Modeling and optimizing I/O throughput of multiple disks on a bus (summary)

*Rakesh Barve** *Elizabeth Shriver*† *Phillip B. Gibbons*† *Bruce K. Hillyer*† *Yossi Matias*‡
*Jeffrey Scott Vitter**

Abstract For a wide variety of computational tasks, disk I/O continues to be a serious obstacle to high performance. The focus of the present paper is on systems that use multiple disks per SCSI bus. We measured the performance of concurrent random I/Os, and observed bus-related phenomena that impair performance. We describe these phenomena, and present a new I/O performance model that accurately predicts the average bandwidth achieved by a heavy workload of random reads from disks on a SCSI bus. This model, although relatively simple, predicts performance on several platforms to within 12% for I/O sizes in the range 16–128 KB. We describe a technique to improve the I/O bandwidth by 10–20% for random-access workloads that have large I/Os and high concurrency. This technique increases the percentage of disk head positioning time that is overlapped with data transfers, and increases the percentage of transfers that occur at bus bandwidth, rather than at disk-head bandwidth.

Introduction We study the performance of workloads consisting of read requests directed to disks that share a SCSI bus. These workloads are relevant to certain multimedia, database, and scientific computing applications that use external memory and out-of-core algorithmic techniques (e.g., [6, 1]). For our experiments, the requests are generated by one process per disk. Each process iterates the following steps: (1) generate a random block address, (2) record a timestamp, (3) issue a seek and a read to the raw disk, (4) record a timestamp when the read request completes, (5) return immediately to step 1. We measured requests ranging from 16 KB to 128 KB, on four hardware configurations. (1) 1–4 Seagate Cheetah disks, Sun Ultra-1, Solaris 2.5.1. (2) 1–4 Seagate Cheetah disks, Sun Sparc-20, Solaris 2.5. (3) 1–4 Seagate Barracuda disks, DEC AlphaStation 600 5/266, Digital UNIX 4.0. (4) 1–7 Seagate Wren-7 disks, Sun Sparc 20, Solaris 2.5.

A number of performance models exist for disks. The analytic disk model of [4] captures bus effects only in the single-disk case. The Pantheon disk simulator [3] incorporates bus contention and other bus effects, but no results have been published that describe the idle periods and head-limited bus transfers that we observe. The method for approximating the throughput of multiple disks on a SCSI bus in [2] sums the seek time, rotational latency, and transfer time, and derates the performance by a contention factor derived from a general queuing model.

*Duke University, visiting Bell Labs.

†Information Sciences Research Center at Bell Labs, contact shriver@bell-labs.com.

‡Tel-Aviv University. Work performed at Bell Labs.

Rounds In our experiments, we typically observe that all disks receive a request, then all disks transmit data back to the host before any disk receives another request. We use the term *rounds* for this periodic convoy behavior. We observed rounds for workloads of random accesses as well as when the accesses on each disk have spatial locality. We were surprised to see rounds. Since the host has the highest SCSI priority, one would expect that soon after a disk completes one request, the host would seize the bus to send another request to that disk, thereby keeping the bus and all the disks busy. Rounds could arise if the operating system kernel implements a fairness policy that forcibly balances the number of requests sent to each disk during periods of heavy I/O, rather than sending requests to disks as soon as possible. The current literature does not discuss rounds as we observed them. In fact, [5] states that even in cases when the load is symmetrically distributed and balanced, one process can monopolize the disks while others starve.

Previous models (e.g., [2]) gave prediction errors greater than 100% for our workloads. We develop a model below that gains accuracy from three factors: It accounts for idle times and overlaps caused by rounds, it distinguishes between the transfer rate from a disk platter through the disk head and the (much higher) rate from a disk's cache to the host, and it considers the effect of a relatively obscure disk control parameter called the *fence* or *buffer full ratio*. The fence determines the time at which the disk will begin to contend for the SCSI bus. A minimum fence (i.e., 0) causes the disk to contend after reading the first sector of data into the disk's internal cache. A maximum fence (i.e., 255) causes the disk to wait until almost all of the requested data has accumulated in the disk cache. When the bus is idle, a low fence value starts sending data sooner, but more of the transfer occurs at the head-limited bandwidth.

Model We developed a model that predicts the read response time for a single disk and for multiple disks on a bus. Because of space limitations, we present only the more interesting multiple disk equations here. In each round, one request for B bytes is served from each of D disks. When the fence is 0, the idle time at the beginning of the round is the expected minimum positioning time over all disks, denoted $\mathsf{MPT}(D)$, plus the overhead time for a SCSI command, denoted Os. The first disk to respond transmits one sector at bandwidth BW_{bus}, and continues at BW_{rot} for an amount of data denoted $\mathsf{L}(B)$. Then the disk disconnects, either because it has transferred the entire request, or because it has reached a track or cylinder boundary. When the request size B is small, the entire request usually resides on a single track, whereas for large requests the expected size of the leading portion is one half the track size. Thus if $B \leq \mathsf{AverageTrackSize}/2$, we approximate $\mathsf{L}(B) = B$, otherwise $\mathsf{L}(B) = \mathsf{AverageTrackSize}/2$. When the first disk disconnects, other disks generally have enough data in their caches that all the remaining data can be sent at BW_{bus}—whenever one disk disconnects, another immediately seizes the bus.

We make two simplifications. Although the first disk sends the first sector at BW_{bus}, we say that the entire leading portion from the first disk is sent at BW_{rot}. Second, the overhead of the disconnect/reconnect is absorbed into the overhead term. Thus the average time per round is given by

$$\mathsf{Os} + \mathsf{MPT}(D) + \frac{\mathsf{L}(B)}{\mathsf{BW}_{rot}} + \frac{DB - \mathsf{L}(B)}{\mathsf{BW}_{bus}}.$$

If the fence is not zero, the bus is idle during the shortest positioning time, then the bus continues to remain idle while that disk reads $B_c = B \cdot (\mathsf{FenceValue}/256)$ bytes of the B bytes into its cache. Next the bus transmits all the data to the host. Thus the average time per round is given by

$$\mathsf{Os} + \mathsf{MPT}(D) + \frac{B_c}{\mathsf{BW}_{rot}} + \frac{DB}{\mathsf{BW}_{bus}}.$$

To extend the model to multiple request sizes, let $\overline{B}$ be the mean request size, and compute the leading portion $\mathsf{L}(\overline{B})$ as a weighted average of the leading portions for each request size. To model a workload that confines requests to a subset of the cylinders, adjust the expected seek time in the obvious way. $\mathsf{MPT}(D)$ is straightforward to estimate experimentally, but we recently obtained an analytic expression.

Validation For uniform random requests of fixed size from 16–128 KB on Wren disks, the *maximum* relative error that we measured is 5.2% for the single disk model and 8.5% for the multiple disks model. On Barracuda disks, the corresponding figures are 8.7% and 10.8%. On Cheetah disks, the figures are 7.0% and 9.2%. The Cheetah model error increases to 14% with 8 KB blocks: the Cheetahs are so fast that the transfer of the first small block sometimes completes before any other disks are ready to use the bus, so the model underestimates the amount of bus idle time.

Pipelining The equations show two opportunities to improve performance: decrease the minimum positioning time, and convert slow BW_{rot} transfers to the faster BW_{bus}. Assuming that during round $j-1$ we know the blocks that will be requested during round j, we propose a pipelining technique to overlap the positioning time for one round with the transfer time of the previous round. Furthermore, this technique stages data in the on-disk caches, so that all transfers are at BW_{bus}. Let $b_{i,j}$ denote the block to be retrieved from disk i in round j. Then the pipelining technique is:

for $0 \leq i \leq D-1$
    Request LoadIntoDiskBuffer($b_{i,0}$) on disk i.
for $0 \leq j \leq$ NumRequests
    for $0 \leq i \leq D-1$
        Read($b_{i,j}$) from disk i. // $b_{i,j}$ *is in buffer*
        Request LoadIntoDiskBuffer($b_{i,j+1}$) on disk i.

The SCSI `Prefetch` command would implement LoadIntoDiskBuffer, but it is an optional command, not supported by the Wren, Barracuda, or Cheetah. We implement LoadIntoDiskBuffer(b) by an `aioread()` of the disk sector just before b, which triggers the disk readahead mechanism to load b into the disk cache.

We evaluate this technique using I/Os ranging from 8–128 KB. For small block sizes, the overhead of the `aioread()` harms performance by several percent. For too many disks (e.g., 7 Wrens on the Sparc-20), the SCSI bus is so saturated that there is little room for improvement. With 2 or 4 disks and moderate or large block sizes, the overlaps gained by pipelining more than compensate for the increased overhead, improving the sustained bandwidth by 10-20%. We have similar results for the DEC Alpha with Barracuda disks, and for Cheetahs on the Sun Ultra-1 (see Table 1). Pipelining does not pay on a Sun Sparc-20 with fast Cheetah disks, because of excessive overhead for an `aioread()` (2.1–2.3 ms; about the time to read 16–32 KB from disk cache to host).

Table 1: Percent improvement pipelining, Cheetah/Ultra-1.

(KB)	D=2	D=3	D=4
8	-8	-12	-13
16	-6	-9	-11
32	-4	-4	-3
64	0	7	16
96	8	18	17
128	12	20	11

Conclusions In measurements of I/O-intensive workloads on several UNIX systems with multiple disks per SCSI bus, we observe a convoy behavior that we call "rounds". We have developed a model that predicts the I/O bandwidth of such systems. Measurements of synthetic workloads (reported in the full paper) indicate that the model is accurate for single or multiple request sizes, under uniform random access or spatial locality. We presented a pipelining technique to coordinate the accesses *across* a collection of disks that share a SCSI bus. It can improve performance by 10–20% by increasing the overlap between disk seeks and data transfers, and by increasing the fraction of transfers that occur at the disk cache transfer rate rather than the slower disk-head rate. Experimental results show the performance regions where pipelining improves overall performance despite the associated overhead.

References

[1] Cormen, T. H., and Hirschl, M. Early experiences in evaluating the parallel disk model with the ViC* implementation. *Parallel Computing 23*, 4 (June 1997), 571–600.

[2] Hennessy, J. L., and Patterson, D. A. *Computer architecture: a quantitative approach.* Morgan Kaufmann Publishers, Incorporated, San Francisco, CA, 1996. 2nd edition.

[3] Ruemmler, C., and Wilkes, J. An introduction to disk drive modeling. *IEEE Computer 27*, 3 (March 1994), 17–28.

[4] Shriver, E. *Performance modeling for realistic storage devices.* PhD thesis, New York University, May 1997. Available at `http://www.bell-labs.com/~shriver/`.

[5] Sinclair, J. B., Tang, J., and Varman, P. J. Placement-related problems in shared disk I/O, In Jain, R., Werth, J., and Browne, J. C., (eds.), *Input/Output in Parallel and Distributed Computer Systems.* Kluwer Academic Publishers, 1996, ch. 12, pp. 271–289.

[6] Vitter, J. S., and Shriver, E. A. M. Algorithms for parallel memory I: two-level memories. *Algorithmica 12*, 2/3 (August and September 1994), 110–47.