Power Conservation Strategies for MEMS-based Storage Devices[†]

Ying Lin Scott A. Brandt Darrell D. E. Long Ethan L. Miller

Storage Systems Research Center
Jack Baskin School of Engineering
University of California, Santa Cruz

Abstract

Power dissipation by storage systems in mobile computers accounts for a large percentage of the power consumed by the entire system. Reducing the power used by the storage device is crucial for reducing overall power consumption. A new class of secondary storage devices based on microelectromechanical systems (MEMS) promises to consume an order of magnitude less power with 10–20 times shorter latency and 10 times greater storage densities. We describe three strategies to reduce power consumption: aggressive spin-down, sequential request merging, and subsector accesses. We show that aggressive spin-down can save up to 50% of the total energy consumed by the device at the cost of increased response time. Merging of sequential requests can save up to 18% of the servicing energy and reduce response time by about 20%. Transferring less data for small requests such as those for metadata can save 40% of the servicing energy. Finally, we show that by applying all three power management strategies simultaneously the total power consumption of MEMS-based storage devices can be reduced by about 54% with no impact on I/O performance.

1 Introduction

As a result of limitations in battery technology, mobile computing devices such as laptop computers, personal digital assistants (PDAs), video camcorders and biomedical monitoring devices must be designed to be energy efficient in order to extend the amount of time that they can operate autonomously. Storage devices account for a major portion of total energy consumption in many of these devices, consuming as much as 20–54% of the power in mobile computers [2, 12]. However, a new class of secondary storage devices based on microelectromechanical systems (MEMS) [1, 15, 20], currently being developed, promises to reduce power use by an order of magnitude compared to disk.

MEMS devices provide non-volatile storage using either physical [20] or magnetic [1] recording techniques to achieve extremely high density storage. To achieve these high densities, MEMS-based storage designs use a non-rotating storage device with storage media on one surface and a large array of read/write heads on another surface directly above the storage media (see Figure 1). By moving the surfaces relative to each other using actuators, each read/write head can access a region of the surface. Access latencies in these small devices are much lower than traditional secondary storage devices because physical movements to locate data are extremely small. Though the bit rate off each read/write head is slow compared to the single read/write head in a disk, overall throughput is increased by having many read/write heads active at once. Initial density estimates are in the range of 10 gigabits per square inch with densities anticipated to reach 300 gigabits per square inch in a few years [8, 19].

MEMS storage devices are expected to consume an order of magnitude less power than disk drives [8]. These devices are expected to have many other significant advantages over disk, including better I/O performance, smaller physical size, lower heat dissipation requirements, and integrated processing and storage [19]. For all of these reasons, MEMS based storage devices are expected to be used in mobile computing applications where power and size are important.

MEMS-based storage has several features that lead to lower power consumption. First, the moving sled in a MEMS-based storage device has much less mass than a disk platter and thus takes less power to move. Equally important, MEMS devices have a much faster transition between active and inactive modes—about 0.5 ms, as compared to several seconds in a traditional disk drive. The electronics of MEMS devices also require less power for read or write operations. Furthermore, accesses are more flexible than in

[†]This research is supported by the National Science Foundation under grant number CCR-073509 and the Institute for Scientific Computation Research at Lawrence Livermore National Laboratory under grant number SC-20010378.

This paper was published in the *Proceedings of the Tenth IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS 2002)*, Fort Worth, TX, October 2002.

disk drives because different read/write heads can be turned on or off as needed to allow for variable-sized accesses to the device.

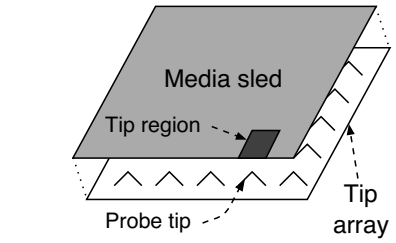
Power conservation strategies for disk drives such as adaptive spin-down [9, 10, 12], caching [2], and non-volatile solid-state memory [2, 22] have proven useful in reducing power consumption. Some of these schemes may not be applicable to MEMS storage devices due to its distinctive features, while others such as caching may be used to further reduce power consumption regardless of the secondary storage device. Additionally, new strategies that may not be feasible in disk drives can be considered in MEMS storage devices. For example, accessing partial sectors is not supported in disk drives but MEMS storage devices may be able to conserve power by using small transfers for metadata and other small requests.

Using the DiskSim [4] MEMS simulator, we investigated the interaction between power consumption and I/O performance based on file system traces [18]. Our experimental results show that the power consumption pattern of MEMS is different from that of conventional disk drives. Based on this analysis, we propose three MEMS power conservation strategies: *aggressive spin-down*, *sequential request merging* and *subsector accesses*. Our experimental results using the DiskSim MEMS simulator show that aggressive spin-down can save 50% of the total energy consumed by the device, potentially at the cost of increasing response time. Merging sequential requests can save up to 18% of the servicing energy and reduce response time by about 20%, and using subsector accesses for metadata requests can save 40% of the energy used to service file system requests. By applying all three power management strategies simultaneously, we show that MEMS-based storage devices can reduce total power usage by about 54%, with no impact on I/O performance.

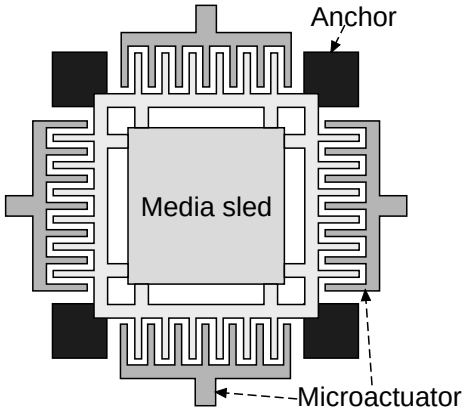
2 Background

Mobile secondary storage technology is dominated by magnetic disks, but the mechanical characteristics of disks limit their performance improvement. Moreover, the superparamagnetic effect will make it increasingly harder for improvements in disk technology to keep up with the improvements in processor and memory. A new storage technology based on Microelectromechanical Systems (MEMS) is being developed with significant performance and cost improvement relative to magnetic disks. A more complete description of these devices can be found elsewhere [7, 8, 19]; we will summarize the design and performance characteristics of MEMS-based storage devices.

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



(a) Probes and media sled



(b) Media sled and actuators

Figure 1. Components of a MEMS-based storage device

stationary read/write probe tips, called a *tip array*, as shown in Figure 1(a). Figure 1(b) shows the media sled suspended above the tip substrate by silicon beams that act as springs, and moved in the *x* and *y* directions by forces generated by lateral resonant microactuators. A request is satisfied by first moving the sled to the correct position and then transferring data. To perform a seek, the microactuators move the media sled to a specific (*x*,*y*) position. Once the seek is complete, data is accessed by the fixed probe tips while the media sled is moving at a constant velocity in the *y* direction. In contrast to disk, the tips remain nearly stationary during a seek, with the exception of minor *x* and *z* dimension adjustments [8], while the sled keeps moving in the *y* direction.

In contrast to disk, a MEMS-based storage device can use multiple tips simultaneously to access data, achieving a high degree of parallelism. Because of power, heat, and wiring limitations, not all tips can be active at the same time. For instance, in the CMU G2 model with 6400 probe tips, only 1280 tips can be active at once, while in the G3 model, 3200 tips are expected to be active simultaneously [19].

The media sled is logically divided into rectangles called *tip regions*; each tip region is accessible by a single read/write probe tip. The smallest unit of accessible data is a

tip sector, consisting of servo information (10 bits) and encoded data/ECC (8 bytes of data encoded as 80 bits). Each tip sector is represented as a triple $\langle x, y, tip \rangle$, where x and y are position coordinates and tip is a tip number. Groups (or tip sets) of 64 tip sectors from the same position of separate regions are combined into 512 byte *logical sectors*, analogous to logical blocks in a hard drive. In keeping with disk terminology, the CMU MEMS research group applies a direct low level data layout, such as tracks, cylinders etc., to MEMS-based storage device. The details of logical-to-physical mapping are discussed in Section 5.

3 Related Work

Prior research in optimization of power management strategies for disks has been effective because disks can be in several states that correspond to different power consumption rates [6, 12]. The goal of power management is to have a disk move between these states in order to minimize power with little or no performance penalty. A common scheme is to spin down drives during extended periods of non-use. A simple spin-down policy uses a fixed threshold to determine when to spin down the disk [3, 6, 12], reducing energy use by as much as a factor of 10 over not spinning the disk down at all. Using short spin-down delays (2–5 seconds of idle time) was more effective than longer spin-down delays (3–5 minutes) at reducing energy consumption, at the cost of several spin-up delays per hour [6].

Wilkes proposed the use of a predictive algorithm for disk management [21] by predicting and adjusting the spin-down delay based on a weighted average of recent activity durations. Golding, *et al.* placed this proposal in a more general framework and studied a number of spin-down delay prediction methods, including arithmetic and geometric adjustments of predicted interarrival times [5]. Douglis, *et al.* used predictive spin-down algorithms depending on previous reference patterns, but expressed disappointment with their results [3]. Later, they varied the spin-down threshold dynamically by adapting to the user's access patterns and priorities [12]. Their *adaptive spin-down* method can reduce the number of spin-ups by up to 50% in some conditions. Helmbold and Long, *et al.* also used prediction combined with a simple and efficient machine learning technique, the *share algorithm*, to determine when to spin the disk down [10]. Their algorithm outperforms any fixed timeout and reduces disk power consumption by about 50% over the energy consumed by a one-minute fixed time-out.

MEMS-based storage devices have been suggested as a possible alternative to disks. They are expected to provide higher storage densities with much lower operational energy. With no rotating parts and lighter mass, their unique physical characteristics enable much simpler and efficient power management, as described by researchers at Carnegie Mellon's Parallel Data Lab [7, 8, 19]. They suggest powering down the sled when the request queue is empty and using fewer tips when possible; however, they do not evaluate the effects of these techniques on MEMS power consumption, response time, and bandwidth, instead comparing MEMS behavior using these techniques to that of a disk. Madhyastha, *et al.* have also modeled the low-level behavior of MEMS-based storage devices [13].

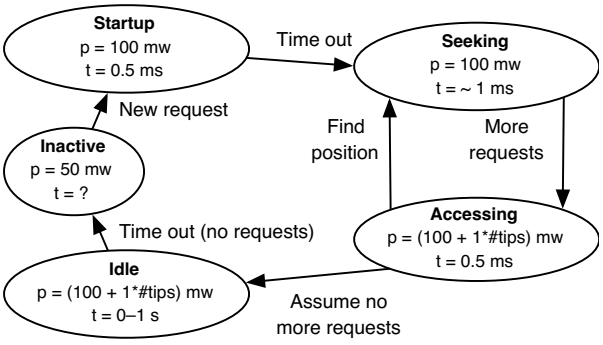
4 Power Model

Before proposing techniques for reducing power consumption, it is first necessary to discuss how energy is consumed in a MEMS-based storage device. Since MEMS-based storage is still in its infancy, some details of the device's power model remain unknown. We have derived a state transition diagram describing MEMS device power usage from specifications in earlier work [1, 19], and will use this model to motivate our schemes to reduce power usage.

4.1 Power Model States

In order to determine how much energy can be saved by power management, the power utilized by the device during normal operations needs to be modeled. During data transfer, each active probe tip and its signal processing electronics consume 1 mW, and keeping the sled in motion requires 100 mW [1, 19]. Thus, a data transfer using 1000 active tips would require $(1000 \times 1)+100=1100$ mW. When the device is in standby mode (the sled is "spun down") power consumption is estimated to be only 50 mW. It takes 0.5 ms for the sled to move from standby to active; during this time, the sled consumes power as it does during active mode. This is in contrast to disks that require several seconds and a large amount of energy to spin up. Based on these guidelines, our model allows a MEMS-based storage device to be in one of five states: **Inactive**, **Startup**, **Seeking**, **Accessing**, and **Idle**, as described in Figure 2.

In comparison, disk drives have four major power modes [12]: **OFF**, in which the device is completely inactive and consumes no energy; **SLEEP**, in which the disk is powered up by 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. The key difference in the state diagram between disks and MEMS-based storage is the much shorter and lower energy transition between **SLEEP** and **IDLE** for MEMS devices. Additionally, overall energy usage for MEMS-based devices is lower because they do not need to rotate a relatively large platter at speeds exceeding 3600 RPM.



Inactive	The device is not moving, with perhaps a small power drain to hold it in place.
Startup	When new requests arrive and the device is in the **Inactive** state, the device switches to the **Seeking** state. Assuming the switch time is 0.5 ms (5×10^{-4} s) and no tips need to be active, the cost to "spin up" the sled is $sled\_active\_power \times startup\_time = (10^{-1})(5 \times 10^{-4}) = 5 \times 10^{-5}$ J.
Seeking	The actuators are moving the sled. In this mode, only the sled contributes to power consumption because all tips are inactive.
Accessing	The sled is moving and any necessary tips are activated to access servo information and data. Thus, both the sled and active tips use power.
Idle	When there are no active requests in the queue, the device is idle. During this time, the sled keeps seeking to the last sector using some tips to access servo information. The duration of time before the device switches to **Inactive** is called the *idle timeout*; this time can be adjusted statically or dynamically.

Figure 2. Power states in which a MEMS-based storage device can operate

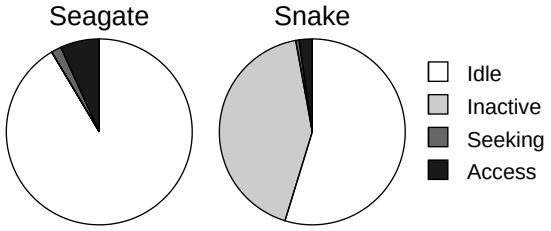


Figure 3. Power consumption distribution in Seagate and Snake workloads

4.2 Distribution of Power Consumption

We ran experiments on two workloads to find out how much power a MEMS storage device consumes in each of the five states. Our experiments used the CMU G2 MEMS-based device, with idle timeout set to 1 second. One trace workload, **Seagate**, is a validation workload of a Seagate ST41601N disk drive and is included with the DiskSim source code distribution. The other workload, **Snake**, was collected from an HP-UX file server in 1992 [18]. We used a one-day subset of the **Snake** workload with 77,372 requests to conduct our experiments.

Table 1 and Figure 3 show the power consumed by the

Table 1. Power consumption distribution in the Seagate and Snake workloads

Energy (joules)	Seagate		Snake	
Startup	5×10^{-5}	$<< 1\%$	0.02	$<< 1\%$
Inactive	10^{-5}	$<< 1\%$	153.4	42.4%
Servicing:				
- Seeking	0.41	1.8%	2.06	0.6%
- Servo	0.18	0.8%	0.92	0.3%
- Transferring data	1.41	6.1%	7.39	2.0%
Servicing (total)	2.00	8.6%	10.37	2.9%
Idle	21.35	91.4%	198.07	54.7%
Total	23.35	100.0%	361.82	100.0%

NOTE: *The Snake trace had a scale factor of 0.1. The* scale factor *of a trace is the value by which request start times are multiplied; thus, a scale factor of 0.1 means a trace that is replayed 10 times faster than the original trace.*

MEMS device in each of the five states under the two workloads. Total servicing energy includes power consumed in seeking, accessing servo information and accessing data. In the Seagate workload, energy consumed in **Idle** dominates the system power consumption, consuming about 90% of the total energy. In the idle state, even though there are no active requests in the queue, the sled still keeps moving and active tips keep accessing servo information until the timeout expires. If the sled could spin down as soon as it completed the last request before the idle period, this idle time power consumption could be eliminated or reduced. In the Snake workload, requests are further apart than in the Seagate workload; thus, the device often goes into the **Inactive** state. As a result, only 55% of the energy is consumed in the **Idle** state, and another 42% is consumed in the **Inactive** state. Since the **Inactive** state consumes much less power than **Idle**, this indicates that the device is inactive for a large fraction of the trace period.

5 Power Conservation Strategies

In designing our power conservation strategies, we noticed that *startup energy* can be omitted because so little energy is consumed moving the device from **Inactive** to **Idle**. Energy consumed when the device is in the **Inactive** state cannot be further reduced without changing the device itself because there is no lower-power state. Energy spent in the **Seek** state could be reduced; however, doing so would require reorganizing the data on the MEMS device and is beyond the scope of this paper. This leaves two states in which power savings can be realized: **Idle** and **Accessing**.

After evaluating the existing power conservation approaches and taking the characteristics of MEMS-based storage into consideration, we have developed three strate-

gies for power conservation in MEMS-based storage devices: aggressive spin-down[1], request merging, and subsector access. Aggressive spin-down eliminates idle energy but slightly increases service time. Request merging and subsector access reduce servicing energy at the same time as they reduce service time sufficiently to cancel out the performance cost of aggressive spin-down. Each of these techniques is described in more detail in the following sections.

5.1 Aggressive Spin-down

Aggressive spin-down deactivates the storage device when there are no requests in the queue, completely avoiding the idle state. This approach eliminates energy consumed in **Idle** by spending all inactive time in **Inactive**, reducing overall power consumption accordingly. The tradeoff in this policy is increased I/O latency for requests that arrive when the device is in **Inactive**.

Aggressive spin-down is usually not the best solution for a disk because the delay to spin up from the inactive to idle state is much higher than access latency in the idle state and spin-up energy may exceed the energy saved by spinning down if the next request arrives too soon. In particular, spinning the disk down immediately after each access is likely to use more energy than is saved because it is so time-consuming and energy-intensive to accelerate a disk to over 3600 RPM. Instead, fixed or adaptive heuristics are used to decide when to spin down the disk. This motivation, however, is not as applicable to MEMS-based devices. As Section 4.1 shows, only 10^{-5} J is consumed when a MEMS device starts up, and it only takes 0.5 ms for this to take place. In comparison, a disk drive can take 2 seconds and 6 J to spin up, and 1 second to spin down [12]. Since idle power consumption for a disk is 1 W, spinning a disk down and back up again requires as much energy as leaving it idle for 6 seconds. The comparable figure for a MEMS-based storage device is 50 ms—the time it takes to switch from **Inactive** to **Idle**. In other words, a MEMS-based device saves energy if spun down when it will be inactive for more than 50 ms. The benefits of this scheme and potential tradeoffs in performance are discussed in Section 6.1.

5.2 Request Merging

Servicing energy ranks second after idle energy in terms of power consumption. Any reduction in servicing energy is likely to carry with it a corresponding reduction in servicing time, which can help to mitigate the performance cost of the aggressive spin-down discussed in the previous section. The choice of scheduling algorithms and data placement, though they may significant affect I/O performance, only influence energy expended during seek—a small fraction of the overall servicing energy. Thus, modified scheduling schemes are unlikely to significantly reduce servicing energy or overall power consumption.

[1] "Spin-down" refers to stopping the rotation of a disk platter. MEMS devices do not have rotating media; nonetheless, we will continue to use the term "spin-down" to refer to the transition between **Idle** and **Inactive**.

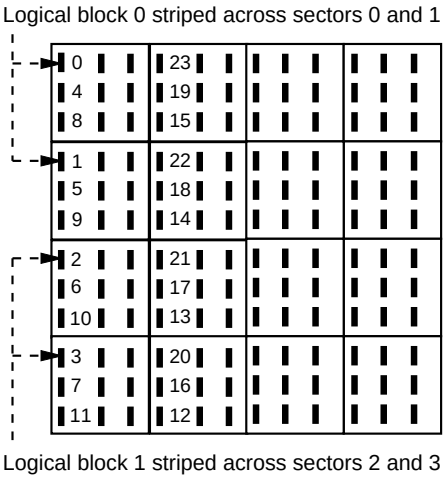


Figure 4. Example of MEMS device low-level data layout

The first technique we propose for reducing servicing energy is request merging, which enhances I/O performance while reducing power consumption. In MEMS devices, the mapping of logical block numbers to physical location is optimized for sequential accesses [8]. Sequential logical blocks are mapped to 64 corresponding sectors from adjacent tip sets, allowing logically sequential blocks to be accessed simultaneously. Because many I/O requests are sequential and very bursty, sequential I/O requests in the queue can be merged by simultaneously activating more tips to service the combined larger request with shorter time and less energy. This method increases the parallelism of concurrent active tips to increase I/O bandwidth and reduce response time. Moreover, it can reduce the energy consumed by reducing the number of sled motions required.

Figure 4 shows a simple example in which it is assumed that one logical block contains only two tip sectors. Suppose a request in the queue is for logical block 0, and the next request is for logical block 1. These two blocks can be read in parallel by merging since the four tips are in the same positions.

In MEMS-based devices, when the tips access data, the media sled moves in the *y* direction. If we do not merge these two requests, then after fulfilling the first request, the sled has to move back to the start point of logical block 1 to handle the second request. The original strategy incurs more seeking overhead and thus increases the response time. With request merging, requests 1 and 2 can be completed simultaneously. This scheme saves power and im-

proves performance by using a single sled motion to satisfy two requests.

Currently, the DiskSim MEMS simulator does not provide request merging; a general mechanism for merging requests is a complicated and challenging problem. We adopted a simple policy for our experiments: if there are two adjacent requests in the I/O queue to sequential locations and both are reads, they are combined into a single large request that replaces the original two requests. We expect that additional gains can be achieved using more advanced merging schemes such as combining requests that differ by a small number of blocks and are accessible by non-contiguous sets of tips with the same sled motion.

5.3 Subsector Accesses

MEMS-based storage devices have the ability to adjust their power consumption during data accesses by reading or writing at a smaller granularity than standard 512 byte blocks. Since active tips dissipate considerably more power than the moving sled during data transfers, reading or writing only the necessary data could save power.

As mentioned in Section 2, a tip sector consists of servo information (10 bits) and encoded data/ECC (80 bits of encoded data) [19]. Groups (or tip sets) of 64 tip sectors from the same position of separate regions are combined into 512 byte logical sectors, analogous to logical blocks in a hard disk. In a disk, it is impossible to access regions of data smaller than 512 bytes, due in part to the need to read the ECC associated with each block. For efficiency reasons, the sector size is also limited by the seek and rotational latencies which would make smaller block accesses less efficient. However, accessing subsectors in MEMS is feasible because the error correcting codes can be computed over data striped across multiple tips. During a request, only the necessary subsectors need be accessed and corresponding tip sets activated; unused tips can remain inactive to conserve power. Moreover, the sizes of the subsectors may be varied on demand.

As a simplified example, if the last block of a request is logical block 0 in Figure 4, the useful data of the last block may only be stored in tip 0. In this situation, tip 1 is useless and should be left inactive. Using subsector accesses, only tips transferring data will be active, saving the power that would have been used by the other tips.

6 Experimental Results

We simulated all three of our power management schemes using DiskSim [4] and the CMU MEMS-based storage device model [6, 19]. DiskSim is a well-validated disk simulator that includes several commercial disk models as well as models for other devices such as MEMS-based storage.

As described in Section 4.2, we used two trace files in our experiments. The Seagate trace file uses relative timestamps while the other, the Snake trace, contains absolute timestamps. The relative timestamps in the Seagate trace are useful in measuring the influence on throughput and response time, while the absolute timestamps in the Snake trace can be used to measure the queued requests. Thus, we used the Seagate trace file to test aggressive spin-down and the Snake trace file to experiment with request merging and subsector accesses. We also combined the three methods together and tested their overall effect on both power consumption and I/O performance on the Snake trace.

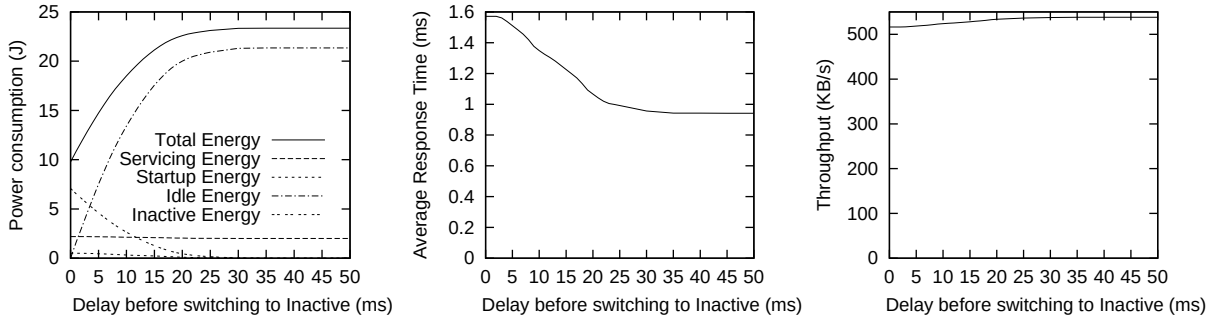
6.1 Aggressive Spin-down

We first simulated the effect of inactivating the MEMS storage device on energy consumption and performance. We simulated different delays before switching the MEMS device from **Idle** to **Inactive**. As expected, power consumption is lowest when the device is switched to inactive mode as soon as the the request queue is empty, as shown in Figure 5(a). This is in contrast to disks, for which the longest and shortest time-outs consume more energy than intermediate time-out values [10].

Figure 5(a) shows the effect on power consumption of increasing the idle time-out from 0 ms to 40 ms. As expected, overall power consumption is lowest when the device is switched immediately to the **Inactive** state, and is twice as high with a 40 ms time-out. With a longer idle timeout, the MEMS-based device must wait longer before switching to the power-conserving **Inactive** state, spending more time in the power-hungry **Idle** state. Power consumption is reduced by 50% by using a time-out of 0 ms and switching immediately to **Inactive** when there are no outstanding requests.

Quickly switching to the **Inactive** state reduces power consumption, but may also increase response time and reduce bandwidth by increasing the latency of any request arriving while the sled is inactive. We measured this effect using two metrics: average response time and average throughput. Figure 5(b) shows that as the time-out decreases from 40 ms to 0 ms, average response time increases by about 66%. Note that the extra response time is about 0.5 ms, which is the startup timeout, indicating that most requests have a startup overhead when the sled is aggressively spun down. Therefore, the average response time increases by the startup cost. Since the response time only increases by 0.5 ms, however, we believe that this added response time will not be noticed by users as long as overall bandwidth is not affected.

Fortunately, an aggressive spin-down policy does not



(a) Effect of idle timeout on power (b) Effect of idle timeout on response time (c) Effect of idle timeout on throughput

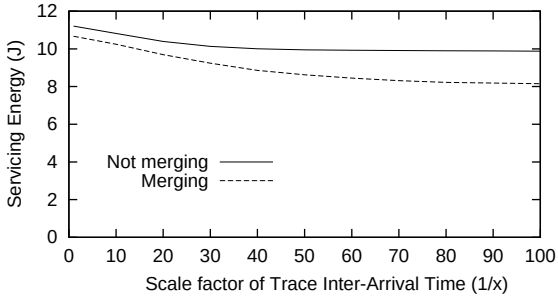
Figure 5. Effect of idle timeout

significantly affect throughput, as Figure 5(c) shows. As the time-out decreases from 40 ms to 0 ms, throughput only drops by about 4%. For this workload, bandwidth is thus largely unaffected by aggressive spin-down, and latency increases by 0.5 ms, hardly noticeable for most users. Thus, we believe that aggressive spin-down with no delay is the best choice for MEMS-based storage devices.

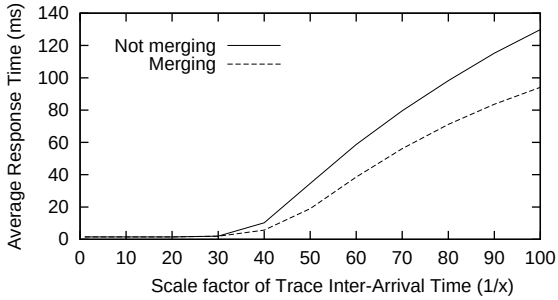
6.2 Request Merging

Request merging requires that a MEMS device allow sufficient tips to be active to read two sectors at the same time. The average request size in the Snake workload is 8 KB, requiring that $16 \times 64 = 1024$ tips be activated at the same time. Since a G3 device allows 3200 tips to be active, more than twice as many tips to be active as needed for a single 8 KB request, two adjacent requests can usually be merged together.

We tested the straightforward strategy of combining adjacent requests in the queue when the requests contain sequential logical blocks using the Snake workload. Figure 6(a) shows that this simple merging method saves up to 18% of the servicing energy for longer request queues. Figure 6(b) shows that merging sequential requests also reduces average response time by about 20%. Improvements in both response time and servicing energy show that using request merging on MEMS-based storage devices is a good idea.



(a) Effect of request merging on servicing energy



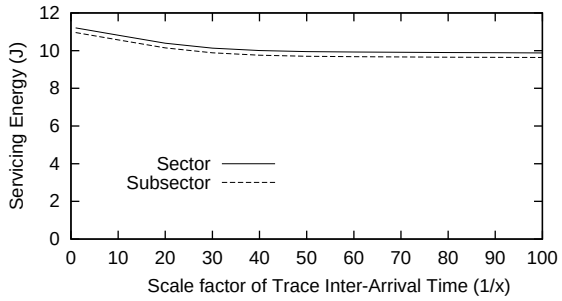
(b) Effect of request merging on response time

Figure 6. Effect of request merging

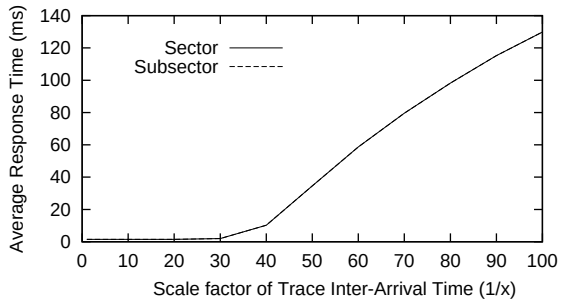
6.3 Subsector Accesses

The third power reduction scheme, accessing subsectors, relies upon the MEMS device's ability to use relatively few tips to access the desired data. This technique is only effective if the operating system knows about the device's ability, however, since a naive system will write an entire (large) sector even if it would be possible to write a small amount. This is reflected in the traces, which were gathered on systems with a relatively large block size. If the workload were gathered at the file system level, we could treat the last few bytes of a file as a subsector read or write. For disk level workloads, if the request size is specified in terms of bytes, only the last logical block of a request can be accessed as a subsector. When disk level traces are used to simulate the subsector method, we assume that the size of the subsector is uniformly distributed for the last logical block of each request. In other words, the number of active tips is randomly selected from 1 to 64 for the last subsector.

We used the Snake trace and the CMU G3 device parameters to measure the effectiveness of the subsector access approach. Figures 7(a) and 7(b) show the results of using subsector accesses on power consumption and response time as compared with those of the original simu-



(a) Effect of subsector access on servicing energy



(b) Effect of subsector accesses on response time

Figure 7. Effect of subsector accesses

lation. These results illustrate that subsector accesses can only save about 3% of the servicing energy and have no effect on average response time. However, the improvement is not significant because the Snake trace is a disk level trace and its average request size is 8KB, which includes 16 logical blocks. Only the last logical block will exploit the subsector approach. Hence, only 1/32, about 3%, of the data accessing energy is saved.

Although these results are not exciting, this approach may also be useful for other types of small I/Os, such as metadata requests, and parallel I/Os. Previous studies have shown that only 13–41% of I/O accesses are to user data while over 50% are to metadata [17, 18]. Metadata requests are small, and will benefit greatly from subsector accesses because the device need not read an entire 512 byte logical sector to retrieve a 128 byte inode. This allows the device to use fewer tips, saving power. In the Snake trace subset we used, about 72% of the I/O accesses are to metadata, providing a much opportunity for power savings.

To measure the effect of subsector accesses for metadata accesses, we ran another set of experiments that reduced the number of tips used for metadata requests. We assumed a 128 byte inode, similar to that used in the Unix Fast File System [14]. Reads of such an inode require 128 bytes, and thus require 16 out of 64 tips to be active. Metadata writes of a new inode similarly require 16 tips to be active. Inode updates, on the other hand, may require significantly less data to be written; the most common update operation

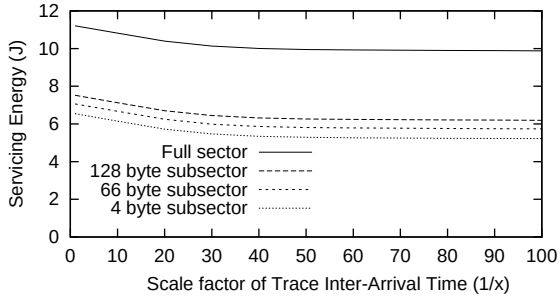


Figure 8. Effect of subsector method applied to metadata requests on servicing energy

in Unix occurs when a file is read, causing the file's access time (a 4 byte value) to be modified. In other cases, however, writes of an existing inode might require all 128 bytes to be updated. Since we did not know the percentage of metadata writes corresponding to each type of update, we considered three mixes of operations: all access time updates at 4 bytes per metadata write, 50% access time updates and 50% full inode writes at $(128+4)/2 = 66$ bytes per metadata write, and all full inode writes at 128 bytes per metadata write.

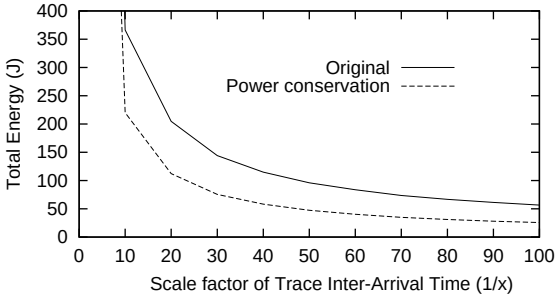
Figure 8 shows that using subsector accesses for metadata dramatically reduces servicing energy in all three scenarios. For the most realistic 50/50 mix, servicing energy is reduced by 33–40%; the other scenarios reduce servicing energy by 29–45%.

Additionally, subsector accesses reduce the number of active tips required for individual requests, providing more available tips that may be used to merge requests. Hence, subsector accesses may enhance the effectiveness of request merging. In the next section, we explore the overall effect by applying all three power conservation approaches together.

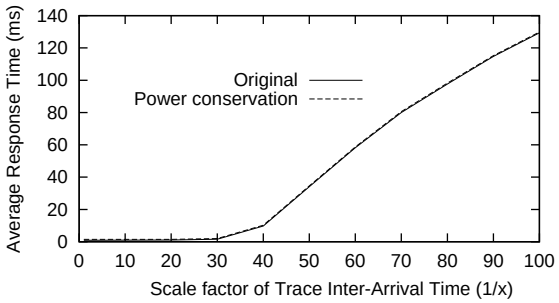
6.4 Combining the Three Strategies

To evaluate the overall effect of the three power conservation strategies, we used the Snake trace to drive experiments based on the G3 MEMS device model. Figure 9(a) shows that by applying all three methods the total energy consumption can be reduced by up to 54%. In this experiment we assumed the 50/50 scenario with an average of 66 bytes per metadata writes for subsector accesses. Since the previous section showed that the change in overall power consumption between 4 byte and 128 byte metadata writes is within 1%, the effect of the actual proportion of old and new metadata writes on overall power consumption is very small.

As discussed in Section 6.1, aggressive spin-down causes average response time to increase due to the additional spin-up overhead. However, request merging reduces



(a) Total energy consumed



(b) Average response time

Figure 9. Combined effect of all power conservation strategies

user delay by servicing sequential requests simultaneously, largely mitigating this effect. Thus, as shown in Figure 9(b), when all the three approaches are applied to MEMS-based storage device together, overall response time is not affected. Thus, these power conservation strategies are particularly appealing because they provide more than 50% reduction in power consumption with no loss in performance.

7 Future Work

Aggressive spin-down can be combined with delayed spin-up to further reduce energy consumption. Instead of restarting as soon as a new request arrives, the sled can wait until several requests queue up or a fixed time has elapsed. The device will spend more time inactive and I/O burstiness will increase, reducing overall power consumption. To some extent, this method avoids unnecessary spin-ups caused by over-aggressive spin-down. Moreover, this technique may result in slightly longer queue lengths, giving request merging more to work with and further improving both performance and power consumption.

Many improvements can be made to the request merging scheme. First, more than two small requests in the queue may be merged. If the MEMS-based device can support more active tips simultaneously in the future, merging more requests is a good way to save more energy. Also, logically sequential requests need not be adjacent in the queue, but can be made adjacent through request reordering. This is especially useful for synchronous I/O workloads where the sequential requests of one process interlace with those of other processes. Using anticipatory scheduling to overcome deceptive idleness in synchronous I/O [11], a larger number of sequential requests from each process can be reordered together and merged together, reducing both seek and access times and boosting I/O performance and energy efficiency. In fact, even the logical sequentiality requirement can be loosened. A very aggressive policy is to merge as many read requests in the queue as possible without exceeding the maximum number of tips that can be activated at the same time. Furthermore, requests can be separated and recombined to achieve a greater degree of parallelism. Our current work only merges read requests—write requests can also be merged. Moreover, if MEMS-based device can support reading and writing at the same time, the merging strategy can consider both read and write requests and still preserve RAW, WAR and WAW data consistency.

Subsectors accesses can also improve power conservation in the future. The trace files used in our experiments are extracted from Unix file system and the average request size is 8 KB, considerably larger than 512 bytes. Applying subsector accesses to the last logical block of each request cannot get a big jump in performance. However, other kinds of file systems such as parallel file systems may produce different results. In many parallel file system traces, for example, the average request size is much smaller (less than 512 bytes) [16], allowing the subsector approach to have a significant improvement in power conservation. Similarly, personal digital assistants and other handheld devices might manage smaller chunks of data, also permitting lower power subblock accesses. An intelligent operating system might further reduce power consumption by noting the specific bytes that change in both metadata and data blocks and only writing back the necessary regions. This approach has the potential to dramatically reduce power consumption by keeping most tips turned off on writes.

8 Conclusions

This paper has described three techniques for reducing overall power consumption by over 50% in a MEMS-based storage device without reducing performance. We accomplished this by reducing the time the device spends idle, making the most of sled movements by combining adjacent requests, and by reducing the number of tips used for small transfers. Our power conservation strategies target the most power-consumptive factors—idle energy and servicing energy. Trace-based experiments show that aggressive spin-down saves 50% of the overall energy, but also increases response time. Merging of sequential requests can save up to 18% of the servicing energy and improve I/O performance.

Though accessing subsectors only saves 3% of the servicing energy based on the last subsector assumption, if it is applied to metadata requests as well over 40% of the servicing energy can be saved. By applying all three power management strategies on the Snake trace, we show that we can reduce power consumption in MEMS-based storage devices by about 54% without suffering any degradation in I/O performance.

Acknowledgments

We are very grateful to David Nagle who first suggested that we pursue this research. We thank Feng Wang and Karen Glocer for their help with the DiskSim simulator, and Zachary Peterson and other members of the Storage Systems Research Center at the University of California, Santa Cruz for their valuable suggestions. Finally, we are grateful to the Parallel Data Lab at Carnegie Mellon for sharing DiskSim and their MEMS hardware specifications with us.

References

[1] L. Carley, J. Bain, G. Fedder, D. Greve, D. Guillou, M. Lu, T. Mukherjee, S. Santhanam, L. Abelmann, and S. Min. Single-chip computers with microelectromechanical systems-based magnetic memory. *Journal of Applied Physics*, 87(9):6680–6685, May 2000.

[2] F. Douglis, R. Cáceres, F. Kaashoek, K. Li, B. Marsh, and J. A. Tauber. Storage alternatives for mobile computers. In *Proceedings of the 1st Symposium on Operating Systems Design and Implementation (OSDI)*, pages 25–37, Monterey, CA, Nov. 1994.

[3] F. Douglis, P. Krishnan, and B. Marsh. Thwarting the power-hungry disk. In *Proceedings of the Winter 1994 USENIX Technical Conference*, pages 292–306, San Francisco, CA, Jan. 1994. USENIX.

[4] G. R. Ganger, B. L. Worthington, and Y. N. Patt. The DiskSim simulation environment version 2.0 reference manual. Technical report, Carnegie Mellon University / University of Michigan, Dec. 1999.

[5] R. Golding, P. Bosch, C. Staelin, T. Sullivan, and J. Wilkes. Idleness is not sloth. In *Proceedings of the Winter 1995 USENIX Technical Conference*, pages 201–212, New Orleans, LA, Jan. 1995. USENIX.

[6] P. M. Greenawalt. Modeling power management for hard disks. In *Proceedings of the 2nd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '94)*, pages 62–66, Durham, NC, Jan. 1994. IEEE.

[7] J. L. Griffin, S. W. Schlosser, G. R. Ganger, and D. F. Nagle. Modeling and performance of MEMS-based storage devices. In *Proceedings of the 2000 SIGMETRICS Conference*, pages 56–65, June 2000.

[8] J. L. Griffin, S. W. Schlosser, G. R. Ganger, and D. F. Nagle. Operating system management of MEMS-based storage devices. In *Proceedings of the 4th Symposium on Operating Systems Design and Implementation (OSDI)*, pages 227–242, Oct. 2000.

[9] D. P. Helmbold, D. D. E. Long, T. L. Sconyers, and B. Sherrod. Adaptive disk spin-down for mobile computers. *ACM/Baltzer Mobile Networks and Applications (MONET)*, 5(4):285–297, 2000.

[10] D. P. Helmbold, D. D. E. Long, and B. Sherrod. 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)*, pages 130–142, Rye, New York, Nov. 1996. ACM.

[11] S. Iyer and P. Druschel. Anticipatory scheduling: A disk scheduling framework to overcome deceptive idleness in synchronous I/O. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, pages 117–130, Banff, Canada, Oct. 2001. ACM.

[12] K. Li, R. Kumpf, P. Horton, and T. Anderson. A quantitative analysis of disk drive power management in portable computers. In *Proceedings of the Winter 1994 USENIX Technical Conference*, pages 279–291, San Francisco, CA, Jan. 1994.

[13] T. Madhyastha and K. P. Yang. Physical modeling of probe-based storage. In *Proceedings of the 18th IEEE Symposium on Mass Storage Systems and Technologies*, pages 207–224, Apr. 2001.

[14] M. K. McKusick, W. N. Joy, S. J. Leffler, and R. S. Fabry. A fast file system for UNIX. *ACM Transactions on Computer Systems*, 2(3):181–197, Aug. 1984.

[15] Nanochip Inc. Nanochip: Array nanoprobe mass storage IC. http://www.nanochip.com/, 1999.

[16] N. Nieuwejaar, D. Kotz, A. Purakayastha, C. S. Ellis, and M. Best. File-access characteristics of parallel scientific workloads. *IEEE Transactions on Parallel and Distributed Systems*, 7(10):1075–1089, Oct. 1996.

[17] D. Roselli, J. Lorch, and T. Anderson. A comparison of file system workloads. In *Proceedings of the 2000 USENIX Annual Technical Conference*, pages 41–54, June 2000.

[18] C. Ruemmler and J. Wilkes. Unix disk access patterns. In *Proceedings of the Winter 1993 USENIX Technical Conference*, pages 405–420, San Diego, CA, Jan. 1993.

[19] S. W. Schlosser, J. L. Griffin, D. F. Nagle, and G. R. Ganger. Designing computer systems with MEMS-based storage. In *Proceedings of the 9th Interational Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 1–12, Cambridge, MA, Nov. 2000.

[20] P. Vettiger, M. Despont, U. Drechsler, U. Urig, W. Aberle, M. Lutwyche, H. Rothuizen, R. Stutz, R. Widmer, and G. Binnig. The "Millipede"– More than one thousand tips for future AFM data storage. *IBM Journal of Research and Development*, 44(3):323–340, 2000.

[21] J. Wilkes. Predictive power conservation. Technical Report HPL-CSP-92-5, Hewlett-Packard Laboratories, Feb. 1992.

[22] M. Wu and W. Zwaenepoel. eNVy: a non-volatile, main memory storage system. In *Proceedings of the 6th Interational Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 86–97. ACM, Oct. 1994.