Performance of a Disk Array Prototype

Ann L. Chervenak and Randy H. Katz
Computer Science Division
Electrical Engineering and Computer Sciences
University of California
Berkeley, California 94720

Abstract

The RAID group at U.C. Berkeley recently built a prototype disk array. This paper examines the performance limits of each component of the array using SCSI bus traces, Sprite operating system traces and user programs.

The array performs successfully for a workload of small, random I/O operations, achieving 275 I/Os per second on 14 disks before the Sun4/280 host becomes CPU-limited. The prototype is less successful in delivering high throughput for large, sequential operations. Memory system contention on the Sun4/280 host limits throughput to 2.3 MBytes/sec under the Sprite Operating System. Throughput is also limited by the bandwidth supported by the VME backplane, disk controller and disks, and overheads associated with the SCSI protocol.

We conclude that merely using a powerful host CPU and many disks will not provide the full bandwidth possible from disk arrays. Host memory bandwidth and throughput of disk controllers are equally important. In addition, operating systems should avoid unnecessary copy and cache flush operations that can saturate the host memory system.

1 Introduction

The increasing performance gap between CPUs and I/O systems threatens to create an I/O bottleneck that will limit overall system performance [Patt88]. Disk arrays are one attractive solution: enormous I/O bandwidth can be achieved if many disks are accessed in parallel on an array. RAID (Redundant Arrays of Inexpensive Disks) adds redundancy in an attempt to make arrays at least as reliable as a single, large disk.

The RAID group at U.C. Berkeley recently built a prototype block-interleaved disk array. This paper examines the performance limits of each component of the array using Sprite operating system traces, user programs and SCSI bus traces. Performance of software for implementing redundancy is considered elsewhere [Cher90,Lee90,Lee91].

We wanted to learn whether a disk array built from off-the-shelf parts could deliver adequate performance for two different workloads. A disk array has many independent disk arms and can perform many small, random I/Os in parallel. Such I/Os are typically generated by database applications and traditional file systems; the metric that describes such activity is the number of I/Os serviced per second. Our prototype performed successfully for this workload, achieving 275 I/Os per second before becoming CPU limited.

A second performance goal for a disk array is high throughput for large, sequential I/O operations typically generated by scientific computing applications; the metric for this workload is the number of megabytes transferred per second. We expected that the VME backplane bandwidth of the host computer would make this latter goal more difficult to achieve. In practice, the VME backplane was an even more severe performance limit than expected, and throughput was also limited by the bandwidth of the host memory system, controller and SCSI string. Under Sprite, the array delivered only 2.3 MBytes/sec.

Section 2 of this paper offers background information, including the hardware configuration of the prototype and an introduction to the performance limits of each component. It also traces an I/O through the Sprite operating system to the disk and back, and discusses the experiments performed for this study. Section 3 discusses the performance of each component and of the entire array. Section 4 draws conclusions about the prototype and discusses changes that would improve performance.

2 Background

2.1 Prototype Hardware Configuration and Performance

The hardware configuration of the prototype is shown in Figure 1. It is composed of off-the-shelf components that use standard interfaces. A Sun4/280 workstation

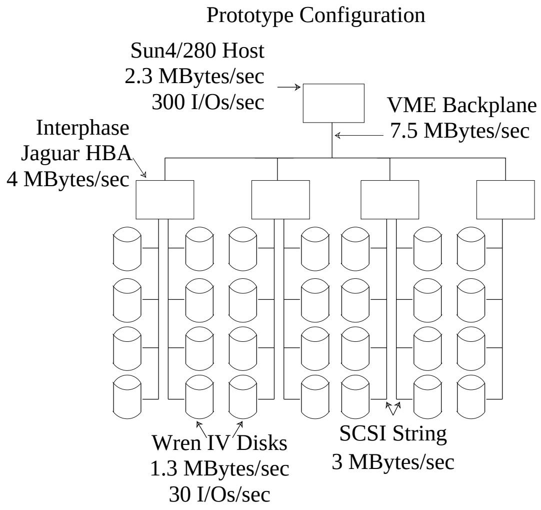

Figure 1: *This illustrates the configuration of the prototype, including the performance of each component.*

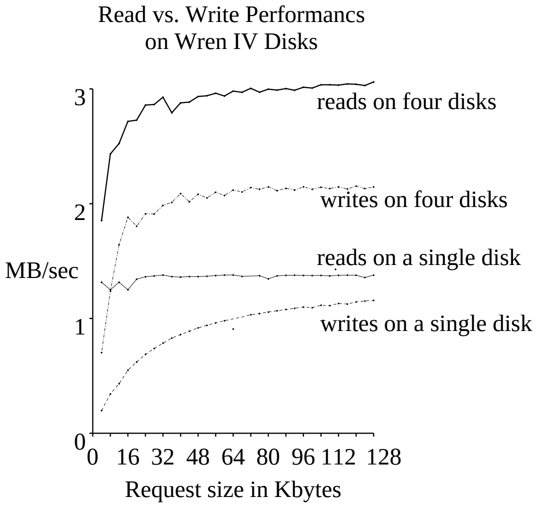

Figure 2: *This graph compares the read and write performance of Wren IV disks for one disk and for four disks on a single SCSI string. Write performance is significantly worse than read performance because the Wren IVs make no use of the disk track buffers to gain additional performance on writes.*

serves as host processor and is attached over a VME backplane to four Interphase Jaguar Host Bus Adaptors (HBAs) [Jaguar]. Each Jaguar HBA or controller manages up to two strings of 5 1/4" Imprimis Wren IV disks [Wren]; the HBA and disks communicate using the SCSI protocol [SCSI]. There are a total of seven SCSI strings, each with four disks attached for a total of 28 disks in the array. The Sprite Operating System [Oust88], a UNIX-compatible operating system designed for networked workstations, runs on the Sun4/280.

Figure 1 also shows the performance limits of each component, examined in detail in Section 3. The Wren IV disks are limited to 1.3 MBytes per second maximum throughput on large, sequential operations. They perform 30 small, random I/Os per second. The SCSI strings are limited in practice to 3 MBytes per second, although we had expected to achieve 4 MBytes/sec. The Interphase Jaguar HBAs can only support 4 MBytes per second from the two strings that they control. The VME backplane provided only about 7.5 MBytes per second, significantly lower than expected. Finally, the Sun4/280 server running Sprite delivered only 2.3 MBytes/sec on large sequential operations and about 275 small random I/Os/sec.

2.2 Trace of an I/O Through Sprite

To understand where bottlenecks are introduced, it is useful to trace an I/O from the application to the disk and back. We begin at Sprite user level, where an application issues a raw device read (or write). The operating system executes the code for a system call, mapping the data buffers in user space into the kernel address space. Next, a "generic" Sprite device driver and a Jaguar HBA-specific device driver set up necessary data structures and submit the command to the HBA. The HBA processes the request (allocates buffers, performs SCSI protocol) and sends the request to the disk. If the request is a read and follows sequentially from the last request, then the disk buffer may already contain the requested data if the disk is configured to read ahead. Otherwise, the disk positions the arm and reads data into its buffer. When a request resides completely in the disk buffer or the minimum amount required for data transfer is in the buffer, the disk transfers data across the SCSI string to the HBA. The HBA uses DMA to write the data to server memory.

Writes behave similarly, except that they do not receive the performance advantage for sequential operations that reads achieve by reading data ahead. The HBA transfers data from the host memory into its own buffers, and then across the SCSI bus to the disk buffer. The disk writes the data to the medium when the arm is properly positioned. Since sequential writes do not get the performance benefits of read ahead, their performance is significantly worse than sequential read performance, as shown in Figure 2. Because this report is concerned with the maximum capabilities of hardware and software on the prototype, we will focus on read operations.

When a disk operation is complete, the HBA interrupts the server. The server completes processing of the operation by copying (in the read case) the data, which has been written to buffers in kernel space, to the address space of the user process that requested it. On the Sun4/280, all I/O is done through the virtual cache, so in addition to the data copy, the buffer used to DMA the data into kernel space must be flushed from the cache.

The copy and cache flush operations incurred during normal Sprite processing tend to limit performance, particularly for large, sequential operations, by using up precious memory system bandwidth. To minimize the effect of software overhead, we wrote a system call (hereafter referred to as the "no-copy" system call) that accepts a buffer containing up to 512 commands and puts them all on the device driver queue. This eliminates much of the Sprite overhead, since returning to user level between commands is not required. Also, this system call allocates a single buffer in the virtual cache where the data from all I/O operations is written. Subsequent operations overwrite this buffer; the data written to the buffer is never copied to the user's address space, but is discarded. Cache flushes between operations are also eliminated.

2.3 Methods

We used several measurement tools in this study. An Ancot SCSI bus analyzer traced activity on the SCSI bus. A modified Sprite kernel using a high-resolution timer traced operating system behavior. User-level programs generated artificial workloads for the prototype and recorded the throughput provided by the array. Some components were difficult to measure directly. We lacked information on internal processing in the Jaguar HBA and the Wren IV disks. These components were treated as "black boxes", and the behavior at their interfaces was observed and interpreted.

An important measurement during testing was the utilization of the Sun4/280 host processor and memory system. Measuring CPU utilization on the prototype is not straightforward, since under intense I/O activity, we believe that the server becomes performance-limited by contention on the memory system. This is caused by the large number of copy and cache flush operations that occur under Sprite. These DMA operations have priority in acquiring the memory bus. If the memory system is very busy with these DMA operations, the CPU will appear busy, because it is slow to fetch the instructions it executes as it idles. Thus, high utilization may reflect contention on the memory bus rather than activity in the CPU. In Section 3, such measurements are called "server" or "host" utilization measurements. They reflect the combined utilization of the CPU and memory system, and are used for comparison between workloads.

3 Performance

This section describes the performance of the prototype components. Sections 3.1 and 3.2 discuss the performance of disks on a single SCSI string for sequential and random operations, respectively. Section 3.3 measures the bandwidth sustained by the Jaguar HBA. Finally, Section 3.4 looks at the bandwidth and I/O rates achieved by the entire array.

3.1 Sequential Reads on a String

This section is divided into two parts. Section 3.1.1 describes sequential I/O issued by the special no-copy system call (designed to eliminate much of the copy and cache flush activity of normal Sprite), and reveals the bandwidth limitation on a single SCSI string caused by SCSI overheads. Section 3.1.2 shows measurements of activity issued as raw device I/O from Sprite user level.

3.1.1 Sequential Reads Issued by the No-Copy System Call

To trace operating system activity for I/Os issued by the no-copy system call, a modified Sprite kernel recorded the following intervals for I/O:

- Time to Submit Command: The interval from the time that the Jaguar HBA indicates it is ready to accept a new command until the command is actually submitted to the HBA. The code for this processing is in the Sprite device driver for the Jaguar HBA.

- Time on Jaguar HBA and Wren IV Disk: Time between the server submitting the command to the Jaguar board and the server receiving an interrupt indicating command completion. This time includes setup time on the Jaguar board, SCSI protocol implementation on the Jaguar and the Wren, seek time, rotation time, and transfer time on the disk.

- Interrupt Processing: The interval between the completion interrupt from the Jaguar and the time that the Jaguar is ready to accept a new command.

Table 1 shows an operating system trace of 32 KByte sequential read operations issued by the no-copy system call. Very little time (less than 200 usec) is spent in the Jaguar device driver between subsequent commands. The majority of time (24.5 msec) is spent on the Jaguar board and on the disk. This corresponds closely to the time (24 msec) required to transfer 32 KBytes of data off the disk head at approximately 1.3 MBytes/sec. The Wren IV disk performs read-ahead, increasing data transfer efficiency. Only adjacent cylinder seeks are required for sequential operations.

Figure 3 shows the bandwidth attained with up to four disks on a single SCSI string performing sequential reads issued by the no-copy system call. The first disk on the string is able to achieve 1.3 MBytes/sec, as much as the Wren IV can deliver. When a second disk is added, the bandwidth approximately doubles for

Time to submit command	120 us
Time on Jaguar board	24500 us
Interrupt processing	57 us

Table 1: *Trace of 32 KByte sequential read operations issued from the special no-copy system call. Since 512 operations are issued to the Jaguar device driver at a time, processing between operations stays in the device driver and is very brief.*

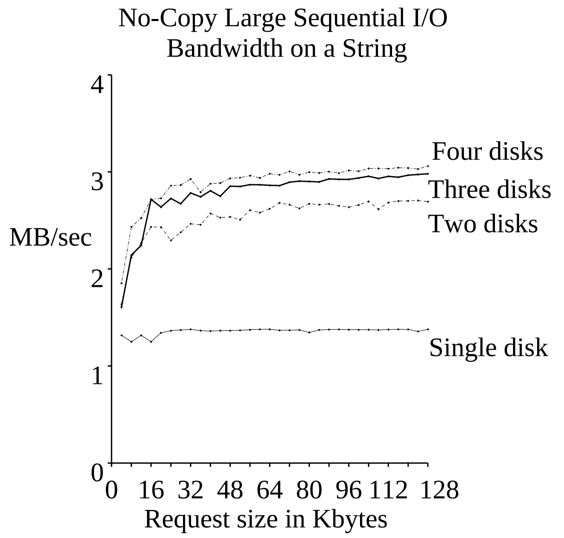

Figure 3: *Bandwidth for sequential read operations issued using the no-copy system call to disks on a single string. A separate process issues requests to each active disk. Because of SCSI overheads, bandwidth on a string is limited to 3 MBytes/sec.*

large request sizes. However, a third and fourth active disk provide very little performance improvement. The maximum bandwidth obtained on the string is around 3 MBytes/sec. This is only 75 % of the 4 MByte/sec string data transfer rate negotiated between the Wren IV disk and the Jaguar HBA on power-up. Trace analysis shows that data bytes do pass between the devices at 250 nsec per byte (4 MBytes/sec). The sustained transfer rate is limited to 3 MBytes/sec by overheads associated with SCSI protocol implementation on the disk and controller. SCSI overheads are described in detail in [Cher90].

Table 2 lists the percentage of time and the normalized time per I/O spent in each phase of the SCSI protocol for a trace of 32 KByte sequential reads. A separate process issued these reads to each of four disks on a single SCSI string. The time spent in the arbitration phase is very short, indicating that four active disks consume very little SCSI bandwidth with arbitration overhead. Most of the other SCSI phases (messages, selection, disconnects, reconnects, etc.) similarly account for very little time in the life of the transaction. Large sequential transfers spend 93% of their time in

Time per SCSI Phase	Percentage (%)	Normalized (usec)
Arbitration	0.06	6.5
Selection	0.20	22
Command	4.02	430
Message Out	0.72	77
Message In	0.22	24
Data Transfer	93.15	10000
Disconnect/Reconnect	0.11	12
Reselection	0.02	2.0
Status	0.45	48
Bus Free	1.05	

Table 2: *SCSI phase trace of 32 KByte sequential reads issued by no-copy system call. Four disks are active during this trace, with a separate process issuing requests to each disk. Note the high (99%) utilization of the SCSI string. The column labeled "Normalized" gives the average time per I/O spent in each phase of the protocol. Average bus free time per transaction is not included in the table, since this number is obscured by the overlap of operations when four disks are active on the string at once. The percentage of time that the bus is actually free in the trace represents the amount of time overall that there is no such overlap.*

the data transfer phase. This is not surprising, since there are no seeks (except between adjacent cylinders) during transactions. The SCSI bus is free only 1% of the time in this trace.

For 32 KByte sequential reads issued by the no-copy system call to four disks, the server utilization measurement (described in Section 2) is 7.69%, indicating that the host memory system and CPU are not a performance limitation. This is not surprising, since operations issued from the no-copy system call avoid copy and cache flush operations that cause host memory system contention.

3.1.2 Sequential Reads Issued from Sprite User Level

This section discusses string performance of sequential I/O operations issued as raw disk requests from Sprite user level. (These I/Os do not go through Sprite's file cache.) The previous section, describing no-copy system call I/O, represents the best performance that can be achieved on a string. There is a drop in performance when executing a "real" operating system. Sprite must return data to the user process that requested it. To do so, Sprite performs a number of copy and cache flush operations that cause contention on the Sun4/280 host's memory system.

The modified Sprite kernel traced the following events for Sprite raw device I/O:

- Start System Call: This interval is the time between the entrance to Sprite system call code and

Start System Call	290 us
Device Driver Time	150 us
Jaguar Driver Time	250 us
Time on Jaguar Board and Wren IV	22800 us
Start Jaguar Interrupt Handling	21 us
DMA Free Time	2700 us
Processing before Copy	420 us
Kernel to User Copy	4230 us
Finish Processing	170 us
Total Time per Call	31000 us
Time Between Subsequent Calls	190 us

Table 3: *Trace of 32 KByte sequential read operations issued from Sprite user level. All Sprite processing is included in this trace. Most of the time is spent on the Jaguar board and in the disk. Copy and cache flush times were also significant.*

the procedure call into device driver code.

- Device Driver Time: Time spent in the "generic" Sprite device driver.
- Jaguar Driver Time: Additional time spent in the device driver specific to the Interphase Jaguar HBA.
- Time on Jaguar HBA Board and Wren IV Disk: Records the interval from when the SCSI command is submitted to the Jaguar board until the first instruction of the Jaguar interrupt handler is executed. This includes setup time on the Jaguar board, SCSI protocol implementation on the Jaguar and the Wren, seek time, rotation time, and transfer time on the disk.
- Start Jaguar Interrupt Handling: Time in the interrupt handler up to the DMA flush.
- DMA Flush: Time spent flushing from the cache the DMA buffer used in the data transfer with the HBA.
- Processing before Copy: Processing time between DMA flush and copy operation.
- Copy Time (Kernel to User for reads): Time to perform the copy of DMA'd data from the kernel buffer to the user space that requested the data.
- Finish Processing: Time to complete system call execution.
- Time between Subsequent Commands: Time between completion of one system call and initiation of the next.

Table 3 shows a trace of operating system activity for 32 KByte sequential reads issued as raw device I/O operations by a single process to a single disk. The time spent on the Jaguar board and the Wren IV disk is approximately the same as that measured for the no-copy system call: essentially the time required to move 32 KBytes of data off the disk head at 1.3 MBytes/sec. Time on the Jaguar board is the largest interval traced.

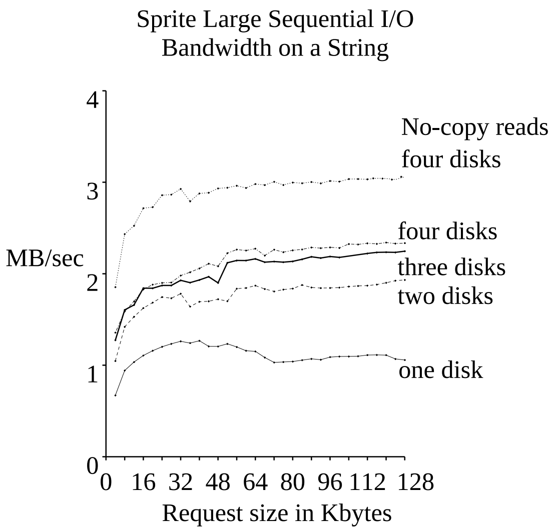

Figure 4: *Bandwidth for sequential read operations issued from Sprite user level to four disks on a single string. A separate process issues requests to each active disk. The top line in the graph is the bandwidth for no-copy system call reads issued to four disks, included for comparison. Issuing operations from Sprite causes a significant performance degradation.*

Two other traced intervals take significant processing time: the flushing of the cache memory used for the DMA operation between the host and a Jaguar HBA, and the copy between kernel and user space. They make up a significant portion (8.7% for the cache flush and 13.6% for the copy) of the average time for a 32 KByte read in the trace, supporting our assertion that they are responsible for contention on the host memory system. The cache flushing rate is 12 MBytes/sec and the data copy rate is 7 MBytes/sec.

Other intervals in the Sprite trace are small by comparison. Processing for operations other than the copy, cache flush, and time spent on the HBA and disk accounts for less than 5% of the lifetime of an operation in Table 3.

Figure 4 shows a graph similar to that of Figure 3. It depicts Sprite raw disk sequential read activity for up to four disks on a single SCSI string. The performance of a single disk on the string is somewhat lower than the corresponding performance for I/O issued by the no-copy system call in Figure 3 (1.2 MBytes/sec vs. 1.3 MBytes/sec). Additional disks on a string provide perform much worse than they did for I/Os issued by the no-copy system call. The maximum bandwidth seen on four disks is around 2.3 MBytes/sec, about 75% of the 3 MBytes/sec bandwidth achieved for no-copy sequential reads, shown in the top line of Figure 4 for comparison.

The overall throughput limit of 2.3 MBytes/sec turns out to be the maximum bandwidth that can be achieved by the array under Sprite on the Sun4/280. Host utilization, which measures 75.6% for 32 KByte operations, is the reason for this limitation. It indicates that the host memory bus is reaching saturation due to the copy operations and cache flushes performed by Sprite. When I/Os were issued by the no-copy system call, the comparable host utilization was 7.69%.

The breakdown of time spent in each SCSI phase is similar to that shown in Table 2.

3.2 Random Reads on a String

This section contains performance measurements of small random read operations, and shows that the array performs quite well for this workload. Sections 3.2.1 and 3.2.2 examine small, random I/O issued by the no-copy system call and as raw I/O from Sprite user level, respectively.

3.2.1 Random Reads Issued by No-Copy System Call

Table 4 shows the operating system trace of 4 KByte random read operations issued by the no-copy system call. The time spent on the Jaguar Board and the Wren IV disk is 33.2 msec. This is approximately equal to the expected phases of execution on the controller and disk: setup time on Jaguar controller (very brief) + an average seek (17.5 msec) + an average rotation (8.33 msec) + SCSI overhead (4 msec) + transfer time off the disk head (3 msec) = 33 msec. This "minimum" time to do a small random I/O operation limits the I/O rates on the Wren IV disks to about 30 per second.

Figure 5 is a graph of I/O rates for small random read operations issued using the no-copy system. The first three disks activated on a string deliver 30 I/Os/sec each for 4 KByte random operations, as expected. When four disks are active on a string, there is a slight performance degradation, with each disk consistently achieving only 29 I/Os/sec. This slight decrease in performance is not the result of SCSI utilization (only 33% in this trace) or host utilization (only 7%); we conclude that the loss is caused by processing on the Jaguar HBA.

Time to submit command	120 us
Time on HBA and Wren IV disk	33200 us
Interrupt processing	52 us

Table 4: *Trace of 4 KByte random read operations issued from the no-copy system call. Most of the time is spent on the HBA and disk, performing a seek and rotation and transferring the data.*

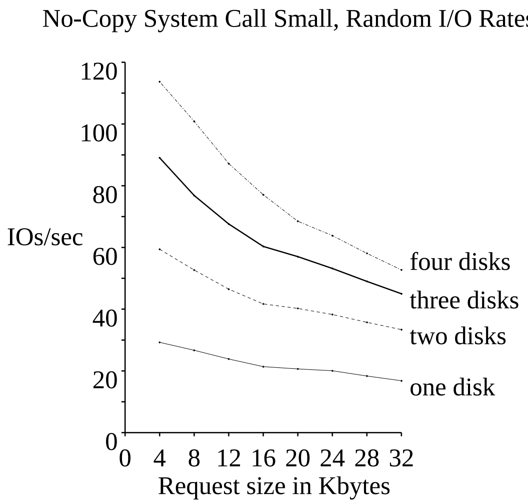

Figure 5: *I/O Rates for random operations generated using the no-copy system call. A separate process issues requests to each active disk.*

Time per SCSI Phase	Percentage (%)	Normalized (usec)
Arbitration	0.43	33
Selection	0.55	42
Command	6.21	480
Message Out	4.29	330
Message In	0.32	25
Data Transfer	20.0	1500
Disconnect/Reconnect	0.38	29
Reselection	0.07	5.4
Status	0.68	52
Bus Free	67.1	

Table 5: *Breakdown of time in SCSI phases for 4KByte random reads issued from the no-copy system call. Four disks are active, with a separate process issuing requests to each disk.*

Table 5 shows the SCSI phase trace for 4 KByte random read requests issued by a separate process to each of four disks on a single string. Compared to SCSI traces of sequential activity, the time spent in the data transfer phase is much lower, and the time that the SCSI bus is free is much greater. Since the transactions in the trace are small (4 KBytes), a short data transfer phase is expected. The decrease in SCSI string utilization is also no surprise. These small operations are dominated by average seek and rotation times (26 msec) plus the time to move data off the disk head into the disk buffer (3.3 msec for 4 KBytes), which dwarf data transfer time on the SCSI bus (less than 1 msec for 4 KByte transfers) and protocol overheads (a few msec). Even with an operation pending on each of the four disks, the SCSI bus will be idle most of the time, because the disks spend most of their time performing seeks and rotations. The other phases of the SCSI protocol still represent a small fraction of the lifetime of a transaction.

No bottleneck is evident for small transactions issued to four disks on a string. The performance increase is nearly linear with the number of disks. SCSI and host utilization are low.

3.2.2 Random Reads Issued from Sprite User Level

Table 6 shows a trace of 4 KByte random read operations issued as raw device I/Os from Sprite user level. Again, about 33 msec is spent on the controller and disk, as expected. The copy and cache flush times are proportional to the size of the object being manipulated, and hence for 4 KByte operations are quite short. Thus, host contention is much lower here (21.3%) than for large sequential operations (75.6%). However, this utilization is significantly higher than that for random operations issued from the no-copy system call, 7.0%. This relatively high utilization for just four active disks suggests that when many disks are active, the host CPU/memory system will become a performance limitation. This is confirmed in Section 3.4.

Figure 6 shows the I/O rates achieved for up to four disks performing small random reads issued from Sprite user level. The top line in Figure 6 shows no-copy read I/O rates for comparison. The SCSI phase breakdown is almost identical to that in Table 5. Compared to Figure 5, Figure 6 shows a modest performance penalty for small random operations, approximately 2 or 3 I/Os per second per active disk (10% per disk). This small performance degradation under Sprite is due to copy and cache flush operations and context switching overhead.

3.3 HBA Performance

This section examines the performance of a Jaguar Host Bus Adaptor, which sustained less bandwidth from its two strings than expected. The lower three lines in Figure 7 show the bandwidth limitation for disks on a single SCSI string, described in Section 3.1.1. When

Start System Call	140 us
Device Driver Time	140 us
Jaguar Driver Time	160 us
Time on Jaguar Board	33600 us
Start Jaguar Interrupt Handling	28 us
DMA Free Time	480 us
Processing before Copy	300 us
Kernel to User Copy	500 us
Finish Processing	85 us
Total Time per Call	35400 us
Time Between Subsequent Calls	370 us

Table 6: *Trace of 4 KByte random read operations issued from Sprite user level. Most of the time is spent on the HBA and disk, performing a seek and rotation and transferring the data.*

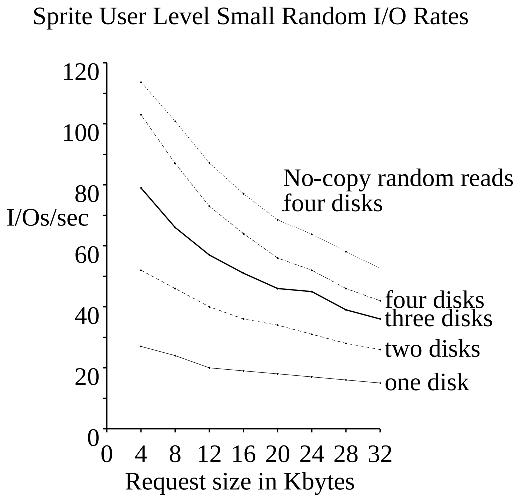

Figure 6: *I/O Rates for small random reads issued from Sprite user level. The top line in the graph shows the I/O rates for small random reads issued from the no-copy system call, included for comparison. Operations issued under Sprite achieve approximately 10% fewer I/Os per disk.*

one of the three disks is moved to the second string of a single HBA, some of the SCSI string contention is relieved, and performance improves. However, the HBA is unable to support the full bandwidth of three disks (3.9 MBytes/sec) even when the disks are spread across two strings. By contrast, the top line in the graph shows that the full bandwidth of three disks is achieved by three HBAs controlling one disk each. These last measurements confirm that the HBAs rather than any other system components limit the throughput. Interphase confirmed that a single Jaguar HBA can support a maximum of 4 MBytes/sec total from its two strings.

3.4 Overall Performance

This section presents overall bandwidth and I/O rate measurements for I/Os issued from the no-copy system call and from Sprite user level. It exposes performance bottlenecks in the VME backplane, the host CPU and the host memory system.

3.4.1 Overall Sequential Performance

Figure 8 shows the performance of the prototype for sequential reads of size 32 KBytes when up to 13 disks are active in the array. Processes that generated I/O activity on particular disks were activated in a round-robin fashion on the strings, to avoid as much string contention as possible. The tests for user level I/O used 11 disks on three strings, while the tests for the no-copy system call I/O used 13 disks on four strings.

The lower line in the graph shows the bandwidth achieved for I/O operations issued from Sprite user

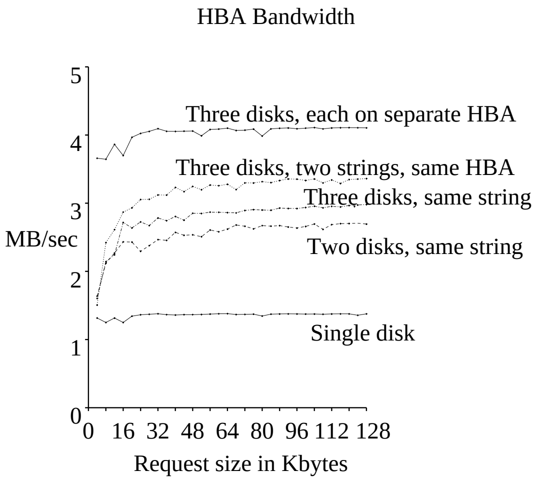

Figure 7: *Bandwidth for sequential reads generated with the no-copy system call for one, two and three disks arranged in various ways on the two strings of a single HBA, and three disks on three HBAs. In each case, a separate process issued requests to each active disk. A single HBA is unable to support the full bandwidth of three disks.*

level. The bandwidth is limited to 2.3 MBytes/sec regardless of the number of active disks. The large number of copy and cache flush operations performed under Sprite causes contention in the Sun4/280's memory system, limiting throughput. For example, eleven disks performing 32 KByte I/Os issued from Sprite user level result in host (CPU + memory) utilization of 97.3%, indicating saturation of the memory system.

The upper line in the graph shows throughput for I/O generated using the no-copy system call. The first few disks deliver their maximum potential bandwidth, but then throughput increases begin to level off. For thirteen active disks, bandwidth reaches about 7.5 MBytes/sec. Performance is limited by saturation of the VME backplane. This bandwidth is surprisingly low; we had expected the VME backplane to sustain 10-12 MBytes/sec. However, completion of operations in strict order of their priorities on the backplane and timeouts for low-priority operations indicate that the VME saturated sooner than expected. Memory system contention is not a limiting factor to performance; host utilization measures only 23.8%.

3.4.2 Overall Small Random I/O Rates

Figure 9 shows I/O rates achieved on 14 disks on four strings, each string on a different HBA, performing small (4 KByte) random reads, where the I/O operations were issued from Sprite user level and from the no-copy system call.

The line for operations issued from the no-copy system call is close to linear, increasing approximately 30 I/Os per second per disk to 420 I/Os per second for 14

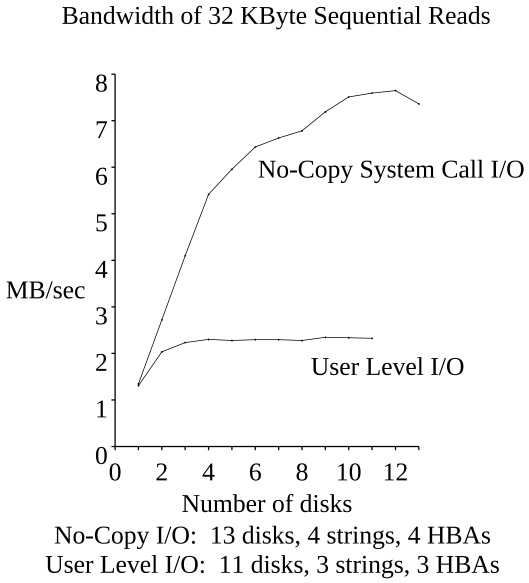

Figure 8: *Bandwidth for 32 KByte Sequential Read operations for up to 13 disks over four strings (each string on a separate HBA) in the prototype. The top line shows performance for I/Os generated from the no-copy system call, and the bottom line for those issued from Sprite user level.*

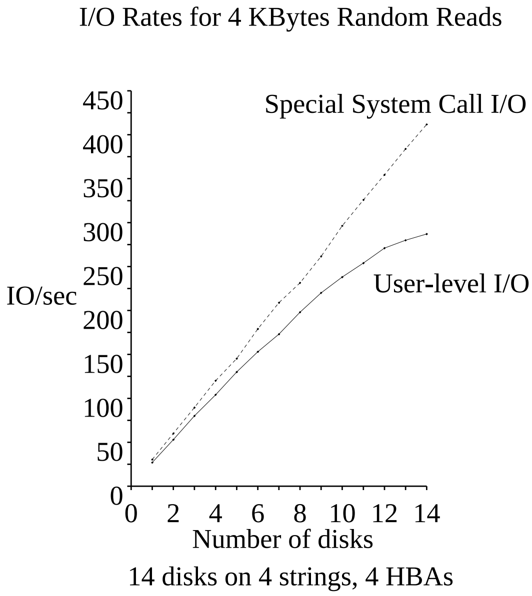

Figure 9: *I/O Rates for 4 KByte random reads performed on 14 disks on four strings (each string on a separate HBA) in the prototype. The top line in the graph shows I/O generated by the no-copy system call; I/O rates achieved increase linearly with the number of disks to 420 I/Os per second. The lower line of the graph shows I/O issued from Sprite User level; these rates are limited to about 275 I/Os per second by memory system contention and context switching overhead.*

disks. The maximum I/O rate of each disk is achieved. The measured host CPU/memory bus utilization in the 14-disk case is 40%, low enough that it doesn't affect the I/O rate achieved by the array.

However, when the requests are issued from Sprite user level, the I/O rates delivered by 14 disks are significantly lower. The first disk on each string contributes about 25 I/Os per second. The I/O rates achieved per disk decrease as more disks are added to the strings. Fourteen disks achieve approximately 275 I/Os per second, close to 20 I/Os per second per disk.

The host CPU/memory bus utilization measured for 14 active disks is 78.4%. This high utilization is not solely the result of memory system contention due to copy and DMA cache flush operations. The last section showed that Sprite could sustain 2.3 MBytes/sec of such activity, and the 14 disks performing 275 I/Os per second generate only 1.1 MBytes/sec of bandwidth. High host utilization is also caused by the host CPU, which is required to perform 275 context switches per second. (A context switch takes about 1 msec in Sprite.)

Although we encounter this host utilization limitation, we consider the performance achieved by the prototype on small random operations (275 I/Os per second) to be excellent. The prototype and Sprite deliver good performance for the small operations typical of current operating systems and databases. Currently, the prototype's performance as a file server is constrained by its Ethernet connection rather than by the number of I/Os/sec it can deliver.

4 Conclusions

This report has described a set of experiments and tools used to evaluate our prototype. These include SCSI traces, operating system kernel traces and programs generating artificial workloads. Table 7 summarizes the performance of each system component compared to its ideal or expected performance.

At the lowest level, performance of the array is limited to the capabilities of the disks. Ignoring Sprite, the Wren IV disks deliver 30 I/Os per second for small, random operations and 1.3 MBytes/sec for large, sequential operations.

Each SCSI string in the prototype has four disks attached. We expected large, sequential transfer rates of close to 4 MBytes/sec on each string, but SCSI protocol overhead limited throughput to 3 MBytes/sec per string. Despite the bandwidth limitation on a SCSI string, we are still happy with our choice of SCSI components. SCSI is a widely available industry standard with readily available small format disks. SCSI disks and controllers are very inexpensive because their price is driven by the PC market. In addition, SCSI implementations are highly integrated and intelligent. We expect faster processors on disks and controllers to alleviate many of the performance penalties in future SCSI implementations.

The Interphase Jaguar HBA sustains only 4 MBytes/sec from the two SCSI strings it controls, rather than their full bandwidth. This HBA was designed for a typical file server workload, in which an HBA commonly supports two or three disks performing small I/Os. It is not surprising that trying to achieve high bandwidth for large sequential operations from both strings stressed the HBA. In retrospect, it was a mistake to use both strings of the HBA.

The VME backplane limits bandwidth on large, sequential transfers to about 7.5 MBytes/sec. This throughput was significantly lower than the expected limitation of 10-12 MBytes/sec.

The host CPU and memory system combine to form the most serious performance limitations. Overall small random I/O rates are limited to 275 I/Os per second by memory system contention and context switching overhead under Sprite. Faster CPUs will alleviate some of the context switching penalties. Memory system contention limits throughput on the prototype to just 2.3 MBytes/sec for large sequential operations performed under Sprite, less than could be provided by just two Wren IV disks. The contention is due to the large number of copy and cache flush operations incurred when running the Sprite operating system. The Sprite group is examining ways to perform I/O directly to user space to avoid unnecessary copy operations. The memory system traffic required by Sprite is exacerbated by the Sun4/280's use of the virtual cache for DMA operations, requiring a cache flush. Newer Sun products avoid this.

Despite the host CPU and memory system limitations, the prototype's performance for small, random I/Os is excellent. A typical server performs 50 to 100 I/Os per second; the prototype provides nearly 300, which will likely saturate an Ethernet. However, the prototype fails to provide adequate performance for workloads requiring high bandwidth. Because off-the-shelf components do not provide the interconnection and memory system with sufficient bandwidth, the RAID group is designing a second prototype. In it, a high bandwidth crossbar will connect a high-speed network, disk interfaces and memory.

5 Acknowledgements

We would like to thank Richard P. Drewes, Ken Lutz, Peter M. Chen, Garth A. Gibson, Edward K. Lee and all the members of the RAID group for their help in this work. We would also like to thank Mendel Rosenblum, John K. Ousterhout and the members of the Sprite group.

We are grateful for the support of our government and industrial sponsors, including Array Technologies, DARPA/NASA (NAG2-591), DEC, Eastman Kodak,

Component	Expected	No-Copy	Bottleneck	Sprite	Bottleneck
Single disk I/O Rates	30 I/O/s	30	Disk	27	Disks + Sprite
Single disk Bandwidth	1.3 MB/s	1.3	Disk	1.3	Disk
String I/O (4 disks)	120 I/O/s	115	Disks/HBA	103	Disks + Sprite
String BW (4 disks)	4 MB/s	3	SCSI overhead	2.3	Host memory
HBA BW (2 strings, 8 disks)	6 MB/s	4	HBA Bandwidth	2.3	Host memory
Overall I/O (14 disks)	420 I/O/s	420	Disks	275	Host CPU, memory
Overall BW (13 disks)	10-12 MB/s	7.5	VME Backplane	2.3	Host memory

Table 7: *Expected versus Actual performance of the prototype for operations issued from the No-Copy special system call and from user-level Sprite raw device operations. In each case, the bottleneck that limited performance for each workload is indicated. I/O rates were measured for 4 KByte random read operations. Bandwidth was measured for 32 KByte sequential read operations. Bandwidth for the "Best Ever" case was for 128 KByte sequential reads. Note that the "expected" 10-12 MByte/sec bandwidth for the array is the expected limit of the VME backplane.*

Hewlett Packard, IBM, Intel Scientific Computers, California MICRO, NSF (MIP 8715235), Seagate (formerly Imprimis), Sun Microsystems and Thinking Machines Corporation. Thanks also go to Interphase Corporation for their technical support. Ann Chervenak was supported in part by a National Science Foundation Fellowship.

References

[Ancot] *SCSI Bus Analyzer/Emulator Model DCS-202 User's Manual*, ANCOT Corporation, Redwood City, CA.

[Chen90a] Peter M. Chen, Garth A. Gibson, Randy H. Katz, David A. Patterson, "An Evaluation of Redundant Arrays of Disks Using an Amdahl 5890", *Proc. of the 1990 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, Boulder, CO, May, 1990.

[Chen90b] Peter M. Chen, David A. Patterson, "Maximizing Performance in a Striped Disk Array", *Proc. of the 1990 ACM SIGARCH 17th Ann. Int. Symp. of Computer Architecture*, Seattle, WA, May, 1990.

[Cher90] Ann L. Chervenak, "Performance Measurements of the First RAID Prototype", U.C. Berkeley Technical Report UCB/CSD 90/574, May, 1990.

[Gibs89] Garth A. Gibson, "Performance and Reliability in Redundant Arrays of Inexpensive Disks", *Proc. of the 1989 Computer Measurement Group (CMG) Annual Conference*, Reno, Nevada, December, 1989.

[Gibs91] Garth A. Gibson, "Redundant Disk Arrays: Reliable, Parallel Secondary Storage", U.C. Berkeley Technical Report UCB/CSD 90/613, Berkeley, CA, March, 1991, Ph.D. Dissertation.

[Gray90] Jim Gray, Bob Host, Mark Walker, "Parity Striping of Disc Arrays: Low-cost Reliable Storage with Acceptable Throughput", *Proc. 16th Int. Conf. on Very Large Data Bases (VLDB)*, Morgan Kaufmann, August, 1990.

[Jaguar] *V/SCSI 4210 Jaguar High Performance VMEbus Dual SCSI Host Adaptor–User's Guide*,Interphase Corporation, Dallas, TX.

[Kim86] Michelle Y. Kim, "Synchronized Disk Interleaving", *IEEE Trans. on Computers*, C-35, 11, November, 1986.

[Lee90] Edward K. Lee, "Software and Implementation Issues in the Implementation of a RAID Prototype", U.C. Berkeley Technical Report UCB/CSD 90/573, May, 1990.

[Lee91] Edward K. Lee, Randy H. Katz, "Performance Consequences of Parity Placement in Disk Arrays, to appear *Fourth Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS IV)*, Palo Alto, CA, April, 1991.

[Livn87] M. Livny, S. Khoshafian, H. Boral, "Multi-disk Management Algorithms", *Proc. of ACM SIGMETRICS*, May, 1987.

[Oust88] John K. Ousterhout, Andrew R. Cherenson, Frederick Douglis, Michael N. Nelson, Brent B. Welch, "The Sprite Network Operating System",*Computer*, February, 1988.

[Oust90] John K. Ousterhout, "Why Aren't Operating Systems Getting Faster As Fast as Hardware?", *Proc. of the Summer '90 USENIX Technical Conference*, Anaheim, CA, June, 1990.

[Patt88] David A. Patterson, Garth A. Gibson, Randy H. Katz,, "A Case for Redundant Arrays of Inexpensive Disks", *Proceedings of the 1988 ACM SIGMOD Conference on Management of Data*, Chicago, IL, June, 1988.

[Redd89] A. L. Narasimha Reddy, Prithviraj Banerjee, "Evaluation of Multiple-Disk I/O Systems," *IEEE Trans. on Computers*, C-38, 12, December, 1989.

[Rose90] Mendel Rosenblum, "The LFS Storage Manager", *Proc. of the Summer '90 USENIX Technical Conference*, Anaheim, CA, June, 1990.

[Sale86] K. Salem, H. Garcia-Molina, "Disk Striping", *IEEE 1986 Int. Conf. on Data Engineering*, 1986.

[SCSI] *SCSI Guidebook*, Adaptive Data Systems, Inc., Pomona, CA, 1985.

[Wren] *Product Specification for Wren IV SCSI Model 94171-344*, Control Data Corporation, Minneapolis, MN.