Storage Alternatives for Mobile Computers*

Fred Douglis
AT&T Bell Laboratories

Ramón Cáceres
AT&T Bell Laboratories

Frans Kaashoek
Massachusetts Institute of Technology

Kai Li
Princeton University

Brian Marsh
D.E. Shaw & Co.

Joshua A. Tauber
Massachusetts Institute of Technology

To appear in the *Symposium on Operating Systems Design and Implementation*, November, 1994

Abstract

Mobile computers such as notebooks, subnotebooks, and palmtops require low weight, low power consumption, and good interactive performance. These requirements impose many challenges on architectures and operating systems. This paper investigates three alternative storage devices for mobile computers: magnetic hard disks, flash memory disk emulators, and flash memory cards.

We have used hardware measurements and trace-driven simulation to evaluate each of the alternative storage devices and their related design strategies. Hardware measurements on an HP OmniBook 300 highlight differences in the performance of the three devices as used on the Omnibook, especially the poor performance of version 2.00 of the Microsoft Flash File System [11] when accessing large files. The traces used in our study came from different environments, including mobile computers (Macintosh PowerBooks) and desktop computers (running Windows or HP-UX), as well as synthetic workloads. Our simulation study shows that flash memory can reduce energy consumption by an order of magnitude, compared to magnetic disk, while providing good read performance and acceptable write performance. These energy savings can translate into a 22% extension of battery life. We also find that the amount of unused memory in a flash memory card has a substantial impact on energy consumption, performance, and endurance: compared to low storage utilizations (40% full), running flash memory near its capacity (95% full) can increase energy consumption by 70–190%, degrade write response time by 30%, and decrease the lifetime of the memory card by up to a third. For flash disks, asynchronous erasure can improve write response time by a factor of 2.5.

1 Introduction

Mobile computer environments are different from traditional workstations because they require light-weight, low-cost, and low-power components, while still needing to provide good interactive performance. A principal design challenge is to make the storage system meet these conflicting requirements.

Current storage technologies offer two alternatives for file storage on mobile computers: magnetic hard disks and flash memory. Hard disks provide large capacity at the lowest cost, and have high throughput for large transfers. The main disadvantage is that they consume a lot of energy and take seconds to spin up

*This work was performed at Panasonic Technologies, Inc.'s Matsushita Information Technology Laboratory.

and down. Flash memory consumes relatively little energy, and has low latency and high throughput for read accesses. The main disadvantages of flash memory are that it costs more than disks—\$30–50/Mbyte, compared to \$1–5/Mbyte for magnetic disks—and that it requires erasing before it can be overwritten. It comes in two forms: flash memory cards (accessed as main memory) and flash disk emulators (accessed through a disk block interface).[1] These devices behave differently, having varying access times and bandwidths.

This paper investigates three storage systems: magnetic disk, flash disk emulator, and directly accessed flash memory. All of these systems include a DRAM file cache. Our study is based on both hardware measurements and trace-driven simulation. The measurements are "micro-benchmarks" that compare the raw performance of three different devices: a typical mobile disk drive (Western Digital Caviar Ultralite CU140), a flash disk (SunDisk 10-Mbyte SDP10 PCMCIA flash disk [21], sold as the Hewlett-Packard F1013A 10-Mbyte/12-V Flash Disk Card [6]), and a flash memory card (Intel 10-Mbyte Series-2 flash memory card [8]). The measurements provide a baseline comparison of the different architectures and are used as device specifications within the simulator. They also point out specific performance issues, particularly with the Microsoft Flash File System (MFFS) version 2.00 [11].

Flash memory is significantly more expensive than magnetic disks, but our simulation results show that flash memory can offer energy reduction by an order of magnitude over disks—even with aggressive disk spin-down policies that save energy at the cost of performance [5, 13]. Since the storage subsystem can consume 20–54% of total system energy [13, 14], these energy savings can as much as double battery lifetime. Flash provides better read performance than disk, but worse average write performance. The maximum delay for magnetic disk reads or writes, however, is much higher than maximum flash latency due to the overhead of occasional disk spin-ups.

We also show that the key to file system support using flash memory is erasure management. With a flash card, keeping a significant portion of flash memory free is essential to energy conservation and performance. With a flash disk, decoupling write and erase latency can improve average write response by a factor of 2.5.

In total, our paper uses both hardware measurements and simulation to contribute two key results: a quantitative comparison of the alternatives for storage on mobile computers, taking both energy and performance into account, and an analysis of techniques that improve on existing systems.

The rest of this paper is organized as follows. The next section discusses the three storage architectures in greater detail. Section 3 describes the hardware micro-benchmarks. Section 4 describes our traces and the simulator used to perform additional studies. After that come the results of the simulations. Section 6 discusses related work, and Section 7 concludes.

2 Architectural Alternatives

The three basic storage architectures we studied are magnetic disks, flash disk emulators, and flash memory cards. Their power consumption, cost, and performance are a function of the workload and the organization of the storage components. Each storage device is used in conjunction with a DRAM buffer cache. Though the buffer cache can in principle be write-back, in this paper we consider a write-through buffer cache: this models the

[1] In this paper, we use *flash disk* to refer to block-accessed flash disk emulators. We use *flash (memory) card* to refer to byte-accessible flash devices. When we wish to refer to the generic memory device or either of the above devices built with it, we refer to *flash memory* or a *flash device*. Note that the flash disk is actually a flash memory card as well, but with a different interface.

behavior of the Macintosh operating system and until recently the DOS file system.

An idle disk can consume 20–54% or more of total system energy [13, 14], so the file system must spin down the disk whenever it is idle. Misses in the buffer cache will cause a spun-down disk to spin up again, resulting in delays of up to a few seconds [5, 13]. Writes to the disk can be buffered in battery-backed SRAM, not only improving performance, but also allowing small writes to a spun-down disk to proceed without spinning it up. The Quantum Daytona is an example of a drive with this sort of buffering. In this paper, we give magnetic disks the benefit of the doubt by simulating this deferred spin-up policy except where noted.

The flash disk organization replaces the hard disk with a flash memory card that has a conventional disk interface. With the SunDisk SDP series, one example of this type of device, transfers are in multiples of a sector (512 bytes). In contrast, the flash card organization removes the disk interface so that the memory can be accessed at byte-level. The flash card performs reads faster than the flash disk, so although the instantaneous power consumption of the two devices during a read is comparable, the flash card consumes less energy to perform the operation.

A fundamental problem introduced by flash memory is the need to *erase* an area before it can be overwritten. The flash memory manufacturer determines how much memory is erased in a single operation. The SunDisk devices erase a single 512-byte sector at a time, while the Intel Series-2 flash card erases one or two 64-Kbyte "segments." There are two important aspects to erasure: *flash cleaning* and performance. When the segment size is larger than the transfer unit (i.e., for the flash card), any data in the segment that are still needed must be copied elsewhere. Cleaning flash memory is thus analogous to segment cleaning in Sprite LFS [19]. The cost and frequency of segment cleaning is related in part to the cost of erasure, and in part to the segment size. The larger the segment, the more data that will likely have to be moved before erasure can take place. The system must define a policy for selecting the next segment for reclamation. One obvious discrimination metric is segment utilization: picking the next segment by finding the one with the lowest utilization (i.e., the highest amount of memory that is reusable). MFFS uses this approach [4]. More complicated metrics are possible; for example, eNVy considers both utilization and locality when cleaning flash memory [24].

The second aspect to erasure is performance. The SunDisk SDP flash disks couple erasure with writes, achieving a write bandwidth of 75 Kbytes/s. The time to erase and write a block is dominated by the erasure cost. The Intel flash card separates erasure from writing, and achieves a write bandwidth of 214 Kbytes/s—but only after a segment has been erased. Because erasure takes a large fixed time period (1.6s) regardless of the amount of data being erased [8], the cost of erasure is amortized over large erasure units. (The newer 16-Mbit Intel Series 2+ Flash Memory Cards erase blocks in 300ms [9], but these were not available to us during this study.) The two types of flash memory have comparable erasure bandwidth; to avoid delaying writes for erasure it is important to keep a pool of erased memory available. It becomes harder to meet this goal as more of the flash card is occupied by useful data, as discussed in Section 5.2.

Another fundamental problem with flash memory is its limited endurance. Manufacturers guarantee that a particular area within flash may be erased up to a certain number of times before defects are expected. The limit is 100,000 cycles for the devices we studied; the Intel Series 2+ Flash Memory Cards guarantee one million erasures per block [9]. While it is possible to spread the load over the flash memory to avoid "burning out" particular areas, it is still important to avoid unnecessary writes or situations that erase the same area repeatedly.

3 Hardware Measurements

We measured the performance of the three storage organizations of interest on a Hewlett-Packard OmniBook 300. The OmniBook 300 is a 2.9-pound subnotebook computer that runs MS-DOS 5.0 and contains a 25-MHz 386SXLV processor and 2 Mbytes of DRAM. The system is equipped with several PCMCIA slots, one of which normally holds a removable ROM card containing Windows and several applications. We used a 40-Mbyte Western Digital Caviar Ultralite CU140 and a 10-Mbyte SunDisk SDP10 flash disk, both of which are standard with the OmniBook, and a PCMCIA 10-Mbyte Intel Series 2 Flash Memory Card running the Microsoft Flash File System [11]. The Caviar Ultralite CU140 is compatible with PCMCIA Type III specifications, and weighs 2.7 ounces, while the flash devices are PCMCIA Type II cards weighing 1.3 ounces. Thus one may consider two 10-Mbyte flash devices as equivalent in size and weight to a single 40-Mbyte hard disk. However, in our simulations we treated the flash devices as though they too stored 40 Mbytes, since their capacities are increasingly rapidly and the difference in energy consumption and performance between individual flash devices of different capacities using the same technology are minimal. Cost does scale with capacity, of course, and must be taken into account. Finally, the CU140 and SDP10 could be used directly or with compression, using DoubleSpace and Stacker, respectively. Compression is built into MFFS 2.00.

We constructed software benchmarks to measure the performance of the three storage devices. The benchmarks repeatedly read and wrote a sequence of files, and measured the throughput obtained. Both sequential and random accesses were performed, the former to measure maximum throughput and the latter to measure the overhead of seeks. For the CU140 and SDP10, we measured throughput with and without compression enabled; for the Intel card, compression was always enabled, but we distinguished between completely random data and compressible data. The compressible data consisted of the first 2 Kbytes of Herman Melville's well-known novel, *Moby-Dick*, repeated throughout each file (obtaining compression ratios around 50%). The Intel flash card was completely erased prior to each benchmark to ensure that writes from previous runs would not cause excess cleaning.

Table 1 summarizes the measured performance for 4-Kbyte reads and writes to 4-Kbyte and 1-Mbyte files, while Figure 1 graphs the average latency and instantaneous throughput for 4-Kbyte writes to a 1-Mbyte file. These numbers all include DOS file system overhead. There are several interesting points to this data:

- Without compression, throughput for the magnetic disk increases with file size, as expected. With compression, small writes go quickly, because they are buffered and written to disk in batches. Large writes are compressed and then written synchronously.
- Compression similarly helps the performance of small file writes on the flash disk, resulting in write throughput greater than the theoretical limit of the SunDisk SDP10.
- Read throughput of the flash card is much better than the other devices for small files, with reads of uncompressible data obtaining about twice the bandwidth of reads of compressible data (since the software decompression step is avoided). Throughput is unexpectedly poor for reading or writing large files. This is due to an anomaly in MFFS 2.00[11], whose performance degrades with file size. The latency of each write (Figure 1(a)) in-

Device	Operation	Throughput (Kbytes/s) Uncompressed		Throughput (Kbytes/s) Compressed	
		4–Kbyte file	1–Mbyte file	4–Kbyte file	1–Mbyte file
Caviar Ultralite CU140	Read	116	543	64	543
	Write	76	231	289	146
SunDisk SDP10	Read	280	410	218	246
	Write	39	40	225	35
Intel flash card	Read	645	37	345	34
	Write	43	21	83	27

Table 1: Measured performance of three storage devices on an HP OmniBook 300.

Device	Operation	Latency (ms)	Throughput (Kbytes/s)	Power (W)
Caviar Ultralite CU140	Read/Write	25.7	2125	1.75
	Idle	—	—	0.7
	Spin up	1000.0	—	3.0
SunDisk SDP10	Read	1.5	600	0.36
	Write	1.5	50	0.36
Intel flash card	Read	0	9765	0.47
	Write	0	214	0.47
	Erase	1600	70	0.47

Table 2: Manufacturers' specifications for three storage devices. Latency for read/write operations indicates the overhead from a random operation, excluding the transfer itself (i.e., controller overhead, seeking, or rotational latency). The Intel erasure cost refers to a separate operation that takes 1.6s to erase 64 or 128 Kbytes (in this case latency and throughput are analogous).

creases linearly as the file grows, apparently because data already written to the flash card are written again, even in the absence of cleaning. This results in the throughput curve in Figure 1(b).

Comparing the different devices, it is obvious that the Caviar Ultralite CU140 provides the best write throughput, since the disk is constantly spinning; excluding the effects of compression, the flash card provides better performance than the flash disk for small files on an otherwise empty card, while its read and write performance are both worse than the flash disk for larger files.

In Table 2 we include the raw performance of the devices, and power consumed, according to datasheets supplied by the manufacturers. As shown, the hard disk offers the best throughput of the three technologies, but consumes many times the power of the flash-based technologies. With regard to the two flash-based devices, the flash card offers better performance than the flash disk, while both devices offer comparable power consumption.

4 Trace-Driven Simulation

We used traces from several environments to do trace-driven simulation, in order to evaluate the performance and energy consumption of different storage organizations and different storage management policies under realistic workloads. This section describes the traces and the simulator, while Section 5 describes the simulation results.

4.1 Traces

We used four workloads, MAC, PC, HP, and SYNTH. For the MAC trace, we instrumented a pair of Apple Macintosh PowerBook Duo 230s to capture file system workloads from a mobile computing environment. The traces are file-level: they report which file is accessed, whether the operation is a read or write, the location within the file, the size of the transfer, and the time of the access. This trace did not record deletions. The traces were preprocessed to convert file-level accesses into disk-level operations, by associating a unique disk location with each file.

We used DOS traces collected by Kester Li at U.C. Berkeley [12], on IBM desktop PCs running Windows 3.1, also at file-level. They include deletions. The traces were similarly preprocessed.

We used disk-level traces collected by Ruemmler and Wilkes on an HP workstation running HP-UX [20]. These traces include metadata operations, which the file-level traces do not, but they are below the level of the buffer cache, so simulating a buffer cache would give misleading results (locality within the original trace has already been largely eliminated). Thus the buffer cache size was set to 0 for simulations of HP. The trace includes no deletions.

Finally, we created a synthetic workload, called SYNTH, based loosely on the *hot-and-cold* workload used in the evaluation of Sprite LFS cleaning policies [19]. The purpose of the synthetic workload was to provide both a "stress test" for the experimental testbed on the OmniBook, and a series of operations that could be executed against both the testbed and the simulator. (Unfortunately, none of our other traces accessed a small enough dataset to fit on a 10-Mbyte flash device.) The comparison between measured and simulated results appears in Section 5.1. The trace consists of 6 Mbytes of 32-Kbyte files, where $\frac{7}{8}$ of the accesses go to $\frac{1}{8}$ of the data. Operations are divided 60% reads, 35% writes, 5% erases. An erase operation deletes an entire file; the next write to the file writes an entire 32-Kbyte unit. Otherwise 40% of accesses are 0.5 Kbytes in size, 40% are between .5 Kbytes and 16 Kbytes, and 20% are between 16 Kbytes and 32 Kbytes. The interarrival time between operations was modeled as a bimodal distribution with 90% of accesses having a uniform distribution with a mean of 10ms and the remaining accesses taking 20ms plus a value that is exponentially distributed with a mean of 3s.

Though only the MAC trace comes from a mobile environment, the two desktop traces represent workloads similar to what would be used on mobile computers, and have been used in simulations of mobile computers in the past [12, 13, 15]. Table 3 lists additional statistics for the nonsynthetic traces.

4.2 Simulator

Our simulator models a storage hierarchy containing a buffer cache and non-volatile storage. The buffer cache is the first level searched on a read and is the target of all write operations. The cache is write-through to non-volatile storage, which is typical of Macintosh and some DOS environments[2]. A write-back cache might avoid some erasures at the cost of occasional data loss. When the secondary store is magnetic disk, an intermediate level containing battery-backed SRAM can buffer writes; in this case, a write-through DRAM buffer cache especially makes sense, since writes to SRAM are fast. In addition, the buffer cache can have zero size, in which case reads and writes go directly to non-volatile storage. A zero-sized buffer cache is applicable only to the HP-UX trace, which has an implicit buffer cache.

We simulated the disk, flash disk, and flash

[2]DOS supports a write-back cache, but after users complained about losing data, write-through caching became a user-configurable option.

	MAC			DOS			HP		
Applications	Finder, Excel, Newton Toolkit			Framemaker, Powerpoint, Word			email, editing		
Duration	3.5 hours			1.5 hours			4.4 days		
Number of distinct Kbytes accessed	22000			16300			32000		
Fraction of reads	0.50			0.24			0.38		
Block size (Kbytes)	1			0.5			1		
Mean read size (blocks)	1.3			3.8			4.3		
Mean write size (blocks)	1.2			3.4			6.2		
Inter-arrival time (s)	Mean	Max	σ	Mean	Max	σ	Mean	Max	σ
	0.078	90.8	0.57	0.528	713.0	10.8	11.1	30min	112.3

Table 3: Summary of (non-synthetic) trace characteristics. The statistics apply to the 90% of each trace that is actually simulated after the warm start. Note that it is not appropriate to compare performance or energy consumption of simulations of different traces, because of the different mean transfer sizes and durations of each trace.

card devices with parameters for existing hard disk, flash memory disk emulator, and flash memory card products, respectively. Each device is described by a set of parameters that include the power consumed in each operating mode (reading, writing, idle, or sleeping) and the time to perform an operation or switch modes. The power specifications came from datasheets; two different set of performance specifications were used, one from the measured performance and one from datasheets.

In addition to the products described in Section 3, we used the datasheet for the NEC μPD4216160/L 16-Mbit DRAM chip [17]. In the case of the SunDisk device, the simulation using raw (nonmeasured) performance numbers is based upon the SunDisk SDP5 and SDP5A devices, which are newer 5-volt devices [3]. Lastly, we also simulated the Hewlett-Packard Kittyhawk 20-Mbyte hard disk, which we refer to as KH, based on its datasheet [7]. In order to manage all the traces, we simulated flash devices larger than the 10-Mbyte PCMCIA flash devices we had for the OmniBook. Based on the characteristics of different-sized Intel flash cards, the variation in power and performance among flash cards of different size are insignificant.

For each trace, 10% of the trace was processed in order to "warm" the buffer cache, and statistics were generated based on the remainder of the trace.

The simulator accepts a number of additional parameters. Those relevant to this study are:

Flash size	The total amount of flash memory available.
Flash segment size	The size of an erasure unit.
Flash utilization	The amount of data stored, relative to flash size. The data are preallocated in flash at the start of the simulation, and the amount of data accessed during

	the simulation must be no greater than this bound.
Cleaning policy	On-demand cleaning, as with the SunDisk SDP5, and asynchronous cleaning, as with the Flash File System running on the Intel flash card. Flash cleaning is discussed in greater detail below.
Disk spin-down policy	A set of parameters control how the disk spins down when idle and how it spins up again when the disk is accessed.
DRAM size	The amount of DRAM available for caching.

We made a number of simplifying assumptions in the simulator:

- All operations and state transitions are assumed to take the average or "typical" time, either measured by us or specified by the manufacturer.
- Repeated accesses to the same file are assumed never to require a seek (if the transfer is large enough to require a seek even under optimal disk layout, the cost of the seek will be amortized); otherwise, an access incurs an average seek. Each transfer requires the average rotational latency as well. These assumptions are necessary because file-level accesses are converted to disk block numbers without the sophistication of a real file system that tries to optimize block placement.
- For flash file systems, while file data and metadata that would normally go on disk are stored in flash, the data structures for the flash memory itself are managed by the simulator but not explicitly stored in flash or DRAM. In the case of the SunDisk SDP5 flash device, there is no need for additional data structures beyond what the file system already maintains for a magnetic disk and the flash disk maintains internally for block remapping. For the Intel flash card, the flash metadata includes state that must be frequently rewritten, such as linked lists.
- For the flash card, the simulator attempts to keep at least one segment erased at all times, unless erasures are done on an as-needed basis. One segment is filled completely before data blocks are written to a new segment. Erasures take place in parallel with reads and writes, being suspended during the actual I/O operations, unless a write occurs when no segment has erased blocks.

5 Results

We used the simulator to explore the architectural tradeoffs between disks, flash disks, and flash cards. We focussed on four issues: the basic energy and performance differences between the devices; the effect of storage utilization on flash energy consumption, performance, and endurance; the effect of combined writes and erasures on a flash disk; and the effect of buffer caches, both volatile and nonvolatile, on energy and performance.

5.1 Basic Comparisons

Tables 4(a)–(c) show for three traces and each device the energy consumed, and the average, mean, and standard deviations of the read and write response times. As mentioned in Section 4.2, the input parameters for each simulation were either based on measurements on the OmniBook (labeled "mea-

Device	Parameters	Energy (J)	Read Response (ms)			Write Response (ms)		
			Mean	Max	σ	Mean	Max	σ
CU140	measured	8,854	2.75	3535.3	50.5	0.93	3505.5	38.1
CU140	datasheet	8,751	2.04	3516.2	48.7	0.77	3493.6	37.8
KH	datasheet	9,945	8.70	1675.0	94.6	1.03	1536.2	30.2
SDP10	measured	1,516	0.50	1001.7	7.6	26.74	586.3	45.6
SDP5	datasheet	1,190	0.35	619.9	4.7	16.07	350.4	27.3
Intel flash card	measured	1,746	0.35	665.6	5.0	32.30	1787.9	78.8
Intel flash card	datasheet	888	0.12	105.2	0.9	5.65	147.3	9.9

(a) MAC trace

Device	Parameters	Energy (J)	Read Response (ms)			Write Response (ms)		
			Mean	Max	σ	Mean	Max	σ
CU140	measured	1,495	9.82	2746.1	58.7	0.42	5.6	0.4
CU140	datasheet	1,466	6.80	2717.6	57.4	0.42	5.6	0.4
KH	datasheet	1,786	17.35	1560.9	131.2	4.56	1476.5	77.3
SDP10	measured	733	2.94	120.2	5.6	36.60	317.6	19.7
SDP5	datasheet	606	1.98	77.5	3.6	21.88	190.6	11.8
Intel flash card	measured	731	1.96	80.8	3.8	38.41	939.0	21.5
Intel flash card	datasheet	451	0.51	17.0	0.8	7.85	459.7	5.2

(b) DOS trace

Device	Parameters	Energy (J)	Read Response (ms)			Write Response (ms)		
			Mean	Max	σ	Mean	Max	σ
CU140	measured	21,370	57.26	3537.4	145.3	30.46	3505.9	152.7
CU140	datasheet	20,659	38.65	3505.2	142.5	22.60	3475.1	151.6
KH	datasheet	28,887	81.96	1620.9	277.0	107.06	1552.9	362.2
SDP10	measured	4,972	10.50	40.4	6.9	138.96	5734.4	101.0
SDP5	datasheet	4,448	6.40	24.9	4.2	82.80	3412.5	60.1
Intel flash card	measured	3,865	6.58	24.8	4.4	155.52	7143.9	182.7
Intel flash card	datasheet	2,167	0.42	1.6	0.3	36.72	1922.9	118.5

(c) HP trace

Table 4: Comparison of energy consumption and response time for different devices, using the MAC, DOS, and HP traces. There was a 2-Mbyte DRAM buffer for MAC and DOS but no DRAM buffer cache in the HP simulations. Disk simulations spun down the disk after 5s of inactivity. Flash simulations were done with flash memory 80% utilized.

sured") or manufacturers' specifications (labeled "datasheet"). Note that it is not appropriate to compare response time numbers between the tables, because of the different mean transfer sizes of each trace. Simulations of the magnetic disks spun down the disk after 5s of inactivity, which is a good compromise between energy consumption and response time [5, 13]. Simulations using the flash card were done with the card 80% full.

Based solely on the input parameters from the datasheets, one may conclude that the Intel flash card consumes significantly less energy than either the Caviar Ultralite CU140 or the SunDisk SDP5. It provides better read performance than either of the other devices, and better write performance than the SunDisk SDP5, but much worse write performance than a Caviar Ultralite CU140 or KH with an SRAM write buffer. This latter discrepancy suggests that an SRAM write buffer is appropriate for flash memory as well, something that we have not explored so far but that is an integral part of the eNVy architecture [24].

When using the numbers for measured performance as input to the simulator, the flash card does not perform as well as the flash disk. In particular, its write performance is worse than the simulated write performance based on the SunDisk SDP10, across all three traces. This discrepancy suggests that when choosing between a flash disk emulator and a flash memory card, one must consider both the hardware and software characteristics of the environment.

We verified the simulator by running a 6-Mbyte synthetic trace both through the simulator and on the OmniBook, using each of the devices. The trace was smaller than the ones described above, in order to fit on the 10-Mbyte flash devices. We used the measured micro-benchmark performance to drive the simulator and then compared against actual performance. All simulated performance numbers were within a few percent of measured performance, with the exception of flash card reads and Caviar Ultralite CU140 writes. The measured mean performance for flash card reads was four times worse than the simulated performance; we believe this is due to overhead from cleaning and from decompression, which are more severe in practice than during the controlled experiments described in Section 3. Measured write performance for the CU140 was about twice as slow in practice as in simulation; we believe this is due to our optimistic assumption about avoiding seeks.

5.2 Flash Storage Utilization

For the Intel flash card, there is a substantial interaction between the storage utilization of flash memory and the behavior of the flash when the flash is frequently written. To examine this behavior, we simulated each trace with 40% to 95% of flash memory occupied by useful data. To do this, we set the size of the flash to be large relative to the size of the trace, then filled the flash with extra data blocks that reduced the amount of free space by an appropriate amount. Under low utilization, energy consumption and performance are fairly constant, but as the flash fills the behavior of the flash degrades, resulting in much greater energy consumption, worse performance, and more erasures per unit time (thus affecting flash endurance). This is because the system must copy "live" data from one erasure unit to another to free up an entire erasure unit. By comparison, the flash disk is unaffected by utilization because it does not copy data within the flash.

Figure 2 graphs simulated energy consumption and write response time as a function of storage utilization for each trace, using the specifications from the Intel flash card datasheet and a 2-Mbyte DRAM cache (no DRAM cache for the HP trace). At a utilization of 95%, compared to 40% utilization, the energy consumption rises by up to 150%, while the average write time increases up to 30%. For the MAC trace, the maximum number of erasures for any one segment over the course of the simulation increases from 7 to 34, while the mean erasure count goes up from 0.9 to 1.9 (110%). For the HP trace the erasure count tripled. Thus higher storage utilizations can result in "burning out" the flash two to three times faster under this workload.

In addition, experiments on the OmniBook demonstrated significant reductions in write throughput as flash memory was increasingly full. Figure 3 graphs instantaneous throughput as a function of cumulative data written, with three amounts of "live" data in the file system: 1 Mbyte, 9 Mbytes, and 9.5 Mbytes. Each data point corresponds to 1 Mbyte of data being overwritten, randomly selected within the total amount of live data. The flash card was erased completely prior to each experiment, so that any cleaning overhead would be due only to writes from the current experiment and the experiments would not interfere with each other. The drop in throughput over the course of the experiment is apparent for all three configurations, even the one with only 10% space utilization, presumably because of MFFS 2.00 overhead. However, throughput decreased much faster with increased space utilization.

5.3 Asynchronous Cleaning

The next generation of SunDisk flash products, the SDP5A, will have the ability to erase blocks prior to writing them, in order to get higher bandwidth during the write [3]. Erasure bandwidth is 150 Kbytes/s regardless of whether new data are written to the location being erased; however, if an area has been pre-erased, it can be written at 400 Kbytes/s. We simulated to compare the SDP5A with and without asynchronous cleaning. Asynchronous cleaning has minimal impact on energy consumption, but it decreases the average write time for each of the traces by 56–61%.

The improvement experienced by asynchronous erasure on the SunDisk demonstrates the effect of small erasure units on performance. Considering again the simulated write response of the SunDisk SDP5 and Intel flash card shown in Tables 4(a)–(c), if the SunDisk SDP5 write response decreased by 60% it would be comparable to the flash card. But as storage utilization increases, flash card write performance will degrade although the performance of the flash disk will remain constant.

5.4 DRAM Caching

Since flash provides better read performance than disk, the dynamics of using DRAM for caching file data change. DRAM provides better performance than flash but requires more power and is volatile. Unlike flash memory, DRAM consumes significant energy even when not being accessed. Thus, while extremely "hot" read-only data should be kept in DRAM to get the best read performance possible, other data can remain in flash rather than DRAM. One may therefore ask whether it is better to spend money on additional DRAM or additional flash. In order to evaluate these trade-offs, we simulated configurations with varying amounts of DRAM buffer cache and flash memory. (As is the case with all our simulations, they do not take into account DRAM that is used for other purposes such as program execution.) We began with the premise that a system stored 32 Mbytes of data, not all of which necessarily would be accessed, and considered hypothetical flash devices storing from 34–38 Mbytes of data. (Thus total storage utilization ranged from 94% with 34 Mbytes of storage down to 84% with 38 Mbytes.) In addition, the system could have from 0–4 Mbytes of DRAM for caching.

Figure 4 shows the results of these simulations, run against the DOS trace using specifications from the datasheets. For the Intel flash card, increasing the amount of flash available by 1 Mbyte, thereby decreasing storage utilization from 94.1% to 91.4%, reduces energy consumption by 25% and average over-all response time by 18%. The incremental benefit on energy consumption of additional flash beyond the first Mbyte is minimal, though adding flash does help to reduce response time. Adding DRAM to the Intel flash card increases the energy used for DRAM without any appreciable benefits: the time to read a block from flash is barely more than the time to read it from DRAM.

Only one curve is shown for the SunDisk SDP5 because increasing the size of the flash disk has minimal effect on energy consumption or performance. In fact, for this trace, even a 500-Kbyte DRAM cache increases energy consumption for the SunDisk SDP5 without improving performance. With the MAC trace, which has a greater fraction of reads, a small DRAM cache improves energy consumption and performance for the SunDisk SDP5, while the Intel flash card shows a less pronounced benefit from lower utilization. Thus the tradeoff between DRAM and flash size is dependent both on execution characteristics (read/write ratio) and hardware characteristics (the difference in performance between DRAM and the flash device).

5.5 NVRAM Caching

So far we have assumed that magnetic disks are configured with an SRAM write buffer that allows the disk to stay spun down if a small write is issued. In practice, SRAM write buffers for magnetic disks are relatively commonplace, though we are unaware of products other than the Quantum Daytona that use the write buffer to avoid spinning up an idle disk. Here we examine the impact of nonvolatile memory on write performance, the effects of deferring spin-up, and the cost-effectiveness of the write buffer. We base our results on a NEC 32Kx8-bit SRAM chip, part μPD43256B, with a 55ns access time [18]. We assume that writes to SRAM can be recovered after a crash, so synchronous writes that fit in SRAM are made asynchronous with respect to the disk.

A 32-Kbyte SRAM write buffer costs only a few dollars, which is a small part of the total cost of a disk system. Under light load, this buffer can make a significant difference in the average write response time, compared to a system that writes all data synchronously to disk. Although SRAM consumes significant energy itself, by reducing the number of times the disk spins up, the SRAM buffer can potentially conserve energy. However, if writes are large or are clustered in time, such that the write buffer frequently fills, then many writes will be delayed as they wait for the disk. In this case, a larger SRAM buffer will be necessary to improve performance, and it will cost more money and consume more energy.

Figure 5 graphs normalized energy consumption and write response time as a function of SRAM size for each of the traces. The values are normalized to the case without an SRAM buffer. As with the other experiments, DRAM was fixed at 2 Mbytes for MAC and DOS and not used for HP;[3] the spin-down threshold was fixed at 5s. For the first two traces, using a 32-Kbyte SRAM buffer improves average write response by a factor of 20 or more, with no difference from larger buffers; for the HP trace a 32-Kbyte buffer only halves the average write response time, but a 512-Kbyte buffer reduces it by another 20%. A small SRAM buffer reduces energy by a much less dramatic amount: 21% for the MAC trace, 15% for DOS, and just 4% for HP, with another 4% reduction with 512 Kbytes of SRAM.

6 Related Work

In addition to the specific work on flash file systems mentioned previously, the research community has begun to explore the use of flash memory as a substitute for, or an addition to, magnetic disks. Cáceres et al. pro-

[3] For this experiment, one should discount the result from the HP trace by comparison to the other two traces. This is because the HP simulation has no DRAM cache, so reads cause the disk to spin up more than with the other simulations (except those reads that are serviced from recent writes to SRAM). The effect of SRAM on energy and response time in the HP environment bears further study.

posed operating system techniques for exploiting the superior read performance of flash memory while hiding its poor write performance, particularly in a portable computer where all of DRAM is battery-backed [2]. Wu and Zwaenepoel discussed how to implement and manage a large non-volatile storage system, called eNVy, composed of NVRAM and flash memory for high-performance transaction processing. They simulated a system with Gbytes of flash memory and Mbytes of battery-backed SRAM, showing it could support the I/O corresponding to 30,000 transactions per second using the TPC-A database benchmark [23]. They found that at a utilization of 80%, 45% of the time is spent erasing or copying data within flash, while performance was severely degraded at higher utilizations [24]. Marsh et al. examined the use of flash memory as a cache for disk blocks to avoid accessing the magnetic disk, thus allowing the disk to be spun down more of the time [15]. SunDisk recently performed a competitive analysis of several types of flash memory on an HP Omnibook 300 and found that the SunDisk SDP5-10 flash disk emulator was nearly an order of magnitude faster than an Intel Flash card using version 2 of the Flash Files System [22]. They also found that performance of the Intel Flash card degraded by 40% as it filled with data, with the most noticeable degradation between 95% and 99% storage utilization.

Other researchers have explored the idea of using non-volatile memory to reduce write traffic to disk. Baker et al. found that some 78% of blocks written to disk were done so for reliability. They found that a small amount of NVRAM on each client was able to reduce client-server file write traffic by half, and NVRAM on the file server could reduce writes to disk by 20% [1]. However, the benefits of NVRAM for workstation clients did not justify its additional cost, which would be better applied toward additional DRAM. This contrasts with our results for a mobile environment, in which larger amounts of DRAM are not so cost effective, but a small amount of NVRAM helps energy consumption and performance. Ruemmler and Wilkes also studied how well NVRAM could absorb write traffic, finding that 4 Mbytes of NVRAM was sufficient to absorb 95% of all write traffic in the systems they traced [20].

Finally, segment cleaning in Rosenblum and Ousterhout's Log-Structured File System (LFS) [19] has a number of similarities to flash cleaning when the flash segment size is a large multiple of the smallest block size. The purpose of Sprite LFS is to amortize write overhead by writing large amounts of data at once; to do so requires that large amounts of contiguous disk space be emptied prior to a write. However, cleaning in LFS is intended to amortize the cost of seeking between segments anywhere on the disk, while flash cleaning is a requirement of the hardware. Kawaguchi et al. [10] recently designed a flash memory file system for UNIX based on LFS, with performance comparable to the 4.4BSD Pageable Memory based File System [16]. They found that cleaning overhead did not significantly affect performance, but they need more experience with cleaning under heavier loads.

7 Conclusions

In this paper we have examined three alternatives for file storage on mobile computers: a magnetic disk, a flash disk emulator, and a flash memory card. We have shown that either form of flash memory is an attractive alternative to magnetic disk for file storage on mobile computers. Flash offers low energy consumption, good read performance, and acceptable write performance.

The main disadvantage of using magnetic disk for file storage on mobile computers is its great energy consumption. To extend battery life, the power management of a disk file system spins down the disk when it is idle. But even with power management, a disk file system can consume an order of magnitude more energy than a file system using flash memory.

Our trace simulation results, using a SunDisk SDP5 and a Caviar Ultralite CU140, show that the flash disk file system can save 59–86% of the energy of the disk file system. It is 3–6 times faster for reads, but its mean write response is a minimum of four times worse. Adding a nonvolatile SRAM write buffer to a flash disk should enable it to compete with newer magnetic disks that are coupled with SRAM buffers.

The flash memory file system (using the Intel flash card) has the most attractive qualities with respect to energy and performance, though its price and capacity limitations are still drawbacks. Even in the presence of disk power management, the flash memory file system can save 90% of the energy of the disk file system, extending battery life by 20–100%. Furthermore, in theory the flash memory file system can provide mean read response time that is up to two orders of magnitude faster than the disk file system. However, its mean write response time varies from 50% to an order of magnitude worse than a CU140 magnetic disk with an SRAM write buffer. Again, adding SRAM to flash should dramatically improve performance, except in situations where flash performance is dominated by cleaning costs.

In practice, hardware measurements showed that there is a great discrepancy between the rated performance of each of the storage media and their performance in practice under DOS. This is especially true with the flash card using MFFS 2.00, whose write performance degrades linearly with the size of the file. Some of the differences in performance can be reduced with new technologies, in both hardware and software. One new technique is to separate the write and erase operations on a flash disk emulator, as the next generation of the SunDisk flash disk will allow. Another hardware technique is to allow erasure of more of a flash memory card in parallel, as the newer 16-Mbit Intel flash devices allow [9]. Newer versions of the Microsoft Flash File System should address the degradation imposed by large files, and in order to take advantage of asynchronous flash disk erasure, file systems for mobile computers must treat the flash disk more like a flash card than like a magnetic disk.

Finally, in our simulation study, we found that the erasure unit of flash memory, which is fixed by the hardware manufacturer, can significantly influence file system performance. Large erasure units require a low space utilization. At 90% utilization or above, an erasure unit that is much larger than the file system block size will result in unnecessary copying, degrading performance, wasting energy, and wearing out the flash device. In our simulations, energy consumption rose by as much as 190%, the average write response increased up to 30%, and the rate of erasure as much as tripled. Flash memory that is more like the flash disk emulator, with small erasure units that are immune to storage utilization effects, will likely grow in popularity despite being at a disadvantage in basic power and performance.

Acknowledgments

We are grateful to P. Krishnan, who did much of the work on the storage simulator used in this study. We also thank W. Sproule and B. Zenel for their efforts in gathering trace data and/or hardware measurements. J. Wilkes at Hewlett-Packard and K. Li at U.C. Berkeley graciously made their file system traces available. Thanks to R. Alonso, M. Dahlin, C. Dingman, B. Krishnamurthy, P. Krishnan, D. Milojičić, C. Northrup, J. Sandberg, D. Stodolsky, B. Zenel, W. Zwaenepoel, and anonymous reviewers for comments on previous drafts. We thank the following persons for helpful information about their products: A. Elliott and C. Mayes of Hewlett-Packard; B. Dipert and M. Levy of Intel; and J. Craig, S. Gross, and L. Seva of SunDisk.

The *Moby-Dick* text we used for compression-related experiments was obtained from the Guttenberg Project at the University of Illinois, as prepared by Professor E. F. Irey from the Hendricks House edition.

Macintosh and PowerBook are trademarks of Apple Corporation. Kittyhawk, OmniBook, and HP-UX are trademarks of Hewlett-Packard Company. Microsoft, MS-DOS, and Windows are trademarks of Microsoft Corporation. Stacker is a trademark of Stac Electronics. Caviar is a trademark of Western Digital. UNIX is a trademark of X/Open.

References

[1] Mary Baker, Satoshi Asami, Etienne Deprit, John Ousterhout, and Margo Seltzer. Non-volatile memory for fast, reliable file systems. In *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 10–22, Boston, MA, October 1992. ACM.

[2] Ramón Cáceres, Fred Douglis, Kai Li, and Brian Marsh. Operating Systems Implications of Solid-State Mobile Computers. In *Proceedings of the Fourth Workshop on Workstation Operating Systems*, pages 21–27, Napa, CA, October 1993. IEEE.

[3] Jeff Craig, March 1994. Personal communication.

[4] Brian Dipert and Markus Levy. *Designing with Flash Memory.* Annabooks, 1993.

[5] Fred Douglis, P. Krishnan, and Brian Marsh. Thwarting the Power Hungry Disk. In *Proceedings of 1994 Winter USENIX Conference*, pages 293–306, San Francisco, CA, January 1994.

[6] Hewlett-Packard. *HP 100 and OmniBook Flash Disk Card User's Guide*, 1993.

[7] Hewlett-Packard. *Kittyhawk HP C3013A/C3014A Personal Storage Modules Technical Reference Manual*, March 1993. HP Part No. 5961-4343.

[8] Intel. *Mobile Computer Products*, 1993.

[9] Intel. *Flash Memory*, 1994.

[10] Atsuo Kawaguchi, Shingo Nishioka, and Hiroshi Motoda. A flash-memory based file system. In *Proceedings of the USENIX 1995 Winter Conference*, New Orleans, January 1995. To appear.

[11] Markus Levy. Interfacing Microsoft's Flash File System. In *Memory Products*, pages 4-318–4-325. Intel Corp., 1993.

[12] Kester Li. Towards a low power file system. Technical Report UCB/CSD 94/814, University of California, Berkeley, CA, May 1994. Masters Thesis.

[13] Kester Li, Roger Kumpf, Paul Horton, and Thomas Anderson. A Quantitative Analysis of Disk Drive Power Management in Portable Computers. In *Proceedings of the 1994 Winter USENIX*, pages 279–291, San Francisco, CA, 1994.

[14] B. Marsh and B. Zenel. Power Measurements of Typical Notebook Computers. Technical Report 110-94, Matsushita Information Technology Laboratory, May 1994.

[15] Brian Marsh, Fred Douglis, and P. Krishnan. Flash Memory File Caching for Mobile Computers. In *Proceedings of the 27th Hawaii Conference on Systems Sciences*, pages 451–460, Maui, HI, 1994. IEEE.

[16] Marshall Kirk McKusick, Michael J. Karels, and Keith Bostic. A pageable memory based file system. In *USENIX Conference Proceedings*, pages 137–144, Anaheim, CA, Summer 1990. USENIX.

[17] NEC. *Memory Products Data Book, Volume 1: DRAMS, DRAM Modules, Video RAMS*, 1993.

[18] NEC. *Memory Products Data Book, Volume 2: SRAMS, ASMs, EEPROMs*, 1993.

[19] Mendel Rosenblum and John Ousterhout. The design and implementation of a log-structured file system. *ACM Transactions on Computer Systems*, 10(1):26–52, February 1992. Also appears in Proceedings of the 13th Symposium on Operating Systems Principles, October 1991.

[20] Chris Ruemmler and John Wilkes. UNIX disk access patterns. In *Proceedings of*

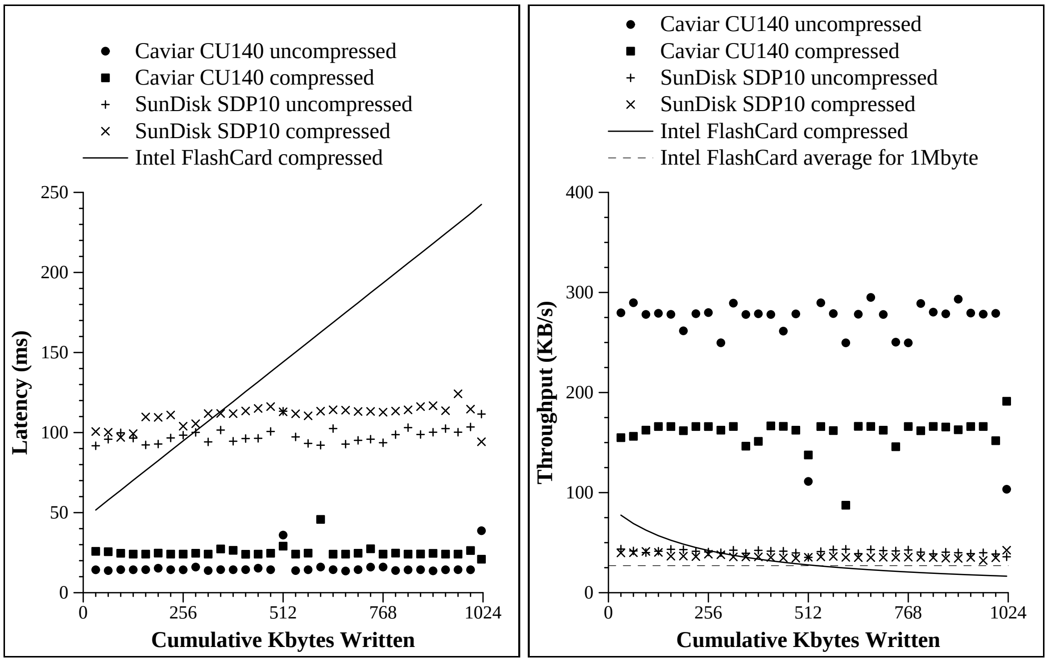
the Winter 1993 USENIX Conference, pages 405–420, San Diego, January 1993.

[21] SunDisk Corporation. *SunDisk SDP Series OEM Manual*, 1993.

[22] SunDisk Corporation, 3270 Jay Street, Santa Clara, CA 95054. *Competitive Analysis 80-40-00002 Rev. 1.0*, 1994.

[23] Transaction Processing Performance Council. *TPC Benchmark A Standard Specification Rev 1.1*.

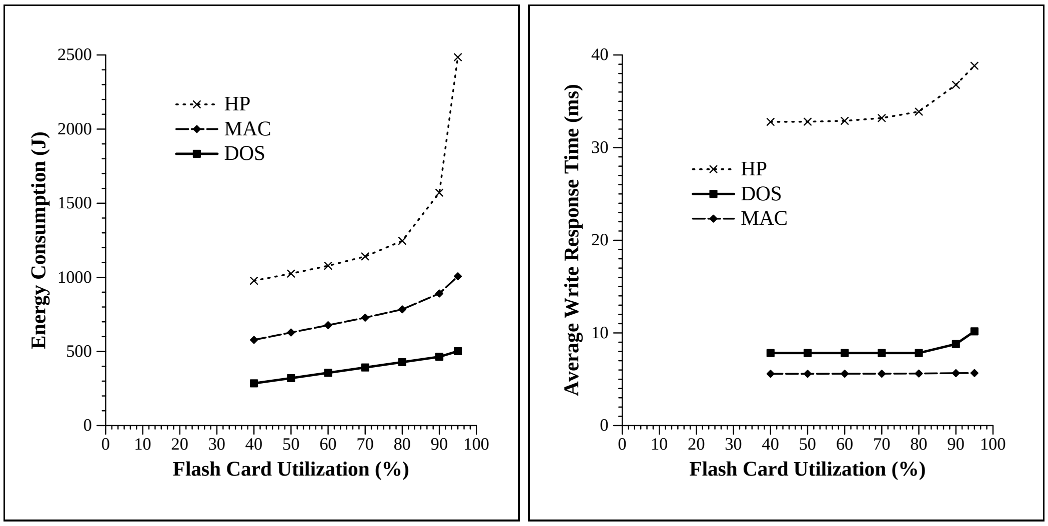
[24] Michael Wu and Willy Zwaenepoel. eNVy: a Non-Volatile, main memory storage system. In *Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems*, San Jose, CA, October 1994. To appear.



(a) Write latency.

(b) Write throughput.

Figure 1: Measured latency and instantaneous throughput for 4-Kbyte writes to a 1-Mbyte file. To smooth the latency when writing via DoubleSpace or Stacker, points were taken by averaging across 32-Kbytes of writes. Latency for an Intel flash card running the Microsoft Flash File System, as a function of cumulative data written, increases linearly. Though writes to the first part of the file are faster for the flash card than for the flash disk, the average throughput across the entire 1-Mbyte write is slightly worse for the flash card. The flash card was erased prior to each experiment. Also, because the CU140 was continuously accessed, the disk spun throughout the experiment.



(d) Energy consumption.

(e) Response time.

Figure 2: Energy and write response time as a function of flash storage utilization, simulated based on the datasheet for the Intel flash card, with a segment size of 128 Kbytes. Each of the traces is shown. Energy consumption increases steadily for each of the traces, due to increased cleaning overhead, but the energy consumed by the HP trace increases the most dramatically with high utilization. Write response time holds steady until utilization is high enough for writes to be deferred while waiting for a clean segment; even so, the MAC trace has constant mean write response. It has a higher fraction of reads, so the cleaner keeps up with writes more easily. The size of the DRAM buffer cache was 2 Mbytes for MAC and DOS and no DRAM was used for HP.

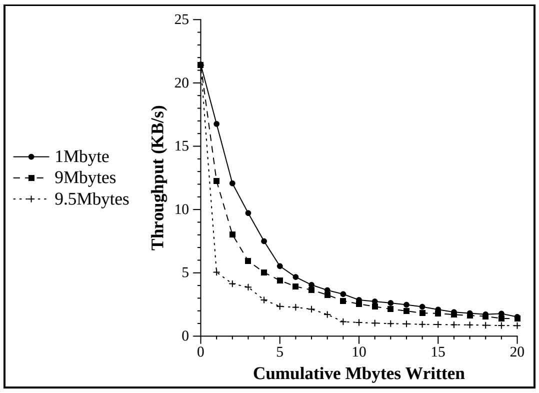
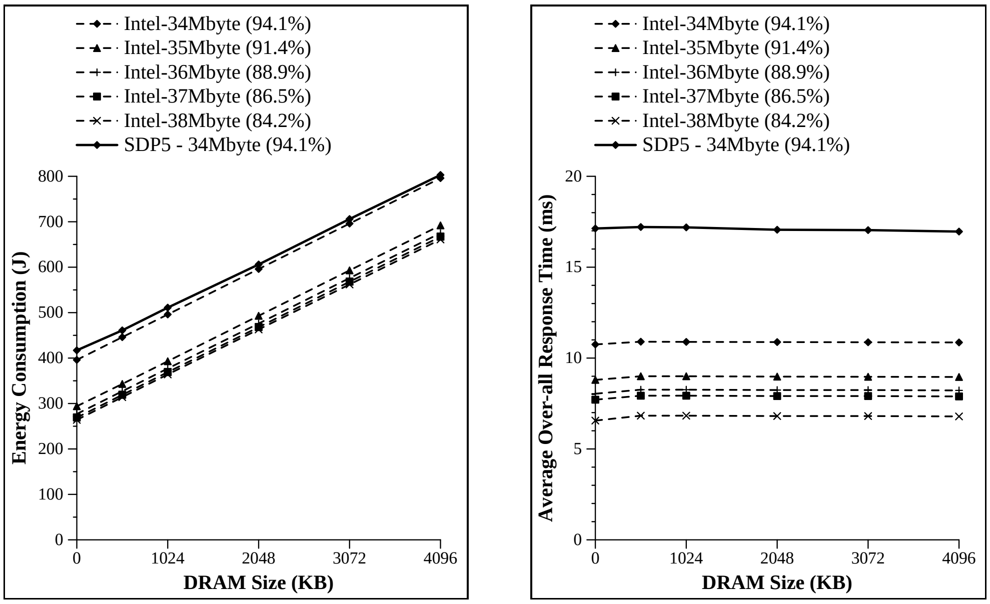


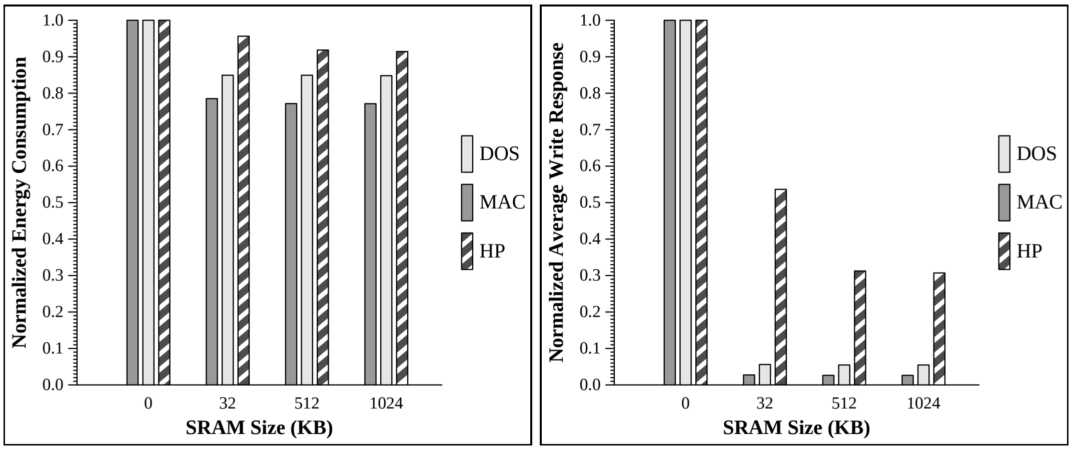
Figure 3: Measured throughput on an OmniBook using a 10-Mbyte Intel flash card, for each of 20 1-Mbyte writes (4 Kbytes at a time). Different curves show varying amounts of live data. Throughput drops both with more cumulative data and with more storage consumed.



(a) Energy consumption as a function of DRAM size and flash size.

(b) Response time as a function of DRAM size and flash size.

Figure 4: Energy consumption and average over-all response time as a function of DRAM size and flash size, simulated for the DOS trace. We simulated multiple flash sizes for the Intel flash card, which shows a benefit once it gets below 80% utilization. Each line represents a 1-Mbyte differential in flash card size, similar to moving along the x-axis by 1 Mbyte of DRAM. Increasing the DRAM buffer size has no benefit for the Intel card. The SunDisk has no benefit due to increased flash size (not shown), and here for this trace it shows no benefit from a larger buffer cache either.



(a) Energy consumption.

(b) Response time.

Figure 5: Normalized energy and write response time as a function of SRAM size for each trace. Results are normalized to the value corresponding to no SRAM. While a 32-Kbyte SRAM write buffer improves energy and response time for each of the traces, the improvement is more significant for MAC and DOS than for HP. Only the HP trace significantly benefits from an SRAM cache larger than 32 Kbytes. Disks were spun down after 5s of inactivity. The size of the DRAM buffer cache was 2 Mbytes for MAC and DOS and and no DRAM was used for HP.