Optimizing MEMS-Based Storage Devices for Mobile Battery-Powered Systems

MOHAMMED G. KHATIB and PIETER H. HARTEL
University of Twente

An emerging storage technology, called MEMS-based storage, promises nonvolatile storage devices with ultrahigh density, high rigidity, a small form factor, and low cost. For these reasons, MEMS-based storage devices are suitable for battery-powered mobile systems such as PDAs. For deployment in such systems, MEMS-based storage devices must consume little energy. This work mainly targets reducing the energy consumption of this class of devices.

We derive the operation modes of a MEMS-based storage device and systemically devise a policy in each mode for energy saving. Three types of policies are presented: power management, shutdown, and data-layout policy. Combined, these policies reduce the total energy consumed by a MEMS-based storage device. A MEMS-based storage device that enforces these policies comes close to Flash with respect to energy consumption and response time. However, enhancement on the device level is still needed; we present some suggestions to resolve this issue.

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

General Terms: Economics, Management, Performance

Additional Key Words and Phrases: Probe storage, energy efficiency, green storage, mobile systems, design space

ACM Reference Format:

Khatib, M. G. and Hartel, P. H. 2010. Optimizing MEMS-based storage devices for mobile battery-powered systems. ACM Trans. Storage, 6, 1, Article 1 (March 2010), 37 pages.
DOI = 10.1145/1714454.1714455 http://doi.acm.org/10.1145/1714454.1714455

This research is supported by the Technology Foundation STW, Applied Science Division of NWO and The Technology Program of the Ministry of Economic Affairs under project TES.06369.

This article subsumes two previous publications: Power Management of MEMS-Based Storage Devices for Mobile Systems, which appears in *Proceedings of the 8th ACM / IEEE International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASEs'08)* and Workload-Based Configuration of MEMS-Based Storage Devices for Mobile Systems, which appears in *Proceedings of the 8 th ACM / IEEE International Conference on Embedded Software (EMSOFT'08)*.

Author's address: m.g.khatib@utwente.nl.

DOI 10.1145/1714454.1714455 http://doi.acm.org/10.1145/1714454.1714455

1. INTRODUCTION

The demand for energy efficient computer systems is more pressing than ever. Mobile devices as well as server systems must be green. The greenness trend coexists with a tendency toward digital media, where almost everything is becoming electronic. As a result, the demand for green high-density storage is increasing rapidly. Researchers are continuously looking for new storage technologies. A future storage technology must be as energy efficient as Flash (5% energy share of a computer system) and as cheap as the disk drive (a few dollar cents per gigabyte).

One such a storage technology is MEMS-based storage which leverages the well-established MEMS fabrication techniques to offer inexpensive storage systems. MEMS-based storage has ultrahigh storage densities ($> 1\,\text{Tb/in}^2$) [Lantz et al. 2007], which corresponds to bit dimensions of $18 \times 18\,\text{nm}^2$. Thanks to MEMS techniques, small footprint MEMS-based storage devices can be produced. For instance, a MEMS-based storage device prototyped by IBM has approximately a 41 mm^2 footprint, so that it can be housed in a SecureDigital (SD) package. For these reasons, MEMS-based storage devices are suitable for mobile applications.

MEMS-based storage promises low cost mainly due to the following reasons. Firstly, they can be manufactured using the well-established batch MEMS fabrication technology [Lantz et al. 2007]. Secondly, these devices can be manufactured using micron-scale fabrication plants, whose equipment was installed ten years ago and had passed their break-even point. Consequently, the need to build dedicated fabrication plants is avoided, unlike for Flash memory. Thirdly, these plants can be used to make future generations of MEMS, since MEMS poses no stringent requirements on the lithography process when increasing the density.

In addition to cost, MEMS-based storage promises density scaling. MEMS-based storage devices employ ultrahigh-density recording techniques that can achieve bit dimensions down to $1.5 \times 1.7\,\text{nm}^2$ [Bennewitz et al. 2002]. In contrast, Flash designers envisage that Flash memory will face significant challenges at technology nodes below 30 nm [Prall 2007; Lai 2008], which has consequences for the endurance and retention of Flash. Table I summarizes the main characteristics of Flash and MEMS-based storage. Some of the figures are taken from our experiments with a Flash card and a MEMS-based storage device.

Problem. For MEMS-based storage to be successful in mobile applications more than low cost and high density are needed. MEMS-based storage devices should be competitive with respect to timing performance and energy consumption. A MEMS-based storage device is mechanical in nature, and hence exhibits seek times and consumes a relatively large amount of energy.

We dedicate this work to investigating how MEMS-based storage can elevate its competence level: consuming little energy and exhibiting short response times. We evaluate the enhanced MEMS-based storage device by comparing it to Flash memory, which exhibits high performance and low energy consumption.

Table I. Characteristics of the Deployed CF NAND Flash Card and the Simulated IBM MEMS Device (the figures of the MEMS-based storage are based on a prototype and thus are subject to change as the technology evolves)

Characteristic	CF card	MEMS-based storage	Unit
Current bit dimensions	45	18	nm^2
Minimum bit dimensions	30	2	nm^2
Seek time	N/A	<1.5	ms
Access time (R/W)	0.025/0.2	0.025	ms/b
Block erase time	1.5	0.025	ms
Energy per bit (R/W)	10	12	nJ/b
Cost	2	<2[a]	$/GB
Shock resistance	>10 G	10 G	m/s^2
Endurance	10^5	10^{8}[b]	cycles
Retention	10[c]	10	years

[a] MEMS-based storage is less restricted by lithography than Flash.
[b] Endurance of MEMS-based storage corresponds to the probe write cycles.
[v] Retention of a NAND cell is affected by the number of times it has been erased.

Contributions. First, we look into optimizing MEMS-based storage as a virgin technology that still requires enhancement on the system level. This research (1) proposes a power state machine for MEMS-based storage devices based on their state-of-the-art and by contrasting them with the disk drive. (2) We systemically visit each operation mode and devise a policy to reduce the energy consumption in that mode. (3) We present a method to format the data layout, so that small design trade-offs become attainable.

For a valid comparison, we take a concrete mobile Flash card and compare it to a simulation-based MEMS card. The devices housed in the cards have similar nominal throughputs. We compare them with respect to response time and energy consumption for mobile applications. Our research highlights the potential of MEMS-based storage when our policies are enforced. (4) The research also highlights the need for further enhancement of the technology on the device level.

The remainder of the article is organized as follows. In Section 2, we briefly introduce MEMS-based storage; Section 3 devises the operation modes of a MEMS-based storage device; Section 4 presents the methodology adopted throughout our experiments; Sections 5 to 7 present the optimization policies for MEMS-based storage; Section 8 evaluates the devised policies in a case study and compares MEMS-based storage with Flash; Section 9 summarizes our experiments with MEMS-based storage in a list of recommendations to foster MEMS-based storage further; Section 10 discusses the related work, and Section 11 concludes.

2. MEMS-BASED STORAGE

Several design models for MEMS-based storage have been proposed [Lantz et al. 2007; Carley et al. 2000; Abelmann et al. 2003]. Although these models adopt different storage and actuation techniques, they have a common architecture. A MEMS-based storage device consists of two distinct physical layers,

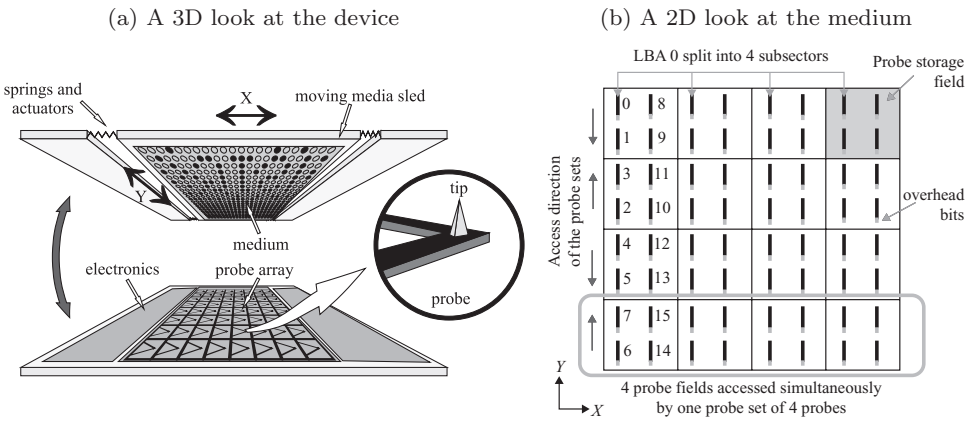

Fig. 1. Three- and two-dimensional views of a MEMS-based storage device. (a) Two layers facing each other where the media sled is attached to springs that suspend it across the probe array; (b) the storage area of a simplified MEMS-based storage device consisting of 4×4 probe fields.

one above the other, as shown in Figure 1(a). The top layer, called the *media sled*, is suspended by springs across the bottom layer, where the Z distance is maintained by nanopositioners. The bottom layer is a two-dimensional array of read/write probes or heads, called the *probe array*. For example, an IBM MEMS prototype [Lantz et al. 2007] has a 64×64 probe array.

Bits can be recorded on a magnetic patterned medium as in the micro scanning probe array memory (μSPAM) [Abelmann et al. 2003] and the CMU MEMStore [Carley et al. 2000]; or on a polymer medium as in the IBM MEMS device [Lantz et al. 2007]. The sled moves independently in the X, Y, and Z directions relative to the probe array. In all design models, each probe sweeps over a bounded area of the media sled, called the *probe (storage) field*, as sketched in Figure 1(b). Consequently, seek times are short. Further, a relatively high (aggregate) data rate is attained by striping a sector—the storage granularity—across a probe set of several probes. A sector is assigned a number, called the logical block address (LBA), by which it is uniquely addressable. Other work explains the data layout in great detail [Griffin et al. 2000a].

To read from or write to the medium, the media sled moves along the Y direction, along which the data tracks lie, as shown in Figure 1(b). While accessing data, the X actuators keep the sled in position along the X direction on the accessed data track, counteracting the restoring force of the springs. A sector is striped across many probes to reduce the response time. Each probes accesses a part of a sector, called a subsector. Figure 1(b) shows how LBA 0 is striped across 4 probes. During inactivity, the springs hold the sled at its resting position, where every probe faces the center position of its probe field.

3. OPERATION MODES

Typically, a mechanical storage device can be in one of three different operation modes when it is switched on: (1) seek, (2) active, and (3) idle; we detail them in the next section. Figure 2 presents the power state machine (PSM) for such a

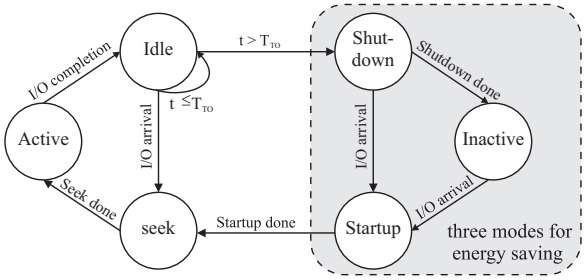

Fig. 2. The three typical operation modes of a mechanical storage device extended by another three modes for energy-saving.

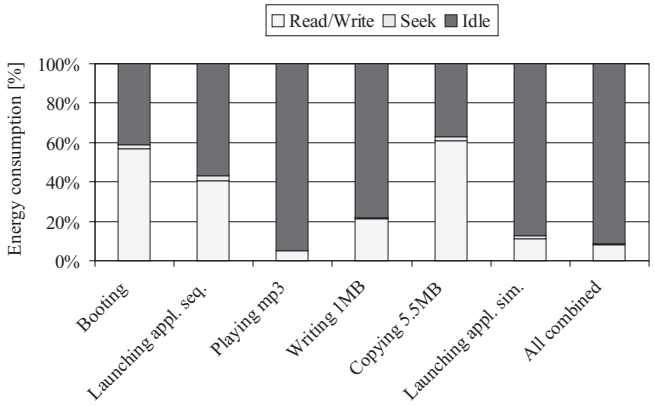

Fig. 3. Energy breakdown for various applications of a MEMS-based storage device that has no power management. Results from simulations with our modeled MEMS-based storage device show that at least 40% of the total energy is consumed in the idle state. Writing a 1 MB file corresponds to taking a picture by a standard mobile-device camera, and copying a 5.5 MB file represents copying a file from/to a removable storage device. Launching applications simultaneously is taken as an example of stressing the device with several consecutive I/O requests, whereas its sequential counterpart is for comparison. This test is important for checking the generality of the conclusions of the power management study.

mechanical device, and shows the possible transitions between the three modes. To reduce its energy consumption, a mechanical storage device implements a low-power mode to halt the moving storage medium. As a result, the simple PSM is extended by another three operation modes, as shown in the gray area of Figure 2. Transitioning between low-power modes and active modes goes via preparation modes, namely the startup and shutdown modes.

Like mechanical storage devices, a MEMS-based storage device consumes a significant amount of energy in the idle mode. Figure 3 gives the energy breakdown for various applications. It shows that at least 40% of the total energy is consumed during idleness, while the media sled is moving in anticipation of requests nearby. Therefore, a MEMS-based storage device should implement power management and have a PSM similar to the full PSM shown in Figure 2.

In the following, we construct the PSM of a MEMS-based storage device by (1) contrasting MEMS-based storage devices with HDDs, and (2) carefully

inspecting their characteristics based on their state-of-the-art. We seek to answer the following two questions:

—How many low-power modes should the PSM have?
—What are the corresponding preparation modes?

We address these questions next, and evaluate our choices in Section 5.

3.1 Contrasting with Hard Disk Drives

A hard disk drive (HDD) has a PSM similar to that of Figure 2. A HDD can, however, implement several low-power modes of different rotation speeds [Gurumurthi 2005] depending on the form factor and whether the HDD is commodity or enterprise equipment. Upon a request arrival, the disk enters the *seek mode* where the head seeks the addressed track. Once the head is aligned over the right track, the head starts reading from or writing to the medium, referred to as the *active mode*. Upon completion of reading or writing data, the disk enters the *idle mode*, where its platters keep rotating in anticipation of impending requests. If power management (PM) is employed, the disk transitions from idle mode to the *shutdown mode*, where it stops the medium, then parks the heads, and switches off a large part of the electronics. Upon shutdown completion, the disk is in *inactive mode*, where only its interface is powered on to wake the disk up when a request arrives. Upon request arrival, the disk goes in the startup mode. In the *startup mode*, the platters spin up and gain a certain speed first. After that, the heads can be loaded to fly over (and not touch) the platters separated by the air pad created by the spinning platters [Jacob et al. 2008, Ch. 17, pp. 631–633]. This spinup activity takes several seconds depending on, among other things, the mass of the platters and the target spinning speed. After spinup, the head seeks the addressed track to satisfy the request, and so on.

In a MEMS-based storage device, unlike in a HDD, the media sled is always suspended by springs across the probe array at a specific distance from the storage medium (see Figure 1(a)), which is actively maintained by the Z nanopositioners. As a result, no mechanical startup overhead exists. To access a MEMS-based storage device that is in inactive mode, the media sled directly seeks the addressed data block along X and Y simultaneously, since motions in the two directions are independent. As a result, the seek time is the maximum of the seek times along X and Y. A MEMS-based storage device has a lightweight sled, exhibits a high storage density, and has a micro scale. These characteristics enable the realization of a high-capacity storage device in a small-form factor. Because there are many probes, the sweep area of one probe is relatively small. As a consequence, seek times are short. At shutdown the sled moves to its resting position, namely the center of the probe field, and remains stationary. Like the seek time, the shutdown time is short, and thus shutdown energy is small.

Example. In the IBM MEMS device [Pantazi et al. 2008], a large number of probes (64×64 probes) share one medium, reducing the storage field of an

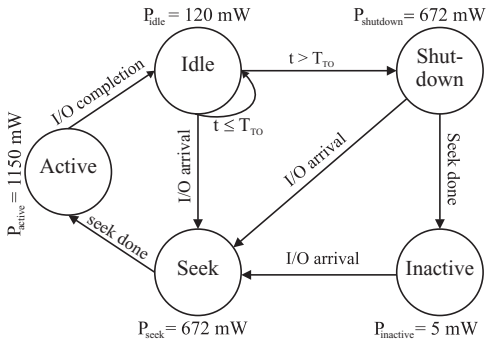

Fig. 4. Our proposed power state machine (PSM) for MEMS-based storage devices. The power figure of the idle and active modes represent the peak power dissipated at maximum displacement. The figures come from a relatively recent prototype of the IBM MEMS.

individual probe to 0.01 mm^2. The media sled weighs approximately 0.1 g, and the recording density is 840 Gb/in^2. In contrast, the smallest disk drive ever, namely the Toshiba 0.85-inch drive, has one single platter that weighs 1g. The accessible platter space per head is approximately 366 mm^2 and the storage density is 30 Gb/in^2 [Toshiba 2004].[1]

In disk drives, the startup, shutdown, and seek energy and delay are incurred every time the disk shuts down and subsequently starts up. Unlike disk drives, MEMS-based storage devices have no startup overhead, but experience small seek and shutdown overheads. The small overheads motivate us to implement a PSM of one low-power mode. We present the PSM of MEMS-based storage devices next.

3.2 Power State Machine

Figure 4 shows the proposed power state machine (PSM) for MEMS-based storage devices, which is an evolution of that proposed by Hong et al. [2006]. The PSM has five operation modes: seek, active, idle, shutdown, and inactive. We detail these modes and their power dissipation figures. The figures come from a relatively recent prototype of the IBM MEMS device [Lantz et al. 2007]. In the *seek mode*, the media sled moves from its current position to the starting position of the next request to service it. Seeking dissipates 336 mW per direction, amounting to 672 mW in total. The seek model is the bang-bang optimal time model, which applies the maximum allowed current to achieve the shortest seek time possible for any given distance. In the *active mode*, the device accesses (i.e., reads or writes) data, where the sled dissipates 60 mW to move along the Y direction at its maximum displacement (of 50 μm) from the center. It also dissipates another 60 mW to stay still in the X direction at the maximum displacement from the center. The power varies depending on the sled position, where peak power is dissipated at maximum displacement to counteract the

[1]Note that the storage density of the disk drive has increased since the time of the 0.85-inch drive; modern hard disk drives exhibit a storage density of 250 Gb/in^2.

Table II. Settings of the Simulated MEMS-Based Storage Device (the figures come from a relatively recent prototype of the IBM MEMS device)

Parameter	Setting	Unit
total number of probes	64×64	probes
probe field area	100×100	μm^2
bit/track pitch	25	nm
per-probe data rate	40	Kbps
maximum throughput	20	MB/s
capacity	7.6	GB
maximum active power	1150	mW
maximum idle power	120	mW
shutdown power	672	mW
inactive power	5	mW
seek power	672	mW

spring forces. That is, the 60 mW above represents the maximum power dissipation. A probe reads or writes at a rate of 40 Kbps.

The total of 4096 probes and the error-correction electronics dissipate 1 W to read/write and correct data (approximately 0.25 mW per probe and its electronics). In the *idle mode*, the device anticipates requests to nearby locations. The sled moves along Y at the read/write velocity, while staying still in X, dissipating a maximum power of 120 mW in total. Upon request arrival, the device goes back into the seek mode. Otherwise, if within a certain time interval that is equal to the timeout (T_{TO}) no request arrives, the device goes into the shutdown mode. In the *shutdown mode*, the sled travels to the center (resting) position from its current position, which is a seek operation, but to the center. The actual travel time of the seeks depends on the distance. If a request arrives while shutting down, the device goes into the seek mode immediately. Otherwise, if no request arrives and the sled reaches the center position, the device goes into the inactive mode. In the *inactive mode*, the sled rests in the center position, the probes are switched off, and the interface awaits requests, dissipating 5 mW (corresponding to the inactive power of a CompactFlash card with which we compare our MEMS-based storage device). No startup mode exists, as explained previously, assuming that the time to start up the electronics is short enough to be negligible. The power figures are shown in Table II.

3.3 Organization

Having detailed the PSM of MEMS-based storage devices, Sections 5 to 7 optimize in three operation modes: (1) idle, (2) shutdown, and (3) active. Each section devises a policy for one of the three modes.

Each of these policies is parameterized. We assess their influence for their optimal settings in view of the principal design targets: energy consumption, response time, and capacity. These policies are as follows.

—The *power management policy* decides the time instance *when* to transition from the idle state to the shutdown state by setting the value of the timeout; we study it in Section 5.

—The *shutdown policy* decides on *how* to transition from the shutdown state to the inactive state; we study it in Section 6.

—The *data-layout policy* decides the way user data is organized, and thus *how* to process incoming requests. Consequently, the policy influences the state transition from the seek state to the active state and from the active state to the idle state; we study it in Section 7.

One state transition is left uncovered in the PSM by the above policies: from the inactive state to the seek state. The time spent in the inactive state is determined by the interarrival time of the workload exercised on the device.

4. EXPERIMENTAL METHODOLOGY

Flash memories are publicly available, whereas MEMS-based storage devices are not. In this work, we adopt an empirical approach to characterize Flash memory and a trace-driven simulation approach for MEMS-based storage.

4.1 MEMS Model

IBM prototyped a MEMS-based storage device of 64×64 probes [Pantazi et al. 2008]. Although their prototype is not publicly available for experiments, sufficient specification data is available in the literature [Lantz et al. 2007]. We use trace-driven simulations and the DiskSim simulator [Bucy et al. 2003], a validated modular simulator for simulating various types and architectures of storage subsystems. We refine the performance and energy models of the CMU MEMS model [Griffin et al. 2000a] to model the IBM MEMS with better accuracy. All the model parameters including the bit dimensions and the per-probe data rate of the model are set to those of the IBM MEMS device [Lantz et al. 2007]. The relevant parameters are summarized in Table II.

We scale the bit dimensions in our MEMS model from $25\,\text{nm} \times 25\,\text{nm}$ to $40\,\text{nm} \times 40\,\text{nm}$, so that the formatted MEMS-based storage device has a capacity that is approximately equal to that of our Flash card, that is, about 2 GB. The scaling maintains a fair comparison with Flash (see Section 8), since seeks in the MEMS-based storage device span all the physical dimensions of a probe storage field, and thus the address space. Consequently, we report the worst-case for seek time and seek energy. Note that the per-probe data rate is preserved when scaling the bit dimensions, so that the read/write time is not influenced.

We cannot tell exactly how close our MEMS model is to reality, since we can not access a MEMS-based storage device to carry out an accuracy study. Based on our understanding of MEMS-based storage, however, we believe that the seek model is the part that causes the most uncertainty in our results. This is because the seek operation involves subtle mechanical dynamics that should be captured. In comparison with a real device, IBM shows that the bang-bang seek model, which we adopt, is 19% off [Sebastian et al. 2006]. We believe that 19% can be considered for the overall model, although the seek time is just a small portion (see Section 8.3). We think that this is an overestimation, which provides a safe margin for our conclusions.

Table III. Statistics of the Four Traces in this Research

Metric	ext3-4K	ext2-4K	ext3-1K	ext2-1K
Number of IOs	2677	1820	5517	2743
Sequentiality percentage	44.1	32.6	62.3	38.0
Write percentage	56.4	39.9	10.0	47.1
Statistics of the Request Size [0.5 KB sector]				
Minimum	8.0	8.0	2.0	2.0
Median	8.0	8.0	2.0	2.0
Maximum	256.0	256.0	256.0	256.0
Mean	43.3	52.9	17.6	32.8
Standard deviation	69.3	82.2	47.0	68.9

4.2 PDA Setup

By targeting mobile environments, we collected a representative trace while exercising the 2 GB SanDisk Standard CompactFlash (CF) card plugged into an HP iPAQ H2215 PDA. This PDA runs an embedded version of Linux (kernel version 2.6.17). Jens Axboe's block trace utility [Kernel Tracing 2009] was used to log I/O events, which were forwarded to a host machine in order to minimize interference with the measurements. The CF card served as the main storage device on which the root filesystem (`rootfs`) was located. Thus, all I/O activities went from and to the CF card.

4.3 Traces

We logged a workload that captures different system and application activities. System activities include booting and starting the graphical user interface. Application activities include launching applications, such as the text editor and web browser; copying files; and creating and deleting files. In addition, the trace contains streaming scenarios at 32 Kbps, 128 Kbps, and 384 Kbps corresponding to different audio and video qualities. Streaming at the three bit rates was carried out from and to the CF card, amounting to six streaming scenarios in total. The CF card was formatted with the `ext3` file system and a block size of 4 KB.

MEMS-based storage devices will communicate with the file system layer in computer systems. As a result, the performance and energy consumption of MEMS-based storage devices is influenced by the type of the file system and its block size. For stronger conclusions, we further trace and simulate with different settings of the I/O subsystem. We captured the same workload on the `ext2` file system, a nonjournaling version of `ext3`. In addition, we formatted each file system with the default maximum block size, 4 KB, and a smaller size of 1 KB. Thus, we carry out simulations against four different traces: (1) `ext3−4K`, (2) `ext2−4K`, (3) `ext3−1K`, and (4) `ext2−1K`. Note that although these traces are induced by the same application scenarios, they differ due to their respective I/O settings. Table III presents the main statistics of these traces.

In the next three sections, we devise and study the three optimization algorithms.

5. THE POWER MANAGEMENT POLICY

A MEMS-based storage device consumes a large amount of energy in the idle mode (see Figure 3). To save on the idle energy, the sled should be shut down if the request queue is empty. The power management (PM) policy decides when to shut down the sled. This section studies the power management policy that a MEMS-based storage device should enforce.

5.1 Fixed-Timeout Power Management

A large body of work on power management (PM) policies for disk drives and processors exists; Benini et al. [2000] give a survey. Generally, there are two types of policies: static and dynamic. Dynamic policies achieve more savings at lower timing performance degradation than static policies do [Lu et al. 2000]. However, dynamic policies demand more processing and memory resources because they keep a history of recent timeout values and power states. Both types of policies can also be employed in MEMS-based storage devices.

The timeout (T_{TO}) in the power state machine (PSM) determines the time of the state transition from idle to shutdown. We parameterize the timeout to study the influence on performance and energy saving. By driving the PSM with real-world traces and changing the value of the timeout, we quantify the energy saving and the resulting performance of the device. As will be shown later, the quantification shows the near-optimality of the fixed-timeout PM policy.

Resting the probe in the center position during inactivity increases the seek distance for the next request if the request addresses the same region of the previous request. This is the case for real-world workloads that exhibit spatial locality of reference (i.e., consecutive requests address nearby locations). A long seek distance results in a long seek time and thus a long response time. As a result, energy saving must be *traded off* for response time (i.e., performance). The tuning parameter of the energy−performance trade-off is the timeout. In general, the larger the timeout, the less the energy saving and the better the performance, and vice versa. We discuss the simulation results in Section 5.2.

We study power management by means of simulation. Simulations are driven by the four traces, which we captured on the CF card. We adopt the power state machine (PSM) presented previously, and set the timeout (T_{TO}) with incremental integral values. The timeout values are 0 to 10 ms. We then increase the timeout to 50 ms in steps of 10 ms to pursue the trend of the energy−performance trade-off.

5.2 Simulation Results

In this section, we discuss the simulation results in detail. The results present the trade-off between the energy consumption and the response time (i.e., the timing performance) for different values of the timeout. We first discuss the trade-off when simulating against the whole `ext3−4K` trace (i.e., all application sessions combined). After that, we present the results for the other three traces (`ext3−1K`, `ext2−4K`, and `ext2−1K`) when simulating against each one as a whole. Last, we discuss the results for each application session individually.

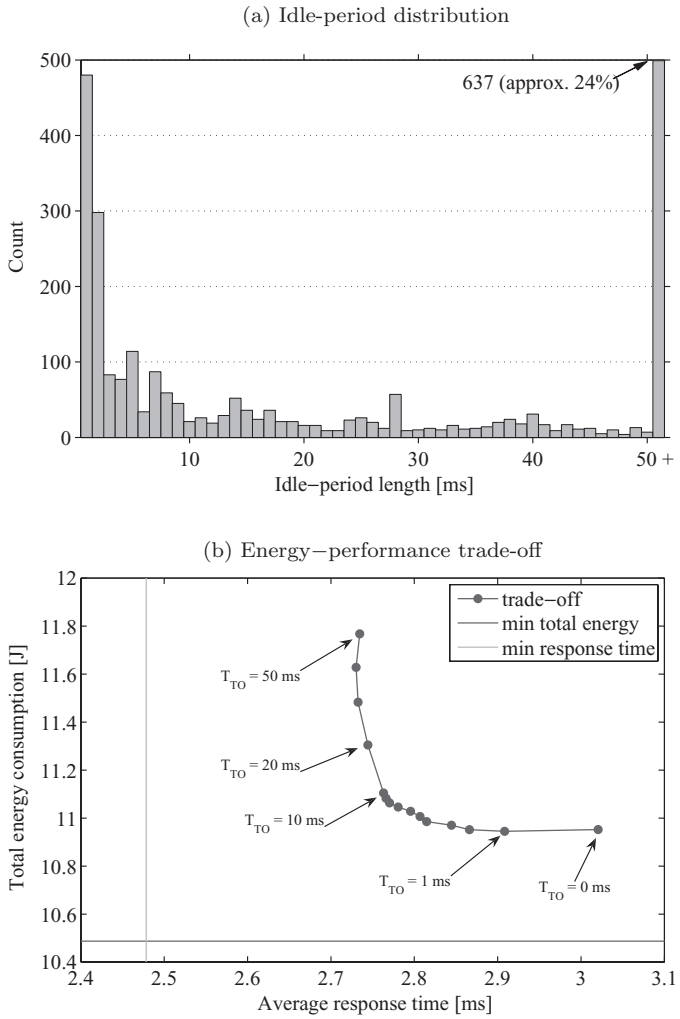

Fig. 5. (a) The distribution of the idle-period length; there exist short (but many), and long (but few) idle periods. (b) Total energy consumption versus average response time when simulating against the whole `ext3−4K` trace. Timeout values 1 to 10 ms make little difference in energy consumption (approximately 2%), but vary in response times (approximately 10%). We show the minimum energy consumption and the minimum response time.

5.2.1 *Whole `ext3−4K` trace.* In general, the energy saving increases as the timeout (T_{TO}) decreases, because more idle periods are exploited to shut down the device. On the other hand, response time increases as the timeout decreases, since longer seek distances are incurred. Figure 5(a) and Figure 5(b) show the histogram of idle-period length and the energy−performance trade-off, respectively, when simulating against the whole `ext3−4K` trace. About 48% of the idle periods lie in the range of 0 to 10 ms.

Figure 5(b) plots the total energy consumption for the whole trace versus the average response time per request. It shows a decrease of 0.3 ms (approximately

10%) in average response time between $T_{TO} = 0$ ms and $T_{TO} = 10$ ms. This decrease is explained in Figure 5(a), where the size of this decrease is proportional to the sum of the heights of the first ten buckets. When $T_{TO} = 10$ ms, the first ten buckets are not exploited, avoiding longer seeks from the center position. In general, these buckets mainly influence the performance, due to their relatively high occurrence frequency. As the timeout increases, less performance degradation is incurred, due to the infrequent occurrence of long idle periods. We also observe an increase in response time at $T_{TO} = 50$ ms relative to $T_{TO} = 40$ ms. This is because the distance between the sled position and the position of the next request is not monotonic with respect to time. That is, it increases and decreases depending on whether the sled has reached its maximum displacement and then reversed its motion direction while waiting for the next request.

Figure 5(b) shows a slight decrease in energy consumption at $T_{TO} = 1$ ms compared to $T_{TO} = 0$ ms. The longer seeks explained above not only worsen performance but also cost extra energy, so that avoiding the first bucket is more profitable than exploiting it. Also, no pronounced difference (approximately 2%) in energy consumption is seen in the range $T_{TO} = 1 - 10$ ms, due to the small energy-saving contributions of their respective buckets compared to that of idle periods longer than 10 ms. From an energy-saving perspective, an occurrence of one idle period of length of 30 ms, for instance, is more profitable than 30 occurrences of an idle period of length 1 ms.

Optimality. To quantify the optimality of the fixed-timeout PM policy with respect to energy saving, we calculate the *minimum energy consumption* (Figure 5(b)): the sum of (1) the energy consumed for reading/writing data; (2) the energy consumed for seeking, which is caused by requests only and not by PM; and (3) the energy consumed in inactivity. Figure 5(b) shows that the energy consumption at $T_{TO} = 10$ ms is within 6% of the minimum (approximately a 0.6 J absolute difference). We also calculate the *minimum response time* (Figure 5(b)): the sum of (1) the transfer time and (2) the seek time due to requests only and not due to PM. Figure 5(b) shows that the difference in response time at $T_{TO} = 10$ ms is within 12% of the minimum (approximately 0.3 ms absolute difference).

However, we see that 8% out of this 12% can not be achieved by any value of the timeout (see $T_{TO} > 10$ ms). The reason is that when the sled is left idle for some time, it travels some distance that should be traveled back if a successive request addresses the same neighborhood. Thus, in mechanical devices there is always some seek distance that is incurred. $T_{TO} = 10$ ms, for example, is (near) optimal, since room of just 4% is left for optimization on response time.

5.2.2 *Other Traces.* We repeat the simulation for the other traces taken for different I/O settings. These are `ext3−1K`, `ext2−4K`, and `ext2−1K`. The simulation results of each trace as a whole agree with the results of the `ext3−4K` trace discussed in the previous section. Table IV presents the trade-offs for the simulated values of the timeout for each trace. For all traces, we observe that the actual minimum energy consumption is achieved when the timeout is in the range 1 to 10 ms. Another important observation is that increasing the timeout in the range 1 to 10 ms decreases the response time by a larger factor than it

Table IV. Energy Consumption versus Response Time for the Four Traces

T_{TO} [ms]	Energy consumption [J] / Response time [ms]							
	ext3−4K		ext3−1K		ext2−4K		ext2−1K	
0	11.0	3.0	10.4	1.8	9.9	3.3	9.7	2.6
1	10.9	2.9	10.3	1.7	9.9	3.2	9.7	2.6
2	11.0	2.9	10.4	1.7	9.9	3.2	9.7	2.6
3	11.0	2.8	10.4	1.7	9.9	3.2	9.7	2.6
4	11.0	2.8	10.4	1.6	9.9	3.2	9.8	2.6
5	11.0	2.8	10.4	1.6	9.9	3.2	9.8	2.5
10	11.1	2.8	10.5	1.6	10.0	3.2	9.8	2.5
20	11.3	2.7	10.7	1.6	10.0	3.2	9.9	2.5
30	11.5	2.7	10.9	1.6	10.1	3.2	10.0	2.5
40	11.6	2.7	11.1	1.6	10.1	3.2	10.1	2.5
50	11.8	2.7	11.2	1.6	10.1	3.2	10.1	2.5
Minimum	10.5	2.5	9.8	1.4	9.6	2.8	9.4	2.2

increases the energy. For example, setting the timeout to 10 ms instead of 1 ms decreases the response time by 7% at a 1% increase in energy consumption for the ext3−4K trace.

5.2.3 *Sessions Separately*. As mentioned earlier, our traces consist of several application sessions. The sessions include (1) booting Linux and starting the graphical user interface; (2) launching several applications sequentially; (3) playing an MP3; (4) writing a picture; (5) streaming from/to the storage device with different bit rates; (6) copying files and directories; and (7) launching several applications simultaneously. Since in reality not all of these sessions necessarily occur together, we split the traces into their respective sessions and simulate for each individual session.

The simulation results of each individual session for all traces agree with those of the entire traces. Figure 6(a), Figure 6(b), and Figure 9(b) show the energy−performance trade-off for sessions 4, 7, and 6, respectively. We observe that setting the timeout larger than zero decreases the response time significantly for a relatively small increase in energy consumption, if any. Further, in conformity with the conclusion for the whole trace, the fixed-time PM policy achieves near-optimal energy saving. If a workload exhibits a large percentage of idle periods that are shorter than 1 ms, setting the timeout to zero not only worsens the response time but also the energy consumption, since the shutdown energy becomes prominent. This is exactly the case for session 6 shown in Figure 9(b).

MEMS-based storage devices do not have startup energy, so that the relative difference between the energy consumption when $T_{\mathrm{TO}} = 10$ ms and the minimum does not exceed 7%, leaving very little room for improvement for dynamic policies. Thus, deploying a fixed-timeout power management policy with a power state machine of one low-power mode is sufficient to achieve a near-optimal energy saving at low performance degradation.

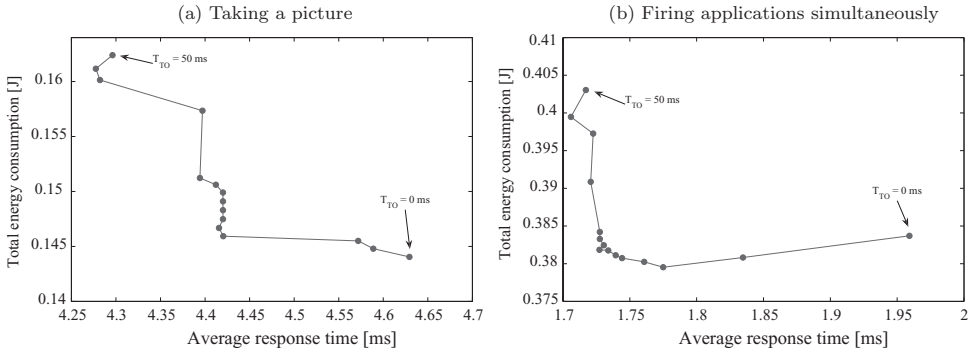

Fig. 6. Energy−performance trade-offs for two application scenarios from the `ext3−4K` trace.

6. THE SHUTDOWN POLICY

The study in the previous section demonstrates the capability of MEMS-based storage devices to shut down aggressively, which can be achieved by lowering the timeout. The more aggressive the shutdowns, the more idle periods are exploited.

Decreasing the timeout, however, increases the number of shutdowns, and thus increases the shutdown energy. As a consequence, aggressive shutdowns can result in more energy consumption. This section exploits the architecture of a MEMS-based storage device to reduce its shutdown energy.

We address the shutdown policy that complements the power management (PM) policy. While the power management policy decides *when* to shutdown, the shutdown policy decides *how* to shutdown. Two shutdown policies are possible; we detail them next.

6.1 Two Shutdown Policies

When shutting down the sled moves to the center position. The (first) shutdown policy, which we used in the previous section, employs the actuators to move the sled to its center position. Figure 7(a) illustrates that the actuators (F_a) in addition to the springs (F_s) exert a force on the sled and accelerate it for some distance. After that, the actuators reverse the force to decelerate the sled so that it stops at the center. The actuators consume energy during acceleration and deceleration. The invested energy shortens the shutdown time, and therefore we call this policy the *performance-efficient shutdown policy*. We employed this policy in the study of power management in the previous section.

A second shutdown policy uses the potential energy stored in the springs only. The springs bring the sled as close as possible to the center (Figure 7(b)) before the actuator starts to decelerate the sled so that it stops at the center. This policy consumes less energy than the previous policy, since it uses the actuators during deceleration only, as Figure 7(b) illustrates. Therefore, we call it the *energy-efficient shutdown policy*. The energy benefit, however, comes at a performance cost. That is, the sled takes longer to reach the center, since it is not accelerated by the actuators. Note that leaving the sled decelerating by following its natural oscillation course consumes no energy, but can have

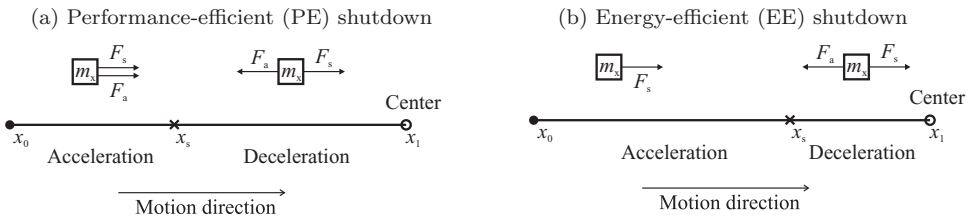

Fig. 7. A sketch of the sled motion toward the center when shutting down with the performance-efficient (PE) and energy-efficient (EE) shutdown policies.

consequences for the reliability of the device. Hence, we rule out this deceleration method.

We devise performance and energy models for the energy-efficient (EE) and performance-efficient (PE) shutdown policies. We devise the analytical models of the shutdown time and energy for both policies in an external technical report [Khatib et al. 2008]. We implement these models in the DiskSim MEMS model to compare them under real-world traces. We also study the interaction between the PM policy and the shutdown policy. For better understanding of the behavior of the two shutdown policies, we provide an analytical (static) study first. After that, we follow up with a trace-based (dynamic) simulation that compares both policies in real environments.

6.2 Analytical Study

The analytical study compares both policies when shutting down from every position within a probe storage field ($100\mu m \times 100\mu m$). We compare both policies with respect to the shutdown time and energy. The parameters of the models of both policies are set to the state-of-the-art figures from the IBM MEMS device [Lantz et al. 2007]. The resting position is the center at the coordinate (0, 0).

Since the motions along the X and Y directions are independent and similar, we present the results for the X direction only. The difference is calculated as $\frac{t_{EE}-t_{PE}}{t_{PE}}$ and $\frac{E_{PE}-E_{EE}}{E_{PE}}$ for the shutdown time and shutdown energy, respectively. The shutdown time and energy of the EE policy are referred to as t_{EE} and E_{EE}, respectively. We normalize the figures to the performance-efficient policy to demonstrate the relative gain in energy saving of the energy-efficient policy and the relative cost of the gain in terms of increase in response time.

6.2.1 *Difference in Shutdown Time.* Figure 8(a) shows that the relative difference in shutdown time (calculated as $\frac{t_{EE}-t_{PE}}{t_{PE}}$) is minimal at the borders of the probe area. It increases as the starting position gets closer to the center, since around the center the spring force is small. When deploying the EE policy, the sled accelerates to the center by the spring force only. The spring force depends on the distance between the sled's current position and the center ($F_s(x) \propto x$). That is, the larger the distance, the more force, but also the longer the distance the sled has to travel. As a result, shutdown times for all positions in the probe area are of the same order of magnitude. For example, the shutdown time at a $5\mu m$ and $45\mu m$ distance (in a range of 0 to $50\mu m$) is 1.6 ms and 2.0ms, respectively.

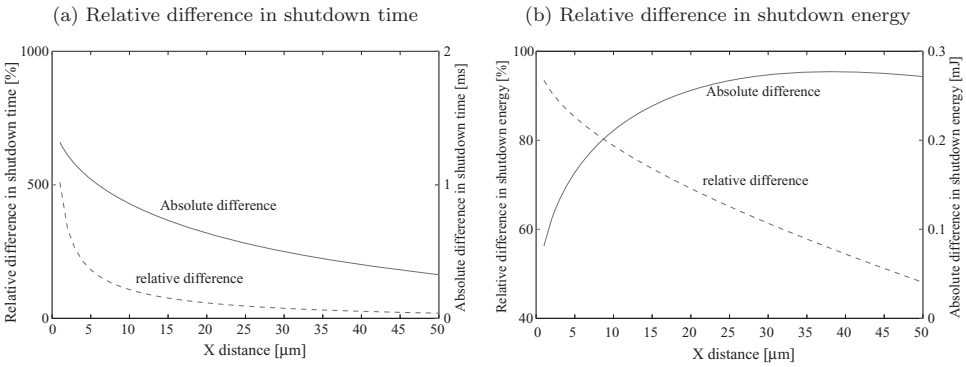

Fig. 8. Absolute and relative differences in shutdown time and energy between the performance-efficient (PE) and energy-efficient (EE) policies as a function of the distance from the center at 0. The relative difference is calculated as $\frac{t_{\mathrm{EE}}-t_{\mathrm{PE}}}{t_{\mathrm{PE}}}$ and $\frac{E_{\mathrm{PE}}-E_{\mathrm{EE}}}{E_{\mathrm{PE}}}$, respectively. The performance-efficient (PE) policy outperforms the energy-efficient (EE) policy but consumes more energy.

In contrast, when deploying the PE policy the shutdown time scales very sensitively with the traveled distance because it is actively accelerated. For example, the shutdown time at a 5μm and 45μm distance is 0.6ms and 1.6ms, respectively. This explains why the difference between both policies increases as the starting position gets closer to the center. Figure 8(a) shows that the absolute difference drops from above 1.25ms to below 0.50ms when the sled is 1μm and 50μm far from the center. The difference near the center is one order of magnitude longer than that at the borders, which explains the prohibitively large relative difference.

6.2.2 *Difference in Shutdown Energy.* Figure 8(b) plots the relative difference in energy consumption (calculated as $\frac{E_{\mathrm{PE}}-E_{\mathrm{EE}}}{E_{\mathrm{PE}}}$). The shutdown energy for the PE policy is larger than that for the EE policy because the former consumes energy for acceleration and deceleration, whereas the latter consumes energy for deceleration only. Similar to shutdown time, the relative energy difference is larger around the center and decreases as the starting position gets further from the center. When deploying the EE policy, the long acceleration phase, which is responsible for the prohibitive difference in shutdown time, consumes no energy. Therefore, no prohibitive energy difference exists around the center, unlike the shutdown time.

Figure 8(b) also plots the absolute difference which is in the sub-milli-Joule order. The difference increases for large distances, since more acceleration energy is consumed by PE. The increasing trend remains up to the point where the deceleration energy consumed by EE becomes significant and the absolute difference starts decreasing. The relative difference however keeps declining, since the energy consumed by PE increases as the distance increases.

6.2.3 *Discussion.* The analytical study shows that the energy-efficient shutdown policy saves more energy than the PE policy. However, the EE policy results in a long shutdown time compared to the PE policy. The difference

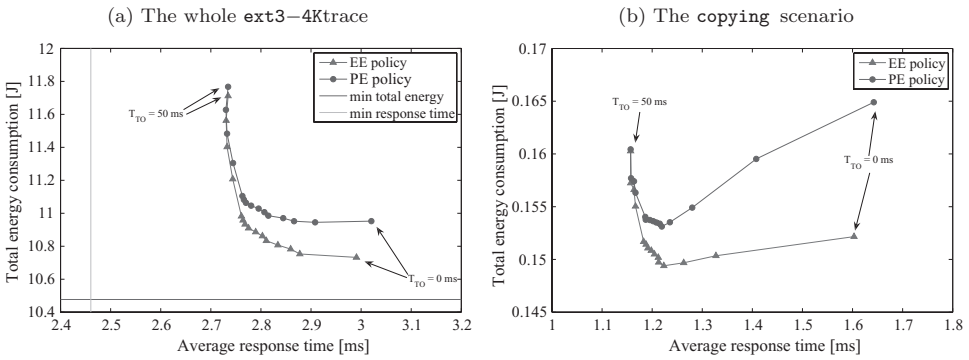

Fig. 9. Energy−performance trade-offs when deploying the performance-efficient (PE) and energy-efficient (EE) shutdown policies.

is prohibitively large near the center and can reach up to 500%, due to the (extremely) slow response of the EE policy in reaching the center.

The slow shutdown performance can be an advantage to real-world applications that exhibit (high) sequentiality and/or locality of reference. That is, moving the sled slowly allows for quick inexpensive seeks to an already visited region if new requests demand further data from that region. In fact, we observe this advantage in our simulations, presented next.

6.3 Trace-Based Study

We implemented the models of both policies in DiskSim to evaluate the performance and energy influence of each shutdown policy in combination with the power management policy. We repeated the experiments discussed in Section 5.2 for all traces with the energy-efficient shutdown policy under real-world traces.

Figure 9(a) plots the trade-off between response time and energy consumption when deploying the EE and PE shutdown policies for the `ext3−4K` trace. We see that the energy decreases and comes even closer to the minimum, further supporting the effectiveness of the fixed-timeout power management policy. At $T_{\mathrm{TO}} = 0$ ms, with the EE policy a MEMS-based storage device consumes less energy than with the PE policy, since shutdown energy (i.e., the overhead) becomes smaller.

Another finding is that the EE policy shortens the response time slightly, compared to the PE policy, and thus enhances the performance. The reason is that for sequential requests the EE policy shortens the response time because it moves the sled slowly to the center. As a result, driving the sled back to a previously visited region takes a small amount of time, and hence consumes a small amount of energy. The difference is noticeable for the small values of the timeout, due to the frequent occurrence of small idle periods, as shown in Figure 5(a).

The energy-saving difference between the two shutdown policies is negligible (approximately 2%) for the whole `ext3−4K` trace, as shown in Figure 9(a). This is because a quarter of the idle periods are longer than 50 ms, so that

their corresponding energy saving outweighs savings corresponding to periods smaller than 1 ms. Note that for traces that lack such long idle periods due to a high arrival rate, the difference in server systems, for example, in energy saving can be large. A similar case is the copying scenario, where the difference is approximately 10%, as Figure 9(b) shows.

Figure 9(a) also shows that even with large values of the timeout, the minimum response time cannot be achieved. The reason is that when the sled is left idle for some time, it travels some distance that should be traveled back if a successive request addresses the same neighborhood. Thus, in mechanical devices there is always some seek distance that is incurred.

Overall, we can conclude that the timeout $T_{\mathrm{TO}} = 10$ ms achieves a near-optimal energy saving (approximately 95%) at a negligible performance loss (4%), relative to the actual minimum represented by the line that connects all the points for $T_{\mathrm{TO}} \geq 10$ ms in Figure 9(a).

We repeated the simulation for the other three traces with the energy-efficient shutdown policy. We observed the same trend as for the `ext3−4K` trace. Our conclusion to deploy a timeout in the range of 1 to 10 ms still holds, supporting the previous conclusions.

7. THE DATA-LAYOUT POLICY

The data layout is concerned with the way user data is organized on the storage medium of a storage device. Hence the data layout influences the response time and the energy consumption of the storage device. For example, placing related data sectors contiguously on the physical medium avoids seeks between the sectors, which results in a short response time and low energy to access data.

In a MEMS-based storage device, the attainable data rate per probe is limited by several factors, including the probe resonance frequency. The per-probe data rate is 40 Kbps in the IBM MEMS device [Lantz et al. 2007], suggesting that systems requiring even moderate transfer rates must use many parallel probes. Therefore, a sector must be striped across many probes, each accesses a subsector.

Because MEMS-based storage devices are mechanical, they must separate physical subsectors by a few separation bits that allow the read channel to access a subsector fully [Jacob et al. 2008, Ch. 18, pp. 650–652]. Therefore, the subsector size should be relatively large compared to the physical overhead to reduce the loss in capacity due to sector striping.

7.1 Three Data-Layout Parameters

Striping influences the timing performance and the capacity of MEMS-based storage devices. The data-layout design space of MEMS-based storage devices widens beyond just block mapping. Three data-layout parameters emerge that must be configured properly to shorten the response time, to reduce energy consumption, and to reduce capacity loss. These parameters are as follows:

(1) *The total number of active probes (N)*—how many probes should operate simultaneously?

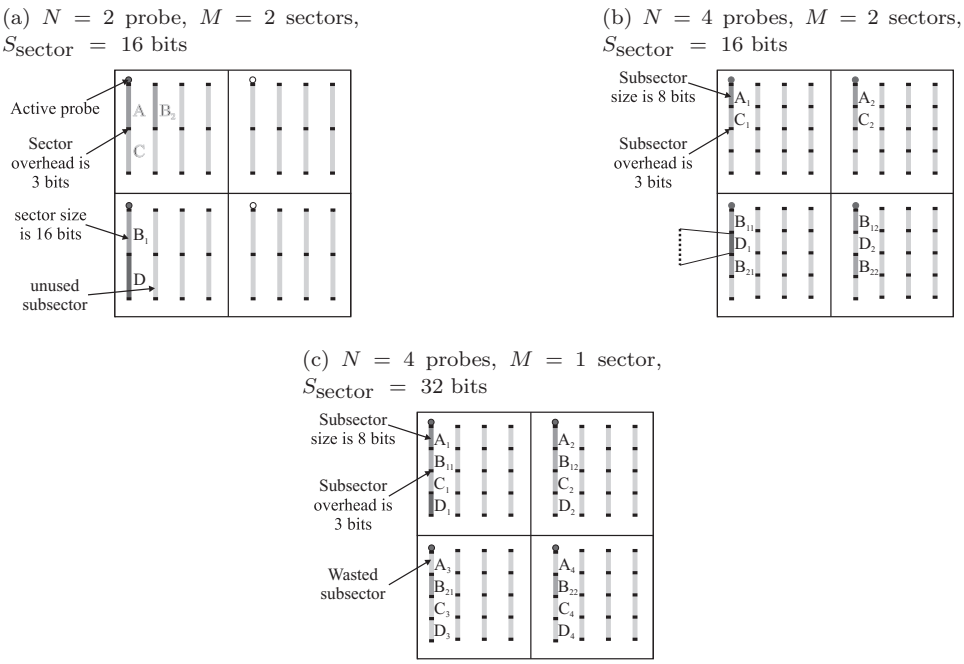

Fig. 10. Three possible configurations of the three data-layout parameters: the total number of active probes, sector parallelism, and sector size. The figure shows a simplified MEMS-based storage device. The first configuration (a) stripes a sector across one probe. By using twice as many active probes (b), a sector is striped across two probes and two sectors are accessible simultaneously. In (c) the sector size doubles, and a sector is striped across all probes so that just one large sector is accessible at a time. Increasing the sector parallelism as in (b) causes external fragmentation, and thus seeks such as from $B_{11}||B_{12}$ to $B_{21}||B_{22}$ ("||" means accessing in parallel), whereas increasing sector size as in (c) causes internal fragmentation, wasting capacity such as A_3 and D_3. Striping increases reliability, when proper encoding schemes are used; data reliability is however outside the scope of this work.

(2) *Sector parallelism (M)*—how many sectors should be simultaneously accessible from the device?
(3) *Sector size (S_{sector})*—should the conventional sector size of 512 bytes stay the same in MEMS-based storage devices?

The straightforward settings of these parameters would be to (1) operate all probes simultaneously to gain peak throughput; (2) access one sector at a time to maximize bandwidth utilization; and (3) keep the sector size intact to access useful data only.

While these settings sound logical, simulations show that none of the three design targets (i.e., response time, energy consumption, and capacity) of a MEMS-based storage device reaches optimality with such a configuration.

Figure 10 depicts a simple MEMS-based storage device with three possible configurations of the data-layout parameters. The first configuration stripes a 16-bit sector across one probe only, and can access two sectors simultaneously. The second configuration stripes the sector across two probes, but it can

still access two sectors simultaneously because it doubles the number of active probes. The third configuration stripes a 32-bit sector across all probes, avoiding fragmentation for large files like the case of file B in the second configuration. Next, we provide an anatomy of the subsector in MEMS-based storage devices.

7.2 The Physical Subsector

A storage device stores user data in physical sectors. In addition to user data, a physical sector contains error-correction code (ECC) data. All types of storage devices have to store ECC data to increase the reliability of the stored user data. The amount of ECC data depends on, among other factors, the sector size and the type of errors the device is prone to. We call the portion of user data of a physical sector, a logical sector. In disk drives, the ECC is one-tenth the size of the logical sector [McCarthy et al. 2002]. We assume that the size of the ECC overhead (S_{ECC}) is even larger in a MEMS-based storage device, and is one-eighth the size of the logical sector (S_{sector}), which is in agreement with figures available from the IBM MEMS device. Thus, the ECC overhead is

$$S_{\text{ECC}} = \left\lceil \frac{S_{\text{sector}}}{8} \right\rceil$$

Mechanical storage devices exhibit a physical overhead in order to address and access user data. This physical overhead is a few bits that separate every two contiguous subsectors as shown in Figure 10(a). The separation bits (1) allow for data buffering before writing a subsector, and (2) keep the clock of the read channel running so that the subsector can be fully read/written. Jacob et al. provide an anatomy of the physical sector in disk drives [Jacob et al. 2008, Ch. 18, pp. 650–652]. The physical overhead in disk drives has a small influence on the capacity because it occurs once per sector. In contrast, in MEMS-based storage devices the physical overhead has to occur at every subsector because every probe accesses a subsector due to striping. We assume that the total physical overhead per subsector is three bits, which amounts to a period of 75μs that is sufficient for processing.

From the preceding, striping a physical sector across K probes results in a physical subsector of size ($S_{\text{p-subsector}}$):

$$S_{\text{p-subsector}} = \left\lceil \frac{S_{\text{sector}} + S_{\text{ECC}}}{K} \right\rceil + 3\,. \tag{1}$$

To avoid very small capacities, we assume that a physical subsector is larger than or equal to 8 bits. To avoid seeks within an access to a subsector, the maximum physical subsector size is smaller than the subtrack size (i.e., the portion of a track located in one probe field) $\frac{\text{field length}}{\text{bit length}} = \frac{100000}{40} = 2500$ bits.

7.3 Influence of Each Parameter

This section studies the influence of each data-layout parameter individually on the three design targets (i.e., energy, performance, and capacity). We simulate against the `ext3-4K` trace. Table V lists the additional settings of our simulated MEMS-based storage device based on the studies conducted in the previous two

Table V. Additional Settings of the MEMS Model Based on the Studies in Sections 5 and 6. (The settings are supplementary to those in Table II)

Parameter	Setting	Unit
Shutdown time-out	1	ms
Shutdown policy	Energy efficient	
Total number of probes	$64 - 4096^a$	probes
Sector parallelism	$1 - 16^a$	sector
Logical sector size	$0.5 - 8^a$	KB

[a] We select values that are a power of two.

sections. The PSM in Figure 4 is deployed. These settings are supplementary to those in Table II; the methodology is detailed in Section 4.

7.3.1 *The Total Number of Active Probes (N)*

7.3.1.1 *Response Time.* Increasing the number of probes a sector is striped across shortens the read/write time because the subsector size decreases, as Eq. (1) shows. Figures 10(a) and (b) show that doubling the number of active probes from 2 to 4 results in smaller subsectors that a single probe has to access per sector; compare A to $A_1||A_2$ ("||" denotes parallel access). Thus, the time to read/write a striped sector is effectively the time a probe takes to read/write one subsector:

$$t_{\mathrm{RW-subsector}} = \frac{S_{\mathrm{p-subsector}}}{r_{\mathrm{probe}}}, \tag{2}$$

where $S_{\mathrm{p-subsector}}$ is the size of the physical subsector from Eq. (1), and r_{probe} is the data rate per probe.

Simulating against `ext3-4K`, Figure 11 plots the response time as a function of the number of probes with a sector size of 512 bytes. Because the minimum subsector size is 8 bits, the maximum number of probes per sector ($K = \frac{N}{M}$) is 512 probes. The response times are normalized to the response time when deploying 64 probes (83ms). Figure 11 confirms the significant influence of the number of probes on response time. When the number of probes doubles, the response time approximately halves.

7.3.1.2 *Energy.* Unlike the response time, which decreases as the number of probes increases, energy to access data does not decrease because more probes are switched on at the same time. Actuation energy, however, decreases. Actuators are powered on to keep the media sled in position on the X direction and to move it along Y. Increasing the number of probes decreases the subsector size, and thus shortens the distance the sled travels along Y, and subsequently the time it is held on X (compare Figure 10(a) to Figure 10(b)). Consequently, the actuation energy decreases. The total read and write energy per physical sector can be written as follows:

$$\begin{aligned} E_{\mathrm{RW-sector}} &= E_{\mathrm{probes}} + E_{\mathrm{actuation}} \\ &= K \cdot P_{\mathrm{probe}} \cdot t_{\mathrm{RW-subsector}} + P_{\mathrm{actuation}} \cdot t_{\mathrm{RW-subsector}}, \end{aligned} \tag{3}$$

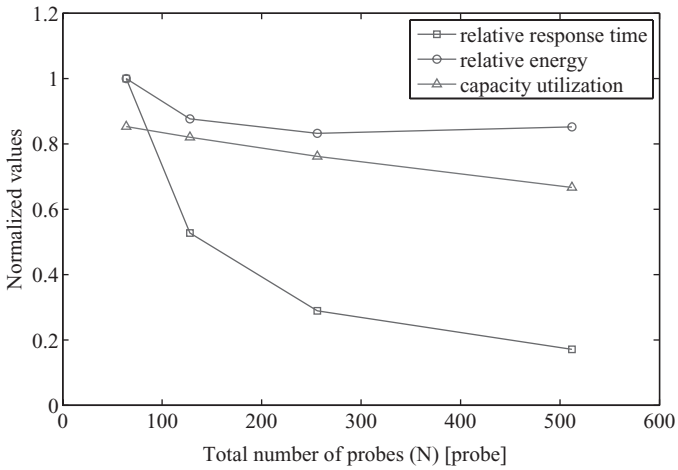

Fig. 11. Relative average response time, relative total energy consumption, and capacity utilization of the simulated MEMS-based storage device as a function of the total number of active probes. Simulations are carried out for sector parallelism (M) of one sector and a sector size (S_{sector}) of 512 bytes. Capacity figures are normalized to the physical (i.e., raw) capacity of the device. The reference values are 83ms, 13.6J, and 3GB.

where K is the number of probes per sector, P_{probe} is the power a probe dissipates to read or write one single bit, and $P_{\text{actuation}}$ is the power dissipated by both X and Y actuators. Note that the probes touch the medium during the actual read and write operations only. From Eq. (3), we can observe that the reduction in the total read/write energy is bounded by the energy consumed by the probes to read or write.

Figure 11 confirms this bound and shows the energy figures normalized to the energy when deploying 64 probes (13.6J). As the number of probes increases, the actuation energy decreases as the total energy does. The energy difference between every two successive points decreases as the number of probes increases, since the actuation energy becomes less significant, as explained by Amdahl's law.[2] A minimum point exists at 256 probes, after which energy starts increasing slowly. This increase is due to the additional overhead bits that need to be accessed (Eq. (1)), which becomes more noticeable (compare three to four bits of overhead per sector in Figure 10(a), respectively Figure 10(b)). Figure 11 reveals that the number of probes has a larger influence on response time than on energy consumption, since it influences the read/write time more than read/write energy.

7.3.1.3 *Capacity*. Several physical bits are needed per subsector to enable its accessibility (see Section 7.2). As the number of probes increases, the subsector size decreases and the relative overhead per sector increases. As a result, the (effective) capacity of the device decreases. Figure 11 shows the utilization

[2]Speeding-up a part of proportion f of a system by a factor s results in an overall speedup of $\frac{1}{(1-f)+\frac{f}{s}}$. The $(1-f)$ term of the denominator tells that the overall speedup is limited by the rest of the system that cannot be enhanced.

of the physical capacity of the device (approximately 3GB). The values of the capacity are normalized to the raw (physical) capacity of the device. Figure 11 shows a loss of 35% (approximately 1GB) in capacity when deploying 512 probes due to the separation bits and the ECC data.

Further, Figure 11 shows that the three design targets compete when designing a MEMS-based storage device: a gain in performance results in a loss in capacity. Also, performance gain can compete with energy reduction.

7.3.2 *Sector Parallelism (M).* Sector parallelism is the number of sectors that are simultaneously accessible from the storage medium. It deals with the number of probes a sector is striped across. If a MEMS-based storage device has N total active probes where each K probes access a sector at a time, the device can access M sectors simultaneously:

$$M = \frac{N}{K}. \tag{4}$$

7.3.2.1 *Performance.* Increasing the sector parallelism results in fewer probes per sector (K). As a consequence, the subsector size increases (Eq. (1)). Increasing the sector parallelism has one positive influence and two negative influences on the performance of MEMS-based storage devices. The positive influence is that increasing the subsector size reduces the overhead (Figure 10(b) versus Figure 10(a)), and thus decreases the read/write time of the overhead bits. On the other hand, one negative influence is that increasing the subsector size increases the number of data bits that a probe has to access, which increases the data read/write time. The second negative influence is underutilizing the sector parallelism, when the request size is not a multiple of the number of simultaneously accessible sectors. If a request demands L sectors from a MEMS-based storage device, which is capable of accessing M sectors simultaneously, the response time for the request (t_{request}) is

$$t_{\text{request}} = t_{\text{RW}} + t_{\text{seek}} \text{ and} \tag{5}$$

$$t_{\text{RW}} = \left\lceil \frac{L}{M} \right\rceil \cdot t_{\text{RW-subsector}}, \tag{6}$$

where $t_{\text{RW-subsector}}$ is the read/write time calculated per subsector as presented in Eq. (2). For example, accessing file D in the MEMS-based storage device shown in Figure 10(b) incurs underutilization of those probes associated with D_2 because it has no useful data. Eq. (5) shows that in addition to the read/write time, a seek time exists. The seek time includes the initial seek time in addition to the seek times incurred due to accessing noncontiguous sectors.

Figure 12 shows the response times for various sizes of the sector parallelism normalized to the response time when sector parallelism is 1 (24ms). It shows that a sector parallelism of eight sectors exhibits the shortest response time when deploying 256 probes. Setting the sector parallelism larger than eight sectors underutilizes the active probes and results in longer response times. Thus, sector parallelism can be tuned based on the characteristics of the expected workload to diminish the two negative influences.

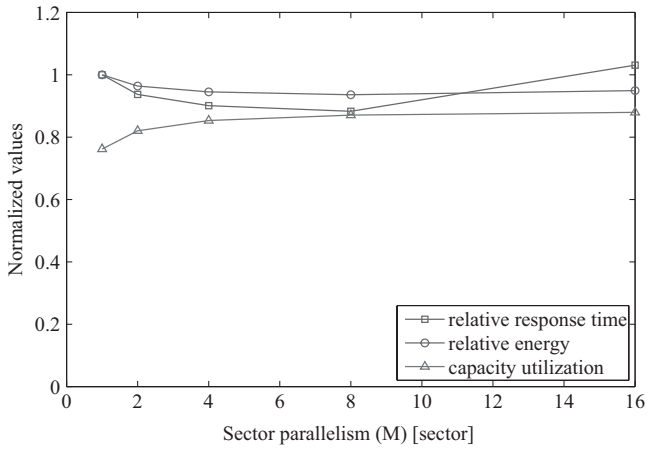

Fig. 12. Relative average response time, relative total energy consumption, and capacity utilization of the simulated MEMS-based storage device as a function of sector parallelism. Simulations are carried out for a total number of probes (N) of 256 probes and a sector size (S_{sector}) of 512 bytes. The reference values are 24ms, 11.2J, and 3GB.

7.3.2.2 *Energy.* A discussion similar to the performance applies to the energy consumption of MEMS-based storage devices as a function of sector parallelism. The total energy consumed to satisfy a request of L sectors is

$$E_{\text{request}} = N \cdot P_{\text{probe}} \cdot t_{\text{RW}}(L) + P_{\text{actuation}} \cdot t_{\text{RW}}(L) + P_{\text{seek}} \cdot t_{\text{seek}} . \quad (7)$$

In addition to the two negative influences on performance, there is a third negative influence on energy. As the parallelism increases, the subsector size increases, which extends the time the medium is held still on the X direction and increases the traveled distance along Y. As a consequence, the actuation energy increases. That said, tuning the sector parallelism, as done for the performance (in the face of the first two negative influences) and employing a larger number of probes simultaneously (in the face of the third influence) reduce energy consumption. Figure 12 shows that, indeed, the energy consumption decreases and is minimal for sector parallelism of eight sectors. We deploy 256 probes simultaneously to minimize the third influence, since it is the minimum in Figure 11. The values are normalized to the energy when sector parallelism is one sector (11.2J).

7.3.2.3 *Capacity.* Increasing sector parallelism has a positive influence on capacity because subsector size increases, and thus the overhead per sector decreases. Figure 12 shows that the loss in capacity of about 0.3GB (see Figure 11) is earned back by formatting with a sector parallelism of eight sectors. Better still, a further reduction in energy consumption as well as response time is attained.

7.3.3 *Sector Size (S_{sector}).* Eq. (1) shows that increasing the size of the logical sector increases the physical subsector size, which is also the result of increasing the sector parallelism. As the sector size increases, the subsector

size increases too, resulting in the same influences when increasing the sector parallelism (Figure 10(c)). The main difference between increasing the sector parallelism and increasing the sector size is that the former can underutilize probes if sectors are not requested. On the other hand, increasing the sector size can underutilize probes if the sector does not fully contain useful user data. Our analysis shows the same trends as those in Figure 12.

7.3.4 *Sector Parallelism versus Sector Size.* Sector parallelism and sector size are two seemingly similar solutions to increase the size of the subsector to mitigate the imposed overhead per subsector. However, sector parallelism and sector size have different effects on the usage of storage space, which in turn influences the response time, energy consumption, and capacity. Increasing the sector parallelism increases external fragmentation, since related sectors are not necessarily spatially colocated. For example, accessing sectors B_{11}, B_{12}, B_{21}, and B_{22} shown in Figure 10(b) cannot be entirely done in parallel, causing one more seek and a read or write of $B_{21}||B_{22}$ after $B_{11}||B_{12}$. On the other hand, increasing the sector size increases internal fragmentation because sectors are not fully utilized when the file system lacks intelligent placement techniques. For example, A_3 and A_4 in Figure 10(c) are wasted storage space.

External fragmentation increases seek and read/write operations, whereas internal fragmentation increases storage-space underutilization. Sector parallelism and sector size can be tuned based on the workload to enhance performance at a still large capacity.

7.4 Design Space

This section studies the design space of the data layout of MEMS-based storage devices composed of all feasible configurations of the three layout parameters listed in Section 7.1. As Table V shows, we consider seven different settings of the total number of probes, five settings of the sector parallelism, and also five settings of the sector size. All settings are a power of two, since the maximum number of probes and the sector size are power of two.

We present the three different views of a three-dimensional design space, where every configuration of the parameters (represented by a circle in Figures 13(a) to 13(c)) exhibits a certain response time, energy usage, and capacity when simulating against the `ext3−4K` trace. In total there are 175 configurations, out of which 20 configurations are infeasible because they either exhibit a subsector size smaller than 8 bits (the minimum) or larger than 2500 bits (the maximum).

Figure 13(a) plots the energy consumption versus the response time. We can identify two trends, referred to as trend A and trend B. Trend A shows that as the number of probes increases, the response time and energy consumption decrease. However, trend B shows that at a certain point the energy consumption increases as the number of probes increases because the energy to access the overhead bits becomes noticeable.

Figure 13(b) shows the effective capacity versus response time. Trend A shows that increasing the number of probes reduces the response time while retaining most of the device physical capacity. This trend corresponds to sector

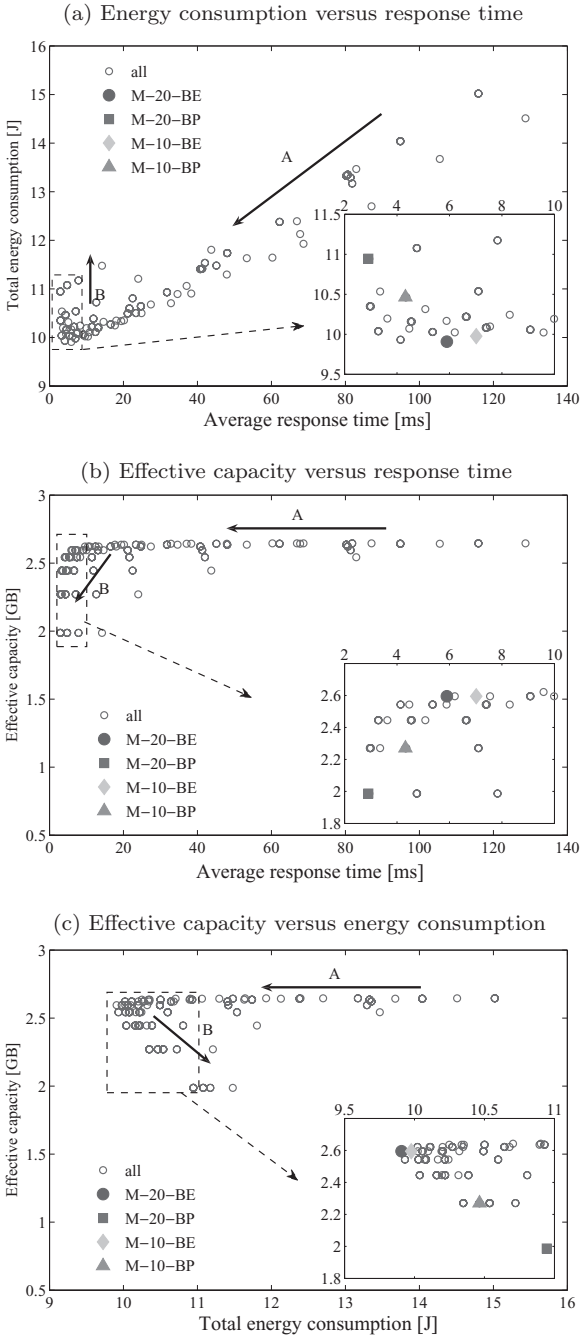

Fig. 13. Trade-offs between the three design targets (energy consumption, response time, and capacity) for the 155 feasible configurations when simulating against the `ext3−4K` trace.

parallelism larger than one sector and/or sector size larger than 512 bytes as shown in Figure 12. Unlike trend A, trend B shows that a loss in capacity occurs if the sector parallelism is one sector and/or sector size is 512 bytes as shown in Figure 11. By deploying large sector parallelism and/or sector size, we can retain a large part of the physical capacity at a negligible loss in response time as shown by the points around 2.5GB.

Figure 13(c) shows the effective capacity versus energy consumption. One trend similar to the previous figure can be observed, namely trend A. Trend B shows that a loss in capacity is accompanied by a loss in energy for configurations with a large number of probes. The reason is that employing many probes simultaneously increases the overhead per sector, causing a loss in energy as well as capacity, unlike trend B in Figure 13(b). Although increasing the overhead increases the response time, a larger decrease in response time occurs by decreasing the number of data bits per probe (see Figure 11), which results in an overall decrease in response time.

Zooming in on the parts where the optima could be found in Figures 13(a) to 13(c), we find that no overall optimal solution exists, but a set of Pareto optimal points. Thus, trade-offs are inevitable. We plot the best-energy (`M-20-BE` and `M-10-BE`) and best-performance (`M-20-BP` and `M-10-BP`) configurations when deploying 4096, respectively, 2048 probes. Here, "M" denotes MEMS; 20 corresponds to the nominal throughput of 20 MB/s, and "BP" denotes best performance. Note that the configurations with `M-20-*` are Pareto optimal,[3] whereas the others are not. We compare these configurations to Flash memory in the next section.

A discussion similar to that for the `ext3-4K` trace is apt for other traces as well. Workloads of different properties (i.e., request size and address alignment) have different optimal performance, energy, and capacity configurations. An overall optimal configuration does not exist, so that a trade-off is inevitable. We also compare the best configurations to a Flash memory for every trace next.

8. A CASE STUDY

In the previous three sections, we have provided a set of policies that increase the chance of MEMS-based storage devices for deployment in mobile systems. To put the enhanced MEMS-based storage device into perspective, we compare it to Flash memory. The comparison highlights the potential of MEMS-based storage technology and positions it with respect to other technologies. We compare the two technologies from two perspectives: response time and energy consumption.

In the following, we compare our simulated MEMS-based storage device with the CF Flash card for applications such as mobile phones and personal digital assistants (PDAs). We assume an environment where best-effort applications are intertwined with streaming applications. The share of each type of application depends mainly on the usage pattern of the mobile system. We simulate

[3]A Pareto optimal solution dominates other solutions on at least one account. In our work, a Pareto optimal configuration outperforms other configurations with respect to response time, energy consumption, or capacity.

against the four traces, each consisting of seven streaming sessions and nine best-effort sessions.

8.1 Methodology

The modeled MEMS-based storage device we use throughout this work has the power state machine (PSM) depicted in Figure 4 with the timeout set to 1 ms for mixed-media. The device employs the energy-efficient policy to shutdown (Section 6). Tables II and V list the settings of our modeled MEMS-based storage device.

We compare to Flash memory, since it exhibits short access time and energy efficiency. We choose the CompactFlash (CF) form because it has superior performance to smaller forms like the Multimedia Card (MMC) and the Secure Digital (SD) card. Our simulated MEMS-based storage device has an effective throughput of 10MB/s, which is within the range of that of the CF card. Note that a MEMS-based storage device has a small footprint ($41mm^2$), so that it can be housed in a CompactFlash (CF) package (and even in an SD package).

8.2 Configurations of the Data Layout

Designing a MEMS-based storage device constitutes a multiobjective optimization problem (see Section 7.4). The designer has to trade off between the design targets: response time, energy-efficiency, and capacity. In this section, we select several configurations of MEMS-based storage and compare them to Flash memory. The configurations have different settings of three data layout parameters: the total number of active probes, the sector parallelism, and the sector size.

We explore the design space of configurations as shown in Section 7.4 and select the overall-best configuration with respect to response time and energy consumption, called M-20-BP (best-performance configuration) and M-20-BE (best-energy configuration), respectively. The letter M denotes MEMS, and 20 denotes the nominal throughput $4096 \times 40\text{Kb/s} = 20\text{MB/s}$. These configurations are highlighted in Figures 13(a) to 13(c). We also present the best-capacity configuration (M-20-BC) to evaluate the loss in capacity resulting from the other two configurations. Thus, we have three configurations for MEMS-based storage in total. Details of these configurations are in Table VI.

The best energy and performance configurations have a nominal throughput of 20MB/s, whereas the SanDisk Standard CF card has a minimum read/write throughput of 10MB/s—although throughputs higher than 10MB/s were observed for this particular card type. To enrich our comparison, we additionally selected the overall best performance and energy configurations out of the configurations that employ just 2048 probes, which have a nominal throughput of 10MB/s. The configurations are called M-10-BP and M-10-BE, respectively, and the settings are presented in Table VI. Next, we discuss the comparison results for the `ext3−4K` trace.

8.3 Results

This section discusses the comparison results.

Table VI. Best Configurations of Our Simulated MEMS-Based Storage Device from a Response Time, Energy Consumption, and Capacity Perspective for the ext3−4KB Trace. (Each configuration is a tuple (total number of active probes, sector parallelism, sector size in KB). Here, "M" denotes MEMS; 20 corresponds to the nominal throughput of 20MB/s; and "BP" denotes best performance.)

Configuration	ext3−4KB
M-20-BP[a]	(4096, 1, 4)
M-20-BE[b]	(4096, 16, 4)
M-20-BC[c]	(64, 4, 8)
M-10-BP[d]	(2048, 1, 4)
M-10-BE	(2048, 16, 2)

[a] M-20-BP: Overall-best performance configuration out of the configurations that have a nominal throughput of 20MB/s.
[b] M-20-BE: Overall-best energy configuration.
[c] M-20-BC: Overall-best capacity configuration.
[d] M-10-BP: Best performance configuration out of the configurations that have a nominal throughput of 10MB/s.

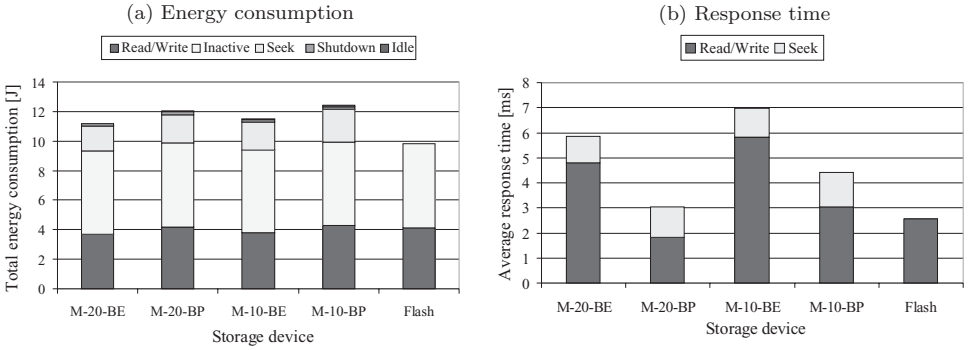

Fig. 14. Energy consumption and response time of the selected best-performance and best-energy configurations of our simulated MEMS-based storage device and the Flash card. The results correspond to simulations and measurements against the ext3−4KB trace. The capacity of the devices is 2.60GB (M-20-BE), 1.99GB (M-20-BP), 2.60GB (M-10-BE), and 2.27GB (M-10-BP), respectively.

8.3.1 *Results for the ext3−4K Trace.* Figures 14(a) and (b) show the energy consumption and response time of the MEMS-based storage configurations, respectively. We exclude the best-capacity configuration (2.65GB), since it exhibits a response time of approximately 116 ms, rendering it impractical.

Figure 14(a) shows that the Flash card consumes less energy than all configurations of the MEMS-based storage device. Further, Flash exhibits the shortest response time, as shown in Figure 14(b). The difference in energy consumption between the MEMS-based storage devices and the Flash card is between 14% and 27%. Recall that to simplify simulations, we model the seek operation with

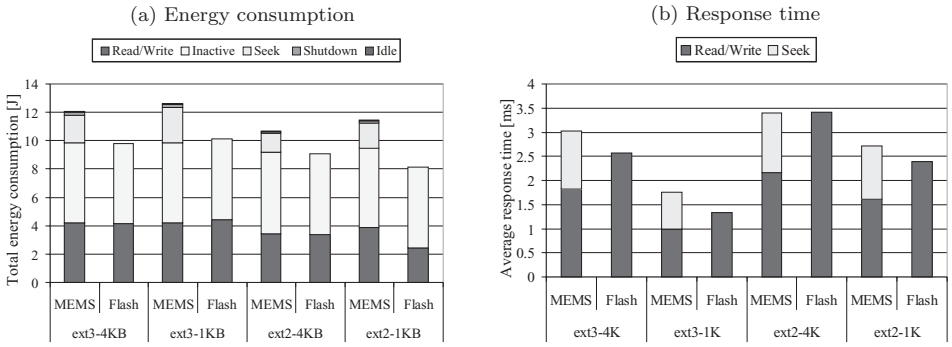

Fig. 15. Comparison between the overall-best performance configuration (M-20-BP) of our simulated MEMS-based storage device and the Flash card for the four traces of the `ext3-4KB` trace.

the time-optimal bang-bang model, which dissipates the maximum power. In reality, a feedback control system is deployed, which consumes less seek energy. Thus, the seek energy reported represents the worst case. The figure also shows the energy breakdown of the four MEMS-based storage devices and the Flash card. The prominent energy components of the MEMS-based storage device for any configuration are the read/write and inactive energy, like the Flash card.

The response time of MEMS-based storage devices varies greatly between configurations. The prominent component here is the read/write time, which varies from 2 to 6ms; the seek time is in the range of 1.0 to 1.5ms. Figure 14(b) shows that the M-20-BP configuration exhibits smaller read/write time than the Flash card. However, with the seek time added, the total response time becomes longer than that of the Flash card. The MEMS-based storage device has between 18% to 171% longer response time than the Flash card.

8.3.2 *Results for the Other Traces.* This section compares MEMS-based storage to Flash memory for the other three traces: `ext3−1KB`, `ext2−4KB`, and `ext2−1KB`. We select the overall-best performance configuration for each trace. The configurations are (4096,4,1), (4096,1,4), (4096,4,1), respectively. Figure 15(a) compares the energy consumption of these configurations with the Flash card. The figure shows that the MEMS-based storage device consumes between 17% to 41% more energy than the Flash card.

Figure 15(b) shows that MEMS-based storage devices exhibit shorter read/write time than the Flash card. However, with the seek time added, the response time of MEMS-based storage devices becomes longer. An exception exists for the `ext2−4KB` trace, whose corresponding best-performance MEMS-based storage device and the Flash card perform equally. Generally, MEMS-based storage devices exhibit between 0% and 31% longer response times.

9. RECOMMENDATIONS

Our comparison studies reveal the need for further enhancement of MEMS-based storage devices to increase their efficacy. In the following, we provide

several potential enhancements backed up by our analyses. We categorize our recommendations based on the design targets:

Energy Consumption and Throughput

Probe data rate. Increasing the attainable data rate per probe shortens the read/write time and reduces the energy consumption. The read/write time and energy are the first and second prominent components, respectively. The read/write time decreases when using more probes per sector, whereas the read/write energy is not influenced. Therefore, enhancing the inherent performance of the probe is necessary for energy efficiency. Further, higher data rates are attainable.

Response Time

Probe-field dimensions. The seek time of a MEMS-based storage device constitutes a large part of the response time. A MEMS-based storage device shuts down aggressively, and, hence, most of the seek times are in fact due to moving from the center (resting) position to the requested position. Reducing the dimensions of the probe field reduces the average traveled distance from the center, and thus reduces the seek time for the majority of requests.

Energy Consumption

Dynamic power gating. Employing dynamic gating to power probes allows to switch (sets of) unused probes on and off on demand. This reduces the read/write energy (the second most important energy component). If unused probes can be switched off, we can efficiently implement large sector parallelism. This is particularly important because, as our results reveal, MEMS-based storage devices are best configured with large sector parallelism to increase the effective capacity.

Probe power dissipation. A large number of probes are deployed to elevate the throughput of a MEMS-based storage device. This results in a relatively large power budget that is required to power-on the probes, approximately 1 W in our simulated device. The probe power dissipation should be reduced in order to target mobile devices as well as to provide SSD-like devices. The reduction in probe power directly reduces the read/write energy, the second prominent component of the total energy.

Actuators. Deploying a large number of probes reduces the actuation energy. Targeting Flash packages, that have a smaller power budget than CompactFlash (CF), such as SD (Secure Digital) and MMC (Multimedia Card), limits the power budget and thus the number of probes that can be deployed simultaneously. As a consequence, the fraction due to actuator power increases. For such small packages, energy-efficient actuators are needed.

Electronics. To save energy, MEMS-based storage devices shut down aggressively, and thus spend a large fraction of the time in inactivity. Consequently, the inactive energy is a significant energy component. MEMS-based storage devices as well as Flash memories can reduce the inactive energy by using lower supply voltage, by applying voltage and frequency scaling techniques, or even by switching off most of the electronics.

Power state machine (PSM). Power management is an efficient way to reduce the energy consumption of a MEMS-based storage device. To enforce the power management policy, a MEMS-based storage device must employ a power state machine (PSM) to track its operation modes. As a result, the device can decide on the time instance to shut down in order to save energy.

10. RELATED WORK

A large body of work exists that studies the enhancement and deployment of MEMS-based storage devices. We discuss it next.

Enhancement. Two works Schlosser et al. [2000] and Hong et al. [2006] address the energy consumption of MEMS-based storage devices. Schlosser et al. compare the energy consumption of MEMS-based storage devices with that of disk drives [Schlosser et al. 2000]. Using file-system benchmarks, Schlosser et al. show that MEMS-based storage devices consume 10× to 50× less energy than disk drives. To save on the idle energy, Hong et al. propose a power state machine of one low-power mode [Hong et al. 2006], and evaluate the energy saving by varying the shutdown timeout in the range of 0 to 50 ms. They use simulation driven by traces from server workloads. Hong et al. recommend to shut down the device immediately—by setting the timeout to zero—after request completion for maximum energy saving at an increase of 0.5 ms in response time on average.

Our work derives the power modes of a MEMS-based storage device, and augments the resulting power state machine with power figures from the IBM MEMS device. We explain why MEMS-based storage has no startup overhead, in contrast to previous work. We model electromagnetic actuators, whose power dissipation varies depending on the distance from the center. We systemically travel through each power mode and present a policy to reduce the energy consumption of MEMS-based storage devices.

Two earlier works by Sivan-Zimet [2001] and Schlosser [2004] configure (but do not investigate) the data layout of MEMS-based storage devices. Both keep the sector at the conventional size of 512 bytes. Sivan-Zimet [2001] configures the data layout, so that the sector parallelism is one sector, where just 320 probes are active at a time. Sivan-Zimet deploys all probes per one sector in order to enhance the throughput, since in her model the per-probe data rate is 1Mbps. Schlosser [2004] configures the data layout of the CMU G2 MEMStore so that the sector parallelism is 10. In his data layout, Schlosser stripes a sector across 64 probes, where 640 probes are active at a time. CMU G2 MEMStore has a per-probe data rate of 700Kbps and an 8-byte (ECC and physical) overhead per subsector.

Our research makes the case for exploiting the knowledge of the expected workload to configure the data layout of MEMS-based storage devices. Exploiting such knowledge helps the designer in finding small trade-offs between the response time, energy consumption, and the capacity of a MEMS-based storage device.

Researchers are working on increasing the effective capacity after formatting a MEMS-based storage device. After adding the coding bits to the user data, striping the sector across probes does not necessarily result in equally-sized subsectors. This leads a large loss in capacity, since probe fields fill unevenly. Varsamou and Antonakopoulos [2008] introduce a method that stripes and circulates subsectors across probe fields. The method corresponds to the low-level data layout.

Our data-layout configuration method works on a level right above Varsamou's method, since it determines the amount of user data that is a part of the sector. In other words, for any resulting sector size by our method, Varsamou's method is needed to achieve high space utilization.

As a mechanical storage device, seek time constitutes a large portion of the response time. Griffin et al. [2000b] investigate the applicability of the scheduling policies from disk drives to MEMS-based storage devices. Such policies are first-come, first-serve (FCFS) and shortest positioning time first (SPTF). They show that these policies can be employed in MEMS-based storage devices. They observe the same respective performance as in disk drives; FCFS and SPTF perform the worst and the best, respectively. Schlosser and Ganger [2004] propose shortest distance first (SDF) as a scheduling policy, which is specific to MEMS-based storage devices. This policy selects requests with the minimum Euclidean distance from the current position of the sled in both X and Y directions. Their experiments show that SDF performs worse than SPTF, and even other moderately performing policies. The reason is that SDF discards the fact that, as in disk drives, in MEMS-based storage devices repositioning (along the X direction) incurs expensive settling. As a result, Schlosser et al. conclude that scheduling policies from disk drives, which optimize along one direction, work well for MEMS-based storage devices. Hong et al. [2006] build on Sclosser et al.'s results and further customize SPTF for MEMS-based storage devices. They logically cluster the medium into several zones of different seek time profiles, and propose zone-based shortest positioning time first (ZSPTF), which prioritizes requests to the currently visited zone over requests to other zones. A variant of SPTF is proposed by Bahn et al. [2009]. Parallelism-aware shortest positioning time first (PA-SPTF) respects the probe parallelism and square structure of the MEMS-based storage device and reduces its response time by 39.2%.

Reducing the seek time results in decreasing the seek energy. We believe that the proposed seek algorithms work well in server applications. This is because in such applications intensive workloads exercise the storage device, so that intelligent scheduling improves the overall performance and prevents starvation. In mobile environments, however, these algorithms result in marginal benefits, since workloads are inherently light. In addition, as our research shows, the

device must shut down soon after a request to save energy, so that seeks are started from the center most of the time.

Deployment. Several roles have been proposed for MEMS-based storage devices: as (1) a disk cache [Hong et al. 2006]; (2) a streaming buffer and cache [Rangaswami et al. 2007]; (3) a replacement for disk drives [Schlosser et al. 2000]; and (4) a full and partial replacement of disks and nonvolatage random access memory (NVRAM) in disk arrays [Uysal et al. 2003]. The general consensus among these works is that MEMS-based storage can elevate the performance of server systems significantly, thanks to its inherent parallelism.

The small form factor, low cost, high rigidity, and high density of MEMS-based storage devices have motivated us to investigate the employment of these devices in mobile battery-powered systems. Enforcing the devised policies, we demonstrate the ability of MEMS-based storage devices to become competitive with Flash memory with respect to energy consumption and response time.

11. SUMMARY

This work devises three policies that enhance the energy-efficiency, timing performance, and the capacity of MEMS-based storage devices. We analyze the characteristics of MEMS-based storage devices, and subsequently devise the power state machine (PSM), which consists of five operation modes (seek, active, idle, shutdown, and inactive) and has no startup state, unlike HDDs.

We show that a MEMS-based storage device consumes approximately 40% of its total energy consumption when idle, so that power management is necessary. Simulation results show that a fixed-timeout power management policy achieves (near) optimal energy saving (95% of the idle energy). We also show that avoiding an immediate shutdown reduces the response time by 10%.

To allow for aggressive shutdowns, a MEMS-based storage device must reduce shutdown energy. We propose to exploit its unique structure to reduce the shutdown energy by using the potential energy of the springs to move the sled to the center position. Our simulations show a reduction by up to 10% in total energy compared to that when actuators are fully used during shutdown.

The third policy is the data-layout policy. We formulate the sector striping problem in MEMS-based storage devices and propose to exploit knowledge of the expected workload to configure the data layout. Simulation results show that such exploitation enhances energy efficiency and timing performance by approximately 10%, while increasing the capacity utilization by 10% relative to the case when no knowledge is exploited.

We use a Flash-based mobile setup for comparison with MEMS-based storage. Enforcing the three devised policies, we show that MEMS-based storage can be competitive with Flash with respect to energy consumption (17% to 41%) and response time (0% to 31%). Further enhancement is still required at the device level to increase the use of MEMS-based storage for mobile systems.

LIST OF ACRONYMS

CF	CompactFlash
ECC	Error-Correction Code
FCFS	First-Come, First-Served
SDF	Shortest Distance First
HDD	Hard Disk Drive
MEMS	Micro-Electro-Mechanical Systems
μSPAM	Micro Scanning Probe Array Memory
NVRAM	Non-Volatile Random Access Memory
PA-SPTF	Parallelism-Aware Shortest Positioning Time First
PDA	Personal Digital Assistant
PM	Power Management
PSM	Power State Machine
SPTF	Shortest Positioning Time First
ZSPTF	Zone-Based Shortest Positioning Time First

ACKNOWLEDGMENTS

Our gratitude goes to Johan B.C. Engelen and Leon Abelmann at the University of Twente for providing us with more insight into MEMS-based storage. We thank the anonymous reviewers for their constructive comments that improved the quality of the manuscript. We also thank the Parallel Data Lab at Carnegie Mellon University for providing us with DiskSim.

REFERENCES

ABELMANN, L., BOLHUIS, T., HOEXUM, A. M., KRIJNEN, G. J. M., AND LODDER, J. C. 2003. Large capacity probe recording using storage robots. *IEE Proc. Sci. Measur. Technol. 150*, 5, 218–221.

BAHN, H., LEE, S., AND NOH, S. H. 2009. P/PA-SPTF: Parallelism-aware request scheduling algorithms for MEMS-based storage devices. *ACM Trans. Storage 5*, 1, 1–17.

BENINI, L., BOGLIOLO, A., AND DE MICHELI, G. 2000. A survey of design techniques for system-level dynamic power management. *IEEE Trans. VLSI Syst. 8*, 3, 299–316.

BENNEWITZ, R., CRAIN, J. N., KIRAKOSIAN, A., LIN, J. L., MCCHESNEY, J. L., PETROVYKH, D. Y., AND HIMPSEL, F. J. 2002. Atomic scale memory at a silicon surface. *Nanotechnology 13*, 499–502.

BUCY, H. S., GANGER, G. R., AND CONTRIBUTORS. 2003. The DiskSim simulation environment version 3.0. Reference manual, School of Computer Science, Carnegie Mellon University.

CARLEY, L. R., BAIN, J. A., FEDDER, G. K., GREVE, D. W., GUILLOU, D. F., LU, M. S. C., MUKHERJEE, T., SANTHANAM, S., ABELMANN, L., AND MIN, S. 2000. Single-chip computers with microelectromechanical systems-based magnetic memory. *J. Appl. Phys. 87*, 9 III, 6680–6685.

GRIFFIN, J. L., SCHLOSSER, S. W., GANGER, G. R., AND NAGLE, D. F. 2000a. Modeling and performance of MEMS-based storage devices. *ACM SIGMETRICS Perform. Eval. Rev. 28*, 1, 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 & Implementation*. 227–242.

GURUMURTHI, S. 2005. Power management of enterprise storage systems. Ph.D. dissertation, The Pennsylvania State University, University Park, PA.

HONG, B., BRANDT, S. A., LONG, D. D. E., MILLER, E. L., AND LIN, Y. 2006. Using MEMS-based storage in computer systems|device modeling and management. *ACM Trans. Storage 2*, 2, 139–160.

HONG, B., WANG, F., BRANDT, S. A., LONG, D. D. E., AND SCHWARZ, T. J. E. 2006. Using MEMS-based storage in computer systems|MEMS storage architectures. *ACM Trans. Storage 2*, 1, 1–21.

JACOB, B., NG, S. W., AND WANG, D. T. 2008. *Memory Systems* (*Cache, DRAM, Disk*). Morgan Kaufmann.

Kernel Tracing. 2009. Kernel trace systems. http://elinux.org/Kernel_Trace_Systems.

KHATIB, M. G., ENGELEN, J. B. C., AND HARTEL, P. H. 2008. Shutdown policies for MEMS-based storage devices–Analytical models. Tech. rep. TR-CTIT-08-03, University of Twente, Enschede.

LAI, S. K. 2008. Flash memories: Successes and challenges. *IBM J. Res. Dev. 52*, 4/5, 529–535.

LANTZ, M. A., ROTHUIZEN, H. E., DRECHSLER, U., HÄBERLE, W., AND DESPONT, M. 2007. A vibration resistant nanopositioner for mobile parallel-probe storage applications. *J. Microelectromechanical Syst. 16*, 1, 130–139.

LU, Y.-H., CHUNG, E.-Y., SIMUNIC, T., BENINI, L., AND DE MICHELI, G. 2000. Quantitative comparison of power management algorithms. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*. ACM, New York, 20–26.

MCCARTHY, S., LEIS, M., AND BYAN, S. 2002. Larger disk blocks or not? In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST '02)*. USENIX, Berkeley, CA.

PANTAZI, A., SEBASTIAN, A., ANTONAKOPOULOS, T. A., BÄCHTOLD, P., BONACCIO, A. R., BONAN, J., CHERUBINI, G., DESPONT, M., DIPIETRO, R. A., DRECHSLER, U., DÜRIG, U., GOTSMANN, B., HÄBERLE, W., HAGLEITNER, C., HEDRICK, J. L., JUBIN, D., KNOLL, A., LANTZ, M. A., PENTARAKIS, J., POZIDIS, H., PRATT, R. C., ROTHUIZEN, H., STUTZ, R., VARSAMOU, M., WIESMANN, D., AND ELEFTHERIOU, E. 2008. Probe-based ultrahigh-density storage technology. *IBM J. Res. Dev. 52*, 4/5, 493–511.

PRALL, K. 2007. Scaling non-volatile memory below 30nm. In *Proceedings of the 22nd IEEE Non-Volatile Semiconductor Memory Workshop*. 5–10.

RANGASWAMI, R., DIMITRIJEVIĆ, Z., CHANG, E., AND SCHAUSER, K. 2007. Building MEMS-based storage systems for streaming media. *ACM Trans. Storage 3*, 2, 31.

SCHLOSSER, S. W. 2004. Using MEMS-based storage devices in computer systems. Ph.D. dissertation, Rep. CMU-PDL-04-104, Carnegie Mellon University, Pittsburgh, PA.

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 3rd USENIX Conference on File and Storage Technologies (FAST '04)*. USENIX, Berkeley, 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*. 1–12.

SEBASTIAN, A., PANTAZI, A., CHERUBINI, G., LANTZ, M., ROTHUIZEN, H., POZIDIS, H., AND ELEFTHERIOU, E. 2006. Towards faster data access: Seek operations in MEMS-based storage devices. In *Proceedings of the IEEE International Conference on Control Applications (CCA '06)*. IEEE, Los Alamitos, CA, 283–288.

SIVAN-ZIMET, M. 2001. Workload based optimization of probe-based storage. M.S. dissertation, University of California, Santa Cruz.

TOSHIBA. 2004. MK4001MTD: 0.85" HDD 4 GB. http://sdd.toshiba.com/main.aspx?Path=StorageSolutions/0.85inchHardDiskDrives/MK4001MTD.

UYSAL, M., MERCHANT, A., AND ALVAREZ, G. A. 2003. Using MEMS-based storage in disk arrays. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03)*. USENIX, Berkeley, CA, 89–101.

VARSAMOU, M. AND ANTONAKOPOULOS, T. 2008. A new data allocation method for parallel probe-based storage devices. *IEEE Trans. Magnetics 44*, 4, 547–554.

Received August 2009; revised December 2009; accepted December 2009