Appeared in the Proceedings of the ACM Sigmetrics Conference, May, 1994, pp. 241-251.

Scheduling Algorithms for Modern Disk Drives

Bruce L. Worthington, Gregory R. Ganger, Yale N. Patt
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor 48109-2122
worthing@eecs.umich.edu

Abstract

Disk subsystem performance can be dramatically improved by dynamically ordering, or *scheduling*, pending requests. Via strongly validated simulation, we examine the impact of complex logical-to-physical mappings and large prefetching caches on scheduling effectiveness. Using both synthetic workloads and traces captured from six different user environments, we arrive at three main conclusions: (1) Incorporating complex mapping information into the scheduler provides only a marginal (less than 2%) decrease in response times for seek-reducing algorithms. (2) Algorithms which effectively utilize prefetching disk caches provide significant performance improvements for workloads with read sequentiality. The cyclical scan algorithm (C-LOOK), which always schedules requests in ascending logical order, achieves the highest performance among seek-reducing algorithms for such workloads. (3) Algorithms that reduce overall positioning delays produce the highest performance provided that they recognize and exploit a prefetching cache.

1 Introduction

Disk subsystems must be carefully managed to compensate for the growing performance disparity between processing and storage components. Disk workloads are often characterized by intense bursts of activity, creating long queues of pending requests [McNu86, Ruem93]. The disk scheduler is responsible for dynamically ordering the pending requests. A significant portion of request service time may be comprised of mechanical positioning delays, which are highly dependent on the relative positions of the various request blocks and the current disk arm position. By taking into account the various delays associated with disk accesses, a scheduler can minimize the total positioning overhead while providing reasonable response times for individual requests.

Over the past 25 years, a variety of scheduling algorithms have been proposed and implemented in both academia and industry. Their relative merit is dependent on the exact workload of disk requests, the scheduler's computational resources, and the available knowledge of disk drive configuration and current status. Most previous studies of scheduling algorithms used simplistic disk models, lacking several important features present in modern disk drives. On-board logic has expanded to allow the mapping of logical data blocks to physical media to be hidden by a high-level interface, making this information unavailable to any external scheduling entity. Small speed-matching buffers have been replaced with large prefetching caches. In this paper, we investigate how these features affect the performance of various scheduling algorithms.

In addition, previously published work has been limited to synthetic workloads. Most of these studies assume that request starting locations are uniformly distributed across the entire disk. While much can be learned using such *random* workloads, the validity of the results remains in question until verified using more "realistic" workloads. We use random workloads to allow comparison with previous work, but most of our experiments utilize extensive traces from various real-world systems. The traces and synthetic workloads are used to drive a very detailed, well-validated disk simulator.

Section 2 describes modern disk drives and how some features can affect scheduling accuracy. Section 3 discusses previous scheduling research. Section 4 describes our methodology, including our validation efforts and the origins of the six traces. Sections 5 and 6 present our results using random workloads and traces, respectively. Section 7 summarizes this work and suggests avenues for future research. Additional data and descriptions, excluded due to space limitations, can be found in [Wort94].

2 Modern Disk Drives

A disk drive contains a set of rapidly rotating platters (on a common axis) coated on both sides with magnetic media. Data are written in circular **tracks** on each **surface**. The minimum unit of data storage is usually a **sector**, which commonly holds 512 bytes of data plus header/trailer information (e.g., error correction data). A **cylinder** is a set of tracks (one from each surface) equidistant from the center of the disk. For example, the innermost cylinder contains

0

the innermost track of each surface. Therefore, a physical media location is defined by its cylinder, surface, and sector. The disk arm holds a set of read/write heads that can be positioned to access any cylinder. The heads are ganged together such that a given disk arm position corresponds to a specific cylinder. In most drives only one read/write head is active at any given time.

Requests serviced by the disk drive may suffer one or more mechanical delays. First, the disk arm must move (**seek**) to the target cylinder. Second, **rotational latency** is incurred while waiting for the target sectors to reach the read/write head. After these delays, the data transfer begins. Additional mechanical delays result if the desired data spans multiple tracks or cylinders.

Current disk drives have several features that can directly affect the accuracy of scheduling algorithms. Some are the result of of efforts to make disks self-contained or self-managed. Clever methods of increasing a disk drive's capacity and effective lifetime also have a major impact. One goal of our work is to determine exactly how these characteristics should factor into scheduling algorithms.

2.1 Host Interface

Most disks possess sufficient on-board logic and processing power to present a relatively clean interface to the host system. Such protocols as the Small Computer System Interface (SCSI) and the Intelligent Peripheral Interface (IPI) are used by a wide variety of disk drive, adapter, and host system manufacturers. The host or intermediate controller presents a request to the disk drive in terms of the starting **logical block number** (LBN) and total request size. The details of the subsequent media access are typically hidden from the host. This approach offloads most of the management overhead associated with the actual data storage from the host or controller to the disk drive electronics. On the other hand, scheduling entities outside of the drive itself often have little or no knowledge of the data layout on the media, the status of any on-board disk cache, and the various overhead delays associated with each request.

2.2 Data Layout

Many systems use LBN-based approximations of seek-reducing algorithms. LBN-based scheduling relies on the sequentiality of logical block to physical block (LBN-to-PBN) mappings. More aggressive algorithms may require highly accurate knowledge of the data layout. As mentioned above, such mappings may be unavailable or very difficult to obtain. The complexity of the mappings is increased by zoned recording, track/cylinder skew, and defect management [Ruem94, Wort94].

2.3 On-Board Cache

The use of memory within embedded disk drive control logic has progressed from small, speed-matching buffers to dynamically-controlled caches containing megabytes of storage. The disk logic can automatically prefetch data into the cache to more quickly satisfy sequential read requests. For example, a disk might take 20 ms to service a read request that requires media access. A subsequent sequential read might take only 2 ms if the data has been prefetched into the on-board disk cache.

The existence of an on-board cache affects scheduling activities in several ways. First, if the disk is configured to prefetch aggressively, the position of the read/write head cannot necessarily be determined by knowledge of the most recent request location and length. The disk will continue reading beyond the end of a previous read request, possibly switching read/write heads or seeking to the next cylinder. Second, cached data can be accessed far more quickly than data obtained from the disk media. Requests that can be satisfied by the cache could therefore be given higher priority.

3 Disk Scheduling Algorithms

Numerous studies have shown that a simple *First Come First Served* (FCFS) disk scheduling policy often results in suboptimal performance. Many scheduling algorithms have been proposed that achieve higher performance by taking into account information about individual requests and the current state of the disk subsystem.

3.1 Seek Delay Reduction

Over 25 years ago, [Denn67] analyzed the advantages of a *Shortest Seek Time First* (SSTF) policy. This algorithm chooses the next request to service by selecting the pending request that will incur the smallest seek delay. It is usually infeasible to predict exact seek times, but SSTF may be closely approximated by using seek distances. SSTF reduces the average response time over a wide range of workloads. However, the potential for starvation of individual requests increases with the disk utilization, resulting in excessive response time variance. Given a heavy workload, the arm tends to hover over a subset of the cylinders in an attempt to exhaust all requests to that region, thereby starving any requests outside of that space.

[Denn67] also examined the SCAN or "elevator" algorithm, which provides a lower response time variance than SSTF with only a marginal increase in the average response time (for the random workloads studied). This algorithm is named for the way the disk arm shuttles back and forth across the entire range of cylinders, servicing all requests in its path. It only changes direction at the innermost and outermost cylinder. Because the disk arm passes over the center region of the disk at more regular intervals than the edges, requests to middle cylinders receive better service. However, every cylinder is reached during both phases of the scan. As a result, SCAN resists starvation more effectively (i.e., has lower response time variance) than SSTF.

Several variations of the SCAN algorithm have been proposed. The *Cyclical SCAN* algorithm (C-SCAN) replaces the bidirectional scan with a single direction of arm travel [Seam66]. When the arm reaches the last cylinder, a full-stroke seek returns it to the first cylinder without servicing any requests along the way. C-SCAN treats each cylinder equally, rather than favoring the center cylinders. The LOOK algorithm, another SCAN variation, changes the scanning direction if no pending requests exist in the current direction of travel [Mert70]. C-SCAN and LOOK can be combined, resulting in the C-LOOK algorithm.

VSCAN(R), proposed in [Geis87], creates a continuum of algorithms between SSTF and LOOK. The R parameter denotes how strongly biased the scheduler is towards maintaining the current direction of travel. VSCAN(0.0) is equivalent to SSTF, and VSCAN(1.0) reduces to LOOK. [Geis87] suggests that VSCAN(0.2) provides a good balance between average response time and starvation resistance.

3.2 Positioning Delay Reduction

Reducing the average seek delay only requires knowledge about the relative seek distances between requested data. To account for rotational latency, more complete information about the actual mapping of data blocks onto the media is necessary. In addition, the current physical location of the active read/write head must be known. Given this information, the scheduler can choose the request with the minimum *positioning* delay (i.e., combined seek and rotational latency). This algorithm was denoted as *Shortest Time First* (STF) in [Selt90] and *Shortest Access Time First* (SATF) in [Jaco91], but we use the term *Shortest Positioning Time First* (SPTF) to clarify the exact focus of the algorithm.

SPTF, like SSTF, suffers from poor starvation resistance. To reduce response time variance, priority can be given to requests that have been in the pending queue for excessive periods of time. The priority may slowly increase as the request ages, or a time limit may be set after which requests are given a higher priority [Selt90, Jaco91].

HP C2247 Disk Drive	
Formatted Capacity	1.05 GB
RPM	5400
Data Surfaces	13
Cylinders	2051
Sectors	2054864
Zones	8
Sectors/Track	56-96
Interface	SCSI-2
256 KB Cache, 1-4 Segments	
Track Sparing/Reallocation	

Table 1: HP C2247 Disk Drive Basic Parameters

4 Methodology and Validation

4.1 Disk simulator

We have developed a detailed, strongly validated disk simulator to compare scheduling algorithms. The simulator accurately models zoned recording, spare regions, defect management, disk buffers and caches, various prefetch algorithms, bus delays, and control and communication overheads. For this paper, the simulator was configured to model the HP C2240 series of disk drives [HP92]. Table 1 lists some basic specifications for the HP C2247. To accurately model this drive, an extensive set of parameters was obtained from published documentation and by monitoring the SCSI activity generated by this particular disk. We extracted a seek curve, control and communication overheads, bus transfer rates, and cache management strategies. We also determined the exact LBN-to-PBN mappings for several of these disks, which provided information about zoning, sparing, and existing defects. The validated simulator configuration, including values for all relevant parameters, is described in [Wort94].

The simulator was validated by exercising an HP C2247 and capturing traces of all SCSI activity. Each traced request stream was run through the simulator, using the observed request interarrival delays. This process was repeated for several workloads with varying read/write ratios, interarrival times, request sizes, and degrees of sequentiality and locality. The average response times of the actual disk and the simulator match to within 0.8% in all cases. Unpredictable host delays partially account for this difference. Greater insight can be achieved by comparing the response time distributions [Ruem94]. Figure 1 shows measured and simulated response time distributions for a sample validation workload. As with most of our validation results, one can barely see that two curves are present. [Ruem94] defines the root mean square horizontal distance between the two distributions as a demerit figure for disk model calibration. The demerit figure for the validation run shown in figure 1 is 0.07 ms, or

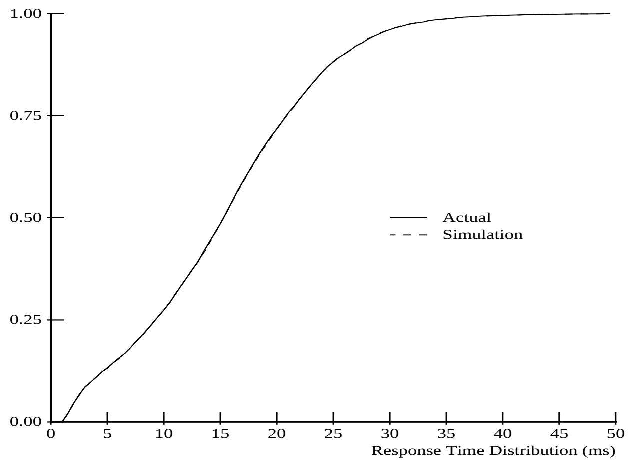


Figure 1: Validation Workload Response Time Distributions: 10K requests, 30% sequential, 30% local, 50% read, exponential request size with 8KB mean, uniform interarrival of 0-22 ms

less than 0.5% of the average response time. The worst-case demerit figure observed over all validation runs was 1.9% of the corresponding average response time.

4.2 Workloads

We use synthetically generated random workloads and extensive traces captured from actual systems. We describe the traces only briefly as they have been described elsewhere in more detail [Ruem93, Rama92]. The traced workloads span a broad range of environments, and each is at least a full workshift (8 hours) in length.

Two of the traces come from Hewlett-Packard systems running HP-UX^{TM}, a version of the UNIX^{TM} operating system [Ruem93]. *Cello* comes from a server at HP Labs used for program development, simulation, mail, and news. *Snake* is from a file server at the University of California at Berkeley used primarily for compilation and editing. While these traces are actually two months in length, we report data for a single week-long snapshot (5/30/92 to 6/6/92).

The other four traces are from commercial VAX^{TM} systems running the VMS^{TM} operating system [Rama92]. *Air-Rsv* is from a transaction processing environment in which about 500 travel agents made airline and hotel reservations. *Sci-TS* is from a scientific time-sharing environment in which analytic modeling software and graphical and statistical packages were used. *Order* and *Report* are from a machine parts distribution company. *Order* was collected during daytime hours, representing an order entry and processing workload. *Report* was collected at night, capturing a batch environment in which reports of the day's activities were generated.

4.3 Disks

We chose to simulate members of the HP C2240 series of drives for two reasons. First, our simulator correctly models the behavior of this line of drives. Second, we do not have detailed specifications for the disks used on the traced systems, nor access to such disks in order to extract this information.

This disk substitution leads to two significant difficulties. First, our base disk (the HP C2247) has a different storage capacity than the disks used by the traced systems. To better match the size of the simulated disks to those in the traced systems, we modify the number of platters (5 for *Sci-TS*, *Order*, and *Report*; and 9 for *Cello*, *Snake*, and *Air-Rsv*) to create disks large enough to contain the active data without excessive unused media. We feel that this is a reasonable approach, as a production line of disks often differs only in the number of physical platters.

The second and more important issue is that HP C2240 disks service requests at a different rate than the disks used in the traced systems. An ideal model would incorporate feedback effects between request completions and subsequent arrivals. Unfortunately, information about how individual request arrivals depend upon previous completions is not present in the traces. Developing a methodology for realistically modeling such feedback effects is an area for future research. The approach used in this work was to scale the traced interarrival times to produce a range of average interarrival rates. When the scale factor is one, the simulated interarrival times match those traced. When the scale factor is two, the traced interarrival times are halved (doubling the average arrival rate). However, even with a identity scaling factor the workload would undoubtedly have been different if the system traced were using HP C2240 disk drives. This is a common problem with this type of trace-driven simulation (which behaves as an open system), but we believe that the qualitative results and insights are valid.

4.4 Metrics

The average response time (across all disks) is the primary metric used for comparing the various scheduling algorithms. The squared coefficient of variation (σ^2/μ^2) is also used, as in [Teor72]. Given a constant average response time, a decrease in the coefficient of variation implies reduced response time variance (i.e., improved starvation resistance).

5 Synthetic Workloads

Synthetically generated random workloads were used in order to replicate previous work and to obtain a starting point for further experiments. The request stream consisted of 8 KB accesses (a typical file system block size) uniformly distributed across the available logical space. The request interarrival times were exponentially distributed, with the mean varied to generate lighter or heavier workloads. The ratio of reads to writes was set to 2:1. Each data point is the average of at least three separate runs of 250,000 disk requests, corresponding to simulated workloads of 50 to 400 hours of activity. Unless otherwise noted, the disk model was configured with 38 slipped tracks (matching the largest defect list found in our experimental HP C2247 disk drives).

The results are partitioned into three groups based on the level of information available to the scheduling algorithms: scheduling based on LBN, scheduling given an accurate LBN-to-PBN mapping, and scheduling with full knowledge (including current disk head position, overhead delays, cache contents, etc.).

5.1 Scheduling by Logical Block Number

Even if the scheduler has little or no knowledge of LBN-to-PBN mappings, it can approximate seek delays via the "distance" between logical block numbers for individual requests. For example, if a request to logical block 100 has just completed, an LBN-based C-LOOK scheduler will select a pending request for logical block 150 over one for logical block 200. The accuracy of this approximation is determined by the scheduling algorithm, the variance in sectors per track between zones, the defect management scheme, and any cache prefetching activity.

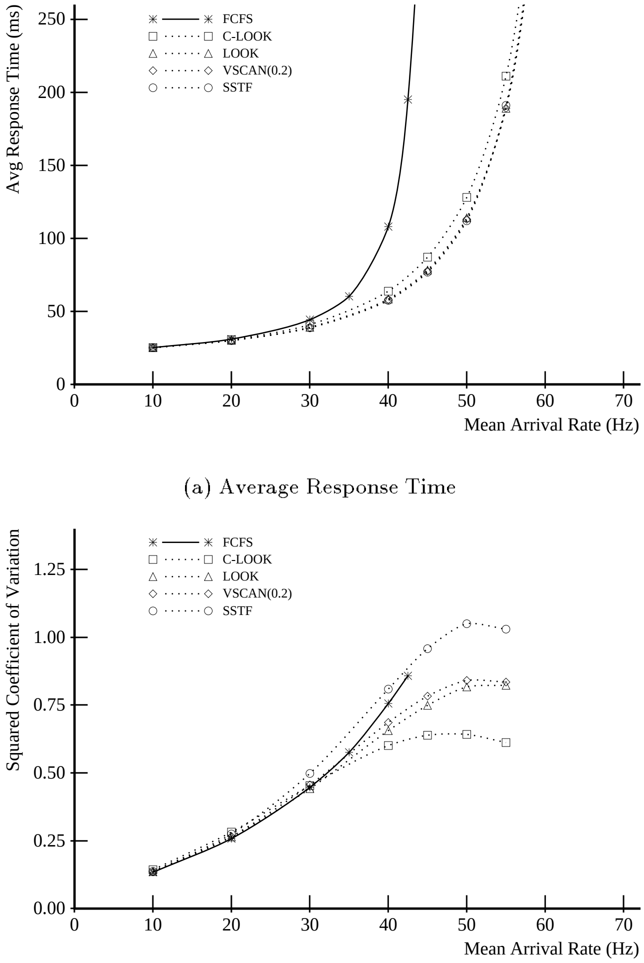
Figure 2a graphs the average response times for FCFS, LOOK, C-LOOK, SSTF and VSCAN(0.2) for a range of average arrival rates. As expected, FCFS quickly saturates as the workload increases. Since C-LOOK is designed to reduce the variance of response times (as well as the mean), its average response times run 5-10% higher than those of LOOK, SSTF, and VSCAN(0.2) for medium and heavy workloads. For sub-saturation workloads, SSTF provides a slightly lower mean response time (up to 1% under VSCAN(0.2) and 2% under LOOK). For workloads at the edge of saturation, SSTF loses to VSCAN(0.2) and LOOK by similar margins.

Figure 2b shows the corresponding squared coefficients of variance. FCFS has the lowest coefficient for lighter workloads. As FCFS begins to saturate and its response time variance expands, C-LOOK emerges as the algorithm with the best starvation resistance. SSTF, as expected, is highly susceptible to starvation.

These results compare well with previous studies of scheduling for random workloads [Geis87, Selt90, Teor72].

5.2 Scheduling with Known Mapping

If the scheduler has knowledge of LBN-to-PBN mappings, it can more accurately predict seek delays and thereby



(a) Average Response Time

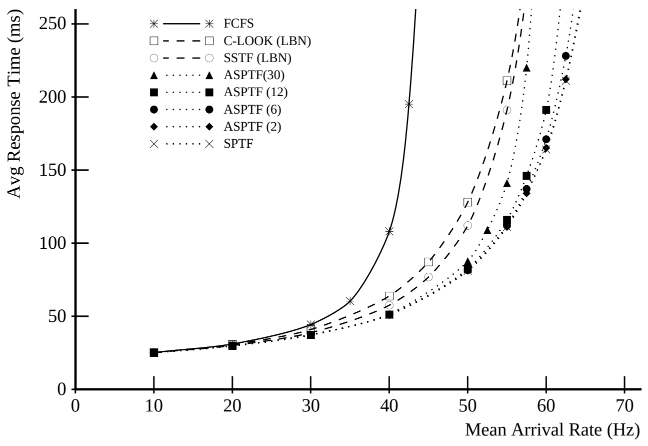
(b) Squared Coefficient of Variation (Response Times)

Figure 2: LBN-Based Scheduling of Random Workloads

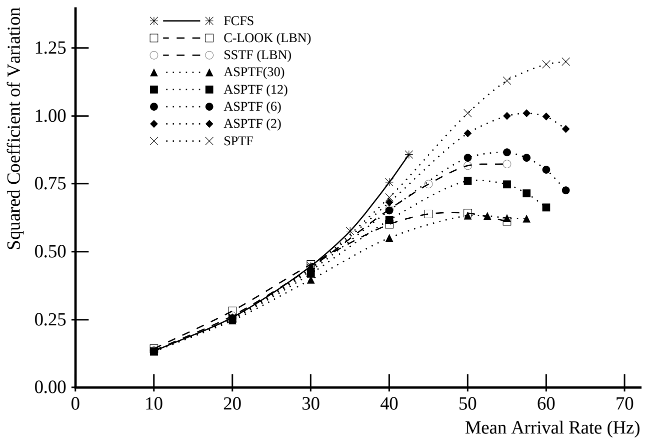
produce better schedules. However, a heuristic for scheduling requests within a cylinder must be chosen. We employ a C-LOOK scheme in order to take advantage of the HP C2247's prefetching cache (see section 6). Thus each algorithm uses C-LOOK within a cylinder (until all requests within the cylinder are satisfied) and one of the experimental algorithms when selecting the next cylinder to service.

The LBN-based and full-map versions of each algorithm produce virtually identical performance, as measured by both average response times and coefficients of variation. As random workloads contain few sequential requests, prefetching provides little or no performance improvement. The cylinder heuristic also has little effect, as individual cylinders rarely contain multiple pending requests.

To determine the influence of excessive defects on the scheduler's accuracy, we modified the simulator to model an HP C2247 with half of its spare tracks (450 out of 900) filled



(a) Average Response Time



(b) Squared Coefficient of Variation (Response Times)

Figure 3: Full-Knowledge Scheduling of Random Workloads

due to randomly grown defects. To maximize the perturbation in the LBN-to-PBN mapping, all of the defects were dynamically reallocated. Even with over 2% of the disk's tracks remapped, full mapping information provides little improvement in performance. SSTF, the algorithm most sensitive to the LBN-to-PBN mapping, improves by less than 1%.

We conclude that maintaining a full LBN-to-PBN mapping is not justified for seek-reducing algorithms when scheduling random workloads.

5.3 Scheduling with Full Knowledge

Given sufficient computational resources, a full LBN-to-PBN mapping, accurate values for various control and communication overheads, and some indication of the drive's current rotational position, the scheduler can select the pending request which will incur the smallest total positioning delay (seek and rotational latency). As with SSTF,

Trace Name	Length (hrs)	Disks	Number of Requests	Avg Size (KB)
Cello	168	8	3,262,824	6.3
Snake	168	3	1,133,663	6.7
Air-Rsv	9	16	2,106,704	5.1
Sci-TS	19.6	43	5,187,693	2.4
Order	12	22	12,236,433	3.1
Report	8	22	8,679,057	3.9

Table 2: Basic Characteristics of the Traces

SPTF is highly susceptible to request starvation. For this reason, we also examined an aging algorithm based on SPTF. Denoted as *Aged Shortest Positioning Time First* (ASPTF), this algorithm is equivalent to the ASATF algorithm proposed in [Jaco91]. ASPTF adjusts each positioning delay prediction (T_{pos}) by subtracting a weighted value corresponding to the amount of time the request has been waiting for service (T_{wait}). The resulting effective positioning delay (T_{eff}) is used in selecting the next request:

$$T_{eff} = T_{pos} - (W * T_{wait})$$

The weight suggested in [Jaco91] translates approximately to ASPTF(6.3). Figures 3a and 3b show performance data for values of W ranging from 0 (SPTF) to 30. As W increases, the average response time slowly grows, but response time variance drops significantly. In all cases, the algorithms that schedule based on positioning times have lower average response times than those based only on seek times. With higher values of W, ASPTF suffers a sharp increase in average response time when the disk begins to saturate. For a sufficiently large W, ASPTF degenerates into FCFS.

An appropriate aging factor may result in an algorithm with SPTF's performance and C-LOOK's starvation resistance. For example, ASPTF(6) and ASPTF(12) match the performance of SPTF for all but the heaviest workloads, yet have much better starvation resistance. In fact, ASPTF(12) has lower response time variance than C-LOOK, even though its coefficient of variation is slightly higher.

Selecting the request with the smallest positioning delay entails significant computational effort (especially for large queues). See [Jaco91] for further information on this topic.

6 Traced Workloads

To gain an understanding of how various scheduling algorithms perform under actual user workloads, we used traces of disk activity collected from systems in use at various industry and research installations. Some basic characteristics of the traces are shown in table 2. They vary widely in read/write ratios, access sizes, interarrival rates, degrees of sequentiality and burstiness.

Trace	Scale	C-LOOK	LOOK	VSCAN	SSTF
Cello	1.5	134	129	129	128
Snake	1.25	71.5	73.7	73.7	73.6
Air-Rsv	2.5	44.2	51.2	51.2	53.9
Sci-TS	2.5	31.0	54.7	55.7	57.5
Order	1.0	29.6	51.7	51.1	51.3
Report	1.0	59.4	63.2	64.8	80.2

(a) Average Response Times (ms) with a Prefetching Cache

Trace	Scale	C-LOOK	LOOK	VSCAN	SSTF
Cello	1.25	145	134	134	133
Snake	0.34	65.5	65.5	65.4	65.2
Air-Rsv	2.0	44.8	43.1	42.9	43.1
Sci-TS	1.69	92.6	64.4	63.6	61.8
Order	0.67	57.1	52.8	52.2	51.4
Report	0.71	60.2	55.3	55.5	54.6

(b) Average Response Times (ms) with a Speed-Matching Buffer

Table 3: Average Response Times for LBN-Based Algorithms (taken from the "knees" of the Average Response Time curves)

The performance of the algorithms is graphed over a range of scaling factors for each trace. Note that the scaling factor (the X-axis) is shown in $\log_2$ scale.

6.1 Scheduling by Logical Block Number

Figures 4-9 show the average response times for LBN-based scheduling of the six traces. As expected, FCFS scheduling performs poorly for all but the lightest workloads. The relative performance of the other algorithms, however, differs from that observed for the random workloads. The results indicate that the disk's prefetching cache plays a large role in determining algorithm effectiveness. If requests arrive at the disk in logically ascending order, some portion (or all) of the data for each sequential read request may already be in the prefetching cache when the request arrives. Therefore, algorithms that preserve any existing read sequentiality may achieve higher performance.

Table 3a lists a sample point from the knee of each average response time graph. The LBN-based C-LOOK algorithm, which always schedules requests in logically ascending order, provides higher performance than LOOK, VSCAN(0.2), and SSTF for all traces except *Cello*. LOOK and VSCAN(0.2) generally outperform SSTF, as they schedule sequential requests in logically ascending order during the "ascending" half of the scan cycle. However, LOOK, VSCAN(0.2), and SSTF all exhibit similar performance when the last of a sequence of requests is scheduled before the first. When this occurs, the requests are scheduled to the disk in logically **descending** order, negating the performance advantage of the prefetching cache.

For *Cello*, C-LOOK produces a higher average response time than the other seek-reducing algorithms. The *Cello* environment is dominated by large bursts of write requests to a single disk [Ruem93]. Almost half of the requests are serviced by this disk, with maximum queue lengths approaching 1000 at the identity scaling factor. In addition, this trace contains the smallest fraction of sequential read requests, thus benefiting the least from the prefetching cache.

Additional evidence of the benefits of optimizing for a prefetching cache can be found by comparing tables 3a and 3b. The latter table lists sample points from experiments with the disk "cache" used only as a speed-matching buffer. Under this restriction, the performance of C-LOOK drops dramatically in relation to the other seek-reducing algorithms. This is best seen in the *Sci-TS* workload: C-LOOK **decreases** the average response time by better than 40% over LOOK, VSCAN(0.2), and SSTF when the cache is enabled, but it **increases** the response time by over 40% when the on-board memory is used as a simple buffer.

For some traces, such as *Report*, the performance improvement of C-LOOK over LOOK and SSTF is greater than would be expected from the improvement in cache hit rate. The ascending order of scheduling maintained by C-LOOK matches the prefetching nature of the disk. That is, the disk continues reading data beyond the end of read requests, often moving the read/write head forward one or more surfaces (or even to the next cylinder). So, requests which are logically forward of an immediately previous read may suffer shorter seek delays.

For all traces, C-LOOK provides the best or equivalent starvation resistance (as evidenced by its squared coefficient of variation). This is not surprising, as C-LOOK was invented for this purpose. Predictably, LOOK has the next highest coefficient, followed by VSCAN(0.2) and SSTF.

6.2 Scheduling with Known Mapping

Using full-map versions of the various algorithms, we observe performance improvements for LOOK, VSCAN(0.2), and SSTF under all workloads except *Cello*. This is not due to reduced seek times via better mapping information, but rather is caused by the heuristic we used to schedule requests within a single cylinder. As described in section 5.2, the scheduler uses C-LOOK, LOOK, VSCAN(0.2), or SSTF when moving between cylinders, but uses C-LOOK "within" a cylinder. Therefore, the cylinder-based versions of LOOK, VSCAN(0.2), and SSTF use the prefetching cache more effectively than their LBN-based counterparts.

To verify the importance of the C-LOOK heuristic relative to the full LBN-to-PBN mapping, we modified the scheduling algorithms to use a fixed number of sectors per cylinder (1024) rather than the actual per-zone values. The resulting performance for all traces and algorithms is

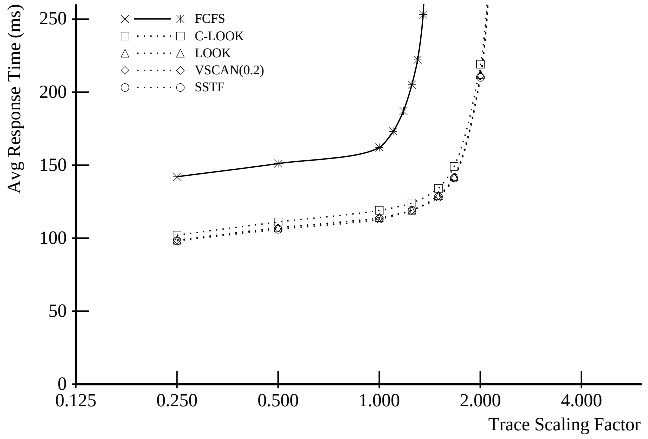


Figure 4: LBN-Based Scheduling of *Cello*

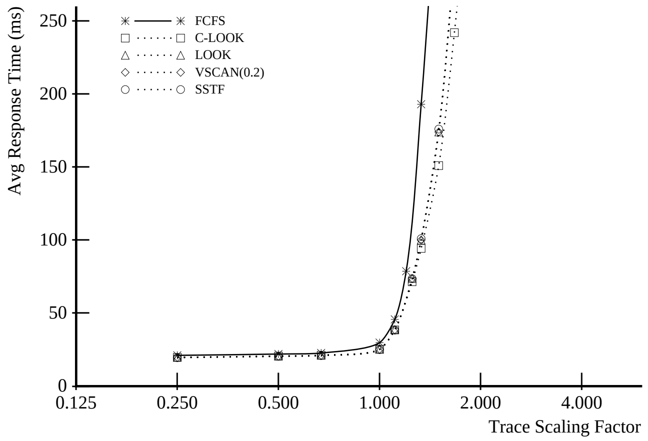


Figure 5: LBN-Based Scheduling of *Snake*

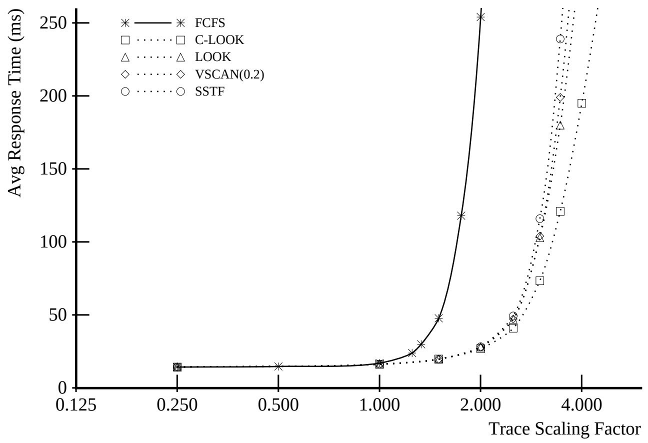


Figure 6: LBN-Based Scheduling of *Air-Rsv*

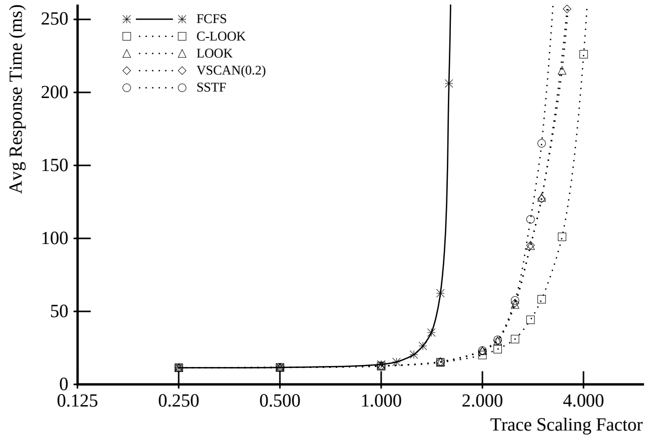


Figure 7: LBN-Based Scheduling of *Sci-TS*

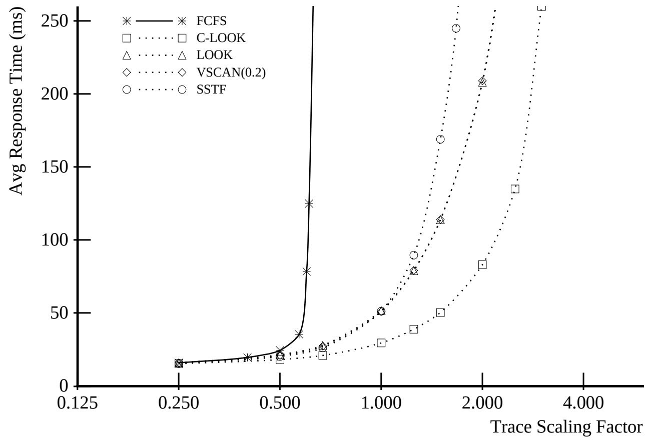


Figure 8: LBN-Based Scheduling of *Order*

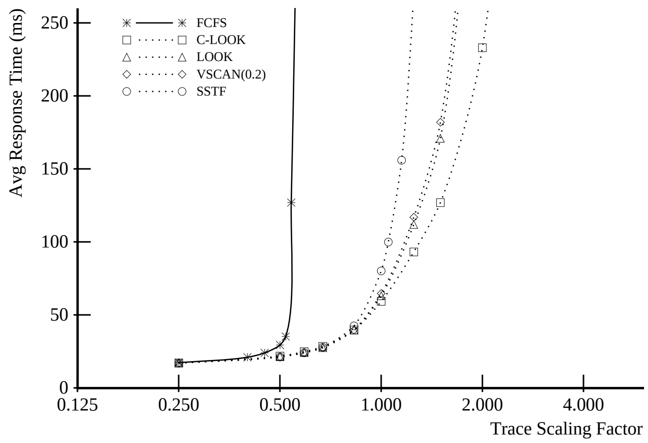


Figure 9: LBN-Based Scheduling of *Report*

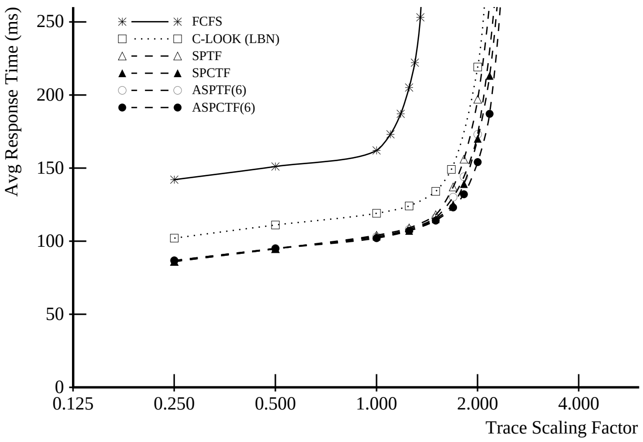


Figure 10: Full-Knowledge Scheduling of *Cello*

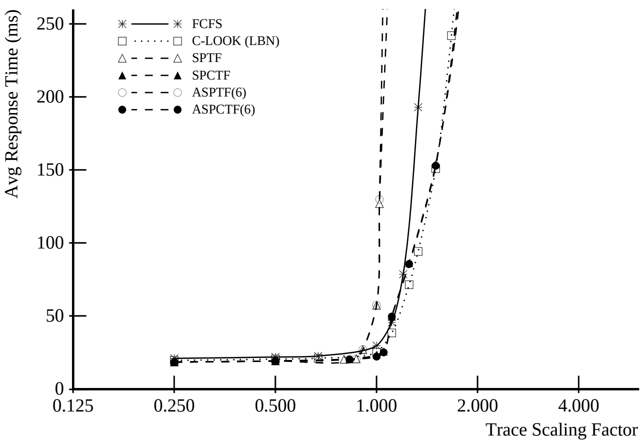


Figure 11: Full-Knowledge Scheduling of *Snake*

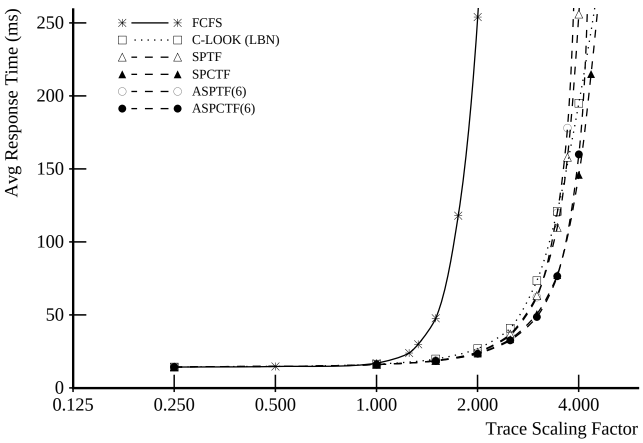


Figure 12: Full-Knowledge Scheduling of *Air-Rsv*

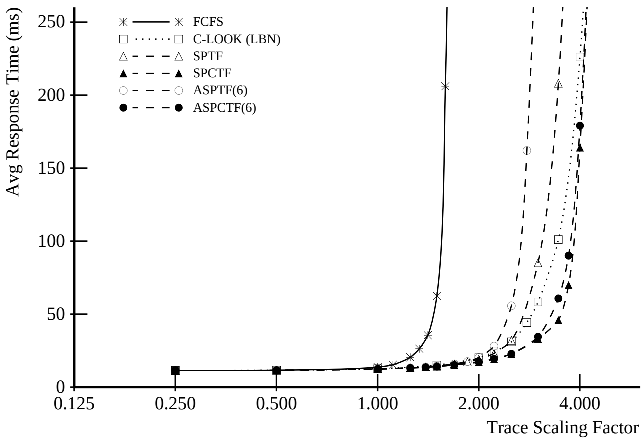


Figure 13: Full-Knowledge Scheduling of *Sci-TS*

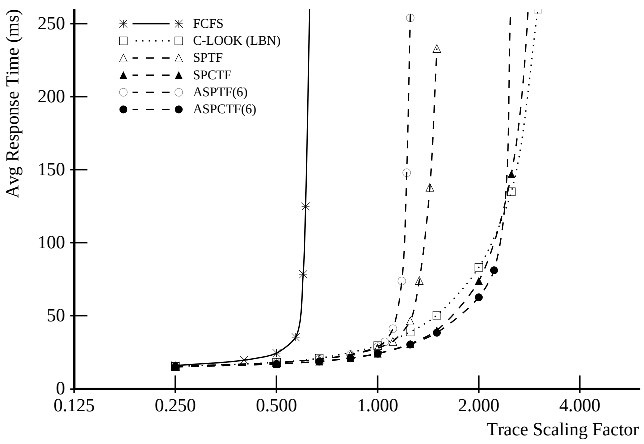


Figure 14: Full-Knowledge Scheduling of *Order*

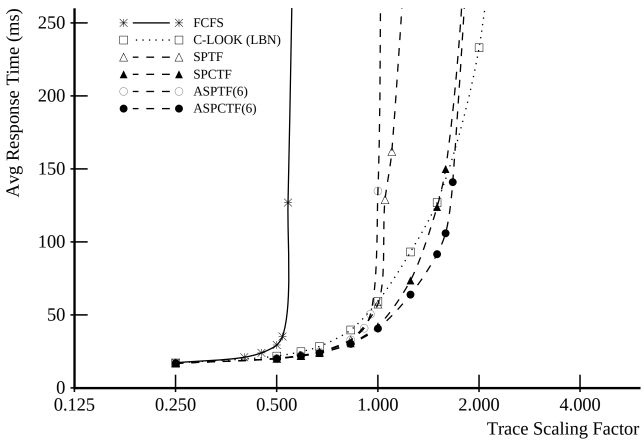


Figure 15: Full-Knowledge Scheduling of *Report*

within 1% of the performance obtained using the actual mappings. We conclude, as was the case for random workloads, that the complexity of maintaining a full LBN-to-PBN mapping is not justified for seek-reducing algorithms.

The relative starvation characteristics of the full-map algorithms generally match the results found for the LBN-based schemes. By satisfying all requests within a cylinder before moving to another cylinder, the algorithms become slightly less resistant to starvation.

6.3 Scheduling with Full Knowledge

Figures 10-15 show average response times for the SPTF-based algorithms described in section 5.3. Because these algorithms do not recognize and exploit the cache directly, we also present data for modified versions which track the contents of the on-board cache and estimate a positioning time of zero for any request that can be satisfied (at least partially) from the cache. The resulting algorithms are denoted as *Shortest Positioning (w/Cache) Time First* (SPCTF) and *Aged Shortest Positioning (w/Cache) Time First* (ASPCTF).

The SPTF-based algorithms have both a significant advantage and disadvantage when compared to C-LOOK. The advantage is the obvious improvement in positioning delay prediction accuracy due to the inclusion of rotational latency information. A full rotation for an HP C2240 series disk takes approximately 11 ms, equivalent to a seek of over 550 cylinders or 27% of the maximum seek distance. The disadvantage is that these algorithms have no inherent bias towards servicing sequential reads in ascending order. Pending requests are selected based on their predicted positioning delay, not their LBN or cylinder. Even when cache knowledge is incorporated, only subsets of a sequential stream may be serviced "in order." As a result, the prefetching cache is less effectively utilized by SPTF-based algorithms.

This trade-off affects response times to different degrees for each of the six traces. For *Cello*, the SPTF-based algorithms are clearly superior to the seek-reducing algorithms. Cache knowledge (i.e., SPCTF and ASPCTF(6)) increases performance by 5-15% under the largest scaling factors. For the other five workloads, the comparison is not as clean. For these traces, SPTF and ASPTF(6) saturate more quickly than C-LOOK. However, they do provide superior performance for some sub-saturation workloads. For *Report*, figure 15 shows that SPTF and ASPTF(6) are significantly (up to 18%) better than C-LOOK in the 0.4-0.9 range of scaling factors. Incorporating cache knowledge widens the window where SPTF-based algorithms are superior to C-LOOK. For *Sci-TS*, SPCTF and ASPCTF(6) provide increased performance over all scaling factors studied.

Figures 16 and 17 present the squared coefficients of variation for two of the traces. Although ASPCTF(6) always

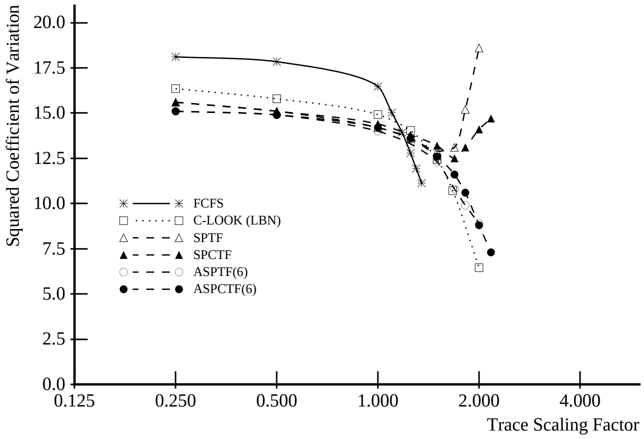


Figure 16: Full-Knowledge Scheduling of *Cello*: Squared Coefficient of Variation (Response Times)

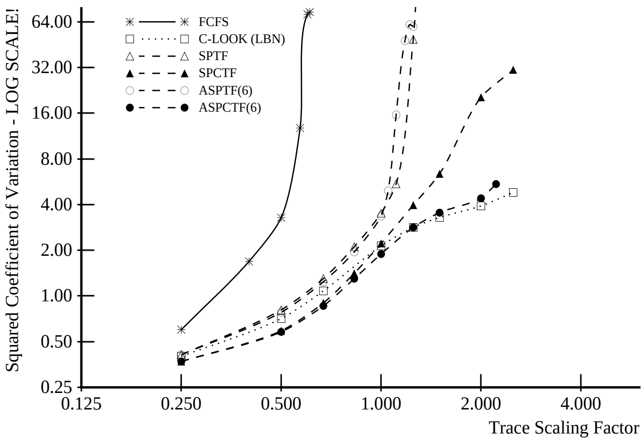


Figure 17: Full-Knowledge Scheduling of *Order*: Squared Coefficient of Variation (Response Times)

has better starvation resistance than SPCTF, their relative performance is workload dependent. For workloads with a significant fraction of sequential read requests, the aging factor can improve both starvation resistance and performance. The aging factor helps to maintain the ordering of sequential requests, provided they were issued in order.

We believe that a modified version of ASPCTF which specifically schedules sequential requests in ascending logical order will increase performance beyond C-LOOK in all cases. This represents another item for future research.

7 Conclusions and Future Work

Modern disk drives have several features which can affect the ability of disk scheduling algorithms to produce good schedules. Using strongly-validated simulation, we have re-examined popular scheduling algorithms to determine the importance of some of these features. Synthetic workloads were used to allow comparisons with previous work. Extensive traces gathered from six different environments were used to represent more "realistic" workloads.

Features such as zoned recording, track/cylinder skew, and defect reallocation complicate the translation of logical block numbers into physical media locations. Although full mapping knowledge may be required for the most complex scheduling algorithms (e.g., SPTF), our results indicate that it provides little or no performance improvement for seek-reducing algorithms (less than 2%).

C-LOOK, which always schedules requests in logically ascending order, best exploits the prefetching cache for workloads with significant read sequentiality. For random workloads, C-LOOK has been shown (both in this work and in previous studies) to provide slightly inferior performance to other seek-reducing algorithms (e.g., SSTF and LOOK). For five of the six real-world traces, however, C-LOOK achieves the highest cache hit rates and lowest average response times. In addition, the LBN-based C-LOOK algorithm is straightforward and relatively simple to implement.

More powerful disk controllers can use variants of the Shortest Positioning Time First (SPTF) algorithm to achieve even higher performance. The use of such algorithms requires thorough knowledge of the disk's current state as well as the management schemes employed by the disk drive firmware. The computational cost (as indicated crudely by our simulation times) is very high. Also, our results indicate that it is critical for such an implementation to recognize the existence of on-board disk caches and optimize for them.

We are currently pursuing several avenues of research in scheduling. First, we are validating our results for other disk drives. There are also drive characteristics which we have not yet explored. Most modern disks can maintain a queue of outstanding requests within their embedded controllers. Scheduling algorithms for such on-board queues may differ in nature from those outside the disk drive package. In addition, most drives can signal completion as soon as write request data have reached the on-board disk cache. The use of such *fast-writes* may require new scheduling techniques to maintain reliability guarantees and effectively utilize the available cache storage.

Finally, we believe that scheduling decisions should be based both on subsystem characteristics and system-level requirements. By understanding the system performance consequences of different disk requests, one can design scheduling algorithms optimized for overall system performance [Gang93]. For example, a request that prevents a critical process from executing should be serviced before a low-priority background request. Such goals must be tempered with effective utilization of the storage resources, resulting in a complicated set of trade-offs which we plan to explore.

8 Acknowledgements

We thank Jim Browning, John Fordemwalt, David Jaffe, Rama Karedla, Richie Lary, Mike Leonard, Joe Pasquale, Rusty Ransford, Chris Ruemmler, and John Wilkes for their insights on various I/O issues. In particular, we thank Chris Ruemmler and John Wilkes (of HP Labs) and Richie Lary and Rama Karedla (of DEC) for the traces. We also wish to acknowledge the other members of our research group at the University of Michigan. Finally, our research group is very fortunate to have the financial and technical support of several industrial partners, including NCR Corporation, Intel, Motorola, SES, HaL, DEC, MTI and Hewlett-Packard.

References

[Denn67] P. J. Denning, "Effects of scheduling on file memory operations", *AFIPS Spring Joint Computer Conference*, April 1967, pp. 9-21.

[Gang93] G. Ganger, Y. Patt, "The Process-Flow Model: Examining I/O Performance from the System's Point of View", *SIGMETRICS*, 1993, pp. 86-97.

[Geis87] R. Geist, S. Daniel, "A Continuum of Disk Scheduling Algorithms", *ACM Transactions on Computer Systems*, February 1987, pp. 77-92.

[HP92] Hewlett-Packard Company, "HP C2240 Series 3.5-inch SCSI-2 Disk Drive, Technical Reference Manual", Part Number 5960-8346, Edition 2, April 1992.

[Jaco91] D. Jacobson, J. Wilkes, "Disk Scheduling Algorithms Based on Rotational Position", Hewlett-Packard Technical Report, HPL-CSP-91-7, Feb. 26, 1991.

[McNu86] B. McNutt, "An Empirical Study of Variations in DASD Volume Activity", *CMG*, 1986, pp. 274-283.

[Mert70] A. G. Merten, "Some quantitative techniques for file organization", Ph.D. Thesis, Technical Report No. 15, U. of Wisconsin Comput. Center, 1970.

[Rama92] K. Ramakrishnan, P. Biswas, R. Karedla, "Analysis of File I/O Traces in Commercial Computing Environments", *ACM SIGMETRICS*, 1992, pp. 78-90.

[Ruem93] C. Ruemmler, J. Wilkes, "UNIX Disk Access Patterns", *Winter USENIX*, 1993.

[Ruem94] C. Ruemmler, J. Wilkes, "Modelling Disks", *IEEE Computer*, March 1994.

[Seam66] P. H. Seaman, R. A. Lind, T. L. Wilson "An analysis of auxiliary-storage activity", *IBM System Journal*, Vol. 5, No. 3, 1966, pp. 158-170.

[Selt90] M. Seltzer, P. Chen, J. Ousterhout, "Disk Scheduling Revisited", *Winter USENIX*, 1990, pp. 313-324.

[Teor72] T. Teorey, T. Pinkerton, "A Comparative Analysis of Disk Scheduling Policies", *Communications of the ACM*, March 1972, pp. 177-184.

[Wort94] B. Worthington, G. Ganger, Y. Patt, "Scheduling for Modern Disk Drives and Non-Random Workloads", U. of Michigan, Technical Report CSE-TR-194-94, 1994.