Processor-Embedded Distributed MEMS-Based Storage Systems for High-Performance I/O

Steve C. Chiu, Wei-keng Liao, and Alok N. Choudhary
Department of Electrical and Computer Engineering
Northwestern University
Evanston, IL 60208-3118 U.S.A.
{schiu,wkliao,choudhar}@ece.northwestern.edu

Abstract

Built upon new data organization and access characteristics, MEMS-based storage devices have come under consideration as an alternative to disks for large data-intensive applications. While not already in commercial production, MEMS-based storage devices have outperformed disks in device-level simulations. Processor-embedded distributed disks improved performance of workloads by offloading application-level processing to the storage. To exploit the potential benefits offered by these emerging storage technologies and offloading models, we propose a processor-embedded distributed MEMS-based storage architecture. Using validated MEMS device models, we evaluate the proposed architecture with representative database and data mining workloads. Our results show that MEMS-based storage improves the overall performance of these workloads over disk-based systems. Furthermore, MEMS-based storage devices transformed the characteristics of several workloads, indicating a shift of performance bottleneck from I/O to the interconnect or processing power of the storage system, which can impact the design points for future storage architectures.

1 Introduction

For data-intensive applications, large disk arrays have traditionally been employed to provide the required bandwidth and reliability via their high level of parallelism. To further improve the performance of storage systems, network-attached storage (NAS) [18] and intelligent disk (IDISK) architectures [13] were proposed to embed user-level processing directly in the storage device. Recently, micro-electromechanical systems (MEMS), a chip-level technology, have come under consideration as an alternative to disks for mass data storage. The performance gap between processor and disk storage has grown to six orders of magnitude due to rapid advancement in processor technology and VLSI fabrication, and is continuing to widen by about 50% per year [22]. As a result, MEMS-based storage is being studied to supplement or even replace disks as the persistent storage medium for large data sets to overcome the I/O bottleneck associated with disks.

Because MEMS-based storage devices hold and access data differently than disks, workloads processed on MEMS-based storage may exhibit different behaviors than those with disk-based systems. For example, the work in [9] described a processor-embedded storage architecture that offloads processing to an array of distributed disks. This placed processing close to data and reduced traffic on the storage network to improve overall performance. But workloads processed on this architecture remained I/O-bound due to the disk's relatively slow mechanical components and large positioning delays. In this paper, we investigate the impact and potential benefits of MEMS-based storage with a distributed and offloaded processing model by the design of a processor-embedded MEMS-based architecture. In addition to performance improvement, we also study the changes in various components of the total response time due to the use of MEMS-based storage. We examine such changes in the characteristics of a workload, e.g. from I/O-bound to compute-bound, or from I/O-bound to network-limited, when shifting from disk-based storage to MEMS-based storage.

Our test results demonstrate that MEMS-based storage systems improved the overall performance over disk-based systems, as we had expected from a high-performance I/O device. MEMS-based storage also changed the characteristics of several workloads. When a workload is transformed from I/O-bound to compute-bound, it suggests that processing power has become the performance bottleneck. Likewise, the shift of a workload from I/O-bound to network-limited suggests the need for faster storage interconnect.

Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04)
IEEE COMPUTER SOCIETY

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

Such fundamental changes could significantly influence storage architecture design. Therefore, we contrast MEMS-based storage systems with a disk-based system to illustrate the importance of these changes.

The remainder of this paper is organized as follows. In Section 2, we review the background work related to processor-embedded disk storage and MEMS-based storage devices. Section 3 presents our distributed MEMS-based storage architecture, and discusses the system architecture and performance models for this work. Section 4 specifies the workloads used to evaluate the disk-based and MEMS-based storage architectures. Section 5 presents our experimental results. Conclusion and future work are provided in Section 6.

2 Background

2.1 Processor-Embedded Disk Storage Devices

Our earlier work proposed a fully distributed smart disk architecture [9]. With an embedded processor, on-disk memory, a network interface controller (NIC), and local disk space, the "smart disk" (SD) offloads processing from the host (or server) processors to reduce data traffic on the storage network. Each SD would run the Linux kernel which occupies approximately 450 KB of memory. The SD's on-disk OS (called SD OS) would execute the SQL primitives and parsed queries from a remote client running a commercial database. For non-database workloads, the SD OS executes the offloaded and parallelized application code (in binaries) which are distributed based on the single-instruction-multiple-data (SIMD) processing model.

Figure 1 illustrates the software architecture for the SD system. Based on the Linux kernel, the SD OS provides services for parsed SQL (for database queries) or distributed code, execution initialization, and data communication at the layer immediately below the user space. These higher level services are in turn supported by modules that perform processing control (for synchronization and states of execution), workload-optimized data access, and memory buffer management. A NIC driver module provides the TCP/IP, InfiniBand [12], or some other high-connectivity network interface. The InfiniBand Architecture (IBA), an emerging server I/O standard developed by the InfiniBand Trade Association (IBTA), was designed to overcome the limitations of the shared bus architecture in today's PCI/PCI-X, Ethernet, Fiber Channel, and other similar networks. IBA uses channel-based fabric, host and target channel adapters, switches and routers to provide simultaneous point-to-point communications between end nodes.

To minimize the footprint of SD OS, the SCSI (or a higher performance) disk driver is assumed to be part of the SD device firmware/hardware layer. Furthermore, to

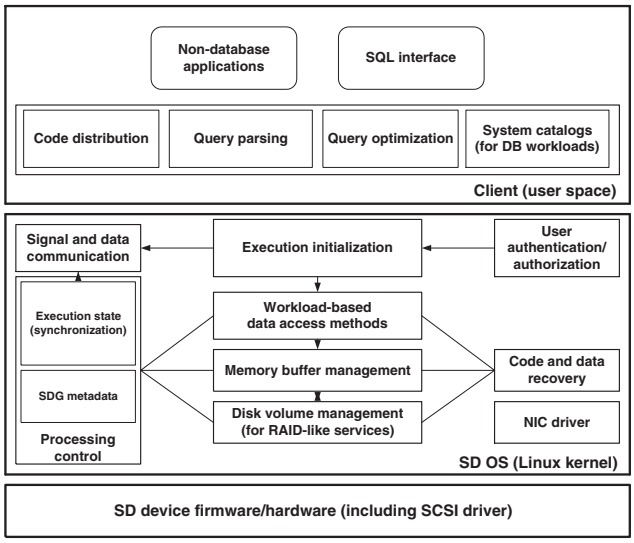

Figure 1. Software architecture for distributed smart storage systems

provide availability and reliability required by I/O-intensive workloads, several additional modules, while not already implemented, have also been designed into the SD OS layer. These include a code and data recovery module which will work with a disk volume manager to support RAID-like services, since a typical SD system will likely employ a large number of SDs. The SD architecture can also be extended to more than one group of SDs (called SDGs) for operations such as the *join* that process two or more separate database tables residing on different SDGs.

Work in [13], [18] and [9], along with the device level models described in [7], have resulted in a well established disk-based architecture. The processing models proposed in [15] and [9] provided the necessary architectures and software designs for processing I/O-intensive workloads on SDs. These studies have also demonstrated the performance improvement provided by distributed SD architectures over traditional host-based and cluster-based architectures.

2.2 MEMS-based Storage Devices

MEMS are micron-sized devices fabricated from photolithographic (an IC production technique) processes [14]. Instead of using a rotating spindle and dividing a disk into sectors, tracks and cylinders, MEMS-based storage devices consist of a moving rectangular media sled and an array of read/write tips. The media sled is spring-mounted over the tips and can be moved by actuators in the X, Y and Z directions. While the system "seeks" in the X direction and reads/writes data in the Y direction, it also moves in the Z direction to control the distance between the tips and the

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

media sled. In the case of CMU's CHIPS [1] storage device, the MEMS-based system contains an array of 80 x 80 tips with each tip accessing a region of 2,500 x 2,500 bits. Of the 6,400 tips, only 1,280 tips may be activated at the same time due to power and heat dissipation constraints. Other MEMS-based storage projects include those undertaken by Agilent [2], IBM's Millipede [3], Kionix [4], and Nanochip [5], to name a few. The differing characteristics of MEMS-based storage devices necessitate re-evaluation of several design issues, as itemized below.

2.2.1 I/O Scheduling Algorithm

Unlike disks, which have been modeled as one-dimensional devices, MEMS-based storage devices follow a two-dimensional data layout and access model. Hence, using the existing disk I/O scheduling algorithms may not fully realize the potential of MEMS-based storage. The work in [22] proposed a minimum spanning tree (MST)-based I/O scheduling algorithm capable of within twice the optimal time for any workload.

2.2.2 Data Placement Scheme

While many contend that MEMS-based storage can also adopt existing data layout models for disks, others have exploited the unique properties of MEMS-based storage with new data placement designs. A data placement scheme called Flexible Retrieval Model (FRM) was proposed for storing relational database tables on MEMS-based storage devices [21]. In our work, we have used a simplified data layout as discussed in Section 3. This simplified data layout policy represents a general-case scenario for many workloads, and has been validated for modeling accuracy when applied to the MEMS-based storage devices used in our evaluation.

3 Distributed Smart MEMS System

3.1 System Architecture

Compared to disk-based storage systems, MEMS-based storage systems can provide greater potential to integrate computation with mass storage to create a complete system-on-chip [8]. Therefore, it would be logical to consider the processing of large data- and I/O-intensive workloads on a distributed system of processor-embedded MEMS, or "smart MEMS" (SM) with the same processing models as discussed in [9]. Lightweight and distributed data-intensive applications that involve PDAs, tablet PCs, mobile sensors, or ubiquitous active storage nodes represent just some of the potential opportunities for this SM storage system.

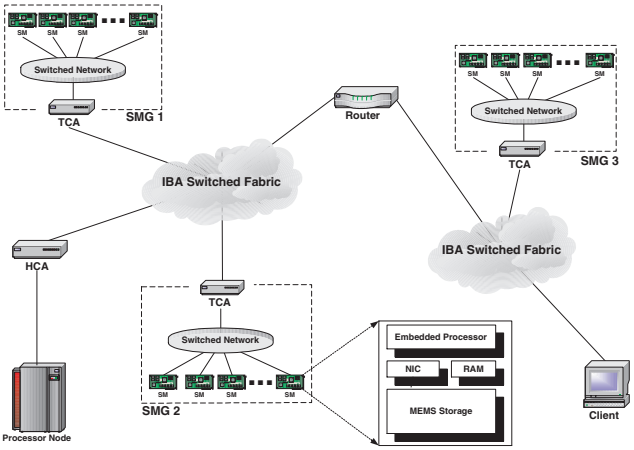

Figure 2. Multiple-SMG architecture over IBA-based network

Figure 2 depicts a distributed MEMS-based architecture in "multiple-SMG" configuration, similar to the multiple-SDG model proposed in [9]. While the overall architecture and storage interconnect remain the same as the disk-based system, the local disk storage has now been replaced with the MEMS-based storage device discussed in Section 2.2. The InfiniBand-based network as specified in [9] is assumed to be the storage interconnect for this proposed multiple-SMG architecture.

3.2 Performance Modeling

The system-level models for the MEMS-based storage proposed in this work is based on the MEMS device performance model used in [19], as well as the specifications used by the *DiskSim 3.0* [7] simulator. At the device level, the MEMS G1 model approximates the storage device in [10] with a larger per-sled capacity (2.56 GB vs. 2.1 GB) and half the number of simultaneously active tips (640 vs. 1280). On the other hand, the MEMS G2 model represents an overall performance level equivalent to that evaluated in [8]. A data organization scheme similar to those specified in [8] and [10] is employed, except for the access method, which is discussed below. While the work in [22] proposes a more advanced I/O scheduling approach, no optimization beyond that provided by *DiskSim*'s SCSI-based block level scheduling is performed during our simulations.

3.2.1 Storage Device Models

Table 1 presents the essential parameters for three generations of MEMS-based storage devices as specified in [19]. Generation 1 (G1) model represents the most conservative

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

Table 1. Parameters for MEMS-based storage devices G1, G2 and G3 (see also [19])

Parameter	G1	G2	G3
bit width (nm)	50	40	30
sled acceleration (g)	70	82	105
access speed (Kbits/s)	400	700	1000
X settling time (ms)	0.431	0.215	0.144
total number of tips	6400	6400	6400
number of active tips	640	1280	3200
max throughput (MB/s)	25.6	89.6	320
number of media sleds	1	1	1
per-sled capacity (GB)	2.56	4.00	7.11
bi-directional access	no	yes	yes

of the three– a per-tip data rate of 400 Kbits/s with unidirectional access for both reads and writes. The G2 model contains several enhancements over G1, including bidirectional (+Y and -Y) access and a 700 Kbits/s per-tip data rate. The G3 model represents the most advanced device with a 1000 Kbits/s data rate, the most active tips, and a much larger throughput. A validated *DiskSim* module for the HP C2490A hard disk is used for our comparison on MEMS-based and disk-based devices. Performance characteristics of HP C2490A are specified in Table 2.

3.2.2 Data Layout and Access

The media sled of the MEMS-based storage is divided into rectangular regions of M x N bits. Each of these regions is accessed by one probe tip. The media sled moves in the X direction (seek), then performs read or write access in the Y direction after the Z actuator makes the sled contact the probe tips. We have used a "streaming" data layout model for the MEMS-based storage, i.e., all the data are placed continuously on the media sled. This data layout policy is an integral part of the validated MEMS modules in *DiskSim 3.0*. Figure 3 depicts the data organization for these MEMS storage device modules. Therefore, unlike the work in [8] and [10], the layout policy makes no attempt to map disk-based device characteristics, i.e. cynliders, tracks and sectors, to the MEMS probe tip sweep area. Since our evaluation considers the SDs and SMs to be distinct storage devices, no such mapping is deemed necessary.

3.2.3 Simulation Models

To help conduct our experiments, a simulation environment was developed to model the SD and SM architectures. It is capable of simulating the computation, communication and I/O behaviors given different system parameters, such as disk and MEMS modules. The simulation environment uses

Table 2. Parameters for HP C2490A

Parameter	Value
RPM	6,400
max bandwidth (MB/s)	10
avg seek time R/W (ms)	9.2/17
number of data surfaces	18
number of cylinders	2630
sector size (bytes)	512
diameter of disk	3.5"
height of disk	1.63"

the *DiskSim 3.0* simulator to estimate the cost of accessing disk-based or MEMS-based devices. Computation and communication times are directly measured by performing the actual tasks as required by the workload on synthetic data in order to provide accurate measurement for all three components of the total execution time. This simulation environment processes the workload and takes measurements for the computation, communication and I/O times, where message passing protocol (MPI) is used for communication among the SDs or SMs.

4 Benchmark Workloads

We selected basic database *scan*, *join*, *sort*, association rules mining, and high-dimensional data clustering for our evaluation. We also adopted the application-level processing algorithms and input data as described in [9]. Since we are interested in comparing between disk-based and MEMS-based storage, we applied equal amount of the workloads on both the SD and SM architectures.

4.1 Database Scan, Join, and Sort

We offloaded *scan*, *join* and *sort* to the SDs and SMs, and assumed that database tables are evenly distributed across all SDs/SMs initially. Each SD or SM executes the same code on its own local data, which is sent by the client during system initialization.

The *scan* primitive sequentially retrieves and evaluates each tuple in the locally stored data. Results from *scan* are sent back to the remote client from the SDs or SMs. The *sort* implements a parallel out-of-core sampling bucket sort and consists of 4 stages: sampling, re-distribution, internal sort, and external merge. The *join* utilizes 2 SDGs or SMGs to perform a database join using a nested loop-based algorithm. While *scan* involves only local storage access, *join* and *sort* could require communication between individual SDs/SMs or SDGs/SMGs, or both.

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

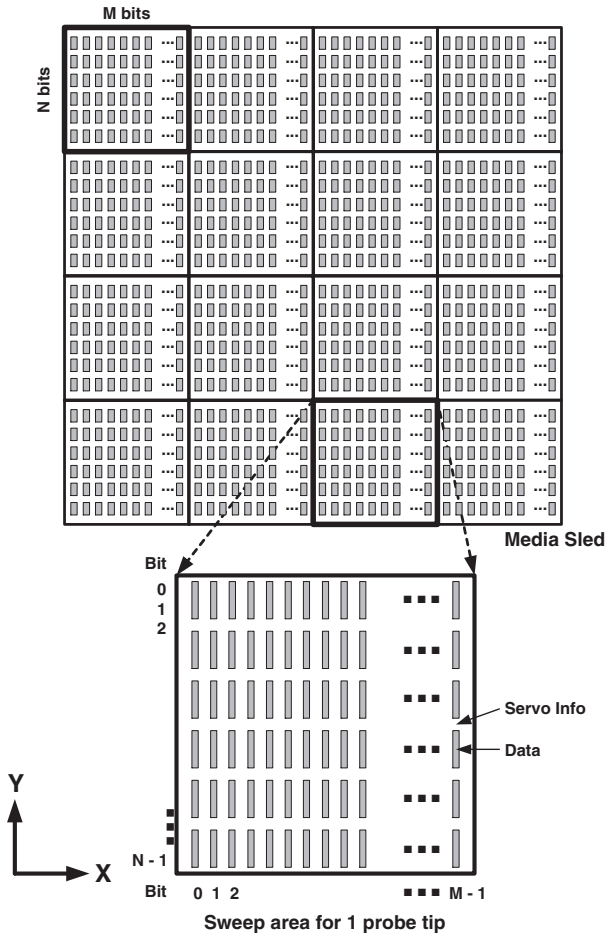

Figure 3. Data organization for MEMS-based storage devices

4.2 Association Rules Mining (ARM)

Association rules mining (or ARM) has historically been employed in the so-called *Market-Basket Analysis*. Given a collection of transactions T and each transaction contains items from an itemset I, an association rule takes on the expression $X \Rightarrow Y$, where $X \subseteq I$ and $Y \subseteq I$. An association rule $X \Rightarrow Y$ holds in T with a confidence C and a support S, if at least $C\%$ of all the transactions (in T) containing X also contain Y, and $X \Rightarrow Y$ exists in at least $S\%$ of all the transactions in T [11] [17]. An ARM process consists of two major phases. In the first phase, a set of frequent itemsets is found. Then, in the second phase, the rules that satisfy the minimum support S and the minimum confidence C are identified. The first phase, which performs frequent set counting (FSC), tends to be much more expensive than the second phase, since I usually contains a large number of distinct items. Based on the work proposed in [11], we developed a FSC primitive to evaluate our SD and SM systems. The Count Distribution (CD) method uses the *Apriori* algorithm proposed in [6], while the Hybrid Distribution (HD) method partitions the frequent itemsets into sufficiently large grids (or sections), then assigns a set of SDs/SMs to each section.

4.3 MAFIA Data Clustering

Data clustering techniques help discover the interesting patterns (also called clusters) from large data sets. These patterns often exist in high-dimensional data space. An efficient data clustering technique will address both data and noise existent in high-dimensional spaces and subspaces, which could result in an exponential growth of the search space for clusters. Our data clustering primitive is based on the parallel adaptive-grid and density-based algorithm (hence MAFIA, standing for Merging of Adaptive Finite Intervals Algorithm) proposed in [16]. The primitive computes the histogram for each attribute, adaptively setting the bin size, then builds candidate dense cells and performs subspace clustering on the local records to compute local dense cells. Finally, reduction is performed to obtain global data on the histogram, dense cells, and bounds for the dense cells. Out-of-core I/O access strategy is used to model our SD and SM systems.

5 Experiments and Results

We evaluated the performance of disk-based distributed storage and that of MEMS-based storage, using *scan*, *join*, *sort*, association rules mining, and high-dimensional data clustering as the workloads. Our simulation platform consists of 32 Pentium III Linux PCs running at 500 MHz and each with up to 512 MB of physical memory and 9 GB of disk storage. The PCs are interconnected with fast Ethernet to model a low-latency high-bandwidth network such as IBA. Thus, the embedded processor is assumed to have a 500 MHz clock speed, and a 64MB maximum on-device memory is configured for our simulation environment. For *scan*, *join* and *sort*, we used TPC-H database generator [20] to populate synthetic data into tables with a scale factor of 1.0, denoting a 1 GB size for the populated data (for the *Lineitem* table). Data generators for producing compatible synthetic input data were used for the ARM FSC and MAFIA (clustering) workloads.

5.1 Database Scan, Join, and Sort

The *scan*, *join* and *sort* primitives discussed in Section 4.1 were evaluated for disk-based systems (represented by the HP C2490A) and MEMS-based systems (represented by the G1, G2 and G3 devices). For all the storage systems under test, the simulation environment was configured to

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

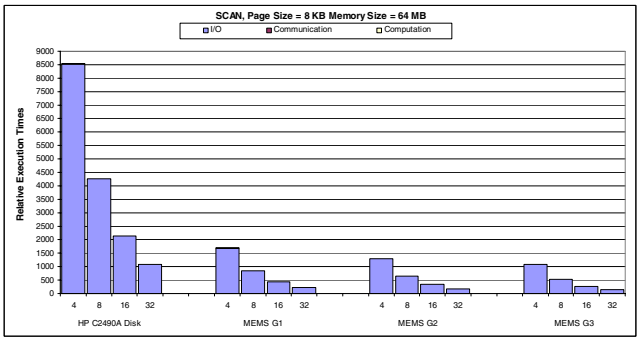

Figure 4. Performance of the *scan*. Each group of four bars represents the respective SD or SM system of sizes 4, 8, 16 and 32.

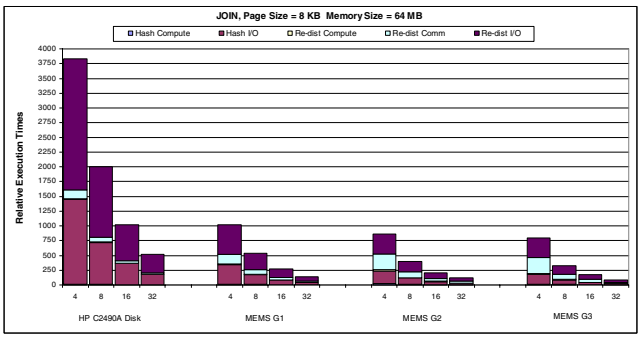

Figure 5. Performance of the *join* during the hash and re-distribution stages for the SD and SM systems of sizes 4, 8, 16 and 32.

use 8 KB as the data page size and 64 MB of on-device memory, a scale factor of 1.0 for the TPC-H data, and the other platform parameters as specified in this section.

Figure 4 presents the performance results of *scan* on four different systems constructed with disks and MEMS G1, G2 and G3, respectively. For each system, the group of four bars in the Figure represents the total response times for system sizes of 4, 8, 16 and 32, from left to right as marked. These results show that for each system, increasing the system size reduces the total response time, due to increased parallelism available from the larger system. Looking across the four systems, we also observed that MEMS-based systems exhibit higher performance than the disk-based system, with the G3 system providing the overall smallest total response time for any given system size. Although the use of MEMS-based devices improved system performance, *scan* remained an I/O-bound workload, i.e., the performance of *scan* is dominated by the I/O time. This characteristic of *scan* is the main motivation for offloading *scan* to the storage device in [9] and in this work.

The performance results of *join*, arranged similarly as those for *scan*, are shown in Figures 5 and 6. There are 3 main stages in *join*: hash, re-distribution, and local nested-loop join. Due to the inefficiency of the last stage, as reported in [9], *join* is considered I/O-bound. However, this problem can be remedied if the nested-loop is enhanced with more efficient algorithms. Figure 6 shows the I/O dominance exhibited by the last stage of *join*, particularly when a disk-based system of smaller size, e.g. 4 SDs, is employed. The I/O cost diminishes rather quickly both as we increase the system size, and as we switch from disk-based systems to MEMS-based systems. Given the best-performing system size for all the systems in this study, i.e. 32 SDs or 32 SMs, the I/O and computation costs by the last stage of *join* are shown in Table 3. Note that our 32-SM G3 storage reduced the I/O cost by more than 10 folds,

Table 3. I/O vs. computation times: local nested-loop join (system size: 32 SDs/SMs)

Storage Device	I/O (sec)	Computation (sec)
HP C2490A	8207.20	2.03
MEMS G1	1656.78	2.13
MEMS G2	1097.25	2.54
MEMS G3	812.04	2.05

compared to the HP C2490A disk-based system.

While overall time reductions are clearly visible as our storage systems changed from disk-based to MEMS-based, there exist other changes in the characteristics of *join* when MEMS-based storage devices are employed instead of disks. The G2 MEMS-based system with 4 SMs recorded a hash I/O time of 219.45 sec and a re-distribution communication time of 272.59 sec. Likewise, another G3 MEMS-based system with 16 SMs incurred a hash I/O time of 41.96 sec and a re-distribution communication time of 50.77 sec. These compared with the disk-based system which always exhibited a difference in the range of one order of magnitude between the large hash I/O and small re-distribution communication times. The implication of this could result in potentially significant change in characteristics of workloads that contains *join* as a component, e.g. full TPC-H queries. The use of MEMS-based storage devices is a major reason for such changes in the workloads, which in our study resulted in certain network-limited stages of the processing becoming dominant over other I/O-bound stages. While this dominance may be small compared to other stages of the processing, such effect would not have existed with disk-based storage systems.

Figure 7 presents our results of executing *sort* on the SD/SM systems. In the case of *sort* the proportions of I/O vs. the other two components (i.e. computation and

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

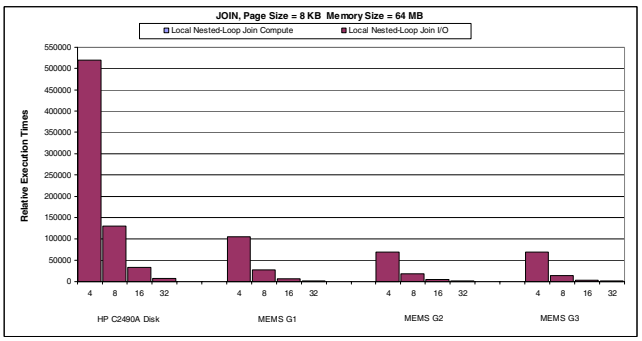

Figure 6. Performance of the *join* during the local nested-loop join stage for the SD and SM systems of sizes 4, 8, 16 and 32.

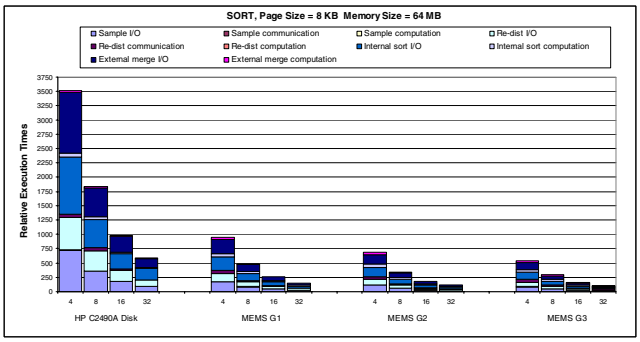

Figure 7. Performance of the *sort* for the SD and SM systems of sizes 4, 8, 16 and 32 throughout all the processing stages.

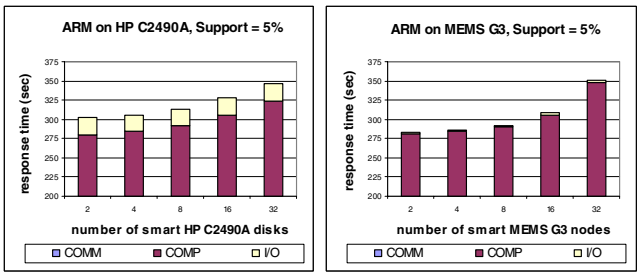

Figure 8. Performance of ARM on the SD and SM systems with the HD method

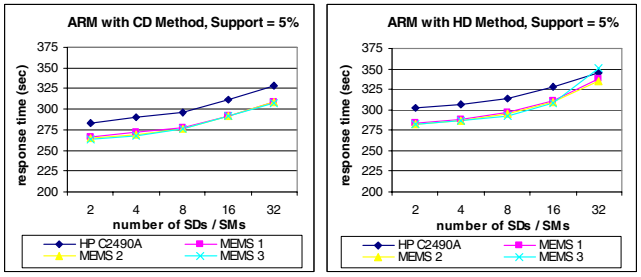

Figure 9. Scalability of ARM with the CD and the HD methods

communication) changed from disk-based to the MEMS-based systems. Table 4 shows some such changes in proportions during various stages of *sort*. For example, during the sampling stage, both disk-based and MEMS-based "size-32" systems incurred virtually identical cost in communication, yet the ratio of I/O vs. communication differs significantly between the two systems, i.e. 90.09/0.22 as compared to 10.53/0.21 due to the reduced I/O cost. While no changes similar to those observed in *join* were noted, the results for *sort* could still impact database-related workloads, since data sizes often vary both statically and dynamically in more complex workloads, such as the TPC-H full queries in commercial DSS systems.

5.2 Association Rules Mining

The frequent set counting (FSC) primitive and synthetic transaction database as discussed in [9] were used. The transaction database had an average transaction size of 20 with each transaction containing up to 25 distinct items. 100,000 transactions were generated for each SD/SM, and the minimum support is 5%. Both the CD and HD methods were implemented. The default platform parameters were used, including the 8 KB page size and 64 MB of on-device memory.

Figure 8 illustrates the compute-bound characteristic of the ARM FSC workload, using the HD method. Although the I/O component of the total response time decreased from disk-based system to G3 MEMS-based system, computation time continued to dominate the overall performance. These results are consistent with our expectation that FSC is a compute-intensive operation. Thus, while the use of MEMS-based storage helped reduce the I/O cost, the on-device embedded processor remains the deciding factor in achieving better overall performance.

Figure 9 demonstrates that the FSC primitive exhibits scalable performance on the disk-based as well as MEMS-based systems with increasing data size. Note that since 100,000 transactions were generated for each SD/SM, data size was increasing linearly with respect to the number of SDs/SMs. Therefore, keeping the response times constant demonstrates scalability. We observed similar scalability behaviors by both the disk-based and MEMS-based architectures, with disk-based systems taking larger response times primarily due to their higher I/O costs.

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

Table 4. I/O vs. computation and communication times (in sec) for *sort* stages

System, Size	Phase	I/O	Compute	Communication
HP C2490A, 4	Re-distribution	573.77	–	49.18
MEMS G3, 4	Re-distribution	72.58	–	49.02
HP C2490A, 8	Internal Sort	501.43	37.65	–
MEMS G2, 8	Internal Sort	77.35	37.87	–
HP C2490A, 4	External Merge	1060.52	36.17	–
MEMS G3, 4	External Merge	121.50	35.62	–
HP C2490A, 32	Sampling	90.09	–	0.22
MEMS G3, 32	Sampling	10.53	–	0.21
HP C2490A, 16	Re-distribution	195.07	–	11.72
MEMS G2, 16	Re-distribution	32.14	–	10.91

5.3 High-Dimensional Data Clustering

The MAFIA primitive discussed in Section 4.3 was executed over a data set of 4,281,782 records (> 342 MB input data size), which contains 20-dimensional data with 5 clusters and each cluster is of 5 dimensions. We measured the performance of MAFIA with system sizes of 1, 2, 4, 8, 16 and 32 for both the SD and SM architectures, using data page sizes of 8K, 16K, 32K, and 64K bytes, and 64 MB of on-device memory. While 8 KB page size represents the default for a PostgreSQL database, we also simulated with the larger page sizes to explore future and more aggressive architectures.

Figure 10 presents the performance results of MAFIA with data page size fixed at 8 KB. The results indicate that both the computation time for populating candidate dense cells and the I/O time required for data access decreased when the number of SDs/SMs was increased. Furthermore, we observed a noticeable change in the workload characteristics between the two systems– MAFIA changed from being I/O-dominant to compute-dominant. In other words, when MEMS-based devices are employed in lieu of disk-based devices, MAFIA is transformed from I/O-bound to compute-bound. Comparisons conducted with other page sizes reflected the same characteristic change of MAFIA. Note that we have used a 500 MHz clock speed for the on-device embedded processor for the SDs/SMs, representing a relatively advanced performance level for an embedded processor given current state of technology.

The scalability of MAFIA was also evaluated for all four types of SD/SM systems, with data page sizes including 8K, 16K, 32K and 64K bytes. Based on the results in Figure 11, we observe improved performance as we increased the data page size, and similar scalability behavior between the three MEMS-based systems. It is noted, however, that the disk-based architecture tends to provide better scalability at smaller system sizes, i.e., between 1 and 4, despite its lower

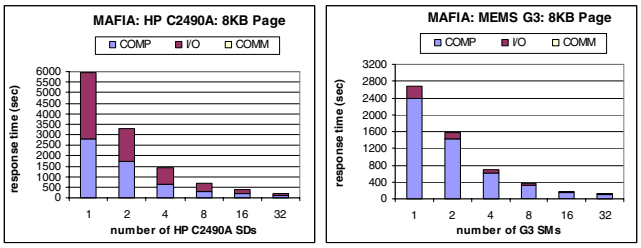

Figure 10. MAFIA performance on the SD and SM (MEMS G3) systems

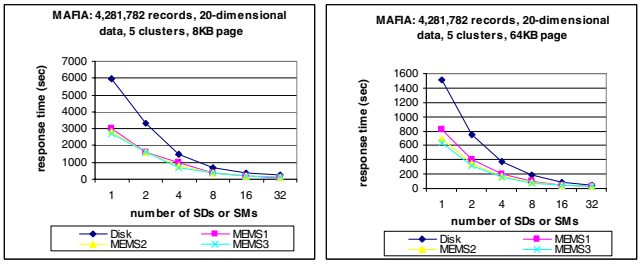

Figure 11. MAFIA scalability with 8KB and 64KB page sizes

performance than the MEMS-based architecture. When system size exceeds 8, this effect diminishes. Although MEMS-based devices differ significantly from disks, both our SD and SM storage systems exhibited similar behaviors in scalability.

5.4 Summary of Workloads

Table 5 summarizes the characteristics of our workloads as processed on the SD and SM systems. We observed that as the underlying storage system shifted from disks to MEMS-based devices, *scan* remained I/O-bound, *join* and

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

Table 5. Workloads on the SD/SM systems. I: I/O-bound, C: compute-bound, N: network-limited

Workload	Disk-Based	MEMS G1	MEMS G2	MEMS G3	Comments on Workload Characteristics
scan	I	I	I	I	remains I/O-bound for both disk-based and MEMS-based storage systems
join	I	I,N	I,N	I,N	I/O-bound for disk-based storage system, but network-limited during re-distribution phase if efficient algorithm is used in lieu of local NL
sort	I	I	I	I	I/O-bound for disk-based storage system, but ratios of I/O vs. computation and communication decreased significantly for MEMS-based systems
ARM FSC	C	C	C	C	remains compute-bound for both disk-based and MEMS-based storage systems
Clustering	I	C	C	C	I/O-bound for disk-based storage system, but compute-bound for MEMS-based storage systems

sort exhibited potential to transform complex queries from I/O-bound to either compute-bound or network-limited. The ARM FSC workload remained compute-bound, and clustering (MAFIA) changed from being I/O-bound to compute-bound. These characteristic changes suggest the need for improved storage architectures to address the new and specific performance bottlenecks of a workload.

6 Conclusion and Future Work

MEMS-based storage devices, with their differing two-dimensional way of holding and accessing data, have significantly changed the characteristics of I/O-intensive workloads compared to those processed with disk-based storage. The work in [9] demonstrated that for disk-based storage systems, offloading user-level processing on to the storage device improved overall performance. Evaluation conducted in this work extended the concept to emerging MEMS-based storage based on the pioneering research in [8], [10], [19], [21] and [22]. Exploiting excess processor cycles available from the embedded peripheral systems (such as storage devices) by taking application processing closer to data has been a trend in computer architecture. Both the SD and SM models investigated in this work are a continuation of this trend. Our experiments demonstrated that MEMS-based storage can provide significant performance improvement over traditional disk-based systems for several representative workloads while exhibiting similarly desirable scalability. These results suggest that MEMS-based storage can be a highly viable option for large data- and I/O-intensive applications.

Given the rapid advancement of MEMS technology, it is conceivable that an integrated MEMS-based processor-memory-storage chip could soon be realized. With such a hardware platform, the processing models introduced in [9] and evaluated in this work are expected to further improve the performance of large data-intensive workloads, altering their basic characteristics in many cases. Although MEMS-based storage devices occupy a higher level than disks in the cost structure of storage media, the performance improvement provided by MEMS-based devices makes them a desirable alternative when configuring high-performance storage systems. When considering the design of offloading user-level processing, the feasibility of computation-storage integration of MEMS-based devices becomes increasingly attractive compared to disks.

Future work includes complex queries, e.g. TPC-H full queries built on the basic database primitives. Besides performance-related issues, our work-in-progress items also include availability and reliability schemes, since these are no less critical than performance designs for large data-intensive distributed storage systems. Of significant value is the possibility of offloading I/O scheduling optimization to the device, especially for MEMS-based storage. The design of hybrid disk- and MEMS-based storage architectures adds yet another dimension to current processing models.

Acknowledgment

The authors would like to thank Dr. Ganger of Carnegie Mellon University for his advice on the *DiskSim* simulator. We also acknowledge the use of the Chiba City Linux PC cluster at Argonne National Laboratory.

References

[1] Carnegie Mellon University CHIPS Research Project URL: http://www.lcs.ece.cmu.edu/research/MEMS.

[2] Agilent Technologies Foundation Research Program URL: http://www.labs.agilent.com/research/foundation.html.

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

[3] IBM Corp. Millipede MEMS Research Program URL: http://www.zurich.ibm.com/st/mems/millipede.html.

[4] Kionix Inc. URL: http://www.kionix.com/knx101.html.

[5] Nanochip Inc. URL: http://www.nanochip.com.

[6] R. Agrawal and J. Shafer. Parallel Mining of Association Rules. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):962–969, December 1996.

[7] J. S. Bucy and G. R. Ganger. *The DiskSim Simulation Environment Version 3.0 Reference Manual*. Carnegie Mellon University, January 2003.

[8] L. R. Carley, G. R. Ganger, and D. F. Nagle. MEMS-Based Integrated-Circuit Mass-Storage Systems. *Communications of the ACM*, 43(11), November 2000.

[9] S. Chiu, W. Liao, and A. Choudhary. Design and Evaluation of Distributed Smart Disk Architecture for I/O-Intensive Workloads. *Lecture Notes in Computer Science*, 2660:230–241, July 2003.

[10] J. L. Griffin, S. W. Schlosser, G. R. Ganger, and D. F. Nagle. Modeling and Performance of MEMS-Based Storage Devices. *ACM SIGMETRICS Performance Evaluation Review*, 28(1):56–65, June 2000.

[11] E. Han, G. Karypis, and V. Kumar. Scalable Parallel Data Mining for Association Rules. *IEEE Transactions on Knowledge and Data Engineering*, 12(3), May 2000.

[12] InfiniBand Trade Association (IBTA). *InfiniBand Architecture Specification Volume 1 Release 1.0.a*, June 2001.

[13] K. Keeton, D. A. Patterson, and J. M. Hellerstein. A Case for Intelligent Disks (IDISKs). *SIGMOD Record*, 27(3), September 1998.

[14] T. Madhyastha and K. Yang. Physical Modeling of Probe-Based Storage. In *Proceedings of the IEEE Symposium on Mass Storage Systems and Technologies*, April 2001.

[15] G. Memik, M. Kandemir, and A. Choudhary. Design and Evaluation of Smart Disk Architecture for DSS Commercial Workloads. In *Proceedings of the International Conference on Parallel Processing*, September 2000.

[16] H. Nagesh, S. Goil, and A. Choudhary. Parallel MAFIA: Parallel Subspace Clustering for Massive Data Sets. *Data Mining for Scientific and Engineering Applications*, March 2001.

[17] S. Orlando, P. Palmerini, R. Perego, and F. Silvestri. An Efficient Parallel and Distributed Algorithm for Counting Frequent Sets. In *Proceedings of VECPAR 2002 High Performance Computing for Computational Science*, June 2002.

[18] E. Riedel. *Active Disks - Remote Execution for Network-Attached Storage*. PhD thesis, Carnegie Mellon University, November 1999.

[19] S. W. Schlosser, J. L. Griffin, D. F. Nagle, and G. R. Ganger. Designing Computer Systems with MEMS-based Storage. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, January 2000.

[20] Transaction Processing Performance Council. *TPC Benchmark H Standard Specification Revision 1.4.0*, June 2001.

[21] H. Yu, D. Agrawal, and A. Abbadi. Tabular Placement of Relational Data on MEMS-based Storage Devices. In *Proceedings of the 29th Conference on Very Large Databases*, September 2003.

[22] H. Yu, D. Agrawal, and A. Abbadi. Towards Optimal I/O Scheduling for MEMS-based Storage. In *Proceedings of the IEEE Goddard Conference on Mass Storage Systems and Technologies*, April 2003.