Operating System Management of MEMS-based Storage Devices

John Linwood Griffin, Steven W. Schlosser,
Gregory R. Ganger, David F. Nagle

May 2000
CMU-CS-00-136

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

MEMS-based storage devices promise significant performance, reliability, and power improvements relative to disk drives. This paper explores how the physical characteristics of these devices change four aspects of operating system management: request scheduling, data placement, failure management, and power management. Adaptations of disk request scheduling algorithms are found to be appropriate for these devices; however, new data placement schemes are shown to better match their differing mechanical positioning characteristics. With aggressive internal redundancy, MEMS-based storage devices can tolerate failure modes that cause data loss for disks. In addition, MEMS-based storage devices enable a finer granularity of OS-level power management because the devices can be stopped and started rapidly and their mechanical components can be individually enabled or disabled to reduce power consumption.

This research is supported by the member companies of the Parallel Data Consortium. At the time of this writing, these companies include CLARiiON Array Development, EMC Corporation, Hewlett-Packard Labs, Hitachi, Infineon Technologies, Intel Corporation, LSI Logic, MTI Technology Corporation, Novell, Inc., PANASAS, L.L.C., Procom Technology, Quantum Corporation, Seagate Technology, Sun Microsystems, Veritas Software Corporation, and 3Com Corporation. We thank IBM Corporation for supporting our research efforts. John Griffin is supported in part by a National Science Foundation Graduate Fellowship.

Keywords: Microelectromechanical systems (MEMS), operating systems, secondary storage, request scheduling, data placement, fault tolerance, power management.

1 Introduction

Decades of research and experience have provided operating system builders with a healthy understanding of how to manage disk drives and their role in systems. This management includes such issues as achieving acceptable performance despite the relatively time-consuming mechanical positioning delays, dealing with transient and permanent hardware problems so as to achieve high degrees of data survivability and availability, and minimizing power dissipation in battery-powered mobile environments. To address these issues, a wide array of OS techniques are used, including request scheduling, data layout, prefetching, caching, block remapping, data replication, and device spin-down. Given the prevalence and difficult nature of disks, most of these techniques have been specifically tuned to the physical characteristics of disks.

When other devices (*e.g.*, magnetic tape, Flash RAM) are used in place of disks, the characteristics of the problems change. Putting new devices behind a disk-like interface is generally sufficient to achieve a working system; however, the OS techniques must be tuned to a particular device's characteristics to achieve the best performance, reliability, power consumption, *etc.* For example, request scheduling techniques are much less important for RAM-based storage devices than for disks, since location-dependent mechanical delays are not involved. Likewise, locality-enhancing block layouts, such as cylinder groups [MJLF84], extents [MK91], and log-structuring [RO92], are not as beneficial. However, for storage devices based on Flash RAM, log-structured file systems with idle-time cleaning can increase both performance and device lifetimes [DCK$^+$94, KL99].

Microelectromechanical systems (MEMS)-based storage is an exciting new technology that will soon be available in computer systems. MEMS are very small scale mechanical structures—on the order of 10s to 1000s of micrometers—fabricated on the surface of silicon wafers [Wis98]. These microstructures are created using the same photolithographic processes used to manufacture other semiconductor devices (*e.g.*, CPUs and memory) [FSR$^+$96]. MEMS structures can be made to slide, bend, and deflect in response to electrostatic or electromagnetic forces from nearby actuators or from external forces in the environment. Using minute MEMS probe tips, data bits can be stored in and retrieved from magnetic media coated on a movable silicon substrate ("media sled") [CBF$^+$00, GSGN00]. Practical MEMS-based storage devices are the goal of major efforts at many research centers, including IBM, Carnegie Mellon University, Hewlett-Packard, and UC Berkeley.

Like disks, MEMS-based storage devices have mechanical and magnetic characteristics that merit specific OS techniques to manage performance, fault tolerance, and power conservation. For example, the mechanical positioning delays (*e.g.*, seek and settle time) for MEMS-based storage devices depend on the current and destination position and velocity of the media sled, just as disks are dependent on the arm position and platter rotational offset. However, the mechanical expressions that characterize sled motion differ from those describing platter and arm motion. Knowledge of these differences impacts both scheduling and layout decisions at the OS level. Similar examples exist for OS fault management and power conservation mechanisms. To assist designers of both MEMS-based storage devices and the systems that use them, an understanding of the options and trade-offs for OS management of these devices must be developed.

Our work takes a first step towards developing this understanding of OS management techniques for MEMS-based storage devices. In this report, we describe the movable media sled design that is being developed independently by several groups. With higher storage densities and lower random access times (<1 ms) than disks, these devices could play a significant role in future systems. After describing a disk-like view of these devices, we compare and contrast their characteristics with those of disks. Building on these comparisons, we explore options and implications for three major OS management issues: performance (specifically, request scheduling and block layout), failure management (media defects, device failures, and host crashes), and power management.

While these explorations are unlikely to represent the final word for OS management of these newly-emerging devices, we believe that several of our high-level results will remain valid: (1) Disk scheduling algorithms can also be adapted to MEMS-based storage devices, resulting in relative values that roughly match their rankings for disks. (2) We find that disk layout techniques can be adapted usefully, but that the Cartesian movement of the sled (instead of rotational motion) allows further refinement of layouts to provide benefit. (3) Striping of data across tips can greatly increase a system's tolerance to media, tip, and electronics faults; in fact, many faults that would cause data loss in disks can be made recoverable in MEMS-based storage devices. (4) Miniaturization and lack of rotation make these devices much more

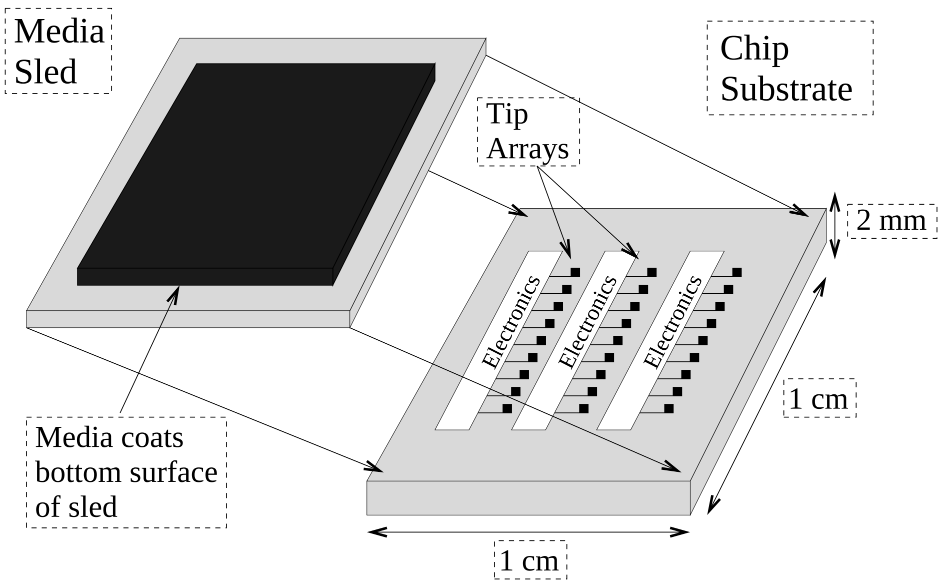


Figure 1: ***The "moving media" model.*** *The media sled is suspended above the array of fixed tips. The sled moves small distances along the X and Y axes, allowing the fixed tips to address 30–50% of the total media area. This yields capacities of gigabytes per square centimeter.*

power-friendly than disks; to first order, power dissipation is a linear function of the number of bits read or written. This makes power optimization equivalent to data access minimization (*e.g.*, adapting the rate at which applications consume data [NSN+97, FS99]). Also, not having the large mechanical delay involved in spinning up or down disks improves the availability of power-optimized MEMS devices by reducing restart times after power-down.

The remainder of this paper is organized as follows. § 2 describes MEMS-based storage devices, focusing on how they are similar to and different from magnetic disks. § 3 describes our experimental setup, including the simulator and the workloads used. § 4 evaluates request scheduling algorithms for MEMS-based storage devices. § 5 explores data layout optimizations. § 6 describes approaches to fault management within and among MEMS-based storage devices. § 7 discusses their power usage characteristics and their impact on power management. § 8 summarizes this work's contributions.

2 MEMS-based Storage Devices

This section describes a MEMS-based storage device and compares and contrasts its characteristics with those of conventional disk drives. The description, which follows that of [GSGN00], maps the devices' access and layout characteristics onto a disk-like metaphor to further clarify similarities and differences.

2.1 Basic Device Description

MEMS-based storage devices use the same basic magnetic recording technologies as disks, relying on MEMS microstructures to position miniature probe tips over specific magnetic media locations. Because long-lasting rotating structures are difficult to achieve in silicon, MEMS-based storage devices are unlikely to include the rotating platters coated with magnetic media that characterize disks. Instead, most current designs contain a movable sled coated with magnetic media. This sled is spring-mounted above a two-dimensional array of probe tips and can be pulled in the X and Y dimensions by electrostatic forces applied by comb actuators at each edge. Unlike disk arms, the probe tips remain stationary under the media (except for minute tip movement to adjust for skewed tracks and sled surface variations). Therefore, the sled is responsible for positioning/seek movements, as opposed to disk platters that share this role with seek arms. Figures 1 and 2 illustrate this MEMS-based storage design.

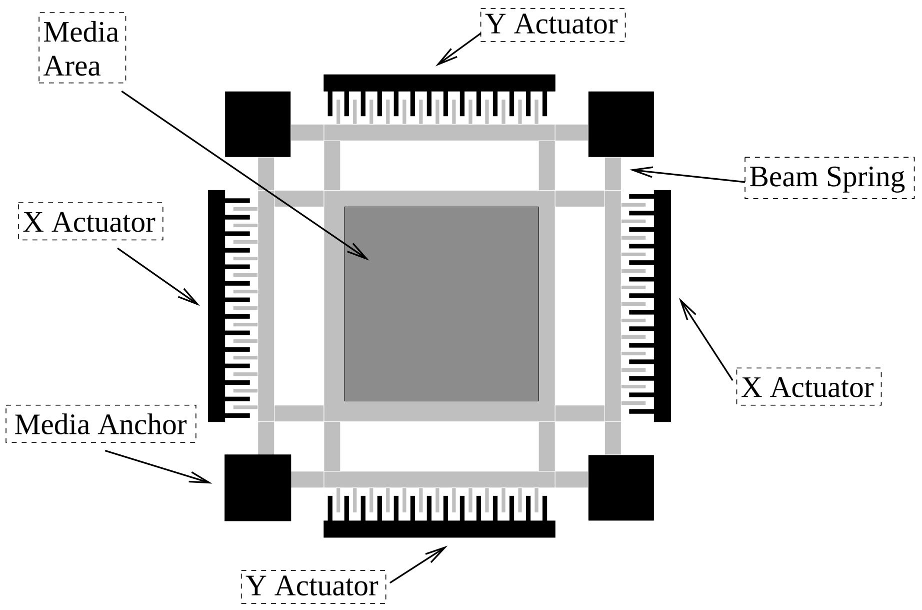


Figure 2: ***The suspended media sled in the moving media model.*** *The actuators, spring suspension, and the media sled itself are shown. Anchored regions are black and the movable structure is shaded grey.*

Concretely, an example MEMS-based storage device [CBF$^+$00] might have a media area on the sled of about 1 cm^2, under which perhaps 10,000 probe tips could be placed. For designs with a bit cell of 0.0025 μm^2 (50 nm per side) and encoding/ECC overheads of roughly 2 bits per byte, these devices have a capacity of about 4 gigabytes per square centimeter. Note the square nature of the bit cells; because the probe tips are so much smaller than disk heads, the bits stored on these devices can have a 1-to-1 aspect ratio, resulting in densities 15–30X greater than those of disk drives. The per-device media areas are smaller than the usable area on disk platters; however, several MEMS-based storage devices could easily be packaged in a disk form factor to increase capacity. The mechanically-positioned components also have much smaller masses than their corresponding disk parts, allowing random access times in only 100s of microseconds. For the default parameters used in this paper, the average random 4 KB access time is 500 μs.

2.2 Low-level Data Layout

The magnetic media on the sled is organized into rectangular regions as shown in Fig. 3. Each rectangular area stores N×M bits (*e.g.*, 2500×2500 bits) and is accessible by exactly one probe tip. The smallest accessible unit of data is a "tip sector" consisting of servo information (10 bits) and encoded data/ECC (80 bits = 8 encoded data bytes). Multiple tip sectors are grouped into *logical sectors*, similar to logical blocks in SCSI disks. Unlike most conventional disks, multiple probe tips can access the media in parallel—thus many tip sectors can be read or written simultaneously. Due to power and heat considerations, it is unlikely that all probe tips can be active simultaneously; rather, we expect groups of 200–2000 tips to be the norm.

In organizing the low-level media structure, we identify each bit by the triple $\langle x, y, tip\rangle$, where $\langle x, y\rangle$ represents bit coordinates within the region addressable by $\langle tip\rangle$. Each active probe tip reads or writes data within a column of bits (called a *tip track*; see Fig. 3) as the media sled moves along the Y axis. A tip track contains M bits, each with identical values for $\langle x, tip\rangle$. Drawing on analogies from disk terminology, we refer to the set of all bits with identical values for $\langle x\rangle$ as a *cylinder* (shown in Fig. 4). In other words, a cylinder consists of all bits that are accessible by any tip without moving the sled along the X axis; there are N cylinders per device. Because only a subset of probe tips can be active at once (recall the power and heat considerations above), cylinders are divided into *tracks*. A track consists of all bits within a cylinder that can be read or written by concurrently active tips. In Fig. 4, tips A1, A2, A3 and A4 are active and the corresponding track is indicated. As with conventional disks, reading or writing a complete cylinder requires multiple passes with track switches (*i.e.*, switching which tips are active) in between.

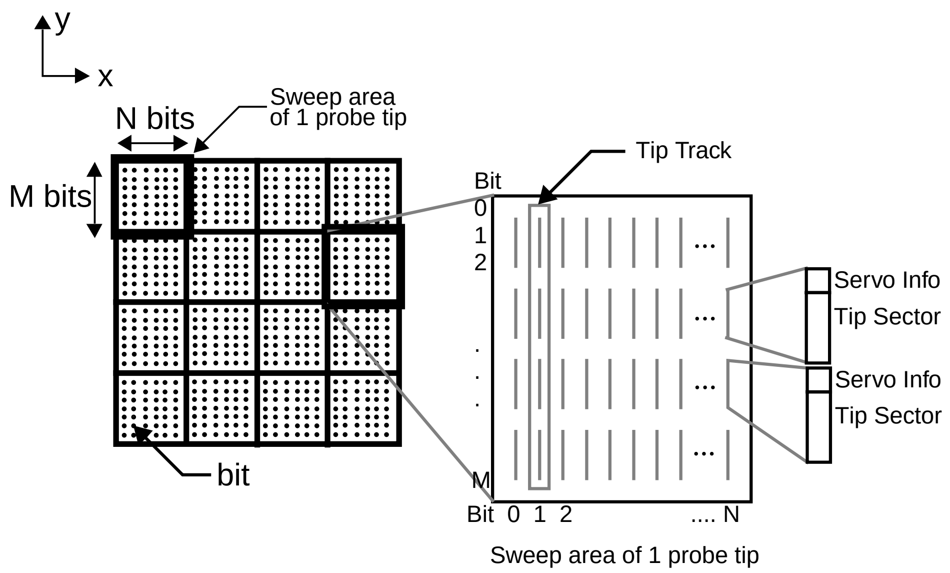


Figure 3: ***Data organization of MEMS-based storage.*** *The illustration depicts a small portion of the magnetic media sled. Each rectangle outlines the area accessible by a single probe tip, with a total of 16 tip regions shown. (A full device contains thousands of tips and tip regions.) Each region stores N×M bits, organized into vertical "tip sectors" containing encoded data and ECC bits. These tip sectors are demarcated by "servo information" strings that identify the sector and track information encoded on a disk. To read or write data, the media passes over the active tip(s) in the ±Y direction while the tips access the media.*

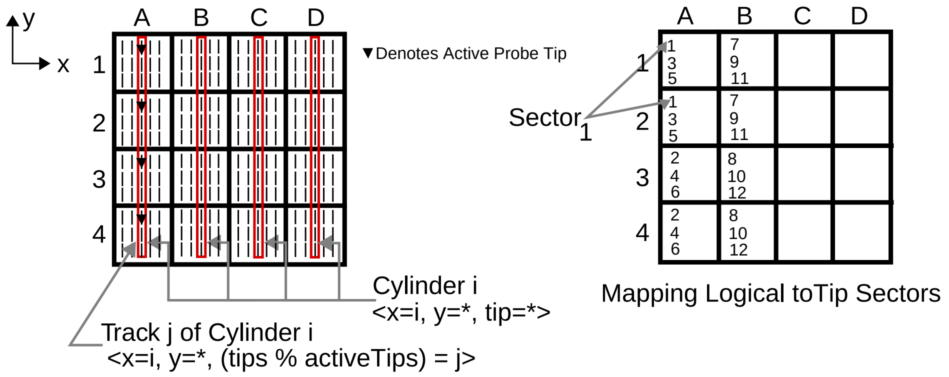


Figure 4: ***Cylinders, Tracks, and Sectors.*** *Cylinder*$_i$ *is defined as all of the columns of data with the same X coordinate:* $\langle x = i, y, tip \rangle$. *Track*$_{i,j}$ *is the subset of a cylinder that is accessible by the concurrently active tips:* $\langle x = i, y, (tip \ \% \ activeTips) \ = j \rangle$. *(Note that activeTips=4 in this figure and that the tips are linearly numbered such that A1=0, A2=1, etc.) Each logical sector in the figure to the right consists of two tip sectors. For example, Sector*$_1$ *consists of the first tip sectors of the two upper tip regions, A1 and A2.*

2.3 Media Access Characteristics

Because multiple tips are active simultaneously, logical sectors can be striped across tip sectors (*i.e.*, under multiple tips) to reduce access time. Fig. 4 illustrates a layout where each logical sector is striped across two tip sectors. In order to read logical sectors 1 and 2, tips A1 through A4 are activated while the sled seeks to the top of cylinder 2 and moves down (in the −Y direction) across the first tip sector. Tip A1 reads half of logical sector 1, tip A2 reads the other half, and tips A3 and A4 read logical sector 2. In this paper, logical sectors of 512 bytes are striped across 64 tip sectors of 8 bytes each.

Media access requires constant sled velocity in the Y direction (and zero velocity in the X direction). This *access velocity* is a design parameter and is determined by the maximum per-tip read and write rates, the bit width, and the sled actuator force. Large transfers may require that data from multiple tracks and/or cylinders be accessed. To switch tracks during large transfers the sled performs a *turnaround* (reversing direction such that $\langle x, y\rangle_{final} = \langle x, y\rangle_{initial}$ and $v_{final} = -v_{initial}$) and switches the set of active tips. The turnaround time is expected to dominate any additional activity (such as the time to switch the set of active tips) during both track and cylinder switches.

Positioning the sled for read or write involves several mechanical and electrical actions. To seek to a desired sector, the appropriate probe tips must be activated, the sled must be positioned so the tips are under the first bit of the pre-sector servo information, and the sled must be moving in the correct direction and velocity. Accomplishing this can be tricky: whenever the sled moves in X (*i.e.*, the destination cylinder differs from the starting cylinder) extra *settling time* must be taken into account—the rapid acceleration and deceleration of the sled causes the spring-sled system to momentarily oscillate in X before damping to $v_x = 0$. In addition, the spring restoring force[1] makes the sled acceleration a function of instantaneous sled position. One or two turnarounds are also necessary whenever the sled is moving in the wrong direction before or after the seek; the turnaround time is also affected by the spring restoring force and is therefore a function of both instantaneous sled position and direction of motion.

2.4 Comparison to Conventional Disks

The remainder of this section enumerates a number of relevant similarities and differences between MEMS-based storage devices and conventional disk drives. With each item, we also discuss consequences for device management issues and techniques.

2.4.1 Mechanical positioning

Both disks and MEMS-based storage devices have two main components of each access' positioning time (seek and rotation for disks; X and Y seeks for MEMS-based storage devices). The major difference is that the two proceed independently for disks, because rotation is independent, whereas the two are explicitly done in parallel for MEMS-based storage devices. Thus, the total positioning time is the greater of the X and Y seek times, making the shorter of the two times irrelevant. The effect of this overlap on request scheduling is explored in § 4.2.

2.4.2 Settling times

For both disks and MEMS-based storage devices, it is necessary for read/write heads to settle over the desired track after a seek. However, the settling time for disks is a relatively small component of most seek times (*e.g.*, 0.5 ms of 1–15 ms seeks). For MEMS-based storage devices, settling time is expected to be a relatively substantial component of seek time (*e.g.*, 0.2 ms of 0.2–0.7 ms seeks). Because the settling time is relatively constant, this has the effect of making seek times more constant, which in turn could reduce (not eliminate) the benefit of both request scheduling and data placement. § 4.4 analyzes this issue in greater detail.

[1]As the sled is displaced during seeks, the springs apply a mechanical restoring force (recall $F_{spring} = k\Delta x$ for spring-mass systems) up to ±75% of the sled actuating force. The spring effects are studied in detail in [GSGN00].

2.4.3 Logical-to-physical mappings

As with disks, we expect the lowest-level mapping of logical block numbers (*LBNs*) to physical locations to be straightforward and optimized for sequential access; this will be best for legacy systems that use these new devices as disk replacements. Such a sequentially optimized mapping scheme fits disk terminology and has some similar characteristics. Nonetheless, the physical differences will make data placement decisions (*i.e.*, mapping of file or database blocks to LBNs) an interesting topic. § 5 explores this area.

2.4.4 Seek time *vs.* seek distance

For disks, seek times are relatively constant functions of the seek distance, independent of the start cylinder and direction of seek. Because of the spring restoring forces, this is not true of MEMS-based storage devices. Short seeks near the edges take longer than they do near the center (as discussed in § 5). Also, turnarounds near the edges take either less time or more, depending on the direction of sled motion. As a result, seek-reducing request scheduling algorithms may not achieve their best performance if they look only at distances between LBNs as they can with disks [WGP94].

2.4.5 Recording density

MEMS-based storage devices use the same basic magnetic recording technologies as disks. Thus, the same types of fabrication and grown media defects can be expected. However, because of the much higher bit densities of MEMS-based storage devices, each such media defect will to affect a much larger number of bits. This is one of the fault management issues addressed in § 6.1.

2.4.6 Numbers of mechanical components

MEMS-based storage devices have many more distinct mechanical parts than disks. Although their very small movements make them more robust than the large disk mechanics, their numbers make it much more likely that some number of them will break. In fact, manufacturing yields may dictate that the devices operate with some number of broken mechanical components. § 6.1.1 discusses this issue.

2.4.7 Concurrent read/write heads

Because it is difficult and expensive for drive manufacturers to enable parallel activity, most modern disk drives use only one read/write head at a time for data access. Even drives that do support parallel activity are limited to only 2–20 read/write heads. On the other hand, MEMS-based storage devices (with their per-tip actuation and control components) could theoretically use all of their probe tips concurrently. Even after power and heat considerations, 100s to 1000s of simultaneously active probe tips is a realistic expectation. This parallelism increases media bandwidth and (as discussed in § 6.1.2) can improve reliability.

2.4.8 Control over mechanical movements

Unlike disks, which rotate at constant velocity independent of ongoing accesses, the mechanical movements of MEMS-based storage devices can be explicitly controlled. As a result, access patterns that suffer significantly from independent rotation can be better served. The best example of this is repeated access to the same block, as often occurs for synchronous metadata updates or read-modify-write sequences. This difference is explored further in § 6.2 and Table 2.

2.4.9 Startup activities

Like disks, MEMS-based storage devices will require some time to ready themselves for media accesses when powered up. Because of the size of their mechanical structures and the lack of rotation, however, the time and power required for startup will be much smaller than disks. The consequences of this fact for both availability (§ 6.3) and power management (§ 7) are explored in this paper.

sled mobility in X and Y	100 μm
bit cell width (area)	40 nm (0.0016 μm^2)
number of tips	6400
simultaneously active tips	1280
tip sector length	80 bits (8 data bytes)
servo overhead	10 bits per tip sector
device capacity (per sled)	3.2 GB
per-tip data rate	700 Kbit/s
sled acceleration	803.6 m/s^2
settling time constants	1
sled resonant frequency	739 Hz
spring factor	75%

Table 1: ***Device parameters used in our experiments.*** *Although MEMS-based storage devices have yet to be completely fabricated and tested, we believe these are reasonable values for initial analyses of these devices.*

2.4.10 Drive-side management

As with disks, management functionality will be split between host OSes and device OSes (firmware). Over the years, increasing amounts of functionality have shifted into disk OSes, enabling a variety of portability, reliability, mobility, performance, and scalability enhancements. We expect a similar trend with MEMS-based storage devices, whose silicon implementation allow direct integration of storage with computational logic.

2.4.11 Speed-matching buffers

As with disks, MEMS-based storage devices access the media as the sled moves past the probe tips at a fixed rate. Since this rate rarely matches that of the external interface, speed-matching buffers are important. Further, since sequential request streams are important aspects of many real systems, these speed-matching buffers will play an important role in prefetching of sequential LBNs. Also, as with disks, most block reuse will be captured by larger host memory caches instead of in the device cache.

2.4.12 Sectors per track

Disk media is organized as a series of concentric circles, with outer circles having longer circumferences than inner circles. This fact led disk manufacturers to use banded (zoned) recording in place of a constant bit-per-track scheme in order to increase density and bandwidth. This results in as much as a 46% difference between the maximum bandwidth at the innermost and outermost tracks [Qua99]. Because MEMS-based storage devices instead organize their media as parallel lines, there is no length difference in “bits-per-track” and banded recording is not relevant. Therefore, block layout techniques that try to exploit banded recording will not provide benefit for these devices. On the other hand, for block layouts that try to consider track boundaries and block offsets within tracks, this uniformity (which was common in disks 10 or more years ago) will simplify or enable correct implementations.

3 Experimental Setup

For our experiments, we use the performance model for MEMS-based storage described in [GSGN00], which includes all of the characteristics described earlier. Although it is not yet possible to validate the model against real devices, both the equations and the default parameters are the result of extensive discussions with groups that are designing and building MEMS-based storage devices. Thus, we hope that the model is sufficiently representative for the insights gained from experiments to be useful.

This performance model has been integrated into the DiskSim simulation environment [GWP98] as a disk-like storage device accessed via a SCSI-like protocol. Table 1 shows default parameters for our MEMS-based storage device simulator. DiskSim provides an infrastructure for exercising the device model with various synthetic and trace-based workloads. DiskSim also includes a detailed, validated disk module that can be parameterized to accurately model a variety of real disks. For reference, some experiments use DiskSim's disk module configured to model the Quantum *Atlas 10K* [Qua99], one of the disks for which publicly available configuration parameters [Dis00] have been calibrated against real-world drives.

Most of the experiments use a synthetically-generated workload that we refer to as the *random* workload. For this workload, request interarrival times are drawn from an exponential distribution; the mean is generally varied to provide a range of workloads. All other aspects of requests are independent: 67% are reads, 33% are writes, the request size distribution is exponential with a mean of 4 KB, and request starting locations are uniformly distributed across the device's capacity. To include more realistic workloads, traces of real storage activity are utilized in some experiments; they are described in the appropriate sections.

4 Request Scheduling

An important mechanism for improving disk efficiency is deliberate scheduling of pending requests. This is important to efficiency because positioning delays are dependent on the relative positions of the read/write head and the destination sector. The same is true of MEMS-based storage devices, whose seek times are dependent on the distance to be travel-led. This section explores the impact of different scheduling algorithms on the performance of MEMS-based storage devices.

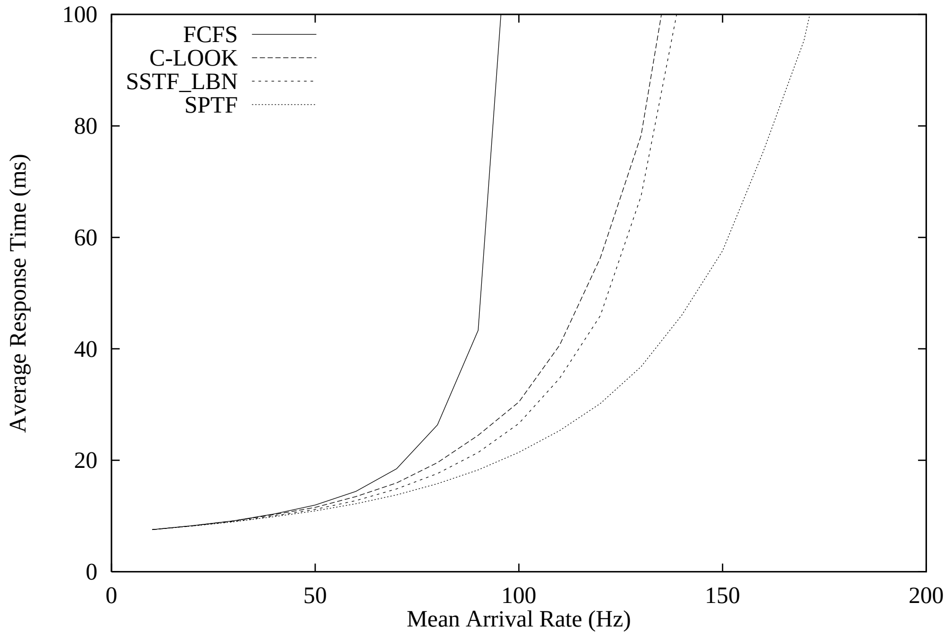
4.1 Disk Scheduling Algorithms

Many disk scheduling algorithms have been devised and studied over the years. Our comparisons focus on four. The simple *First Come First Served* (FCFS) algorithm often results in suboptimal performance, but we include it for reference. The *Shortest Seek Time First* (SSTF) algorithm was designed to select the request that will incur the smallest seek delay [Den67], but this is rarely the way it functions in practice. Instead, since few host OSes have the information needed to compute actual seek distances or predict seek times, most SSTF implementations use the difference between the last accessed LBN and the desired LBN as an approximation of seek time. This simplification works well for disk drives [WGP94], and we label this algorithm as "SSTF_LBN". The *Cyclical LOOK* (C-LOOK) algorithm [SLW66] services requests in ascending LBN order, starting over with the lowest LBN when all requests are "behind" the most recent request. The *Shortest Positioning Time First* (SPTF) policy selects the request that will incur the smallest positioning delay [SCO90, JW91]. For disks, this algorithm differs from others in that it explicitly considers both seek time and rotational latency.

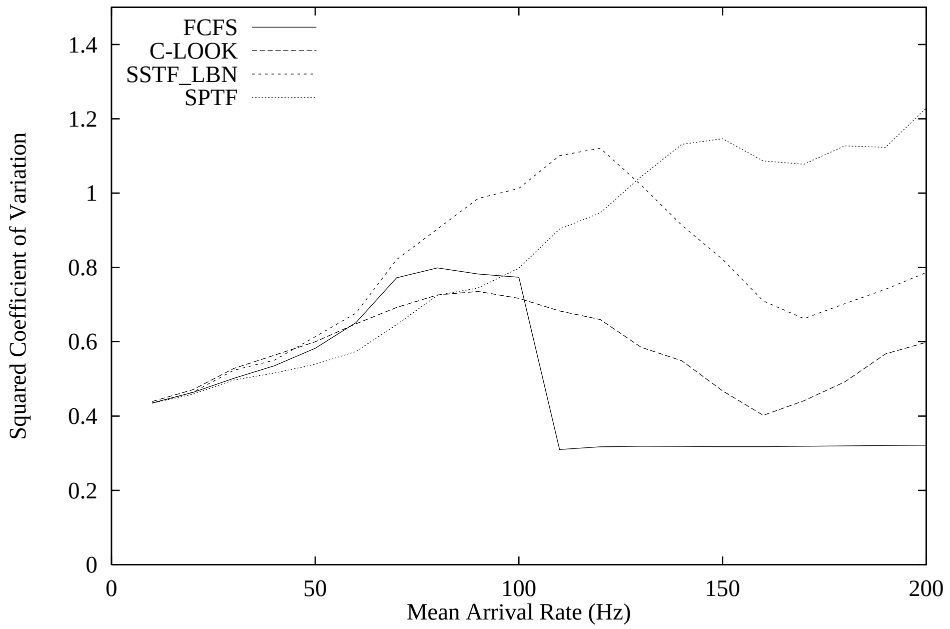
For reference, Fig. 5 compares these four disk scheduling algorithms for the Atlas 10k disk drive and the *random* workload (§ 3) with a range of request arrival rates. Two common metrics for evaluating disk scheduling algorithms are shown. First, the average response time (queue time plus service time) shows the effect on average case performance. As expected, FCFS saturates well before the other algorithms as the workload increases. SSTF_LBN outperforms C-LOOK, and SPTF outperforms all other schemes. Second, the squared coefficient of variation (σ^2/μ^2) is the metric of "fairness" (or starvation resistance) used in [TP72, WGP94]; lower values indicate better starvation resistance. As expected, C-LOOK avoids the starvation effects that characterize the SSTF_LBN and SPTF algorithms.

4.2 MEMS-based Storage Device Scheduling

Existing disk scheduling algorithms can be adapted to MEMS-based storage devices, once these devices are mapped into a disk-like interface. Most, including FCFS, SSTF_LBN, and C-LOOK, only use knowledge of LBNs and assume that differences between LBNs are reasonable approximations of positioning times. SPTF, which addresses disk seeks and rotations, is a more interesting case. While MEMS-based storage devices do not have a rotational latency component, they do have two positioning time components: the X dimension seek and the Y dimension seek. As with disks, only one of these (seek time for disks; the X dimension seek for MEMS-based storage devices) is approximated well by a linear LBN space. Unlike disks, the two positioning components proceed in parallel, with the greater hiding the lesser. The settling time delay makes most X



(a) Average response time (ms)



(b) Squared coefficients of variation (σ^2/μ^2)

Figure 5: ***Comparison of scheduling algorithms for the random workload on the Atlas 10K disk (§ 4.1).***

dimension seek times larger than most Y dimension seek times. SPTF will only outperform SSTF (which minimizes X movements, but ignores Y) when the Y component is the larger.

Fig. 6 shows how well these algorithms work for the default MEMS-based storage device on the *random* workload with a range of request arrival rates. In terms of both performance and starvation resistance, the algorithms finish in the same order as for disks – SPTF provides the best performance and C-LOOK provides the best starvation resistance. However, their performance relative to each other merits discussion. For example, the difference between FCFS and the LBN-based algorithms (C-LOOK and SSTF_LBN) is larger for MEMS-based storage devices, because the seek time is a much larger component of the total service time. In particular, there is no subsequent rotational delay. Also, the average response time difference between C-LOOK and SSTF_LBN is smaller for MEMS-based storage devices, because both algorithms reduce the X seek times into the range where X and Y seek times are comparable. Since neither addresses Y seeks, the greediness of SSTF_LBN is less effective. SPTF, which does address Y seeks, obtains additional performance.

4.3 Traces of Disk Activity

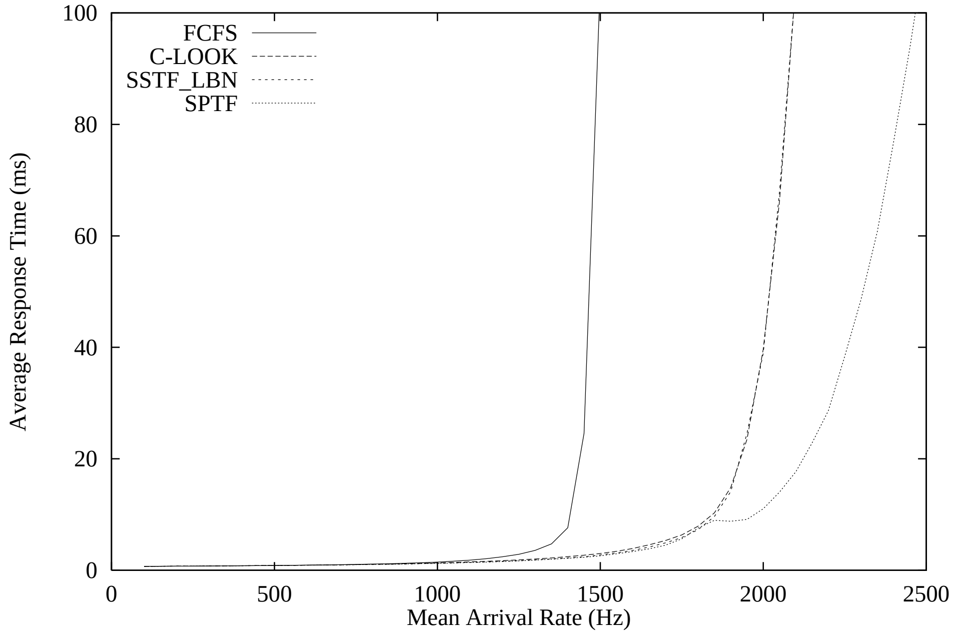
To evaluate performance and scheduling of MEMS-based storage devices under more realistic workloads, we use two traces of real disk activity. The *TPC-C* trace comes from a TPC-C testbed, consisting of Microsoft SQL Server atop Windows NT[2]. The hardware was a 300 MHz Pentium II system with 128 MB of memory and a 1 GB test database striped across two Quantum Viking disk drives. The trace captures one hour of disk activity for TPC-C, and its characteristics are described in more detail in [RFGN00]. The *Cello* trace comes from a Hewlett-Packard system running the HP-UX^{TM} operating system. It captures disk activity from a server at HP Labs used for program development, simulation, mail, and news. While the total trace is actually two months in length, we report data for a single week-long snapshot (5/30/92 to 6/6/92). This trace and its characteristics are described in detail in [RW93].

Figs. 7(a) and 7(b) show how the scheduling algorithms perform for the *Cello* and *TPC-C* workloads, respectively. The relative performance of the algorithms on the Cello trace is very similar to the *random* workload. One noteworthy difference between TPC-C and Cello is that SPTF outperforms the other algorithms by a much larger margin for *TPC-C*. This occurs because the scaled-up version of the workload includes many concurrently-pending requests with very small inter-LBN distances. LBN-based schemes do not have enough information to choose between such requests, often causing small (but expensive) X-dimension seeks. SPTF addresses this problems and thus performs much better.

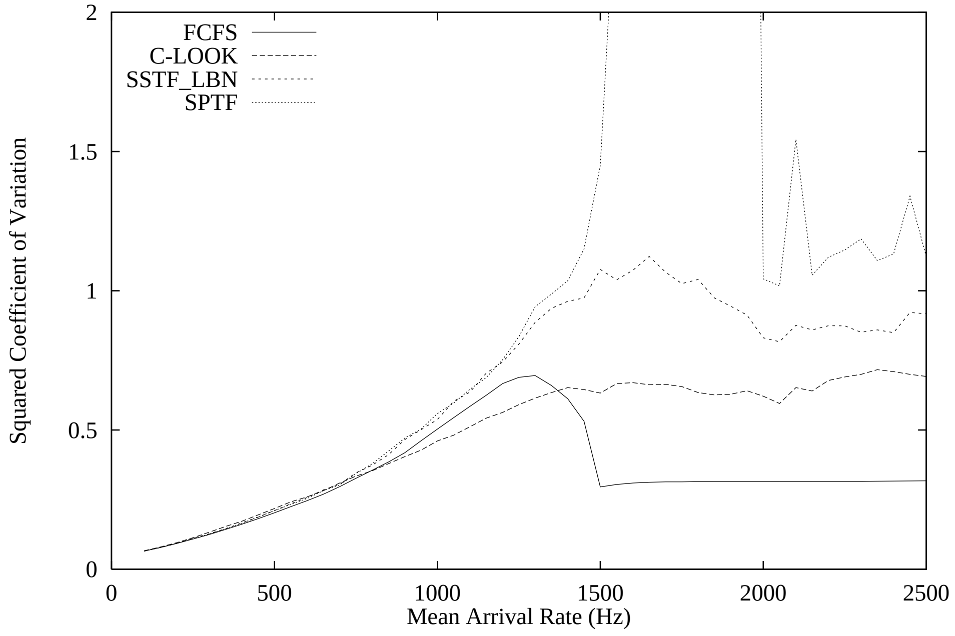
4.4 Interaction of SPTF & Settling Times

Originally, we had expected SPTF to outperform the other algorithms by a greater margin for MEMS-based storage devices. Our investigations suggest that the value of SPTF scheduling is highly dependent upon the settling time component of X dimension seeks. With large settling times, X dimension seek times dominate Y dimension seek times, making SSTF_LBN closely approximate SPTF. With small settling times, Y dimension seek times are a more significant component. To illustrate this, Fig. 8 compares the scheduling algorithms with the number of settling time constants set to 0 and 2 (recall that the default is 1). As expected, with 2 settling time constants, SSTF_LBN is very close to SPTF. With zero settling time constants, which may be achievable with active damping control systems, SPTF outperforms the other algorithms by a large margin.

[2]Re-using traces collected from other systems presents two main difficulties. First, the capacity of the disks in the traced systems is smaller than that of the storage devices simulated herein. As a result, not all of our simulated devices' capacities are utilized by these traces, which tends to reduce the maximum mechanical positioning delays. The second and more difficult issue is that our simulated devices are newer and significantly faster than the disks used in the traced systems. Ideally, the appropriate feedback effects between request completions and subsequent arrivals would be included in the simulation. Unfortunately, the necessary information is not present in the traces. Instead, we replicate an approach used in previous disk scheduling work for dealing with this problem [WGP94]: we scale the traced inter-arrival times to produce a range of average inter-arrival times. When the scale factor is one, the simulated inter-arrival times match those traced. When the scale factor is two, the traced inter-arrival times are halved, doubling the average arrival rate. While imperfect, we believe that this approach to dealing with this common problem of trace-driven storage simulations yields valid qualitative results and insights.

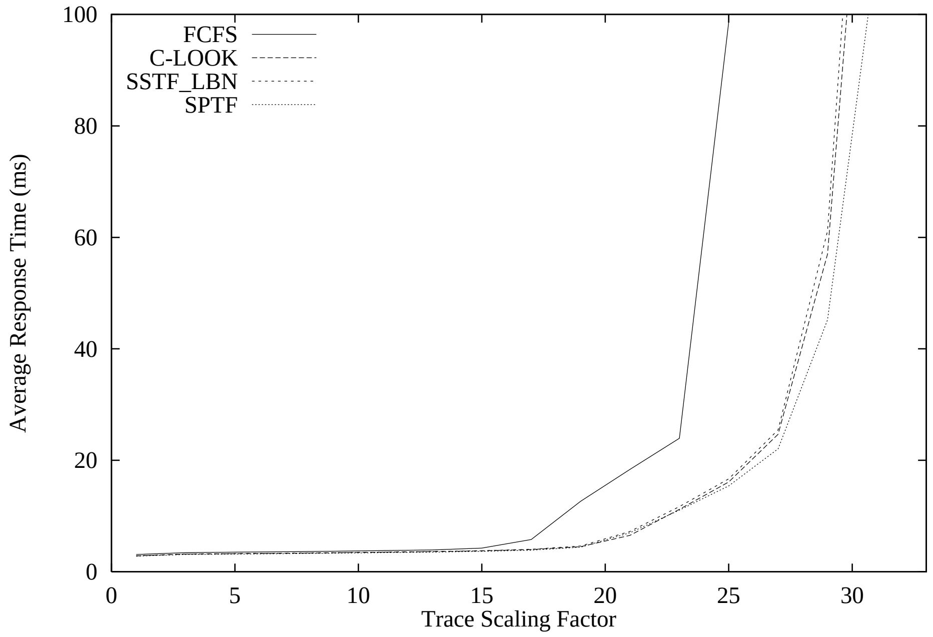


(a) Average response time (ms)

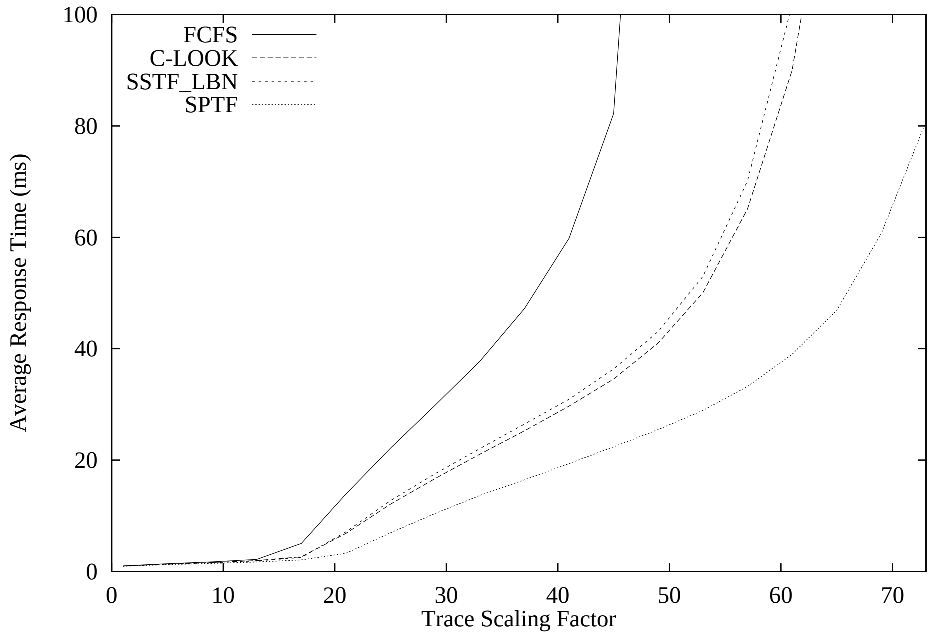


(b) Squared coefficients of variation (σ^2/μ^2)

Figure 6: ***Comparison of scheduling algorithms for the random workload on the MEMS-based storage device (§ 4.2).*** *We are not yet able to explain the odd behavior of SPTF between 1500 and 2000 requests/sec.*

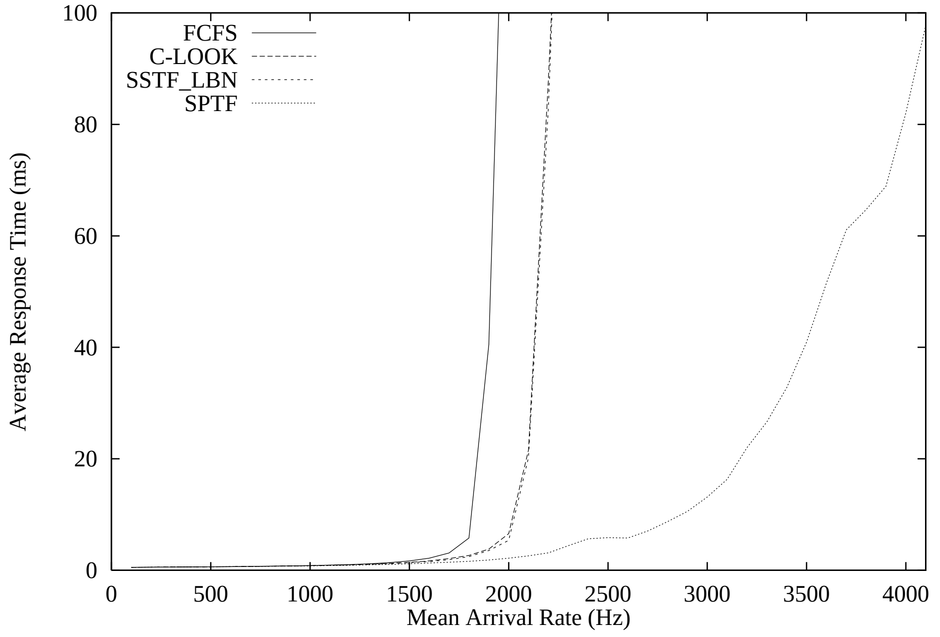


(a) *Cello* average response time

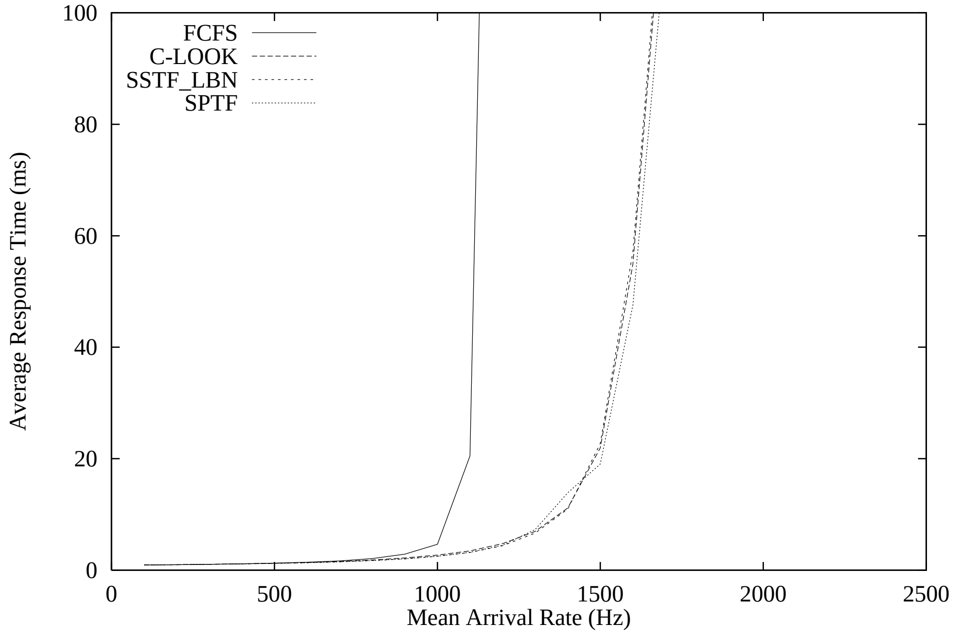


(b) *TPC-C* average response time

Figure 7: ***Comparison of scheduling algorithms for the* Cello *and* TPC-C *workloads on the MEMS-based storage device* (§ 4.3).**



(a) *Random (zero time constants)* average response time



(b) *Random (two time constants)* average response time

Figure 8: ***Comparison of average performance for zero and two settling time constants, respectively* (§ 4.4).** *These are in comparison to the default model (*Random *with one time constant) shown in Fig. 6(a).*

0.521 $\langle -800, 800\rangle$ *0.339*	0.490 $\langle -400, 800\rangle$ *0.319*	0.483 $\langle 0, 800\rangle$ *0.314*	0.490 $\langle 400, 800\rangle$ *0.319*	0.521 $\langle 800, 800\rangle$ *0.339*
0.508 $\langle -800, 400\rangle$ *0.313*	0.478 $\langle -400, 400\rangle$ *0.290*	0.470 $\langle 0, 400\rangle$ *0.284*	0.478 $\langle 400, 400\rangle$ *0.290*	0.508 $\langle 800, 400\rangle$ *0.313*
0.506 $\langle -800, 0\rangle$ *0.309*	0.476 $\langle -400, 0\rangle$ *0.285*	0.468 $\langle 0, 0\rangle$ *0.279*	0.476 $\langle 400, 0\rangle$ *0.285*	0.506 $\langle 800, 0\rangle$ *0.309*
0.508 $\langle -800, -400\rangle$ *0.313*	0.478 $\langle -400, -400\rangle$ *0.290*	0.470 $\langle 0, -400\rangle$ *0.284*	0.478 $\langle 400, -400\rangle$ *0.290*	0.508 $\langle 800, -400\rangle$ *0.313*
0.521 $\langle -800, -800\rangle$ *0.339*	0.490 $\langle -400, -800\rangle$ *0.319*	0.483 $\langle 0, -800\rangle$ *0.314*	0.490 $\langle 400, -800\rangle$ *0.319*	0.521 $\langle 800, -800\rangle$ *0.339*

Figure 9: ***Difference in request service time for subregion accesses* (§ 5.1).** *This figure divides the area accessible by an individual probe tip into 25 subregions, each 400×400 square bits centered at the tuple $\langle x,y\rangle$, where $\langle x,y\rangle$ represents the sled offset at the center of each subregion. Each box shows the average request service time (in milliseconds) for 10,000 requests starting and ending inside that subregion. The upper numbers represent the service time when X settle time is included in calculations; numbers in italics represent the service time for zero X settle time. Note the average service time differs by 10–20% between the centermost and outermost subregions.*

5 On-Device Data Layout

Space allocation and data placement for disks continues to be a ripe topic of research. We expect the same to be true of MEMS-based storage devices. In this section, we discuss how the characteristics of MEMS-based storage positioning costs affect placement decisions for small local and large sequential transfers. A bipartite layout is proposed and shown to have potential for improving performance.

5.1 Small, skewed accesses

As with disks, short distance seeks are faster than long distance seeks. Unlike disks, MEMS-based storage devices' spring forces change the effective actuator force and therefore affect the sled positioning time. Fig. 9 shows the impact of springs forces for seeks inside different "subregions" of a single tip's media region. The spring forces increase with increasing sled displacement from the origin (*e.g.*, the outermost boxes in Fig. 9.) As a result, distance is not the only component to be considered when finding good placements for small, popular data items—offset relative to the center should also be considered.

5.2 Large, sequential transfers

Streaming media transfer rates for MEMS-based storage devices and disks are similar: 28.5–19.5 MB/s for the Atlas 10K [Qua99]; 79.6 MB/s for MEMS-based storage. Positioning times, however, are very different—MEMS devices enjoy an order of magnitude shorter positioning times. This makes positioning time relatively insignificant for large transfers (*e.g.*, hundreds of sectors). Fig. 10 shows the request service times for a 256 KB read with respect to the X distance between the initial and final sled positions. Requests traveling 1000 cylinders (*e.g.*, from the sled origin to maximum sled displacement) only incur a 10% penalty. This lessens the importance of ensuring locality for data that will be accessed in large, sequential chunks. In contrast, seek distance is a significant issue with disks, where long seeks more than double the total service time for 256 KB requests.

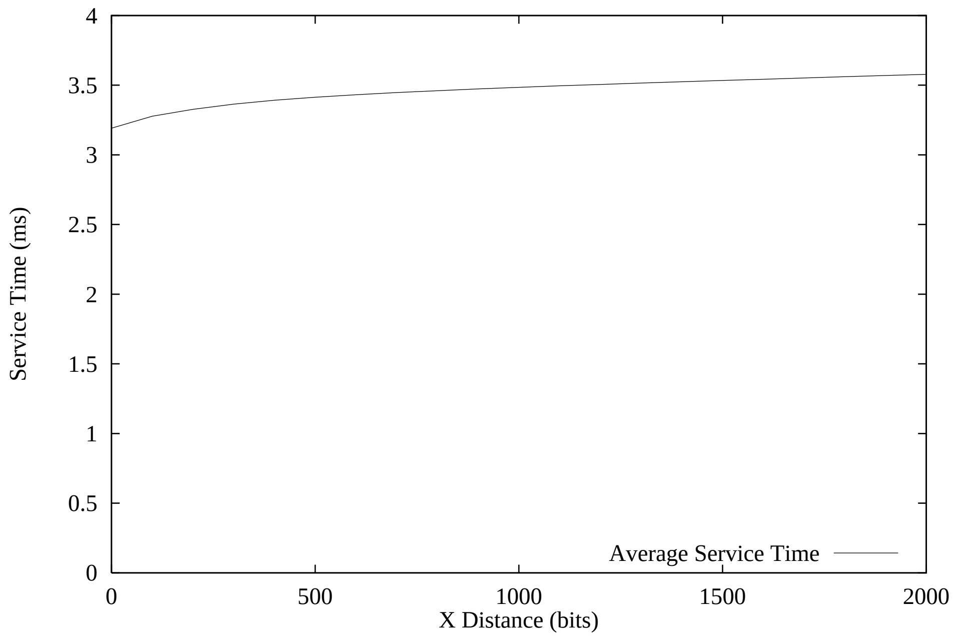


Figure 10: ***Request service time* vs. *X seek distance for large (256 KB) requests*** **(§ 5.2).** *Note that large X seeks only increase the service time by 12%.*

5.3 A data placement scheme for MEMS-based storage devices

To take advantage of the above characteristics, we implemented a 25-subregion bipartite layout scheme. Small data are placed in the centermost subregion; long, sequential streaming data are placed in the outermost subregion. Two layouts are tested: a five-by-five grid of subregions (shown in Fig. 9) and a simple "columnar" division of the LBN space into 25 columns (*e.g.*, each subregion contains 100 contiguous cylinders).

We compare these layout schemes against the "organ pipe" layout [VC90, RW91], an optimal disk-layout scheme. In the organ pipe layout, the most frequently accessed blocks are placed in the center of the disk. Blocks of decreasing popularity are distributed to either side of center, with the least frequently accessed blocks located the farthest from the center on both sides. Although this scheme is optimal for disks, blocks must be periodically shuffled to maintain the frequency distribution. Further, the layout requires some state to be kept indicating each block's popularity and interdependence.

To evaluate these layouts we created a workload of 10,000 read requests, 89% "small" (4 KB) requests and the remainder "large" (400 KB) requests. For the subregioned layouts, the large requests were directed to the ten leftmost and ten rightmost subregions, while the small requests mapped to the centermost subregion. In the organ pipe layout, we created a distribution of one large request for every eight small requests.

Our results (Fig. 11) show that all three layout schemes achieve a 13–20% improvement in average access time over a simple linear layout. (In comparison, the Atlas 10K disk achieves a 13% performance gain between the organ pipe and simple layouts.) Subregioned and columnar layouts both provide a 10–15% improvement over organ pipe. Further, the two layouts do not incur organ pipe's overhead of keeping popularity and interdependency data or periodically reshuffling blocks on the media. For the "no settle" case, the subregioned layout provides the best performance as it optimizes both X and Y.

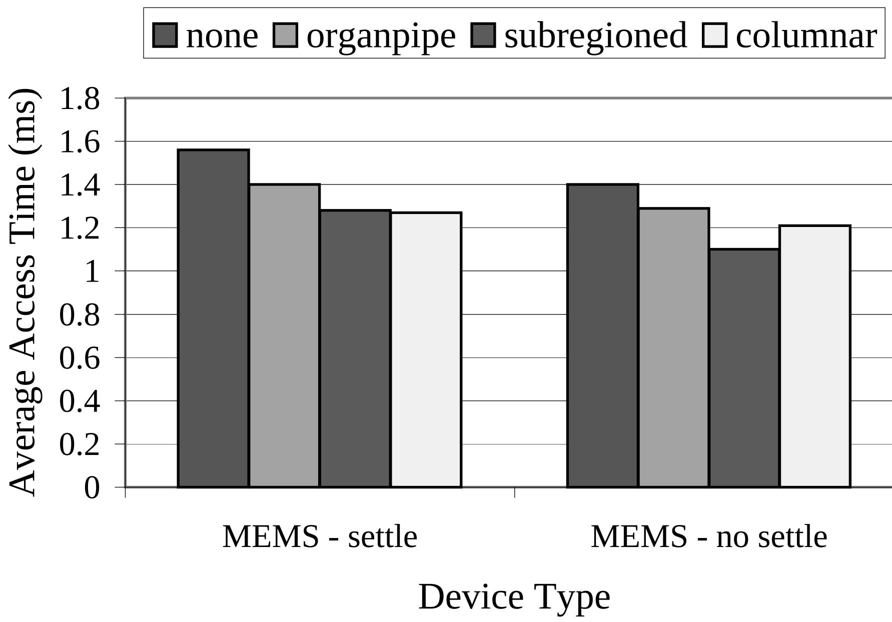


Figure 11: ***A comparison of various layout schemes for MEMS-based storage devices* (§ 5.3).** *The "MEMS-nosettle" graph shows the same experiment run with zero X settling time. For the default device, the organ pipe, subregioned and columnar layouts achieve a 13–20% performance improvement over the simple layout. It is interesting to note that an optimal disk layout technique does not provide the best performance for MEMS-based storage. Further, for the "no settle" case, the subregioned layout outperforms the others by an additional 20%.*

6 Failure Management

Fault tolerance and recoverability are significant considerations for storage systems. In many ways, fault management for MEMS-based storage devices will be similar to fault management for conventional disks[3]. Although there will likely be more defective or failed parts in MEMS-based storage (because of the large number of distinct parts compared to disks and the fact that bad parts cannot be replaced before assembly), individual component failures can be made less likely to render a device inoperable than in disks. This section discusses three aspects of failure management: internal faults, device failures, and recoverability from system crashes.

6.1 Internal faults

The common failure modes for disk drives include recoverable failures (*e.g.*, seek errors, media defects, bit errors, lost sectors) and non-recoverable failures (*e.g.*, head crashes, motor or arm actuator failure, drive electronics or channel failure). MEMS-based storage devices have similar failure modes with analogous mechanical causes; however, the ability to incorporate multiple tips into failure avoidance schemes allows MEMS-based storage devices to employ more internal redundancy and improved fault tolerance.

6.1.1 Tip and media failure

For disk drives, unrecoverable media defects are handled by re-mapping logical block numbers to non-defective locations, with data often being lost when defects "grow" during operation. In MEMS-based storage, each sector is striped across many tips, so localized media or single tip defects (the common occurrences) can be completely recovered with error correction codes (see § 6.1.2). In addition, striping could significantly reduce

[3]Although MEMS-based storage devices don't exist yet, MEMS components have been built and tested for many years. Results show that isomorphically scaling an object alters the relative influence of various physical effects, significantly improving the relative strength of smaller objects. This effect has been shown to make MEMS-based storage components less fragile than their disk counterparts [Mad97].

layout scrambling from defect management. For example, instead of "slipping" LBNs over defective sectors or re-mapping them to spare sectors elsewhere in a cylinder or zone, as is done in some disks, defective sectors in MEMS-based storage could be re-mapped to the *same tip sector* on one of several dedicated "spare tips." Re-mapping to the same tip sector guarantees that a re-mapped sector can be accessed at the same time as the original (now damaged) sector. This eliminates the performance penalty incurred when re-mappied disk sectors break the physical sequentiality of access; it also improves the predictability of storage accesse times.

Failure of a conventional disk's read/write head or control logic generally renders the entire device inoperable. MEMS-based storage replicates these functions across thousands of components. With so many components, failure of one or more is not only possible, but probable. Individual probe tips can break off or "crash" into the media; fabrication variances will produce faulty tips or tip-specific logic. Fortunately, many problems can be handled using the same mechanisms that handle media failures. Striping and ECC can overcome the loss of an entire tip region without any loss of data or capacity. This yields an interesting trade-off between capacity and fault tolerance—on tip failure, the operating system can choose to sacrifice device capacity (by converting regular tips into spare tips) or sacrifice fault tolerance in that tip region (by converting spare tips into regular tips).

6.1.2 Read and write errors

MEMS-based storage devices read or write data striped across multiple tips. For example, each 512 B sector is striped across 64 tips. Unfortunately, with increased tip parallelism comes increased opportunity for one or more tips to suffer a read or write error. As with conventional disks, powerful error correction codes will correct minor recording or sensing errors. These codes may be encoded both horizontally (by switching on extra ECC tips during each access) and vertically (by using an N-bit-per-byte encoding under each tip). The horizontal ECC is useful for recovery from missing tip sectors. The vertical portion of the ECC can identify tip-sectors that should be treated as missing (*i.e.*, converting large errors into erasures).

In addition, MEMS-based storage devices are much faster at handling errors that require a second pass over the media. In a disk, re-reading a sector suffers an entire rotational latency penalty. In MEMS-based storage, the sled need only turn around (see Table 2).

6.1.3 Seek errors

To read or write a sector, disks first seek to the associated track and then read the servo bursts, verifying that the head is over the correct track and computing the rotational latency before the desired sector passes under the head. The penalty for a seek error is composed of the new tracking time (about 1–2 ms for short re-seeks) and up to the entire rotational latency (6 ms for 10,000 RPM disks) for the sector to pass under the head again.

MEMS-based storage devices also contain tracking information stored in servo bursts; this information is duplicated across all tips and is read and verified by every tip involved in a data access. The penalty for a seek error could involve up to two turnarounds in the Y direction (0.04–1.11 ms each) and short seeks in possibly both the X and Y directions.

6.2 Device failures

MEMS-based storage devices are susceptible to similar non-recoverable failures as disk drives: strong external mechanical or electrostatic forces could damage the actuator comb fingers or induce spring failure, manufacturing defects could surface, and the device electronics or channel could fail. These failures should look like and be handled in the same manner as disks. Inter-device redundancy and periodic backups are appropriate mechanisms for dealing with such problems.

Interestingly, MEMS-based storage's mechanical characteristics are a better match than disks for the common read-modify-write operations used in some fault-tolerant schemes (*e.g.*, RAID-5). While conventional disks suffer a full rotation to return to the same sector, MEMS-based storage devices can quickly turn around, significantly reducing the read-modify-write latency (as shown in Table 2). The small incremental cost for returning to the same sector obviates the need for the many optimizations [MC93, SGH93, Men95] that have been developed to address this problem.

	Atlas 10K		MEMS	
# sectors	8	334	8	334
read	0.14	6.00	0.13	2.19
reposition	5.98	0.00	0.07	0.07
write	0.14	6.00	0.13	2.19
total (ms)	6.26	12.00	0.33	4.45

Table 2: ***A comparison of read-modify-write times for 4 KB (8 sector) and track-length (334 sector) transfers.*** *334 sectors is the longest track length in the Atlas 10K disk. Conventional disks must wait for a complete platter rotation during read-modify-write operations, whereas MEMS-based storage devices need only turn the sled around. (Depending on sled position and the spring factor, turnaround time varies nonlinearly from 0.036 ms–1.11 ms with 0.063 ms average.) This characteristic is particularly helpful for code-based redundancy schemes (*e.g., *RAID-5; see* § 6.2*) or for verify-after-write operations.*

6.3 Recovery from host system crashes

As they do with disks, file systems and databases must maintain internal consistency between persistent objects stored on MEMS-based storage devices [CMB+81, Hag87, GR93, GP94]. Although synchronous writes will still not be desirable, the much lower service times for MEMS-based storage devices should decrease the penalty for these writes [LC97, WPA99].

Another benefit is the rapid initialization (0.5 ms) of MEMS-based storage devices. No spindle spin-up time is required, so initialization is almost immediate. In contrast, high-end disk drives can take 25 seconds to spin-up [Qua99]. Further, MEMS-based storage devices do not exhibit the power surge inherent to spinning up disk drives, so power spike avoidance techniques (*e.g.*, serializing the spin-up of multiple disk drives) are unnecessary—all of the devices may be initialized concurrently.

7 Power Management

Significant effort has gone into reducing a disk drive's power consumption, including (1) reducing active power consumption and (2) introducing numerous power-saving modes employed during idle times [DKM94, LKHA94, LSD99]. Supporting these power-saving modes requires OS power management software that controls power mode transitions as varying levels of electronics and the spindle motor are powered down. Since restarting components can significantly increase access time (ranging from 40 ms to over 2 seconds when restarting the spindle motor [IBM99, IBM00]), power management software must constantly make trade-offs between reducing power and increasing access time.

The power characteristics of MEMS-based storage devices enable a much simpler OS power management scheme: a single idle mode that stops the sled and powers down non-essential electronics. With no rotating parts and a very light mass, the sled's restart time is very small (estimated at under 0.5 ms). This imperceptible penalty enables aggressive idle mode use, switching from active to idle as soon as the I/O queue is empty.

Further, 90% of a MEMS-based storage device's power is used for sensing and recording operations, making the media sled's power consumption negligible. This results in a flat power-per-bit-accessed consumption rate and creates another set of power optimizations: minimizing the amount of data transferred [NSN+97, FS99]. In a manner similar to power optimizations for wireless communication (where aggressive compression can significantly save power), the embedded computational logic in MEMS-based storage devices could be used to compress data arriving at the media in order to minimize the number of active tips per access.

8 Conclusion

Our work compares and contrasts MEMS-based storage devices with disk drives and provides a foundation for focused operating system management of these new devices. We describe and evaluate approaches for operating system tuning of request scheduling, data placement, failure management, and power management techniques in order to match the physical characteristics of MEMS-based storage.

For scheduling decisions, we find adaptations of disk scheduling algorithms to be appropriate for MEMS-based storage devices. The impact of settling time on sled seek time is the key consideration when choosing among these algorithms. For large settling times (where X direction seeks will generally dominate Y direction seeks), LBN-based algorithms that minimize sled movement in the X direction (*e.g.*, SSTF_LBN, C-LOOK) achieve good performance without the overhead of calculating the exact positioning times for each outstanding request (SPTF). Layout decisions at the OS level also depend on the physical characteristics of the device. For devices with large spring factors, we find that small, random requests are optimally confined to the centermost subregion whereas large, sequential requests may be placed anywhere on the media with minimal (<10%) penalty. This encourages a bipartite layout scheme; our experiments suggest such a layout yields up to a 20% improvement in request service times over a simple linear layout.

The characteristics of MEMS-based storage devices also impact failure and power management at the OS level. The large amount of internal parallelism among probe tips allows faulty tip regions to be remapped to spare probe tips with no negative impact on request service time. Because of the nature of the sled motion, read-modify-write requests are handled with nearly zero repositioning overhead—this has a strong positive impact on code-based redundancy schemes such as RAID-5. The small initialization/startup time of these devices (∼0.5 ms) allows both fast recovery from host crashes and fine-grain idle power management by the OS. Finally, because power consumption is a near-linear function of the number of active tips, the OS can manage device power dissipation by controlling both request size and the maximum number of active tips.

Continuing this work, we are exploring the impact MEMS-based storage devices will have on the structure of computer systems and the memory hierarchy [SGNG00] and investigating applications that directly benefit from the unique characteristics of these devices.

References

[CBF$^+$00] L. Richard Carley, James A. Bain, Gary K. Fedder, David W. Greve, David F. Guillou, Michael S.-C. Lu, Tamal Mukherjee, Suresh Santhanam, Leon Abelmann, and Seungook Min. Single chip computers with MEMS-based magnetic memory. *Journal of Applied Physics*, 87(to appear), 2000.

[CMB$^+$81] Donald D. Chamberlin, Astrahan M. Morton, Michael W. Blasgen, James N. Gray, W. Frank King, Bruce G. Lindsay, Raymond Lorie, James W. Mehl, Thomas G. Price, Franco Putzolo, Patricia Griffiths Selinger, Mario Schkolnick, Donlad R. Slutz, Irving L. Traiger, Bradford W. Wade, and Robert A. Yost. A history and evaluation of System R. *Communications of the ACM.*, 24(10):632–646, October 1981.

[DCK$^+$94] Fred Douglis, Ramon Caceres, Frans Kaashoek, Kai Li, Brian Marsh, and Joshua Tauber. Storage alternatives for mobile computers. In *Proceedings of the First Symposium on Operating Systems Design and Implementation*, pages 25–37, Monterey, CA, November 1994. USENIX Assoc.

[Den67] Peter Denning. Effects of scheduling on file memory operations. In *IFIPS Spring Joint Computer Conference*, pages 9–21, April 1967.

[Dis00] Database of validated disk parameters for DiskSim, February 2000. http://www.ece.cmu.edu/-˜ganger/disksim/diskspecs.html.

[DKM94] Fred Douglis, P. Krishnan, and Brian Marsh. Thwarting the power-hungry disk. In *Proceedings of the Winter USENIX Technical Conference*, pages 292–306, San Francisco, CA, January 1994. USENIX Association, Berkeley, CA.

[FS99] Jason Flinn and M. Satyanarayanan. Energy-aware adaptation for mobile applications. In *Proceedings of the 17th ACM Symposium on Operating System Principles*, Kiawah Island Resort, Charleston, SC, December 1999. Published as *Operating Systems Review*, **33**(5):48–63.

[FSR$^+$96] Gary K. Fedder, Suresh Santhanam, Michael L. Reed, Steven C. Eagle, David F. Guillou, Michael S.-C. Lu, and L. R. Carley. Laminated high-aspect-ratio microstructures in a conventional CMOS process. In *Proceedings of the IEEE Micro Electro Mechanical Systems Workshop*, pages 13–18, San Diego, CA, February 1996.

[GP94] Gregory R. Ganger and Yale N. Patt. Metadata update performance in file systems. In *Proceedings of the First Symposium on Operating Systems Design and Implementation*, pages 49–60, November 1994.

[GR93] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques.* Morgan Kaufmann Publishers, San Mateo, CA, 1993. ISBN 1-55860-190-2.

[GSGN00] John Linwood Griffin, Steven W. Schlosser, Gregory R. Ganger, and David F. Nagle. Modeling and performance of MEMS-based storage devices. In *Proceedings of the 2000 ACM SIGMETRICS Conference*, Santa Clara, CA, June 2000.

[GWP98] Gregory R. Ganger, Bruce L. Worthington, and Yale N. Patt. The DiskSim simulation environment version 1.0 reference manual. Technical Report CSE-TR-358-98, The University of Michigan, Ann Arbor, February 1998.

[Hag87] Robert Hagmann. Reimplementing the Cedar file system using logging and group commit. In *Proceedings of the 11th ACM Symposium on Operating System Principles*, Austin, Texas, November 1987. Published as *Operating Systems Review*, **21**(5):155–162.

[IBM99] IBM. IBM family of microdrives, June 1999. http://www.storage.ibm.com/hardsoft/diskdrdl/-micro/datasheet.pdf.

[IBM00] IBM. IBM Travelstar 32GH, 30GT, and 20GN 2.5-inch hard disk drives, April 2000. http://www.storage.ibm.com/hardsoft/diskdrdl/travel/32ghdata.pdf.

[JW91] David M. Jacobson and John Wilkes. Disk scheduling algorithms based on rotational position. Technical Report HPL-CSP-91-7, Hewlett-Packard Laboratories, February 1991.

[KL99] Han-joon Kim and Sang-goo Lee. A new flash memory management for flash storage system. In *Proceedings of the 23rd Annual International Computer Software and Applications Conference*, October 1999.

[LC97] David E. Lowell and Peter M. Chen. Free transactions with Rio Vista. In *Proceedings of the 16th ACM Symposium on Operating System Principles*, Saint Malo, France, October 1997. Published as *Operating Systems Review*, **31**(5):92–101.

[LKHA94] Kester Li, Roger Kumpf, Paul Horton, and Thomas Anderson. A quantitative analysis of disk drive power management in portable computers. In *Proceedings of the Winter USENIX Technical Conference*, pages 279–292, January 1994.

[LSD99] Yung-Hsiang Lu, Tajana Simunic, and Giovanni De Micheli. Software controlled power management. In *7th International Workshop on Hardware/Software Codesign*, pages 157–161, May 1999.

[Mad97] Marc Madou. *Fundamentals of Microfabrication.* CRC Press, Boca Raton, Fla., 1997. ISBN 0-8493-9451-1.

[MC93] Jai Menon and Jim Courtney. The architecture of a fault-tolerant cached RAID controller. In *ACM International Symposium on Computer Architecture*, pages 76–86, San Diego, CA, May 1993.

[Men95] Jai Menon. A performance comparison of raid-5 and log-structured arrays. In *IEEE International Symposium on High-Performance Distributed Computing*, pages 167–178, Washington, DC, August 1995. IEEE Computer Society Press, Washington, DC.

[MJLF84] Marshall K. McKusick, William N. Joy, Samuel J. Leffler, and Robert S. Fabry. A fast file system for UNIX. *ACM Transactions on Computer Systems*, 2(3):181–197, August 1984.

[MK91] Larry W. McVoy and Steve R. Kleiman. Extent-like performance from a UNIX file system. In *Proceedings of the Winter USENIX Technical Conference*, pages 33–43, January 1991.

[NSN$^+$97] Brian D. Noble, M. Satyanarayanan, Dushyanth Narayanan, James Eric Tilton, Jason Flinn, and Kevin R. Walker. Agile application-aware adaptation for mobility. In *Proceedings of the 16th ACM Symposium on Operating System Principles*, Saint Malo, France, December 1997. Published as *Operating Systems Review*, **31**(5):276–289.

[Qua99] Quantum Corporation. *Quantum Atlas 10K 9.1/18.2/36.4 GB Ultra 160/m S Product Manual III SCSI Hard Disk Drives: Ultra SE SCSI-3 Version*, August 1999.

[RFGN00] Eric Riedel, Christos Faloutsos, Gregory R. Ganger, and David F. Nagle. Data mining on an OLTP system (nearly) for free. In *Proceedings of the ACM SIGMOD Conference*, page to appear, Dallas, Texas, May 2000.

[RO92] Mendel Rosenblum and John K. Ousterhout. The design and implementation of a log-structured file system. *ACM Transactions on Computer Systems*, 10(1):26–52, February 1992.

[RW91] Chris Ruemmler and John Wilkes. Disk shuffling. Technical Report HPL-91-156, Hewlett Packard, October 1991.

[RW93] Chris Ruemmler and John Wilkes. UNIX disk access patterns. In *Proceedings of the Winter USENIX Conference*, pages 405–420, January 1993.

[SCO90] Margo Seltzer, Peter Chen, and John Ousterhout. Disk scheduling revisited. In *Proceedings of the Winter USENIX Conference*, pages 313–324, January 1990.

[SGH93] Daniel Stodolsky, Garth Gibson, and Mark Holland. Parity logging: overcoming the small write problem in redundant disk arrays. In *ACM International Symposium on Computer Architecture*, pages 64–75, San Diego, CA, May 1993.

[SGNG00] Steven W. Schlosser, John Linwood Griffin, David F. Nagle, and Gregory R. Ganger. Designing computer systems with MEMS-based storage. Technical Report CMU-CS-00-137, Carnegie Mellon University School of Computer Science, Pittsburgh, Pennsylvania, May 2000.

[SLW66] Philip H. Seaman, Robert A. Lind, and Troy L. Wilson. On teleprocessing system design, part iv: An analysis of auxilliary storage activity. *IBM Systems Journal*, 5(3):158–170, 1966.

[TP72] Toby J. Teorey and Tad B. Pinkerton. A comparative analysis of disk scheduling policies. *Communications of the ACM*, 15(3):177–184, March 1972.

[VC90] Paul Vongsathorn and Scott D. Carson. A system for adaptive disk rearrangement. *Software—Practice and Experience*, 20(3):225–242, March 1990.

[WGP94] Bruce L. Worthington, Gregory R. Ganger, and Yale N. Patt. Scheduling algorithms for modern disk drives. In *Proceedings of the 2000 ACM SIGMETRICS Conference*, pages 241–251, May 1994.

[Wis98] Kensall D. Wise. Special issue on integrated sensors, microactuators and microsystems (MEMS). *Proceedings of the IEEE*, 86(8):1531–1787, August 1998.

[WPA99] Randolph Y. Wang, David A. Patterson, and Thomas E. Anderson. Virtual log based file systems for a programmable disk. In *Symposium on Operating Systems Design and Implementation*, pages 29–43, New Orleans, LA, February 1999. ACM.