Power Management of MEMS-Based Storage Devices for Mobile Systems

Mohammed G. Khatib, and Pieter H. Hartel
Department of Computer Science, University of Twente
Enschede, the Netherlands
m.g.khatib@utwente.nl, p.h.hartel@utwente.nl

ABSTRACT

Because of its small form factor, high capacity, and expected low cost, MEMS-based storage is a suitable storage technology for mobile systems. MEMS-based storage devices should also be energy efficient for deployment in mobile systems. The problem is that MEMS-based storage devices are mechanical, and thus consume a large amount of energy when idle. Therefore, a power management (PM) policy is needed that maximizes energy saving while minimizing performance degradation.

In this work, we quantitatively demonstrate the optimality of the fixed-timeout PM policy for MEMS-based storage devices. Because the media sled is suspended by springs across the head array in MEMS-based storage devices, we show that these devices (1) lack mechanical startup overhead and (2) exhibit small shutdown overhead. As a result, we show that the combination of a PM policy, that fixes the timeout in the range of 1–10 ms, and a shutdown policy, that exploits the springs, results in a near-optimal energy saving yet at a negligible loss in performance.

Categories and Subject Descriptors

D.4.2 [**Operating Systems**]: Storage Management—*Secondary storage*

General Terms

Design, Experimentation

Keywords

Power Management, Energy, MEMS, Probe-Based Storage

1. INTRODUCTION

Users of battery-powered mobile systems require increasingly large storage capacities to store large amounts of digital content. However, the storage device in a mobile system must satisfy a stringent set of requirements: (1) exhibit a short response time, (2) consume little energy, and (3) have low cost per gigabyte. Disk drives, for example, cost 0.5 $/GB but consume too much energy for a small mobile system. Therefore, disks are used mostly in laptop computers. By contrast, flash is energy efficient, but flash costs about 6.0 $/GB and is thus mainly used in small mobile systems. An ideal storage device should be as performance- and energy-efficient as flash and as cheap as disk.

One storage technology that has the potential to satisfy these three requirements is storage based on Micro Electromechanical Systems (MEMS) [1,2]. Enabled by high storage densities above >1 Tb/in^2, MEMS technology promises to deliver a large capacity, a small form-factor, and a lower cost than flash. The challenge for MEMS-based storage devices is to be energy efficient to have chance in mobile systems. Because MEMS devices are mechanical and have a moving media sled, they consume a large fraction of energy when idle as shown in Figure 1. Therefore, MEMS-based storage devices must deploy power management to save energy effectively while not severely compromising on performance.

We propose to deploy a fixed-timeout power management (PM) policy in MEMS-based storage devices. This work quantitatively shows that the fixed-timeout PM policy is capable of achieving near-optimal energy saving at a negligible loss in performance. We carefully study the architecture and the work model of MEMS-based storage devices based on re-

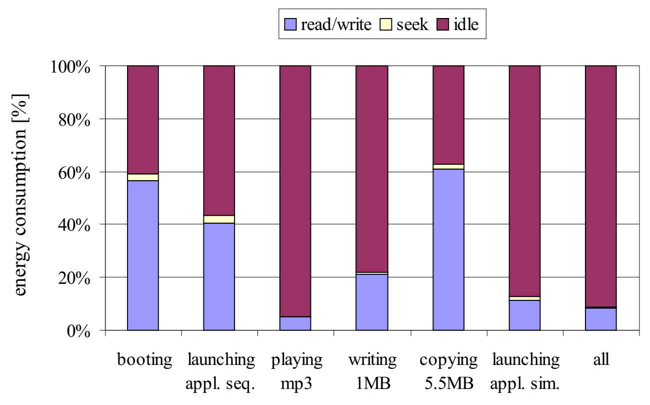
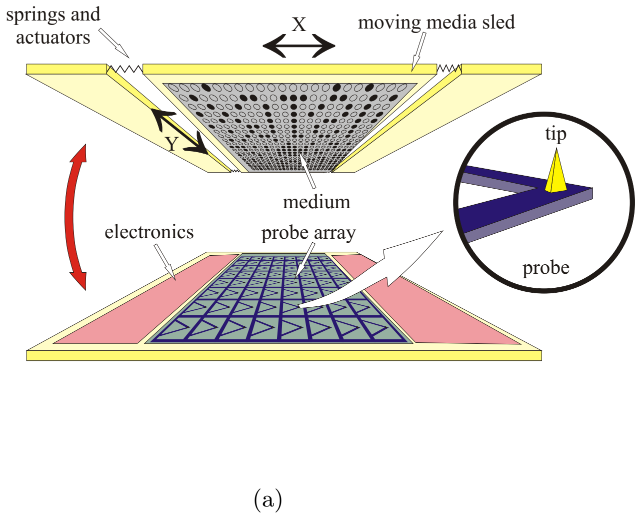


Figure 1: Energy breakdown for various applications of a MEMS-based storage device that has no power management. ***Idle energy constitutes at least 40% of the total energy.***



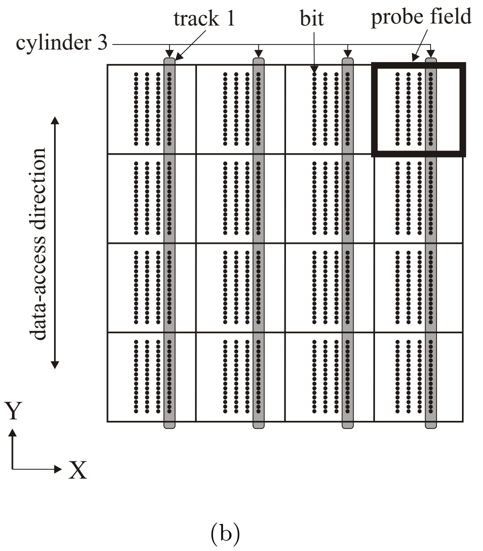


Figure 2: Three- and two-dimensional views of a MEMS-based storage device. ***(a) Two layers facing each other where the media sled is attached to springs that suspend it across the probe array. (b) The storage area of a simplified MEMS-based storage device consisting of*** 4×4 ***probes. The storage area is logically divided into 16 storage fields each accessible by a single probe.***

cent knowledge and figures from the prototyped IBM MEMS device [1]. Because the media sled is suspended by springs across the head array in MEMS-based storage devices, we show that these devices (1) lack startup overhead and (2) exhibit small shutdown overhead, if their spring structure is exploited at shutdown.

In this work, we deal with two complementary types of policies: the power management policy and the shutdown policy. While the power management policy decides *when* to shutdown the sled, the shutdown policy decides *how* to shutdown the sled. For power management, we propose a fixed-timeout policy. For shutdown we propose to exploit the springs to move the sled to its resting position. We show that the combination of a PM policy, that fixes the timeout in the range of 1–10 ms, and a shutdown policy, that exploits the springs, results in a near-optimal energy saving yet at a negligible loss in performance. Simulations using traces from a mobile system show that with this combination an energy saving of at least 95% is possible at a 4% increase in response time.

The remainder of this paper is organized as follows. In the next section, we briefly introduce MEMS-based storage. Section 3 studies related work. Section 4 discusses the distinctive characteristics of MEMS-based storage devices and modifies the power state machine proposed previously [3]. We carry out several simulations to study the energy efficiency. Section 5 presents the evaluation methodology and discusses the simulation results. Section 6 presents an energy-efficient shutdown policy, studies it statically, and tests it under real-world traces. The last section concludes.

2. BACKGROUND

Several design models for MEMS-based storage have been proposed [1, 4, 5, 2]. 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, one above the other, as shown in Figure 2a. 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, the IBM MEMS device [1] has a 64×64 probe array. Probes can be clustered in groups to reduce the complexity of the circuitry.

The top layer is the media sled on which bits are recorded. Bits can be recorded on a magnetic patterned medium as in µSPAM [5] and the CMU MEMStore [4]; a polymer medium as in the IBM MEMS device [1]; or a phase-change medium as in the Nanochip MEMS device [2]. 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 2b. Consequently, seek times shorten and a relatively high (aggregate) data rate is attainable by operating many probes simultaneously, so that each probe accesses a small part of a sector, called a *subsector*.

The media sled and the probe array in MEMS-based storage devices are separated by a distance of few nanometers, which is maintained actively by nanopositioners. Depending on the recording technique, the probes may make contact with the medium to read or write data. If the medium is magnetic, then data are read or written without contact, like in disk drives. Conversely, in the IBM MEMS device, the probes make contact with the medium only at read (or write) to create (or sense) depressions in the medium. No friction exists during seeks and during motion from one bit to another, because the probes touch the medium only on demand and repel after that (i.e., no steady contact). To access data on the medium, the media sled moves along the Y direction, along which data tracks lie as shown in Figure 2b.

While accessing data, the X actuators keep the sled still along the X direction on the accessed data track, counteracting the spring restoring force. When resting, the springs hold the sled at its resting position, where every probe faces the center position of its probe field.

As of early 2008, an IBM prototype [1] can record a single bit in an area of 26 nm by 26 nm, whereas Nanochip [2] claims a 15 nm by 15 nm bit cell area with the potential to reach a scale of 2 nm by 2 nm. With such high densities, a single memory chip has a capacity of 1 TB per die. These devices have potentially low cost for three reasons. Firstly, they can be manufactured using the well-established batch MEMS fabrication technology [1]. Secondly, these devices can be manufactured using micron-scale fabrication plants, whose equipment were installed ten years ago and have passed their break-even point, avoiding the need to build dedicated fabrication plants, unlike for flash memory. Thirdly, these plants can be used to make future generations of MEMS, since MEMS poses no requirements on the lithography process when increasing the density [2].

3. RELATED WORK

Power Management (PM) plays a key role to maximize the energy saving while controlling the performance penalty. A large body of work on PM policies exists; Benini et al. [6] give a survey. These policies fall under two general categories: static and dynamic policies. Dynamic policies achieve more savings at lower penalty than static policies do [7]. However, dynamic policies demand more processing and memory resources, because they have to keep a history of recent timeout values and power states. Dynamic policies can be also applied to MEMS-based storage devices. Contrasting MEMS-based storage devices with disk drives, we show (in the next section) that MEMS-based storage devices have distinctive characteristics, that invite us to study the fixed-timeout PM policies first. Indeed, our simulation results confirm the validity of our choice and show that less than 5% of energy-saving room with a performance degradation of 4% is left for more sophisticated policies to optimize for.

Previous work focuses on deploying MEMS-based storage devices as a cache or a replacement for disk drives in server systems [8, 9, 10]. There are two works, which study the energy consumption of MEMS-based storage devices [8, 3]. Schlosser et al. [8] study the integration of MEMS-based storage devices into computer systems. The authors propose one low-power mode and fix the timeout throughout their study, because they focus on comparing energy consumption of MEMS-based storage devices against disk drives. Using file-system benchmarks, they show that MEMS-based storage devices consume $10\times-50\times$ less energy than disk drives. Lin et al. [3] use the same MEMS model developed at CMU [11] and evaluate the energy saving by varying the timeout in the range of 0–50 ms using server traces. They recommend to shut down the device immediately (i.e., timeout is zero) after request completion for maximum energy saving at an increase by 0.5 ms in response time on average.

Since MEMS-based storage was still in its infancy, the model used by Schlosser and Lin is based on estimation, so that (a) high performance figures are assumed; compare the assumed 1000 Kbps to the actual 40 Kbps per-probe data rate. This model also assumes that MEMS-based storage devices (b) exhibit startup overhead and (c) lack shutdown overhead. Conversely, we show in the next section that MEMS-based storage devices lack startup overhead, because they have a spring-suspended media sled across the read/write heads, unlike disk drives. Further, unlike the previous work, we show that MEMS-based storage devices exhibit shutdown overhead to bring the media sled stationary. Subsequently, we (1) refine the CMU model according to points a–c and refine the model settings with figures from a prototype of the IBM MEMS-based storage device [1]

This work tailors MEMS-based storage devices to mobile systems and thus mainly focuses on energy consumption. (2) We do not only study the fixed-timeout power management policy, but also quantify its energy saving (and the corresponding performance degradation) to reason about its optimality. In addition to the power management policy, (3) we derive and implement a model for an energy-efficient shutdown policy that exploits the spring structure. (4) We study for different I/O settings and therefore capture traces for the `ext3` (journaling) and `ext2` (non-journaling) file system. Each file system is formatted with a block size of 1 KB and 4 KB. Doing so, our conclusions are applicable to a broad range of mobile systems.

4. MODELING

This section contrasts MEMS-based storage devices with disk drives. Based on this contrast and an analysis of the distinctive characteristics of MEMS devices, we derive a power state machine for MEMS devices.

4.1 Characteristics of MEMS

MEMS-based storage devices enjoy unique architectural and physical characteristics. Among their several architectural characteristics, MEMS-based storage devices exhibit (1) a spring-suspended media sled, (2) nanopositioners along the Z direction, (3) independent motions along the X and Y directions, and (4) a large number of probes sharing one medium (64×64 probes). The physical characteristics are: (5) the light-weight media sled (0.1 g), (6) the high storage density (1 Tb/in^2), and (7) the small dimensions of the device (41 mm^2). In contrast to MEMS-based storage devices, in the smallest disk drive, namely the Toshiba 0.85-inch disk drive, the single platter weighs 1 g, the accessible platter space per head is approximately 366 mm^2, the storage density is 30 Gb/in^2 and the drive has almost the same dimensions as its platter: about 366 mm^2.

To access a disk drive, that is in inactive mode, the platters should 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, which is created by the spinning platters [12]. This spinup activity takes several seconds depending on, among others, the mass of the platters and the target spinning speed. After spinup, the head seeks to the addressed track, which takes up to 20 ms. Once the head is above the right track, the head may wait for the platter to rotate until the addressed data block comes beneath the head for reading or writing. Thus, three activities take place: (1) spinup, (2) seek, and (3) rotation. At shutdown, the head is unloaded first and then the platters are spun down.

Unlike in disk drives, in MEMS-based storage devices, the media sled is suspended by springs across the probe array at a specific distance (see Figure 2a), which is actively maintained by the Z nanopositioners (characteristics 1 and 2). As a result, no startup latency and thus startup energy exist.

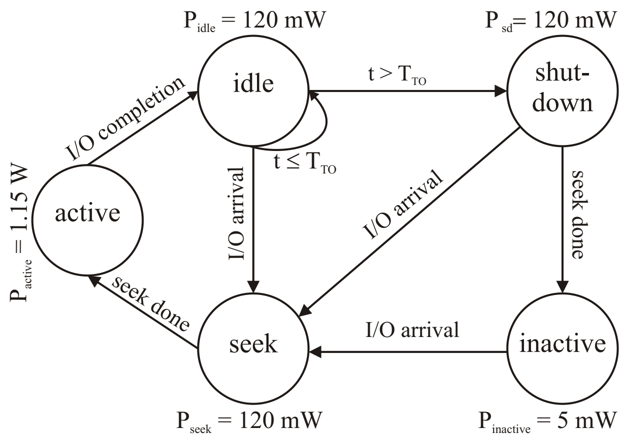


Figure 3: Power State Machine (PSM) of MEMS-based storage devices

To access a MEMS-based storage device, that is in inactive mode, the media sleds seeks along X and Y simultaneously to the addressed data block, since they can move independently at the same time (characteristic 3). Therefore, the seek time is the maximum of the seek times along X and Y and the difference between the seek times along X and Y resembles the rotational latency in disk drives. Characteristics 5 through 7 enable the realization of a high capacity storage device in a small form factor. Further, characteristic 4 shortens the sweep area for one probe ($0.01\,\text{mm}^2$). As a result, seek times are short (less than 2 ms). At shutdown the sled is brought stationary at its resting position, namely the center of the probe field. Like the seek time, the shutdown time and thus energy is also short, thanks to characteristics 4–7. In Section 6, we show that the springs (characteristic 1) can be exploited to reduce the shutdown energy further.

In disk drives, startup, shutdown, and seek energy and latency are incurred every time the disk shuts down and subsequently starts up. Unlike disk drives, MEMS-based storage devices have no start-up overhead, but have small seek and shutdown overheads. The small overheads suggest implementing a simple and aggressive PM policy for MEMS-based storage devices to reduce energy. This leads us to present the power model discussed in the next section.

4.2 Power state machine

MEMS-based storage devices consume a large amount of energy when the media sled is moving (see Figure 1). Therefore, to save energy the sled should be shut down if the device is idle and its request queue is empty. However, resting the probe in the center position of its storage field can increase the seek distance for the next request, if it addresses the same region of the previous request. This is the case for real-world workloads, which exhibit spatial locality of reference (i.e., consecutive requests address nearby locations) optimized by the I/O subsystem. A long seek distance results in a long seek time and thus a long response time. As a result, energy saving should be traded off for response time. Based on the discussion in Section 4.1, we deploy a Power State Machine (PSM) with one inactive mode, in which the sled stands still. We deploy a fixed-timeout PM policy to transition into the inactive mode. 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.

Table 1: Settings of the model of the simulated MEMS-based storage device

Total # of probes	64×64	probes
# of active probes	4096	probes
Bit/Track pitch	40	nm
Probe sweep area	100×100	μm^2
Per-probe data rate	40	Kbps
Capacity	2.0	GB

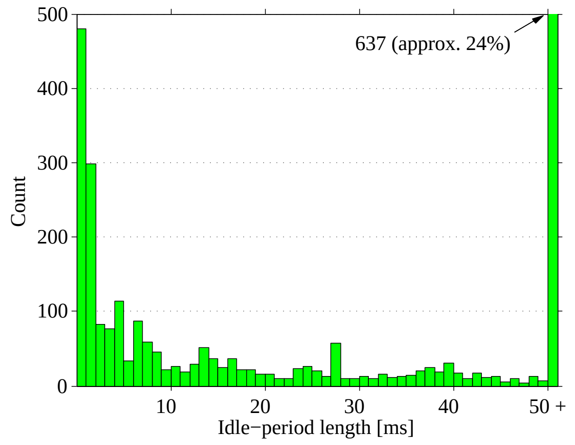
Figure 3 shows the proposed PSM of MEMS-based storage devices. It consists of five states: seek, active, idle, shutdown and inactive. In the *seek state* the media sled moves from its current position to the starting position of the next request to service it. Seeking dissipates 60 mW [1] per direction, amounting to 120 mW in total. In the *active state*, the device accesses (i.e., reads or writes) data, where the sled dissipates 60 mW to move along the Y direction and another 60 mW to stay still in the X direction, since voice-coil actuators are used. The 4096 probes and the error-correction electronics dissipate 1 W to read/write and correct data (approx. 0.25 mW per probe and its electronics). In the *idle state*, the device waits for requests. The sled moves along Y, while staying still in X, dissipating 120 mW in total. In the idle state, the device anticipates requests to nearby locations. Upon request arrival, the device goes back into the seek state. Otherwise, if within a certain time interval, that is equal to the timeout (T_{TO}), no request arrives, the device goes into the shutdown state. In the *shutdown state*, the sled travels to the center (resting) position from its current position. The travel time depends on the distance. If a request arrives while shutting down, the device goes into the seek state immediately. Otherwise, if no request arrives and the sled reaches the center position, the device goes into the inactive state. In the *inactive state*, the sled rests in the center position, the probes are switched off, and the interface awaits requests, dissipating 5 mW. No startup state exists as explained previously.

5. EXPERIMENTS

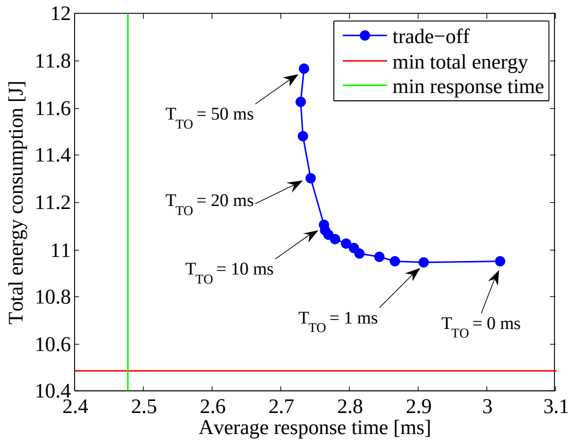
In the following, we report on our simulations for a MEMS-based storage device using the PSM presented previously. We explain our setup first.

5.1 Methodology

IBM demonstrated a MEMS-based storage device of 64×64 probes. Although their prototype is not available for experiments, sufficient specification data are available in the literature [1]. We use trace-driven simulations to evaluate the fixed-timeout PM policy. We use the DiskSim simulator [13]; a validated modular simulator for simulating various types and architectures of storage subsystems. We refine the energy model of the seek operation in the CMU MEMS model [11] to account for non-constant power dissipation across the medium. Also, we refine the CMU model to include time and energy models for the shutdown [14]. The models use the bang-bang optimal control model, which captures the dynamics of the system and factors in all forces during the sled motion [15, 16]. Further, all parameters including, the bit dimensions and the per-probe data rate of



(a)



(b)

Figure 4: (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 with the whole ext3–4K trace;** ***Timeout values 1–10 ms make little difference in energy consumption (approx. 2%) but vary in response time (approx. 10%).***

the model are set to those of the IBM MEMS device [1]. Table 1 summarizes the key settings.

Since our design is targeted at mobile applications, we gathered traces on an HP iPAQ H2215 PDA. An embedded version of Linux (kernel version 2.6.17) has been ported onto this PDA. Jens Axboe's block trace utility [17] was used to log I/O events, which are forwarded to a host machine, so that the gathered traces were not contaminated by the operations needed to store trace records locally. The CF card functioned as the main storage device on which the root file system was located. As a result, all I/O activities went to and from the CF card. We logged different system and application activities. System activities included booting and starting the Graphical User Interface, whereas application activities include: firing applications, such as the text editor and web browser; taking photos; streaming audio and video from/to the storage device; and creating, copying and deleting files. We also measured the energy consumption of the CF card and recorded it on the host machine for energy comparison.

MEMS-based storage devices are expected to serve as storage devices in future computer systems. They will communicate with the file system layer as flash memories do at present. 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. To strengthen our simulation study, we traced and simulated with different settings of the I/O subsystem. We captured the aforementioned scenarios on `ext3`, the default Linux file system. We also captured on `ext2`, a non-journaling version of `ext3`. In addition, we formatted each file system with the default maximum block size, 4 KB, and a smaller one, 1 KB.

We use the power model presented in the previous section with different integral values of the timeout (T_{TO}). The timeout values are $0 - 10$ ms. We then increase the timeout by 10 up to 50 ms, because the traces have long, but few, idle periods. The models and traces we use in this work are available for reproducibility [18]. The following section presents and discusses the simulation results.

5.2 Results

In this section, we discuss in detail the simulation results. The results present the trade-off between the energy consumption and the response time (i.e., the performance penalty) for different values of the timeout. We first discuss the trade-off when simulating with the whole ext3–4K trace; all scenarios combined. After that, we present the results for the other three traces (ext3–1K, ext2–4K, and ext2–1K), when simulated as a whole. Last, we discuss the results of each scenario individually.

5.2.1 Whole ext3–4K trace

Power management exploits idle periods to save energy. Thus, the distribution of the idle-period length for a given trace/workload shows the potential in energy saving and the compromised performance. That is, the longer the idle period, the larger the energy saving (or the less the energy consumption). In general, the more frequent the idle period, the longer the response time, since real workloads exhibit some locality, so that seeking from the center increases the response time. The length of the idle period is influenced by the inter-arrival time, the size, and the seek distance of the requests.

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, because longer seek distances are incurred. Figures 4a and 4b show the histogram of idle-period length and the energy–performance trade-off, respectively, when simulating with the whole ext3–4K trace. About 48% of the idle periods lie in the range of 0–10 ms.

Figure 4b shows the total energy consumption for the whole trace versus the average response time per request. It shows a decrease of 0.3 ms (approx. 10%) in average response time between $T_{\mathrm{TO}} = 0$ ms and $T_{\mathrm{TO}} = 10$ ms. This is

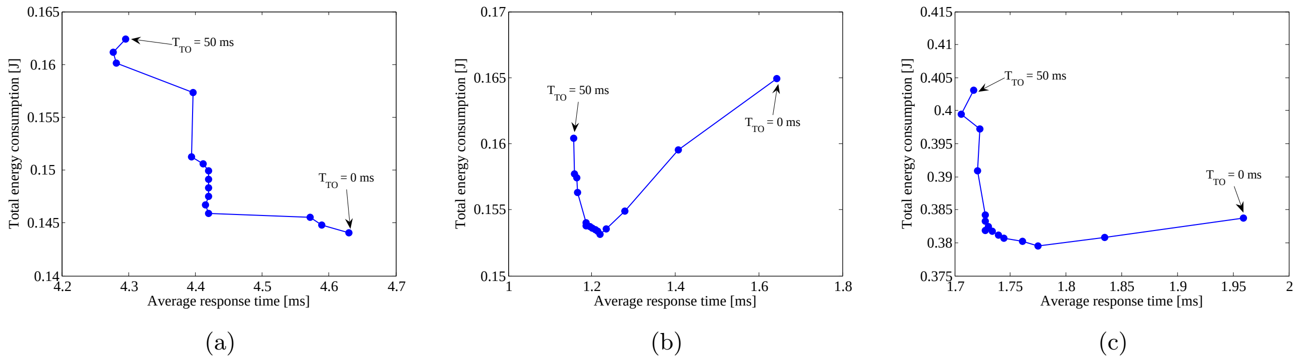


Figure 5: Performance-energy trade-offs for three scenarios of the ext3-4K trace: (a) writing a picture, (b) copying files and directory, and (c) firing several applications simultaneously

explained in Figure 4a, where the size of this decrease is proportional to the sum of the heights of the first ten buckets. When $T_{\text{TO}} = 10\,\text{ms}$, the first ten buckets are not exploited, avoiding longer seeks from the center position. In general, these buckets mainly influence the performance, because of their relatively high occurrence frequency. As the timeout increases, less performance penalty is incurred, because of the infrequent occurrence of large idle periods.

Figure 4b shows a slight decrease in energy consumption when $T_{\text{TO}} = 1\,\text{ms}$ compared to $T_{\text{TO}} = 0\,\text{ms}$. The longer seeks explained above do 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 (approx. 2%) in energy consumption is seen in the range $T_{\text{TO}} = 1 - 10\,\text{ms}$, because of the small energy-saving contribution 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 example, is more profitable than 30 occurrences of an idle period of length of 1 ms.

To quantify the optimality of the energy saving of the fixed-timeout PM policy, we define a *theoretical energy minimum*: the sum of (1) the energy consumed to read/write data, (2) the energy consumed due to seeks, which are caused by requests only and not due to PM, and (3) the inactive energy. Figure 4b shows that the energy consumption at $T_{\text{TO}} = 10\,\text{ms}$ is within 6% of the theoretical minimum (approx. 0.6 J absolute difference). We also define a *theoretical performance minimum*: the sum of (1) the transfer time and (2) the seek time due to requests only and not due to PM. Figure 4b shows that the response time difference is within 12% of the theoretical minimum (approx. 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 $T_{\text{TO}} > 10\,\text{ms}$) due to the inevitable shutdown and seek-from-center overheads. Thus, $T_{\text{TO}} = 10\,\text{ms}$, for example, is optimal as its performance penalty does not exceed 4% compared to others.

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 2 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 1–10 ms and not at $T_{\text{TO}} = 0\,\text{ms}$ or $T_{\text{TO}} > 10\,\text{ms}$. Another important observation is that increasing the timeout in the range 1–10 ms decreases the response time by a larger factor than it 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.

Table 2: Energy consumption versus response time for our 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
theoretical min.	10.5 / 2.5	9.8 / 1.4	9.6 / 2.8	9.4 / 2.2

Thanks to the distinctive characteristics of MEMS-based storage devices (see Section 4.1), no startup energy exists, so that the relative difference between the energy consumption when $T_{\text{TO}} = 10\,\text{ms}$ and the theoretical minimum does not exceed 7%, leaving very little room for improvement to dynamic policies. Thus, deploying a fixed-timeout Power Management (PM) policy with a PSM of one low-power mode is sufficient to achieve a near-optimal energy saving at low performance penalty.

5.2.3 *Scenarios separately*

As mentioned earlier, our traces consist of several scenarios each corresponding to an individual application or system scenario. The scenarios are (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 necessarily all of these scenarios occur together, we

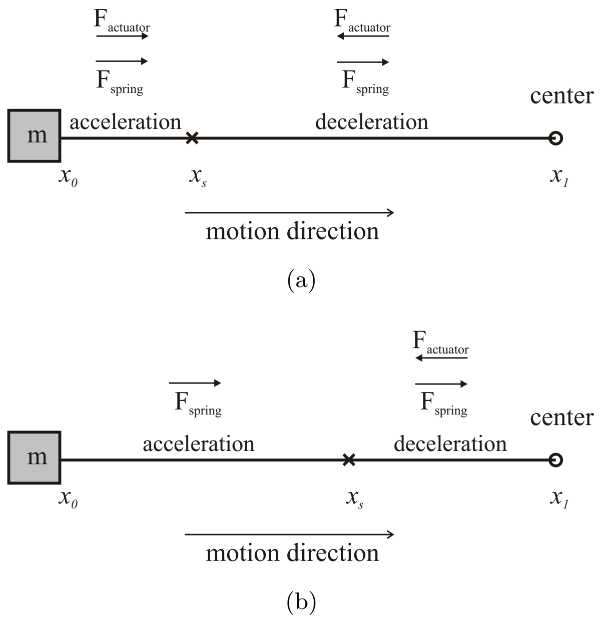


Figure 6: (a) The performance-efficient shutdown policy versus (b) the energy-efficient policy. ***The former uses the actuators for acceleration and deceleration, whereas the latter uses the actuators just for deceleration.***

split the traces into their respective scenarios and simulate for each individual scenario.

The simulation results of the scenarios in all traces agree with those of the entire traces. Figures 5a, 5b, and 5c show the performance–energy trade-off for scenarios 4, 6, and 7, respectively. The key observation is that setting the timeout larger than 0 decreases the response time by large part for a relatively negligible increase in energy consumption, if any. If a workload exhibits a large percentage of idle periods that are shorter than 1 ms, setting the timeout to 0 not only worsens the response time, but also the energy consumption, since the shutdown energy becomes prominent. This is exactly the case for scenario 6 shown in Figure 5b.

6. ALTERNATIVE SHUTDOWN POLICY

In the first part of this paper, we study the power management policy which decides when to shutdown the media sled of a MEMS-based storage device. We demonstrate the effectiveness of the fixed-timeout power management policy, thanks to the spring-suspended sled, so that no startup overhead exists. In the rest of the paper, we deal with a policy that complements the power management policy, namely the *shutdown 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

In the previous part, we experiment with a shutdown policy that employs the actuators to move the sled to the center. Figure 6a illustrates how the actuators accelerate the sled for some distance before they reverse to decelerate it so it stops at the center. The actuators consume energy in acceleration and deceleration. The invested energy shortens the shutdown time and thus we call this policy the *performance-efficient shutdown policy.*

In this work, we propose another shutdown policy that uses the potential energy stored in the springs. The springs bring the sled as close as possible to the center (Figure 6b) before the actuators starts to decelerate the sled so it stops at the center. This policy consumes less energy than the previous policy, since it employs the actuators for deceleration only as Figure 6b illustrates. Thus, 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 actively accelerated.

We devise analytical models for the energy-efficient (EE) and performance-efficient (PE) shutdown policies. Due to space limitations, we provide a complete modeling study of the shutdown times along X and Y in a separate technical report [14]. We integrate these models into DiskSim to compare them under real-world traces and to study the interaction between the power management policy and the shutdown policy. For a better understanding of the behavior of the two policies, we provide an analytical (static) study first. The analytical study compares both policies when shutting down from every position within a probe field. After that, we follow up with a trace-based (dynamic) simulation that compares both policies in real environments.

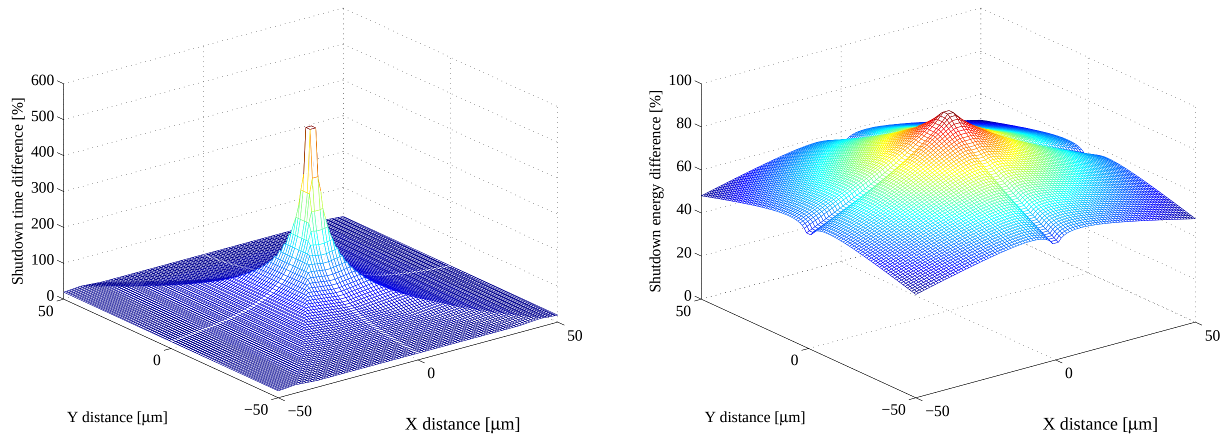
6.2 Analytical study

Before comparing the behavior of the two policies under real-world traces, we compare their shutdown behavior from every position within a probe field. The parameters of the models of both policies are set to the state-of-the-art figures of the IBM MEMS device [1]. Figure 7 shows the relative difference in shutdown time and energy when shutting down from every position within a probe storage field ($100\,\mu\text{m}\times 100\,\mu\text{m}$). The resting position is the center at the coordinate (0,0).

6.2.1 Comparison

Figure 7a shows that the relative difference in shutdown time (calculated as $\frac{t_{\mathrm{EE}}-t_{\mathrm{PE}}}{t_{\mathrm{PE}}}$) is minimal at the borders of the probe area. It, however, increases as the starting position gets closer to the center. When deploying the EE policy, the sled accelerates to the center by the spring force only. This force depends on the distance between the sled's current position and the center. 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 µm and 45 µm distance (in a range of 0–50 µm) is 1.6 ms and 2.0 ms, respectively. 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 5 µm and 45 µm distance is 0.6 ms and 1.6 ms, respectively. This explains why the difference between both policies increases as the starting position gets closer to the center. The difference nearby the center is orders of magnitude longer than that at the borders, explaining the prohibitive by large relative difference.

Figure 7b plots the relative difference in energy consump-



(a) (b)

Figure 7: Relative difference in (a) shutdown time and (b) energy between the performance-efficient (PE) shutdown policy and the energy-efficient (EE) policy. ***The difference for every position within a probe field is calculated as*** $\frac{t_{EE}-t_{PE}}{t_{PE}}$ ***and*** $\frac{E_{PE}-E_{EE}}{E_{PE}}$***, respectively.***

tion (calculated as $\frac{E_{\mathrm{PE}}-E_{\mathrm{EE}}}{E_{\mathrm{PE}}}$). The shutdown energy when deploying the PE policy is larger than when deploying 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 part, 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.

6.2.2 Discussion

Our analytical study shows that the energy-efficient shutdown policy saves more energy than the PE policy in two cases. These cases are (1) when data are located at the borders and (2) when data at the center are frequently accessed. The saving is at least 40% relative to the PE policy.

The EE policy shows worse timing performance than the PE policy however. This is especially true around the center, where the difference can be up to 600%, because of the (extremely) slow behavior of the EE policy in reaching the center. That said, the slow shutdown performance can be of an advantage to real-world applications for two reasons. Firstly, from a cost viewpoint, shutdown is an overhead and not an inherent task of accessing data. That means it should be done as cheap as possible and not as quickly as possible. Secondly, the slow motion benefits those applications that exhibit (high) sequentiality and/or locality of reference. In other words, moving the sled slowly allows for quick inexpensive seek to an already visited region, if new requests demand further data from this region as we observe in our simulations show presented next.

6.3 Trace-based study

We integrate the models of both policies in DiskSim [13] to evaluate the performance and energy influence of each shutdown policy. We repeat the experiments discussed in Section 5.2 for all traces with the energy-efficient shutdown policy under real-world traces.

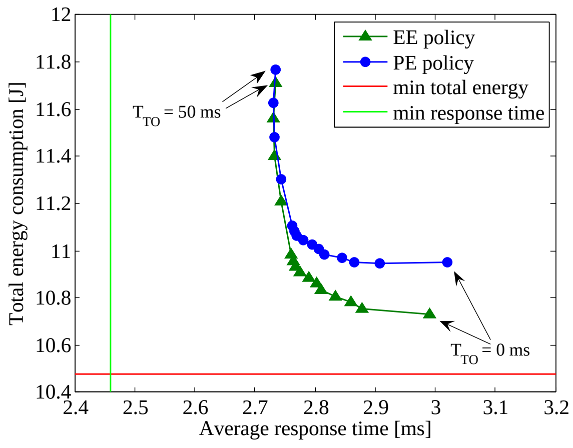
6.3.1 Whole ext3–4K trace

Figure 8a plots the trade-off between response time and energy consumption when deploying the EE shutdown policy and performance-efficient policy for the ext3-4K trace. We see that the energy decreases and becomes even closer to the theoretical minimum, further supporting the effectiveness of the fixed-timeout power management policy. At $T_{\mathrm{TO}} = 0\,\mathrm{ms}$, with the EE policy a MEMS device consumes less energy than with the PE policy, since shutdown energy (i.e., the overhead) becomes smaller.

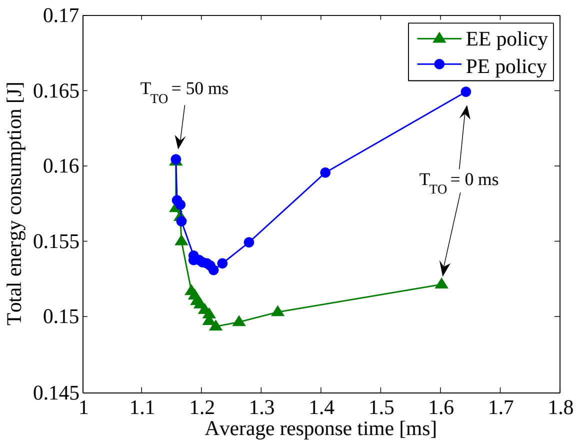
Another finding, in contrast to the static study, is that the EE policy (slightly) shortens the response time compared to the PE policy, thus it is also performance efficient as well. The reason is that the EE policy shortens the response time for sequential requests, since it moves the sled slowly to the center. As a result, driving the sled back to a previously visited region takes less time and thus consumes less energy. The difference is noticeable for the small values of the timeout, because of the large occurrence frequency of small idle periods as shown in Figure 4a.

The difference in response time between the two policies is negligible for this workload, because it contains streaming activities, which have large response times (since they access large amounts of data). For workloads that lack streaming activities the difference becomes more pronounced as shown in Figure 8b for scenario 6 (copying files and directories). We concluded in Section 5.2 that setting the timeout $T_{\mathrm{TO}} > 0\,\mathrm{ms}$ shortens the response time at a slight increase in energy consumption. Figure 8a confirms the validity of this conclusion when deploying the EE policy.

Figure 8 shows also that the theoretical performance minimum can not be achieved even by large values of the timeout. 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



(a)



(b)

Figure 8: Energy–performance trade-off when deploying the performance-efficient (PE) policy and energy-efficient shutdown (EE) policy shown for simulations with (a) the whole ext3-4K trace and (b) scenario-6 trace.

Table 3: Energy consumption versus response time for the four traces when deploying the energy-efficient shutdown policy

T_{TO} [ms]	Energy consumption [J] / Response time [ms]			
	ext3–4K	ext3–1K	ext2–4K	ext2–1K
0	10.7 / 3.0	10.4 / 1.8	9.8 / 3.3	9.7 / 2.6
1	10.8 / 2.9	10.3 / 1.7	9.8 / 3.2	9.7 / 2.6
2	10.8 / 2.9	10.4 / 1.7	9.8 / 3.2	9.7 / 2.6
3	10.8 / 2.8	10.4 / 1.7	9.8 / 3.2	9.7 / 2.6
4	10.8 / 2.8	10.4 / 1.6	9.8 / 3.2	9.8 / 2.6
5	10.9 / 2.8	10.4 / 1.6	9.8 / 3.2	9.8 / 2.5
10	11.0 / 2.8	10.5 / 1.6	9.9 / 3.2	9.8 / 2.5
20	11.2 / 2.7	10.7 / 1.6	9.9 / 3.2	9.9 / 2.5
30	11.4 / 2.7	10.9 / 1.6	10.0 / 3.2	10.0 / 2.5
40	11.6 / 2.7	11.1 / 1.6	10.0 / 3.2	10.1 / 2.5
50	11.7 / 2.7	11.2 / 1.6	10.1 / 3.2	10.1 / 2.5
theoretical min.	10.5 / 2.5	9.8 / 1.4	9.6 / 2.8	9.4 / 2.2

mechanical devices there is always a minimum seek distance that is incurred regardless of the timeout.

Hence, we can conclude that the timeout $T_{\mathrm{TO}} = 10\,\mathrm{ms}$ achieves a near-optimal energy saving (approx. 95%) at a negligible loss in performance (4%), relative to the actual minimum represented by the line that connects all the points for $T_{\mathrm{TO}} \geq 10\,\mathrm{ms}$ in Figure 8a.

6.3.2 *Other traces*

We repeat the simulation for the other three traces with the energy-efficient shutdown policies. As Table 3 shows, we observe the same trend as for the ext3-4K trace. Our conclusion to deploy a timeout in the range of 1–10 ms still holds as well.

7. CONCLUSIONS

We enhance the energy efficiency of MEMS-based storage devices for deployment in mobile systems. MEMS-based storage devices have a moving media sled and thus consume a large fraction of energy when idle. Therefore, power management (PM) is needed.

This work quantitatively shows the optimality of the fixed-timeout power management policy for MEMS-based storage devices. Because the moving media sled is suspended by springs across the head array in MEMS-based storage devices, these devices lack mechanical startup overhead. To enhance the energy saving of the fixed-timeout policy further, we propose a to exploit the spring structure to reduce the shutdown energy.

We gather traces on a flash card plugged into a PDA for simulation for different usage scenarios, file systems, and block sizes. We simulate for each usage scenario individually and all scenarios combined. Our simulations show that a combination of a PM policy, that has a timeout in the range of 1–10 ms, and a shutdown policy, that exploits the spring structure, results in an energy saving of at least 95% at a 4% increase in response time.

8. ACKNOWLEDGEMENTS

We wish to thank the Parallel Data Lab at Carnegie Mellon University for sharing DiskSim with us. Also, 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 devices. This research is supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Dutch Ministry of Economic Affairs under project number TES.06369.

9. REFERENCES

[1] M. A. Lantz, H. E. Rothuizen, U. Drechsler, W. Häberle, and M. Despont, "A Vibration Resistant Nonopositioner for Mobile Parallel-Probe Storage Applications," *Journal of Microelectromechanical Systems*, vol. 16, pp. 130–139, February 2007.

[2] "Nanochip Inc.." `http://nanochipinc.com/tech.htm`. Accessed in November 2007.

[3] Y. Lin, S. A. Brandt, D. D. E. Long, and E. L. Miller, "Power conservation strategies for mems-based storage devices," in *Proceedings of the Tenth IEEE/ACM*

International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS 2002), pp. 53–62, IEEE Computer Society, 2002.

[4] L. R. Carley, J. A. Bain, G. K. Fedder, D. W. Greve, D. F. Guillou, M. S. C. Lu, T. Mukherjee, S. Santhanam, L. Abelmann, and S. Min, "Single-chip computers with microelectromechanical systems-based magnetic memory (invited)," *Journal of Applied Physics*, vol. 87, no. 9 III, pp. 6680–6685, 2000.

[5] L. Abelmann, T. Bolhuis, A. M. Hoexum, G. J. M. Krijnen, and J. C. Lodder, "Large capacity probe recording using storage robots," *IEE Proceedings: Science, Measurement and Technology*, vol. 150, no. 5, pp. 218–221, 2003.

[6] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 8, pp. 299–316, June 2000.

[7] Y.-H. Lu, E.-Y. Chung, T. Šimunić, L. Benini, and G. De Micheli, "Quantitative comparison of power management algorithms," in *Proceedings of the conference on Design, automation and test in Europe (DATE)*, pp. 20–26, ACM Press, 2000.

[8] S. W. Schlosser, J. L. Griffin, D. F. Nagle, and G. R. Ganger, "Designing computer systems with MEMS-based storage," in *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 1–12, 2000.

[9] B. Hong, F. Wang, S. A. Brandt, D. D. E. Long, and S. J. Thomas J. E. Schwarz, "Using mems-based storage in computer systems—mems storage architectures," *Transactions on Storage*, vol. 2, no. 1, pp. 1–21, 2006.

[10] M. Uysal, A. Merchant, and G. A. Alvarez, "Using mems-based storage in disk arrays," in *FAST '03: Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, (Berkeley, CA, USA), pp. 89–101, 2003.

[11] J. L. Griffin, S. W. Schlosser, G. R. Ganger, and D. F. Nagle, "Modeling and performance of MEMS-based storage devices," in *Proceedings of ACM SIGMETRICS 2000*, (Santa Clara, California, 17-21 June), pp. 56–65, 2000.

[12] B. Jacob, S. W. Ng, and D. T. Wang, *Memory Systems (Cache, DRAM, Disk)*, ch. 17, pp. 631–633. Morgan Kaufmann, 2008.

[13] H. S. Bucy, G. R. Ganger, and Contributors, "The DiskSim simulation environment version 3.0," reference manual, School of Computer Science, Carnegie Mellon University, January 2003.

[14] M. G. Khatib, J. B. Engelen, and P. H. Hartel, "Shutdown Policies for MEMS-Based Storage Devices – Analytical Models," Tech. Rep. TR-CTIT-08-03, Jan. 2008.

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

[16] B. Hong and S. A. Brandt, "An analytical solution to a MEMS seek time model," Tech. Rep. UCSC-CRL-02-31, Storage Systems Research Center, University of California, Santa Cruz, Sept. 2002.

[17] "Kernel Trace Systems." `http://elinux.org/Kernel_Trace_Systems`. Accessed in November 2007.

[18] "Mobile Traces." `http://sourceforge.net/projects/ipaq-cf-traces`.