Modeling and Scheduling of MEMS-Based Storage Devices

John Linwood Griffin, Steven W. Schlosser,

Gregory R. Ganger, David F. Nagle

November 1999

CMU-CS-00-100

School of Computer Science

Carnegie Mellon University

Pittsburgh, PA 15213

Abstract

MEMS-based storage devices are seen by many as promising replacements for disk drives. Fabricated on CMOS, MEMS-based storage uses thousands of small, mechanical probe tips to access Gigabytes of nonvolatile storage. This paper takes a first step towards understanding the performance characteristics of these devices. Using trace-driven simulation and models based on the physical equations that govern the device's basic characteristics, this work explores how different physical characteristics (e.g., acceleration, data rates) and scheduling algorithms impact the design and performance of MEMS-based storage. Our results show that MEMS-based storage can improve storage access rates by a factor of 5 over conventional disk-based storage, with average access times of under 2 ms. Further, our analysis of scheduling algorithms shows that the relative benefits of request scheduling are similar to standard disks.

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 Laboratories, Hitachi, IBM Corporation, Infineon Technologies, Intel Corporation, LSI Logic, MTI Technology Corporation, Novell Inc., Procom Technology, Quantum Corporation, Seagate Technology, and 3Com Corporation. John Griffin is supported by a National Science Foundation Graduate Fellowship.

Keywords: MEMS, simulation, request scheduling

1 Introduction

For over 30 years, magnetic disks have been the on-line secondary storage components of choice. They continue to defend this position despite repeated predictions of their demise due to new technologies, such as bubble memories or holographic stores, or improvements in existing DRAM technologies. A new challenger, based on micro-electro-mechanical systems (MEMS), has arisen with very promising performance and cost characteristics.

MEMS are very small scale mechanical structures — on the order of 10s to 1000s of micrometers — fabricated on the surface of silicon wafers [18]. These microstructures are created using the same photolithographic processes used in manufacturing other semiconductor devices (e.g., CPUs and memories). MEMS structures can be made to slide, bend, or deflect in response to an electrostatic or electromagnetic force from a nearby actuator or from external forces in the environment. Using minute probe tips mounted on movable actuators, data bits can be stored in and retrieved from magnetic media coated on a silicon substrate. Practical MEMS-based storage devices are the goal of major efforts at many research centers, including IBM, CMU, and UC-Berkeley.

While disks have improved dramatically over 30+ years, their fundamental performance characteristics have not changed significantly; read/write heads are still positioned over concentric tracks of media by actuators, and data bits are still read and written as they rotate under these heads. Research and experience over the years have provided a healthy understanding of disk performance and the large mechanical delays involved, enabling the creation of useful and accurate models (e.g., [13, 17, 7, 12]). This same understanding has also led to a variety of algorithms for reducing mechanical delays via disk request scheduling (e.g., [2, 4, 14, 19]).

Like disks, MEMS-based storage devices will have mechanical and layout characteristics that will determine their performance for given workloads. For example, their mechanical positioning delays will depend on the current and destination positions. Also, data bits will be stored in lines, similar to tracks on disks, and non-zero delays will be involved with moving from one line to another. However, MEMS-based storage devices will not consist of the rotating platter and comb-like actuation components that characterize conventional magnetic disks. As a result, their performance characteristics will be very different. To assist designers of both MEMS-based storage devices and the systems that use them, an understanding of their performance must be developed.

This paper takes a first step towards developing this understanding of the performance characteristics of MEMS-based storage devices. It discusses a design point, based on a movable media

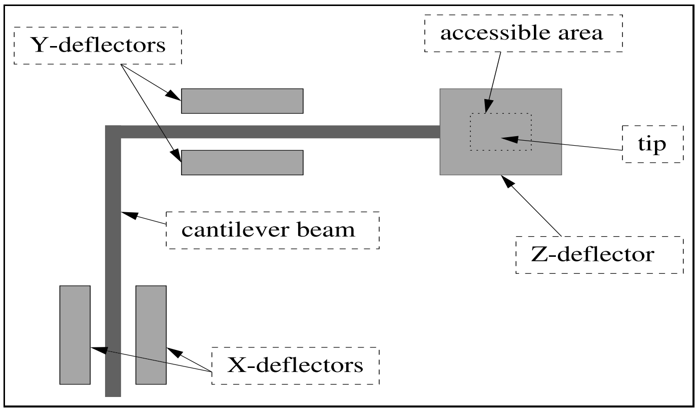


Figure 1: ***A simple cantilevered read/write tip for the "fixed media" model.*** *Here the X-deflectors and Y-deflectors are capable of positioning the tip very quickly anywhere in the relatively small accessible area; about 1% of the cantilever footprint is accessible by the tip. The Z-deflector maintains a constant distance between the tip and the media.*

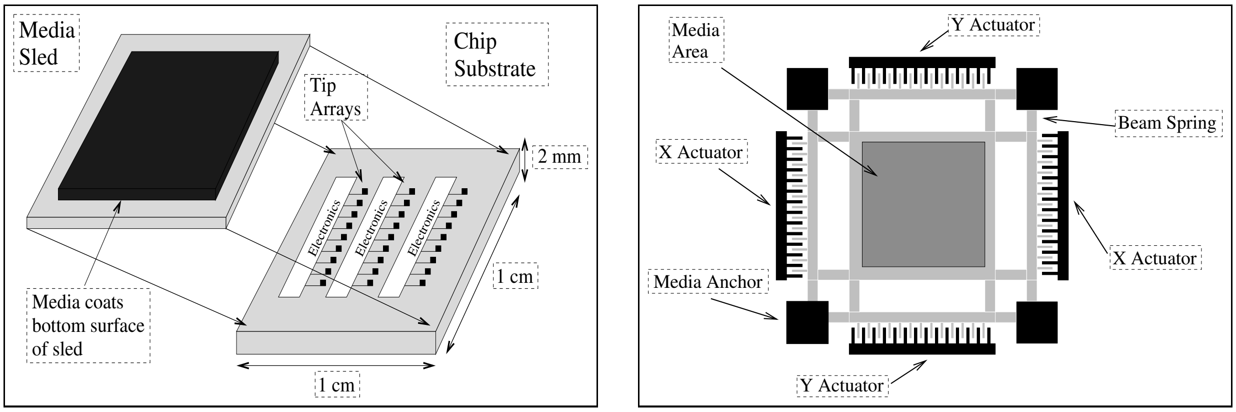
sled, that appears to offer sufficient capacity and significant performance benefits. The mechanical and layout characteristics of this model are described and timing equations for various actions are presented. Using a simulation model of the devices, we explore their performance characteristics and their sensitivity to key model parameters. We also explore the benefits of request scheduling for MEMS-based storage devices.

The remainder of the paper is organized as follows. Section 2 describes MEMS-based storage devices, focusing on the characteristics of those based on movable media sleds. Section 3 describes our experimental setup, including the simulator and the workloads used. Section 4 describes a performance model for MEMS-based storage devices and uses it to explore their performance characteristics. Section 5 evaluates request scheduling algorithms for MEMS-based storage devices. Section 6 discusses related work. Section 7 summarizes the paper's contributions.

2 MEMS-based Storage

2.1 High-level description

MEMS-based microstructures can be used in a wide range of ways to build storage devices. However, the nature of MEMS makes some approaches better than others in terms of robustness, manufacturability, capacity, etc. For example, it is very difficult to build reliable rotating components such as disk platters in a MEMS process. More promising designs utilize spring-suspended beam structures that are actuated using electrostatic or electromagnetic forces. These designs can be used to produce very accurate positioning systems that can very quickly move structures with



(a) Media sled and read/write tips (b) Media sled, beam springs, and actuators

Figure 2: ***An example of the "moving media" model.*** *In (a), we see how the media sled is attached above the fixed tips. The sled can move up to 100 μm along the X and Y axes, allowing the fixed tips to address 30–50% of the total media area. In (b), we see the actuators, the spring suspension, and the media sled itself. Anchored regions are shown in black and the movable structure is in grey.*

nanometer precision.

Most current MEMS-based storage device designs consist of several thousand read/write tips suspended on actuated beams over a substrate of magnetic media. Parallel use of many tips can provide high data rates. One design, shown in Figure 1, utilizes cantilevered beams to suspend a probe tip over a fixed magnetic substrate. Voltages applied to X-deflectors and Y-deflectors generate electrostatic forces that move the tip over different locations in the accessible area. Once in place, the tip can read or write a particular bit using standard magnetic storage techniques. In this case, the only moving part is the beam itself, which has very little mass, yielding positioning times in 100s of microseconds. Unfortunately, the space efficiency (i.e., the percentage of potential media area that can be used for storage) is only about 1%. By comparison, conventional rotating disk drives utilize about 50% of the platter area. While this model is a useful first step in visualizing MEMS-based storage devices, its expected capacity of tens of megabytes, limits its usefulness.

A more space efficient design, shown in Figure 2, reverses the design by coating the magnetic media onto a "sled" that is actuated over a set of relatively stationary beam-suspended probe tips [9]. In this case, the sled is suspended with springs and actuated in two dimensions, X and Y, by about 100 μm. The tips are fixed below the sled and are actuated in the Z axis to track the possibly non-uniform surface of the media. They can also be actuated in the X axis by a few tens of nanometers. This small X actuation is used either to keep the tips aligned with the data moving past and may also allow individual tips to read data with slightly different X offsets. Sharing the

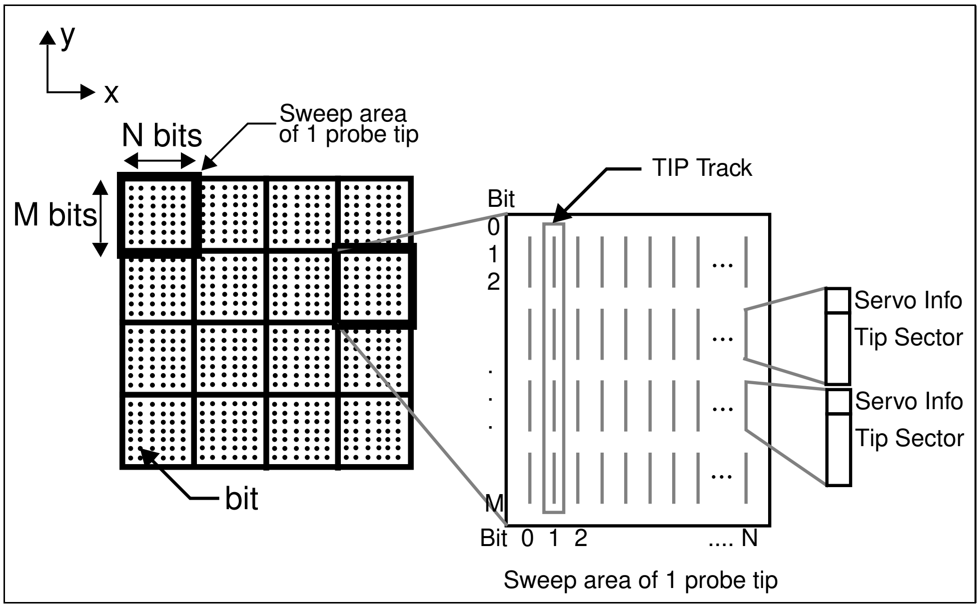


Figure 3: ***Data organization of MEMS-based storage.*** *The illustration depicts a small portion of a MEMS-based storage device. Each rectangle outlines the area accessible by a single probe tip, with a total of 16 probe tip regions shown. A full device contains 1000s of rectangular regions (i.e., 1000s of probe tips). Each rectangular region stores NxM bits, organized into* tip sectors, *containing data and ECC bits. These follow periodic sequences of* servo info *bits as shown. This servo information is expected to require around 10% of the capacity of the device. To read data, the media passes over the active tip(s) in the* $+/-$ *Y direction, allowing them to read 1 or more tip sectors each.*

large media actuators across all of the tips improves space efficiency to 30–50%, at a cost in access speed due to the vastly greater mass of the moving parts. However, to achieve disk-like capacities, this type of tradeoff is necessary. The remainder of this paper focuses on MEMS-based storage devices that use moving media sleds.

2.2 Device Characteristics and Data Layout

MEMS-based magnetic media is organized in rectangular regions as shown in Figure 3. Each rectangle stores NxM bits and is accessible by a single probe tip. Similar to conventional disks, individual bits cannot be read by a tip; instead, each probe tip reads a linear sequence of bits consisting of data, error correction codes, and servo information. Each grouping of data and ECC that follows the servo information is called a *tip sector*, and represents the smallest accessible unit of data. Multiple tip sectors can be viewed as *logical sectors* (like logical blocks in SCSI disks). Unlike most conventional disks, multiple probe tips can access the media in parallel. Thus, many tip sectors are read or written at once as the media sled slides sequentially past the set of active

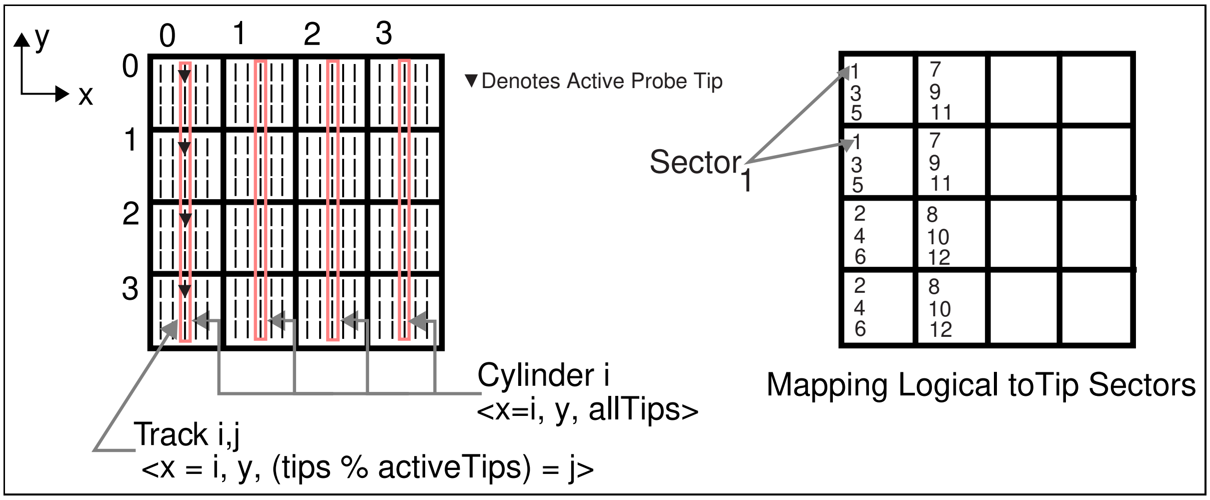


Figure 4: ***Cylinders, Tracks, and Sectors.*** *$Cylinder_i$ is defined as all of the columns of data with the same X coordinate: <x=i, y, tip>. $Track_{i,j}$ is the subset of a cylinder that is accessible by the concurrently active tips: <x=i, y, (tip % activeTips) = j>. (Note that activeTips=4 in this figure.) 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 of the leftmost (x=0) column.*

probe tips. Due to power and heat considerations, however, not all of the device's thousands of tips can be active at a time.

To organize a device's bits, we identify each bit by a triple, <*x,y,tip*>, where *x,y* represent coordinates within the rectangular region addressable by *tip* (Figure 3). Drawing an analogy to disk terminology, the set of all bits with a given value of *x* can be viewed as a *cylinder* (Figure 4). That is, a cylinder consists of one column of bits per probe tip. Since only a subset of the tips can be active at any one time, we divide a cylinder into *tracks*, which are those bits within a cylinder that can be read or written by concurrently active tips. As with a conventional disk, reading or writing a complete cylinder requires multiple passes with track switches (i.e., tip switches) in between. Figure 4 illustrates this bit organization with a simple example. In this case, there are sixteen total tips, of which only four can be active. The four leftmost tips are active and the current track is indicated.

Parallelism among probe tips allows a logical sector's data bits to be spread across multiple tracks. After the media sled is positioned, probe tips can only start accessing bits at one of the servo info bursts; therefore, the tip sector represents a useful minimum number of bits per tip contributing to a given logical sector. Of course, this may mean that multiple logical sectors are accessed concurrently. Figure 4 illustrates one possible layout in which the first tip sector of probe tips [0,0] and [0,1] (i.e., the probe tips in the rectangle [column 0, row 0] and [column 0, row 1]) form the first logical sector. The first tip sectors of probe tips [0,2] and [0,3] form the second logical sector. Logical sectors 1 and 2 are read simultaneously, doubling bandwidth in this example.

To read or write data, the media sled must first move to the desired sector. More specifically, the sled must be positioned over the first bit of the appropriate pre-sector servo burst. Further, the appropriate probe tips must be activated and the sled must be moving at the required access velocity and in the proper direction (i.e., the +/− Y direction). Thus, positioning the media sled involves several mechanical and electrical actions. First, moving the sled to a given $<x,y>$ position may require movement in both the X dimension and the Y dimension. Because the sled accelerates quickly and then decelerates to a standstill, X-dimension seeks require an extra settling time for the actuator-spring-sled assembly to stop oscillating. Second, the active tips may need to be changed. Third, the sled must be accelerated along the column (i.e., Y-dimension). The first two steps can proceed in parallel, followed by the third.

Once media access begins, the media sled moves along the column at the access velocity, which includes a movement of the probe tip(s) in the Z dimension. As with conventional disks, large data transfers may require that data from multiple tracks and/or cylinders be accessed. To switch tracks, the sled must perform a *turnaround*, which consists of decelerating to a stop, reversing direction, and accelerating to the access velocity. Simultaneously, the active tips are disabled and the tips for the new track are activated. To switch cylinders, a turnaround is also necessary, as is a small seek to the next column. In both cases, the turnaround time is expected to dominate the additional activity.

3 Experimental apparatus

Simulation. To explore the performance of MEMS-based storage devices, we have developed a simulation model based on the equations, algorithms, and parameters described in Section 4.1. 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 the experiments to be useful.

To accurately model and compare storage systems, we integrated this device model into the DiskSim simulation environment [3]. 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 Seagate Technologies *Cheetah 4LP ST-34501W* [15]. Using the Cheetah configuration parameters, DiskSim's model has been validated against a real Cheetah disk drive. Using the model validation scheme of [12], the demerit figure is 0.16ms (or 1.3% of the corresponding average access time of 12.2ms).

Workloads. Our experiments use three workloads. Most experiments use a synthetically-generated workload that we refer to as the *random* workload. Request inter-arrival 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 4KB, and request starting locations are uniformly distributed across the device's capacity. To include more realistic workloads, two traces of real storage activity are also utilized; they are described in Section 5.

4 Performance of MEMS-based Storage Devices

This section describes a performance model of MEMS-based storage devices, its performance given reasonable default parameters, and its sensitivity to the settings of various parameters.

4.1 Modeling Performance for MEMS-Based Storage Devices

In describing how one can model the performance of a MEMS-based storage device, it is useful to first consider basic disk performance models for reference. The service time for a disk access is often computed as:

$$time_{service} = time_{seek} + latency_{rotate} + time_{transfer}$$

The seek time, $time_{seek}$, can be computed as a function of the distance in cylinders that must be travelled. Accurate functions consist of an acceleration/deceleration component for short seek distances, a linear component (representing the maximum velocity of the seek arm) for longer seeks, and a significant settling delay (e.g., 1 ms) for all non-zero seeks [12]. The rotational latency, $latency_{rotate}$, can be computed as the product of the fraction of the track that must rotate by before reaching the first desired sector and the time for a full revolution. Since the disk rotates continuously, detailed simulation requires accounting for all advances in time, including the seek time for the access being serviced. The continuous rotation also tends to make $latency_{rotate}$ independent from access to access; in less detailed models, a uniform distribution $[0..time_{rotate}]$ is a reasonable approximation. The media transfer time, $time_{transfer}$, can be computed as the product of the number of sectors desired divided by the number of sectors per track (in the relevant zone) and the time for a full revolution. Detailed models must also account for all track and cylinder boundaries crossed by the range of desired sectors, since each crossed boundary adds a repositioning delay equal to the corresponding skews in the logical-to-physical mapping [12].

The service time for MEMS-based storage devices can be modeled with a similar equation:

$$time_{service} = time_{seek} + time_{transfer} \tag{1}$$

The obvious difference is the absence of rotational latency. Less obvious from the equation is the much more complicated nature of the $time_{seek}$ term. Recall that the movable media sled must both seek to the correct $<x,y>$ position and attain the proper media access velocity in the proper direction. The actuation mechanisms and control loops for X and Y positioning are independent, which allows the two to proceed in parallel. Thus,

$$time_{seek} = max(time_{seek\_x}, time_{seek\_y})$$

Equations from classical first-order mechanics (e.g., $\Delta x = v_0 t + \frac{1}{2}at^2$) can be used to compute both $time_{seek\_x}$ and $time_{seek\_y}$, though each also involves an additional component. For example:

$$time_{seek\_x} = 2 * \left(\frac{-v_{0x} + \sqrt{{v_{0x}}^2 + a_x|\Delta x|}}{a_x} \right) + time_{settle} = 2 * \sqrt{|\Delta x|/a_x} + time_{settle} \tag{2}$$

The equation's reduction is possible because $v_{0x} = 0$. This equation assumes that sled movements are sufficiently short that maximum sled velocity is not reached during seeks; given the very small distances involved, this is expected to be true. The large fractional component is the solution to the mechanics equation given earlier for a distance of $\frac{1}{2}\Delta x$. Multiplying this by 2 gives the time required to move Δx columns by accelerating at full rate for half of the distance and then decelerating at full rate for the second half of the distance, targetting a standstill at the destination. The extra component in this equation is $time_{settle}$, which represents the time required for the actuator-spring-sled assembly to stop oscillating after the applied force is removed. $time_{settle}$ is dependent on the resonant frequency of the system, f:

$$time_{settle} = \frac{1}{2\pi f} * number_{timeconstants} \tag{3}$$

where $number_{timeconstants}$ is a function of how much damping is needed before the probe tips can begin to robustly access the media. This oscillation could be damped by the spring/mass system itself or by the atmosphere. More likely, the system will have a closed-loop control system that actively damps the oscillations using the actuators. However, as these devices can not yet be measured directly, we approximate their operation with the parameter, $number_{timeconstants}$.

The equation for the Y dimension is similar:

$$time_{seek\_y} = 2 * \left(\frac{-v_{0y} + \sqrt{{v_{0y}}^2 + a_y|\Delta y|}}{a_y} \right) + time_{turnaround} * number_{turnarounds} \tag{4}$$

There are two major differences. First, v_{0y} is rarely zero because the device is moving at the data access velocity. Further, since the device needs to be moving at the access speed after positioning, the equation (with $v_{0y} = v_{access}$) can be doubled for half the distance as was done with the X dimension equation — starting with an initial velocity, the device accelerates at maximum rate for half the distance and then decelerates back to the original velocity over the second half. The second difference is that the $time_{settle}$ term is replaced by the time for turnarounds. Because the media is not brought to a stop in the Y axis, oscillation is not a significant problem. However, since the media sled may be moving in the wrong direction before the seek and/or after the seek, it may be necessary to reverse its direction once or twice. For each such turnaround:

$$time_{turnaround} = 2 * \frac{v_{access}}{a_y} \tag{5}$$

The $time_{transfer}$ component of the MEMS-based storage device service time is less distinct from that of conventional disks, but does vary in two ways. First, the time to transfer a single sector is the product of the number of tips over which each sector is striped, the rate at which bits are read ($v_{access} \div width_{bit}$), and the percentage of bits read that are actual data (e.g., rather than servo and ECC). Second, the time to transfer a range of sectors must take into account the fact that multiple sectors can be accessed in parallel; the number of sectors accessed in parallel is the number of concurrently active tips divided by the number of tips per sector. As with conventional disks, when a range of sectors to be transfered crosses a track or cylinder boundary, a track or cylinder switch is required. The sequential track switch time is equal to the turnaround time, since switching the active tips is expected to take less than this time. The sequential cylinder switch time can be computed as a single cylinder seek, but optimizations of the control loop can be expected to reduce this time to the turnaround time by taking advantage of the tips' ability to deflect small

per-tip Y dimension	100 μm
per-tip X dimension	100 μm
bit width	50 nm
number of tips	6400
simultaneously active tips	1280
tips per sector	64
encoding efficiency	8 data bits in 10
servo bursts	10 bits per 90
sled acceleration	114.8 m/s^2
access velocity	20 mm/s
settling time constants	1
sled resonant frequency	220 Hz
total capacity	2.56 GBytes

Table 1: ***Default parameters for the MEMS-based storage device model.***

distances in the X dimension.

4.2 Performance of the Default Model

The above performance model has been built into the DiskSim simulator, allowing us to evaluate its performance under different workloads and parameter settings. This section explores MEMS-based storage device performance given the default parameters listed in Table 1. Based on detailed discussions with engineers designing and building MEMS-based storage devices, we believe these default values are reasonable starting points for evaluating these devices.

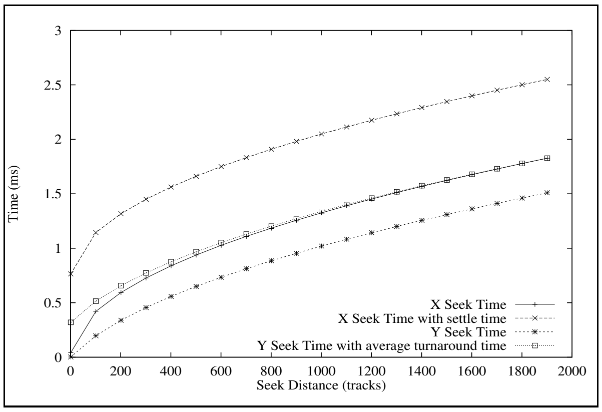
Table 2 summarizes performance metrics for the default MEMS-based storage device under a *random* workload of 100,000 requests. We see measured values for the components that make up the total service time described in Equation 1: $time_{seek}$ and $time_{transfer}$. From the numbers we see that the average service time is dominated by the average seek time.

Settling time is computed by applying the parameters of the model to Equation 3. This is a constant time that is added onto any seek in the X direction. The time for a single turnaround is computed by applying the model parameters to Equation 5. The average number of turnarounds for the *random* workload was measured to be slightly fewer than one per seek. Therefore, the average turnaround time added to the Y seek computation (Equation 4) is slightly less than the time for a single turnaround.

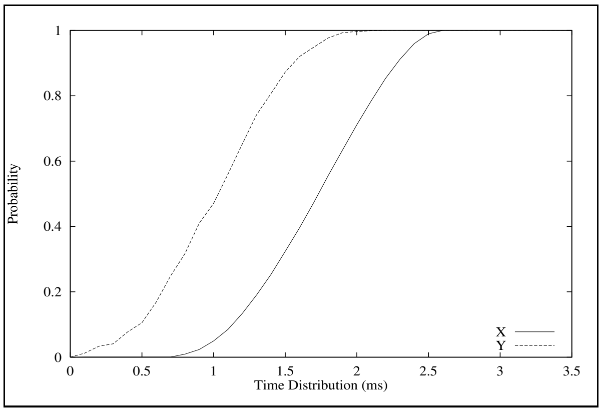
Figure 5(a) shows plots of $time_{seek}$ for seeks in both the X direction and the Y direction. First, we see that the X component of seeks will usually dominate the Y component because the sled has an initial velocity, v_0, in the Y axis equal to the access velocity. Further, when we add in

Average service time	1.96 ms ($\sigma = 0.46$)
Maximum service time	5.03 ms
Average seek time	1.81 ms ($\sigma = 0.42$)
Maximum seek time	2.72 ms
Average transfer time	0.16 ms ($\sigma = 0.21$)
Maximum transfer time	3.73 ms
Settling time	0.72 ms
Single turnaround time	0.35 ms
Average per-request turnaround time	0.31 ms

Table 2: ***Performance characteristics of the default MEMS-based storage device model. (Standard deviations in parentheses.)***



(a) Seek time versus distance



(b) Measured seek time distribution

Figure 5: **Seek times for the default MEMS-based storage device.**

the average settle time in X and the average turnaround time in Y, we see this gap grow even larger. Comparing these results to the X and Y seek time distributions measured from the *random* workload shown in Figure 5(b), we see that the maximum seek times in both axes match up nicely.

4.3 Performance Sensitivity to Model Parameters

To better understand the MEMS storage device's performance and design tradeoffs, we performed several experiments varying fundamental design parameters. Specifically, we varied the acceleration, the width of bits, the resonant frequency of the media sled, the number of time constants required for settling after seeks, and the access velocity of the system.

Figure 6 shows how increasing acceleration can significantly decrease average access time because it decreases both seek and turnaround times. Since seek time is the dominant component of access time, this is an important physical design optimization to consider. In contrast, increasing velocity has both a positive and negative impact on performance. Figure 7 shows that a small velocity increase (1 to 10 mm/sec) significantly improves average response time. After 20 mm/sec, however, faster sleds actually hurt performance unless acceleration is very fast (e.g., 400 m/s^2). There are two reasons for this behavior. First, accelerating the media to its final velocity takes longer with a higher velocity. Second, turnaround times increase because it takes longer to decelerate/accelerate the sled when changing direction. Therefore, designers should choose a read/write velocity that is near the bottom of the curve. While this translates to a fairly low per-tip data rate of around 400K bits per second, parallel access by multiple active tips provides the high aggregate data rates seen in Table 2.

After the sled moves in the X dimension, some time is required to damp the motion to the point where data can be accessed. This settling time is based on the resonant frequency as described in Equation 3. Our simulations approximate this effect by waiting a specified number of time constants before allowing data access, with the baseline model using a single time constant. Figure 9 shows the inverse relationship between resonant frequency and access time. Conversely, Figure 10 shows that the response time is directly proportional to the number of time constants required for settling.

In looking at the various sensitivities of the model to its parameters, it is tempting to choose optimal values for each parameter to try and create an optimal device. However, this study does not show the complex interdependencies between parameters other than the effect of increasing acceleration on velocity sensitivity. Even more complex are the tradeoffs involved in actually fabricating the device. For example, the use of a more sophisticated spring suspension could affect the damping characteristics of the system, changing its dependence on the resonant frequency. While it is important to explore the various degrees of freedom in the model, many of the tradeoffs remain unclear because the devices have not yet been built.

5 Scheduling of Requests

An important mechanism for improving disk drive efficiency is the 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 highly dependent on the distance to be travelled. This section explores the impact of different scheduling algorithms on the performance of MEMS-based storage devices.

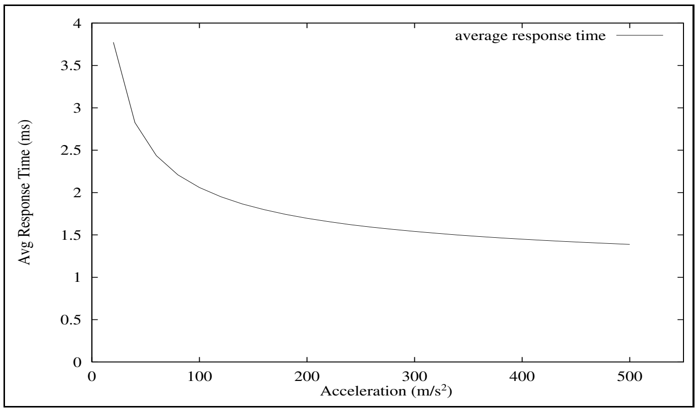


Figure 6: **Sensitivity of MEMS-based storage device performance to rate of acceleration.**

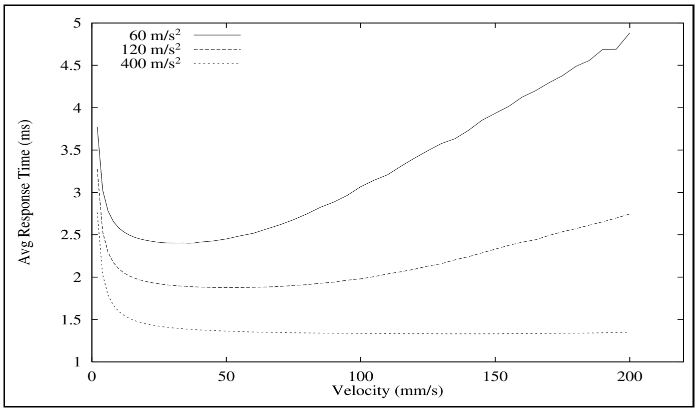


Figure 7: **Sensitivity of MEMS-based storage device performance to the access velocity at three different accelerations.**

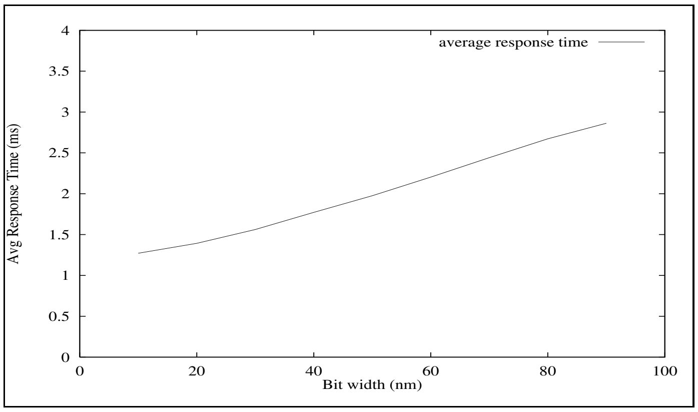


Figure 8: **Sensitivity of MEMS-based storage device performance to bit width.**

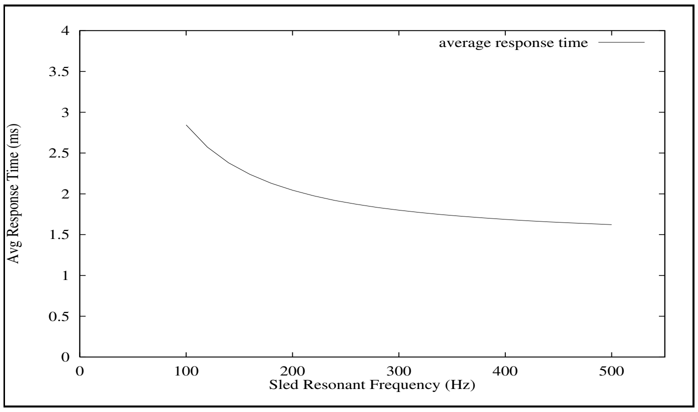


Figure 9: **Sensitivity of MEMS-based storage device performance to resonant frequency.**

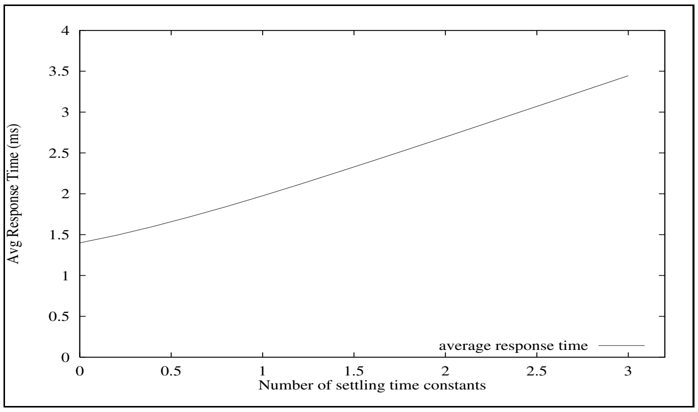


Figure 10: **Sensitivity of MEMS-based storage device performance to the number of time constants required for settling.**

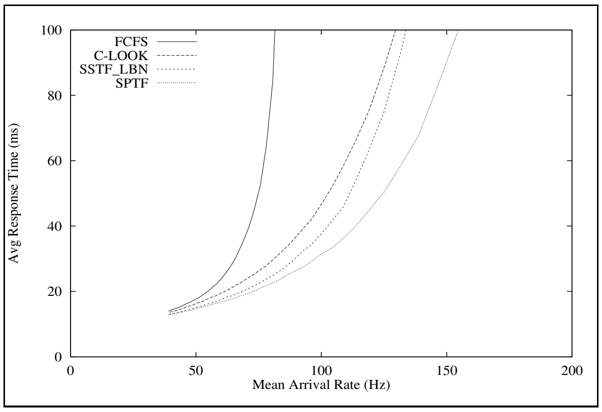
5.1 Disk Scheduling Algorithms

Many disk scheduling algorithms have been devised and studied over the years. Our comparisons focus on four. The simple *First Come First Served* (FCFS) policy often results in suboptimal performance, but we include it for reference. The *Shortest Seek Time First* (SSTF) policy is designed to select the request that will incur the smallest seek delay, but this is rarely the way it functions in practice. Instead, since it is not generally feasible for host systems to identify actual seek distances or predict seek times, most SSTF implementations use the difference between the last accessed LBN (Logical Block Number) and the desired LBN as an approximation of seek time. This simplification works well for disk drives [19], and we will label this algorithm as "SSTF_LBN". The *Cyclical LOOK* (C-LOOK) policy 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 [14, 8]. For disk drives, this policy differs from others in that it explicitly considers both seek time and rotational latency.

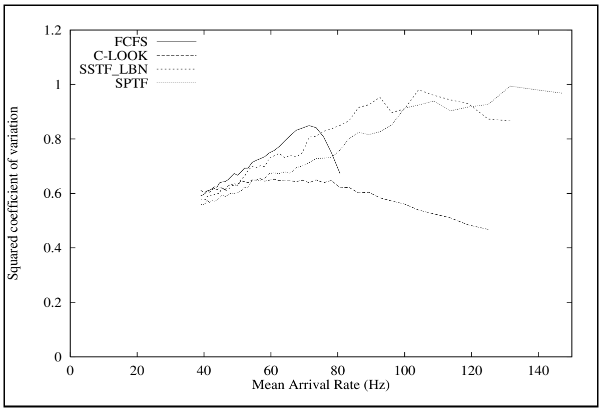
For reference, Figure 11 compares these four disk scheduling algorithms for the Cheetah disk drive and the *random* workload with a range of request arrival rates. Two common metrics for evaluating disk scheduling algorithms are shown. First, the average response time (queue time plus service time) shows 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 [16, 19]; lower values indicate better starvation resistance. As expected, C-LOOK avoids the request starvation effects that characterize the greedy SSTF_LBN and SPTF algorithms.

5.2 Request Scheduling for MEMS-Based Storage Devices

Existing disk scheduling algorithms can be applied directly to MEMS-based storage devices. 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



(a) Average Response Times (ms)



(b) Squared Coefficients of Variation

Figure 11: **Comparison of scheduling algorithms for the random workload on the Cheetah disk drive.**

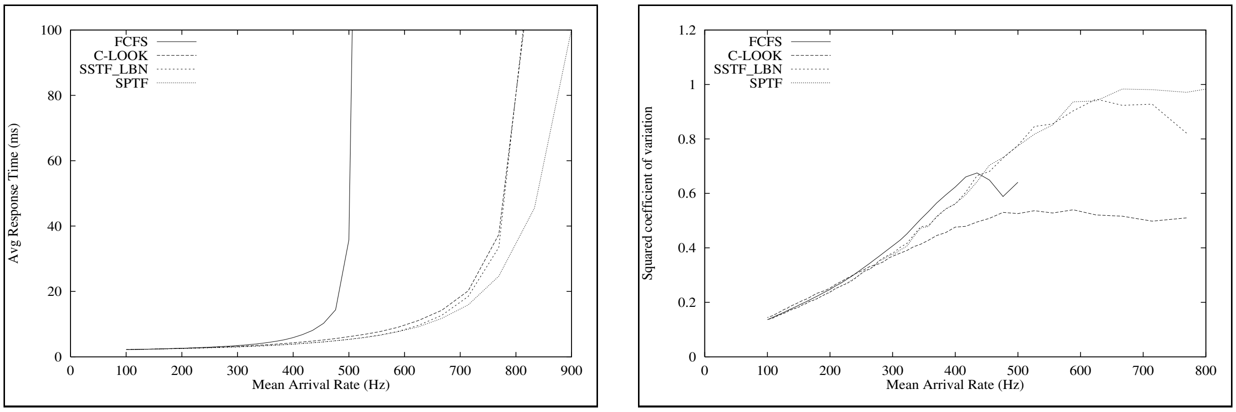
disks, the two positioning components proceed in parallel, with the greater hiding the lesser. As previously shown in Figure 5, the settling time delay makes most X dimension seek times larger than most Y dimension seek times. SPTF will only outperform SSTF (which minimizes X movements, but ignores Y) when the Y component is the larger.

Figure 12 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.

5.3 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[1]. The hardware was a 300 MHz Pentium

[1]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



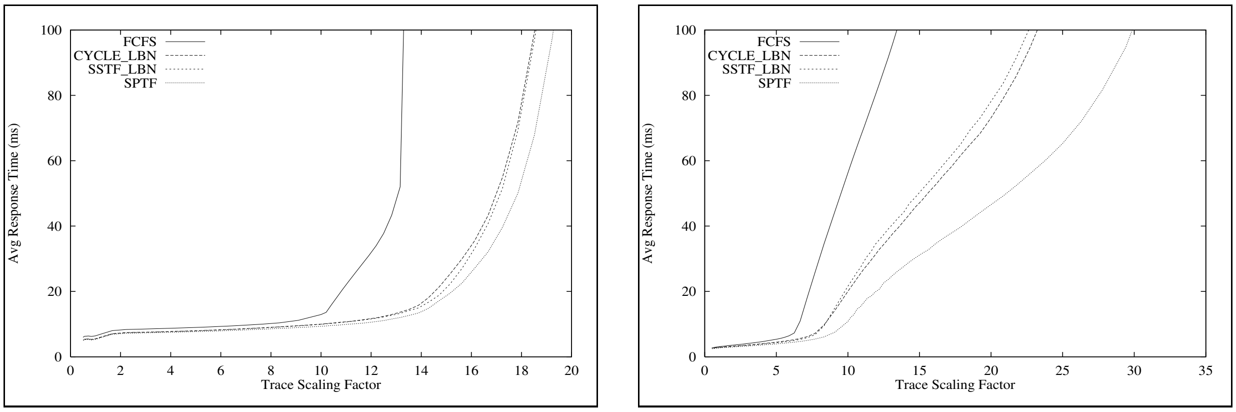
(a) Average Response Times (ms) (b) Squared Coefficients of Variation

Figure 12: **Comparison of scheduling algorithms for the random workload on the MEMS-based storage device.**

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 [10]. 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 [11].

Figures 13(a) and (b) show how the scheduling algorithms perform for the *Cello* and *TPC-C* workloads, respectively. The relative performance of the algorithms is very similar to the *random* workload. The one noteworthy difference 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.

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 [19]: 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) *Cello* (b) *TPC-C*

Figure 13: **Comparison of scheduling algorithms for the *Cello* and *TPC-C* workloads on the MEMS-based storage device.**

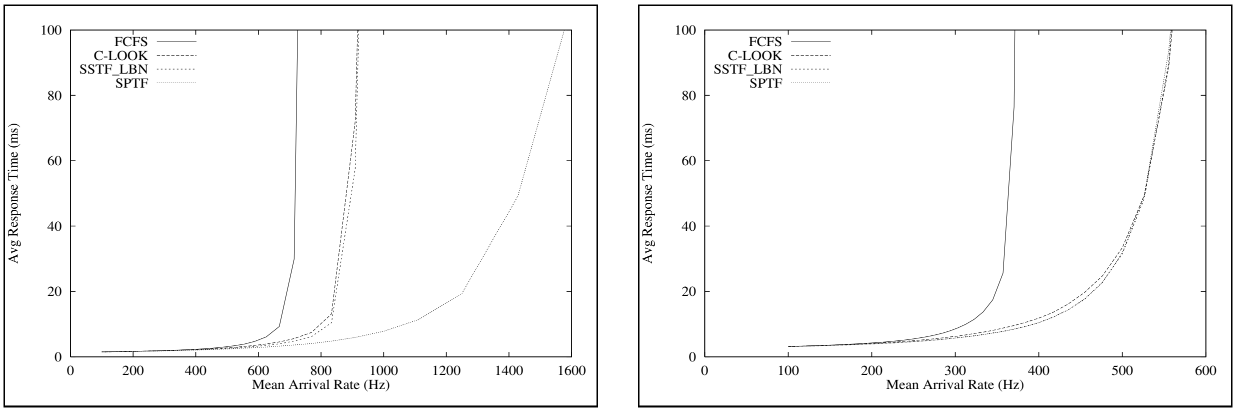
5.4 Interaction of SPTF and Settling Times

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, Figure 14 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 0 settling time constants, SPTF outperforms the other algorithms by a large margin.

6 Related Work

Our ability to simulate and understand the performance of MEMS-based storage devices builds on the solid foundation of previous work with conventional disk performance. Some of the many relevant papers and efforts are identified in other sections, and there are too many others to list here. Similarly, the scheduling algorithms explored in Section 5 come from previous work. Many other algorithms (e.g., [13, 2, 4, 8, 19]) and evaluations/comparisons of their efficacy for disks (e.g., [1, 16, 6, 5, 19]) have been described in the literature. Based on our experiments, we believe that most of the insights offered in these works will also be relevant to MEMS-based storage devices.

We are not currently aware of other efforts to model, characterize, and enhance the performance



(a) *Random* with zero time constants (b) *Random* with two time constants

Figure 14: **Comparison of average performance for zero and two settling time constants, respectively.** *The default model uses one settling time constant and is shown in Figure 12(a).*

of MEMS-based storage devices.

7 Conclusions

This paper develops a performance model for MEMS-based storage devices and uses it to evaluate their performance. The results of this study provide both MEMS researchers and computer system researchers with a significant glimpse into the potential performance wins and design tradeoffs of MEMS-based storage. Overall, these devices provide average service times of under 2 ms. Further, request scheduling algorithms increase performance for MEMS-based storage devices like they do for disks. Continuing this work, we are exploring: (1) how to best structure MEMS devices given the complex interactions between physical parameters; (2) what are the appropriate file system and OS structures to manage such devices; and (3) how to use MEMS-based storage in supporting a wide range of current and future applications such data mining, speech recognition, and portable computing.

References

[1] E. Coffman, L. Klimco, and B. Ryan. Analysis of scanning policies for reducing disk seek times. *SIAM Journal of Computing*, 1(3):269–279, September 1972.

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

[3] G. Ganger, B. Worthington, and Y. Patt. The disksim simulation environment version 1.0 reference manual. Technical Report CSE-TR-358-98, The University of Michigan, Ann Arbor, February 1998.

[4] R. Geist and S. Daniel. A continuum of disk scheduling algorithms. *ACM Transactions on Computer Systems*, pages 77–92, February 1987.

[5] R. Geist, R. Reynolds, and E. Pittard. Disk scheduling in system v. In *ACM SIGMETRICS Conference*, pages 59–68, may 1987.

[6] M. Hofri. Disk scheduling: Fcfs vs. sstf revisited. *Communications of the ACM*, 23(11):645–653, November 1980.

[7] D. Hunter. Modeling real dasd configurations. Technical Report Research Report RC8608, IBM, September 1997.

[8] D. Jacobson and J. Wilkes. Disk scheduling algorithms based on rotational position. Technical Report Technical Report HPL-CSP-91-7, Hewlett-Packard Laboratories, February 1991.

[9] L. R. Carley, J. A. Bain, G. K. Fedder, et al. Single chip computers with mems-based magnetic memory. In *44th Annual Conference on Magnetism and Magnetic Materials*, November 1999.

[10] E. Riedel. *Active Disks - Remote Execution for Network-Attached Storage*. PhD thesis, Carnegie Mellon University, 1999.

[11] C. Ruemmler and J. Wilkes. Unix disk access patterns. In *Winter USENIX Conference*, pages 405–420, jan 1993.

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

[13] P. Seaman, R. Lind, and T. Wilson. On teleprocessing system design, part iv, an analysis of auxilliary storage activity. *IBM Systems Journal*, 5(3):158–170, 1966.

[14] M. Seltzer, P. Chen, and J. Ousterhout. Disk scheduling revisited. In *Winter USENIX Conference*, pages 313–324, jan 1990.

[15] Seagate Technology. St-34501w (cheetah 4lp family). http://www.seagate.com/support/disc/specs/st34501w.shtml, September 1997.

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

[17] N. Wilhelm. A general model for the performance of disk systems. *Journal of the ACM*, 24(1):14–31, January 1977.

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

[19] B. Worthington, G. Ganger, and Y. Patt. Scheduling algorithms for modern disk drives. In *ACM SIGMETRICS Conference*, pages 241–251, may 1994.