Appears in the Proceedings of the 2nd USENIX Conference on File and Storage Technologies, San Francisco, CA, March 2003

Using MEMS-based storage in disk arrays

Mustafa Uysal Arif Merchant Guillermo A. Alvarez
Hewlett-Packard Laboratories
1501 Page Mill Road, Palo Alto, CA 94304, USA

Abstract

Current disk arrays, the basic building blocks of high-performance storage systems, are built around two memory technologies: magnetic disk drives, and non-volatile DRAM caches. Disk latencies are higher by six orders of magnitude than non-volatile DRAM access times, but cache costs over 1000 times more per byte. A new storage technology based on microelectromechanical systems (*MEMS*) will soon offer a new set of performance and cost characteristics that bridge the gap between disk drives and the caches. We evaluate potential gains in performance and cost by incorporating MEMS-based storage in disk arrays. Our evaluation is based on exploring potential placements of MEMS-based storage in a disk array. We used detailed disk array simulators to replay I/O traces of real applications for the evaluation. We show that replacing disks with MEMS-based storage can improve the array performance dramatically, with a cost performance ratio several times better than conventional arrays even if MEMS storage costs ten times as much as disk. We also demonstrate that hybrid MEMS/disk arrays, which cost less than purely MEMS-based arrays, can provide substantial improvements in performance and cost/performance over conventional arrays.

1 Introduction

Disk arrays [16] are the main building blocks used to satisfy the performance and dependability requirements of current high-end storage systems. A disk array consists of a large number of disk drives, partially used to store redundant data that will allow transparent recovery from disk failures; controllers that interface with client hosts and maintain redundant data; and large battery-backed, non-volatile RAM (NVRAM) caches that allow optimizations such as prefetching, write-behind, and background destaging to mitigate the effects of high disk latencies. Most modern disk array architectures are based on the two-level NVRAM/disk hierarchy.

Guillermo A. Alvarez is now with IBM's Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA. Email: {uysal,arif}@hpl.hp.com, alvarezg@almaden.ibm.com.

The access latency gap between disk and NVRAM is currently almost six orders of magnitude (10 ms vs 50 ns), and is widening by about 50% per year. NVRAM costs about three orders of magnitude more per byte than disk drives. While specific applications may enjoy high hit rates from array caches, Wong and Wilkes [25] show that in most cases, NVRAM caches in high-end arrays can only hold 5% of the working set of applications, leading to low hit ratios. NVRAM is much less reliable than disk drives: typical mean time to failure for battery-backed NVRAM is only about 15K hours, compared to over a million hours for disk drives [19]. As a result, almost all disk arrays keep at least two copies of all dirty data in separate NVRAM buffers, further increasing cost. Finally, battery packs are cumbersome, as they must be capable of supplying enough power for the whole array; they can reach hundreds of pounds in weight and many cubic feet in size.

A disruptive new storage technology based on microelectromechanical systems (*MEMS*) will soon offer a new set of performance, cost and reliability characteristics that bridge the gap between NVRAM and disk drives. MEMS-based storage consists of chips containing thousands of small, mechanical probe tips that access data located on flat rectangles of storage media. The media is moved in two dimensions over fixed probe-tip heads, until the desired bits coincide with the heads. Positioning delays for MEMS-based storage are much smaller and more deterministic [8] than those of conventional disk drives. First, there is no rotational delay component in the positioning times. Second, MEMS-based storage is expected to achieve much higher densities (260–720 Gbit/in^2) [3], so seek distances are much shorter than in disk drives. Finally, since moving parts have much smaller masses than those in disks, they are much easier to accelerate. As a result, MEMS-based storage has the potential to bridge the cost and performance gaps between disk drives and NVRAM.

We explore the cost/performance implications of incorporating MEMS-based storage into disk array architectures. The total space of possible disk array architectures is too large to be explored systematically: the possibilities include the use of different data layouts, re-

dundancy schemes, caching methods, partial and complete replacement of disk and NVRAM with MEMS storage, variation in the proportions of MEMS storage, disk and NVRAM, as well as combinations of these methods. This study is intended to narrow the focus of future explorations by finding a few disk array architectures where the use of MEMS is most beneficial. We do this by devising a number of novel architectures to examine the potential placements of MEMS-based storage in a disk array. We concentrate on the performance part of users' requirements [28], as no predictions are yet available of the reliability and availability characteristics of MEMS chips—not even the cost of mass-produced chips is known precisely. Given that MEMS-based storage chips will not be commercially available before 2004, we used a detailed simulator replaying I/O traces from real applications for our performance study. By providing insight into the various architectural tradeoffs, our resulting cost/performance analysis can be seen as a first-cut indication on where to best spend money when designing disk arrays using MEMS-based storage devices. We found that replacing disks with MEMS storage in disk arrays improves both the performance and the performance/cost significantly, even if the MEMS storage costs ten times as much per byte as disks do. We also found that some hybrid MEMS/disk architectures offer an intermediate performance and cost between conventional disk arrays and MEMS-based arrays, with a performance/cost similar to MEMS-based arrays.

The remainder of this paper is organized as follows. Section 2 gives an overview of MEMS-based storage devices. We describe the disk array architectures under consideration in Section 3, and evaluate their performance and cost/performance characteristics in Section 4. Section 5 surveys related work; Section 6 contains a final discussion.

2 MEMS-based storage basics

MEMS-based storage chips consist of arrays of scanning probe tips that access a rectangular storage media sled. MEMS storage chips are built using standard photolithographic CMOS processes, and are expected to be massively produced around 2004. While the final design parameters for MEMS-based storage chips are still an active area of study, we concentrate on high-level device characteristics, as they relate to the present work. Carley, Ganger and Nagle [3] and Griffin *et al.* [8] provide detailed descriptions of MEMS-based storage.

As shown in Figure 1, MEMS-based storage chips contain one (or more than one, depending on design decisions [6]) rectangular array of several thousand probe tips. Data is stored on a rectangular media sled that moves in two dimensions with respect to the array of tips. We study the prevalent variety in which data storage is magnetic, as in disk drives; other recording materials such as phase-change media as in re-writable CDs are also possible. Each probe tip accesses a rectangular region on the media sled, and no two such regions overlap. For a given access, the media sled simultaneously seeks in the x and y directions until the corresponding tip is positioned on top of the start bit; then, the sled keeps moving in the y direction, while reading or writing consecutive bits along its trajectory. Since multiple probe tips can be active at any given time, most proposed data layouts rely on bit-interleaving, with multiple tips performing parallel reads or writes.

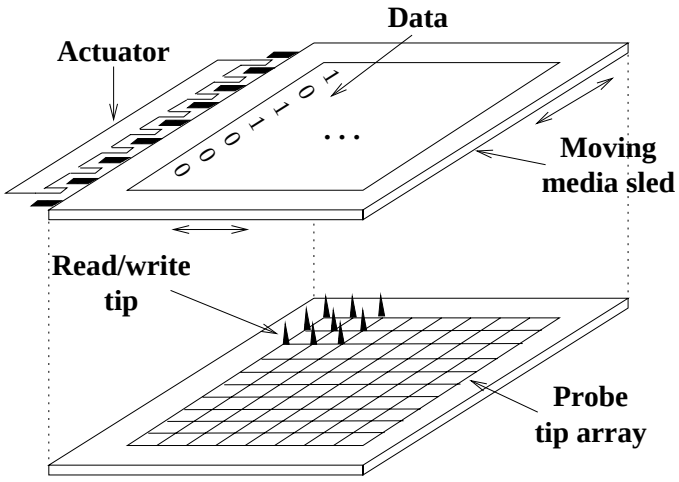


Figure 1: High-level schematic of a MEMS-based storage chip. All four sides of the media sled have actuators, and every crossing point in the tip array has a read/write tip. The sled is supported on top of the array by an ensemble of cantilevered springs, that move in two dimensions to seek to the coordinates where the data are.

This design has several important consequences. First, stored data is persistent and does not depend on continuous availability of a power source, as in battery-backed DRAM caches. Second, positioning delays depend on the relative positions of the sled and of the destination coordinates. Third, positioning delays are much smaller than in disk drives as there are no rotational delays, ranges of motion are in the order of a few millimeters, and components have small masses. Schlosser *et al.* found, using simulation [20], that typical access times for MEMS are in the order of 1–2 ms. The advantage over disk drives is still more pronounced for random workloads, where the disk spends most of the time positioning the head over the right bits instead of actually transferring data to/from the platters. Thus, MEMS positioning times are not only smaller on average than those of disks,

bit width (nm)	50
sled acceleration (g)	70
access speed (kbit/s)	400
settling time on x (ms)	0.431
total tips	6400
simultaneously active tips	640
max. throughput (MB/s)	25.6
number of sleds	1
per-sled capacity (GB)	2.56

Table 1: MEMS-chip parameters.

but their variances are also much lower.

Table 1 summarizes the parameters of the MEMS-based storage chips we used. These parameters correspond to conservative predictions [20] for the characteristics of the first generation of MEMS-based storage chips. Our simulated chips do not allow bidirectional reads, *i.e.*, accesses along the y axis must always be done while moving the sled in the same direction.

3 MEMS-based array architectures

Current high-end disk arrays store data in two main locations. They typically contain a fully-associative NVRAM cache in the order of tens of gigabytes. User data is ultimately stored in the *back-end* disk drives, for a total capacity of many terabytes. For fault tolerance, arrays keep redundant data at both levels in the memory hierarchy: as mirror copies or erasure-correcting codes on disks (RAID) [1, 16], and as dirty blocks mirrored in separate NVRAM cache banks in independent power domains. Disk arrays organize data storage into *Logical Units* (LUs), exporting a linear address space of blocks to client hosts.

The NVRAM cache in the disk arrays serves several purposes. First, it acts as a speed-matching buffer between the disks and storage area networks. Second, it allows the array to report the completion of write accesses as soon as the dirty data is in the (fault-tolerant) cache, without waiting for the disk write to complete. This optimization, commonly known as write-behind, decreases I/O service times and allows writes to be performed more efficiently in the background. Third, the NVRAM cache exploits the temporal locality in the workloads: multiple overwrites on the same data in the NVRAM cache are folded into a single write to the back-end during destaging; similarly, multiple read accesses to the same data can be directly served from the cache. Finally, read-ahead optimizations can exploit spatial locality in the workloads.

Our primary goal is to propose and evaluate alternative ways in which MEMS-based storage could improve both the performance and the cost/performance ratio of current disk arrays. We study architectures that use MEMS as either a total replacement for all back-end disks, or as a replacement for only some of them (hybrid architectures), or as a total replacement of current NVRAM cache. The hybrid architectures we have studied include several different data layouts and corresponding IO access policies, in order to determine if the different characteristics of disks and MEMS storage can be exploited for better performance. Despite the obvious fact that many other ways exist to incorporate MEMS into storage architectures, this methodology includes multiple points across the cost/performance spectrum for a reasonable degree of coverage of the potential alternatives.

3.1 MEMSdisk: Array disk replacement

The MEMSdisk architecture replaces each disk drive in the disk array by a bank of MEMS-based storage devices of the same capacity. Since the access latencies are much smaller for MEMS-based devices, the MEMSdisk architecture provides an upper bound on performance for all arrays that utilize MEMS-based storage for a fixed cache size. However, this comes at a potentially high cost per byte—up to an order of magnitude more expensive than disk drives of the same capacity.

3.2 MEMSmirror: Hybrid mirrored back-end

A RAID1/0 Logical Unit(LU) in a conventional disk array comprises a number of disk pairs where both disks in each pair contain exactly the same data. Writes complete as soon as they are written to the redundant NVRAM cache. The data is later flushed to the disks in the background, when the disks are otherwise idle, or when the cache starts filling up. Reads of data not found in the cache, however, require disk accesses, which have substantial latency.

MEMSmirror, depicted in Figure 2, alleviates this problem by having hybrid mirrored pairs: one disk drive and one bank of MEMS storage of the same capacity. Reads of data not in the cache are directed to the MEMS copy, which has much lower latency and higher throughput than the disk copy. Since the disk copy only handles writes, it can sustain a fairly high throughput, and the disk latencies are not an issue because of the NVRAM cache.

3.3 Logdisk: Hybrid replication with log-structured disk storage

As an attempt to get as close as possible to the performance of a purely MEMS-based array without the consequent cost, we propose an alternative where the data in MEMS storage devices is mirrored for redundancy on magnetic disks. In order to diminish the impact of slow

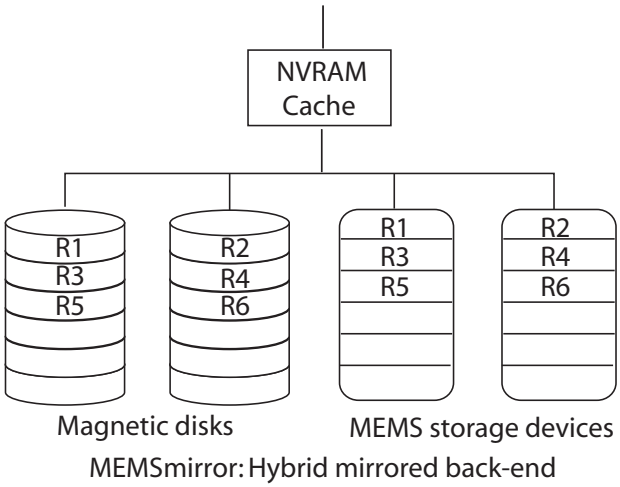


Figure 2: MEMSmirror: each disk is mirrored by a MEMS bank of equal capacity

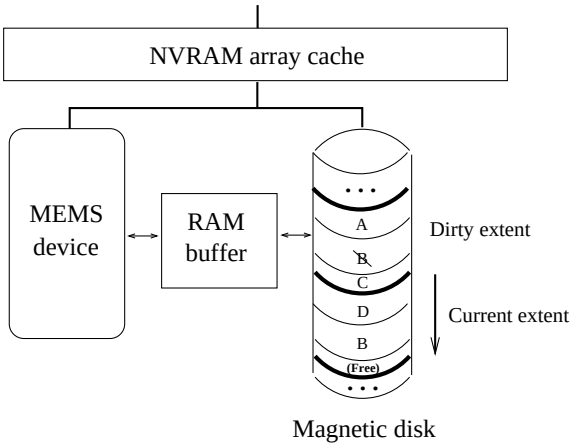


Figure 3: LogDisk architectural diagram: the RAM buffer allows asynchronous transfers between MEMS and disks. The current extent on disk is used in append-only mode, from top to bottom in the figure. Newer writes may render old data (*e.g.*, block B) invalid.

positioning times of the disk copy, data is written to disks as a log for near-to-zero positioning latency. Our disks are standard—they follow the standard update-in-place allocation policies, but we use them in a mostly append-only fashion.

Consider an array of n MEMS storage devices M_1, M_2, . . ., M_n, plus "mirror" disks D_1, D_2, . . ., D_n, of greater capacity than the corresponding MEMS devices. The MEMS storage devices are organized as a RAID 0 (stripe only, no redundancy) array [4]. Other components include NVRAM for saving metadata and for temporary storage of writes, and a RAM buffer for copying data between the MEMS storage devices and the disks. All data reads are serviced from the MEMS array. Writes are inserted into the NVRAM write cache and later flushed to both the MEMS array and the magnetic disk copies (when both copies are written, the data is cleared from the write cache.) Figure 3 depicts a simplified version of the LogDisk architecture, containing a single pair of devices.

Updates to the data stored in M_i are mirrored in D_i in a log-structured fashion. The disk writing algorithm is designed to minimize the fraction of time the disk head spends idle or seeking. The space in each disk is divided into fixed-size extents, one of which is marked as *current*. To write data on the disk copy, the array creates a fresh (*active*) copy of each overwritten block at the end of the current extent, and updates metadata stored in NVRAM to reflect the new status. Disks are therefore mostly used in sequential mode to append new data, sustaining their peak transfer rates with minimal positioning overhead. An I/O operation which overwrites a data block may supersede portions of a previously-written extent on disk, thus making the older extent sparsely populated with active data. Such extents are transformed into usable empty space by cleaning. At most one extent is being cleaned at any point in time. During idle periods the data corresponding to active blocks is read from the MEMS devices into the RAM buffer; whenever the disk is idle (*i.e.*, there is no write data in the write cache to be written to disk), it appends this data to the current extent, marking the corresponding data block in the extent being cleaned as invalid. An extent is *dirty* as long as it has active blocks; it becomes clean when no active blocks remain in it. The operation of the log-structured disk is quite similar to that of the log-structured file system LFS [18].

3.4 DualStripe: Hybrid replication for multiple access types

When redundancy is provided in a storage array by replicating the data, the replicas can be stored in different ways to optimize performance. In the LogDisk architecture described above, the disk copy is organized as a log to minimize the cost of writes; reads are directed primarily to the MEMS copy. However, disks can perform sequential reads very efficiently; we now describe an architecture in which the disk copy can service sequential accesses. The DualStripe architecture dynamically detects the sequentiality characteristics of the workload, and services accesses from the devices that are best suited for them according to the recent access history.

Consider an array with n MEMS storage devices, m magnetic disks, and a mirrored NVRAM write cache (Figure 4). The MEMS devices are organized in a

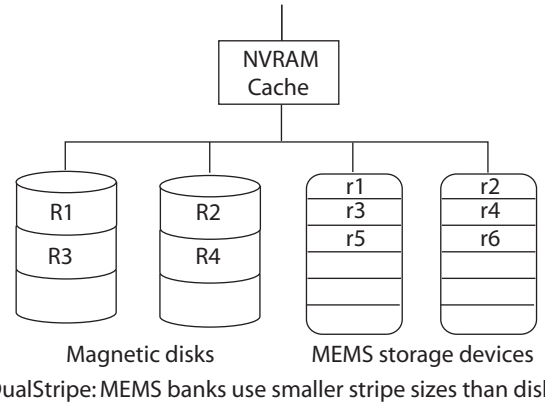


Figure 4: DualStripe hybrid array. Data is mirrored, one copy on magnetic disks with large stripes, another on MEMS storage devices with a small stripe size. Total capacity is equal on MEMS and disk.

RAID 0 layout and store one copy of the data stored on the array; the disks similarly form a RAID 0 group and store another copy. The stripe unit size for the MEMS RAID 0 group is small, to distribute accesses evenly and avoid hot-spots. The stripe unit size for the disk RAID 0 group is large, to reduce the positioning time cost for large sequential reads or writes.

Data written to the array is stored in the NVRAM write cache, to be flushed to the copies on MEMS and on disk when the devices are idle, or when forced because the cache's high-water mark is reached. Read data are obtained from the cache if present there. If not, they are read preferentially from the MEMS copy if the queue is short and from the disk copy if the queue length exceeds a threshold. However, if the read is detected to be part of a sequential run, the data are read from disk and a large subsequent block is prefetched to serve future requests in this sequential run. We expect this architecture to perform well for workloads where a substantial fraction of the reads are sequential.

Sequentiality detection works as follows: the array controller keeps a record of the addresses of the last $sequentialityWindow$ read requests. When a new read request arrives, this record is checked to see if the $sequentialityThreshold$ data blocks sequentially previous to this have been recently accessed. If so, the request is treated as part of a sequential run. In our implementation, we used $sequentialityWindow = 300$ and $sequentialityThreshold = 32KB$.

3.5 MEMScache: MEMS as array cache

The array architectures described so far explore the use of MEMS as a part of a redundancy scheme: for example, to store one of a pair of data replicas. In this section, we look at the other alternative: use of MEMS as a replacement for the NVRAM cache. Any redundant organization can be used for the disk back-end; in our implementations we have assumed a RAID-1/0 layout.

The operation of the MEMS primary cache is similar to that of the usual NVRAM cache. Reads are served from the cache if the data is already present in the cache; otherwise, the data are fetched from disk, and kept in the cache until flushed. Writes are saved in the cache, to be flushed in the background; usually, there are two copies of a data in the cache for redundancy. The dirty data is flushed to the back-end disk drives, when the amount of dirty data in the cache reaches a high-water mark; flushing continues in the background until the remaining dirty data is less than a low-water mark. The flushing process considers the location of dirty blocks in the disk storage so that: (a) the dirty blocks with continuous addresses are aggregated to be flushed in larger chunks and (b) the dirty blocks are written in ascending order of the addresses so that the access pattern for the flushes at the back-end is as close to sequential as possible.

The MEMS cache is a RAID-5 array of MEMS storage devices, organized as a log of cache lines. When data is written into the cache (whether due to a read from disk or an external write), one or more cache lines are appended to the log. If those addresses existed in the cache already, the corresponding locations are marked empty and later reclaimed by a log-cleaning process.

4 Experimental evaluation

We compared the performance and the cost/performance of the proposed architectures against a DiskOnly architecture using synthetic workloads and application IO traces. Performance comparisons are made by ignoring cost and looking at the proposed architectures configured to have equal capacity: we call these the *iso-capacity* comparisons. Cost performance is compared in two sets of experiments: by comparing the performance of MEMSdisk and DiskOnly architectures configured to cost same (*iso-cost* comparisons) and by comparing the cost/performance ratio of selected configurations of all architectures. Since MEMS-based storage chips are not currently available, we made these comparisons using a detailed simulator. We present the experiments and the results below.

4.1 Evaluation environment

We used a detailed event-driven storage system simulator called Pantheon [26]. Pantheon contains independent modules for separate components of the storage system, such as disks, controllers, non-volatile caches, array controllers, and buses. Each module's simulation can be

made extremely accurate, up to the extreme of running the code from the corresponding component's firmware. To exercise the simulated system, we had Pantheon generate synthetic workloads, and replay traces taken on real systems. In the configurations we used, Pantheon issued each I/O at the same time it was issued in the original trace (for the same replay speed), regardless of whether previous accesses had completed or not.

Our instantiations of Pantheon contain MEMS modules based on the state-of-the-art performance model described by Sivan-Zimet and Madhyastha [21]. We configured those modules to simulate a conservative version of the first generation of MEMS-based device characteristics used by Schlosser *et al.* [20]. Table 1 contains the parameters for the MEMS-based storage device characteristics used in our study. We also updated the disk models in Pantheon to simulate a disk drive based on an aggressive extrapolation of performance characteristics of modern high-performance disks — 3 ms average seek time, 20K rpm rotational speed, and a transfer bandwidth of 125 MB/s to/from the media. We configured the simulator to use four back-end buses to connect the disks to the disk array controller, we used an extrapolated bus bandwidth of 1GB/s for our simulations.

Our cost-performance comparisons used a cost of $6/GB for disks, based approximately on 2001 list prices for enterprise-class disks [17] and $8/MB for NVRAM, based on recent Dallas Semiconductor list prices. Since the cost for MEMS is unknown, we varied the relative per-byte cost of MEMS storage to disks between 1 and 10.

4.2 The workloads

In our evaluation, we used both synthetic workloads and several application traces: a file server containing home directories of a research group (*cello*), an e-mail server for a large company (*omail*), a database server running an on-line decision-support benchmark (*tpcd*), and a transaction processing benchmark (*tpcc*).

The filesystem trace (*cello*) represents one hour of user activity on April 20, 1999 on our main file server at HP Labs. The server stored a total of 63 filesystems containing user home directories, news server pools, customer workload traces, HP-UX OS development infrastructure, etc., for a total of 238 GB user data in a 479 GB physical storage. This is a typical I/O workload for a research group, involving software development, trace analysis, simulation, e-mail, etc. This trace has 370,000 I/O requests, with an average size of 23KB.

The *omail* workload is taken from the trace of accesses done by an OpenMail e-mail server [10] on a 640GB message store; the server was configured with 4487 users, of whom 1391 were active. The *omail* trace has 1.1 million I/O requests, with an average size of 7KB.

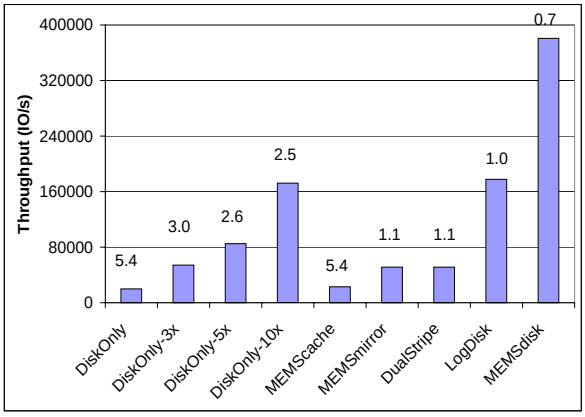


Figure 5: Performance comparison of architectures using synthetic workloads. Numbers above bars show normal-usage latency in *ms* (taken at 50% utilization).

The *tpcd* workload represents decision support systems; it consists of queries 5 and 7 from the TPC-D benchmark at the 300GB scale factor. This benchmark displays long complex database queries with both sequential and random accesses. The *tpcd* trace has 71,000 I/O requests, with an average size of 56KB.

The *tpcc* workload represents on-line transaction processing environments. It is based on a mid-range TPC-C benchmark configuration using one disk array and a two-processor server; overall, the transaction rate was 16.5K tpmC. The *tpcc* trace has 4.2 million I/O requests with an average size of 2KB.

We also used synthetic workloads in our experiments. Request sizes were drawn from an exponential distribution with a mean of 4KB, start addresses were drawn from a uniform distribution over the entire available device address range. The workloads had a varying ratio of read and write requests with 67% reads and the 33% writes as the default ratio, and a request inter-arrival times from an exponential distribution with a variable mean to simulate a variety of workload access intensity.

4.3 Results

Since MEMS-based devices have the potential to affect both the throughput and latency characteristics of disk arrays, we consider both performance metrics. Our baseline is the conventional DiskOnly architecture, *i.e.*, the combination of NVRAM cache and disk back-end found on current disk arrays, with a 2 GB of raw NVRAM and a 2 TB raw physical disks. For the MEMScache, we used 100 GB of logical MEMS storage (120GB raw including

the parity). Given that most of the architectures we introduced have a higher cost per byte than DiskOnly, it is legitimate to ask what the performance of DiskOnly would be if the extra money spent on MEMS were to be spent on additional disks instead, to get more spindles in the backend. If the data is striped over all disks, there are two potential performance advantages: more disk arms imply more potential parallelism, and partially-empty disks incur shorter seeks. To address this question we studied the *Isocost-X* architectures, *i.e.*, instances of DiskOnly in which the number of disk drives is increased until the cost matches that of a MEMSdisk architecture, assuming that the per-byte cost ratio of MEMS storage to disk is X.

4.3.1 Synthetic workloads

Our first set of experiments were designed to outline the performance of all the architectures studied. We used synthetically-generated workloads, which are easily scaled, to determine the maximum throughput and normal-usage of each architecture. The maximum throughput is found by measuring throughput with an offered load of 1 million IO/s, which is well above the throughput limit of any of the architectures. We then measured the *normal-usage* latency at 50% utilization by using a workload with an IO rate equal to half the maximum throughput.

Figure 5 shows the measured maximum throughput and normal-usage latencies for all the architectures studied. As one would expect, the MEMSdisk architecture is the clear leader in maximum throughput, with 380,000 IO/s, a 20-fold increase over the DiskOnly architecture. Among the hybrid (disk+MEMS) architectures, the LogDisk architecture is the best, with an approximately 177,000 IO/s maximum throughput; the MEMSdisk and the DualStripe architectures had substantially lower performance, with around 51,000 IO/s, as they are not as efficient as the LogDisk architecture for writes. In the DiskOnly-X iso-cost architectures, which increased the number of disk spindles to match the cost of the MEMSdisk architecture, the maximum throughput increased with the cost factor X, but even with a MEMS/disk cost factor of $X = 10$, the throughput was approximately half that for the MEMSdisk and slightly less than that for the LogDisk. The MEMScache architecture that replaced the NVRAM cache with a MEMS cache, improved the performance only slightly as the larger, MEMS-based cache was not effective for this workload.

The normal-usage latency numbers reflected the presence (or absence) of MEMS in the architecture: the DiskOnly architectures show a latency of 2.5–5.4ms, whereas the architectures using MEMS storage show a latency of 0.7–1.1ms.

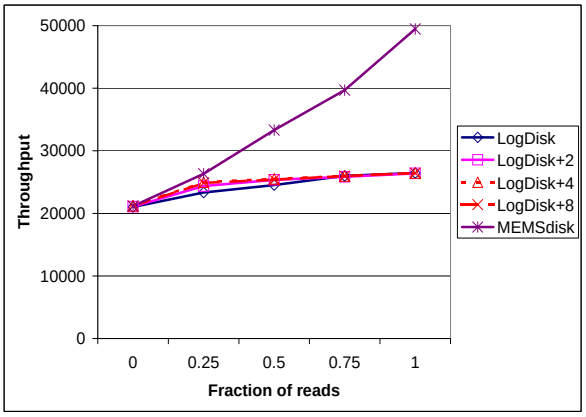
We conclude that arrays using MEMS storage will offer substantially higher throughputs and lower latencies than those using disks alone, even if the number of disk spindles is increased. The hybrid architectures, which combine disks and MEMS devices, improved the IO latency significantly, but only the LogDisk showed a significant improvement in throughput for the synthetic workloads. In the next section, we examine the LogDisk architecture more closely to better understand its performance characteristics.

4.3.2 Comparing LogDisk variants

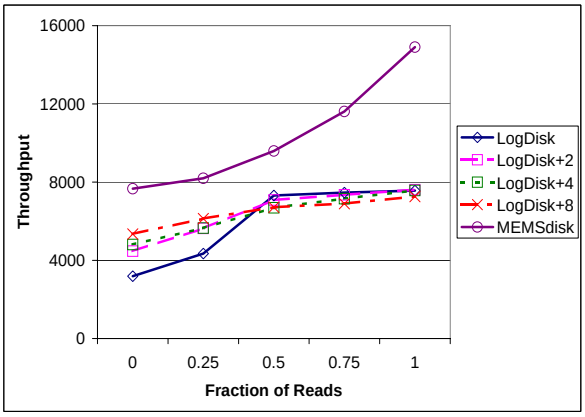
Our hybrid disk/MEMS architectures are predicated on the assumption that, with an appropriate layout and access policies, mirroring data on MEMS storage and disks can provide most of the performance of purely MEMS-based arrays at a lower cost. By default, we used MEMS storage and disks of equal capacity; however, it takes several MEMS chips to equal the capacity of a single disk, and these MEMS chips can sustain a higher aggregate IO rate than the disk. For the 36 GB disks we used in our experiments, we used 15 MEMS chips; each disk was able to sustain about 250 IO/s and each MEMS chip were sustaining about 1000 IO/s with 4K random requests. To determine whether increasing the number of disks would improve the performance of hybrid architectures, we compare the maximum throughput obtained from a LogDisk architecture with a varying number of disks. We use synthetic workloads with varying fractions of reads, for small (4KB) IOs and large (64KB) IOs.

Figure 6 shows the throughput obtained. The baselines are MEMSdisk (with only MEMS storage) and LogDisk (with equal capacities of MEMS and disk storage). The LogDisk architecture has two disks and 30 MEMS chips organized into two logical disks of 15 chips each. LogDisk-K represents a LogDisk variant with 2+K disks. For small IOs, the maximum throughput in LogDisk does not improve significantly when the number of disks is increased. Since the LogDisk uses large sequential writes to the disks with a bandwidth of about 125 MB/s per disk, disks do not become a bottleneck and the additional disks provide no performance benefits. For large IOs, the maximum throughput improves, especially for workloads with mostly writes, when the number of disks is increased by 2; beyond that, the improvements due to increasing the number of disks are small. For workloads with at least 50% reads, there is little difference between the various LogDisk variants.

Overall, we conclude that providing a number of disks to match the capacity of the MEMS storage is adequate for



(a) Small IOs (4KB)



(b) Large IOs (64KB)

Figure 6: Maximum throughputs for LogDisk variants with increased numbers of disks. LogDisk+K has K additional disks.

the LogDisk architecture. Adding extra disks for the log improves the maximum throughput when the workload consists mostly of large writes and the demand for disk bandwidth for log-writes is the greatest.

4.3.3 Comparisons using application traces

In order to be able to saturate all array architectures by replaying traces, we varied the intensity by scaling all inter-arrival times in the traces by a constant factor. Hence, the scale factor of one corresponds to replaying the trace at its original speed; the scale factor of two corresponds to replaying the trace twice as fast, and so on.

Figures 7 and 8 present the iso-capacity results, where equal capacity MEMS storage is used for the disks they replace. The MEMSdisk architecture had the highest maximum throughput for all the workloads studied and the lowest latencies for all but one (*tpcd*). Compared with the DiskOnly architecture, MEMSdisk decreased the mean I/O latency by a factor of between 4.0 and 6.5 at the knee of the latency curve for the DiskOnly architecture and increased the maximum throughput by a factor of between 4 and 28.

As expected, the hybrid architectures (MEMSmirror, DualStripe, and LogDisk) have a performance between that of DiskOnly and MEMSdisk. Maximum throughput ranged between 3 and 20 times that of DiskOnly. Among the hybrid architectures, LogDisk had the best performance for three of the four traces: *cello*, *omail* and *tpcc*. While all of the hybrid architectures improve read throughput by accessing the data in the MEMS copy, only the LogDisk architecture offers a significantly higher write throughput for a wide range of workloads

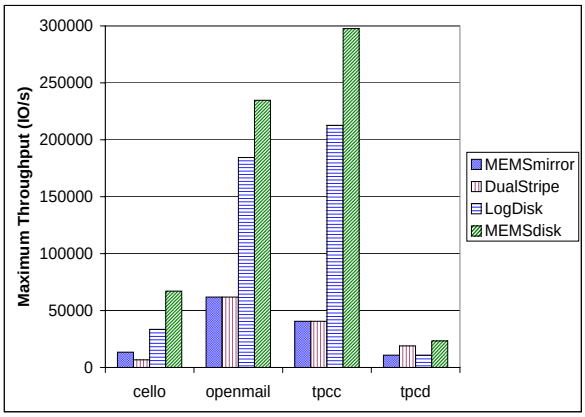
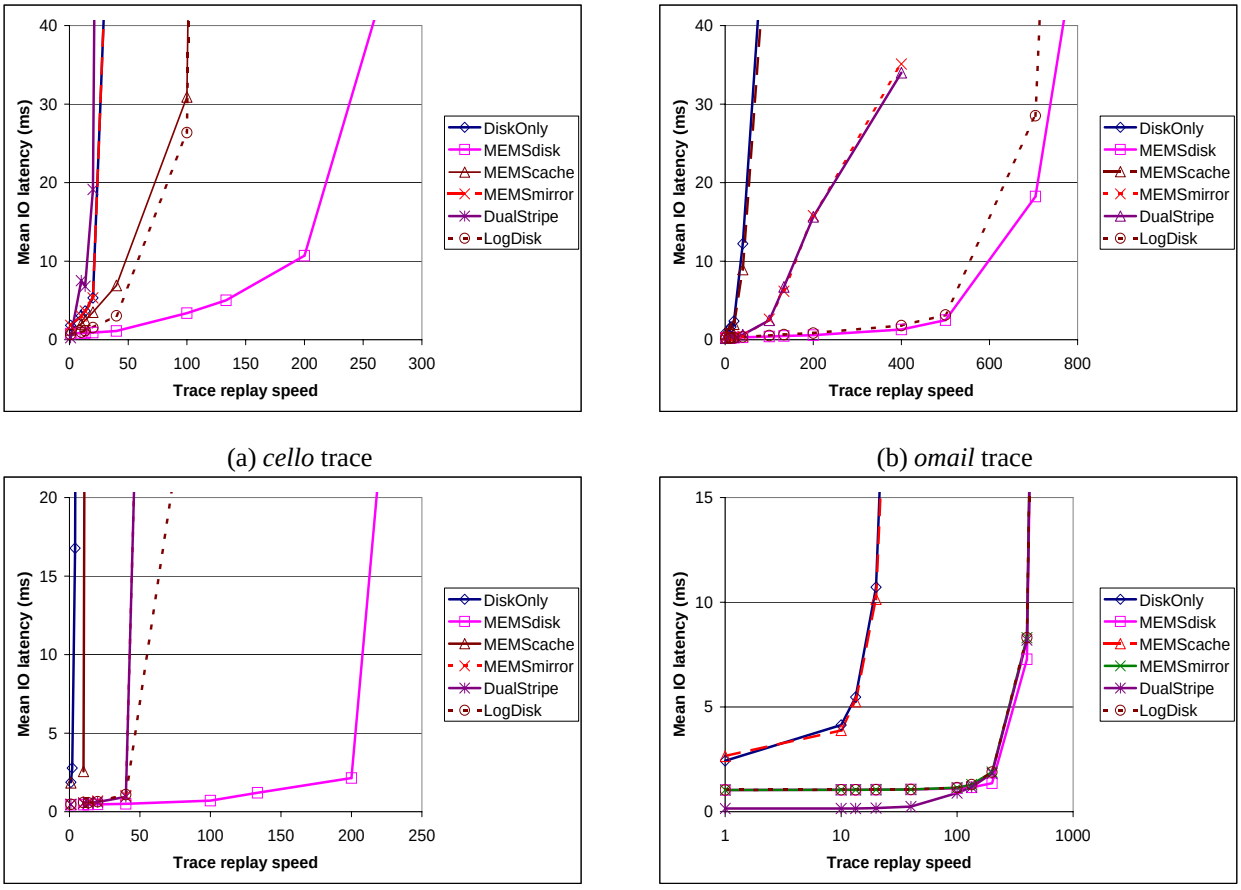


Figure 7: Maximum throughputs for hybrid architectures for the trace workloads.

(Section 4.3.2). For the *tpcd* workload, DualStripe had the highest throughput among the hybrid architectures and an I/O latency even lower than MEMSdisk. The aggressive prefetching behavior of DualStripe is particularly well suited to this workload, which exhibits mainly large sequential reads.

Replacing the NVRAM cache in a conventional disk array with a (much larger) MEMS cache is effective in reducing average response time, but does not improve throughput. For the original trace replay speed, MEMScache was able to reduce the I/O latency between 23% and 82%; this improvement in I/O latency is far lower than in the architectures where MEMS devices stored at least one copy of the data. The MEMS cache is ineffective for cold misses for the read operations, which contributes substantially to the latency. Although the MEMS



(a) *cello* trace

(b) *omail* trace

(c) *tpcc* trace

(d) *tpcd* trace

Figure 8: Comparison of architectures of the same capacity for the *cello*, *omail*, *tpcc*, and *tpcd* traces.

cache is able to sustain write bursts of much longer duration, there is a substantial performance penalty for each write compared to the NVRAM cache. On the other hand, the larger MEMS cache reduces the load on the back-end by eliminating most over-writes and coalescing dirty blocks in the cache.

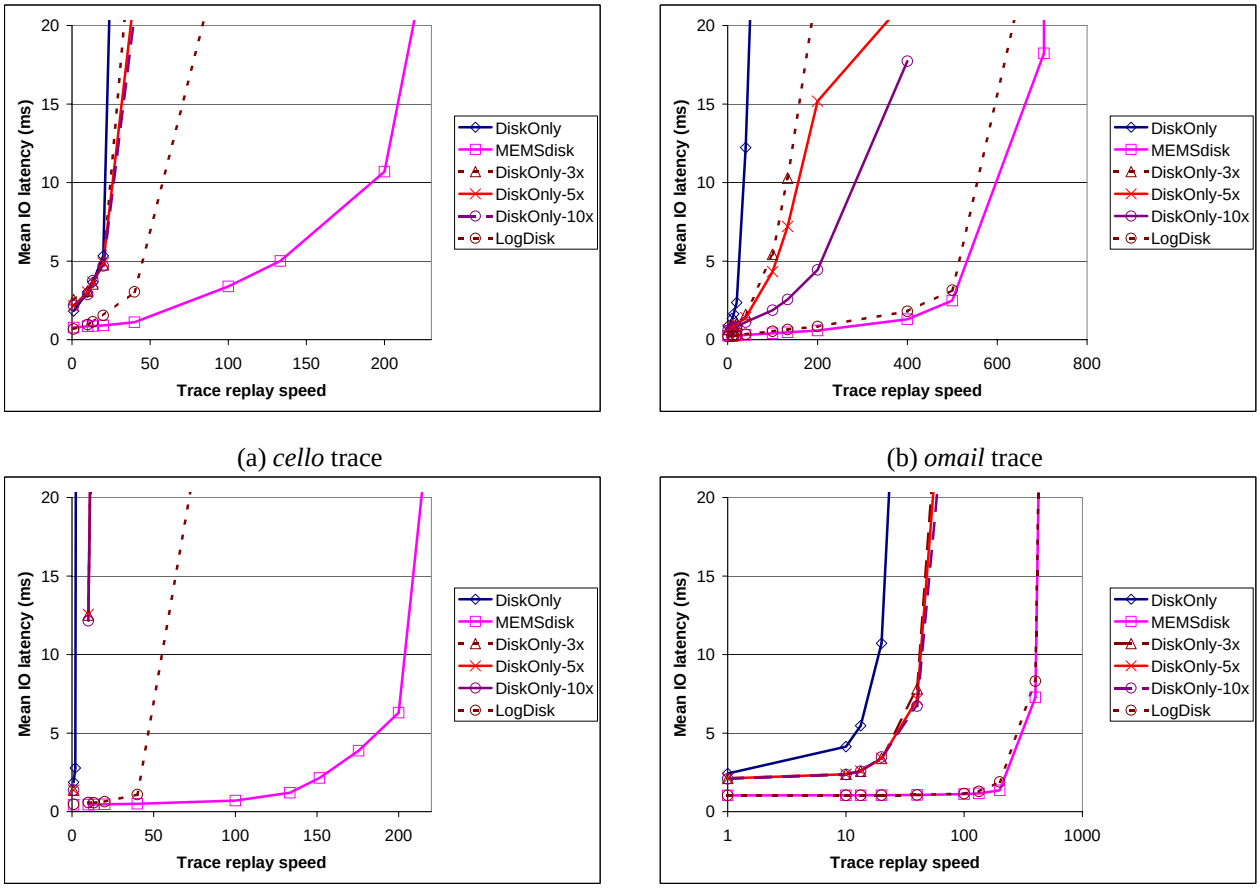
Overall, we conclude that replacing back-end disks in a disk array with MEMS storage has a dramatic impact on the performance of the array. A large part of that improvement can be obtained by placing only one of the two replicas of the data on MEMS storage, and it is effective to organize the disk replica in a log-structured fashion. Replacing the array NVRAM cache with a much larger MEMS cache is less effective in reducing the latency, and does not improve the throughput significantly.

4.3.4 Cost/performance analyses

We now study cost/performance ratios in two different ways. First, we compare MEMSdisk with DiskOnly architectures of equal cost, *i.e.*, the "Isocost-X" architectures. Then, we compare the cost/performance of several hybrid architectures with DiskOnly and MEMSdisk.

Recall that the architecture cost is based on the cost of the disks, NVRAM and MEMS used. We used a cost of $6/GB for disks, based approximately on 2001 list prices for enterprise-class disks [17] and $8/MB for NVRAM, based on recent Dallas Semiconductor list prices. Since the cost for MEMS is unknown, we varied the relative per-byte cost of MEMS storage to disks between 1 and 10.

Figure 9 compares the performance of disk arrays with MEMS-based storage and several iso-cost architectures. For the sake of clarity, we have shown only one hybrid architecture (LogDisk). The results show that array architectures with MEMS-based storage always exhibit lower latencies than purely disk-based ones, even when the number of disk spindles is increased. The maximum throughput offered by MEMS-based arrays is also substantially higher than that for DiskOnly architectures.



(a) *cello* trace

(b) *omail* trace

(c) *tpcc* trace

(d) *tpcd* trace

Figure 9: Comparison of architectures of equal cost for trace replays, in scenarios where the cost per byte for MEMS is 1 time, 3 times, 5 times, and 10 times that of magnetic disks.

We conclude that it is more cost-effective to replace disks with MEMS storage than simply to add more disks.

Figure 10 compares the performance per unit cost of the hybrid architectures against MEMSdisk and DiskOnly. Performance is measured by the maximum throughput achieved, averaged across the four trace workloads. As expected, MEMSdisk is the most cost-effective architecture when the MEMS-based storage costs no more than disks. LogDisk and MEMScache have similar cost-performance, with LogDisk sligtly higher. The performance/cost of LogDisk declines more slowly than that of MEMSdisk as MEMS cost increases: when the MEMS/disk cost ratio is 10, its performance/cost exceeds that of MEMSdisk by 38%. The performance/cost of the remaining hybrid architectures (MEMSmirror and DualStripe) is about the same as that of the DiskOnly architectures; however, when the MEMS/disk cost ratio is 10, the performance/cost of these hybrids drops below that of DiskOnly.

5 Related work

This paper combines the use of MEMS storage devices with several different redundancy schemes and layouts in efficient storage array architectures. The physical characteristics and performance of MEMS-based storage devices are discussed in several papers from the CMU Parallel Data Laboratory [3, 20, 8].

The use of redundant data layouts for reliability, load balance and improved performance is well established [1, 2, 16], and these are commonly used in modern disk arrays. In most such layouts, the performance of the disk is limited by the disk head seek time and rotational delays, particularly for workloads with small, non-sequential I/Os. Several mechanisms have been proposed to ameliorate the impact of positioning time for writes. A write cache can substantially reduce the number of disk writes and the perceived delay for writes [22, 9]; however, for reliability, these caches must generally use expensive NVRAM, ideally in a redundant configuration.

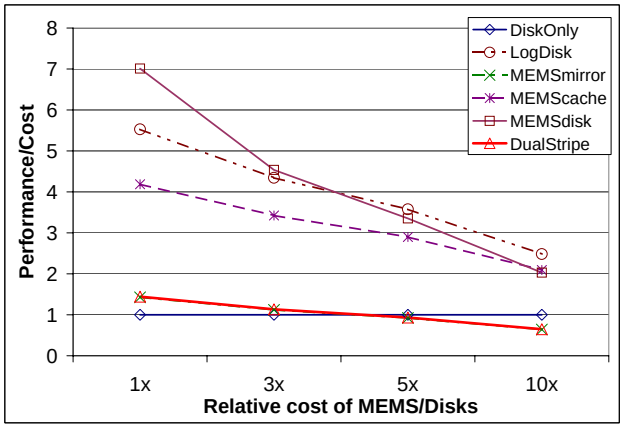


Figure 10: Performance/Cost of disk array architectures with MEMS-based storage.

Several papers have explored the use of logging (writing data to a sequential log) or eager-writing (writing to an unused location near the current position of the disk head). RAPID-Cache [12] provides redundancy to the NVRAM write cache through a logging disk, which is less expensive than replicating the NVRAM cache. In DCD (Disk Caching Disk)[11] and in Trail [5], a log disk is used to cache writes. Eager-writing was explored in the Loge disk controller [7]: writes were made to the free block closest to the current location of the disk head, and its location maintained through an indirection table in NVRAM. In Distorted Mirrors [23], data is mirrored, but only one of the copies (the master copy) is kept in a fixed order. Blocks in the master copy are updated in-place, but in the slave copy a block update can be written to the free block closest to the disk head. The main advantage is that write costs for the slave copy are lower than for mirrored disks where both copies are in a fixed location. In Doubly distorted mirrors [15], this is amended to defer the update to the master copy; the block is kept in a RAM cache, and redundancy is maintained by writing slave copies to both disks using eager-writing. The master location is updated from cache (and the slave location on that disk released) when there is a read from that cylinder, or the cache fills up, in which case the dirtiest cylinder is written out. Although this requires three disk writes for an update, the overall cost is lower than that of updating the master block immediately. Dual striping [13] attempts to reduce the cost of positioning time in a mirrored layout for reads while allowing load balancing across disks by using a large stripe size for one copy and a small stripe size for the other. Large reads can use the large stripe copy to reduce positioning time costs while small reads go to the small stripe copy.

The cost of small writes is particularly severe in RAID-4 and RAID-5 layouts, where every small write engenders four physical accesses: a read each of the old data and the corresponding parity (in order to compute the new parity value), and a write of the new data and the new parity. Parity logging [24] buffers parity updates in NVRAM and a log disk, eventually writing the parity out as large writes. AutoRAID [27] organizes the data hierarchically, with a RAID-10 level acting as a cache for a RAID-5 level; the RAID-5 level is log structured. Hot Mirroring [14] similarly combines RAID-1 and RAID-5 layouts, keeping hot data in the RAID-1 portion and cold data in the RAID-5 portion.

6 Conclusions and future work

We explored the performance and the performance/cost implications of incorporating MEMS-based storage into disk array architectures. We examined several possible placements for the MEMS storage in the disk array by (1) replacing all the disks with MEMS storage, (2) replacing the NVRAM cache with MEMS storage, and (3) replacing half the disks with MEMS storage. In the latter case, we proposed several novel alternative disk-array architectures designed to take advantage of the combination of disks and MEMS storage.

Replacing the disks with MEMS storage improves performance substantially in terms of latency (by a factor of 4 – 6.5) and throughput (by a factor of 4 – 28) depending on workload, but at high cost. Performance/cost, based on the average throughput of the trace workloads used, ranges between 2–7 times that of DiskOnly, depending on the MEMS/disk cost ratio.

The hybrid architectures, which store one copy of the data in MEMS storage, are able to achieve a significant fraction of the performance benefits of completely replacing disks with MEMS storage in disk arrays. Of the hybrid architectures studied, the LogDisk architecture offered the most consistent improvement in performance and the best performance/cost. The performance/cost of LogDisk is similar to that of purely MEMS-based arrays, and better than DiskOnly by a factor of 2.5–5.5, depending on the MEMS/disk cost ratio. Average latency is substantially lower than DiskOnly for all the hybrid architectures — by a factor of between 4 and 16 for the trace workloads studied here.

Replacing the NVRAM cache with a (much larger) MEMS cache is effective in reducing average response time by as much as 82%, but does not improve throughput because working set sizes are large. However, this may still be worth doing because of the low cost, as the performance/cost improvement over the conventional architectures ranged between 2.1–4.2, depending on the MEMS/disk cost ratio.

Overall, we conclude that replacing disks with MEMS storage in disk arrays will improve performance and performance/cost, even if MEMS storage costs ten times as much as disks on a per-byte basis. Placing one copy of the data on MEMS storage is also effective, offering an intermediate cost and performance between conventional disk arrays and purely MEMS-based arrays.

Extensions of our work include studying the performance of the different alternatives after an unrepaired failure, in degraded mode and during online reconstruction. Another extension would be to incorporate reliability metrics into the architectural comparison when reliability estimates become available for MEMS-based storage devices.

Acknowledgements

Thanks are due to Miriam Sivan-Zimet for writing the initial version of the MEMS simulator module, Tara Madhyastha for making it available to us, Steve Schlosser for discussions on the subject of MEMS-based storage, Dick Henze and Bill Hooper for helping us find out the costs of the different technologies, and John Wilkes for helping us disentangle many Pantheon snares.

References

[1] G.A. Alvarez, W.A. Burkhard, and F. Cristian. Tolerating multiple failures in RAID architectures with optimal storage and uniform declustering. In *Proceedings of the 24th International Symposium on Computer Architecture (ISCA)*, pages 62–72. ACM Press, June 1997.

[2] D. Bitton and J. Gray. Disk shadowing. In Francois Bancilhon and David J. DeWitt, editors, *Proceedings of 14th International Conference on Very Large Data Bases (VLDB)*, pages 331–8. Morgan Kaufmann, August 1988.

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

[4] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, and David A. Patterson. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, 1994.

[5] T.C. Chiueh. Trail: a track-based logging disk architecture for zero-overhead writes. In *Proceedings of 1993 IEEE International Conference on Computer Design ICCD*, pages 339–343, October 1993.

[6] I. Dramaliev and T.M. Madhyastha. Optimizing probe-based storage. In *2nd USENIX Conference on File and Storage Technologies (FAST)*, Mar-Apr 2003.

[7] R.M. English and A.A. Stepanov. Loge: a self-organizing storage device. In *Proceedings of USENIX Winter'92 Technical Conference*, pages 237–51. USENIX, January 1992.

[8] 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*, pages 56–65, June 2000.

[9] T. Haining and D. Long. Management policies for non-volatile write caches. In *Proceedings of the 18th IEEE International Performance, Computing and Communications Conference*, pages 321–328, February 1999.

[10] Hewlett-Packard Company, Palo Alto, CA. *OpenMail Technical Reference Guide*, 2.0 edition, 2001. Part No. B2280-90064.

[11] 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 (ISCA)*, pages 169–178. ACM Press, May 1996.

[12] Y. Hu, Q. Yang, and T. Nightingale. RAPID-cache - a reliable and inexpensive write cache for disk I/O systems. In *Proceedings of the Fifth International Symposium on High-Performance Computer Architecture (HPCA)*, pages 204–213. IEEE Computer Society, January 1999.

[13] A. Merchant and P.S Yu. Analytic modeling and comparisons of striping strategies for replicated disk arrays. *IEEE Transactions on Computers*, 44(3):419–433, March 1995.

[14] K. Mogi and M. Kitsuregawa. Hot mirroring: a method of hiding parity update penalty and degradation during rebuilds for RAID5. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 183–194, June 1996.

[15] C.U. Orji and J.A. Solworth. Doubly distorted mirrors. In *Proceedings of the 1993 ACM SIGMOD conference*, pages 307–316. ACM Press, May 1993.

[16] D.A. Patterson, G. Gibson, and R. H. Katz. A case for redundant arrays of inexpensive disks

(RAID). In Harran Boral and Per-Ake Larson, editors, *Proceedings of 1988 ACM SIGMOD International Conference on Management of Data*, pages 109–16, June 1988.

[17] D. Reinsel. Worldwide hard disk drive market forecast and analysis, 2000-2005. IDC, May 2001. IDC Report 24603.

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

[19] S. Savage and J. Wilkes. AFRAID – a frequently redundant array of independent disks. In *Proc. of the 1996 USENIX Technical Conference*, pages 27–39, January 1996.

[20] 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-IX)*, pages 1–12. ACM Press, November 2000.

[21] M. Sivan-Zimet and T. Madhyastha. Workload based optimization of probe-based storage. In *Proceedings of SIGMETRICS*, pages 256–7, June 2001.

[22] 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*, pages 123–132, May 1990.

[23] J.A. Solworth and C.U. Orji. Distorted mirrors. In *Proceedings of the First International Conference on Parallel and Distributed Information Systems*, pages 10–17. IEEE Computer Society, December 1991.

[24] D. Stodolsky, G. Gibson, and M. Holland. Parity logging: Overcoming the small write problem in redundant disk arrays. In *Proceedings of the 20th Annual International Symposium on Computer Architecture*, pages 64–75. IEEE Computer Society Press, May 1993.

[25] T.M. Wong and J. Wilkes. My cache or yours. In *Proc. of the 2002 USENIX Annual Technical Conference*, June 2002.

[26] J. Wilkes. The Pantheon storage-system simulator. Technical report, Storage Systems Program, Hewlett-Packard Laboratories, Palo Alto, CA, 29 December 1995.

[27] J. Wilkes, R. Golding, C. Staelin, and T. Sullivan. The HP AutoRAID hierarchical storage system. In *Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles*, pages 96–108. ACM Press, December 1995.

[28] J. Wilkes and R. Stata. Specifying data availability in multi-device file systems. *Operating Systems Review*, 25(1):56–9, January 1991.