Using MEMS-Based Storage in Computer Systems — Device Modeling and Management

Bo Hong[†]
Symantec Corporation
Scott A. Brandt
Darrell D. E. Long
Ethan L. Miller
University of California, Santa Cruz
and
Ying Lin[†]
Microsoft Corporation

MEMS-based storage is an emerging non-volatile secondary storage technology. It promises high performance, high storage density, and low power consumption. With fundamentally different architectural designs from magnetic disk, MEMS-based storage exhibits unique two-dimensional positioning behaviors and efficient power state transitions. We model these low-level, device-specific properties of MEMS-based storage and present request scheduling algorithms and power management strategies that exploit the full potential of these devices. Our simulations show that MEMS-specific device management policies can significantly improve system performance and reduce power consumption.

Categories and Subject Descriptors: D.4.2 [**Storage Management**]: Secondary storage; D.4.8 [**Performance**]: Modeling, Simulation

General Terms: Algorithms, Management, Performance

Additional Key Words and Phrases: MEMS-Based storage, request scheduling, power management

1. INTRODUCTION

MEMS-based storage, *a.k.a.* probe-based storage, is an emerging non-volatile secondary storage technology that promises seek times ten times faster, storage densities ten times greater, and power consumption one to two orders of magnitude lower than hard disks [Carley et al. 2000; Toigo 2000; Vettiger et al. 2000]. Based on microelectromechanical po-

Author's address: B. Hong, Symantec Corporation, 350 Ellis St., Mountain View, CA 94043; S. A. Brandt, D. D. E. Long, and E. L. Miller, Department of Computer Science, University of California, 1156 High St., Santa Cruz, CA 95064; Y. Lin, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052. The contact author is D. D. E. Long.
[†] Work done while at the University of California, Santa Cruz.

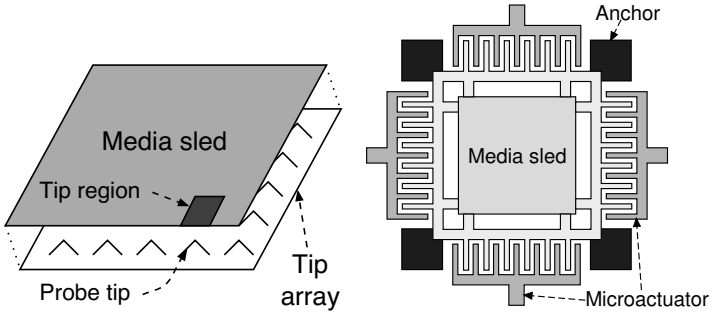
sitioning systems, MEMS-based storage has fundamentally different architectural designs and manufacturing processes from magnetic disks. It uses a non-rotating storage device with storage media on one surface and a large array of read/write probe tips on another surface directly above the storage media. By moving the surfaces relative to each other using MEMS actuators, each read/write probe can access a region of the surface. MEMS-based storage can provide initially 2–10 gigabytes of storage in a single chip as small as a dime, with high throughput, high resistance to shock, and low entry cost. For all of these reasons, MEMS-based storage is an appealing next-generation storage technology, particularly for mobile computing and high-end systems, where size and power or performance are important. We focus on power management and request scheduling on MEMS-based storage to take advantage of its low-level, device-specific characteristics.

There has been strong research interest in the roles and corresponding management policies of MEMS-based storage in computer and database systems since 1999 [Griffin et al. 2000a; 2000b; Schlosser et al. 2000; Uysal et al. 2003; Yu et al. 2003]. By comparing external behaviors and performance of MEMS storage devices and a hypothetical "super" disk, Schlosser and Ganger [Schlosser and Ganger 2004] concluded that MEMS devices were much like disks, and today's storage interfaces and abstractions were also suitable for such devices, except for their efficient accesses to two-dimensional data structures, such as relational database tables [Yu et al. 2003].

Treating MEMS devices as small, low-power, fast disk drives has an obvious advantage in that systems can easily leverage the overall superior performance of MEMS-based storage and promote such an emerging technology for quicker and wider adoption when it is available. However, our research shows that systems that taking advantage of the unique properties of MEMS-based storage can provide even better performance.

One of the fundamental tasks of operating systems is to efficiently manage resources. Of particular concern are the secondary storage devices, which are frequently a limiting factor in computer system performance. Beyond the standard interfaces and abstractions, operating system researchers and designers often resort to device-specific knowledge in order to more efficiently utilize the hard disks. Examples include FFS [McKusick et al. 1984], LFS [Rosenblum and Ousterhout 1992], and track-aligned extents [Schindler et al. 2002] on data layout; Shortest Positioning Time First (SPTF) [Jacobson and Wilkes 1992; Seltzer et al. 1990] and freeblock scheduling [Lumb et al. 2002] on request scheduling; and various disk "spin-down" policies [Helmbold et al. 1996; Li et al. 1994] on power management. This is also true for MEMS-based storage devices. Our research demonstrates significant benefits from MEMS-specific management of both request scheduling and power conservation.

As a result of its architectural designs, MEMS-based storage has unique non-rotating, two-dimensional, independent positioning behaviors. Disk-analogous request scheduling algorithms that only consider MEMS device movements in one dimension are therefore unlikely to be optimal. We developed Zone-based Shortest Positioning Time First *(ZSPTF)*, a new MEMS-specific scheduling algorithm that partitions the MEMS device into a two-dimensional array of *zones*, within which seek times are considered to be equivalent. ZSPTF traverses zones in Circular Scan (C-SCAN) order and services requests within each zone in Shortest Positioning Time First (SPTF) order. This algorithm significantly outperforms other disk-analogous algorithms with average performance comparable to SPTF and service fairness similar to C-SCAN. ZSPTF also avoids the computational complexity that



(a) Probes and media sled (b) Media sled and actuators

Fig. 1. Components of a MEMS-based storage device.

plagues SPTF and its variants.

A single MEMS device is expected to consume much less power than a disk. However, due to its limited capacity per device, the power consumption efficiency index of MEMS-based storage, as defined by watt per gigabyte, is comparable to that of low-power disks [Hitachi Global Storage Technologies 2004; Seagate Technology, Inc. 2004]. As MEMS storage is likely to be used in small portable devices, power management remains an important factor in overall system performance. Unlike disks, MEMS devices can quickly switch from low-power modes to active mode with little energy overhead. They may also be able to turn on or off hundreds to thousands read/write tips as needed. We have developed three MEMS-specific power conservation strategies that take advantage of these features – *aggressive spin-down*, *sequential request merging*, and *subsector accesses*. We show that, using these techniques, MEMS storage device energy consumption can reduced by up to 50% with little or no negative impact on I/O performance.

2. MEMS-BASED STORAGE

We base the physical parameters of our experimental MEMS device model on the specification from Carnegie Mellon University (CMU) [Carley et al. 2000]. Although there are several designs of MEMS-based storage described in the literature, we used the CMU model because all of these designs share many similarities in both architectures and expected performance and the CMU design was disclosed in the greatest detail.

Figure 1 shows the details of a MEMS-based storage device. The device consists of a surface coated with magnetic media, called a *media sled*, and a two-dimensional array of stationary read/write probe tips, called a *tip array*, as shown in Figure 1(a). The media sled is suspended above the tip substrate by silicon beams that act as springs, and moved independently and in parallel in the x and y directions by forces generated by lateral resonant microactuators, as shown in Figure 1(b). Data is written and retrieved by moving the media sled at a constant velocity in the y direction while the tip array remains stationary. Sled velocity in the x direction needs damping to zero before data transfer to avoid off-track interference, which leads to *settling time*. A MEMS device can activate multiple tips at the same time. Data can then be striped across multiple tips, providing a considerable amount of parallelism. However, power and heat considerations limit the number of probe tips that can be active simultaneously. For instance, out of 6400 probe tips, 1280 tips can be active at once in the CMU **G2** model while 3200 tips in the **G3** model [Schlosser et al. 2000].

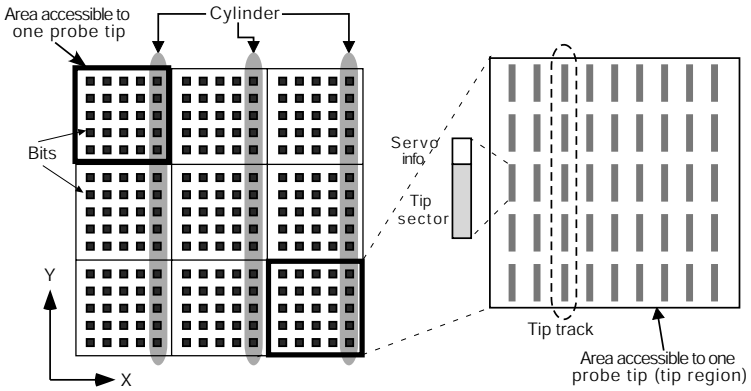


Fig. 2. Data layout on a MEMS device.

Table I. Default MEMS-based storage device parameters.

Per-sled capacity (GB)	3.2
Max. throughput (MB/s)	89.6
Number of tips	6400
Maximum concurrent tips	1280
Sled mobility in x and y (μm)	100
Sled acceleration in x and y (m/s^2)	803.6
Sled access speed (mm/s)	28
Sled resonant frequency (Hz)	739.0
X settling time (ms)	0.215
Spring factor	75%
Media bit cell size (nm^2)	40×40
Bits per tip region ($M \times N$)	2500×2500

Figure 2 illustrates the low level data layout of a MEMS-based storage device. The media sled is logically broken into non-overlapping *tip regions*, defined by the area that is accessible by a single tip, approximately 2500 by 2500 bits in size. It is limited by the maximum amount of sled movement. Each tip in the MEMS device can only read data in its own tip region. The smallest unit of data in a MEMS storage device is called a *tip sector*. Each tip sector, identified by the tuple $\langle x, y$,*tip*$\rangle$, has its own servo information for positioning and its own error correction information. The set of bits accessible to simultaneously active tips with the same x coordinate is called a *tip track*, and the set of all bits (under all tips) with the same x coordinate is referred to as a *cylinder*. A set of concurrently accessible tip sectors is grouped as a *logical sector*. For faster access, logical blocks can be striped across logical sectors.

Table I summarizes the physical parameters of MEMS-based storage used in our research, based on the predicted characteristics of the CMU **G2** MEMS model. While the exact performance numbers depend upon the details of that specification, the techniques themselves do not.

3. MODELING MEMS-BASED STORAGE

Because MEMS-based storage is still under development, our simulations are based on current proposed device architectures and specifications. Using these specifications, we

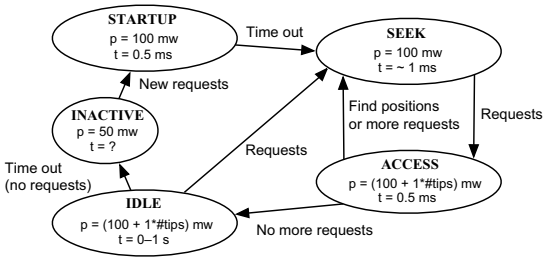


Fig. 3. Power states in which a MEMS-based storage device can operate.

have developed models for MEMS storage device power consumption characteristics and positioning behaviors of MEMS-based storage.

3.1 Modeling Power Consumption

As specified elsewhere [Carley et al. 2000; Schlosser et al. 2000], when a MEMS device transfers data the active probe tips and their signal processing electronics consume 1 mW each. Keeping the media sled in motion requires an additional 100 mW. Thus, a data transfer using 1000 active tips would require $(1000 \times 1) + 100 = 1100$ mW. When idle, the media sled constantly seeks to and rereads the last sector accessed, consuming the same power as when transferring data. When placed in an inactive mode, the device consumes only 50 mW (the sled is "spun down"), but takes 0.5 ms to move from inactive to active; during this time, the sled consumes only slightly more power than it does during the active mode. The overall energy usage for MEMS-based storage is much lower than disk.

We derived a state transition diagram describing the MEMS device power usage from these specifications. A MEMS storage device can be in one of five states, as shown in Figure 3: INACTIVE, in which the device remains stationary, with perhaps a small power drain to hold it in place; IDLE, in which the sled is not servicing requests but keeps seeking to the last sector using a few tips to access servo information; SEEK, in which the actuators move the sled and all tips are inactive; ACCESS, in which the sled moves in the y direction and necessary tips are activated to access servo information and data; and STARTUP, in which the device switches from INACTIVE to SEEK within 0.5 ms and the cost to "spin up" the sled is 5×10^{-5} J, assuming no active tip is required during this switch. The duration of time before the device switches from IDLE to INACTIVE is called the *idle timeout*, which can be adjusted statically or dynamically. In comparison, disk drives have four major power modes [Li et al. 1994]: OFF, in which the device is completely inactive and consumes no energy; SLEEP, in which the disk is powered up but the platter is not spinning; IDLE, in which the disk is spinning but no data is being transferred; and ACTIVE, in which the disk is spinning and either seeking or accessing data.

From the perspective of power state transitions, the key difference between MEMS storage and disk technologies is that the transition from INACTIVE to SEEK in MEMS-based storage is much more cost-effective, in terms of time and power consumption, than the transition from SLEEP to IDLE in disk. INACTIVE and SLEEP are the lowest power states for the two kinds of devices before they are shut down, and SEEK and IDLE are the states that they need to be in before servicing requests. In particular, MEMS only consumes slightly more power during this transition than it does in IDLE because the MEMS media sled is very light and it only needs accelerating to a relatively low speed (28 mm/s),

requiring a tiny amount of extra kinetic energy ($\sim 10^{-7}$ J). Disks may take several seconds to accelerate relatively large and heavy rotating platters to speeds exceeding 3600 RPM. During this transition, disks require power several times higher than in the IDLE mode.

3.2 Modeling Positioning Time

An accurate and tractable positioning time model is important for understanding performance characteristics of MEMS-based storage. We developed an analytical MEMS positioning model, taking into account the external force (constant but bidirectional, $\pm F$), the spring force, and the initial and final access velocities, which are opposite for odd and even-indexed bit columns. The model is similar to a positioning time model proposed by CMU [Griffin et al. 2000a]. In contrast, the CMU model assumed piece-wise constant spring forces and was solved iteratively. As a result, its accuracy and computational complexity depend on the granularity of the iteration. Our model has no such limitation.

In MEMS-based storage, media sled positioning in the x and y dimensions can proceed in parallel because the actuation mechanisms and control loops in x and y are independent. Therefore,

$$t_{seek} = \max(t_x, t_y), \tag{1}$$

where t_{seek} is the overall seek time and t_x and t_y are the seek times in x and y. The sled seek in x consists of a base seek plus a settling time, which is a function of the resonant frequency of the system. The sled seek in y consists of a base seek plus any necessary turnaround time, which is a function of the actuator and spring forces. Both the settling time and turnaround time can be calculated from the physical parameters of MEMS-based storage.

A base seek is a one-dimensional movement from x_0 to x_1, where the movement and the access velocities are in the same direction. The seek consists of two phases: acceleration and deceleration. The actuators accelerate the sled toward the destination in the acceleration phase and reverse polarity and decelerate the sled to its final destination and velocity in the deceleration phase. In addition to the actuator force, the sled springs constantly pull the sled toward its center-most position. Because the kinetic energy of the sled is unchanged at the beginning and the end of the base seek, we know when and where to reverse the polarity of the actuators:

$$x_m = \frac{x_0 + x_1}{2} + \frac{k}{4F}(x_1^2 - x_0^2), \tag{2}$$

where x_m is the position at which actuators reverse polarity, from positive to negative, k is the spring constant, and F is the actuator force.

The phases of acceleration and deceleration in the base seek are described in Equations 3 and 4:

$$\ddot{x} = a - \frac{kx}{m}, \tag{3}$$

$$\ddot{x} = -a - \frac{kx}{m}, \tag{4}$$

where m is the sled mass, a is the acceleration by the actuators, and x is the sled displacement. Given their marginal conditions, we can solve Equations 3 and 4 (Complete solutions are available in a technical report [Hong and Brandt 2002]). Using the physical parameters in Table I, we can very accurately estimate the seek time between any two positions in the sled.

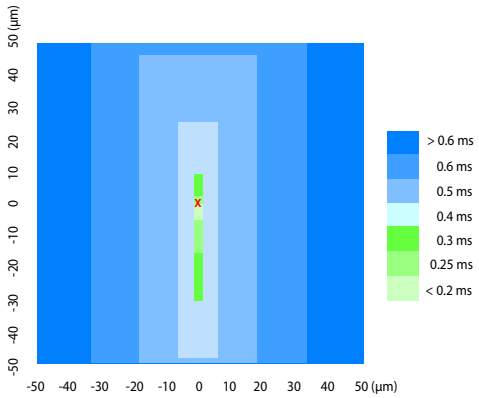


Fig. 4. Seek time equivalence regions from the center to even-indexed bit columns.

4. REQUEST SCHEDULING OF MEMS-BASED STORAGE

Researchers have been optimizing request scheduling to improve disk efficiency for decades [Jacobson and Wilkes 1992; Seltzer et al. 1990; Worthington et al. 1994]. By mapping a disk-analogous data layout on MEMS-based storage (Section 2), Schlosser and Griffin *et al.* [Griffin et al. 2000a; 2000b; Schlosser and Ganger 2004; Schlosser et al. 2000] concluded that existing disk request scheduling algorithms can be applied efficiently on MEMS, sometimes even better than a MEMS-specific Shortest (Euclidean) Distance First algorithm. These algorithms implicitly assumed that the seek cost in the x dimension is either infinite or equal to the cost in y.

These assumptions about the cost ratio of seeks in the x and y dimensions are not true for MEMS-based storage. Figure 4 shows the seek times from the center of a MEMS sled to its even-indexed bit columns, where the *seek time equivalence regions* are rectangular with an x:y size ratio of on average about 1:10. This means that it is cheaper to move one unit of distance in the x direction than to move more than ten units in the y direction. This is in direct contrast to hard drives, where most algorithms implicitly assume that it is faster to access any sector in the current track than to access any sector in any other track. These results suggest that scheduling algorithms that take advantage of knowledge of this ratio are likely to outperform those based on the disk-based model of tip cylinders, which assumes an implicit ratio of 1:∞.

4.1 The Zone-based Shortest Positioning Time First Algorithm

It is known that Shortest Positioning Time First (SPTF) scheduling generally provides the best performance but suffers high response time variations [Jacobson and Wilkes 1992; Seltzer et al. 1990; Worthington et al. 1994]. We developed Zone-based Shortest Positioning Time First (ZSPTF), an algorithm that divides the MEMS storage media into a set of *zones* based on seek time equivalence regions. Zones are serviced in a fixed order to guarantee fairness, *e.g.* in a C-SCAN (Circular Scan) order, which prevents a large number of new requests in lower-numbered zones from indefinitely delaying the service of requests in higher-numbered zones. Multiple requests within a zone are serviced in a SPTF order, which reduces the average seek distance, and thus the average response time, by grouping nearby requests together. Once all of the pending requests in the current zone have been

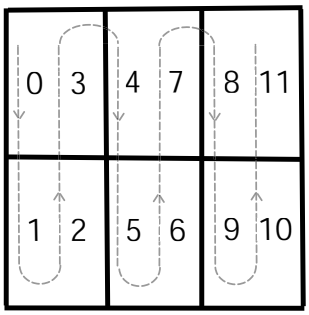


Fig. 5. An example of partitioning the MEMS storage area and the traversed order. The sled is divided into 12 zones.

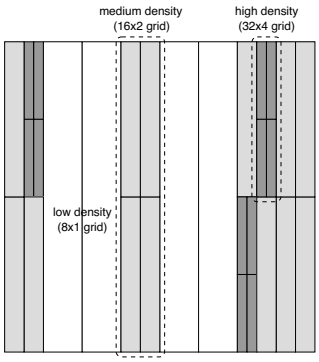


Fig. 6. Fractal breakup of the MEMS device grid.

serviced, the algorithm moves on to the next zone with pending requests. Figure 5 shows an example of partitioning the MEMS storage media into 12 zones and the order in which zones are traversed. Note that each cell in Figure 5 contains two zones because ZSPTF divides even-indexed and odd-indexed bit columns, which have opposite access directions, into different zones.

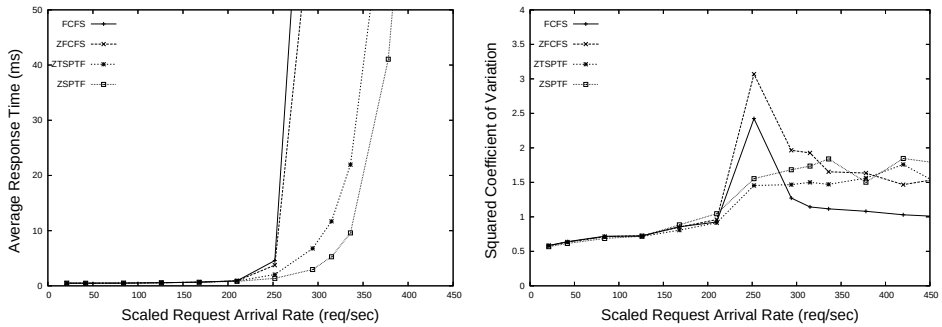
ZSPTF is computationally simpler than SPTF and its variants. SPTF must recompute the positioning time of each request in its queue after fulfilling one request, and the time needed to do so is proportional to the queue length. With longer queue lengths, this recomputation may not be practical. In ZSPTF, the average queue length of each zone will on average be much less than that of SPTF. Because zones are small and based on seek time equivalence regions, this recalculation may not be necessary after each request.

The in-zone (local) optimization of ZSPTF may become less efficient than the global optimization of SPTF when there are too few requests in each zone. To address this issue, we have developed a variable-sized zoning technique we call *pyramiding*. Instead of optimizing within fixed-sized zones, pyramiding merges nearby zones with too few requests and schedules all requests from the merged zones together.

Figure 6 illustrates an example of pyramiding in ZSPTF. In this example, ZSPTF can function as either a 8×1, 16×2, or 32×4 grid depending on the request rate. Although the instantaneous request rate is difficult to measure, we can use the request queue length to approximate this rate and to determine the granularity of the grid. With aggressive pyramiding policies, ZSPTF tends to behave like SPTF because it merges as many zones as possible into one larger zone. Pyramiding improves the performance of ZSPTF at the cost of slightly higher variability. In fact, pyramiding can provide a spectrum of performance, variability, and computational complexity between ZSPTF and SPTF.

4.2 Experimental Analysis

We implemented our scheduling algorithms in DiskSim [Ganger et al. 1999], which has been used in earlier studies on MEMS seek algorithms [Griffin et al. 2000b; Schlosser and Ganger 2004; Schlosser et al. 2000]. We used two one-hour workloads to exercise the simulator. *Cello* is a news server workload and *hplajw* is a user workload, both of which are from Hewlett-Packard Laboratories. The average request arrival rate is 33.0 requests



(a) Average response times (b) Response time variations

Fig. 7. Different in-zone scheduling algorithms on the user trace.

per second for *cello* and 20.9 requests per second for *hplajw*. In general, *cello* and *hplajw* are characterized by random accesses and highly sequential accesses, and *hplajw* is more bursty than *cello*. We scaled the traced inter-arrival times to increase the request arrival rates and explore a range of workload intensities.

In addition to the average response time, another important metric for request scheduling algorithms is the squared coefficient of variation of response times (σ^2/μ^2), where σ is the standard deviation of response times and μ is the average response time. This metric measures the consistency of request response times [Worthington et al. 1994], providing a measure of fairness and starvation resistance.

4.2.1 *Comparison of In-Zone Algorithms.* The essence of zone-based algorithms is to partition the MEMS storage media into a set of seek-time-constrained regions and service these regions in a fixed order. One question that arises is what scheduling algorithm to use within each zone. To determine this we implemented several zone-based scheduling algorithms, including Zone-based FCFS (ZFCFS), ZSPTF, and an arrival-time-constrained version of ZSPTF (ZTSPTF), and evaluate their performance and fairness. ZTSPTF improves the fairness of ZSPTF in a way similar to FCFS by only servicing requests that arrive before the scheduler enters a zone.

We found that a good in-zone algorithm is still critical to avoid unnecessary settling and turnaround times even though the regions are considered to be seek-time-equivalent. ZSPTF performs much better than ZFCFS. It provides 7–13% higher throughput than ZTSPTF at the same response time level under the moderately- to heavily-scaled *hplajw* user workloads, but their overall starvation resistance is quite similar, as shown in Figures 7(a) and 7(b). Both of them have very similar performance and fairness under the random *cello* server workloads. In short ZSPTF has the best performance with reasonable variability among these algorithms.

4.2.2 *Comparison of Different Zone Sizes.* The number of zones into which the device is divided will affect the performance of ZSPTF. Fewer zones leads to longer average queue lengths for each zone, allowing SPTF to perform better but causing higher variability of queueing times. Larger numbers of zones leads to lower variability, *i.e.* greater fairness and starvation resistance, but higher average seek times.

As illustrated in Figure 4, seek time equivalence regions, *i.e.* zones, are rectangular with an x:y size ratio of around 1:10. By choosing an in-zone seek time threshold from 0.3

ms, 0.4 ms, or no constraint, ZSPTF divides a MEMS storage device into 320, 80, or 20 zones, respectively. Simulations showed that ZSPTF exhibits similar performance and fairness under different zone sizes for the randomly-accessed *cello* server workload. For the sequentially-accessed *hplajw* user workload, the smallest zone size marginally improves the starvation resistance of ZSPTF but significantly degrades its performance. ZSPTF performs better under the largest zone size than under the medium zone size by on average 5% when the workload intensity is scaled up; however, the squared coefficient of variation of ZSPTF is also increased by up to 142%. Therefore, we empirically chose the medium zone size, which partitions the media into 80 zones, because it provides a good trade-off between efficiency and fairness.

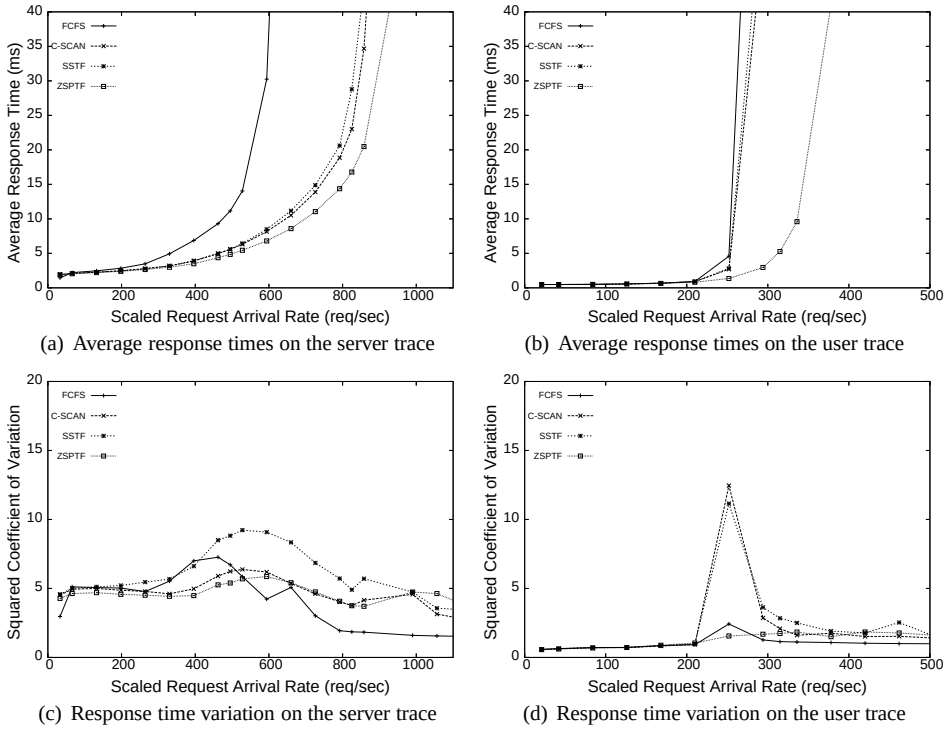
4.3 Comparison of ZSPTF with Existing Algorithms

Disk request scheduling algorithms can be adapted to MEMS-based storage once MEMS devices are mapped onto a disk-like interface. Our comparisons focus on six: FCFS, C-SCAN, Shortest Seek Time First (SSTF), SPTF, Aged SPTF (ASPTF) [Jacobson and Wilkes 1992], and Grouped SPTF (GSPTF) [Seltzer et al. 1990]. FCFS has good performance only under light workloads, but we include it here as a baseline for comparison. C-SCAN services requests in ascending logical block number (LBN) order. SSTF uses the number of tracks between the last accessed LBN and the desired LBN as an estimate of the seek time. SPTF always services the request with the smallest positioning delay from the current position, explicitly considering seek times in both x and y dimensions for MEMS. Besides positioning times, ASPTF also considers request queueing delays. The aging factor w of ASPTF can be varied from zero (pure SPTF) to infinity (pure FCFS). As the request queue builds up, the queueing delay becomes a dominant factor in ASPTF so the algorithm tends to behave like FCFS. Experimentally, we chose $w = 5$ because ASPTF(5) exhibits good performance and fairness across a range of request rates.

Grouped SPTF divides the device logical address space into a set of regions, each containing consecutive logical blocks. It groups outstanding requests in a region and schedules them together in a SPTF order. GSPTF is conceptually similar to Zone-based SPTF, both of which address the fairness and scheduling efficiency problems of SPTF by partitioning the request queue based on the spatial locality of the requests. ZSPTF measures the spatial locality by the actual physical request locations, as illustrated by two-dimensional *zones*; in contrast, GSPTF measures it by the linear logical block number of the request. Under the current disk-analogous MEMS data layout, a zone in ZSPTF contains a set of non-consecutive logical blocks; however, a group in GSPTF always contains consecutive logical blocks. This difference has its performance and fairness implications on ZSPTF and GSPTF, as we will show in Figures 9(b) and 9(d). For fair comparison, we also partitioned the MEMS device into 80 regions in GSPTF.

Pyramiding improves the performance of ZSPTF by dynamically merging nearby zones with too few requests. The performance, variability, and computational complexity of ZSPTF depend upon the aggressiveness of pyramiding, *i.e.* the queue length threshold selection. We empirically chose small queue lengths, of 4–8 requests, for good trade-offs between performance and fairness.

Figure 8(a)–8(d) show the average response times and squared coefficients of variation of response times of FCFS, C-SCAN, SSTF, and ZSPTF under a range of scaled workload intensities for the *cello* server and *hplajw* user traces. As expected, all the algorithms have similar response times and fairness under light workloads. ZSPTF significantly outper-



(a) Average response times on the server trace

(b) Average response times on the user trace

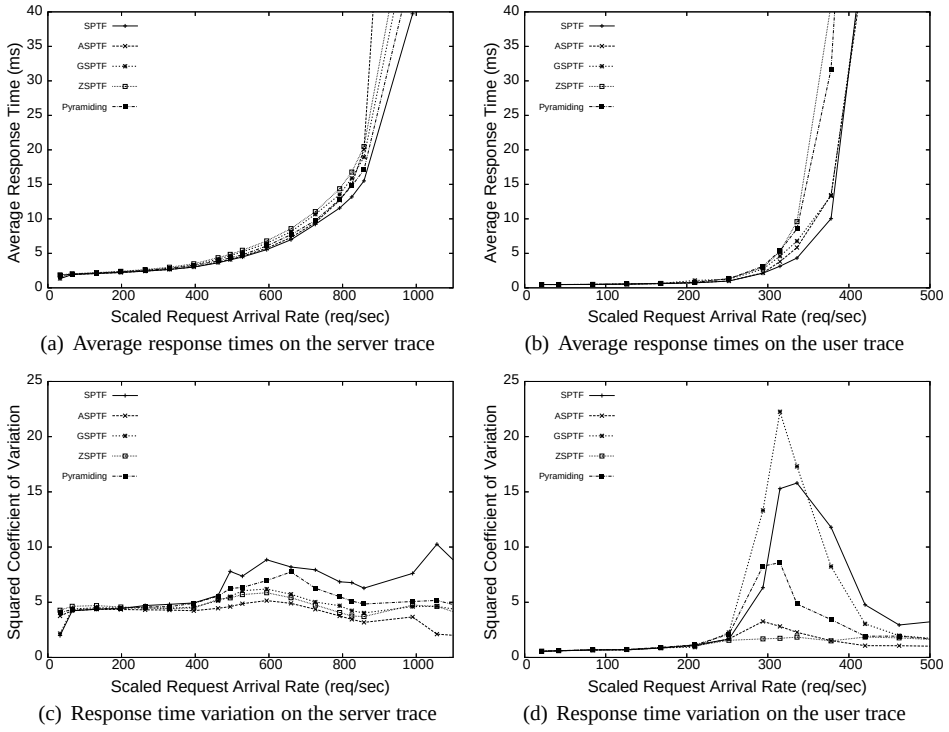
(c) Response time variation on the server trace

(d) Response time variation on the user trace

Fig. 8. Performance comparison of ZSPTF and non-SPTF-based scheduling algorithms.

forms C-SCAN and SSTF, providing 13–32% higher throughput under moderately- and heavily-scaled workloads. In general, ZSPTF and C-SCAN have similar service fairness because they traverse the media area in a similar order. Both of them have lower service variability than SSTF.

Figure 9(a)–9(d) show the performance and fairness of SPTF-based scheduling algorithms under the scaled server and user workloads, including SPTF, ASPTF, GSPTF, ZSPTF, and pyramiding. SPTF always has the best performance but suffers high variability. GSPTF provides a slightly higher throughput than ZSPTF under the *cello* server workload. SPTF and ASPTF perform better than ZSPTF by 7–10% higher throughput under moderately-scaled workloads, in which the queues of SPTF and ASPTF are long enough for effective optimization but the average in-zone queue length of ZSPTF is not. Pyramiding alleviates this problem and achieves better performance than ZSPTF for all request arrival rates. Under heavy *cello* server workloads, ZSPTF provides 6–9% higher throughput than ASPTF. This is due to the fact that ASPTF degrades to FCFS as the queueing delay increases. However, SPTF, ASPTF, and GSPTF perform better than ZSPTF by 7-12% under moderate and heavy *hplajw* user workloads because ZSPTF cannot take advantage of the sequentiality of the user workload as much as SPTF, ASPTF, and GSPTF. The reason is that the data layout was done assuming a disk-based model rather than a MEMS-based model. This inherently punishes algorithms that employ a MEMS-based model. MEMS-specific data layouts should further enhance the benefit of MEMS-specific request scheduling algorithms; in the future we plan to examine such layout schemes.



(a) Average response times on the server trace

(b) Average response times on the user trace

(c) Response time variation on the server trace

(d) Response time variation on the user trace

Fig. 9. Performance comparison of ZSPTF and other SPTF-based scheduling algorithms.

SPTF suffers high response time variability. ASPTF usually provides the best service fairness among the five SPTF-based scheduling algorithms, which is often better than C-SCAN. ZSPTF has 8–42% higher squared coefficients of variation than ASPTF. GSPTF has up to 7% higher variability than ZSPTF under the *cello* server workload. It exhibits large response time variations under the moderately-scaled, highly sequential *hplajw* user workloads. Although GSPTF intends to improve the variability of SPTF by partitioning the request queue and only servicing requests within a region at one time, a steady request stream to the region can prevent the scheduler from moving to other regions, resulting in potential starvation. This effect is illustrated in the highly sequential user workload. Instead of containing consecutive logical blocks, a zone in ZSPTF consists of a set of non-consecutive logical blocks that are physically mapped to the device closely. Therefore, the potential of starvation under sequential workloads is greatly reduced. The variability of pyramiding ranges between the variability of ZSPTF to SPTF as does its average response time. Essentially, both ZSPTF and pyramiding constrain the scheduling scope based on request locations to guarantee service fairness and pyramiding has looser geographical restrictions than ZSPTF. Thus, pyramiding can provide better performance than ZSPTF but with the cost of higher variability.

Although ASPTF has the overall best performance, its high computational cost prevents its use in real systems. To quantify this, we ran some experiments on a Linux machine (Pentium IV 2.6 GHz) and found that SPTF and ASPTF take 1.8 μs for each entry when reordering the request queue. This means that for a queue length of 300 requests SPTF and

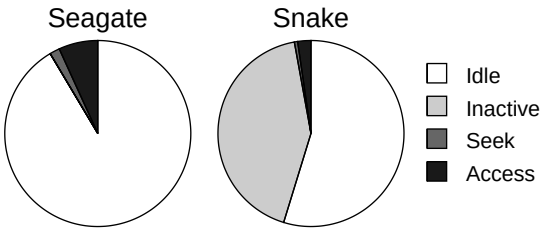


Fig. 10. Power consumption distributions in the *Seagate* and *Snake* workloads

ASPTF take more than 0.5 ms to determine which request to service next, which is equal to the average seek time of MEMS devices. Therefore, it is impractical to apply SPTF and ASPTF in real systems. By contrast, the queue length of each zone in ZSPTF is on average 10–100 times smaller and the computational cost of computing SPTF within each zone is consequently limited and tolerable.

By leveraging MEMS-specific seek time equivalence regions, ZSPTF significantly outperforms other disk-analogous scheduling algorithms. Based on its superior average response time, low coefficient of variation, and low computational cost, we believe that ZSPTF is an ideal algorithm for use with MEMS-based storage devices.

5. POWER CONSERVATION OF MEMS-BASED STORAGE

Storage devices account for a major portion of the total energy consumption in many mobile computing systems [Douglis et al. 1994; Li et al. 1994]. In such systems, energy efficiency largely determines the amount of time that they can operate autonomously. A MEMS storage device is expected to consume only a small fraction of power that a disk requires. However, the power efficiency index of MEMS-based storage, as defined by watt per gigabyte, is still about tens of milliwatts per gigabyte of storage because of its limited capacity per device. This is comparable to the index of low-power disks [Hitachi Global Storage Technologies 2004; Seagate Technology, Inc. 2004]. Consequently, effective power management is still important for MEMS-based storage.

5.1 Distribution of Power Consumption

As with disk drives, the energy consumption of MEMS storage devices depends on the workloads they serve. We empirically examined the power consumption distributions of MEMS-based storage under two workloads using a detailed storage subsystem simulator, DiskSim [Ganger et al. 1999]. The idle timeout was set to be 1 second. *Seagate* is a validation workload of a Seagate ST41601N disk drive and *Snake* is a one-day HP-UX file server trace [Ruemmler and Wilkes 1993]. We replayed the *Snake* trace 10 times faster than it was collected to increase the workload intensity.

Figure 10 shows that MEMS devices consume 55–91% of the total energy in IDLE under these workloads. In the IDLE state, the sled still keeps moving and active tips keep accessing servo information even though there are no outstanding requests in the queue, wasting a significant amount of energy. Energy consumed in SEEK and ACCESS account for 3–9% of the total energy consumption. Energy consumed in INACTIVE cannot be further reduced without changing the device itself because there is no lower-power state.

5.2 Power Conservation Strategies

MEMS storage devices have a very efficient transition between INACTIVE and SEEK, making it feasible to aggressively "spin-down" such devices. Accesses in MEMS devices are more flexible because different read/write heads may be turned on or off as needed to allow for variable-sized accesses. Thus, a smart scheduler can rearrange request streams for better power utilization. These device-specific features motivate three simple and effective power conservation strategies for MEMS-based storage, mainly addressing energy consumption in IDLE, SEEK, and ACCESS, which are not applicable to disks.

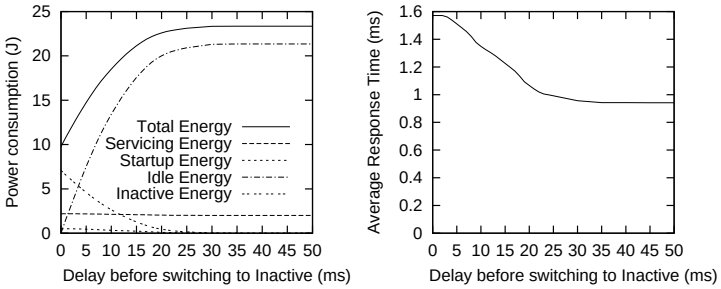
5.2.1 *Aggressive Spin-down.* Aggressive spin-down deactivates the MEMS device to its lowest-power state, INACTIVE, when there are no requests in the queue, completely avoiding the IDLE state and reducing overall power consumption accordingly. The trade-off is increased I/O latencies for requests that arrive when the device is in INACTIVE.

Aggressive spin-down is usually not the best solution for disk drives because of the high cost, in terms of both time and energy, associated with spinning up a disk to over 3600 RPM before it can service new requests. Spinning-up a disk from its low-power state to its active state may require more energy than if the disk was always in the active state [Li et al. 1994]. Therefore, fixed or adaptive heuristics are used to decide when to spin down the disk [Golding et al. 1995; Helmbold et al. 1996; Li et al. 1994]. This motivation, however, is not as applicable to MEMS-based storage. Unlike disks, MEMS-based storage has fast and energy-efficient state transitions, as described in Section 3.1. By fully exploiting device idle periods, aggressive spin-down can potentially achieve the best power conservation.

5.2.2 *Request Merging.* MEMS storage devices spend a considerable amount of energy seeking to and accessing data. Any reduction in seek and access energy is likely to carry with it corresponding reductions in request response times, which can mitigate the performance penalty of aggressive spin-down.

Many workloads, including mobile computing workloads, exhibit strong sequential data access patterns. In MEMS devices, logical sectors are mapped to physical tip sectors in a way that allows logically sequential sectors to be accessed together. Unlike disk, MEMS-based storage can turn on hundreds to thousands of read/write tips at one time. It may be feasible for MEMS devices to merge sequential requests in the queue by simultaneously activating more tips to fulfill the combined larger request. In addition, servicing separate requests together can reduce seek and data transfer overheads. By leveraging the increased parallelism and bandwidth, this method can reduce both energy consumption and request response times.

5.2.3 *Subsector Accesses.* In MEMS storage devices, a 512 byte logical sector consists of a group of 64 tip sectors from the same position of separate tip regions, each of which contains 10 bits of servo information and 80 bits of ECC-encoded data [Schlosser et al. 2000]. Accessing subsectors in MEMS is feasible because error correcting codes can be computed over data striped across multiple tips. Active tips dissipate considerably more power than the moving sled during data transfers and MEMS devices can possibly adjust their power consumption by only reading or writing necessary subsectors instead of standard 512 byte sectors. In contrast, disks cannot access regions of data smaller than 512 bytes due in part to the need to read the ECC data associated with each sector. Subsector accesses cannot improve response times; however, they can reduce the number of active



(a) Power consumption (b) Average response time

Fig. 11. Effect of idle timeout.

tips required for individual requests, providing more available tips that may be used for request merging, thereby enhancing its effectiveness.

5.3 Experimental Results

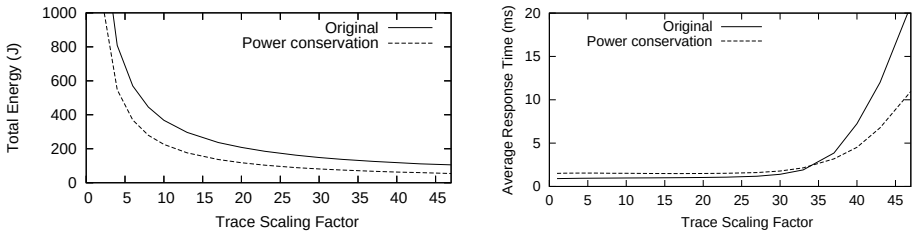
We implemented all three power conservation strategies in DiskSim [Ganger et al. 1999]. The default MEMS physical parameters are shown in Table I. We used the *Seagate* and *Snake* traces, as described in Section 5.1, to exercise the simulator.

To explore a range of workload intensities, we scaled the traced inter-arrival times to produce a range of average inter-arrival times. A scaling factor of one corresponds to replaying the trace at its original speed, a scaling factor of two corresponds to halving the traced inter-arrival times and replaying the trace twice as fast, and so on.

5.3.1 *Aggressive Spin-down.* We studied the effects of spin-down thresholds by using different delays before switching MEMS-based storage devices from IDLE to INACTIVE. By decreasing the idle timeout from 40 ms to 0 ms, energy consumption can be reduced by more than 50% under the *Seagate* workload, as shown in Figure 11(a). In contrast to disks, for which the longest and shortest timeouts consume more energy than intermediate timeout values [Helmbold et al. 1996], aggressive spin-down with zero delay achieves the lowest energy consumption. This is because the energy cost associated with spinning-up a MEMS sled is very low thanks to its light mass and relatively slow motion.

Aggressive spin-down inevitably adds extra device spin-up latencies, about 0.5 ms, to requests that arrive when the device is in INACTIVE. This performance penalty is insignificant when the workload is intensive because the request queue is unlikely to be empty, as shown in Figure 11(b). During light loads, we believe that this added response time of 0.5 ms will not be noticed by users. Thus, aggressive spin-down with no delay is simple and probably the best choice for MEMS-based storage.

5.3.2 *Request Merging.* A general request merging mechanism is a challenging problem. We adopted a straightforward strategy of combining queued read requests with sequential logical sectors. To avoid unnecessary performance penalty, an idle MEMS device is activated immediately once there are outstanding requests in the queue. This technique can be easily implemented in the controller logic. For the purpose of research, we increased the number of concurrently active tips from 1280 to 3200 without changing other physical parameters of the CMU **G2** MEMS model. Consequently, the MEMS device can service requests with sizes up to 25 KB, instead of 10 KB, at one time.



(a) Total energy consumed (b) Average response time

Fig. 12. Combined effect of all power conservation strategies.

This simple merging method is quite effective for the sequential *Snake* workload for both energy saving and performance improvement, because request merging results in fewer MEMS sled motions and faster data transfers. It saves about 10–18% of the seek and access energy, and reduces the average response time by up to 43% when the request queue builds up as the trace scaling factor increases, as shown in Figure 12(b).

5.3.3 *Subsector Accesses.* Accessing subsectors to save power can be useful for workloads with small I/Os, such as metadata-intensive workloads and parallel I/Os. Besides the ability of MEMS-based storage to activate only a few necessary tips for data transfers, this techniques also relies upon the cooperation of operating systems to provide detailed location and size information about the desired data; otherwise, a naive system will access an entire (large) sector even if it would be possible to access a small amount.

Metadata accesses are typically small: a UNIX-like i-node consists of 128 bytes, only one fourth of the standard 512 byte logical sector; the most-frequently-accessed fields in the i-node can be even as small as four bytes. Due to the lack of detailed information on metadata accesses in the current block-level disk traces, we varied the amount of requested data in each i-node access from four bytes to 128 bytes (full i-node access) and evaluated the power consumptions accordingly. Subsector accesses can save 29–45% of the access energy in the *Snake* workload, in which 72% of I/O accesses are to metadata. Because access energy only accounts for a small fraction of the total energy consumption, the overall energy saving by this technique is marginal for *Snake*.

Power becomes one of the most critical resources in the increasingly important mobile and embedded computing environments, where data are often accessed in much smaller pieces than in a general-purpose computing environment. Subsector access demonstrates additional energy saving potential of MEMS storage devices for those environments. It is worthwhile for operating systems designers and application developers to fully exploit this feature of MEMS storage, making the system more power friendly.

5.3.4 *Combining the Three Strategies.* We evaluated the overall effect of the three power conservation strategies using *Snake* under different workload intensities. By applying all three methods the total energy consumption can be reduced by half, as shown in Figure 12(a). The reduction is mostly due to aggressive spin-down, which in turn increases average request response times by up to 0.5 ms due to extra spin-up overheads, especially when the workload intensity is low. As the workload intensity increases, this effect is less significant because the request queue is unlikely to be empty. Request merging reduces request delays by servicing seperately-issued sequential requests at one time, and provides 43% better performance under heavily-scaled *Snake* workloads, as shown in Figure 12(b).

In short, using these three strategies we can obtain more than 50% reduction in power consumption with an acceptable performance penalty under light workloads and with even better performance under heavy workloads.

6. RELATED WORK

MEMS-based storage is an alternative secondary storage technology currently being developed. Besides CMU [Carley et al. 2000], IBM has developed a prototype device, called *Millipede* [Vettiger et al. 2000] that, unlike the magnetic recording of the CMU design, writes data by moving probe tips in the z direction and making tiny physical marks on the media. Additional hardware research is also being done at Hewlett-Packard Laboratories [Toigo 2000]. Recently, there has been interest in modeling MEMS storage device behavior [Griffin et al. 2000a; Madhyastha and Yang 2001]. Parameterized MEMS performance prediction models [Dramaliev and Madhyastha 2003; Sivan-Zimet and Madhyastha 2002] were also proposed to narrow the design space of MEMS-based storage.

Schlosser and Ganger [Schlosser and Ganger 2004] suggested that roles and policies proposed for MEMS-based storage should be examined under two objective tests, *specificity* and *merit*, focusing on the use of MEMS-specific features and potential performance benefits, respectively. By comparing performance of MEMS devices and hypothetical "super" disks, they concluded that MEMS storage devices are much like disks, except for their efficient accesses for two-dimensional data structures. Our research demonstrates that significant benefits can be obtained on power management and request scheduling by leveraging the MEMS-specific low-level device details, such as fast power state transitions and unique two-dimensional seeks.

6.1 Request Scheduling

Although MEMS-based storage has unique two-dimensional seek behaviors, Schlosser *et al.* [Griffin et al. 2000a; 2000b; Schlosser et al. 2000; Schlosser and Ganger 2004] showed that disk-analogous one-dimensional scheduling, such as SSTF and C-LOOK, can be applied efficiently, even better than Shortest (Euclidean) Distance First (SDF) scheduling, by placing data on MEMS devices in longitudinally sequential tracks, similar to tracks on disks. These algorithms assume that the seek cost in the x dimension is either infinite (for SSTF and C-LOOK) or equal to the cost in y (for SDF). Our detailed modeling and analyses of MEMS seek times provides a more accurate x:y cost ratio. By leveraging this more detailed device-specific knowledge, ZSPTF achieves even better performance.

6.2 Power Management for Storage Subsystems

MEMS-based storage is expected to consume much less power than disks. Griffin and Schlosser *et al.* [Griffin et al. 2000b; Schlosser et al. 2000] compared the energy consumptions of MEMS and disks. They also suggested powering down the sled and using fewer tips when possible; however, they did not evaluate the effects of these techniques. Our research illustrated that the fast and energy-efficient transition from standby to active in MEMS makes aggressive spin-down a simple and probably the best "spin-down" policy for MEMS devices. In contrast, the spin-down threshold for disks requires dynamic changes to adapt user behaviors and priorities to achieve better performance and energy saving [Douglis et al. 1995; Golding et al. 1995; Helmbold et al. 1996; Li et al. 1994].

7. FUTURE WORK

Aggressive spin-down can be combined with delayed spin-up to further reduce energy consumption by lengthening device inactive periods. This may lead to longer queue lengths and thus longer queuing delays. However, smarter request merging schemes can alleviate this problem and even further improve power efficiency by coalescing more requests, increasing sequentiality of interleaved request streams by queue reordering, and exploiting the high degree of parallelism in data accesses of MEMS storage. Subsector accesses on MEMS devices can dramatically reduce power consumption during data transfers. It is feasible in many parallel file systems, where the average request size is quite small (less than 512 bytes) [Nieuwejaar et al. 1996]. Similarly, personal digital assistants and other handheld devices may routinely manage smaller chunks of data, providing greater opportunity for both request merging and subsector accesses.

Data layout is one of the central considerations in system designs. File systems have long clustered related data together; for example, the Berkeley Fast File System [McKusick et al. 1984] lays out data in cylinder groups. We believe that grouping related data within zones, similar to a method suggested by Schlosser *et al.* [Griffin et al. 2000b], and using a zone-based scheduling algorithm will provide considerable performance improvement over mechanisms based on disk-analogous logical block numbers.

8. CONCLUSIONS

As an emerging secondary storage technology, MEMS-based storage promises high storage density, high bandwidth, low access latency, and low power consumption. As a result of its architectural designs, MEMS-based storage has two-dimensional positioning mechanisms, low-cost power state transitions, and variable-sized data accesses. Existing request scheduling algorithms and power management policies designed for disks are unaware of these device-specific characteristics and thus cannot exploit the full potentials of MEMS-based storage. We have shown that device management policies that take advantage of the unique features of MEMS-based storage can provide even better performance.

We developed an analytical positioning time model for MEMS-based storage and identified seek time equivalence regions on MEMS devices. This information allows a MEMS-specific request scheduling algorithm, ZSPTF, to achieve efficient request reordering locally while keeping good fairness as a whole. We also found that aggressively switching a MEMS device to its lowest power state can obtains the best power savings without significant performance penalties, because power state transitions in MEMS devices are very efficient. The performance degradation can largely mitigated by MEMS servicing multiple requests simultaneously, thanks to its variable-sized data accesses.

ACKNOWLEDGMENTS

We are grateful to Greg Ganger, David Nagle, and the Parallel Data Laboratory at Carnegie Mellon for their help in our research. We also thank John Wilkes of Hewlett-Packard Laboratories for providing us with traces.

This research is supported by the National Science Foundation under grant number CCR-0073509 and the Institute for Scientific Computation Research at Lawrence Livermore National Laboratory under grant number SC-20010378. Additional support for the Storage Systems Research Center was provided by Engenio, Hewlett-Packard Laboratories, Hitachi Global Storage Technologies, IBM Research, Intel, Microsoft Research,

Network Appliance, and Veritas.

REFERENCES

CARLEY, L., BAIN, J., FEDDER, G., GREVE, D., GUILLOU, D., LU, M., MUKHERJEE, T., SANTHANAM, S., ABELMANN, L., AND MIN, S. 2000. Single-chip computers with microelectromechanical systems-based magnetic memory. *Journal of Applied Physics 87,* 9 (May), 6680–6685.

DOUGLIS, F., CÁCERES, R., KAASHOEK, F., LI, K., MARSH, B., AND TAUBER, J. A. 1994. Storage alternatives for mobile computers. In *Proceedings of the 1st Symposium on Operating Systems Design and Implementation (OSDI)*. Monterey, CA, 25–37.

DOUGLIS, F., KRISHNAN, P., AND BERSHAD, B. 1995. Adaptive disk spin-down policies for mobile computers. In *Proceedings of the Second USENIX Symposium on Mobile and Location-Independent Computing*. USENIX, 121–137.

DRAMALIEV, I. AND MADHYASTHA, T. 2003. Optimizing probe-based storage. In *Proceedings of the Second USENIX Conference on File and Storage Technologies (FAST)*. USENIX Association, San Francisco, CA, 103–114.

GANGER, G. R., WORTHINGTON, B. L., AND PATT, Y. N. 1999. The DiskSim simulation environment version 2.0 reference manual. Tech. rep., Carnegie Mellon University / University of Michigan. Dec.

GOLDING, R., BOSCH, P., STAELIN, C., SULLIVAN, T., AND WILKES, J. 1995. Idleness is not sloth. In *Proceedings of the Winter 1995 USENIX Technical Conference*. USENIX, New Orleans, LA, 201–212.

GRIFFIN, J. L., SCHLOSSER, S. W., GANGER, G. R., AND NAGLE, D. F. 2000a. Modeling and performance of MEMS-based storage devices. In *Proceedings of the 2000 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. ACM Press, Santa Clara, CA, 56–65.

GRIFFIN, J. L., SCHLOSSER, S. W., GANGER, G. R., AND NAGLE, D. F. 2000b. Operating system management of MEMS-based storage devices. In *Proceedings of the 4th Symposium on Operating Systems Design and Implementation (OSDI)*. USENIX Association, San Diego, CA, 227–242.

HELMBOLD, D. P., LONG, D. D. E., AND SHERROD, B. 1996. A dynamic disk spin-down technique for mobile computing. In *Proceedings of the 2nd Annual International Conference on Mobile Computing and Networking 1996 (MOBICOM '96)*. ACM, Rye, New York, 130–142.

HITACHI GLOBAL STORAGE TECHNOLOGIES. 2004. Hitachi Disc Product Datasheets. http://www.hgst.com/.

HONG, B. AND BRANDT, S. A. 2002. An analytical solution to a MEMS seek time model. Tech. Rep. UCSC-CRL-02-31, Storage Systems Research Center, University of California, Santa Cruz. Sept.

JACOBSON, D. M. AND WILKES, J. 1992. Disk scheduling algorithms based on rotational position. Tech. Rep. HPL-CSP-91-7rev1, Hewlett-Packard Laboratories, Concurrent Systems Project. Mar.

LI, K., KUMPF, R., HORTON, P., AND ANDERSON, T. 1994. A quantitative analysis of disk drive power management in portable computers. In *Proceedings of the Winter 1994 USENIX Technical Conference*. San Francisco, CA, 279–291.

LUMB, C. R., SCHINDLER, J., AND GANGER, G. R. 2002. Freeblock scheduling outside of disk firmware. In *Proceedings of the 2002 Conference on File and Storage Technologies (FAST)*. Monterey, CA.

MADHYASTHA, T. AND YANG, K. P. 2001. Physical modeling of probe-based storage. In *Proceedings of the 18th IEEE Symposium on Mass Storage Systems and Technologies*. IEEE Computer Society Press, Monterey, CA, 207–224.

MCKUSICK, M. K., JOY, W. N., LEFFLER, S. J., AND FABRY, R. S. 1984. A fast file system for UNIX. *ACM Transactions on Computer Systems 2,* 3 (Aug.), 181–197.

NIEUWEJAAR, N., KOTZ, D., PURAKAYASTHA, A., ELLIS, C. S., AND BEST, M. 1996. File-access characteristics of parallel scientific workloads. *IEEE Transactions on Parallel and Distributed Systems 7,* 10 (Oct.), 1075–1089.

ROSENBLUM, M. AND OUSTERHOUT, J. K. 1992. The design and implementation of a log-structured file system. *ACM Transactions on Computer Systems 10,* 1 (Feb.), 26–52.

RUEMMLER, C. AND WILKES, J. 1993. Unix disk access patterns. In *Proceedings of the Winter 1993 USENIX Technical Conference*. USENIX Association, San Diego, CA, 405–420.

SCHINDLER, J., GRIFFIN, J. L., LUMB, C. R., AND GANGER, G. R. 2002. Track-aligned extents: Matching access patterns to disk drive characteristics. In *Proceedings of the 2002 Conference on File and Storage Technologies (FAST)*. Monterey, CA.

SCHLOSSER, S. W. AND GANGER, G. R. 2004. MEMS-based storage devices and standard disk interfaces: A square peg in a round hole? In *Proceedings of the Third USENIX Conference on File and Storage Technologies (FAST)*. USENIX, USENIX Association, San Francisco, CA, 87–100.

SCHLOSSER, S. W., GRIFFIN, J. L., NAGLE, D. F., AND GANGER, G. R. 2000. Designing computer systems with MEMS-based storage. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. ACM Press, Cambridge, MA, 1–12.

SEAGATE TECHNOLOGY, INC. 2004. Seagate Disc Product Datasheets. http://www.seagate.com/products/datasheet/.

SELTZER, M., CHEN, P., AND OUSTERHOUT, J. 1990. Disk scheduling revisited. In *Proceedings of the Winter 1990 USENIX Technical Conference*. 313–323.

SIVAN-ZIMET, M. AND MADHYASTHA, T. M. 2002. Workload based modeling of probe-based storage. In *Proceedings of the 2002 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. ACM Press, Marina Del Rey, CA, 256–257.

TOIGO, J. W. 2000. Avoiding a data crunch – A decade away: Atomic resolution storage. *Scientific American 282*, 5 (May), 58–74.

UYSAL, M., MERCHANT, A., AND ALVAREZ, G. A. 2003. Using MEMS-based storage in disk arrays. In *Proceedings of the Second USENIX Conference on File and Storage Technologies (FAST)*. USENIX Association, San Francisco, CA, 89–101.

VETTIGER, P., DESPONT, M., DRECHSLER, U., URIG, U., ABERLE, W., LUTWYCHE, M., ROTHUIZEN, H., STUTZ, R., WIDMER, R., AND BINNIG, G. 2000. The "Millipede"—More than one thousand tips for future AFM data storage. *IBM Journal of Research and Development 44*, 3, 323–340.

WORTHINGTON, B., GANGER, G., AND PATT, Y. 1994. Scheduling algorithms for modern disk drives. In *Proceedings of the 1994 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. 241–251.

YU, H., AGRAWAL, D., AND ABBADI, A. E. 2003. Tabular placement of relational data on MEMS-based storage devices. In *Proceedings of the 29th Conference on Very Large Databases (VLDB)*. Morgan Kaufmann, Berlin, German, 680–693.

Received December 2004; revised December 2004; accepted December 2004