Power consumption on MEMS-based storage device

Ying Lin
linying@cse.ucsc.edu

Abstract

For mobile applications, power dissipation is crucial. Compared with disk-based storage, MEMS-based storage provides a more low-power and robust solution for portable applications. This project presented several methods to reduce the total energy in MEMS-based storage device and analyzed the trade off between I/O performance and power dissipation. Based on our experiments on a real workload (HP snake trace), aggressively spin-down method can reduce total energy by 50%, merging sequential requests method can save servicing energy by 18%, and sub sector access method is estimated to reduce servicing power consumption by about 30%.

1. Introduction

Besides of a better I/O performance by an order of magnitude, MEMS-based storage devices have many significant advantages over disk drives, such as small physical size, good portability, low power, and potential to integrate processing within the same substrate [1]. For portable applications such as notebook PCs, PDAs, video camcorders and biomedical monitoring, they need a more robust and low power consuming storage device, because many of these applications involve rapid device rotation (e.g., rapidly turning a PDA) and are prone to inducing shock (e.g. rapidly turning a PDA.). MEMS device has a high reliability by employing multiple sleds or RAID scheme on single sled, increasing actuator force, decreasing sled mass and increasing spring force to increase the shock tolerance [1]. But the most important advantage comes from its low power physic characteristics:

(1) Sled has much less mass than disk platter and takes far less power to keep in motion.

(2) The electronics of MEMS devices lower power requirements for operating the read/write tips.

(3) MEMS device has a faster transition between active and standby mode. (About 0.5ms [2]) So it can efficiently employ a standby mode—stopping sled movement during periods of inactivity, to save power.

Because MEMS-based storage device is a totally brand-new device, first of all, we had to make clear where the power goes when the device is working. In the following subsections, we summarized the whole power consumption distribution in different states of MEMS based storage device.

1.1 Power consumption model in MEMS-based storage device

From [1][2], we have collected some preliminary data about MEMS device power consumption: each active probe tip and its signal processing electronics consume *1mW;*

keeping Sled in motion is *0.1W*. Therefore, in active mode (data access), 1,000 simultaneously active tips, for example, would consume *1.1W* totally. Also, for standby mode (stopping sled movement) power consumption is estimated to be *0.05W*. Because the time switching between active and standby mode is only *0.5ms*, we can easily switch between active mode and standby mode.

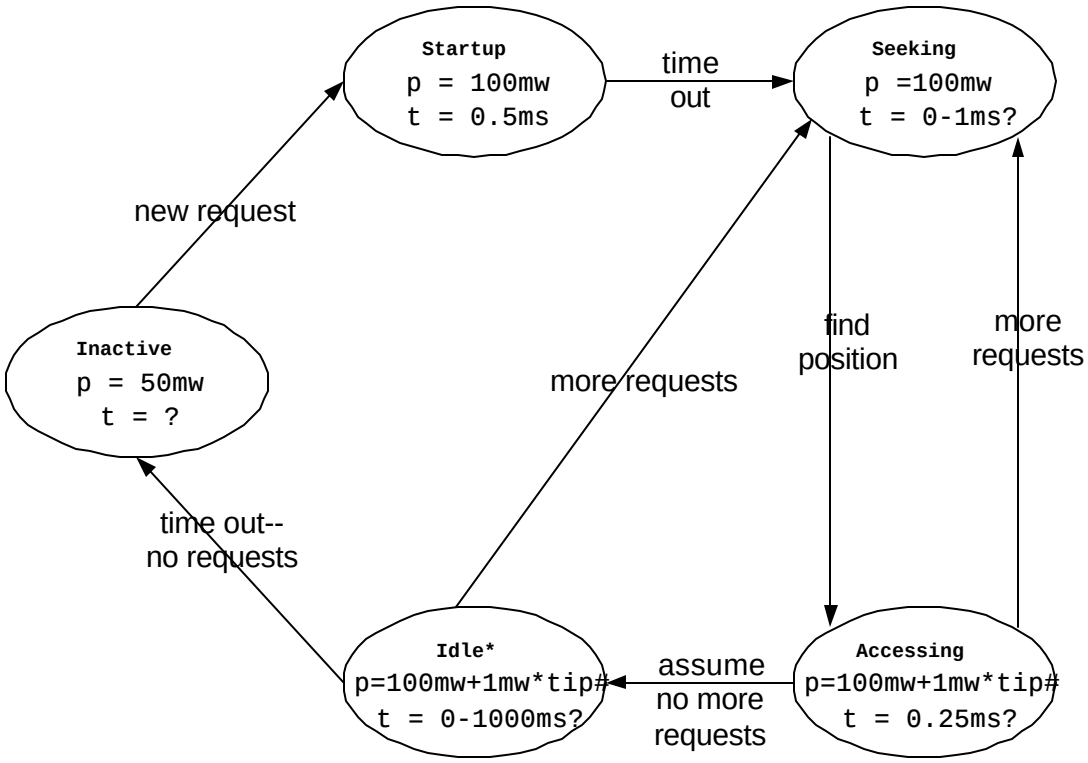

Figure 1. Power state switching process in MEMS-based storage device

Figure 1 demonstrates how the five power states—Inactive, Startup, Seeking, Accessing and Idle, coordinate to work in MEMS-based storage device. Their exact descriptions are as follows:

(1) "Seeking": Sled is driven by the actuators, such that media access can be performed. In this mode, only active sled contribute to power consumption.

(2) "Accessing": Sled is moving and necessary tips are activated to access servo information and data. Thus, active sled and active tips both consumed power.

(3) "Inactive" or "Standby": Sled is not moving, with perhaps a small power drain to hold it in place. In this state, power only consumed by inactive sled.

(4)"Startup" means the switch mode from inactive to active mode. Because the switch time is only 0.5ms, we get the following function: (Energy for tips on and off omits.)
Switch power consumption = sled_active_power_mw * startup_time_ms / 1000000.0
= 100 *0.5 / 1000000.0 =0.000005 J

(5) "Idle" means the waiting mode for MEMS to make sure that there are no active requests in the queue and decide to switch from active to inactive mode. There is a before_inactive_delay (idle time out) parameter in the parameter file, which defines how long the MEMS wait to confirm previous request is the last request in the queue. In the idle mode sled keeps active (repeat doing seek loop) and tips are active (repeat accessing servo information). So if it takes a long time in idle mode, the power will be greatly consumed by the active tips and sled.

1.2 Power distribution in MEMS-based storage device

To decide which state in the whole picture dominates the total power consumption, we set up an experiment to calculate specific energy consumption in the four modes separately:
(1) Servicing energy in active mode—seeking, accessing servo information and accessing data.
(2) Inactive energy for inactive mode.
(3) Startup energy for switch from inactive mode to active mode.
(4) Idle energy for switch from active mode to inactive mode – seeking and accessing servo information until exceed idle time out.

We used two workloads to test the power distribution of CMU G2 MEMS-based device (Idle time out is set as 1s). Trace.seagate is an internally generated synthetic workload (traces of validating I/O activity on Seagate ST41601N, within disksim/valid directory). Snake.920530.disk6.srt is a HP general-purpose web server real workload.

From the below figure 2, we can find that 90% energy is consumed in idle mode. In this mode, even though there is no active request in the queue, the sled keeps moving and active tips keep accessing servo information until the time exceeds the idle time out (1000ms). After waiting for the delay, if there are still no new active requests, MEMS can make sure that previous request is the last one and switch to inactive mode. (This mode only consumes sled inactive power—50mw.)

Therefore, it is feasible for us to take effort in predicting when the last request comes and spin down sled directly from active to inactive mode. Thus, we can increase the spin down time and reduce the power consumption by eliminating or reducing the time of idle mode (waiting cost for switch from spin up to spin down).

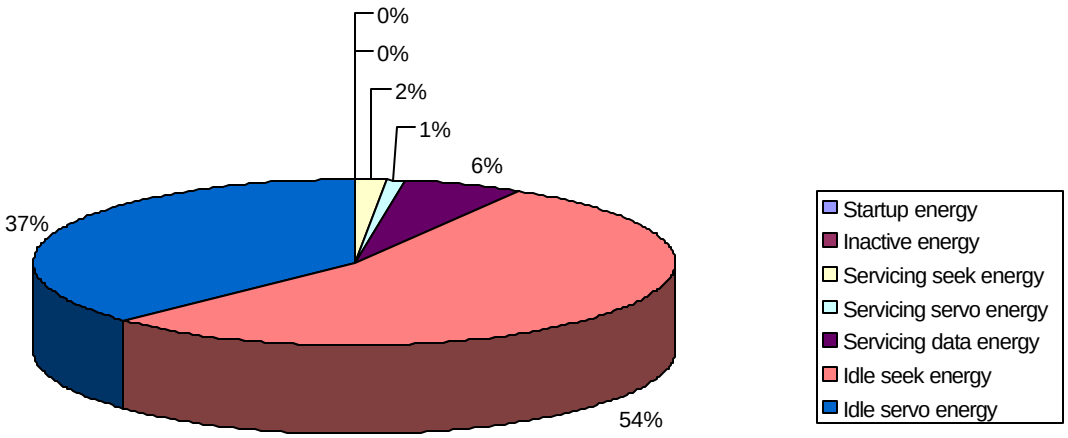

Figure 2. Power distribution in seagate trace

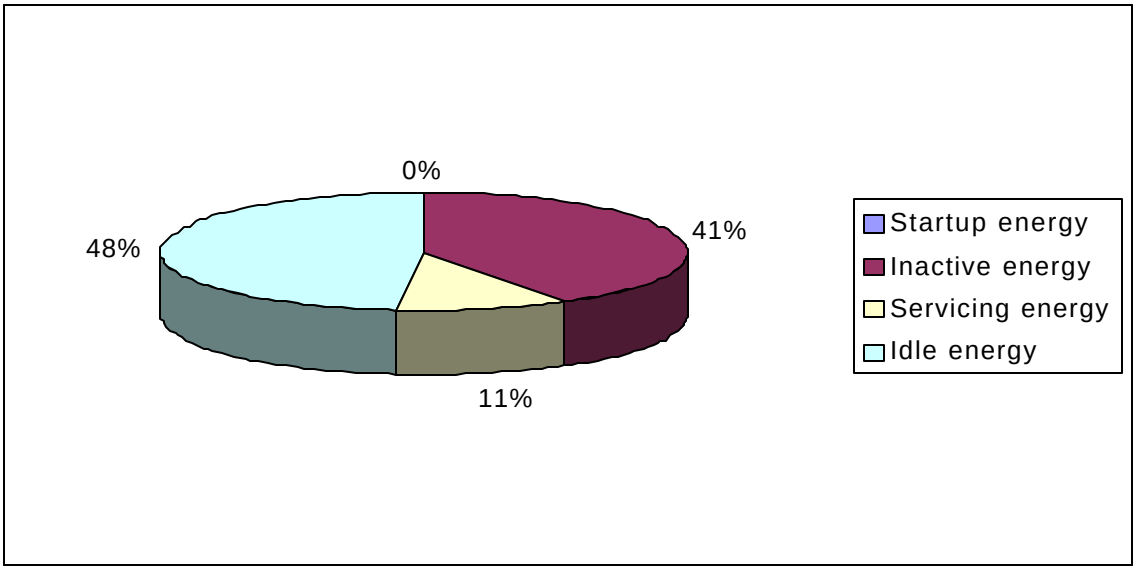

Figure 3. Power consumption in snake workload

From Figure 3, we've got the similar conclusion as that of Figure 2. Idle energy occupies almost 50% of total energy. Because the Inactive state is the lowest power state, it should last as long as possible. And servicing energy is the second part after idle energy that we should take effort on.

In Section 2, related work is mentioned. Section 3, 4 and 5 focus on how to reduce idle energy, increase inactive state time and reduce servicing energy to reduce the total power consumption in MEMS-based storage device.

2. Related work

Some related work on MEMS-based storage device has been done in CMU and UCSC. But the previous result about power consumption in MEMS device is very limited.

- Some useful assumptions in MEMS power consumption are found in [1][2].
- Using the physical parameters in [2], we can compute the power consumption in different energy state.
- Available simulator:
  We have the DiskSim simulator, which integrated the MEMS CMU model. We modified related modules to do our power consumption simulations.

3. Reduce Idle energy—Aggressively spinning down sled

From section 1.2, we've known idle energy is the dominant factor (50%-90%) in whole power distribution. The aggressive spinning down sled method is effectively reduce idle energy, thus save the total power consumption. But it causes I/O performance suffer a little for the low power solution.

3.1 Effect of different Idle time out on power consumption

We tuned the before_inactive_delay parameter to get different idle time-out and run simulations to see how the power consumption changes. The simulation results are demonstrated in the below Figure 4 about effect of idle time out on power consumption.

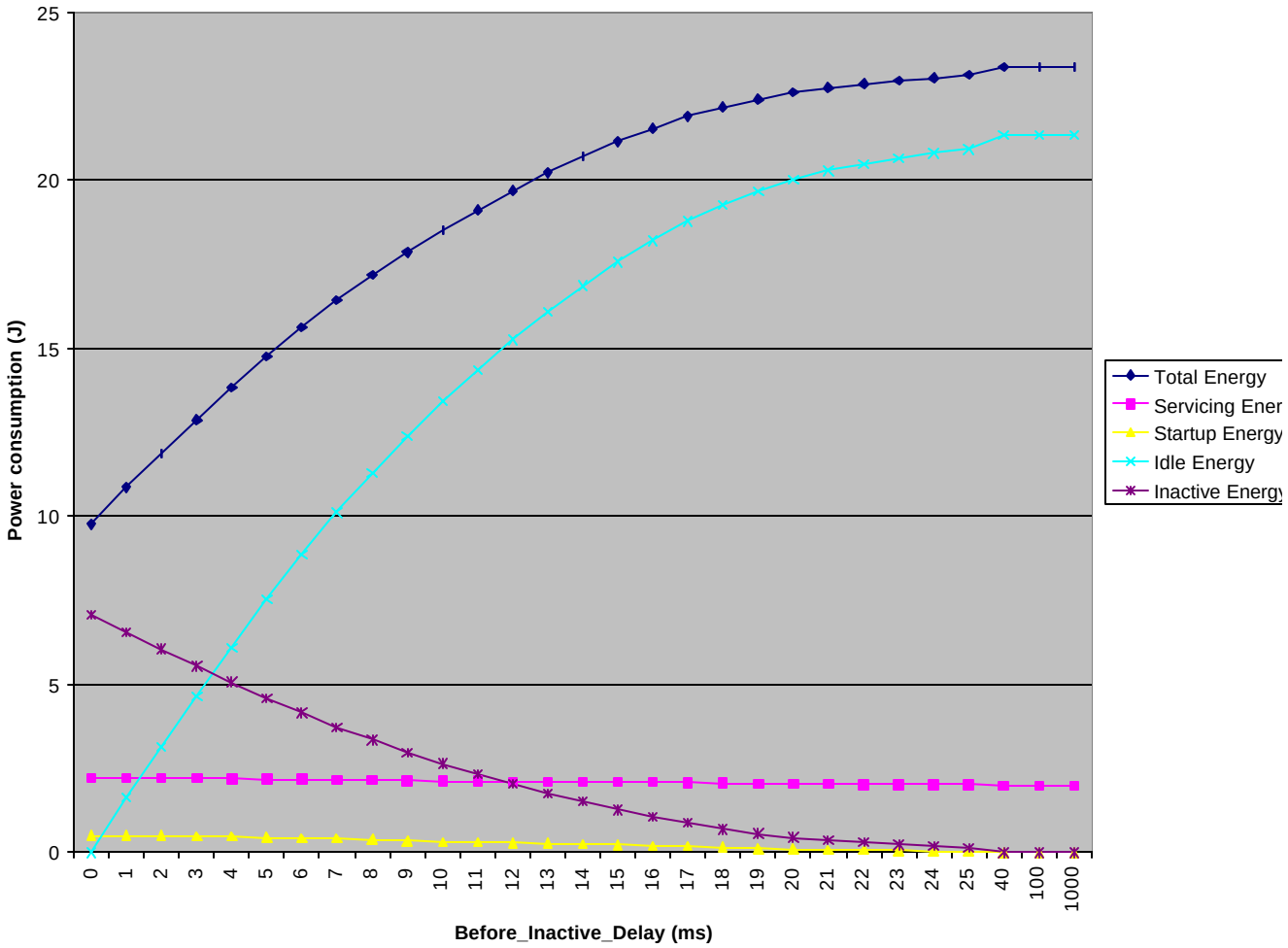

Figure 4. Effect of idle time out on power consumption

From the above figure, we can found when the Before_Inactive_Delay increase from 0ms to 40ms, the total energy expended more than 2 times. With the longer idle time out, MEMS device has to wait for longer time to switch from the active state to inactive state; thus, it spends more time on idle state and less time on inactive state. Because more power is consumed in idle state than inactive state, with the greatly increase of Idle energy and slightly decrease in inactive energy, the total power consumption is enhanced with the growth of Before_Inactive_Delay.

Note: When the Before_Inactive_Delay is greater than about 40ms, it hardly changes the power consumption, because of the workload I chose. For trace.seagate, there are always new requests coming within the delay greater than 40-50ms. So, the delay value makes no difference for power state switch timing when it is over 40-50ms.

Reducing the delay threshold is definitely a good choice. If it is feasible for us to take effort in predicting when the last request comes and spins down sled directly from active to Inactive mode. Thus, we can increase the inactive time and reduce the power

consumption by eliminating or reducing the time of idle state. The next sub section gives us the answer.

3.2 Unfeasibility of adaptive idle time out prediction on MEMS-based storage device

As for disk drive, it is well known that spinning the disk down when it is not in use can save energy. But since spinning disk back up consumes a significant amount of energy, spinning the disk down immediately after each access is likely to use more energy than is saved. So it needs an intelligent strategy for deciding when to spin down the disk. But this motivation is not applicable for MEMS device, from our former analysis, we've known that there only 0.5ms*100mw energy is used for MEMS to startup. The dilemma for deciding keeping spinning or spin down won't exist. We can aggressively spin down the sled after each access. From the Figure 4, when fixed time_out is zero, we can get the minimal power consumption. And with the time_out increases, the total energy increases as well. The result is totally different for disk drive. From the Figure 3 in Reference [3], we found that both the biggest and smallest fixed time_outs consume more energy than any time_out between them. Therefore, even if we add in the adaptive algorithm to set up the MEMS device time_out based on learning sequence of idle trails, and we can get a flat line on figure about energy use of fixed time-outs, but the energy definitely cannot be saved more than we can get if we use the approximate zero fixed time_out.

Therefore, aggressive spinning down the sled is the best decision from the power consumption point of view. But whether it brings negative effects on I/O performance, we analyzed the trade off between I/O performance and power consumption in the next sub section.

3.3 Effect of aggressive spin-down on I/O performance

We used two metrics to evaluate the I/O performance—average throughput and average response time. The following results still come from the simulation on snake workload.

From Figure 5, we found with the Before_Inactive_Delay decreases from 40ms to 0ms, the average response time increases around 66%. In other words, if we want to use smaller Before_Inactive_Delay to get lower power consumption, we have to suffer from the worse response time.

Note: For the same reason in previous power analysis, the delay value makes no difference for power state switch timing when it is over 40-50ms. So, there is no extra time consumed in extra state switch. After about 40ms, the average response keeps the same with the growth of Before_Inactive_Delay.

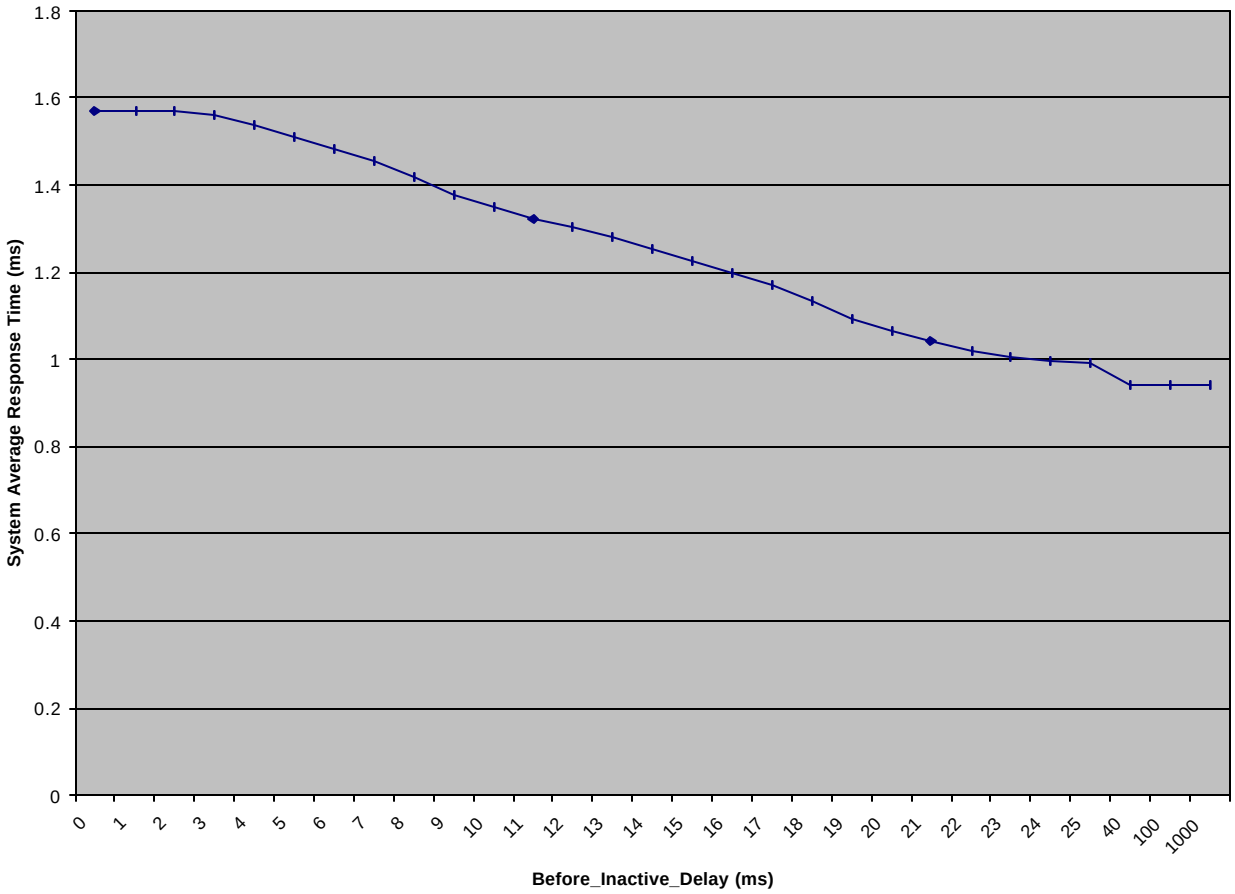

Figure 5. Effect of idle time out on average response time

From the below Figure 6, we can find that the throughput doesn't have a great change with the growth of size of fixed time_out. When time_out decreases from 10000ms to 0ms, the throughput only decreases over 4%. This means if we use zero fixed time_out, throughput won't suffer a lot.

Reconsider Figure 5, the biggest difference in response time is about 0.5 ms, which is the estimated value for startup time out. Thus, when aggressive spinning down sled, almost each request has a startup time out overhead. In the next sub section, we concerned about the effect of startup time on I/O performance.

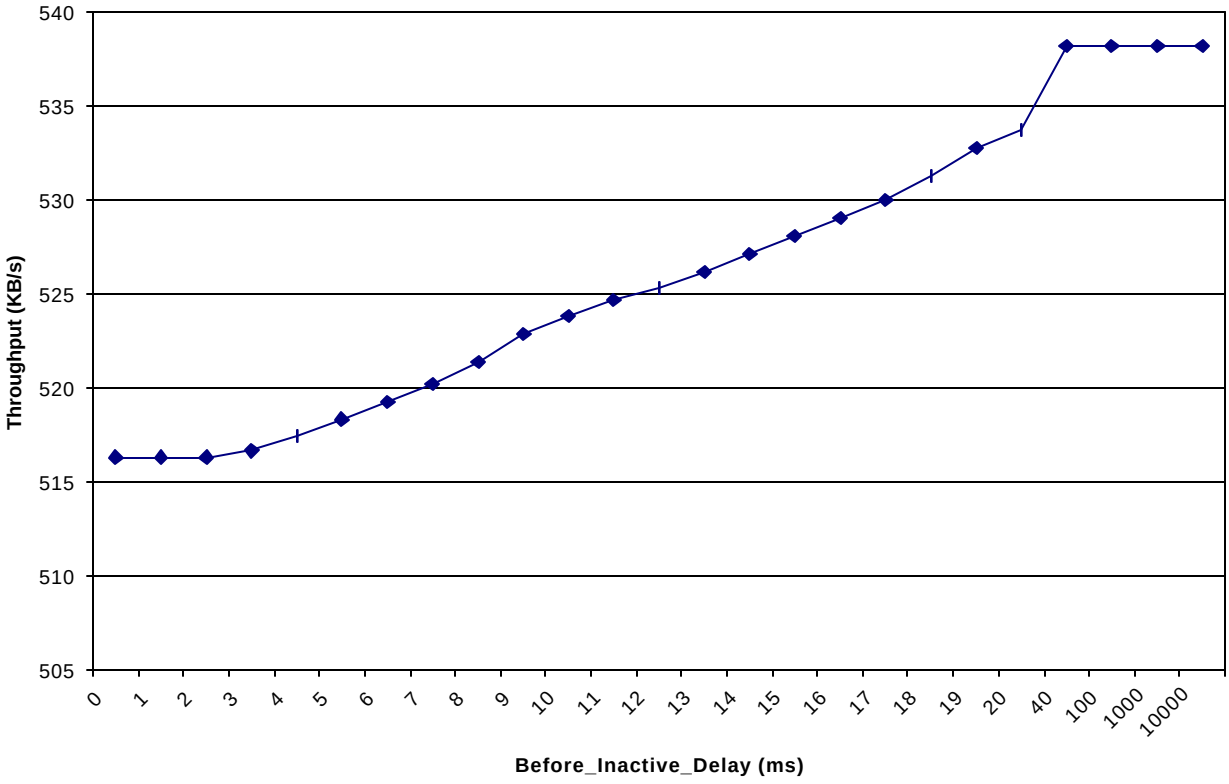

Figure 6. Effect of idle time out on average throughput

3.4 Sensitivity of startup time out

With no rotating parts and little mass, the media sled's startup time is very small (estimated at under 0.5ms in CMU model [1]). And based on our previous experiments on synthetic workload, the average seeking time in MEMS is about 1ms. So, startup time is comparable to seeking time and it should be sensitive to I/O performance. We used DiskSim's validation workload—trace.seagate (10,000 requests, 50% reads, 30% sequential, requests is exponential distributed with a mean size of 8KB) to testify the sensitivity of Startup time.

The following figures show that startup time cause the average response time and throughput change linearly. Because we use the immediately spin down method (idle delay is 0) on the sparse workload, an extra startup overhead is added for almost each request.

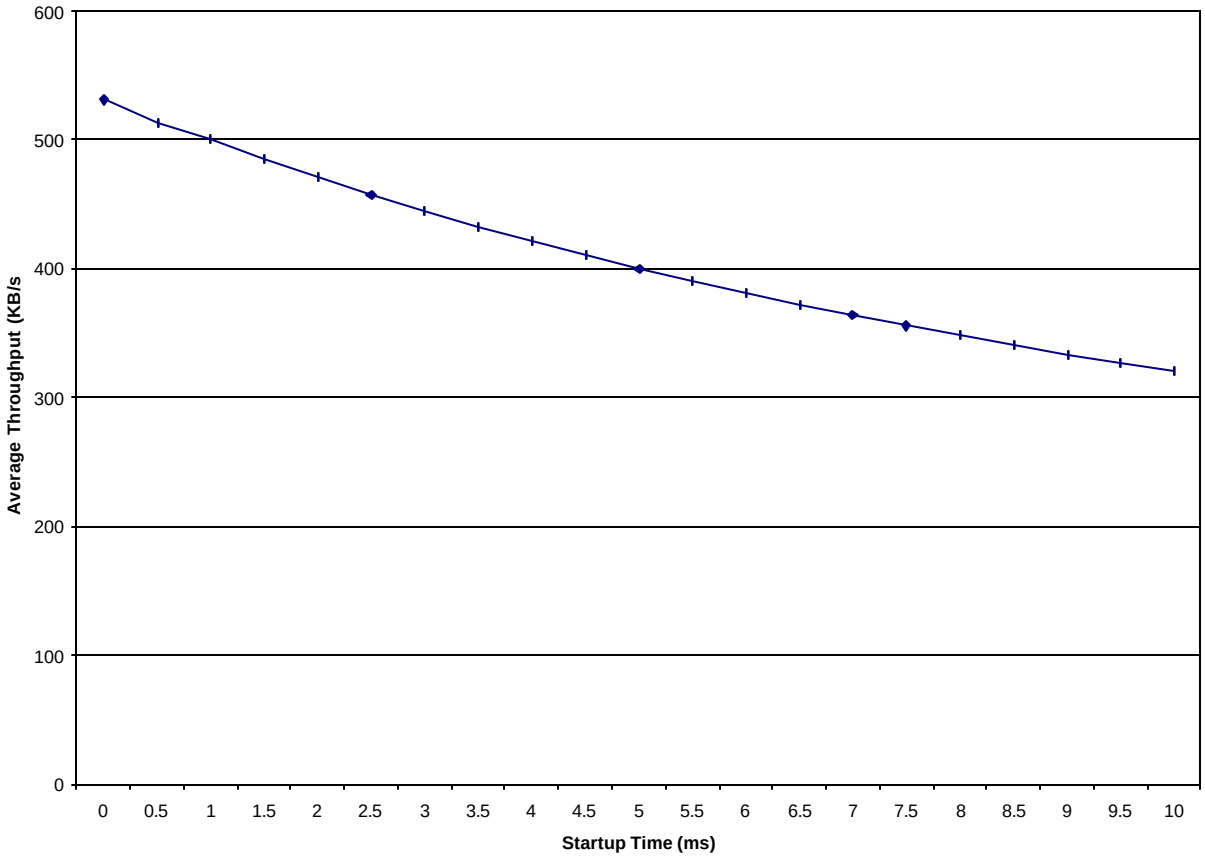

Figure 7. Effect of startup time on average throughput

Startup time out is a physical parameter for MEMS-based storage device. Its great sensitivity to throughput and response time cause the lost of I/O performance on aggressive spinning down sled method.

In general, from previous analysis results, we've got that because of rapid transition between power-save and active modes, MEMS sled can greatly save power by aggressive spinning sled down as soon as the I/O queue is empty, obviating the need for complex idle delay prediction. The power wasted in idle state will reduce to zero with a little power overhead in spinning sled back up; and the overall power consumption can be saved by more than 50%, compared with the energy consumed when idle delay is 1s. As for I/O performance, the average throughput may not be suffered by the immediate spinning down method, while response time may increase 0.5ms for undesirable startups.

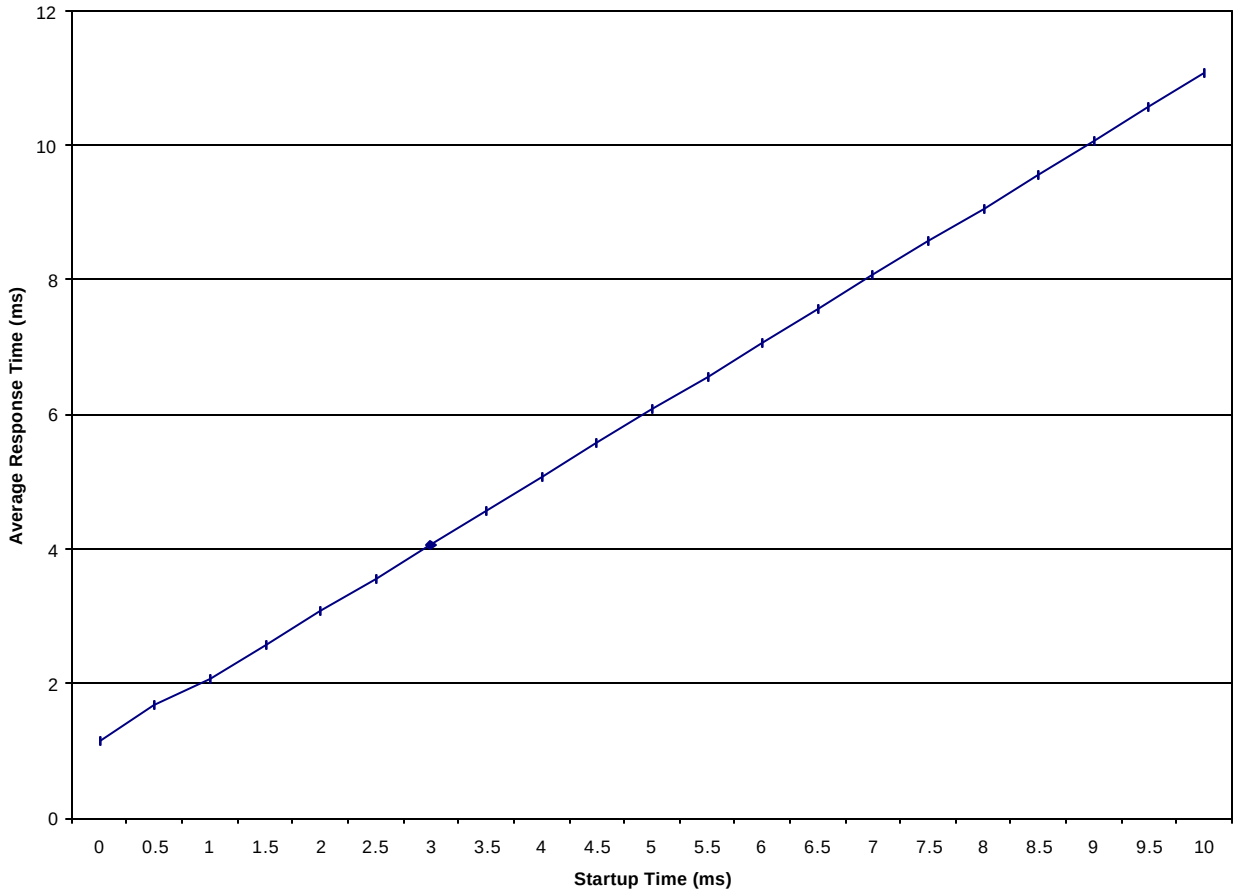

Figure 8. Effect of startup time on average response time

4. Reduce Service energy—Merging sequential requests

Except the idle energy (now it can be zero), servicing energy ranks the second most part in the whole power distribution. We used "merging requests" method to reduce servicing energy and evaluated their effect on I/O performance.

4.1 Simple implementation

From the Figure 4 of [4], we can see the mapping of logical block numbers to physical location is optimized for sequential accesses. Sequential logical blocks are mapping to 64 same-position sectors from adjacent tip sets. So, logically sequential blocks can be accessed simultaneously in MEMS. Because the most of I/O requests are sequential and very busty, we can merge sequential I/O requests in the queue and simultaneously activate more tips to service the combined larger request with shorter time. This method increase the parallelism of concurrent active tips to increase I/O bandwidth and reduce response time. And it may not cause the overhead in power consumption by activating more tips simultaneously. For MEMS, when accessing data, not only the active tips consume power, the sled also uses power to keep moving. If the

accessing time decreases by merging sequential requests, the power will be saved in the sled part. (For the tips part, power consumption keeps the same.1000 tips activated for 0.5 ms use the same power as 500 tips activated for 1ms.)

We used a simplest implementation of this method to evaluate the performance-- For adjacent two requests in the I/O queue, if they are both read requests and their logical block number are sequential, they are combined to one big request and replace the original separate two requests. (Of course, there are some constraints for the merging. For example, if the combined request needs to use more tips than the simultaneously active tip number limit defined by the device physical parameter, they cannot be combined.)

4.2 Experiment setup

We used snake workload on CMU G3 MEMS device simulation. G3 generation MEMS device can simultaneously activate 3200 tips. Average request size is 8K in snake workload, which means for each request, almost16*64 tips are activated at the same time. If two 8K requests are merging, 1024*2 tips are active to access data.

The results are demonstrated in the Figure 9 and Figure 10.

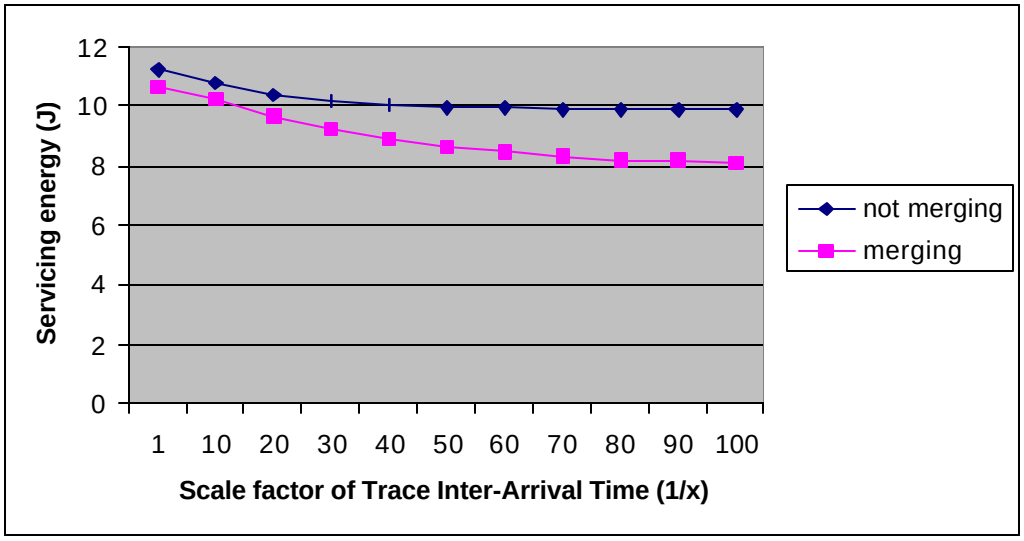

Figure 9. Effect of merging method on servicing energy

From Figure 9, we can see the simple merging method can save about 18% servicing energy.

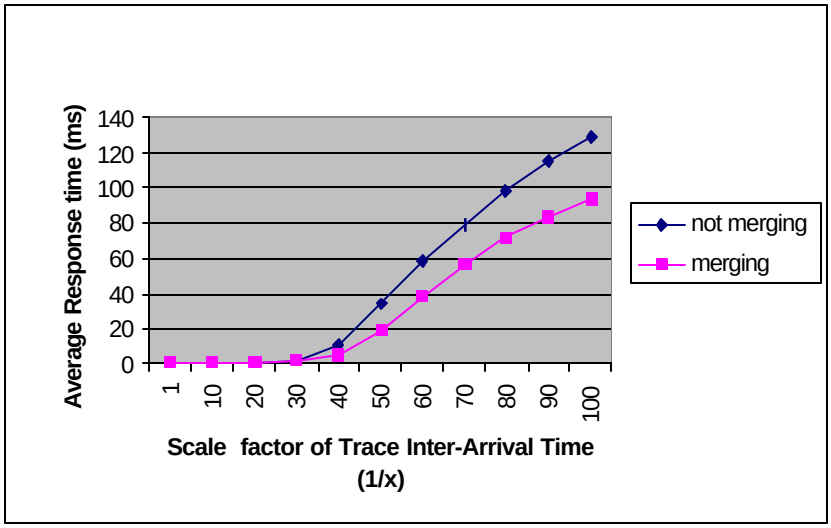

Figure 10. Effect of merging method on average response time

Figure 10 denotes the merging sequential requests method can reduce average response time by about 20%.

There is a lot of improvement space for the smart merging requests method. For example, we can combine more than two adjacent requests in the queue. Also, the logical sequential requests don't have to be adjacent in the queue. Both read and write requests can also be combined. Further, requests can also be separated and recombined to enhance parallelism.

5. Reduce Service energy—Accessing sub sector

MEMS-based storage device has the ability to adjust its power consumption during data accesses by reading or writing at a smaller granularity than standard 512 byte blocks. Since active tips dissipate more power than moving sled, reading/writing only necessary data could save power. The device only needs to activate as many tips as necessary to satisfy a request, which should result in power savings.

5.1 Low level data layout on CMU MEMS model

The smallest accessible unit of data in MEMS is a "tip sector", consisting of servo information (10 bits) and encoded data/ECC (80bits = 8 bytes encoded data bytes). Groups (or tip sets) of 64 tip sectors from same position of separate regions are be combined into 512-byte "logical sectors", analogous to logical blocks in SCSI. The logical-to-physical mapping is demonstrated in Figure 4 of [4]. During a request, only those logical blocks needed to satisfy the request are accessed and the corresponding tip sets are activated; unused tips remain inactive to conserve power.

5.2 Assumption of sub sector access in MEMS-base storage device

Accessing sub sectors in MEMS is feasible, because the error correcting codes can be computed over data striped across multiple tips. In CMU model, each 512-byte logical block and its ECC are stripped across 64 tips. Accessing sub sectors cannot affect internal faults recovery in MEMS.

5.3 Experiment setup

DiskSim simulator only supports the trace formats for file system level workload. Snake workload is a file system level trace. (Arrival time, start LBN and LB numbers is provided for each request.) As for one request, only its last logic block is possible to be accessed as a sub sector, if the corresponding operation system level workload is provided. So I used the snake workload to simulate the sub sector accessing. For the last block within each individual request, the sub sector size is uniformly distributed. I randomly select the active tip number from 1 to 64 for the last sub sector and keep the other tips within that tip set inactive.

We used Snake trace and CMU G3 device parameters to simulate accessing sub sector method. The results of its effect on power consumption and I/O performance are compared with that of original simulator are as follows:

Scale Factor to Trace Inter-Arrival Time = 0.1:

	Original	Subsector
Total energy (J)	224.232490	223.985658
Servicing energy (J)	10.817354	10.570522
Startup energy (J)	1.882250	1.882250
Inactive energy (J)	211.532886	211.532886
Response time average (ms)	1.524451	1.524451

Scale Factor to Trace Inter-Arrival Time = 0.01:

	Original	Subsector
Total energy (J)	29.991416	29.744633
Servicing energy (J)	9.880290	9.633507
Startup energy (J)	0.247550	0.247550
Inactive energy (J)	19.863575	19.863575
Response time average (ms)	129.842917	129.842917

From the above results, we can see that the sub sector method can save servicing energy about 3% without affecting average response time. The improvement is not obvious for the following reasons:

- The snake trace is a file system level trace and its average request size is 8KB, much larger than 512B
- The method I used only assumed the last logical block within one request is accessed as a sub sector. If we assume each request in the workload can be accessed as a sub sector, the servicing energy is estimated to save by around 30%.

If we can use real operation system level trace, the results will be more convincible. And for parallel file system trace, the average request size is about 200B (< 512B) other than 8KB in Unix file system, the sub sector method will have a significant improvement in power saving. We will continue to study this method in the future.

6. Conclusion

For MEMS-based storage device, idle energy is the dominant part in whole power distribution. Using immediate spin-down method, the total energy can be saved by about 50% and the overhead is the increase of average response time caused by undesirable startups. Merging sequential requests method can save 18% servicing energy and benefit the power consumption and I/O performance at the same time. Accessing sub sector method is where we can gain more in the future, because when accessing data, more power drains by thousands of tips than by single sled. By the above methods, the low-power MEMS-based storage device will approach the optimal energy efficient goal and has significant applications in mobile computers.

Reference:

[1] Steven W. Schlosser, John Linwood Griffin, David F. Nagle, Gregory R. Ganger, "Designing Computer Systems with MEMS-based Storage", 9th International Conference on Architectural Support for Programming Languages and Operation Systems, 2000.

[2] L.R. Carley, J. A. Bain, G. K. Fedder, D. W. Greve, D. F. Guillou, M. S. C. Lu, etc, "Single Chip Computers with Microelectromechanical Systems-based Magnetic Memory", Journal of Applied Physics, 87(9): 6680-6685, 1 May 2000.

[3] "Adaptive Disk Spin-Down for Mobile Computers", David P. Helmbold, Darrell D.E. Long, etc.1998

[4] J.Griffin, S. Scholosser, G.Ganger and D. Nagle,"Operating system management of MEMS-based storage devices", 2000