Microsoft Research Ltd. Technical Report MSR-TR-2008-169, November 2008

Migrating enterprise storage to SSDs: analysis of tradeoffs

Dushyanth Narayanan, Eno Thereska, Austin Donnelly, Sameh Elnikety, Antony Rowstron
Microsoft Research Cambridge, UK
dnarayan,etheres,austind,samehe,antr@microsoft.com

Abstract

Recently, flash-based solid-state drives (SSDs) have become standard options for laptop and desktop storage, but their impact on enterprises has not been studied. Provisioning enterprise storage is challenging. It requires optimizing for the performance, capacity, power and reliability needs of the expected workload, all while minimizing financial costs.

This paper, through analysis of a number of enterprise workloads, provides insights as to when, and how, SSDs should be incorporated into the enterprise storage hierarchy. We describe an automated tool that, given device models and a block-level trace of a workload, determines the least-cost storage configuration. It analyzes the factors that drive the configuration choice, and computes the price points at which different SSD-based solutions will become cost-effective.

Our optimization framework is flexible and can be used to design a range of storage hierarchies. When applied to current workloads and prices we find the following in a nutshell: for many enterprise workloads capacity dominates provisioning costs and the current per-gigabyte price of SSDs is between a factor of 3 and 3000 times higher than needed to be cost-effective for full replacement. We find that SSDs can provide some benefit as an intermediate tier for caching and write-ahead logging in a hybrid disk-SSD configuration. Surprisingly, the power savings achieved by SSDs are comparable to power savings from using low-power SATA disks.

1. Introduction

Solid-state drives (SSDs) are rapidly making inroads into the laptop market and are likely to encroach on the desktop storage market as well. How should this technology be incorporated into enterprise storage? Enterprise storage is complex and different, both in scale and requirements, from laptop and desktop storage. It must meet workload requirements — which vary widely — in terms of capacity, performance, and reliability while minimizing cost.

While solid-state storage has advantages such as fast random-access reads and low power consumption, it also has disadvantages such as high cost per gigabyte. Recently there have been many proposals on exploiting these characteristics of solid-state storage to redesign storage systems, either replacing disks entirely (Prabhakaran et al. 2008; Woodhouse 2001) or using solid-state storage to augment disk storage, e.g., to store file system metadata (Miller et al. 2001). However, there has been no analysis of the costs and benefits of these architectures for real enterprise workloads.

Intuitively, SSDs are favored by workloads with a high demand for random-access I/Os, especially reads, and a relatively low capacity requirement. However, this observation by itself ignores two considerations. First, for any given workload we need to quantify the "IOPS requirement" as well as the capacity, bandwidth, and fault-tolerance requirements and then evaluate the tradeoffs between device types. Second, "performance at any cost" is a luxury that enterprises can rarely afford to pursue. Hence, in general, performance must be scaled appropriately by the dollar cost.

In this paper we take the point of view of a storage administrator who wishes to re-provision or upgrade a set of storage volumes. This administrator must answer two questions. First, what are the capacity, performance, and fault-tolerance requirements of the workload? Second, what is the least-cost configuration that will satisfy the requirements? Since this paper is focused on the tradeoffs between SSDs and mechanical disks, we consider three types of configuration for each volume: disk-only, SSD-only, and hybrid.

1.1 Goals, non-goals and contributions

The main contribution of this paper is an analysis of the SSD/disk tradeoff for real enterprise workload traces. We show that the benefits of SSDs are workload-dependent, and give an understanding of the parameter space for the tradeoffs involved.

An additional contribution is our use of a modeling and optimization approach to answer questions about SSDs. Optimization and solvers combined with performance models have been used previously to automate storage provisioning (Anderson et al. 2005; Strunk et al. 2008), but to our knowledge this is the first work to use them specifically to examine the use of SSDs. Our contribution lies in the novelty and the importance of this new application domain, rather than on our optimization techniques or performance models, which are simple first-order models.

It is important to note that our analysis does not attempt to forecast price trends, which are driven by both economic and technological forces. Instead we evaluate different SSD-based configurations for several real enterprise workloads at today's prices; and we also compute the price points at which SSD-based configurations that are currently too expensive will become cost-competitive in the future. We also note that disks and SSDs are only part of the cost of supporting enterprise storage; other costs include networking, enclosures, cooling, etc. and are not addressed in this paper.

Broadly we found that SSDs will fully replace disks once the SSDs' cost per GB drops by 1–3 orders of magnitude. Waiting for full replacement is not necessary, however. Using the solid-state storage as a caching tier is more promising in the short to medium-term: a small amount of solid-state storage used as a read cache benefits up to 25% of our traced workloads, and for 10% of them this can be done cost-effectively already at today's SSD prices. Solid-state storage is very effective as a write-ahead log, potentially reducing write response times and requiring only modest capacity and bandwidth. However, current flash-based devices wear out after a certain number of writes per block, and we found this to be a concern when absorbing an entire volume's writes on a small solid-state device.

Surprisingly, we also found that although SSD-based solutions consume less power than enterprise disk based solutions, the resulting savings are 1–3 orders of magnitude lower than the initial investment required. Hence, power savings are unlikely to be the main motivation in moving from disks to SSDs. We found low-speed SATA disks to be competitive with SSDs in terms of performance and capacity per watt. Interestingly these disks also offer much higher capacity and performance per dollar than either enterprise disks or SSDs. However, they are generally believed to have lower reliability and the tradeoff between cost and reliability requires further retrospective studies.

2. Background and related work

This section provides a brief overview of solid-state drives. It then describes the enterprise workload traces that drive our analysis. It concludes with related work.

2.1 Solid-state drives

Solid-state drives (SSDs) provides durable storage through a standard block I/O interface such as SCSI or SATA. They have no mechanical moving parts and hence no positioning delays. Access latencies are sub-millisecond, compared to many milliseconds for disks. Commercially available SSDs today are based on NAND flash memory (henceforth referred to as "flash"). While other solid-state technologies have been proposed, such as magnetic RAM (M-RAM) or phase change memory (PCM) (Intel News Release 2008), these are not commercially available yet and their cost/performance tradeoffs are unknown. The analysis in this paper uses device models extracted from flash-based SSDs; our optimization framework would apply equally to other solid-state technologies.

Server	**Function**	**Vols**	**GB**
usr	User home dirs	3	1367
proj	Project dirs	5	2094
print	Print server	2	452
hwm	Hardware monitoring	2	39
resproj	Research projects	3	277
proxy	Firewall/web proxy	2	89
src1	Source control	3	555
src2	Source control	3	355
webstg	Web staging	2	113
term	Terminal server	1	22
websql	Web/SQL server	4	441
media	Media server	2	509
webdev	Test web server	4	136
exchange	Corporate mail	9	6706

Table 1. Enterprise servers traced.

NAND flash memory has two unusual limitations, although both of these can be mitigated by using layout remapping techniques at the controller or file system level. The first limitation is that small, in-place updates are inefficient, due to the need to erase the flash in large (64–128 KB) units before it can be rewritten. This property results in poor random-access write performance at the block interface level, for the current generation of SSDs. However, with appropriate remapping techniques (Agrawal et al. 2008; Birrell et al. 2007; Woodhouse 2001), random-access writes can be converted to sequential writes, which have good performance on SSDs. The second limitation of NAND flash is *wear*: the reliability of the flash degrades after many repeated write-erase cycles. Most commercially available flash products are rated to withstand 10,000–100,000 write-erase cycles. Using one of a variety of wear-leveling algorithms (Gal and Toledo 2005), the wear can be spread evenly across the flash memory to maximize the lifetime of the device as a whole. These algorithms impose a small overhead in the form of additional writes for compaction and defragmentation; however, to a first order, we can assume that a flash device does not wear out until the amount of data written to it equals the size of the flash memory multiplied by the number of rated cycles.

2.2 Enterprise workload traces

Our analysis is explained in the context of real enterprise workload traces. These traces were collected at the block level, below the file system buffer cache but above the storage tier. The traces covered 45 volumes across 14 servers. Table 1 shows the servers traced, and the number of volumes and total storage capacity for each.

The traces were collected from two different data centers. The first 13 servers shown in Table 1 are in a small data center servicing about 100 on-site users (Narayanan et al. 2008a), and covering a range of services typical of small to medium enterprise data centers: file servers, database servers, caching web proxies, etc. Each server was configured with a RAID-1 system volume and one or more RAID-5 data volumes: in all, the 36 volumes traced contained 179 disks. These traces cover every block-level I/O request serviced by each volume on each server for a period of one week, starting from 5pm GMT on the 22nd February 2007. The number of requests traced was 434 million, of which 70% were reads.

The second set of traces are from the last server listed in Table 1: a production Exchange e-mail server serving around 5000 users (Worthington and Kavalanekar 2008). They cover 9 volumes over a period of 1 day (starting from 10:39pm GMT on the 12th December 2007). Each volume on this server is configured as a RAID-10 array, except for the system volume which is configured as a RAID-1 array. In total there were 102 disks in the Exchange volumes. The traces contain 61 million requests, of which 43% are reads.

We note that there are other kinds of enterprise workloads (e.g., OLTP) not captured by our traces. Our optimization framework can be applied to block-level traces of any workload and our tool can automatically perform the analysis.

2.3 Related work

There has been considerable research both in optimizing solid-state storage systems and into hybrid systems with both solid-state and disk storage. For example, there have been proposals for optimizing B-trees (Nath and Kansal 2007; Wu et al. 2004) and hash-based indexes (Zeinalipour-Yazti et al. 2005) for flash-based storage in embedded systems; to partition databases between flash and disks (Koltsidas and Viglas 2008); to optimize database layout for flash storage (Lee et al. 2008); to use flash to replace some of the main memory buffer cache (Kgil and Mudge 2006); to use M-RAM to store file system metadata (Miller et al. 2001); and to build persistent byte-addressable memory (Wu and Zwaenepoel 1994). These studies tend to emphasis “better performance at all cost”; ours is a tradeoff analysis of multiple metrics normalized by dollar and power cost. Furthermore, we move beyond benchmarks and use real enterprise workloads in our evaluation.

In the laptop storage space, vendors are also manufacturing hybrid disks (Samsung 2006) that incorporate a small amount of flash storage within the disk. Windows Vista’s ReadyDrive technology (Panabaker 2006) makes use of these hybrid drives to speed up boot and application launch, as well as to save energy by spinning the disk down.

The closest work to ours in its aims is that by Baker et al. (Baker et al. 1992) exploring the use of battery-backed non-volatile RAM (NVRAM) in the Sprite distributed file system to reduce write traffic to disks. NVRAM is widely adopted in enterprise storage, especially in high-end disk array controllers, but due to high costs is usually deployed in small sizes. The analysis was based on traces collected in 1991 on four file servers running a log-structured file system serving 40 disk-less clients in a university research environment. Enterprise storage workloads in 2007, such as those used in this paper, are substantially different: they include web servers, web caches, and database back ends for web services, with orders of magnitude more storage than in 1991. Furthermore, flash memory today is an order of magnitude cheaper than NVRAM and raises the possibility of replacing disks entirely. This paper explores the new space of workloads and technologies.

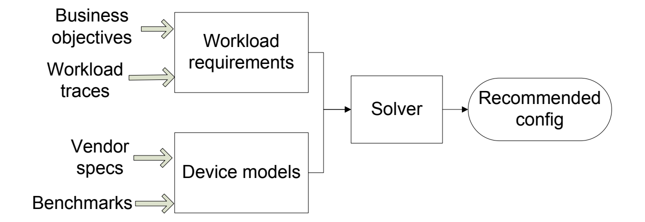


Figure 1. Tool steps

3. Modeling the solution space

This section outlines the modeling and solver framework that is used to analyze the solution space of having SSDs in a storage architecture. Figure 1 illustrates the basic steps taken in coming up with a solution. First, *workload requirements* are collected. Some of them may be easily expressed in terms of business objectives, e.g., nines for availability. Others, such as performance expectations, can be automatically extracted by our tool from workload I/O traces. Next, *device models* are created for the hardware under consideration. Such models capture device characteristics like dollar cost, power consumption, device reliability (usually reported by the vendor, and then refined over time as more empirical data is available (Schroeder and Gibson 2007)), and performance, which our tool automatically extracts using synthetic micro-benchmarks. Finally, a solver component finds a configuration that minimizes cost while meeting the other objectives. The configuration is either a single-tier configuration — an array of devices of some type — or a two-tier configuration where the top and bottom tiers could use different device types (e.g., SSDs and disks).

3.1 Extracting workload requirements

Workload requirements make up the first set of inputs to the solver. Table 2 lists the requirements used in this paper. Several of them, such as capacity, availability and reliability can be straightforward to specify in a service level objective. Performance, however, can be more difficult. Our tool helps an administrator understand a workload’s inherent performance requirements by extracting historical performance metrics.

A workload could be characterized in terms of its mean and peak request rates; its read/write ratio; and its ratio of sequential to non-sequential requests. However, just using these aggregate measures can hide correlations between these workload attributes, e.g., a workload might have many

Metric	Unit
Capacity	GB
Random-access reads	IOPS
Random-access writes	IOPS
Random-access I/Os	IOPS
Sequential reads	MB/s
Sequential writes	MB/s
Availability	Number of nines
Reliability	Number of nines

Table 2. Workload requirements.

random-access reads and streaming writes but few streaming reads and random-access writes. Hence, we prefer to consider random-access reads, random-access writes, sequential reads, and sequential writes each as a separate workload requirement. In addition we consider the total random-access I/O rate: for mixed read/write workloads, this could be higher than the individual read or write I/O rates.

Many workloads, including the ones analyzed in this paper, have time-varying load, e.g., due to diurnal patterns. Hence, provisioning storage for the mean request rates will be inadequate. Our models primarily consider percentiles of offered load rather than means. By default we use the peak request rate, i.e. the 100^{th} percentile, but other percentiles, such as the 95^{th} can also be used. All the analyses in this paper use peak sequential transfer and non-sequential I/O rates on a time scale of 1 min.

Computing the peak read (or write) transfer bandwidth from a trace is straightforward: it is the maximum read (or write) rate in MB/s observed over any minute in the trace. Usually, this corresponds to a period of sequential transfer. If the workload has no sequential runs this will reflect a lower rate of transfer, indicating that sequential transfer is not a limiting factor for this workload.

When computing the "IOPS requirement" of a workload, we must take care to filter out sequential or near-sequential runs. Since mechanical disks in particular can sustain a much higher rate of sequential I/Os than random-access ones, sequential runs could cause an overestimate of the true random-access IOPS requirement of the workload. In general, the locality pattern of a sequence of I/Os could fall anywhere between completely random and completely sequential: however, to keep the models simple we choose to classify each I/O in the workload trace as either sequential or non-sequential. We use LBN distance between successively completed I/Os to classify the I/Os: any I/O that is within 512 KB of the preceding I/O is classified as sequential. This threshold is small compared to the typical disk track size, but large enough to correctly detect sequential readahead, which in Windows could have a small amount of reordering. The read, write, and total IOPS requirements of the workloads are then based on the non-sequential I/O rates averaged over a 1 min time scale.

Metric	Unit
Capacity	GB
Performance	IOPS (read, write)
	MB/s (read, write)
Reliability	MTBF
Wear rate(SSDs)	GB/day
Power consumption	W (idle and active)
Purchase cost	$

Table 3. Device model attributes.

3.2 Device models

Device models make up the second set of inputs to the solver. Table 3 shows the metrics our tool uses to characterize a device. The models are empirical rather than analytical. Our tool runs a sequence of synthetic tests to extract all the performance attributes. Sequential performance is measured using sequential transfers (for disks, requests are issued on the outermost tracks and innermost tracks — the difference in bandwidth can be as high as 50%. However, we only report on bandwidth on the outermost tracks in this paper). Random-access performance is based on issuing concurrent 4 KB requests uniformly distributed over the device, with a concurrency level of 256. The aim is to measure the number of IOPS that the device can sustain under high load. All synthetic tests are short in duration, around 20 seconds. We found this sufficient to verify what vendors were reporting.

SSD random-access writes Our tool considers read and write performance as separate metrics. In general, mechanical disks have equivalent read and write performance. However, current SSDs perform much worse on random-access writes than random-access reads or sequential I/O. This problem can be solved by making these writes sequential through block-level remapping (Agrawal et al. 2008) or a log-structured file system (Rosenblum and Ousterhout 1991; Woodhouse 2001). We have also implemented a block-level remapping scheme (Narayanan et al. 2008b). Since SSDs do not have any positioning delays, read performance is not affected by these layout changes. Some overhead is added for garbage-collection or log cleaning: enterprise workloads in general have sufficient idle time due to diurnal patterns to perform these maintenance tasks without impacting foreground workloads (Golding et al. 1995; Narayanan et al. 2008b). For completeness we included a simple "log-structured SSD" model in our tool, which assumes all writes are made sequential and that log cleaning is a free background activity.

Scaling Enterprise storage volumes usually consist of multiple homogeneous devices in a RAID configuration. This gives fault-tolerance for a configurable number of disk failures (usually one) as well as additional capacity and performance. Our device models are based on the assumption that both capacity and performance will scale linearly with

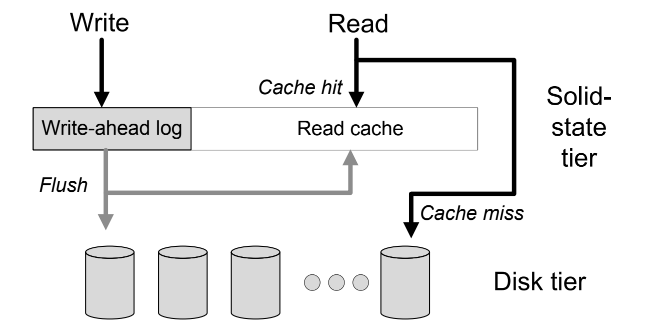


Figure 2. Two-tier architecture using SSDs and disks. Writes go to a log-structured partition on flash and eventually are flushed to disk. Non-streaming, frequently read "hot" blocks are serviced from flash, whereas other reads are serviced from the disk layer. Updates to "hot" blocks are first written to the log, then reflected on the read cache.

the number of devices added. This assumption can be validated in practice using the same synthetic tests above. The level of fault-tolerance is a separately configurable parameter, and the model automatically adds in the one additional device which does not contribute additional performance or capacity but does add to the fault-tolerance.

Reliability and fault-tolerance In general, storage device reliability is hard to predict. Recent work in this area has shown that traditional reliability metrics need to be extracted through empirical observation after a number of years in operation (Jiang et al. 2008; Schroeder and Gibson 2007). Such empirical numbers are still lacking for SSDs.

It is known that NAND flash memories suffer from *wear*: the reliability of the memory decreases as it is repeatedly erased and overwritten. Typical enterprise SSDs specify a wear tolerance of 10,000–100,000 write cycles. We convert this vendor metric into a *wear rate*: the average number of GB/day that can be written to the device without exceeding the wear tolerance limit over the device lifetime (we use a default lifetime of 5 years).

Since this paper is focused on solid-state storage, and wear is a novel, SSD-specific phenomenon, we include it in our device models. Currently we do not model other failures, such as mechanical failures in disks. Given the widespread use of RAID for fault-tolerance, we do not believe that reliability differences are an important factor for provisioning, however verifying this remains future work.

3.3 Tiered models

The cost and performance characteristics of solid-state memory are in between those of main memory (DRAM) and traditional storage (disks). Hence, it also makes sense to consider solid-state devices not just as a replacement for disks, but also as an intermediate storage tier between main memory and disks. This tier could cache more data than DRAM, since it is cheaper per gigabyte than DRAM, and hence improve read performance. Unlike DRAM, solid-state memories also offer persistence. Hence, this tier could also be used to improve write performance, by using it as a write-ahead log which is lazily flushed to the lower tier. Several storage and file system architectures (Miller et al. 2001; Panabaker 2006) have been proposed to use solid-state memory as a cache and/or a write-ahead log. This paper quantifies the benefits of these approaches for enterprise workloads.

Our tool supports hybrid configurations where solid-state devices are used as a transparent block-level intermediate tier. Caching and write-ahead logging could also be done at a higher level, e.g., at the file system, which would allow policies based on semantic knowledge, such as putting meta-data in the cache. Such policies however would require changes to the file system, and our block-level workload traces do not include such semantic information. We compute the benefits of transparent block-level caching and logging without assuming any changes in the file system or application layer.

Figure 2 shows the flexible architecture we assume. The solid-state tier is divided into a write-ahead log and a larger read cache area. In practice, the write-ahead log is smaller than the read cache; also, the location of the write log would be periodically moved to level the wear across the NAND storage device. The write-ahead log could also be replicated over multiple SSDs depending on workload fault-tolerance requirements. However, the read cache has only clean blocks and does not need such fault tolerance. Our models assume the caching and write-ahead logging policies described below, which are based on our experiences with the traced workloads as well as key first-order properties of SSDs and disks. They can easily be extended to cover other policies.

3.3.1 Read caching

Our workloads are server I/O workloads and hence have already been filtered through large main-memory buffer caches. Hence, there is little short-term temporal locality in the access patterns. However, there could be benefit in caching blocks based on long-term access frequency. Also, the benefit of caching is highest for random-access reads: disk subsystems are very effective at servicing streaming runs. Based on these observations, we use the following caching policy. We rank the logical blocks in each trace according to the number of random accesses to each; accesses are classified as random or sequential as described previously in Section 3.1. The total number of reads (random and sequential) is used as a secondary ranking metric. This policy acts like an "oracle" by computing the most frequently accessed blocks before they are accessed: hence it gives a reasonable upper bound on achievable hit rates.

For a given cache size of H blocks, the tool splits the workload trace into two, such that accesses to the hottest H blocks go to the top tier, and the remainder to the bottom tier. It then computes the best configuration independently for each tier. In theory, each tier could be provisioned with any device type; in practice, the only solutions generated are those with a solid-state tier on top of a disk tier. By iterating over values of H, the solver finds the single- or two-tiered configuration that satisfies workload requirements and has the lowest total cost.

3.3.2 Write-ahead log

It is straightforward to absorb writes to a storage volume on a much smaller solid-state device; the written data can then be flushed in the background to the underlying volume. For low response times and high throughput, the solid-state device should be laid out using a log structure, for example by using a versioned circular log which we implemented transparently at the block level (Narayanan et al. 2008b). Writes can be acknowledged as soon as they are persistent in the log. The log space can be reused when the writes become persistent on the underlying volume.

Write-ahead logging can be combined with read caching as shown in Figure 2. In this case all writes, and all reads of hot blocks, are redirected to the solid-state tier. To guarantee sequential performance for the writes however, writes are sent to a separately allocated log area on the solid-state device. These are then lazily flushed to the disk tier, and if appropriate to the read cache. In theory, this could cause double-buffering of blocks in the log and in the cache; however, as we will see in Section 4, the size of the write-ahead log required for our workloads is very small and hence this is not a concern.

Background flushes can take advantage of batching, coalescing, overwriting of blocks, and low-priority I/O to reduce the I/O load on the lower tier. The efficacy of these optimizations depends on the workload, the flushing policy, and the log size. Hence, we evaluate both extremes of this spectrum: a "write-through log" where writes are sent simultaneously to the disk tier and to the log, and a "write-back" log where the lazy flushes to the disk tier are assumed to be free.

To find the cost of adding a write-ahead log (with or without read caching) the tool must estimate both the capacity as well as the performance required from the caching tier. The log capacity is measured as the size of the largest write burst observed in the workload trace: the maximum amount of write data that was ever in flight one time. The performance requirements are derived by splitting the workload traces according to the caching/logging policy: writes are sent to both tiers whereas reads are split between the tiers according to the blocks that they access.

3.4 Solver

Given workload requirements, and per-device capabilities as well as costs, the solver finds the least-cost configuration that will satisfy the requirements. Any cost metric can be used: in this paper we use purchase cost in dollars and power consumption in watts. These could be combined into a single dollar value based on the anticipated device lifetime and energy costs; in our analyses we show the two costs separately. The number of devices required $N(d, w)$ of any particular device type d to satisfy the requirements of workload w is:

$$N(d, w) = \max_{m} \left\lceil \frac{r_m(w)}{s_m(d)} \right\rceil + F(w) \qquad (1)$$

where m ranges over the different metrics of the workload requirement: capacity, random-access read rate, etc. $r_m(w)$ is the workload's requirement for the metric m (measured in GB, MB/s, IOPS, etc.) and $s_m(d)$ is the device's score on that metric measured in the same units as $r_m(w)$. In other words, the number of devices is determined by the most costly metric to satisfy. The workload's fault-tolerance requirement $F(w)$ is specified separately as the number of redundant devices required.

The best device $d_{opt}(w)$ for a given workload w is then

$$d_{opt}(w) = \operatorname*{argmin}_{d} N(d, w) \cdot C(d) \qquad (2)$$

where $C(d)$ is the cost metric being minimized, e.g., dollar cost. The cost of this configuration is then

$$C_{opt}(w) = N(d_{opt}(w), w) \cdot C(d_{opt}(w)) \qquad (3)$$

To allow for tiered solutions, the workload trace is split into two separate workloads $w_{top}(w, H)$ and $w_{bot}(w, H)$, where H is the size H of the top tier. The performance metrics for the two workloads are extracted from the two derived traces. The lower tier has the same capacity and fault-tolerance requirements as the original workload. For the top tier the capacity requirement is simply H; the fault-tolerance is set to 0 by default for the read cache. The cost of such a tiered configuration is then given by:

$$C_{tier}(w, H) = C_{opt}(w_{top}(w, H)) + C_{opt}(w_{bot}(w, H)) \qquad (4)$$

and the minimum-cost tiered configuration has a cost

$$C_{opt/tier}(w) = \min_{H} C_{tier}(w, H) \qquad (5)$$

In theory, H is a continuously varying parameter. However, in practice, solid-state memories are sold in discrete sizes, and very large solid-state memories are too expensive to be part of a least-cost tiered solution. Additionally, each value of H requires us to reprocess the input traces: the most expensive step in the tool chain. Hence, our tool searches a small number of discrete values for H: powers of 2 ranging from 4–128 GB.

The solver is currently implemented as 1580 unoptimized lines of C and Python. The run time is dominated by the time to split each workload trace for each value of H, and to extract the workload requirements: this is linear in the size of the trace file, which uses a text format. The traces used in this paper vary in size from 40 KB–10 GB; the median trace (255 MB) took 712 s to process on a 2 GHz AMD Opteron.

4. Results

This section has four parts. First, it presents the device characteristics extracted by the tool from a number of representative enterprise-class storage devices, and compares them in

Device	Price (US$)	Capacity (GB)	Power (W)	Sequential xfer (MB/s)		Random access (IOPS)		Wear rate (GB/day)
				Read	Write	Read	Write	
Memoright MR 25.2	739	32	1.0	121	126	6450	351	500
	509	16	1.0					
	389	8	1.0					
Seagate Cheetah 10K	339	300	10.1	85	84	277	256	n/a
	123	146	7.8					
Seagate Cheetah 15K	172	146	12.5	88	85	384	269	n/a
	349	300	12.5					
Seagate Momentus 7200*	150	200	0.8	64	54	102	118	n/a
	53	160	0.8					

Table 4. Storage device characteristics. *All devices except the Momentus are enterprise-class.

terms of their capabilities per dollar of purchase cost. Second, it analyzes the hypothesis of whether disks can be fully replaced by SSDs for each workload at a lower dollar cost, while satisfying workload requirements. The third part is a similar analysis for SSDs as an intermediate storage tier used as a cache or write-ahead log. The final part is an analysis and discussion of power considerations, including a comparison of the enterprise-class SSDs and disks with a low-power SATA disk.

4.1 Analysis of device characteristics

Table 4 shows the characteristics of four representative devices used in this paper: the Memoright MR 25.2 is a recently released enterprise class SSD; the Seagate Cheetah is a commonly used high-end enterprise class disk available in two speeds, 10,000 rpm and 15,000 rpm. The Seagate Momentus is a low-power, low-speed drive that is not currently used for enterprise storage. We defer discussion of its potential merits in enterprise until Section 4.4.

For each device we considered multiple versions with different capacities. The dollar costs are based on US on-line retail prices as of June 2008 and the power consumptions are based on vendor documentation. The performance characteristics were extracted using the micro-benchmarks described in Section 3 from one version of each device type (the first one listed in each group).

Thus each device is characterized by multiple dimensions: capacity, sequential read bandwidth, etc. Our tool considers all these dimensions as well as all the variants of each device (different capacities, log-structured SSDs, etc.). However, to understand the tradeoffs between the devices it helps to visualize the most important features. Figure 3 shows the three most important axes for this paper, normalized by dollar cost: capacity (measured in GB/$), sequential read performance (MB/s/$), and random-access read performance (IOPS/$). For each device type, we chose the version having the highest value along each axis. The figure does not show write performance: in general read and write performance are equivalent for the mechanical disks, and sequential read and write performance are equivalent for the SSD. Furthermore, the poor random-write performance of SSDs can be avoided by converting random writes to sequential writes using log-structured techniques (Birrell et al. 2007; Rosenblum and Ousterhout 1991; Woodhouse 2001).

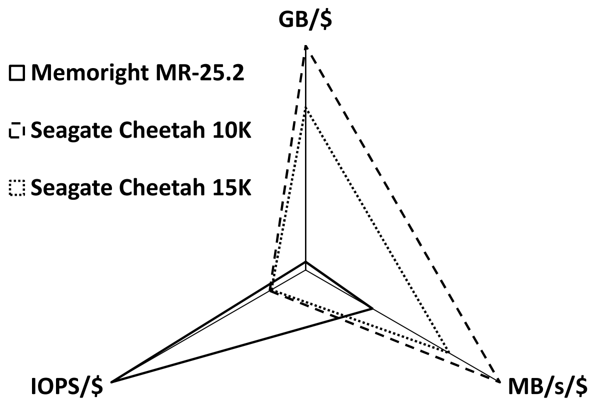


Figure 3. Device capabilities

Several observations can be made at this stage, even without considering workload requirements. SSDs provide higher random-access performance per dollar (IOPS/$) whereas the enterprise disks win on capacity and sequential performance per dollar. It seems likely that the main factor in choosing SSDs versus disks is the trade-off between the capacity requirement and the "IOPS requirement" of the workload. Sequential bandwidth also favors disks, but the difference is smaller than the order-of-magnitude differences on the other two axes.

Interestingly, the faster-rotating 15K disk has no performance advantage over the 10K disk when performance is normalized by dollar cost. This suggests that there could be room for even slower-rotating disks in the enterprise market (this will lead us to consider a slow disk in Section 4.4).

4.2 Replacing disks with SSDs

A natural question this section seeks to answer is "what does it take to replace disks with SSDs?" Whole-disk replacement

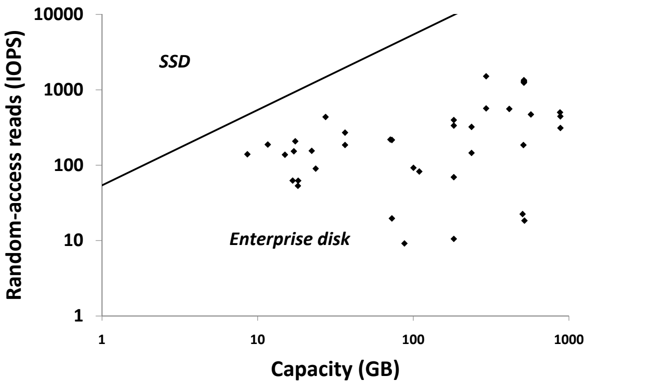


Figure 4. IOPS/capacity trade-off (log-log scale)

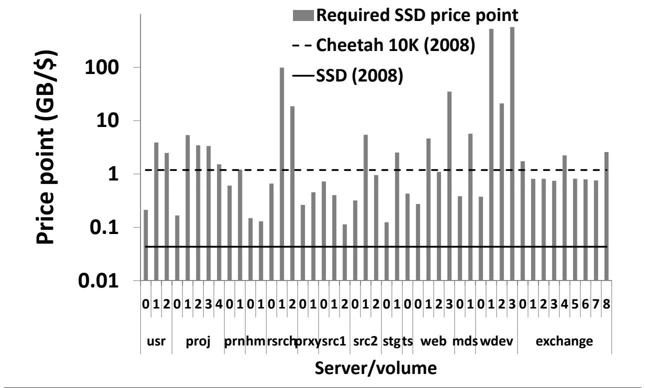


Figure 5. Price point at which SSDs can replace disks (log scale). 0 is the system volume, the others contain data.

has the appeal of requiring no architectural changes to the storage subsystem. This section answers the question by using the provisioning tool to find the least-cost device type for each workload.

Figure 4 shows a plot of the two key metrics: read IOPS on the y-axis and capacity on the x-axis. Each workload is represented by a point: we see that none of them fall in the top-left region, which is the space where replacing disks with SSDs would be the best choice. The separating line represents points where the cost for capacity equals the IOPS cost. Using the device parameters in Table 4, the Cheetah 10K was the best choice for all 45 workloads. In all cases the provisioning cost was determined by either the capacity or the read IOPS requirement. In other words, the sequential transfer and random-access write requirements are never high enough to dominate the provisioning cost.

At today's prices SSDs cannot replace enterprise disks for any of our workloads: the high per-gigabyte price of SSDs today makes them too expensive even for the smallest, most IOPS-intensive workloads. At what capacity/cost will SSDs become competitive with enterprise disks? Figure 5 shows this price point (expressed in GB/$) for each volume, on a log scale. For reference, it also shows the current price points for the Memoright SSD and the Cheetah 10K disk.

The break-even point varies from 3–3000 times the capacity/cost of today's SSDs. Some smaller volumes, especially system volumes (numbered 0 in the figure) require only a 2–4x reduction in SSD price to consider replacement of disks by SSDs. However, most volumes will require a reduction of 1–3 orders of magnitude. For 21 of 45 volumes, the break-even point lies beyond the current capacity per dollar of the Cheetah 10K: this is because capacity dominates the provisioning cost of these volumes even when using disks. Assuming that the per-gigabyte cost of both disks and SSDs will decrease in the future with disks continuing to be cheaper per GB than SSDs, at some point the IOPS requirement will become the cost-determining factor.

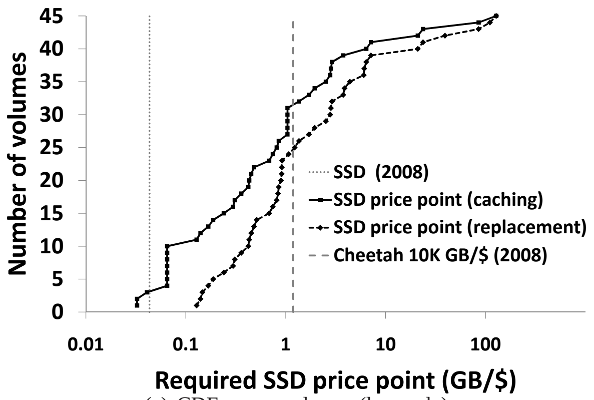
4.3 Two-tiered configurations

This section answers the question "what are the benefits/costs of augmenting the existing storage hierarchy with SSDs?" Specifically, we consider the two-tiered configuration described in Section 3.3, where the SSD tier functions as a write-ahead log as well as a read cache for frequently read, randomly-accessed blocks. Our analysis shows that the amount of storage required for the write-ahead log is small compared to SSD sizes available today. Thus it is reasonable to allocate a small part of the solid-state memory for use as a write-ahead log and use the rest as a read cache. We first present the analysis for the write-ahead log, and then for a combined write-ahead log/read cache tier.

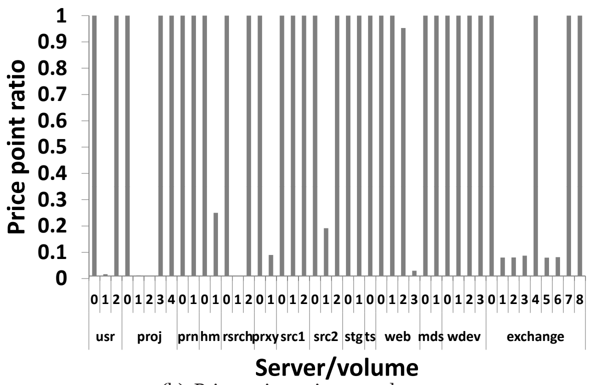
4.3.1 Write-ahead log

Across all volumes, the maximum write burst size, and hence the maximum space required for the write-ahead log, was less than 96 MB. The peak write bandwidth required was less than 55 MB/s: less than half the sequential write bandwidth of the Memoright SSD. Thus, a single SSD can easily host the write-ahead log for a volume using only a fraction of its capacity. Even with fault tolerance added, the capacity and bandwidth requirement is low. We repeated the analysis, but this time sharing a single write-ahead log across all volumes on the same server. The peak capacity requirement increased to 230 MB and the peak bandwidth requirement remained under 55 MB/s. The peak bandwidth requirement did not increase significantly because across volumes peaks are not correlated.

This analysis is based on a "write-through" log, which does not reduce the write traffic to the disk tier. We also re-analyzed all the workloads assuming a "write-back" log that absorbs all write traffic, i.e., assuming that the background flushes are entirely free. We found that this reduction in write traffic did not reduce the provisioning requirement for the disk tier for any of the workloads; this was expected since the limiting factor for the workloads was always capacity or read performance. Thus, while a larger log with a lazy write-



(a) CDF across volumes (log scale)



(b) Price point ratio per volume

Figure 6. SSD price points for caching versus replacement

back flush policy can reduce load on the disk tier, the load reduction is not enough to reduce the provisioning cost.

Given the small amount of log space required, it seems clear that if there is any solid-state memory in the system, a small partition should be set aside as a write-ahead log for the disk tier. This can improve write response times but will not reduce the cost of provisioning the disk tier.

4.3.2 Write-ahead log with read cache

Capacity is clearly an obstacle to replacing disks entirely with SSDs. However, if a small solid-state cache can absorb a large fraction of a volume's load, especially the random-read load, then we could potentially provision fewer spindles for the volume. This would be cheaper than using (more) DRAM for caching, since flash is significantly cheaper per gigabyte than DRAM. It is important to note we are not advocating replacing main memory buffer caches with solid-state memory. Our workload traces are taken below the main memory buffer cache: hence we do not know how effective these caches are, or the effect of removing them.

The blocks to store in the read cache are selected according to random-access read frequency as described in Section 3. Reads of these blocks are sent to the top (SSD) tier, and other reads to the disk tier. All writes are sent simultaneously to both tiers, with the top tier serving as the low-latency write-ahead log in "write-through" mode.

We used the solver to test several tiered configurations for each workload, with the capacity of the solid-state tier set to 4, 8, 16, 32, 64 and 128 GB. In all cases the performance of a single SSD was sufficient to handle the I/O load placed on the solid-state tier. For 16 out of 45 volumes, the caching tier also reduced the number of spindles required for the disk tier. However, for the remaining 29 volumes, caching does not reduce the cost of the disk tier: these volumes are already provisioned for capacity and/or have little cacheability at the block level (perhaps due to main-memory buffer caches.)

At the current SSD capacity/cost, only 3 volumes benefited overall. For all the others, the cost of the cache tier outweighed any savings in the disk tier. For each volume, we evaluated the lowest SSD capacity/cost at which a two-tier solution could compete with a single, disk-based tier: the cache size to achieve this varied across volumes. Figure 6(a) shows this as a cumulative distribution across volumes; for comparison we also show the CDF of the price points required for complete replacement of disks. Figure 6(b) shows the ratio of the SSD price point required for a tiered configuration to that required for full replacement of disks by SSDs. If the ratio is close to 1, then caching is of little benefit, since the price point at which caching becomes cost-effective will also allow full replacement. However, for at least 13 of the workloads, the price point ratio is much lower than 1 and caching will become cost-effective before replacement does.

4.4 Power

The previous sections used purchase cost as the metric to minimize while satisfying workload requirements. Here we look at the implications of minimizing power consumption instead. We made the conscious choice to analyze, in addition to the enterprise disks and SSDs, a non-enterprise SATA disk in this section: the Seagate Momentus 7200. We are aware that SATA and SCSI disks are different, especially in terms of their reliability (Anderson et al. 2003). However, we chose not to ignore a SATA-based solution for two reasons. First, there is much work on storage clusters of commodity cheap hardware, where reliability is handled at a higher level. Second, recent empirical research on the reliability of disk drives has shown inconclusive results (Jiang et al. 2008; Schroeder and Gibson 2007). However, we caution that the analyses here are based primarily on performance, capacity and power metrics. More empirical evidence is required to make any strong claim about reliability.

Figure 7 shows the four devices compared by capacity, sequential bandwidth, and IOPS, all normalized by the device's power consumption (for the performance axes we use the read performance). We use idle power numbers (device ready/disks spinning, but no I/O), since our workloads have

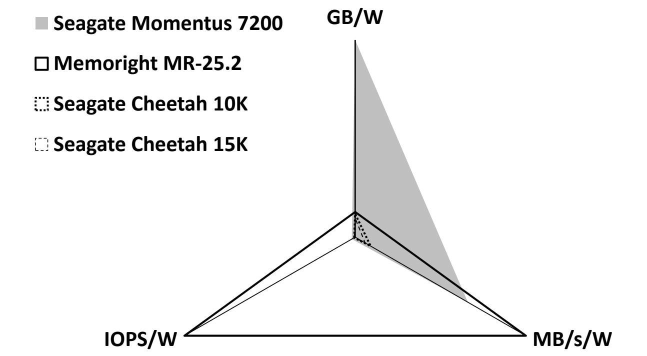


Figure 7. Device capabilities per watt

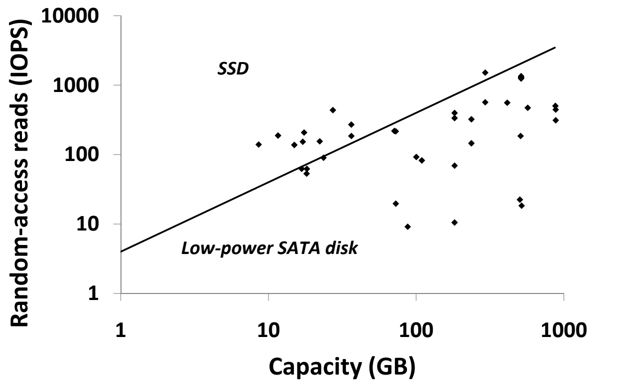


Figure 8. IOPS/capacity trade-off when optimizing for power (log-log scale)

considerable idleness over their duration. Using active power does not make much difference in our calculations. The main takeaway is that, when scaled per watt, SSDs have much better performance and comparable capacity to the enterprise disks. However, the low-power, low-speed Momentus does far better than the SSD in terms of gigabytes per watt. We also found that the Momentus significantly outperformed the Cheetahs on capacity and performance per dollar; however, the price advantage might just reflect market forces, whereas power is more a property of the underlying technology.

For our workloads, the tool reports the SSD as the lowest-power (single-tier) solution for 11 out of 45 workloads, and chooses the Momentus for the remaining 34. Again, read IOPS and capacity were the limiting factors for all workloads. Figure 8 shows the workloads as points on these two axes: the graph is divided according to the device providing the lowest-power solution.

The analysis so far has been independent of energy prices. In general however, the cost of power consumption must be balanced against that of provisioning the hardware, by computing the overall cost over the device lifetime or upgrade period (typically 3–5 years). Figure 9 shows the "5-year break-even energy price" (in $/kWh), i.e., the energy price at which the power savings over 5 years of an SSD-based solution will equal its additional purchase cost. We show the break-even price for the SSD against the Cheetah for all 45 volumes, and for the SSD against the Momentus for the 11 volumes for which the SSD-based solution was more power-efficient. For reference we show the commercial US energy price as of March 2008 (US Department of Energy 2008).

The main takeaway is that the break-even points are 1–3 orders of magnitude above the current energy prices: even if we allow a 100% overhead in energy costs for cooling and power supply equipment, we need energy prices to increase by factor of 5 for the power savings of SSDs to justify the initial cost for even the smallest volumes. Thus, perhaps surprisingly, power consumption alone is not a compelling argument for SSDs; however, falling SSD per-gigabyte prices, and an increase in energy prices motivate replacement of smaller volumes with SSDs sooner rather than later.

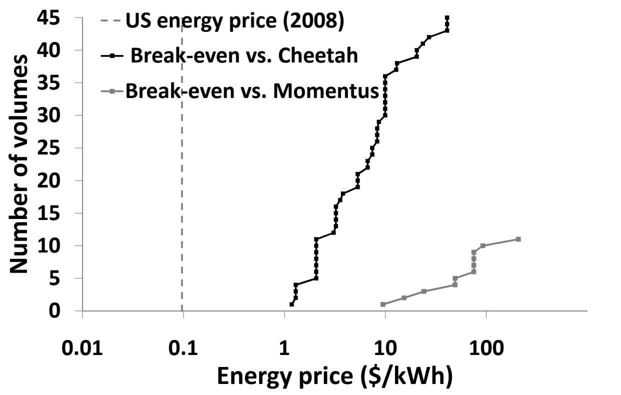


Figure 9. 5-year break-even energy price point for SSDs (log scale)

4.5 Reliability and wear

In this paper so far we have considered performance, capacity, dollar cost and power consumption as the metrics of interest for provisioning storage. While reliability is also an important metric, we believe the pervasive use of RAID for fault-tolerance we believe makes it a less important factor. Moreover, the factors that determine disk reliability are still not conclusively known, and are largely estimated from empirical studies (Jiang et al. 2008; Schroeder and Gibson 2007). Such empirical evidence is lacking for SSDs.

However, flash-based SSD vendors do provide a wear-out metric for their devices, as the number of times any portion of the flash memory can be erased and rewritten before the reliability begins to degrade. Based on this and the long-term write rate seen by each of our workload volumes, we computed the time at which this limit would be reached, for each volume. Figure 10 shows the CDF of this wear-out time in years on a log scale. Note that this assumes that in the long term, the wear is evenly distributed over the flash memory: this can be done through various wear-leveling techniques (Birrell et al. 2007; Gal and Toledo 2005). These techniques generally avoid in-place updates

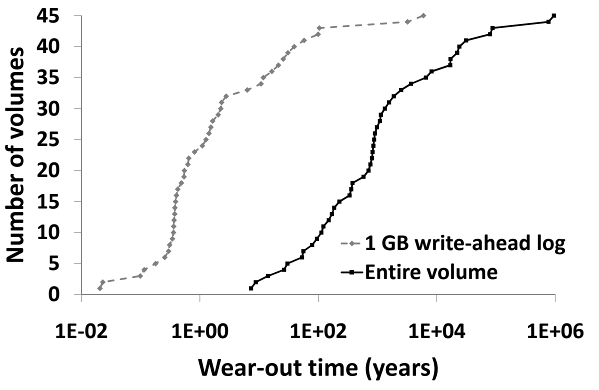


Figure 10. Wear-out times (log scale)

and hence require some additional background writes for defragmentation. However, even if we conservatively assume a high overhead of 50% (one background write for every two foreground writes), the wear-out time is well above 5 years for all volumes, and above 100 years for the majority of volumes. Hence, we do not expect that wear will be a major contributor to the total cost of SSD-based storage.

Wear can be a concern, however, for flash used as a write-ahead log. Here a small flash device absorbs a relatively large number of writes. The dotted line in Figure 10 shows the CDF of estimated wear-out time for a 1 GB flash used as a write-ahead log for each of our workloads. 32 out of 45 workloads have a wear-out time under 5 years. Thus, while flash-based SSDs can easily provide the performance and capacity required for a write-ahead log, wear is a significant concern here and will be need to be addressed. If a large flash memory is being used as a combined read cache and write log, one potential solution is to periodically rotate the location of the (small) write log on the flash.

5. Conclusion

The main contribution of this paper is an analysis of the complex SSD/disk tradeoffs (capacity, performance, power, reliability and dollar cost) guided by an optimization framework and real enterprise workload traces. From a system's perspective our framework helps in making informed decisions on how to best configure a storage system.

This work can be extended in several ways. Better reliability models for low-power disks and SSDs would enhance the analyses in this paper. The device models and solver also have room for improvement. Currently we use simple first-order performance models and an exhaustive-search solver, and we treat performance requirements as hard constraints. Each of these aspects could be extended, for example by using more complex device models (Bucy et al. 2008), heuristic solvers (Anderson et al. 2005), and utility functions to describe tradeoffs (Strunk et al. 2008).

References

Nitin Agrawal, Vijayan Prabhakaran, Ted Wobber, John D. Davis, Mark Manasse, and Rina Panigrahy. Design tradeoffs for SSD performance. In *USENIX Annual Technical Conference*, Boston, MA, June 2008.

Dave Anderson, Jim Dykes, and Erik Riedel. More than an interface - SCSI vs. ATA. In *Proc. USENIX Conference on File and Storage Technologies (FAST)*, San Francisco, CA, March 2003.

Eric Anderson, Susan Spence, Ram Swaminathan, Mahesh Kallahalla, and Qian Wang. Quickly finding near-optimal storage designs. *ACM Trans. Comput. Syst.*, 23(4):337–374, 2005. ISSN 0734-2071. doi: http://doi.acm.org/10.1145/1113574.1113575.

M. Baker, S. Asami, E. Deprit, J. Ousterhout, and M. Seltzer. Non-volatile memory for fast, reliable file systems. In *Proc. International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Boston, MA, October 1992.

Andrew Birrell, Michael Isard, Chuck Thacker, and Ted Wobber. A design for high-performance flash disks. *Operating Systems Review*, 41(2):88–93, 2007.

John S. Bucy, Jiri Schindler, Steven W. Schlosser, and Gregory R. Ganger. The DiskSim simulation environment version 4.0 reference manual. Technical Report CMU-PDL-08-101, Carnegie Mellon University, May 2008.

Gal and Toledo. Algorithms and data structures for flash memories. *CSURV: Computing Surveys*, 37, 2005.

Richard Golding, Peter Bosch, Carl Staelin, Tim Sullivan, and John Wilkes. Idleness is not sloth. In *Proc. USENIX Annual Technical Conference*, New Orleans, LA, 1995.

Intel News Release. Intel, STMicroelectronics deliver industry's first phase change memory prototypes. `http://www.intel.com/pressroom/archive/releases/20080206corp.htm`, February 2008.

Weihang Jiang, Chongfeng Hu, Yuanyuan Zhou, and Arkady Kanevsky. Are disks the dominant contributor for storage failures? A comprehensive study of storage subsystem failure characteristics. In *USENIX Conference on File and Storage Technologies (FAST)*, San Jose, CA, February 2008.

Taeho Kgil and Trevor N. Mudge. Flashcache: a NAND flash memory file cache for low power web servers. In *Proc. International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, Seoul, Korea, October 2006.

Ioannis Koltsidas and Stratis Viglas. Flashing up the storage layer. In *Proc. International Conference on Very Large Data Bases (VLDB)*, Auckland, New Zealand, August 2008.

Sang-Won Lee, Bongki Moon, Chanik Park, Jae-Myung Kim, and Sang-Woo Kim. A case for flash memory ssd in enterprise database applications. In *Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD*, 2008.

E. Miller, S. Brandt, and D. Long. HeRMES: High-performance reliable MRAM-enabled storage. In *IEEE Workshop on Hot Topics in Operating Systems (HotOS)*, May 2001.

D. Narayanan, A. Donnelly, and A. Rowstron. Write off-loading: Practical power management for enterprise storage. In *Proc.*

USENIX Conference on File and Storage Technologies (FAST), February 2008a.

Dushyanth Narayanan, Austin Donnelly, Eno Thereska, Sameh Elnikety, and Antony Rowstron. Everest: Scaling down peak loads through I/O off-loading. In *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI)*, San Diego, CA, December 2008b.

Suman Nath and Aman Kansal. Flashdb: Dynamic self tuning database for NAND flash. In *Information Processing in Sensor Networks (IPSN)*, 2007.

Ruston Panabaker. Hybrid hard disk and ReadyDrive technology: Improving performance and power for Windows Vista mobile PCs. `http://www.microsoft.com/whdc/winhec/pres06.mspx`, May 2006.

Vijayan Prabhakaran, Thomas L. Rodeheffer, and Lidong Zhou. Transactional flash. In *To appear in Proc. Symposium on Operating Systems Design and Implementation (OSDI)*, San Diego, CA, December 2008.

Mendel Rosenblum and John Ousterhout. The design and implementation of a log-structured file system. In *Proc. ACM Symposium on Operating Systems Principles (SOSP)*, Pacific Grove, CA, October 1991.

Samsung. MH80 SATA product data sheet, June 2006.

B. Schroeder and G. A. Gibson. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? In *Proc. USENIX Conference on File and Storage Technologies (FAST)*, San Jose, CA, February 2007.

John Strunk, Eno Thereska, Christos Faloutsos, and Gregory Ganger. Using utility to provision storage systems. In *Proc. USENIX Conference on File and Storage Technologies (FAST)*, San Jose, CA, February 2008.

US Department of Energy. Average retail price of electricity to ultimate customers by end-use sector, by state, April 2008 and 2007. `http://www.eia.doe.gov/cneaf/electricity/epm/table5_6_a.html`, August 2008.

David Woodhouse. JFFS: The journalling flash file system. In *Ottawa Linux Symposium*, 2001.

Bruce Worthington and Swaroop Kavalanekar. Characterization of storage workload traces from production Windows servers. In *IISWC*, 2008.

Chin-Hsien Wu, Li-Pin Chang, and Tei-Wei Kuo. An efficient b-tree layer for flash-memory storage systems. In *Real-Time and Embedded Computing Systems and Applications (RTCSA)*, Gothenburg, Sweden, August 2004.

Michael Wu and Willy Zwaenepoel. eNVy: A non-volatile main memory storage system. In *Proc. Internatinoal Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, San Jose, CA, October 1994.

D. Zeinalipour-Yazti, S. Lin, V. Kalogeraki, D. Gunopulos, and W. Najjar. Microhash: An efficient index structure for flash-based sensor devices. In *Proc. USENIX Conference on File and Storage Technologies (FAST)*, San Francisco, CA, 2005.