 we also carried out a test to see whether read times varied for files created early in an empty filesystem versus files created later. On a single disk connected directly to a PC there is a clear variation as described in Figure 3 and [11]. Early files are expected to be created near the outer tracks resulting in faster reads than for later files, which are created near the inner rings. We created 16 files that nearly filled the 1 TB 4 + 1 RAID5 virtual partition. Each file was 54.5 GB, and the

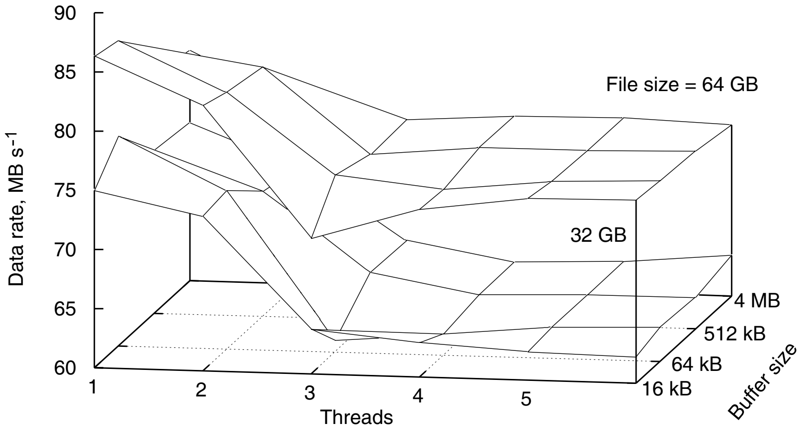

Figure 7. Read performance of 1C:1 × (4 + 1), a one controller, five disk RAID5.

Table III. Read data rates (MB s^{-1}) for 16 files that almost fill a 4 + 1 RAID5. Single thread, 256 MB buffer size.

File	Rate	File	Rate
1	86.14	9	86.02
2	85.95	10	86.18
3	85.77	11	86.38
4	86.05	12	86.25
5	86.11	13	86.20
6	85.99	14	86.11
7	86.10	15	85.48
8	86.31	16	85.10

16 files collectively amounted to 874 GB out of 928 GB total available space (i.e. 95%). With such a high utilization, the impacts of zoned recording (if any) would be clearly evident. However, read access times were consistent for all 16 files, as shown in Table III. We conjecture that the RAID controller hardware of the FAStT600 makes no assumptions about the zone on which a particular block lands. Thus, the maximum read rate of the FAStT600 cannot exceed the *minimum* rate that applies to the blocks on the innermost tracks of the disks multiplied by the number of disks. In fact, the rate for a 4 + 1 RAID5 is approximately (86/5 =) 17.2 MB s^{-1}, which is about half the inner-track 'raw' rate from Figure 3 and [11]. The policies used to allocate blocks in RAID units influence the effective data rate;

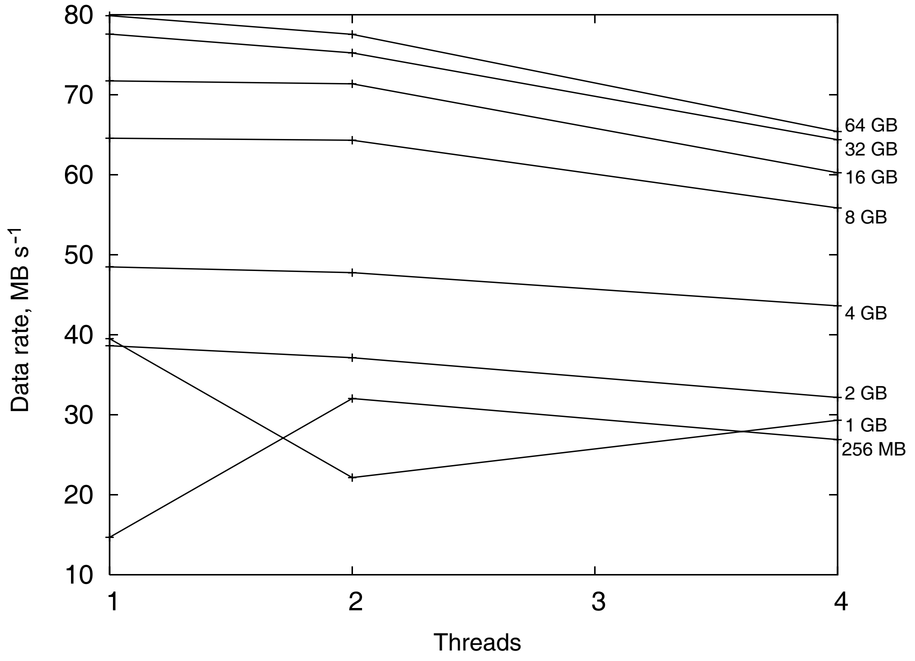

Figure 8. Read performance of 1C:1 $\times$ (8 + 1), a one controller, nine disk RAID5, 4 MB buffer size.

some of these are discussed in [5, Section 5.2]. Unfortunately, we have no knowledge of what is actually implemented inside the FAStT600.

5.2.4. 1C:1 $\times$ (8 + 1)*: a nine disk RAID5*

The RAID5 unit evaluated in this experiment has nine disks (eight data and one parity) and corresponds exactly to a quarter of the full four LUNs setup in the FAStT600 data pool used at the OSC. Peak read rates obtained are 80 MB s^{-1} for 64 GB files (Figure 8). There is the very minor impact of buffer size and we have thus illustrated only 4 MB buffer sizes in this and the following plots. There is again no advantage gained by multi-threading.

5.2.5. 1C:2 $\times$ (8 + 1)*: one controller and 18 disks*

This configuration has two (8 + 1) RAID5s connected to one controller (Figure 9). Performance is significantly better than the performance of a single (8 + 1) RAID5. Furthermore, there is a small increase in performance as threads increase, demonstrating that multi-threading is having a beneficial impact in keeping the disks busy. There is an anomaly in the timings for this setup: the data rates for 64 GB files (marked with an asterisk in Figure 9) are about half those for 32 GB files.

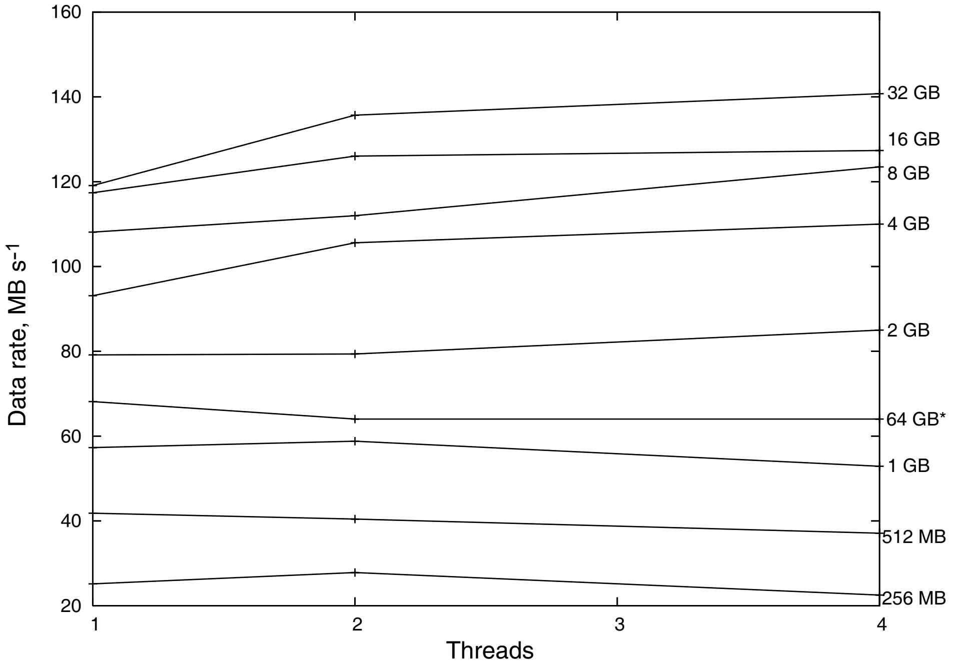

Figure 9. Read performance of 1C:2 × (8 + 1), a one controller system with two nine-disk RAID5s for 4 MB buffer size. Note the anomalous rates for 64 GB files.

This is probably because the controller has reached the limit of its performance with 32 GB files and any further increase in load results in a catastrophic decrease in performance. It is significant that the two-controller version of this setup (described below) does not suffer from this anomaly.

5.2.6. 2C:2 × (8 + 1)*: two controllers and 18 disks*

This configuration (Figure 10) has a definite but not substantial improvement over the corresponding single-controller configuration. There are no anomalies of the type discussed above, but the data rates for large file sizes (8–64 GB) are tightly clustered, showing that the system has begun to saturate. (In Figure 10, the plots for 16 and 32 GB lie between the 8 and 64 GB plots but have been omitted to avoid congesting the figure.) For large file sizes, there are distinct improvements in performance as the number of threads increases from two to four.

5.2.7. 2C:4 × (8 + 1)*: the full 36-disk RAID5*

Figure 11 describes the sustained read performance of the full four-way (8 + 1) RAID5 unit. It is clear that for very large files (2–64 GB) the maximum achieved data rate is in the 250–280 MB s^{-1} range.

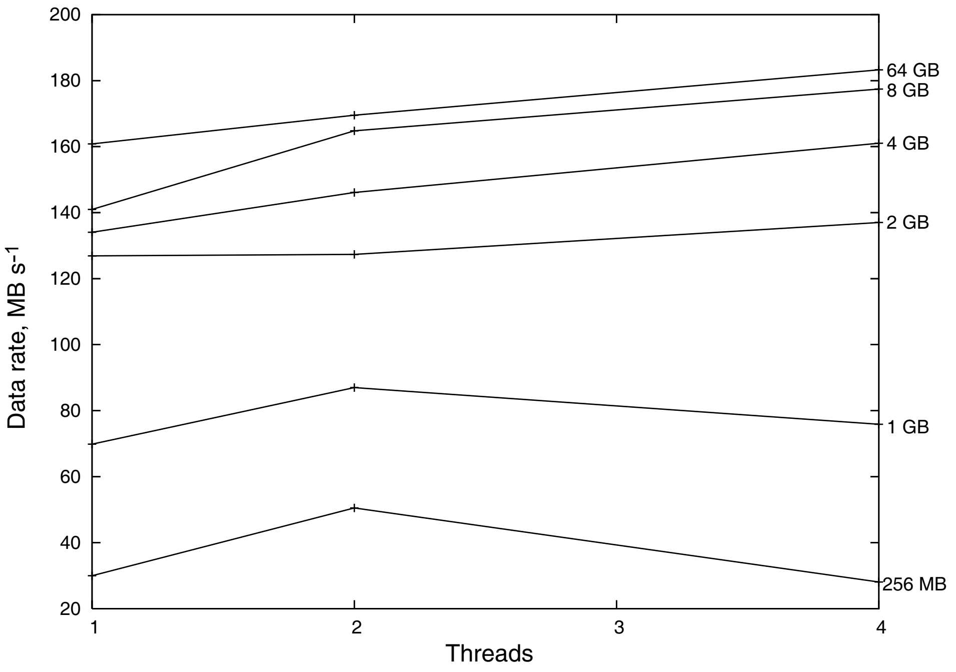

Figure 10. Read performance of 2C:2 × (8 + 1): a two-controller system with two nine-disk RAID5s, 4 MB buffer size.

This maximum rate is achieved for four threads, emphasizing the need to use multi-threading to keep the xio and the FAStT600 busy. We have omitted some plots in this figure for clarity. Figure 12 shows a fine-grained analysis of the full four-way (8 + 1) RAID5 for a 1 GB file. This surface demonstrates that it is clearly advantageous to use multiple threads because: (a) the xio nodes are effectively four-processor units and can only be kept busy with more than four threads; and (b) the FAStT600 has the capacity to respond to these multiple threaded requests. Of course, this improvement in performance with increasing number of threads cannot continue indefinitely. The overhead of handling threads exceeds the advantage of multi-threading after about eight threads.

5.3. Analyzing results for reads

The peak read rates obtained in the preceding sections are somewhat disappointing. However, do the experimental observations of variously sized RAID units form a coherent whole? To answer this question we plotted the read rates obtained for 64 GB files using four threads and a buffer size of 64 MB for each of the six experiments. Figure 13 shows the surface obtained. The points on this surface corresponds to the points (four threads, 4 MB buffer size, 64 GB file size) from each of

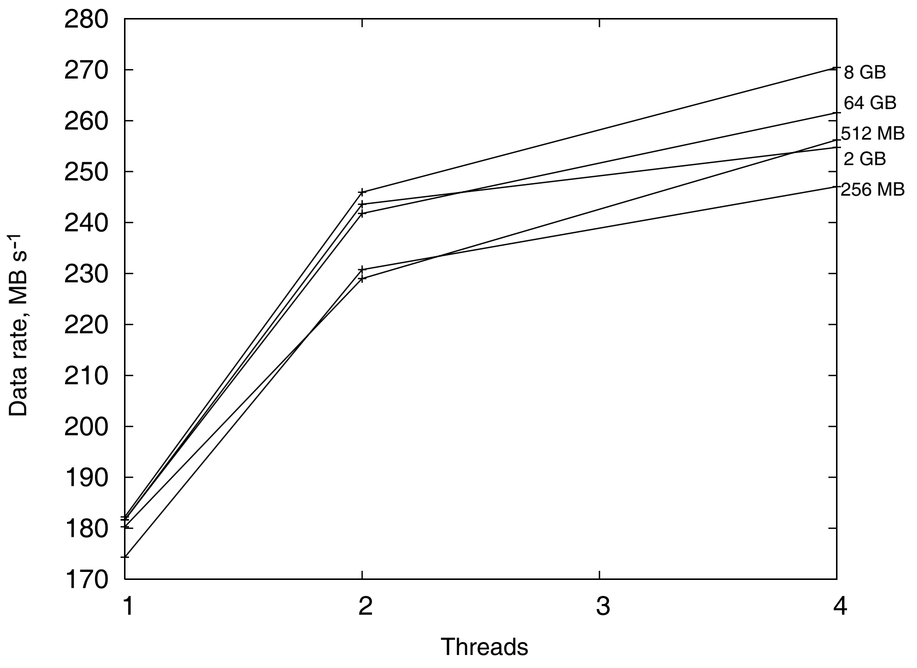

Figure 11. Read performance of 2C:4 × (8 + 1), the full system with two controllers and four nine-disk RAID5s.

the six experiments. The surface of Figure 13 shows reasonable progression in data rate as system resources increase and its shape is plausible, with no abrupt discontinuities. We have differentiated the points that correspond to one- or two-controller configurations. The bifurcation in the surface is caused by this difference.

The surface of Figure 13 demonstrates that the six experiments we have carried out are part of a coherent series and, in particular, that the peak read rates for the full two-controller four-way 8 + 1 RAID follow from the trend set by the smaller RAIDs.

5.4. Write experiments

To test write performance, we ran a large number of individual tests in isolation, each of which wrote dummy data to a single new file from a single thread, and varied file size and application buffer size to try to find the peak performance. Several reasons motivated our choice of using a single thread to write to new files (as opposed to using a variable number of threads to overwrite sections of existing files). For one, we discovered that performance for the four-way 8 + 1 setup was worse when two or three threads were used to overwrite existing sections of a file, compared with just using a single thread. Next, while focusing our attention on the one-thread case, we discovered that performance was better

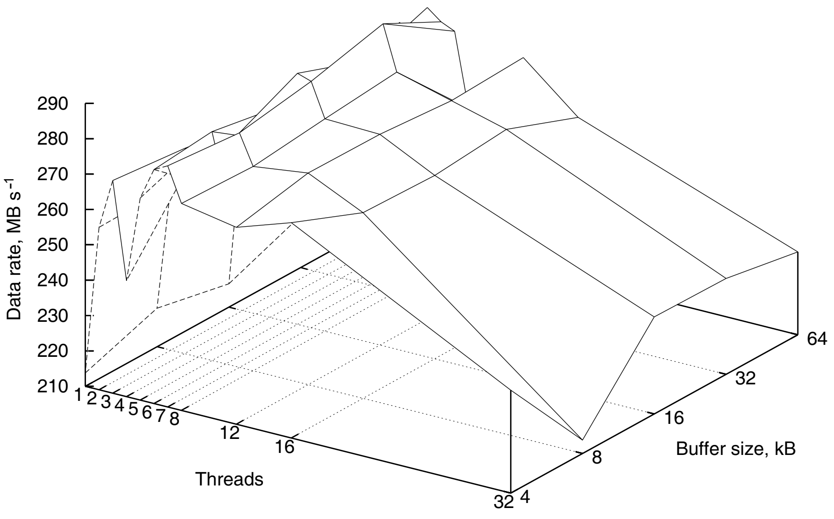

Figure 12. Detailed study of the impact of multi-threading and buffer size on data rate in full system (1 GB file).

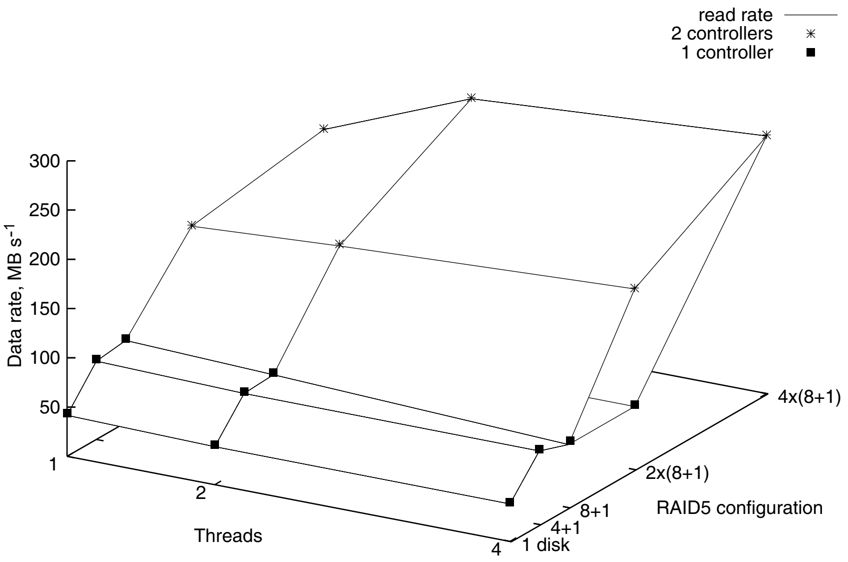

Figure 13. A unified view of 64 GB file read times for all RAID configurations studied in this article.

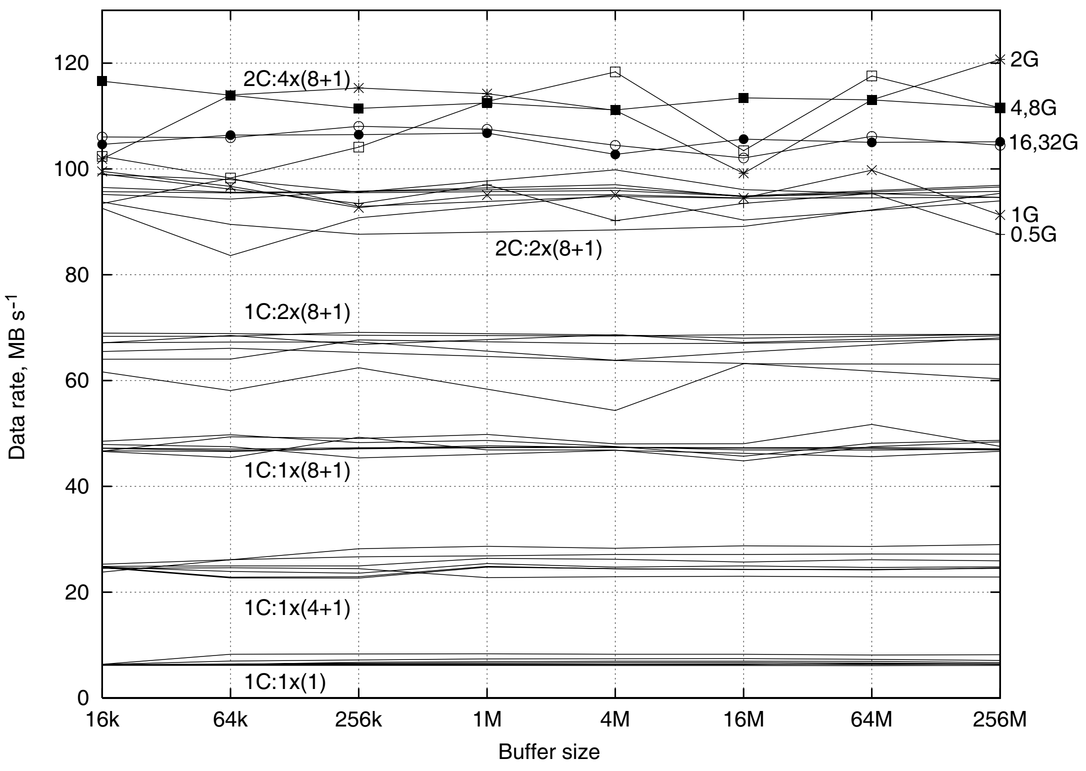

Figure 14. Single-thread write tests on various RAID5 configurations. The plots for configuration 2C:4 × (8 + 1) have large spreads and are indicated with points.

when we created new files, versus overwriting old files. For a detailed discussion of why only a single thread can be used for creating new files, see [12].

For write tests, in order to get fair results for this system, it was necessary to perform the tests using file sizes sufficiently large that the effects of write buffering were eliminated. When experimenting with file sizes in the range of 8–256 MB, we found that the write performance was inflated to the point of proving that write buffering was having an effect on performance. For example, we discovered an 85 MB s^{-1} write performance for the single disk in a LUN setup (the setup denoted 1C:1×(1) in Figure 5), which is impossible when the disk's stated best access time is only 67 MB s^{-1}, according to manufacturer specifications. By using large enough file sizes, we eliminate the impact of write buffering, as the system cannot buffer all of a large file; that is, eventually the write buffers will run out of room and will necessitate that the data are flushed to disk. Figure 14 presents the results of write experiments on all the configurations of Figure 5. For each configuration, measured data rates for file sizes of 0.5–32 GB are presented, for buffer sizes varying from 16–256 MB. For the smaller configurations, there is essentially no variation of data rate with either buffer size or file size.

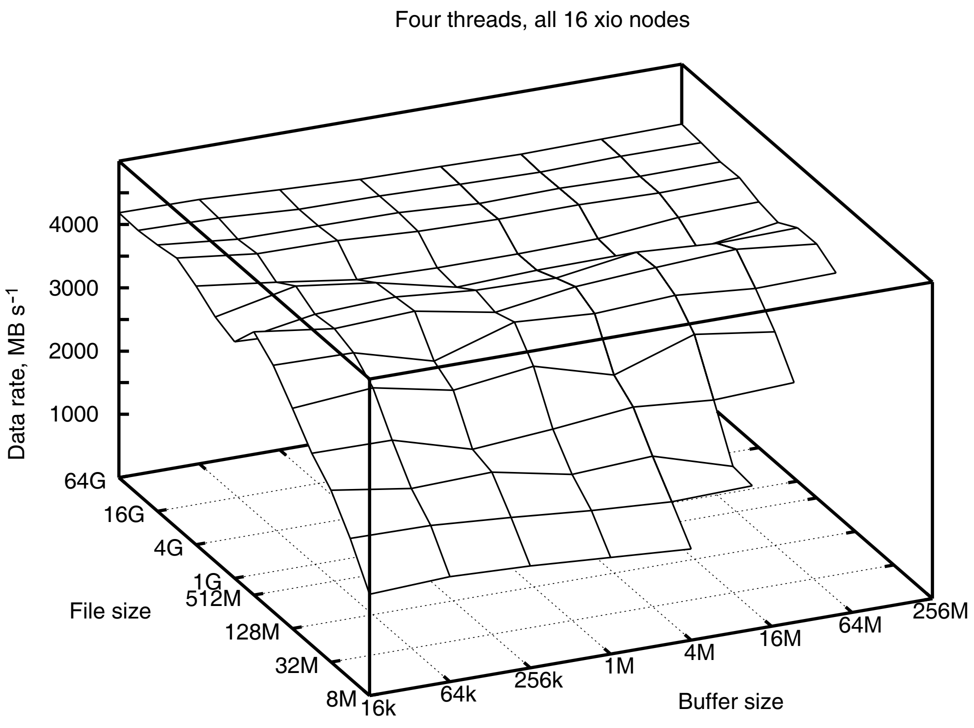

Figure 15. Aggregate bandwidth of 16-way parallel read test. The maximum bandwidth obtained is 4.3 GB s^{-1}.

The achieved write data rate is a small fraction of the read data reported above for these RAID sizes. For the full 2C:4×(8 + 1) RAID, whose plots are marked with points in Figure 14, there is a spread of data rates, but this is uncorrelated with either file or buffer size. We conjecture that this spread is due to probabilistic effects arising from the mapping of 256 kB blocks of the four-way RAID0s in the xio processors onto the 64 kB blocks of the four-way (8 + 1) RAID5s in the FAStT600s (Figure 2). As there are two controllers in the FAStT600 managing four LUNs, some amount of contention must necessarily arise. The maximum write data rate achieved is about 120 MB s^{-1} compared with a maximum read rate of approximately 280 MB s^{-1}.

5.5. 16-way parallel read test

We carried out this test to evaluate the aggregate bandwidth achieved by the system when all 16 xio processors are active, each accessing its own filesystem. This stresses the fibre channel switch to the maximum extent possible. In this test, we used four threads per machine and started all 16 reads in unison. The results, shown in Figure 15, indicate that the maximum aggregate data rate is 4262 MB s^{-1}, which corresponds to 266 MB s^{-1} average per machine, a figure quite close to our measured maximum of 280 MB s^{-1}. The 4.3 GB s^{-1} figure is a high watermark that serves as a reference point for all applications developed for this system.

6. CONCLUSIONS

We have presented a systematic methodology for empirically evaluating the performance of a MSS. This methodology has been used to evaluate the performance of sustained read and write data rates for the FAStT600 storage pool at the OSC. We have shown that the maximum sustained read and write data rates for this system are 280 MB s^{-1} and 120 MB s^{-1}, respectively (per storage node). The write figures are, however, well below the theoretical factor of $\frac{8}{9}$ of the read rate. The FASTt600 controller appears to be the bottleneck in the OSC system.

Our recommendations for developers of application programs for the OSC FAStT600 storage pool are as follows. (Although these guidelines are for a specific MSS, they are likely to be of relevance to other systems as well.)

(1) Multiple threads should be used for reads. The developer should experiment with four to eight threads to identify the number that gives the best results. As the xio is effectively a four-processor machine, more than four threads are normally needed to keep it fully occupied.
(2) Multiple threads may be useful when updating parts of an existing large file. However, when creating a large file, there is no option but to use a single thread with attendant low performance.
(3) Our read results apply to sustained reads of very large files (16–64 GB). For smaller files, the sustained read rates will probably be poorer. For very small files, the rates will rise again because of the effects of caching in various parts of the system.

For our future work we plan to apply the methodology described here to other storage architectures, including a high-throughput RAID3 configuration using fibre channel or SATA disks, and a low-cost controller-poor RAID5 configuration of SATA disks.

We have attempted to demonstrate the value of our methodology in clarifying the limitations to performance of large RAID systems. With the proliferation of large databases, we expect larger and larger MSSs to be designed and implemented in the future. The experimental analysis methodology described and exemplified in this paper will be of value to future systems as well. An organization planning to acquire a large number of RAID controllers incorporating hundreds or thousands of disks can gain valuable insight into the performance of the proposed system by first carrying out a set of experiments, following our methodology, on a single controller and its associated disks.

ACKNOWLEDGEMENTS

The authors are indebted to Joel Saltz and Stan Ahalt for their encouragement of this research. They thank Rick Mohr for acquiring a Maxtor disk and helping to set it up for the stand-alone experiments. Shahid Bokhari and Benjamin Rutt are grateful to Tahsin Kurc, Umit Catalyurek, Michael X. Zhang, and Leslie Southern for many useful discussions. Shahid Bokhari thanks Kelly Sitz and Barbara Woodall for arranging access to the OSC. The authors also thank the anonymous referees for their valuable comments, which have helped improve the presentation of this paper.

REFERENCES

1. Chen PM, Lee EK, Gibson GA, Katz RH, Patterson DA. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys* 1994; **26**(2):145–185.
2. Ruemmler C, Wilkes J. An introduction to disk drive modeling. *IEEE Computer* 1994; **27**(3):17–28.
3. Ng SW. Advances in disk technology: Performance issues. *IEEE Computer* 1998; **31**(5):75–81.
4. Gray J, Horst B, Walker M. Parity striping of disk arrays: Low-cost reliable storage with acceptable throughput. *Proceedings of the 16th International Conference on Very Large Data Bases 1990*, McLeod D, Sacks-Davis R, Schek H-J (eds.). Morgan Kaufmann: San Francisco, CA, 1990; 148–161.
5. Thomasian A. Data allocation and scheduling in disks and disk arrays. *Performance Tools and Applications to Networked Systems: Revised Tutorial Lectures* (*Lecture Notes in Computer Science*, vol. 2965), Calzarossa MC, Gelenbe E (eds.). Springer: Berlin, 2004; 357–384.
6. IBM Corporation. IBM TotalStorage DS4300. *Technical Report TSD00007-USEN-06.pdf*.
7. IBM Corporation. IBM TotalStorage Product Guide. *Technical Report TSB00364-USEN-12*.
8. Cisco Corporation. Cisco MDS 9500 series of multilayer directors. http://www.cisco.com/en/US/products/ps5990/prod_brochure09186a00801ce93e.html [17 August 2005].
9. Maxtor Corporation. MaxLine plus II 250 GB AT product manual. Part Number: 1905.
10. Ra E. Maxtor MaXLine Plus II 250 GB SATA Edition, Introduction. http://www.storagereview.com/articles/200309/200309147Y250M0_1.html [17 August 2005].
11. Ra E. Maxtor MaXLine Plus II 250 GB SATA Edition, Low-Level results. http://www.storagereview.com/articles/200309/200309147Y250M0_2.html [17 August 2005].
12. Bokhari S, Rutt B, Wyckoff P, Buerger P. An evaluation of the OSC FAStT600 turbo storage pool. *Technical Report OSUBMI TR 2004 n02*, Department of Biomedical Informatics, Ohio State University, 2004. http://web.bmi.ohio-state.edu/resources/techreports/OSUBMI_TR_2004_n02.pdf [17 August 2005].