USENIX Association

Proceedings of FAST '03: 2nd USENIX Conference on File and Storage Technologies

San Francisco, CA, USA
March 31–April 2, 2003

Optimizing Probe-Based Storage

Ivan Dramaliev
Department of Computer Science
Tara Madhyastha
Department of Computer Engineering
University of California Santa Cruz
1156 High Street, Santa Cruz, CA 95064
{ivan,tara}@soe.ucsc.edu

Abstract

Probe-based storage, also known as micro-electric mechanical systems (MEMS) storage, is a new technology that is emerging to bypass the fundamental limitations of disk drives. The design space of such devices is particularly interesting because we can architect these devices to different design points, each with different performance characteristics. This makes it more difficult to understand how to use probe-based storage in a system. Although researchers have modeled access times and simulated performance of workloads, such simulations are time-intensive and make it difficult to exhaustively search the parameter space for optimal configurations. To address this problem, we have created a parameterized analytical model that computes the average request latency of a probe-based storage device. Our error compared to a simulated device using real-world traces is small (less than 15% for service time). With this model we can identify configurations that will satisfy specific performance objectives, greatly narrowing the search space of configurations one must simulate.

1 Introduction

In the last 20 years microprocessor speeds have been improving by 50% to 100% per year [12] and memory capacities have been increasing by 60% per year [8]. Unfortunately, secondary storage has not kept pace. There are several limitations for today's disk technologies, such as disk rotational speed, bit density (which is limited by the superparamagnetic effect [23, 4]) and read-write head technology [23]. Academia and industry are developing new technologies to bypass these limitations. These technologies include holographic storage [14, 20, 23], atomic force microscopy (AFM) [11, 5, 23], and MEMS storage [3, 2, 6, 23]. Each of these storage alternatives has inherent tradeoffs that govern its use in a system. In this paper, we focus on the performance characteristics of one specific new technology: MEMS storage. MEMS storage technology is based on an array of atomically-sharp probe tips that read and write data on the storage medium. While several alternative styles of MEMS storage are currently being explored, all of these anticipated devices share certain common characteristics: they support high throughput, high parallelism and high density. Data is accessed in a disk on a rotating media platter, while in MEMS storage the media does not rotate but moves in a rectilinear fashion. The design space of MEMS storage is particularly interesting because we can architect these devices to different design points, each with different performance characteristics. This makes it more difficult to understand how to use probe-based storage in a system. Exhaustive simulation is at best extremely time-consuming and at worst impossible, depending on the search space. In our experiments it took approximately 20 minutes on an Intel Pentium 3 500MHz machine to run the simulation of a workload using a single configuration. The design space that were concerned with included over a million configurations, and it would take 14 days on a 1000 node cluster to test them all. To address this problem, we have created a parameterized analytical model that computes the average request latency of a MEMS storage device. Our error compared to a simulated device using real-world traces is small (within 15% error for service time). We used this model to identify optimal configurations given such constraints as capacity and throughput.

The remainder of this paper is organized as follows. In Section 2 we describe related work. We present architecture and design layout of a MEMS storage device in Section 3. In Section 4 we derive the equations governing the service time of a MEMS storage device under uniformly distributed accesses. We use this model to identify optimal configurations subject to user-specified constraints in Section 5. We conclude in Section 6.

2 Related Work

Current projects on probe-based storage include those by the Carnegie Mellon Center for High Integrated Informa-

tion Processing and Storage Systems (CHI^{2}PS^2 [3]), Despont *et al.* [25], Mamin *et al.* [11], and the Atomic Resolution Storage (ARS) project at Hewlett-Packard Laboratories [23]. While these projects use different recording technologies, they are all based on tip arrays and media sleds. Thus, the models we describe can be tuned with different input parameters to describe these systems.

Disk modeling has been traditionally used to design storage systems that use hard drives, and a complete survey of this work is beyond the scope of this paper. Because disk performance is so workload-dependent, most simplifying assumptions cause large errors [16]. Shriver [19] developed some analytic models to incorporate effects of disk caching and I/O workload variation.

Because probe-based storage devices do not yet exist, it is particularly useful to create models of them that can yield insights into performance. Yang and Madhyastha created several physical models for seek time of probe-based storage, defining a performance range for a specific hardware configuration [10]. A different model was presented by Schlosser and Griffin *et al.* [6, 17], who conclude that a probe-based storage device can improve applications performance by a factor of three over disks. They compare and contrast probe-based storage and traditional disk drives, and study how aspects of the operating system need to be changed when a system is built with probe-based storage [7]. Uysal *et al* [24] evaluate several hybrid MEMS/disk architectures, showing that hybrid architectures can give performance benefits similar to replacing disks with probe-storage devices (at lower price-performance). Ying *et al* [9] used this seek-time model to devise policies for power conservation. However, these studies all rely upon trace-driven simulation of traces. In contrast, our focus here is to develop an analytical model for a wide range of probe-based storage characteristics that can be used for such performance research.

Probe-based storage devices are much faster than traditional disk drives, making the question of how probe-based storage may be integrated into the memory hierarchy very important. Griffin *et al* [18] show that using probe-based storage as a disk replacement will improve overall application runtime by a factor of 1.9–4.4, and when used as a disk cache can improve I/O response time by up to 3.5 times.

3 Probe-Based Storage

Probe-based storage devices have a wide range of configurable parameters. In Section 3.1 we describe how a probe-based storage device works and highlight the parameter space. In Section 3.2 we show how data may be stored on such a device.

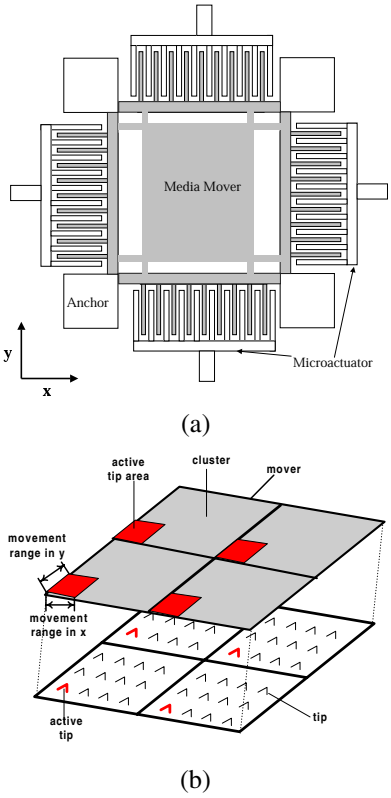


Figure 1: (a) Mover and microactuators. The device is shown from above; the tip array lies below the mover surface. (b) A probe-based storage device that includes one mover (the shaded areas) above a tip array. The mover is divided into four clusters, each of which has 12 dedicated tips. One tip per cluster is active at a time (the shaded tips), accessing the tip areas depicted by the dark rectangles. The mover has limited range of movement over the tip array in the X and Y directions.

3.1 Architecture

Figure 1a is a top view of a typical probe-based storage device. In this figure, the shaded parts move and the unshaded parts are stationary. The media *mover* is suspended above a surface on which a grid of many probe *tips* are embedded. Collectively, the tip array is the logical equivalent of the read/write heads of a traditional disk drive. Voltage applied to the fingers of the microactuator combs exerts electrostatic forces on the mover that cause it to move in the X and Y directions, overcoming the forces exerted by the anchors and beams that keep it in place. To service a read or write request, the mover first repositions itself so that the tip array can access the required data. This repositioning time is called *seek time*. The mover then accesses the data while moving at a constant velocity in the Y direction, incurring *transfer time*.

Configurable parameter	Default value	Symbol
Movement range		
in X (5–80μm)	40μm	δ_x
in Y (5–80μm)	40μm	δ_y
Active tips (10–2560)	320	T_{active}
Tips in Y		
per cluster(1-20)	10	T_y
Physical parameter		
Settle time	200μs	t_{settle}
X-move time	1ms	t_{XM}
Velocity	0.05m/s	v_0
Turn around time	400μs	t_{TA}
Tip change time	0	t_{TS}
Bit width	50nm	d_b
Acceleration	250m/s^2	a_0
Active tips per cluster	1	$T_{cluster}$
Workload parameter		
Average request size	calculated from workload	r
Runlength	calculated from workload	r_l

Table 1: Configurable and physical probe-based storage architecture parameters.

Figure 1b magnifies the mover area from Figure 1a. This shows that a mover is divided into one or more *clusters*. Each cluster can read data independently of the others, which provides higher parallelism to the device. As an example, the mover in Figure 1b is subdivided into four clusters. Each cluster is a media area that is accessed by many tips, only one of which can be active at a time. Using several tips in parallel, one from each cluster, compensates for the low data rate of each individual tip, which is on the order of 1Mbit/sec. The number of bits accessed simultaneously is equivalent to the number of clusters per mover times the number of movers in the device. We call this the number of *active tips*.

The mover's range of movement and the bit size determine the amount of data that can be manipulated by one tip, or *tip area*. Because several tips are active at a time, different tip areas of the mover are manipulated simultaneously, as depicted by the shaded rectangles in Figure 1b. Different areas of the mover are accessed by switching between sets of active tips.

Many architectural configurations are possible for probe-based storage. For example, we might vary the number of active tips, the mover's movement range, the media density, and so on. The values summarized in Table 1 are reasonable defaults for probe-based storage devices [25, 6, 1], and are the parameters we used in simulations unless otherwise specified.

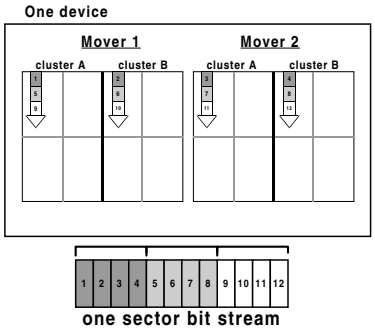


Figure 2: Example data layout over two movers in one MEMS device.

3.2 Data Layout

Traditional disk data layout minimizes seek time and rotational latency. Analogously, we chose a layout for probe-based storage that has a similar sequential ordering to minimize movement for consecutive requests accessing sequential data.

Each mover is divided into several clusters, each of which is an area covered by a grid of tips. For each cluster, only one read/write tip can be active at a time. This cluster design helps to minimize power consumption while increasing the size of the swept area. Given this parallel architecture, where several tips can read/write data in parallel, it was reasonable to stripe the data between the clusters. We chose to work with bit-level striping because it maximizes the throughput when there is only one outstanding request.

Figure 2 shows the layout of a MEMS device with two movers, two clusters per mover and four tips per cluster. This makes the tip area a quarter of the cluster's area. One tip per cluster may be active at a time; therefore, in this device, four tips are active at a time. Consequently, a sector of data is read/written by the four active tips simultaneously. The sector is bit-interleaved between the two clusters, as depicted by the pattern of the bitstream in Figure 2. The data is accessed while the mover is moving in the $\pm y$ direction. Figure 3 shows how data is accessed on a single cluster.

A single device may contain several movers, increasing parallelism. To exploit this secondary level of parallelism, we stripe data bitwise among all the clusters of these movers. For example in Figure 2 there are four simultaneously active tips, one per cluster, and the bitstream

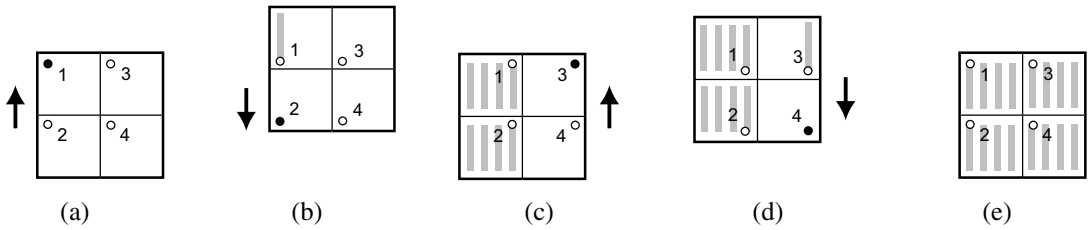


Figure 3: Data access for a single mover with four clusters and one tip per cluster. (a) Tip 1 is activated and the mover slides up, reading the column of bits the tip passes over. (b) Tip 2 is activated after the shaded bits have been read, and the mover reverses direction. (c) After all columns have been read, with an x-move between each column to reposition the tips over the new column of bits, tip 3 is activated and the mover slides up, analogously to (a). (d) Tip 4 is activated and the mover reverses direction, analogously to (b). (e) All bits in the mover are read.

is striped among them all. First the stream is split into groups of four bits each, as depicted by the different gray shades. Then each such group of four bits is interleaved between the four active tips, as shown by the arrows in Figure 2.

4 Service Time Model

The performance of a particular probe-based storage device depends on its configuration and the workload that it is subject to. We can measure performance using a model of probe-based storage implemented in Pantheon, an I/O device simulator created by Hewlett-Packard [26, 16]. We used this model to explore the behavior of probe-based storage under several workloads and identify a configuration with performance characteristics similar to disks [21]. However, the number of different configuration parameters is too large to use such a simulator to explore the design space exhaustively.

To address this problem, we created an analytical model that predicts the response time for various configurations. We make the simplifying assumption that requests are uniformly distributed across the device. This assumption does not hurt us as significantly as it would with disk drives because seek time is dominated by transfer time except at large degrees of parallelism [22]. Furthermore, as shown in [21], seek time is not as sensitive to other architectural parameters as transfer time.

4.1 Performance Dependencies

Based on the architecture and the layout model described in Section 3, we know that the design space of probe-based storage is relatively complex and involves many parameters such as the number of active tips per device, mover's movement range, and acceleration. To choose one configuration over another, we must understand how the physical configuration affects the performance. Therefore, we analyze the dependencies in the model between these parameters and the service time, which we wish to minimize.

Figure 4 is a dependency graph showing how performance is related to probe-based storage parameters [21]. The graph edges represent the dependencies, the leaf nodes are parameters, and the remaining nodes are intermediate variables. The dependencies in this graph reflect the parameters and layout presented in Section 3, although other models could be used both for the layout and for the physical behavior of the device [10]. The target in the graph is the service time, which is the sum of seek time and transfer time, as shown in Equation 1. We model each of these components separately:

$$t_{service} = t_{seek} + t_{transfer} \quad (1)$$

Our model for seek time, described in Section 4.1.2, assumes uniformly distributed requests. To adjust for sequentiality in workloads, we parameterize Equation 1 with the runlength calculated from the trace. To compute the runlength, we calculate the length, in bytes, of each sequential run in the trace and average these runs to compute the runlength in bytes, r_l.

Only the first request of a sequential run will require repositioning the mover, incurring seek time. Thus the ratio of average request size (r) to runlength (r_l) represents the fraction of requests that require seeks. We modified the service time equation (1) to use the runlength and obtained Equation 2:

$$t_{service} = \frac{r}{r_l} t_{seek} + t_{transfer} \quad (2)$$

4.1.1 Transfer Time Model

Transfer time ($t_{transfer}$), the time to actually read/write the data, has four components. The first is the time to read

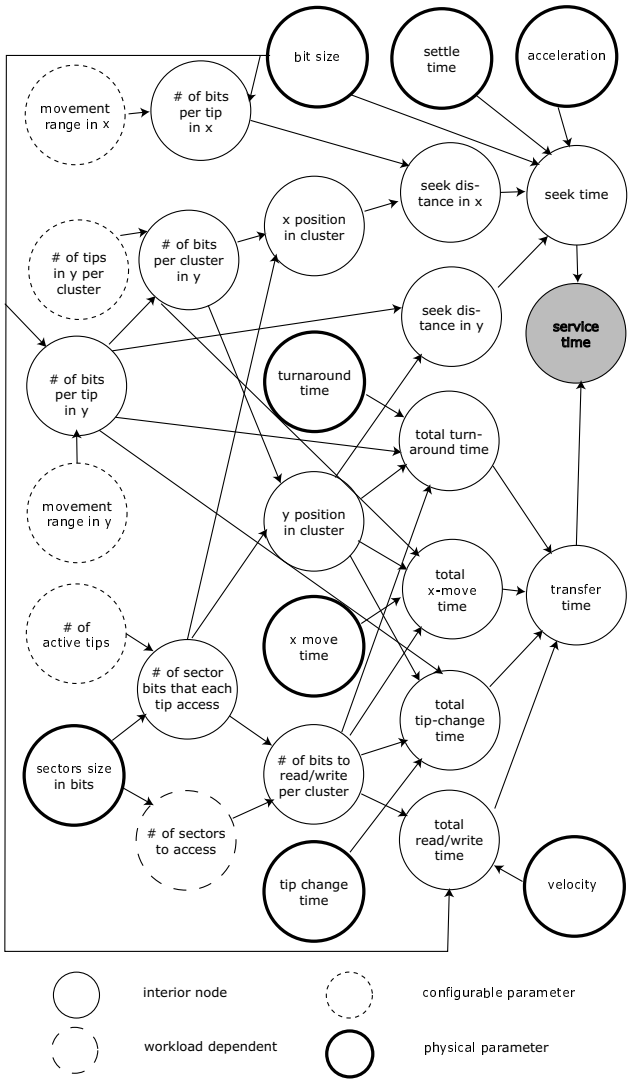


Figure 4: A simplified design space parameter dependency graph. The head of the graph represents the service time, which is the minimization target. From the graph we can identify the parameters (represented by the leaves) on which service time depends.

or write the data by moving the mover over the tip array with a constant velocity (*read/write* time, or t_{RW}). Second, if a request spans more than one bit column, we must add the time that it takes for the mover to reverse direction (e.g., from moving down to moving up); we call this time the *turnaround* time (t_{TA}). To minimize wasted space, a sector can begin at any bit within a column, and continue at the next column. The MEMS device controller is responsible for tracking the starting positions of sectors. Thus, a single read/write request may require one or more bit column movements in the X direction. This third component is called an *x-move* (t_{XM}). The last component is the time that it takes to switch between sets of active tips, or *tip switch* time (t_{TS}).

Thus, the transfer time is a sum of four terms: n_{TA}, the number of turnarounds, multiplied by t_{TA}, the turnaround time; n_{TS}, the number of tip switches times t_{TS}, the tip switching time; n_{XM}, the number of moves in X, times t_{XM}, the time to move one bit in X, and t_{RW}, the time it takes to actually read the data. Equation 3 shows this combination.

$$t_{transfer} = n_{TA} \times t_{TA} + n_{TS} \times t_{TS} + n_{XM} \times t_{XM} + t_{RW} \quad (3)$$

As described in Section 3.2,the data to be read or written is divided among all active tips, which work in parallel. Each tip reads r_f bits, which is equal to the number of bits per request ($8\ r$) divided by the number of active tips T_{active}, as shown in Equation 4.

$$r_f = \frac{8r}{T_{active}} \quad (4)$$

The read/write time, t_{RW}, depends on the number of bits r_f that each active tip has to read, the velocity of the device (v_0), and the bit width d_b (to translate the number of bits into distance). To read or write each bit, it will take d_b/v_0 time for each active tip to move over it. This relationship is given in Equation 5.

$$t_{RW} = \frac{r_f \times d_b}{v_0} \quad (5)$$

The number of turnarounds, X-moves, and tip changes depend on the number of requested bits that each active tip accesses, the starting position, and the the number of bits in a bit column ($n_{bitsinY}$). To calculate this quantity we divide the movement range in Y by the bit-width. Using the layout described in Section 3.2, the number of turnarounds is the same as the number of tip switches, because a tip switch is necessary at every turnaround. The X-move component also depends on the number of tips in Y per cluster that determines the cluster dimension (i.e., the number of bits in one cluster column).

Specifically, the average number of turnarounds (or tip switches) per request is the ratio of the number of bits each tip must access to service the average size request (r_f) and the number of bits in each column ($n_{bitsinY}$), as shown in Equation 6.

$$n_{TA} = \frac{r_f}{n_{bitsinY}} \quad (6)$$

The average number of X-moves is the ratio of the average requested number of bits per active tip (r_f) divided

by the product of the number of bits in Y times the number of tips in Y per cluster (T_y), as shown in Equation 7.

$$n_{XM} = \frac{r_f}{n_{bitsinY} \times T_y}. \quad (7)$$

The number of bits in Y is equal to the movement range in Y divided by the bit-width.

Substituting Equations 4–6 in Equation 3 gives us an expression for the transfer time in milliseconds shown in Equation 8.

$$t_{transfer}(\delta_y, T_{active}, T_y, r) = \frac{8rd_b}{T_{active}}(\frac{1}{\delta_y}(t_{TA} + \frac{t_{XM}}{T_y}) + \frac{1}{v_0}) \quad (8)$$

4.1.2 Seek Time Model

Seek time is the time that it takes to reposition the mover above the relevant tips when servicing a new non-sequential request. To model the seek time we used a probabilistic approach. We assume that the starting locations of requests are uniformly distributed across the device.

Because the movement in X and Y may be considered independently [10, 6], the seek time can be computed as the greater of the seek times in X and Y. We computed the average seek time when the seek in X is greater ($t_{seek\ x}(\delta_x)$), and the average seek time when the seek in Y is greater ($t_{seek\ y}(\delta_y)$) and estimated these probabilities a and b, which add up to 1, to obtain Equation 9. We reason that a higher average seek time in one direction will have a higher impact on the overall seek time, so $a/b = t_{seek\ x}(\delta_x)/t_{seek\ y}(\delta_y)$.

$$t_{seek}(\delta_x, \delta_y) = a \times t_{seek\ x}(\delta_x) + b \times t_{seek\ y}(\delta_y) \quad (9)$$

While several models exist to estimate the seek time, the one we adopted is based on simple acceleration rules from Newtonian mechanics [13, 10] and is given in Equation 10, for a specified distance δ. The mover can seek achieve a much higher velocity than is used to read/write because it can accelerate during the seek. At the end of a seek, the data must be accessed by moving in the Y direction. Therefore, a settle time, t_{settle} (for the mover to position itself accurately) applies only to the calculated seek time in X, since the mover has to come to a complete stop in that direction. Notice that seeking takes place within one tip area. Therefore, the seeking distance and the seek time are relatively short, and depend on the mover movement range.

$$t(\delta) = \sqrt{\frac{2\pi\delta}{a_0}} \quad (10)$$

We calculate the average seek time in X and Y as follows. First, we calculate the probability function $p(d)$ of an incoming request incurring a certain movement d over the tip array either X or Y (we can consider each direction independently). Because the starting sectors are uniformly distributed, the distances in each direction will be the distance between two uniformly distributed starting sectors. This probability will vary linearly with distance and will be at its maximum for a zero displacement, and it will be equal to zero when the displacement is equal to the movement range. The form of the equation is $p(d) = c \quad bd$, where c and b are constants. Because p is a probability function, the area under it in the range 0 to δ, where δ is the movement range in X or Y, which is $c\delta/2$ will be equal to 1. Additionally, when $d = \delta$, the probability is equal to zero, $p(\delta) = 0$. Using these constraints, we can calculate the values for the constants c and b, which we substitute back in the expression for $p(d)$ to obtain Equation 11:

$$p(d) = 2(\delta \quad d)/\delta^2 \quad (11)$$

Unfortunately, we need to calculate seek time distributions, not distance distributions. We can use Equation 10 to express t as a function of a displacement d, converting Equation 11 from the distance domain to the time domain. This gives us the probability of a seek incurring time t in either X or Y. After substituting 250 m/s^2 for the physical parameter acceleration, a_0, we obtain Equation 12:

$$p(t) = \frac{500t}{\pi\delta}(1 \quad \frac{250t^2}{2\pi\delta}) \quad (12)$$

The movement ranges δ_x and δ_y in X and Y from Table 1 specify the maximum distance we can move in each direction. We use them in Equation 10 to find the maximum seek time t_{max} in X or Y, shown in Equation 13.

$$t_{max} = \sqrt{\frac{2\pi\delta}{250}} \quad (13)$$

The actual seek time is the greater of the seek times in X and Y, so the probability distribution function $P(t)$ of the seek time when it is greater in one direction than the other will be different than $p(t)$. Reasoning that taking the maximum will bias us towards larger seek times, we approximated $P(t)$ with Equation 14.

$$P(t) = \alpha p(t)t \quad (14)$$

To solve for the constant factor α we integrated $P(t)$ from 0 to t_{max} and normalized it (because it is a probability function). Using our default values for physical

parameters, we obtained $\alpha = 75\sqrt{5}/(8\sqrt{\pi\delta})$, or $\alpha = 11.83/\sqrt{\delta}$.

The average value of the seek time in one direction, taken over all the requests for which its seek time was greater than that in the other direction, can be estimated by integrating its probability function over all possible times (0 to t_{max}), shown in Equation 15.

$$t_{average\ seek}(\delta) = \int_0^{t_{max}} P(t)tdt \tag{15}$$

When we substitute in Equations 12, 13 and 14 for $P(t)$ we obtain Equation 16:

$$t_{average\ seek}(\delta) = \frac{1}{8}\sqrt{\frac{\pi\delta}{5}} \tag{16}$$

Seeking in X involves a settling time t_{settle} that we add to the prediction for the average seek time in X. The final formulae for seek times in X and Y are shown in Equations 17 and 18.

$$t_{seek\ x}(\delta_x) = \frac{1}{8}\sqrt{\frac{\pi\delta_x}{5}} + t_{settle} \tag{17}$$

$$t_{seek\ y}(\delta_y) = \frac{1}{8}\sqrt{\frac{\pi\delta_y}{5}}, \tag{18}$$

We can now substitute Equations 17 and 18 in Equation 9 to obtain an expression for the seek time, which is shown in Equation 19.

$$t_{seek}(\delta_x, \delta_y) = \frac{1}{\sqrt{\delta_x} + \sqrt{\delta_y} + 8t_{settle}\sqrt{\frac{5}{\pi}}} (\sqrt{\delta_x} + 8t_{settle}\sqrt{\frac{5}{\pi}})(\frac{1}{8}\sqrt{\frac{\pi\delta_x}{5}} + t_{settle}) + \delta_y\frac{1}{8}\sqrt{\frac{\pi}{5}}) \tag{19}$$

4.2 Service Time Model

In Section 4.1.2 and Section 4.1.1 we obtained expressions for the seek and transfer times. We can now substitute these equations (19 and 8) for t_{seek} and $t_{transfer}$ in Equation 2. This gives us an expression for the service time as shown in Equation 20:

Workload	*Average request size*	*Runlength*
cello92	6.4 KB	6.5 KB
snake	6.8 KB	8.9 KB
cello99	6.8 KB	7.2 KB
tpcd	29.3 KB	252.0 KB

Table 2: Average request size and runlength values for workloads used in our simulations.

$$t_{service}(\delta_x, \delta_y, T_{active}, T_y, r, r_l) = \frac{r}{r_l(\sqrt{\delta_x} + \sqrt{\delta_y} + 8t_{settle}\sqrt{\frac{5}{\pi}})} (\sqrt{\delta_x} + 8t_{settle}\sqrt{\frac{5}{\pi}})(\frac{1}{8}\sqrt{\frac{\pi\delta_x}{5}} + t_{settle}) + \delta_y\frac{1}{8}\sqrt{\frac{\pi}{5}}) + \frac{8rd_b}{T_{active}}(\frac{1}{\delta_y}(t_{TA} + \frac{t_{XM}}{T_y}) + \frac{1}{v_0}) \tag{20}$$

4.3 Model and Simulation Comparisons

To compare our model with the performance of a prototypical probe-based storage device we conducted experiments using the Pantheon simulator from HP. We used a probe-based storage device simulator that implements the architecture and layout described in Section 3 [22], using a variant of the unconstrained sled model [10, 22]. We used four workloads: 1992 cello (4% sequential, /news partition, most requests smaller than 8KB, sector size is 256 bytes) [15], 1992 snake (23% sequential, /usr2 partition, most requests smaller than 8KB, sector size is 512 bytes) [15], 1999 cello (disk 128, 30% sequential, large requests where more than half are larger than 8KB, sector size is 1024 bytes) [22] and 1999 tpcd (disk 80, which accounted for 12.5% of the requests). All traces recorded one week of activity. In our simulations, we issue the requests at the original times they occur in the traces, so queuing behavior in the original workloads will not be as pronounced on the faster probe-based storage devices.

All original traces were recorded by HP Laboratories. The average request size and runlength for each workload are shown in Table 2.

We used as default physical parameter values from Table 1 and varied configurable parameters one at a time within the ranges shown in same table. These ranges were selected because they represent reasonable variation

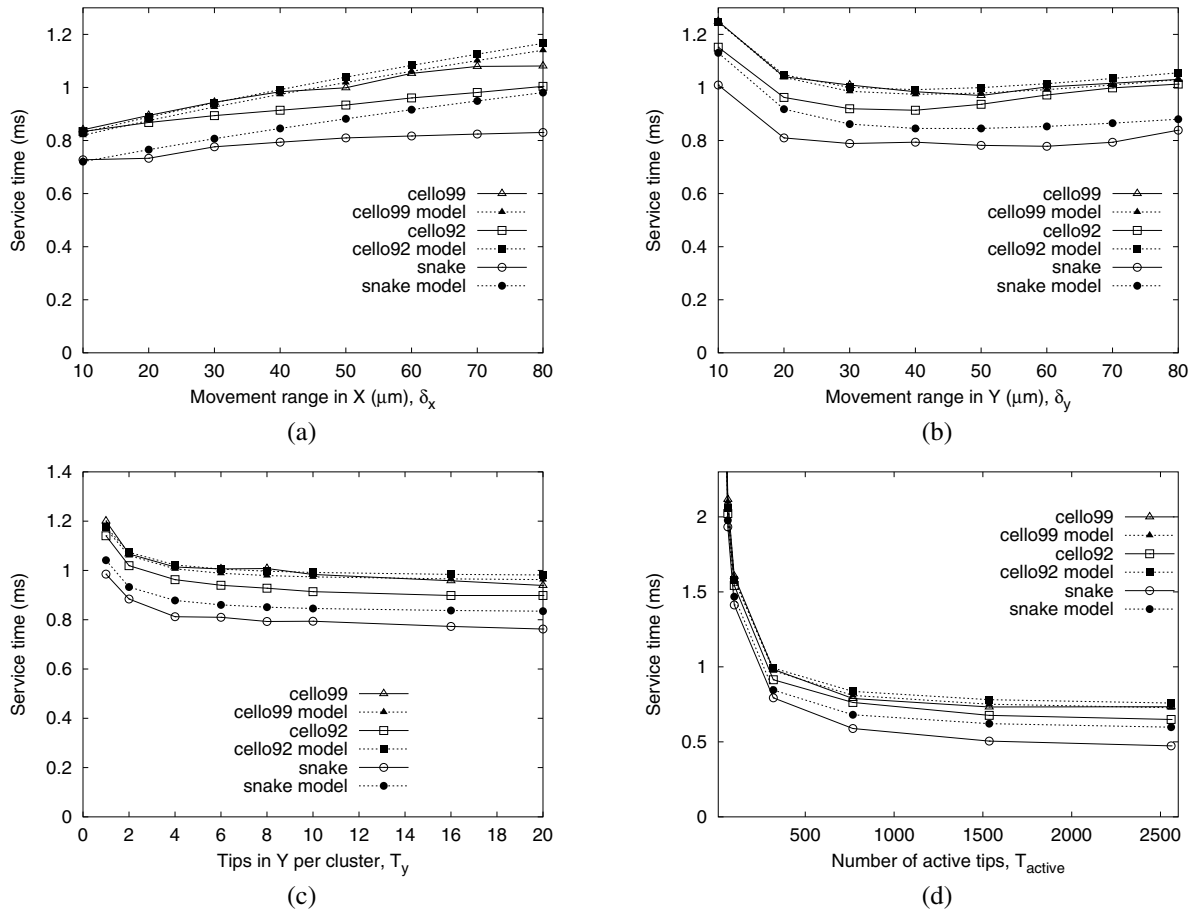


Figure 5: Graphs of service time of the simulated snake, cello92 and cello99 workloads and values calculated with our probe-based storage model. The initial configuration used had a 40μm movement range in both X and Y, 320 active tips and 10 tips in Y per cluster. (a) varying the movement range in X; (b) varying the movement range in Y; (c) varying the number of active tips; (d) varying the number of tips in Y per cluster.

around default values for which behavior is well understood [22].

Figures 5a–d show the results we obtained for snake, cello92, and cello99 workloads, and 6a–d show our results for the tpcd workloads. In Figure 5a, we see that for snake and cello92, as δ_x increases, the predicted service time deviates more from the experimental values. This is because the original disk from which the traces are taken is smaller than the probe-storage devices we are mapping those requests to. Thus, seeks occur over only a small fraction of the movement range in X, and our model overestimates the seek time as the device capacity increases. For cello99 and tpcd, traces taken from disks with higher capacity, this effect is much smaller. In fact, the error decreases in the case of the tpcd workload, as we can see from Figure 6a.

In general, model error decreases as we increase movement range in Y (Figure 5b). Because the sectors are ordered vertically, seeks are distributed across the entire Y movement range during the workload simulation. Thus the seek time predicted by our model better approximates the experimentally obtained values. This error becomes even smaller for higher movement ranges in Y, when seeks in Y dominate those in X. We can see from Figure 5b that when the movement range in Y is greater than 20μm, service time for cello92, snake and cello99, which have similar request sizes and runlengths, does not change significantly. Increasing the movement range in Y increases the average number of bytes that can be read before the mover must turnaround, however, these workloads with small requests do not benefit from this. In contrast, tpcd has a much larger request size and runlength, hence its service time is minimized for greater values of the movement range in Y, as shown in Figure 6b.

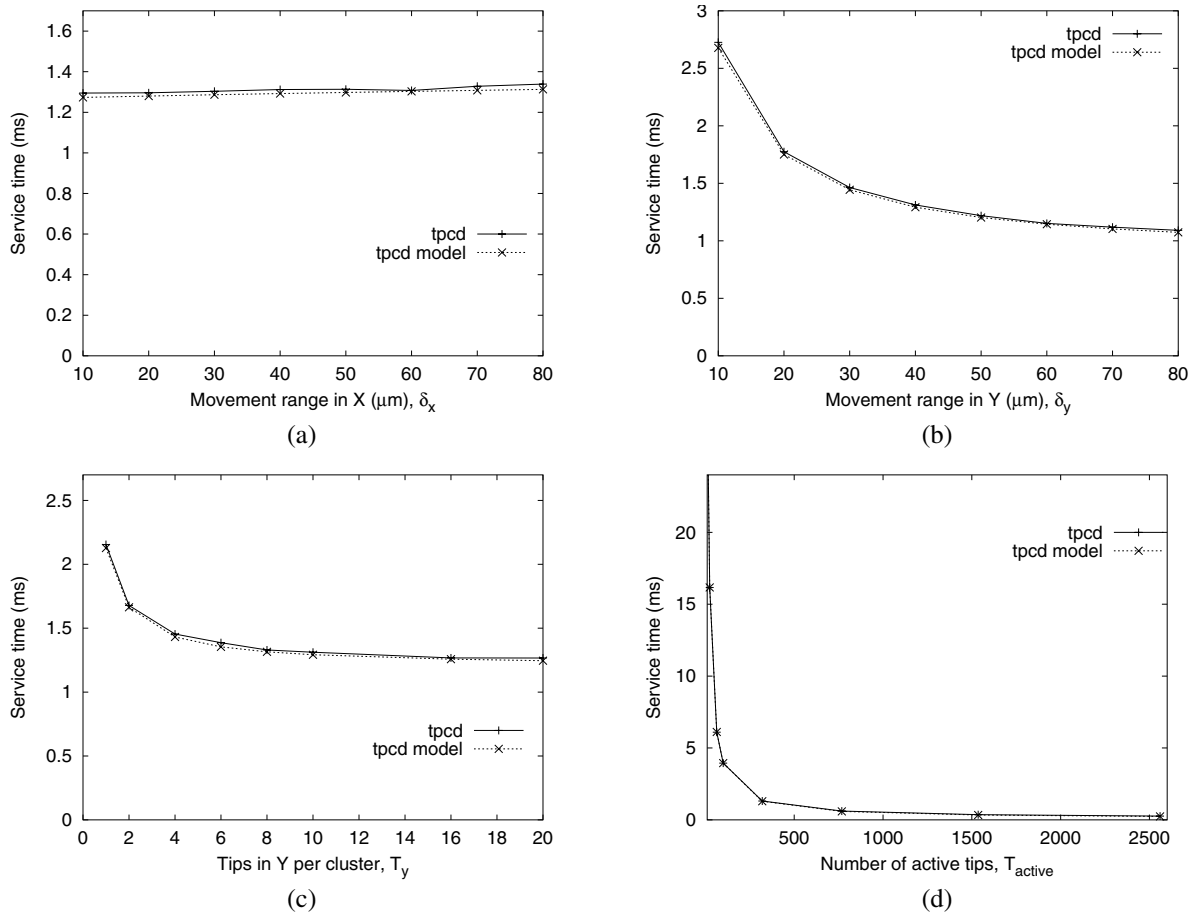


Figure 6: Graphs of the service time of the simulated tpcd workload and values calculated using our probe-based storage model. The initial configuration used had a 40μm movement range in both X and Y, 320 active tips and 10 tips in Y per cluster. (a) varying the movement range in X. (b) varying the movement range in Y. (c) varying the number of active tips. (d) varying the number of tips in Y per cluster.

Figures 5c and 6c confirm our expectation that the number of tips in Y per cluster has a negligible effect on performance at values greater than two.

Finally, Figures 5d and 6d show that increasing the number of active tips increases the level of parallelism inside the device, decreasing transfer time. The smaller transfer time at higher numbers of active tips makes errors in seek time more pronounced, causing the deviation between models and simulation.

We show the model error for each configurable parameter in Table 3. The percent error is calculated by computing the average percentage difference of the model-predicted values from the simulated values at each data point. The RMS difference is the root-mean squared vertical difference between the model predictions and the simulated workload value (in ms). The data in Table 3 show that our model closely approximates the simulated service time for all four workloads. Even though we mapped workloads to devices with different capacities and assumed random distribution of requests, our model was within 8% of the simulated values.

5 Optimal System Design

We used our probe-based storage model to optimize design of MEMS devices for specific performance goals. We used the range of values for each configurable parameter shown in Table 1 to bound our search, with the exception of the number of tips in Y per cluster. Because this parameter has a negligible contribution to the transfer time when its value is greater than 2, as can be seen from Figures 5b and 6b, we set it to the default value 10.

To find the optimal set of configurable parameters, we exhaustively searched the entire range of the configurable

Varied parameter	*Workload*							
	cello99		cello92		snake		tpcd	
	mean error	RMS error	mean error	RMS error	mean error	RMS error	mean error	RMS error
δ_x	2.2 %	0.03 ms	8.0 %	0.10 ms	7.8 %	0.09 ms	1.4 %	0.02 ms
δ_y	0.8 %	0.01 ms	6.2 %	0.07 ms	8.3 %	0.08 ms	1.4 %	0.02 ms
A	1.1 %	0.02 ms	6.0 %	0.07 ms	8.4 %	0.08 ms	3.2 %	0.03 ms
T_y	1.6 %	0.02 ms	6.6 %	0.07 ms	6.7 %	0.06 ms	1.4 %	0.02 ms

Table 3: Comparison of the service time of several simulated workloads to our model as both a percentage difference and a root mean squared difference, as we vary several parameters.

parameter space to find a set of values that optimized the given metric. We varied each parameter by intervals of 1μm for the movement range in both X and Y and 10 for the number of active tips.

5.1 Minimizing Service Time for a Fixed Capacity and Request Size

For a given bit density, the capacity of a MEMS device is determined by the product of active tips and movement ranges in X and Y. We looked for a configuration that minimized mean response time for a probe-based storage device with a fixed capacity of 2GB.

Given this constraint, our model identified an optimal configuration for each workload as shown in Figure 7b. We can see that the optimum number of active tips is at the top of the range for the values of that parameter. This is not surprising, because additional active tips lower service time. The movement ranges in X and Y are small, which also reduces service time. If we compare the values of the optimal configurations to the behavior of the experimentally obtained values for the service time in Figure 7a we can see that our predictions point out very well where the minimum of the service time will occur.

Note that the error is greatest for the tpcd workload, where the model generally underestimates the service time. Specifically, the model underestimates the seek time for tpcd because the distribution of seeks in the real workload does not fit our assumption of a uniformly random distribution. The tpcd workload has a significant number of large seeks that increase the overall average. As we change the movement range in X and Y, the model prediction changes more quickly than the real workload. Thus, at higher values of δ_y, the simulated and predicted service times converge.

5.2 Minimizing Service Time for a Fixed Number of Active Tips

In reality, design of a probe-based storage device is likely to be constrained by cost. We approximate this by fixing the number of active tips at 320 (assuming that the number of tips is proportional to the cost of the device) and determining a configuration that minimizes service time.

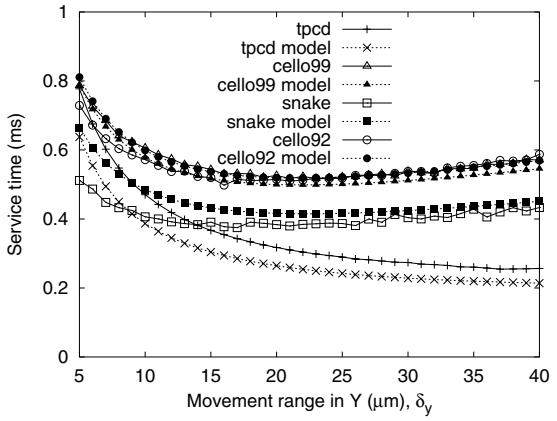
Figure 8b shows the configurations that minimize the service time for each workload. Figure 8a shows the service time when the movement range in Y is varied. Our model again does an accurate prediction of where the minimum service time occurs for all four workloads. In the case of snake, cello99 and cello92 this is just around 30μm, while in the case of tpcd this happens at the upper bound for the movement range in Y, 80μm. This makes sense because the tpcd workload has larger request sizes and runlengths, and therefore can benefit from the fewer turnarounds offered by long movements in Y.

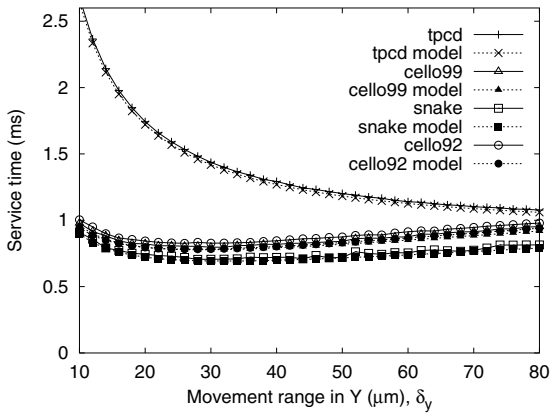
6 Conclusions

The design space of probe-based storage devices is vast, but can be explored more quickly with the use of a parameterized model. We have presented a model for a probe-based storage device that allows a system designer to predict performance for a set of configurable parameters. We validated this model using a simulator for probe-based storage and several workloads and found that the error was within 15%. We successfully used this model to quickly identify optimal configurations to satisfy different performance objectives.

Acknowledgments

We are grateful to John Wilkes and Chuck Morehouse at HP Labs for important discussions and the use of the Pantheon simulator and traces. Thanks also to the anonymous FAST reviewers whose comments, together with shepherding by John Wilkes, greatly improved the draft. This research is based upon work supported by the National Science Foundation under grants NSF CCR-0073509 and 0093051.





Optimal Configuration	cello99	cello92	tpcd	snake
$\delta_x(\mu$m)	9	9	5	9
$\delta_y(\mu$m)	22	22	40	22
T_{active}	2560	2560	2560	2560
Service time				
simulated (ms)	0.52	0.51	0.26	0.38
predicted (ms)	0.50	0.52	0.22	0.42
error	3.8%	1.9%	15.4%	13.5%
Default Configuration	cello99	cello92	tpcd	snake
Service time				
simulated (ms)	0.98	0.91	1.31	0.79
predicted (ms)	0.97	1.00	1.38	0.85
Difference from optimal				
simulated	88%	78%	403%	108%
predicted	94%	92%	527%	102%

Figure 7: Performance of configuration minimizing service time when capacity and request sizes are fixed. Above: Graph of the service time for a range of values for the movement range in Y for the simulated trace workloads. The movement range in X was adjusted at each point to keep the capacity at 2GB. Below: Experimentally obtained and predicted service times for the four workloads, and optimal configurations.

Optimal Configuration	cello99	cello92	tpcd	snake
$\delta_x(\mu$m)	5	5	5	5
$\delta_y(\mu$m)	29	27	80	32
T_{active}	320	320	320	320
Service time				
simulated (ms)	0.80	0.83	1.08	0.70
predicted (ms)	0.77	0.78	1.06	0.69
error	3.8%	6.0%	1.9%	1.4%
Default Configuration	cello99	cello92	tpcd	snake
Service time				
simulated (ms)	0.98	0.91	1.31	0.79
predicted (ms)	0.97	1.00	1.38	0.85
Difference from optimal				
simulated	23%	9.6%	21.3%	13%
predicted	26%	28.2%	30.2%	23%

Figure 8: Performance of configuration minimizing service time for a fixed number of active tips. Above: Graph of the service time for a range of values for the movement range in Y for the simulated trace workloads. Below: Experimentally obtained and predicted service times for the four workloads and optimal configurations.

References

[1] M. C. Abraham, H. Schmidt, R. J. Ram, T. A. Savas, H. I. Smith, M. Hwang, and C. Ross. MFM studies of nanomagnetic arrays. http://rleweb.mit.edu/sclaser/mmm99/sld001.htm, Massachusetts Institute of Technology, Seminconductor Laser Group, Feb 2000. Accessed Dec. 2002.

[2] L.R. Carley, J.A. Bain, and G.K. Fedder. Single chip computers with MEMS-based magnetic memory. In *44th Annual Conference on Magnetism and Magnetic Materials*, November 1999.

[3] Center for highly integrated information processing and storage systems (CHIPS) home page. http://www.chips.ece.cmu.edu, Carnegie Institute of Technology, Carnegie Mellon University. Accessed Dec. 2002.

[4] S. H. Charap, Pu Ling Lu, and Yanjun He. Thermal stability of recorded information at high densities. In *IEEE Transactions on Magnetics*, volume 33, pages 978–983, January 1997.

[5] M. Despont, J. Brugger, U. Drechsler, U. Dürig, W. Häberle, M. Lutwyche, H. Rothuizen, R. Stutz, R. Widmer, H. Rohrer, G. Binnig, and P. Vettiger. VLSI-NEMS chip for AFM data storage. In *Technical Digest. Twelfth IEEE International Conference on Micro Electro Mechanical Systems*, pages 564–69, Orlando, FL, January 1999.

[6] J. L. Griffin, S. W.Schlosser, G. R Ganger, and D. F. Nagle. Modeling and performance of MEMS-based storage devices. In *Proceedings of ACM SIGMETRICS 2000*, pages 56–65, (Santa Clara, CA), June 2000.

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

[8] J. L. Hennessy and D. A.Patterson. *Computer Architecture A Quantitative Approach*. Morgan Kaufmann, 1996.

[9] Ying Lin, Scott A. Brandt, Darrell D. E. Long, and Ethan L. Miller. Power conservation strategies for MEMS-based storage devices. In *Proceedings of the 10th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '02)*, Fort Worth, TX, October 2002.

[10] Tara M. Madhyastha and Katherine Pu Yang. Physical modeling of probe-based storage. In *Proceedings of the 18th IEEE Symposium on Mass Storage Systems*, pages 207–224, April 2001.

[11] H. J. Mamin, B. D. Terris, L. S. Fan, S. Hoen, R. C. Barrett, and D. Rugar. High-density data storage using proximal probe techniques. *IBM Journal of Research and Development*, 39(6):681–99, November 1995.

[12] G. Moore. Progress in digital integrated electronics. In *Proceedings of the IEEE Digital Integrated Electronic Device Meeting*, pages 11–13, 1975.

[13] Personal communication. Dr. Charles C. Morehouse, HP laboratories, May 2001.

[14] Demetri Psaltis and Geoffrey W. Burr. Holographic data storage. *IEEE Computer*, 31(2):52–60, February 1998.

[15] Chris Ruemmler and John Wilkes. Unix disk access patterns. In *Proceedings of Winter'93 USENIX Conference*, pages 405–420, January 1993.

[16] Chris Ruemmler and John Wilkes. An introduction to disk drive modeling. *IEEE Computer*, 27(3):17–28, March 1994.

[17] S. W. Schlosser, J. L. Griffin, D. F. Nagle, and G. R. Ganger. Filling the memory access gap: A case for on-chip magnetic storage. Technical Report CMU–CS–99–174, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, November 1999.

[18] Steven W. Schlosser, John Linwood Griffin, David Nagle, and Gregory 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*, pages 1–12, Boston, Massachusetts, November 2000.

[19] Elizabeth Shriver, Arif Merchant, and John Wilkes. An analytic behavior model for disk drives with readahead cach es and request reordering. In *Proceedings of ACM SIGMETRICS 1998*, pages 183–191, Madison, WI, June 1998.

[20] Glenn T. Sincerbox, editor. *Selected Papers on Holographic Storage*, volume MS 95 of *SPIE Milestone series*. International Society for Optical Engineering, 1994.

[21] Miriam Sivan-Zimet. Workload based optimization of probe-based storage. Technical Report UCSC–CRL–01–06, University of California, Santa Cruz, July 2001. http://www.cse.ucsc.edu/tara/papers/miri_thesis.pdf.

[22] Miriam Sivan-Zimet and Tara M. Madhyastha. Workload based optimization of probe-based storage. In *Proceedings of the 2002 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 256–257, Marina Del Ray, CA, June 2002.

[23] J. W. Toigo. Avoiding a data crunch. *Scientific American*, 279(5):58–74, May 2000.

[24] Mustafa Uysal, Arif Merchant, and Guillermo Alvarez. Using MEMS-based storage in disk arrays. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03)*, San Francisco, CA, April 2003. To appear.

[25] P. Vettiger, M. Despont, U. Drechsler, U. Drig, W. Hberle, M. I. Lutwyche, H. E. Rothuizen, R. Stutz, R. Widmer, and G. K. Binnig. The "Millipede"—more than one thousand tips for future AFM data storage. *IBM Journal of Research and Development*, 44(3):323, 1999.

[26] John Wilkes. The Pantheon storage-system simulator. Technical Report HPL–SSP–95–14, Storage Systems Program, Hewlett-Packard Laboratories, Palo Alto, CA, May 1996. http://www.hpl.hp.com/SSP/papers/PantheonOverview.pdf.