Exploring the Usage of MEMS-based Storage as Metadata Storage and Disk Cache in Storage Hierarchy

Bo Hong

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

1 Introduction

The huge disparity between memory access times and disk access times has been the subject of extensive research. CPU speed has been increasing rapidly but disk access latency has lagged behind—disk transfer rates have been increasing at 40% per year, while seek times and rotational latency have been increasing at less than 10% per year [13]. This disparity has created a performance bottleneck in computer systems. The performance of storage systems could be even worst under workloads with small, non-sequential I/Os. Studies [26] showed that the majority of disk accesses (57%) are writes, the majority (67–78%) of writes are to metadata, and 10–18% of all write requests are overwrites of the last block written out. Moreover, metadata traffics are quite bursty because operating systems typically flush metadata to disk periodically to make the file system robust against system failures.

Several techniques have been proposed to reduce the cost of small writes to improve system performance [12, 15, 25, 26, 30]. A write cache can substantially reduce write traffics to disks and the perceived delay for writes [12, 26, 30]; for reliability, these caches typically use expensive NVRAM, whose cost is about three orders of magnitude higher per byte than disks. Ruemmler and Wilkes [26] showed that even a small amount of NVRAM can greatly reduce meta-data write traffics to disks (90%). However, the high cost of NVRAM limits its utilization as a write buffer. Even in favorable cases, NVRAM caches in high-end disk arrays can only hold 5% of the working set of applications, leading to low hit ratios [36]. Moreover, battery-backed NVRAM is much less reliable than disks, whose mean time to failure is only about 15K hours [27].

A new class of secondary storage devices based on microelectromechanical systems (MEMS) [4, 21, 34] currently being developed promise seek times 10–20 times faster than hard drives, storage densities 10 times greater, and power consumption an order of magnitude lower. MEMS devices provide non-volatile storage using either physical [34] or magnetic [4] recording techniques to achieve extremely high density storage. In order to achieve these high densities, MEMS-based storage designs use a non-rotating storage device with storage media on one surface and a large array of read/write heads on another surface directly above the storage media. By moving the surfaces relative to each other using MEMS actuators, each read/write head can access a region of the surface. The per-byte cost of MEMS-based storage devices is expected to be less than one order of magnitude higher than disks and about two orders of magnitude less than NVRAM.

Access latencies in MEMS-based storage devices are much lower than traditional secondary storage devices because there is no rotational latency and physical movements to locate data are small. Though the bit rate of each read/write head is slow compared to the single read/write head in a disk, overall throughput is increased by having many read/write heads active at once. Initial density estimates are in the range

of 10 gigabits per square inch with densities anticipated to reach 300 gigabits per square inch in a few years [11, 28]. Because it is very small, the capacity of a MEMS media sled is about 1–10 GB but a MEMS-based storage device can achieve much higher capacity by using multiple media sleds. MEMS-based storage devices are expected to have many other significant advantages over hard disks, including better I/O performance, smaller physical size, lower heat dissipation requirements, and integrated processing and storage [28]. For all of these reasons, MEMS-based storage devices are an appealing next-generation storage technology.

Although the best way of incorporating MEMS-based storage devices in current storage hierarchy is to directly replace hard disks, the higher cost and lower capacity of these devices might prevent users from throwing disks away. One question that arises is how to utilize the unique performance characteristics of MEMS-based devices to bridge the gaps between main memory and disk and improve the overall system performance.

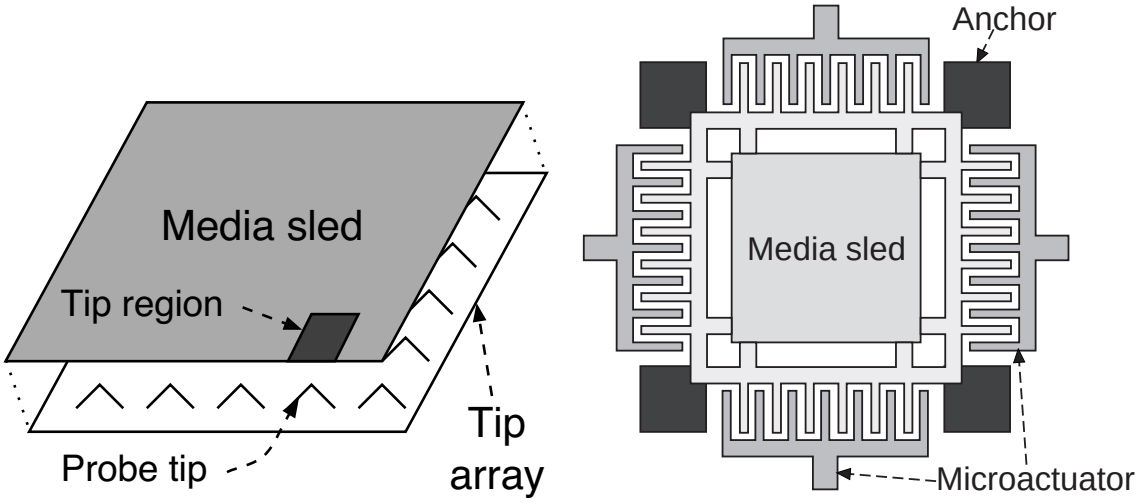
As discussed in [26], a substantial I/O traffic to disk is metadata and metadata traffic is quite bursty. The storage used by metadata in a file system is very small (1–2%) [2]; therefore, metadata only takes several gigabytes in a 100 GB file system. Furthermore, most of metadata requests are small because file systems typically need to only update several fields in the metadata structure. Considering the aforementioned reasons, we proposed to use MEMS-based devices as a dedicated metadata storage in file systems. File systems store, retrieve, and update metadata directly on the MEMS-based storage device. A MEMS-based storage device can provide much better response times on metadata traffics than a conventional disk and its capacity is large enough to hold all metadata of a quite large file system. In our proposed method, metadata are also stored in the disk in a traditional way. The file system flushes "dirty" metadata on the MEMS-based device to disk periodically, making the system more robust against the crash of disk or MEMS device and the disk still demountable. Because the dedicated MEMS storage offloads bursty and less sequential metadata traffics from the disk, the disk can better service other requests, improving the system performance. Our results showed that we can improve the system performance by 28–46% on a user workload, depending on how much the metadata traffic counts for the whole workload.

Wang [35] also proposed to use a MEMS-based storage device as a large non-volatile disk write buffer, which is totally transparent to file systems. Write requests that arrive at the disk driver is patched into logs and flushed to the MEMS write buffer as soon as the MEMS device is idle. The results showed that the average response time can be improved by a factor of 5 to 15 using a 128 MB MEMS write buffer. This technique can be feasibly adopted in our work. Combining this technique with our proposed dedicated MEMS metadata storage, we can improve the system performance by a factor of 3.3 to 8.2. Moreover, due to the high determinism of seek times of MEMS-based storage devices, we can also promise better consistency on system performance than a disk system by a factor of 2.4 to 5.7.

2 Background

It is important to note that because MEMS-based storage devices are still in their infancy, many of the details are still uncertain. There are several proposed architectures [10, 21, 28, 34], and we have based the physical parameters of our experimental model on the specification from Carnegie Mellon University (CMU) [10, 28].

Figure 1 shows the details of a MEMS-based storage device. The device consists of a surface coated with magnetic media, called a *media sled*, and a two-dimensional array of stationary read/write probe tips, called a *tip array*, as shown in Figure 1(a). Figure 1(b) shows the media sled suspended above the tip substrate by silicon beams that act as springs, and moved in the x and y directions by forces generated by lateral resonant microactuators. A request is satisfied by first moving the sled to the correct position and then transferring data. To perform a seek, the microactuators move the media sled to a specific (x,y) position. Once the seek



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

Figure 1: Components of a MEMS-based storage device.

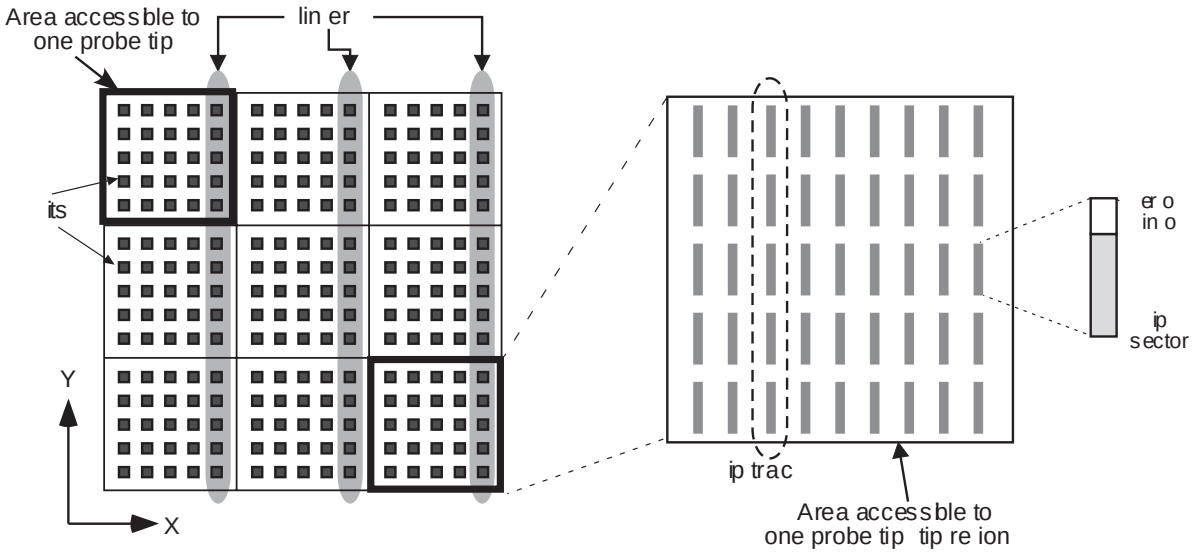


Figure 2: Data layout on a MEMS device.

is complete, data is accessed by the fixed probe tips while the media sled is moving at a constant velocity in the *y* direction. In contrast to disk, the tips remain nearly stationary during a seek, with the exception of minor *x* and *z* dimension adjustments [11], while the sled keeps moving in the *y* direction.

In contrast to disk, a MEMS-based storage device can use multiple tips simultaneously to access data, achieving a high degree of parallelism. Because of power, heat, and wiring limitations, not all tips can be active at the same time. For instance, in the CMU G2 model with 6400 probe tips, only 1280 tips can be active at once, while in the G3 model, 3200 tips are expected to be active simultaneously [28].

The media sled is logically divided into rectangles called *tip regions*; each tip region is accessible by a single read/write probe tip. The smallest unit of accessible data is a *tip sector*, consisting of servo information (10 bits) and encoded data/ECC (8 bytes of data encoded as 80 bits). Each tip sector is represented as a triple $\langle x, y, tip \rangle$, where *x* and *y* are position coordinates and tip is a tip number. Groups (or tip sets) of 64 tip sectors from the same position of separate regions are combined into 512 byte *logical sectors*, analogous to

logical blocks in a hard drive. In keeping with disk terminology, the CMU MEMS research group applies a direct low level data layout, such as tracks, cylinders etc., to MEMS-based storage device.

Figure 2 illustrates the low level data layout of a MEMS-based storage device. The media sled is logically broken into *tip regions*, defined by the area that is accessible by a single head, approximately 2000 by 2000 bits in size. Each tip in the MEMS device can only read the data in its own tip region; this limits the maximum sled movement to the dimensions of a single tip region. The smallest unit of data in a MEMS-based storage device is called a *tip sector*. Each tip sector, identified by the tuple $\langle x, y, tip \rangle$, has its own servo information for positioning as well as its own error correction information. The set of bits accessible to a single tip with the same *x* coordinate is called a *tip track*, and the set of all bits (under all tips) with the same *x* coordinate is referred to as a *cylinder*. Also, the set of tip sectors that can be accessed by simultaneously active tips is known as a *logical sector*. For faster access, disk sectors can be striped across logical sectors.

3 Related Work

With the approaching physical limit of current magnetic recording technology [6] there are a number of researchers working to overcome this obstacle. Efforts to develop alternative storage technologies have resulted in many promising technologies, including holographic storage [23, 29] and attempts to investigate their integration into the storage hierarchy [24], AFM [8, 19], and probe-based magnetic recording technologies [3, 4, 5, 7, 22, 21, 20, 32, 34].

Probe-based orthogonal magnetic recording offers an effective means to overcome the limits imposed by stability issues in longitudinally recorded media. When coupled with research on MEMS devices [16], there are now promising approaches of applying this technology to provide storage devices capable of orthogonal recording on magnetic media using many micro-machine read-write heads. Microspring designs have been proposed and are claimed close to release by Nanochip [3, 21, 20], and a cantilever/sled design is the subject of intensive research by Carnegie Mellon's Center for Highly Integrated Information Processing and Storage Systems (CHI^2PS^2) [4, 5, 1]. Current projects on MEMS-based storage also include those by the *Millipede* [34] project at IBM, the Atomic Resolution Storage (ARS) project at Hewlett-Packard Laboratories [32], and Sandia National Laboratory. While these projects use different recording technologies, they are based on tip arrays and media sleds.

In many systems, the performance of disks is limited by positioning delays, particularly for workloads with small, non-sequential I/Os. Several techniques have been proposed to reduce the cost of small writes to improve system performance [12, 15, 25, 26, 30]. A write cache can substantially reduce write traffics to disks and the perceived delay for writes [12, 30]; for reliability, these caches typically use expensive NVRAM. Ruemmler and Wilkes [26] showed that even a small amount of NVRAM can greatly reduce meta-data write traffics to disks. However, the high cost of NVRAM limits its utilization as a write buffer. In DCD (Disk Caching Disk) [15], a small log disk is used to cache writes. LFS [25] uses memory buffer to collect small writes and writes them back to a disk segment in a long sequential write.

MEMS-based storage devices promise to provide much better response time and bandwidth compared to conventional disks. It is important to study how to integrate MEMS-based storage into the storage hierarchy. Griffin and Schlosser *et al.* [11, 28] showed that overall application runtime can be improved by a factor of 1.9–4.4 when using MEMS-based storage as a disk replacement and I/O response time can be improved by up to 3.5 times when using MEMS-based storage as a disk CACHE. Uysal *et al.* [33] evaluated potential gains in performance and cost by incorporating MEMS-based storage in disk arrays. Wang [35] studied the performance impact of replacing main memory (typically DRAM) with a MEMS-based device and using MEMS devices as disk write buffer. The results showed that using MEMS devices as replacement of main memory significantly degrades overall system performance and he concluded that a MEMS-based storage

device is basically a block device and not suitable for high-speed random access. However, using MEMS devices as disk write buffer can improve the average response time by a factor of 5 to 15.

Recently, there has been interest in modeling the behavior of MEMS-based storage devices. Because these devices do not yet exist, it is useful to build accurate and tractable models of them for understanding their performance characteristics. Griffin *et al.* [10] proposed a positioning model of MEMS-based storage devices, which takes into account the external force, the spring force, and the initial and final access velocities. Madhyastha and Yang [18] have also studied MEMS modeling, with an emphasis on creating more accurate models of MEMS devices. They developed a more realistic access time model that does not assume only an ideal acceleration and takes into account damping and restoring spring forces. Their work informed the analytical seek time analysis provided.

Because of the unique performance characteristics of MEMS-based storage devices, the efficiency of using existing disk scheduling algorithms on MEMS devices needs to be reexamined. Studies [11, 28] showed that for a limited class of MEMS devices, one-dimensional placement and scheduling can be applied efficiently. In that work, data was placed on the MEMS-based device in longitudinally sequential tracks, similar to tracks on a disk drive. However, this method is preferable only if a long longitudinal seek takes less time than a smaller latitudinal seek. They found that for MEMS devices Shortest Positioning Time First (SPTF) had the lowest average access time, but the variability of SPTF was very high, as would be expected for a greedy algorithm continually searching for the least expensive "next step". They also explored the use of standard disk-type algorithms such as Shortest Seek Time First (SSTF) and Circular Look (C-LOOK), with minor modifications, on MEMS devices. Their results show that such algorithms can be successfully adapted to MEMS storage. Hong *et al.* [14] further studied the the characteristics of positioning behaviors of MEMS-based storage devices and proposed a Zone-based Shortest Positioning Time First (ZSPTF) scheduling algorithms. ZSPTF divides the MEMS storage media into a set of *zones* based on seek time equivalence regions. Zones are serviced in a C-SCAN (Circular Scan) order and multiple requests within a zone are serviced in SPTF order. Simulations showed that ZSPTF has average response times comparable to SPTF but with response time variability comparable to C-SCAN.

MEMS-based storage devices promise to reduce power use by an order of magnitude over conventional disk. Lin *et al.* [17] examined the power model of a MEMS-based storage device and performed a quantitative analysis of the power distribution among different working modes. Based on that analysis, they presented three strategies to reduce power consumption: aggressive spin-down, merging of sequential requests, and subsector accesses. The results showed that by applying all three power management strategies simultaneously the total power consumption of MEMS-based storage devices can be reduced by about 54% with no impact on I/O performance.

4 MEMS-based Storage Architecture

The huge disparity between memory access times and disk access times has created a performance bottleneck in computer systems. The performance of storage systems could be even worst under workloads with small, non-sequential I/Os. Studies [26] showed that the majority of disk accesses (57%) are writes, the majority (67–78%) of writes are to metadata, and 10–18% of all write requests are overwrites of the last block written out. Moreover, metadata traffics are quite bursty because operating systems typically flush metadata to disk periodically to make the file system robust against system failures.

To alleviate the impact of the widening performance gap between main memory and disk, modern computer system typically employs large memory buffers. A significant percentage of raw disk traffic can be absorbed by these buffers. But most writes requests, especially small metadata write requests, must be sent to the non-volatile disk storage to guarantee system consistency and thus benefit little from the large memory buffer. Therefore the traffic between the disk and main memory is dominated by small writes, particularly

metadata traffics [26]. Therefore, an efficient way to reduce write traffics could significantly improve the overall system performance.

As a new emerging storage technology, MEMS-based storage device is an promising alternative for both dedicated metadata storage and disk cache. Compared to conventional disk, MEMS-based storage device has much better performance characteristics. It has 10–50 times faster positioning time and 10 times higher bandwidth than disk. Due to the limitation of its physical size, the capacity of a MEMS media sled is lower than disk but still reasonable, which is about 1–10 GB. Greater capacity can be achieved by using multiple media sleds in a MEMS device. Moreover, MEMS-based storage is non-volatile, ensuring the file system consistency and correctness. Therefore, we propose to use MEMS-based storage device to store metadata in file systems and further combine our proposed method with a technique proposed by Wang [35], which uses MEMS device as disk write buffer.

4.1 MEMS-based Metadata Storage

In general-purpose Unix work environment, a substantial amount of disk operations are to metadata [26]. Moreover, the traditional Unix file system and its variants, such as HP-UX, tend to flush metadata periodically to make file systems reliable and consistent. These operations on metadata create bursty workloads and become a performance bottleneck of the storage system.

Because MEMS-based storage device is much faster than hard disk and non-volatile, we propose to use MEMS-based storage device as dedicated metadata storage. File systems store, retrieve, and update metadata directly on the dedicated MEMS metadata storage device. This approach works at the disk driver level and needs support from file systems: a metadata operation identifier is added to each request that reaches the disk driver. The disk driver maintains a mapping table which identifies the actual location of a disk metadata request on the MEMS device. A metadata request is redirect to the MEMS device when the disk driver detects it.

A MEMS-based storage device can provide much better response times on metadata traffics than a conventional disk. In our proposed method, metadata are also stored in the disk in a traditional way. The file system flushes “dirty” metadata on the MEMS-based device to disk periodically, making the system more robust against the crash of disk or MEMS device and the disk still demountable. Because the dedicated MEMS storage offloads bursty and less sequential metadata traffics from the disk, the disk can better service other requests, improving the system performance.

4.2 Using MEMS-based Storage Device as Both Metadata Storage and Disk Cache

A potential drawback of using MEMS metadata storage is that its performance heavily depends on the percentage of metadata operations in the workloads. Although it can significantly improve system performance under metadata-operation intensive workloads, the dedicated MEMS metadata storage device does less well under workloads with light metadata traffics.

To better utilize the MEMS device and improve the system performance, we combine our proposed method with a technique of using MEMS device as disk write buffer [35]. A small amount of storage (100–500 MB) on the dedicated MEMS metadata storage is used as disk write buffer. All metadata operations are serviced by the MEMS device and other write and hitted read operations are serviced by the MEMS disk write buffer.

MEMS disk write buffer takes advantage of fast response and non-volatile properties of the MEMS device. The MEMS device is used as a large non-volatile write buffer for disk. It is totally transparent to file systems. Therefore, no special revision needs to be applied to current file systems. All write requests that arrive at disk driver will be patched into logs and be flushed to MEMS device as soon as MEMS device is idle. The disk driver maintains the disk-MEMS mapping table, which indicates whether a disk sector is

device capacity	4.6 GB
number of tips	6400
maximum concurrent tips	3200
sled mobility in x and y	90 μm
sled acceleration in x and y	1033.2 m/s^2
sled access speed	30 mm/s
sled resonant frequency	1108.5 Hz
spring factor	75%
media bit cell size	30$\times$30 nm^2
bits per tip region (M$\times$N)	3000$\times$3000

Table 1: Default MEMS-based storage device parameters.

Statistics	01/13/1999	01/14/1999
number of requests	353,914	41,963
number of writes	161,531	31,997
percentage of writes	45.6%	76.3%
number of metadata requests	21,756	11,012
percentage of metadata requests	6.1%	26.2%
number of metadata write requests	12,612	9, 927
logical sequentiality	15.5%	17.4%

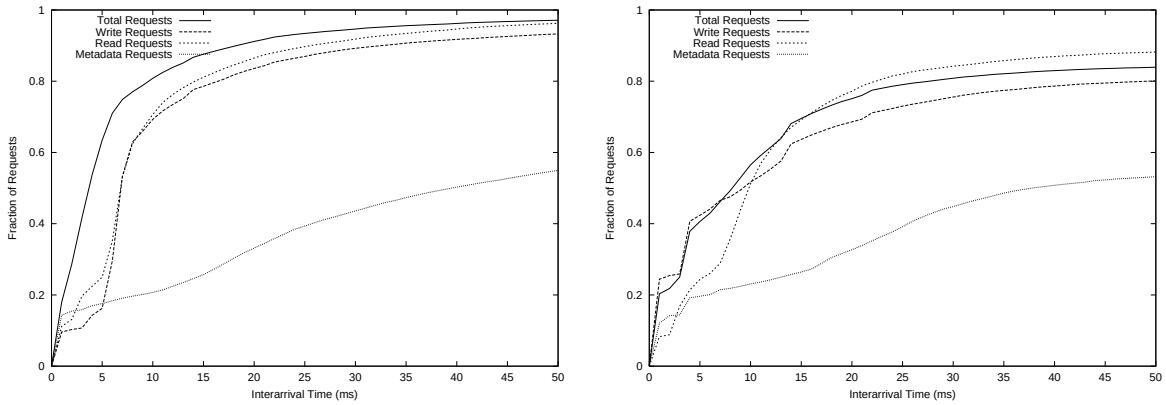
Table 2: Basic statistics of two one-day user traces.

active on the MEMS device. Therefore, all write requests can be serviced quickly and the disk bandwidth can be fully used by read requests. Simulations [35] showed that the average response time can be improved by a factor of 5 to 15 using a 128 MB MEMS write buffer.

5 Experimental Setup

Because MEMS storage devices are not readily available, we used DiskSim [9] to simulate requests and device service. This simulator has been used in previous studies of using MEMS-based storage in storage hierarchy [11, 28, 35], making it a good choice to allow direct comparison of our methods with prior work in the area. All of the simulations in Section 6 used the default parameters reported in Griffin *et al.* [11], as shown in Table 1.

Many studies of storage performance analysis use traces of real file system requests to produce more realistic results. We used one trace, labeled *user*, that was collected from an HP-UX workstation in 1999. This trace is the only modern workload with metadata operation information we currently have. The system was also traced and studied in 1992 [26]. The user trace is two one-day (01/13/1999 and 01/14/1999) subsets of the HP Hplajw user disk (Seagate Barracuda 4, 4.3 GB) traces. The basic statistics of these traces are summarized in Table 2. The average request arrival rate is 4.1 requests per second for the 01/13/1999 trace and 0.5 requests per second for the 01/14/1999 trace, respectively. The percentage of write requests of the 01/13/1999 and 01/14/1999 traces is 45.6% and 76.3%, respectively. The percentage of metadata requests of the 01/13/1999 and 01/14/1999 traces is 6.1% and 26.2%, respectively; the percentage of metadata writes in total write requests of the 01/13/1999 and 01/14/1999 traces is 7.8% and 31.0%. The logical sequentiality (percentage of requests that are at adjacent disk addresses or addresses spaced by the file system interleave factor) of the 01/13/1999 and 01/14/1999 traces is 15.5% and 17.4%, respectively.



(a) The 01/13/1999 trace

(b) The 01/14/1999 trace

Figure 3: Interarrival times for total requests, read requests, write requests, and metadata requests.

Although the daily average arrival rates of these traces are not high, the workloads are very bursty, as shown in Figure 3. The burstiness of metadata traffic and write traffic has great impact on the performance of our proposed MEMS metadata storage and MEMS disk write buffer. The more bursty the metadata and write traffics, the more the MEMS metadata storage and MEMS disk write buffer can mitigate the disk burden, and the higher performance gain we can obtain.

Figures 3(a) and 3(b) show the distributions of interarrival times of total requests, read requests, write requests, and metadata requests for the 01/13/1999 and 01/14/1999 traces, respectively. The fraction of requests with interarrival times less than 10 ms is 78.8% for the 01/13/1999 trace and 52.9% for the 01/14/1999 trace, respectively. In general, write traffic is more bursty than read traffic, particularly as shown in Figure 3(b). Metadata traffic is less bursty; however, a substantial fraction of metadata requests have interarrival times less than 1 ms, which is because operating systems periodically flush metadata to disk.

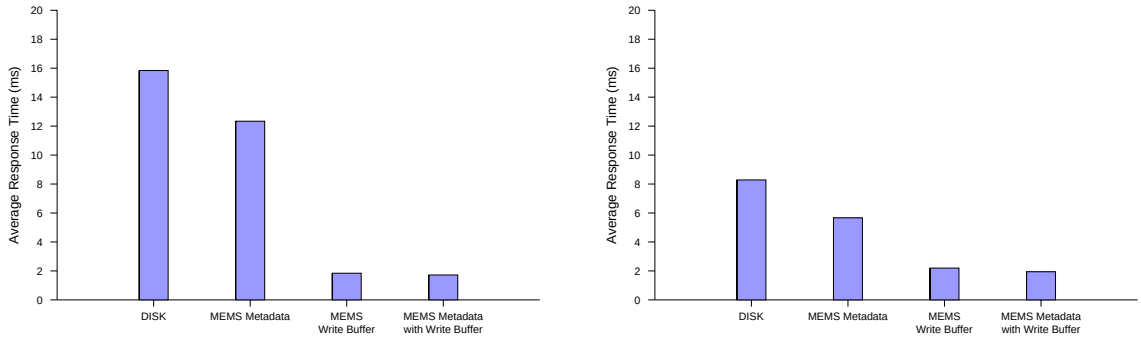
Because our work is simulation-based, we have to rely on existing block-level traces. The MEMS metadata storage method needs to map the metadata locations from the disk logical block numbers (LBNs) to the MEMS LBNs. As a variant of the standard UNIX fast file system, the HP-UX (10.20) file system also pre-allocates storage spaces for metadata. By analyzing the user traces, we found that the pre-allocation policy is to allocate 128 KB of metadata space for each 2048 KB of free space. Therefore, it is easy to directly map the LBNs of metadata from the disk address space to the MEMS metadata storage address space.

To generate the results we modified a version of DiskSim to include our MEMS metadata storage. Because an accurate disk model of Seagate Barracuda 4 is unavailable, we used the Quantum Atlas 10K disk (10025 rpms, 9GB) to act as our base disk although the workload was traced in a older Seagate disk. In the simulations on the MEMS metadata storage plus MEMS disk write buffer method and the pure MEMS disk write buffer method, we used 128 MB MEMS storage to be the disk write buffer. All of the simulations in Section 6 used the default parameters reported in Griffin *et al.* [11], as shown in Table 1.

6 Experimental Results

In this section, we will analyze the performance of storage systems using pure disk (DISK), disk with MEMS metadata storage (MMD), disk with MEMS write cache (MCD), and disk with both MEMS metadata storage and MEMS write cache (MMCD) under the two one-day (01/13/1999 and 01/14/1999) user workloads described in Section 5.

Figures 4(a) and 4(b) show the average response times of the storage system configurations of DISK,



(a) The 01/13/1999 trace

(b) The 01/14/1999 trace

Figure 4: Average response times for different architectural configurations.

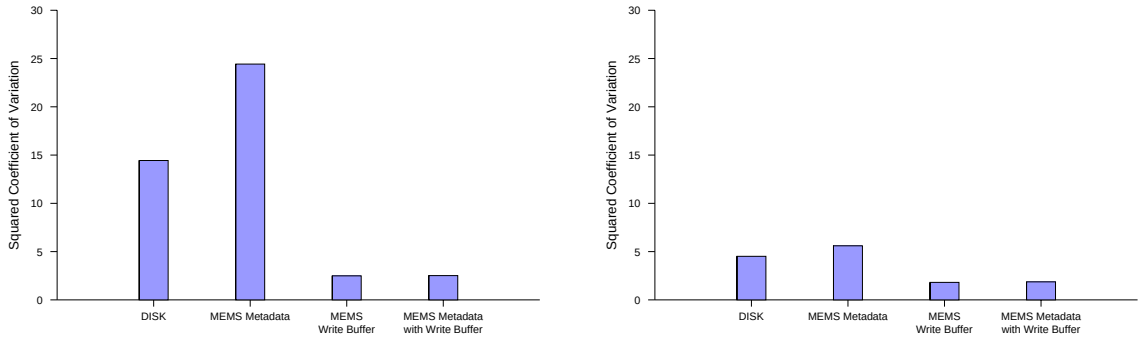
MMD, MCD, and MMCD under the 01/13/1999 and 01/14/1999 user traces, respectively. Compared to DISK, we can achieve 28–46% system performance improvement by using MMD, 270–760% system performance improvement by using MCD, and 330–820% system performance improvement by using MMCD, respectively.

The performance improvement by MMD is substantial (28–46%) but less significant than MCD and MMCD because the metadata traffic is not the majority of the simulated traces (6.1% and 26.2% for the 01/13/1999 and 01/14/1999 traces, respectively). Therefore, the majority of requests are still serviced by the slow disk, whose response time dominates the overall system performance. Because MMD offloads the metadata traffic from the disk to the MEMS metadata storage, the disk performance is improved by 11–21%. The performance improvement by MMD heavily depends on the workloads. The heavier the metadata traffic, the more performance benefit the system can gain from MMD.

MCD can improve the overall system performance by a factor of 2.7–7.6 under the workloads we studied. As discussed in Section 5, a significant percentage of requests in these workloads are writes (45.6% and 76.3% for the 01/13/1999 and 01/14/1999 traces, respectively) and in general write traffic is more bursty than read traffic. MCD can service the bursty write traffic very quickly on the MEMS device and significantly reduce the traffic to the slow disk. Because the traffic to the disk in MCD is less bursty than that in a pure disk system, the average response time of the disk in MCD is improved by 60–200%. Like MMD, the performance improvement by MCD also heavily depends on the workloads. The heavier the write traffic, the more performance benefit the system can gain from MCD. Fortunately, studies [26] showed that the majority of disk accesses (57%) are writes in general-purpose UNIX file systems. Therefore, we can expect a significant performance improvement by using MCD in a real system.

Combining the techniques of MMD and MCD, we found that MMCD can further improve the overall system performance by a factor of 3.3–8.2 under the workloads we studied. MMCD has the best performance among the different system configurations we studied because both the metadata and bursty write traffic are serviced by the fast MEMS device and only the non-metadata read requests are serviced by the slow disk.

In addition to the average response time, another important factor for system performance is the squared coefficient of variation of response times (σ^2/μ^2), where σ is the standard deviation of response times and μ is the average response time. This metric measures the consistency of the service time for requests [31, 37]. A low coefficient of variation means that the service times for the requests are likely to be near the average, while a high coefficient of variation is an indication that some requests may get fast service at the expense of others that can suffer starvation. In other words, it is a measure of fairness and starvation resistance of system services.



(a) The 01/13/1999 trace

(b) The 01/14/1999 trace

Figure 5: Squared coefficients of variation of response times for different architectural configurations.

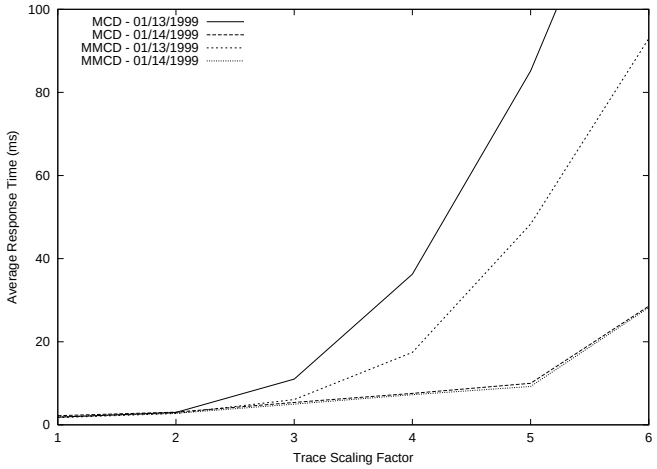


Figure 6: Average response times of MCD and MMCD.

Because MEMS-based storage devices have greater determinism in seek times than disks, they have tighter bounds on the expected service times. Figures 5(a) and 5(b) show the squared coefficients of variation of response times of the storage system configurations of DISK, MMD, MCD, and MMCD under the 01/13/1999 and 01/14/1999 user traces, respectively.

MCD and MMCD have much lower squared coefficients of variation of response times than DISK because a significant percentage of requests (50–80% for write requests and metadata requests) are serviced by the MEMS device, which has inherently more deterministic seek times than a disk. Therefore, the overall variabilities of MCD and MMCD are lower than DISK by a factor of 2.4 to 5.7, as shown in Figures 5(a) and 5(b). MMD has higher variability than DISK because a fast MEMS device can increase the overall variability of a disk-dominant system. In general, DISK, MMD, MCD, and MMCD have higher squared coefficients of variation of response times for the 01/13/1999 trace than for the 01/14/1999 trace. This is because the average request arrival rate of the 01/13/1999 trace is much higher than that of the 01/14/1999 trace so the queuing effects on the disk and the MEMS device are greater for the 01/13/1999 trace than for the 01/14/1999 trace.

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

We exercised the storage systems using pure disk (DISK), disk with MEMS metadata storage (MMD), disk with MEMS write cache (MCD), and disk with both MEMS metadata storage and MEMS write cache (MMCD) under the scaled traces. Both of the DISK and MMD systems become saturated quickly when the scaling factor increases because their overall performance is dominated by the performance of the slow disk. Therefore, we only studied the performance of the MCD and MMCD systems under scaled user traces.

Figure 6 shows the average response times of MCD and MMCD with different scaling factors for the 01/13/1999 and 01/14/1999 user traces. MCD and MMCD have similar average response times under the original and scaled 01/14/1999 traces and their performance is very good when the scaling factor is ≤ 5. When the scaling factor is > 5, the slow disk becomes saturated and the overall performance of MCD and MMCD suffers from the poor disk performance. MCD and MMCD works less well under the scaled 01/13/1999 traces than under the scaled 01/14/1999 traces because the 01/13/1999 trace has much higher average request arrival rate than the 01/14/1999 trace. Therefore, the disk becomes saturated more quickly under the 01/13/1999 trace than under the 01/14/1999 trace when the scaling factor increases. MMCD has better performance, up to 64–108%, than MCD under scaled 01/13/1999 traces because the very bursty metadata traffic is serviced by the fast MEMS-based device. As shown in Figure 3(a), about 15% of metadata requests have interarrival times less than 1 ms. When the trace is scaled up, the very bursty metadata traffic can quickly saturate the slow disk.

Although the MEMS write buffer method contributes the most performance improvement, the MEMS metadata storage method has the advantage of being able to quickly service metadata traffics, which can be very bursty. It also provides better system robustness by storing the important information of metadata in the MEMS device and the disk.

In summary, using a dedicated MEMS metadata storage device, we can achieve 28–46% system performance improvement on a user workload, depending on how much the metadata traffic counts for the whole workload. By combining a technique of using MEMS device as disk write buffer, we can improve the system performance by a factor of 3.3 to 8.2. Using these techniques, we can also promise better consistency on system performance than a disk system by a factor of 2.4 to 5.7.

7 Future Work

Due to the cost and capacity limitations, MEMS-based storage devices cannot totally replace disks in the foreseen future. We hope to integrate MEMS devices into current storage hierarchy to make the overall performance almost the same as that of using pure MEMS devices, and to achieve the capacity/price ratio comparable to that of traditional disk systems. This idea is very similar to the virtual memory system, which tries to build a fast memory system at the size and price of a disk system. The difference is that a MEMS virtual disk provides persistence data view, while virtual memory system is mainly designed for temporary data. MEMS virtual disks can be viewed as much fast disks and work under virtual memory systems.

In a MEMS virtual disk system, disk addresses are partitioned into large chunks (segments) and mapped to the MEMS device as needed. All requests are serviced at the MEMS device and the I/Os between the MEMS device and the disk are only large sequential chunks. We expects that the relative large capacity of MEMS devices ($\geq$ 1GB) can exploit the both temporal and spatial locality of the working set. Therefore, most requests can be serviced quickly by the MEMS device.

The MEMS virtual disk introduces a lot of interesting issues, such as what the segment replacement policy in the MEMS device should be, what the optimized size ratio for the MEMS device and the underlying disk system is, and so on. We plan to implement the MEMS virtual disk simulator in Disksim and study these problems in detail.

8 Conclusions

As new types of storage devices are developed, it is necessary to exploit their usages in current storage hierarchy. We proposed to use MEMS-based storage devices as dedicated metadata storage in file systems. Our simulations show that we can improve the system performance by 28–46% on a user workload, depending on how much the metadata traffic counts for the whole workload. By combining a technique of using the MEMS-based device as a disk write buffer, we can improve the system performance by a factor of 3.3 to 8.2. These techniques also promise better consistency on system performance than a disk system by a factor of 2.4 to 5.7.

Acknowledgments

I am grateful to Feng Wang for helping me understand Disksim and his codes on MEMS disk cache. I also thank Professor Ethan Miller for his guidance on this project.

References

[1] Center for highly integrated information processing and storage systems (CHIPS). www.chips.ece.cmu.edu, Carnegie Institute of Technology, Carnegie Mellon University.

[2] M. G. Baker, J. H. Hartman, M. D. Kupfer, K. W. Shirriff, and J. K. Ousterhout. Measurements of a distributed file system. In *Proceedings of the 13th ACM Symposium on Operating Systems Principles (SOSP '91)*, pages 198–212, Oct. 1991.

[3] C. Brown. Microprobes promise a new memory option. *E.E. Times*, page 6, Jan. 1998.

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

[5] L. R. Carley, G. R. Ganger, and D. F. Nagle. MEMS-based integrated-circuit mass-storage systems. *Communications of the ACM*, 43(11):72–80, Nov. 2000.

[6] S. H. Charap, P. L. Lu, and Y. He. Thermal stability of recorded information at high densities. *IEEE Transactions on Magnetics*, 33, Jan. 1997.

[7] T. Davis. Realizing a completely micromechanical data storage system (Kionix, Inc.). In *Diskcon 99 International Technical Conference*, 1999.

[8] M. Despont, J. Brugger, U. Drechsler, U. Dürig, W. Häberle, M. Lutwyche, H. Rothuizen, R. Stutz, R. Widmer, H. Rohrer, G. Binnig, and P. Vettiger. VLSI-NEMS chip for AFM data storage. In *Technical Digest. IEEE International MEMS 99 Conference. Twelfth IEEE International Conference on Micro Electro Mechanical Systems*, Orlando, FL, USA, Jan. 1999. IEEE.

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

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

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

[12] T. R. Haining and D. D. E. Long. Management policies for non-volatile write caches. In *Proceedings of the 18th IEEE International Performance, Computing and Communications Conference (IPCCC '99)*, pages 321–328, Phoenix, Feb. 1999. IEEE.

[13] J. L. Hennessy and D. A. Patterson. *Computer Architecture—A Quantitative Approach*. Morgan Kaufmann Publishers, 3rd edition, 2003.

[14] B. Hong, S. A. Brandt, D. D. E. Long, E. L. Miller, K. A. Glocer, and Z. N. J. Peterson. Zone-based shortest positioning time first scheduling for MEMS-based storage devices. Submitted to the 2003 USENIX Annual Technical Conference.

[15] Y. Hu and Q. Yang. Dcd—-disk caching disk: A new approach for boosting i/o performance. In *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, pages 169–178, May 1996.

[16] D. A. Koester, K. W. Markus, and M. D. Walters. Hot topics: MEMS: small machines for the microelectronics age. *Computer*, 29(1):93–94, Jan. 1996.

[17] 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 10th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '02)*, Fort Worth, TX, Oct. 2002. To appear.

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

[19] H. J. Mamin, B. D. Terris, L. S. Fan, S. Hoen, R. C. Barrett, and D. Rugar. High-density data storage using proximal probe techniques. *IBM Journal of Research and Development*, 1995.

[20] Nanochip Inc. Preliminary specifications advance information (Document NCM2040699). Nanochip web site, at http://www.nanochip.com/nctmrw2.pdf, 1996–9.

[21] Nanochip Inc. Nanochip: Array nanoprobe mass storage IC. Nanochip web site, at http://www.nanochip.com/preshand.pdf, 1999.

[22] Nanochip, Inc. Nanochip, Inc. Product Overview. In *Diskcon 99 International Technical Conference*, 1999.

[23] D. Psaltis and G. W. Burr. Holographic data storage. *IEEE Computer*, 31(2):52–60, Feb. 1998.

[24] S. Redfield and J. Willenbring. Holostore technology for higher levels of memory hierarchy. In *Eleventh IEEE Symposium on Mass Storage Systems*, Oct. 1991.

[25] M. Rosenblum and J. K. Ousterhout. The design and implementation of a log-structured file system. *ACM Transactions on Computer Systems*, 10(1):26–52, Feb. 1992.

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

[27] S. Savage and J. Wilkes. AFRAID – A Frequently Redundant Array of Independent disks. In *Proceedings of the 1996 USENIX Annual Technical Conference*, Jan. 1996.

[28] 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 (ASPLOS)*, pages 1–12, Cambridge, MA, Nov. 2000.

[29] G. T. Sincerbox and ed. Selected papers on holographic storage, vol.ms 95 of SPIE milestone series. In *Internal Society for Optical Engineering*, 1994.

[30] J. A. Solworth and C. U. Orji. Write-only disk caches. In H. Garcia-Molina and H. V. Jagadish, editors, *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, May 23-25, 1990*, pages 123–132. ACM Press, 1990.

[31] T. J. Teorey and T. B. Pinkerton. A comparative analysis of disk scheduling policies. *Communications of the ACM*, 15(3):177–184, Mar. 1972.

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

[33] M. Uysal, A. Merchant, and G. A. Alvarez. Using MEMS-based storage in disk arrays. Submitted to the 2003 Conference on File and Storage Technologies (FAST).

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

[35] F. Wang. Using MEMS-based storage device in storage hierarchy. M.sc. thesis, Department of Computer Science, University of California, Santa Cruz, 2002.

[36] T. M. Wong and J. Wilkes. My cache or yours? making storage more exclusive. In *Proceedings of the 2002 USENIX Annual Technical Conference*, pages 161–175, Monterey, CA, June 2002. USENIX.

[37] B. Worthington, G. Ganger, and Y. Patt. Scheduling algorithms for modern disk drives. In *Proceedings of the 1994 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 241–251, May 1994.