Migrating Server 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 enterprise server storage has not been studied. Provisioning server storage is challenging. It requires optimizing for the performance, capacity, power and reliability needs of the expected workload, all while minimizing financial costs.

In this paper we analyze a number of workload traces from servers in both large and small data centers, to decide whether and how SSDs should be used to support each. We analyze both complete replacement of disks by SSDs, as well as use of SSDs as an intermediate tier between disks and DRAM. We describe an automated tool that, given device models and a block-level trace of a workload, determines the least-cost storage configuration that will support the workload's performance, capacity, and fault-tolerance requirements.

We found that replacing disks by SSDs is not a cost-effective option for any of our workloads, due to the low capacity per dollar of SSDs. Depending on the workload, the capacity per dollar of SSDs needs to increase by a factor of 3–3000 for an SSD-based solution to break even with a disk-based solution. Thus, without a large increase in SSD capacity per dollar, only the smallest volumes, such as system boot volumes, can be cost-effectively migrated to SSDs. The benefit of using SSDs as an intermediate caching tier is also limited: fewer than 10% of our workloads can reduce provisioning costs by using an SSD tier at today's capacity per dollar, and fewer than 20% can do so at any SSD capacity per dollar. Although SSDs are much more energy-efficient than enterprise disks, the energy savings are outweighed by the hardware costs, and comparable energy savings are achievable with low-power SATA disks.

EuroSys'09, April 1–3, 2009, Nuremberg, Germany.
Copyright © 2009 ACM 978-1-60558-482-9/09/04…$5.00

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 server storage, which is complex, and different both in scale and requirements from laptop and desktop storage? A server storage system must meet capacity, performance, and reliability requirements — which vary widely across workloads — while minimizing cost.

While solid-state storage offers fast random-access reads and low power consumption, it also has disadvantages such as high cost per gigabyte. There have been many proposals to exploit the characteristics of solid-state storage to redesign storage systems, either replacing disks entirely [Prabhakaran 2008, Woodhouse 2001] or using solid-state storage to augment disk storage, e.g., to store file system metadata [Miller 2001]. However, there has been no cost-benefit analysis of these architectures for real workloads.

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 these requirements? The approach in this paper is to find the least-cost configuration that meets known targets for performance (as well as capacity and fault-tolerance) rather than maximizing performance while keeping within a known cost budget. Since this paper is focused on the tradeoffs between SSDs and mechanical disks, we consider three types of configuration for each workload: disk-only, SSD-only, and two-tiered (with the SSD tier above the disk tier).

Intuitively, SSDs are favored by workloads with a high demand for random-access I/Os, especially reads, and a relatively low capacity requirement. To decide whether a given workload is better served by SSDs than disks, we first define and measure the workload's requirements along several axes: capacity, random-access I/O rate, sequential transfer rate, and fault-tolerance. We also characterize the capabilities of each device type using the same metrics, normalized by dollar cost. Given these two sets of parameters, we compute the total cost of different configurations, and choose the one that meets workload requirements at the lowest cost.

1.1 Contributions

The main contribution of this paper is an analysis of the SSD/disk tradeoff for a range of workloads. Our workload traces are taken from a variety of servers in both large and small data centers. Our analysis is based on matching each workload with the storage configuration that meets its requirements at the lowest dollar cost. We note that storage devices such as disks and SSDs are only part of the cost of supporting enterprise storage; other costs such as networking, enclosures, or cooling are not addressed in this paper.

We find that replacing disks by SSDs is not the optimal solution for any of our workloads. This is due to the very low capacity/dollar offered by SSDs, compared to disks. Depending on the workload, the capacity/dollar will need to improve by a factor of 3–3000 for SSDs to become competitive with disks. While the capacity/dollar of SSDs is improving over time, an improvement of several orders of magnitude seems unlikely [Hetzler 2008]. Thus replacement of disks by SSDs does not seem likely for any but the smallest volumes in our study, such as system boot volumes.

Using SSDs as a caching tier above the disk-based storage also has limited benefits. By absorbing some of the I/O in the SSD tier, such hybrid configurations could reduce the load on the disk tier. This load reduction might then allow reduced provisioning of the disk tier and hence save money overall. However, for 80% of the workloads in our study, the SSD tier did not reduce the cost of the disk tier at all. For a further 10% of the workloads, the SSD tier enabled some reduction in the cost of the disk tier, but the cost of the SSD tier outweighed any savings in the disk tier. Only for 10% of the workloads was the hybrid configuration the optimal configuration. Similarly we found that using SSDs as a write-ahead log, while requiring only modest capacity and bandwidth in the SSD tier and potentially reducing write response times, does not reduce provisioning costs. Additionally, current flash-based SSDs 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 SSD.

We also found that although SSD-based solutions consume less power than enterprise disk based solutions, the resulting dollar 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. These disks also offer much higher capacity and performance per dollar than either enterprise disks or SSDs. However, they are generally believed to be less reliable and the tradeoff between cost and reliability requires further study.

An additional contribution of this paper 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 2005, Strunk 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.

2. Background and related work

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

2.1 Solid-state drives

Solid-state drives (SSDs) provide 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 widely used in storage devices today. In this paper we use device models extracted from flash SSDs; our approach would apply equally to these other technologies.

Flash storage has two unusual limitations, but both 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 relatively large units (64–128 KB) before it can be rewritten. This results in poor random-access write performance for the current generation of SSDs. However, random-access writes can be made sequential by using a log-structured file system [Rosenblum 1991, Woodhouse 2001] or transparent block-level remapping [Agrawal 2008, Birrell 2007]. Since SSDs do not have any positioning delays, read performance is not affected by these layout changes. Some overhead is added for log cleaning: server workloads in general have sufficient idle time, due to diurnal load patterns, to perform these maintenance tasks without impacting foreground workloads [Golding 1995, Narayanan 2008a].

The second limitation of 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 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 amount of additional wear in the form of additional writes for compaction and defragmentation.

2.2 Workload traces

Our analysis uses real workload traces collected from production servers in both large and small data centers. There are two sets of large data center traces: the first is from an Exchange server that serves 5000 corporate users, one of many

Data center size	Server	Function	Volumes	Spindles	Total GB
Large	exchange	Corporate mail	9	102	6706
	msn-befs	MSN storage server	4	28	429
Small	usr	User home dirs	3	16	1367
	proj	Project dirs	5	44	2094
	print	Print server	2	6	452
	hm	H/w monitor	2	6	39
	resproj	Research projects	3	24	277
	proxy	Web proxy	2	4	89
	src1	Source control	3	12	555
	src2	Source control	3	14	355
	webstg	Web staging	2	6	113
	term	Terminal server	1	2	22
	websql	Web/SQL server	4	17	441
	media	Media server	2	16	509
	webdev	Test web server	4	12	136

Table 1. Enterprise servers traced.

responsible for Microsoft employee e-mail. The Exchange traces cover 9 volumes over a 24-hour period, starting from 2:39pm PST on the 12th December 2007. The system boot volume has 2 disks in a RAID-1 configuration, and the remaining volumes each have 6–8 disks in a RAID-10 configuration. The Exchange traces contain 61 million requests, of which 43% are reads.

The second set of large data center traces is from 4 RAID-10 volumes on an MSN storage back-end file store. These traces cover a 6-hour period starting from 12:41pm MST on the 10th March 2008. They contain 29 million requests, of which 67% are reads. Both the Exchange and MSN storage traces are described in more detail by Worthington [2008].

The third set of traces was taken from a small data center with about 100 on-site users [Narayanan 2008a], and covers a range of servers typical of small to medium enterprises: file servers, database servers, caching web proxies, etc. Each server was configured with a 2-disk RAID-1 system boot volume and one or more RAID-5 data volumes. These traces cover 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.

All the traces were collected at the block level, below the file system buffer cache but above the storage tier. In this paper we consider each per-volume trace as a separate workload, and thus a workload corresponds to a single volume: we use the terms "volume" and "workload" interchangeably throughout the paper. Table 1 shows the servers traced, with the number of volumes, total number of spindles, and total storage capacity used by each. All the traces used in this paper are or will be available on the Storage Networking Industry Association's trace repository [SNIA 2009].

2.3 Related work

There has been much research in optimizing solid-state storage systems and also on hybrid systems with both solid-state and disk storage. For example, there have been proposals for optimizing B-trees [Nath 2007, Wu 2004] and hash-based indexes [Zeinalipour-Yazti 2005] for flash-based storage in embedded systems; to partition databases between flash and disks [Koltsidas 2008]; to optimize database layout for flash storage [Lee 2008]; to use flash to replace some of the main memory buffer cache [Kgil 2006]; to use M-RAM to store file system metadata [Miller 2001]; and to build persistent byte-addressable memory [Wu 1994]. These studies aim to improve storage performance but do not consider the dollar cost, i.e., when does the SSD-based solution cost less than adding more disk spindles to achieve the same performance? In this paper we use performance and capacity normalized by dollar cost to answer this question.

In the laptop storage space, vendors are 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. However the use of such hybrid or two-tiered technologies in the server space depends on the cost-performance-capacity tradeoff, which is analyzed by this paper.

Hetzler [2008] has recently argued, based on an analysis of the supply-side cost factors behind the two technologies, that flash-based storage will not achieve capacity/dollar comparable to that of disks in the foreseeable future. This is complementary to our "demand-side" work, which shows the capacity/dollar required for flash to "break even" with disk-based solutions, for different workloads. When taken

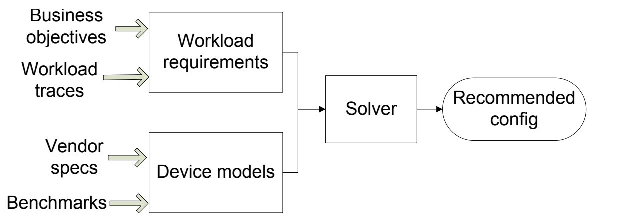

Figure 1. Tool steps

together, the two analyses present a discouraging prognosis for flash in server storage.

There have also been several cost-benefit analyses of various storage architectures and hierarchies, including those that examine the use of non-traditional storage technologies, i.e., storage not based on mechanical disks. However, to our knowledge, ours is the first study to specifically examine the cost-performance-capacity tradeoff for solid-state devices using real workloads. Here we briefly describe some of the most related previous work.

Uysal [2003] describes a cost-performance analysis of the tradeoffs involved in using microelectromechanical storage (MEMS) in the storage hierarchy. The analysis covers both replacing NVRAM with MEMS, and replacing some or all of the disks with MEMS. It shows that for workloads where performance rather than capacity dominates the provisioning cost, MEMS can be competitive with disk when it comes within 10x of the price per gigabyte of disks. However, MEMS-based storage is not commercially available today and its cost per gigabyte is unknown.

Baker [1992] explored the use of battery-backed non-volatile RAM (NVRAM) in the Sprite distributed file system to reduce disk write traffic. NVRAM is widely adopted in server 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 diskless clients in a university research environment. Server 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.

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. For example, availability is commonly expressed as the number of nines required, where

Capacity	GB
Random-access read rate	IOPS
Random-access write rate	IOPS
Random-access I/O rate	IOPS
Sequential read rate	MB/s
Sequential write rate	MB/s
Availability	Nines[†]
Reliability (MTBF[‡])	Years

† $Nines = -\log_{10}(1 - FractionOfTimeAvailable)$
‡ Mean Time Between Failures.

Table 2. Workload requirements and units.

3 nines, for example, means that the data is available 99.9% of the time. Other requirements, 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 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 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

Capacity	GB
Sequential performance	MB/s (read, write)
Random-access performance	IOPS (read, write)
Reliability	MTBF
Maximum write-wear rate	GB/year
Power consumption	W (idle and active)
Purchase cost	$

Table 3. Device model attributes and units

request rate (the 100^{th} percentile) but other percentiles such as the 95^{th} can also be used. All the analyses in this paper use peak I/O rates averaged over 1 min intervals.

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.

Random-access performance is measured in IOPS, i.e., I/O operations completed per second. When defining this "IOPS requirement" based on a workload trace, 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 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 classify each I/O in the workload trace as either sequential or non-sequential. We use LBN (logical block number) 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 chosen to be large enough to correctly detect sequential readahead, which on Windows file systems results in a small amount of request re-ordering at the block level. 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.

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 mechanical disks, we measured both the outermost 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: data requiring fast streaming performance is typically placed on these tracks. Random-access performance is based on issuing concurrent 4 KB requests uniformly distributed over the device, with a queue length of 256 requests maintained in the OS kernel. 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.

"Log-structured" SSD Our tool considers read and write performance as separate metrics. In general, mechanical disks have equivalent read and write performance. For the current generation of SSDs, while sequential read and write performance are equivalent, random-access writes are much slower than sequential writes unless a log structure is used (Section 2.1). In our analyses we consider both flavors of SSD. The default SSD model uses the measured random-write performance of the device, without assuming any remapping or log structure. The "log-structured" SSD model uses a random-write performance equal to the measured sequential write performance; it assumes that any cleaning or compaction activity will be done in the background with no impact on foreground performance.

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 the number of devices added. This assumption can be validated in practice using the synthetic tests described above. The level of fault-tolerance is a separately configurable parameter, and the model automatically adds in the appropriate number of additional device(s).

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 2008, Schroeder 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. This is measured as the number of write cycles that the SSD can tolerate before data retention becomes unreliable, i.e., the number of times that each block in the SSD can be erased and overwritten. Typical enterprise SSDs specify a wear tolerance of 10,000–100,000 write cycles. We convert this metric into a maximum write-wear rate R_{wear} (expressed in GB/year) as follows: for an SSD with a capacity C (expressed in GB), a wear tolerance of N_{wear} cycles, and a desired MTBF (mean time between failures) of T_{fail} (measured in years):

$$R_{wear} = \frac{C \cdot N_{wear}}{T_{fail}}$$

In this paper we use a default T_{fail} of 5 years, to match the typical target lifetime of a storage device before it is upgraded or replaced.

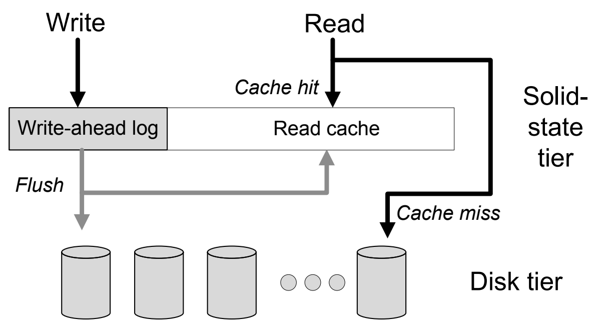

Figure 2. Two-tier architecture using SSDs and disks.

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) 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 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 server 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 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 flash storage device. The write-ahead log could also be replicated over multiple SSDs depending on workload fault-tolerance requirements. The read cache, by contrast, has only clean blocks and does not need such fault tolerance. Note that this is an inclusive caching model, where blocks stored in the SSD cache tier are also stored in the disk tier. Our tool also supports an exclusive, or "partitioning" model where each block is stored on either SSD or disk. However, given the huge disparity in capacities between SSDs and disk, a partitioning model provides negligible capacity savings. In this paper, we only use the more common (and simpler) inclusive model.

Our models use 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, and cache replacement policies such as LRU (least recently used). that aim to exploit this locality are unlikely to do well. 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 already efficient for streaming reads.

We have implemented two different caching policies in our models. The first is LRU. The second is a "long-term random-access" (LTR) policy that caches blocks based on their long-term random-access frequency. To do this, 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 to each block is used as a secondary ranking metric. Note that this policy acts like an "oracle" by computing the most frequently accessed blocks before they are accessed: hence while LRU gives an achievable lower bound on the cache performance, the LTR policy described here gives an upper bound.

For a given cache size of H blocks, the tool splits the workload trace into two, such that accesses to cached 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. The solver then iterates over values of H to find the least-cost configuration.

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 block-level versioned circular log [Narayanan 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 cached blocks, are redirected to the solid-state tier. To guarantee sequential performance for the writes, they are sent to a separately allocated log area on the solid-state device. They are then lazily flushed to the disk tier, and updated in the read cache if required. In theory, this could cause double-buffering of blocks in the write-ahead log and in the read 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. 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. The performance requirements for the write-ahead log and the disk tier are then derived by splitting the original workload trace according to the logging policy: writes are duplicated on the log and the disk tier for the "write-through" policy, and sent only to the log for the "write-back" policy (the eventual flushes to disk 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 log. 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 at one time. This gives us the amount of log space required to absorb all writes when using the "write-through" logging policy. In theory, a larger log could give us better performance when using "write-back", by making background flushing more efficient. However, in practice, we found that "write-back" did not reduce the load on the disk tier sufficiently to have an impact on provisioning: hence we did not explore this tradeoff further, but only considered the log capacity required for "write-through".

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 the cost per watt of powering the device, if known; 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. In this case we are assuming "parity-based" fault-tolerance such as RAID-5, where each additional device provides tolerance against one additional device failure. The model can also be extended to replication-based fault-tolerance such as mirroring.

The minimum cost for satisfying the workload requirements is then

$$C_{min}(w) = \min_d N(d, w) \cdot C(d) \qquad (2)$$

where $C(d)$ is the cost per device of device type d.

For two-tiered solutions, the workload trace is split into two traces $w_{top}(w, H)$ and $w_{bot}(w, H)$, where H is the size 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 for the read cache and to $F(w)$ for the write-ahead log. The cost $C_{min/tiered}(w)$ of the optimal tiered configuration is then given by:

$$C_{tiered}(w, H) = C_{min}(w_{top}(w, H)) + C_{min}(w_{bot}(w, H)) \qquad (3)$$

$$C_{min/tiered}(w) = \min_H C_{tiered}(w, H) \qquad (4)$$

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 restricts the values of H to 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; a typical trace of size 255 MB took 712 s to process on a 2 GHz AMD Opteron.

3.5 Limitations

Our analyses are based on simple first-order models of device performance. These can estimate workload requirements and device throughput in terms of random or sequential I/O rates, but not per-request response times. We believe

	Price US$	Capacity GB	Power W	Read MB/s	Write MB/s	Read IOPS	Write IOPS	Maximum write-wear rate GB/year
Memoright MR 25.2 SSD	739	32	1.0	121	126	6450	351	182500
Seagate Cheetah 10K	339	300	10.1	85	84	277	256	n/a
Seagate Cheetah 15K	172	146	12.5	88	85	384	269	n/a
Seagate Momentus 7200*	150	200	0.8	64	54	102	118	n/a

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

these first-order estimates of throughput are adequate for coarse-grained decisions such as provisioning. For more detailed characterization of I/O performance, or for workloads with specific response time requirements, more sophisticated performance models [Bucy 2008, Popovici 2003] can be substituted for the simple first-order models used here.

Our analyses are also limited by the workload traces available to us, which are block-level, open-loop traces. Thus, while they present a realistic picture of the offered load to the storage layer, this is only a lower bound on the potential I/O rate of the application: if the storage system were the bottleneck for the application, then a faster storage system would result in a higher I/O rate. The storage systems that we traced are well-provisioned and we do not believe that they were a performance bottleneck, and hence the measured storage load is a good approximation of the workload's "inherent" storage I/O requirement.

Since our traces are taken at the block level, we only analyze system changes below the block level. For example, we do not model the effect of changing the main memory buffer caches, or of using file system specific layout strategies, e.g., putting only metadata on the SSDs.

4. Results

This section has four parts. First, it presents the device characteristics extracted by the tool from a number of representative storage devices, and compares them in terms of capabilities per dollar of purchase cost. Second, it examines for each workload whether disks can be fully replaced by SSDs 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 four representative devices used in this paper. The Memoright MR 25.2 was chosen to represent SSDs since it was the fastest available mid-range (US$1000–5000 per device) SSD at the time of the analysis. The Seagate Cheetah is a widely 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; it is included as an interesting additional data point. We defer discussion of the Momentus until Section 4.4.

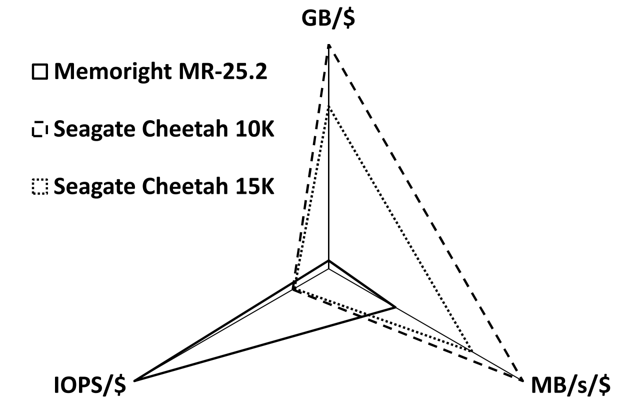

Figure 3. Device capabilities normalized by dollar cost.

For each device we considered multiple versions which differed in per-device capacity, since this leads to different cost-performance-capacity-power tradeoffs. One representative device (shown in Table 4) was chosen to extract the performance characteristics using the micro-benchmarks described in Section 3: we assumed that all versions of a given model would have similar performance. The dollar costs are US on-line retail prices as of June 2008 and the power consumptions are based on vendor documentation.

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.

Several observations can be made at this stage, even without considering workload requirements. SSDs provide more random-access IOPS per dollar whereas the enterprise disks win on capacity and sequential performance. It seems likely that the main factor in choosing SSDs versus disks is the trade-off between the capacity requirement and the IOPS re-

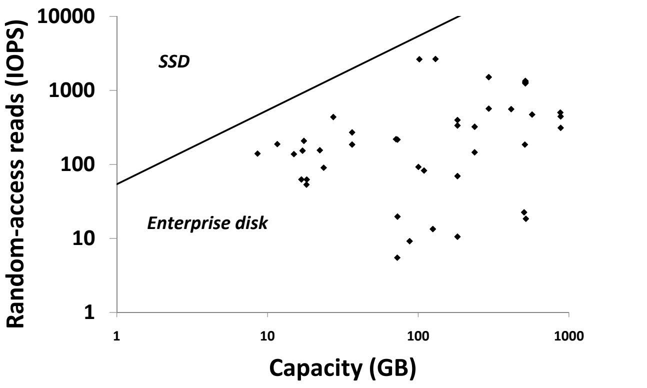

Figure 4. IOPS/capacity trade-off (log-log scale)

quirement 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.

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 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 for each of our traced volumes, for a single-tiered configuration.

Of the three enterprise-class devices shown Table 4, the Cheetah 10K disk was the best choice for all 49 volumes. In all cases the provisioning cost was determined by either the capacity or the random-read IOPS requirement. In other words, the sequential transfer and random-access write requirements are never high enough to dominate the provisioning cost. This was the case even for the default (non log-structured) SSD model, where random-access writes are more expensive than when using a log structure.

For workloads where capacity dominates the provisioning cost, clearly disks are a more cost-effective solution. However, while IOPS dominated the cost for some disk-based solutions, capacity always dominated the cost when using SSDs, due to the low capacity/dollar of SSDs. Thus when both capacity and performance requirements are considered, disks always provided the cheapest solution. Figure 4 shows the different workloads' requirements as points plotted on the two axes of random-read IOPS and capacity. The line separates the two regions where SSDs and disks respectively would be the optimal choice; none of our workloads fall in the "SSD" region.

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 workloads with the highest IOPS/GB ratio. At what capacity/dollar will SSDs become competitive with enterprise disks? We can expect that disks will continue to provide a higher capacity/dollar than SSDs, even as the latter improve over time. However (keeping the other device parameters fixed) at some point the capacity/dollar will be high enough so that IOPS rather than capacity will dominate the cost, and SSDs will become a better choice than disks.

We define the break-even point $SSD_{GB/\$}$ as the point at which the cost of an SSD-based volume (dominated by capacity) will equal that of a disk-based volume (dominated by IOPS). This gives us:

$$\frac{W_{GB}}{SSD_{GB/\$}} = \frac{W_{IOPS}}{Disk_{IOPS/\$}} \tag{5}$$

and hence

$$SSD_{GB/\$} = \frac{W_{GB}}{W_{IOPS}} Disk_{IOPS/\$} \tag{6}$$

where W_{GB} and W_{IOPS} are the workload's capacity and IOPS requirements respectively, and $Disk_{IOPS/\$}$ is the $IOPS/\$$ of the Cheetah 10K disk. In other words, SSDs will become competitive with disks when the capacity cost of the SSD equals the IOPS cost of the disk, for a given workload. Figure 5 shows this break-even point for each volume, on a log scale. For reference, it also shows the current capacity/dollar for the SSD and the Cheetah 10K disk.

The break-even point varies from 3–3000 times the capacity/dollar of today's SSDs. Some smaller volumes, especially system volumes (numbered 0 in the figure) require only a 2–4x increase in SSD capacity/dollar to consider replacement of disks by SSDs. However, most volumes will require an improvement of 1–3 orders of magnitude. For 21 of 45 volumes, the break-even point lies beyond the current capacity/dollar of the Cheetah 10K disk: this is because capacity dominates the provisioning cost of these volumes today even when using disks. If we assume that disks will retain a capacity/dollar advantage over SSDs in the future, for these workloads the capacity/dollar must first increase to the point where capacity is no longer the dominant cost for disks; beyond this point SSDs will become competitive since they cost less per IOPS.

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.

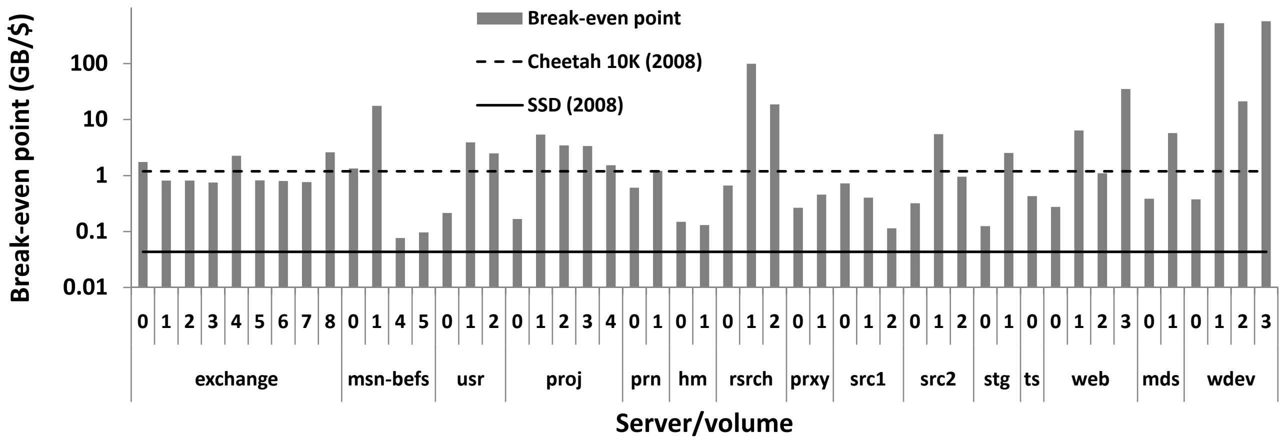

Figure 5. Break-even point at which SSDs can replace disks, on a log scale. For comparison we show the current capacity/dollar for enterprise SSDs and disks. Volumes numbered 0 are system boot volumes.

4.3.1 Write-ahead log

Across all workloads, the maximum write burst size, and hence the maximum space required for the write-ahead log, was less than 96 MB per volume. The peak write bandwidth required was less than 55 MB/s per volume: 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-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 one of the two policies described in Section 3.3.1: least-recently-used (LRU) and long-term-random (LTR). 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, in line with the "write-through" log policy.

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. For each workload we compared the best two-tiered configuration with the single-tiered disk-based configuration. Only for 4 out of 49 volumes was the two-tiered configuration the better choice at today's SSD capacity/dollar. For each workload, we also calculated the break-even capacity/dollar at which the best two-tier solution would equal the cost of a disk-based solution. For 40 out of 49 volumes, this break-even point was the same as the break-even point for replacing all the disks with SSDs. In other words, the cache tier does not absorb enough I/Os for these 40 volumes to enable any reduction of spindles in the disk tier. In some cases, this is because the volume's provisioning is determined by capacity requirements, which are not reduced by caching. In other cases, the cache hit rates are too low to significantly reduce the peak load on the disk tier: we speculate that this is because main-memory buffer caches have already removed much of the access locality.

For the 9 workloads where caching has some benefit, Figure 6 shows the break-even point: the SSD capacity/dollar required for the two-tiered solution to become competitive. The last 4 workloads in the figure have a break-even point below today's SSDs capacity/dollar, i.e., they can already

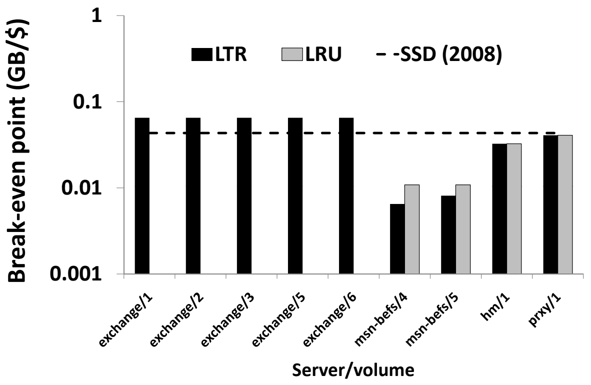

Figure 6. Break-even points for workloads to become cacheable (log scale). Only the LTR policy is shown for the Exchange volumes; with LRU there is no benefit and hence no break-even point for these volumes.

benefit from an SSD cache tier. The first 5 workloads appear to be cacheable with a break-even point not far from today's SSD capacity/dollar, but only with the LTR policy. Since the LTR policy is used in "oracle" mode, it is not clear that its benefits are achievable in practice. With LRU these workloads show no benefit from caching. However, we only have 24-hour traces for the Exchange workloads; longer traces are needed to decide if the workloads contain longer-term locality patterns that LRU could exploit.

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 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 2008, Schroeder 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 random-read IOPS, all normalized by the device's power consumption. We use idle power numbers (device ready/disks spinning, but no I/O), since our workloads have 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,

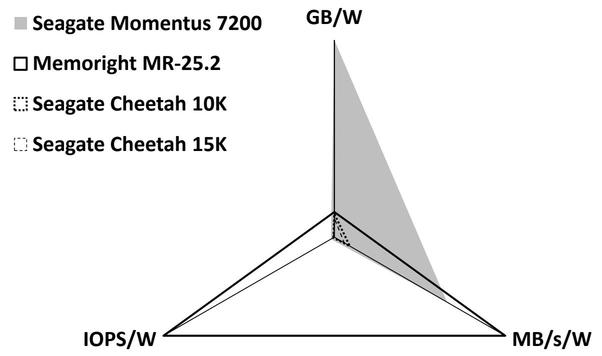

Figure 7. Device capabilities normalized by power consumption.

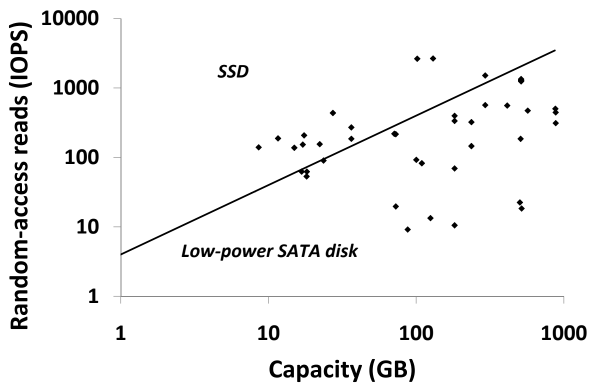

Figure 8. IOPS/capacity trade-off when optimizing for power (log-log scale)

low-speed Momentus does far better than the SSD in terms of gigabytes per watt, and better than the enterprise disk on all three power-normalized metric. We also found that the Momentus significantly outperformed the Cheetahs on capacity and performance per dollar; however, the price advantage might only reflect market forces, whereas power is more a property of the underlying technology.

For our workloads, the SSD was the lowest-power single-tiered solution for 11 out of 49 workloads, and chooses the Momentus for the remaining 38. Again, random-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). This is the energy price at which the power savings over 5 years of an SSD-based solution will equal the additional purchase cost. We show the break-even price for the SSD against the Cheetah for all 49 volumes, and for the SSD against the Momentus

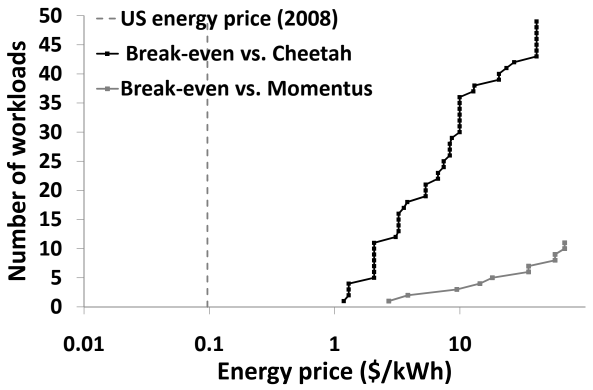

Figure 9. CDF of 5-year break-even energy price point for SSDs (log scale)

for the 11 volumes where the SSD was more power-efficient than the Momentus. For reference we show the commercial US energy price as of March 2008 [US Department of Energy 2008].

Note that 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, a combination of falling SSD per-gigabyte prices rising energy prices could motivate the replacement of disks by SSDs in a few of the smaller volumes.

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 that the pervasive use of RAID for fault-tolerance 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 2008, Schroeder 2007]. Such empirical evidence is lacking for SSDs. 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. Here we show the results of an analysis based on the wear metric: our purpose is to judge whether SSDs deployed in any particular configuration are likely to fail before their nominal lifetime, i.e., before the next hardware upgrade.

Based on the long-term write rate of each workload, we computed the time after which the wear-out 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 relies on using wear-leveling techniques [Birrell 2007, Gal 2005] that avoid in-place updates

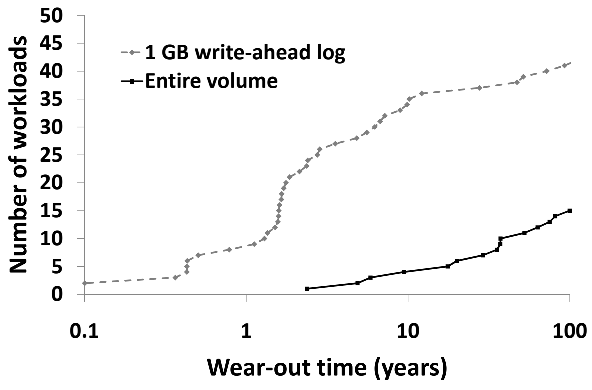

Figure 10. CDF of wear-out times (log scale)

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 majority of volumes have wear-out times exceeding 100 years. All volumes with the exception of one small 10 GB volume have wear-out times of 5 years or more. 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 relatively small flash device absorbs all the writes sent to a volume. 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. Here we assume no overhead, since a circular log used for short-term persistence does not need to be defragmented. However, due to the relatively small size of the log, each block in the log gets overwritten frequently, and 28 out of 49 workloads have a wear-out time of less than 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 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

Flash-based solid-state drives (SSDs) are a new storage technology for the laptop and desktop markets. Recently "enterprise SSDs" have been targeted at the server storage market, raising the question of whether, and how, SSDs should be used in servers. This paper answers the above question by doing a cost-benefit analysis for a range of workloads. An additional contribution of the paper is a modeling and optimization approach which could be extended to cover other solid-state technologies.

We show that, across a range of different server workloads, replacing disks by SSDs is not a cost-effective option at today's prices. Depending on the workload, the capacity/dollar of SSDs needs to improve by a factor of 3–3000 for SSDs to be able to replace disks. The benefits of SSDs as an intermediate caching tier are also limited, and the cost of provisioning such a tier was justified for fewer than 10% of the examined workloads.

Acknowledgments

We thank Bruce Worthington, Swaroop Kavalanekar, Chris Mitchell and Kushagra Vaid for the Exchange and MSN storage traces used in this paper. We also thank the anonymous reviewers and our shepherd John Wilkes for their feedback.

References

[Agrawal 2008] Nitin Agrawal, Vijayan Prabhakaran, Ted Wobber, John D. Davis, Mark Manasse, and Rina Panigrahy. Design tradeoffs for SSD performance. In *USENIX Annual Technical Conference*, pages 57–70, Boston, MA, June 2008.

[Anderson 2003] Dave Anderson, Jim Dykes, and Erik Riedel. More than an interface - SCSI vs. ATA. In *Proc. USENIX Conference on File and Storage Technologies (FAST)*, pages 245–257, San Francisco, CA, March 2003.

[Anderson 2005] 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.

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

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

[Bucy 2008] 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 2005] Eran Gal and Sivan Toledo. Algorithms and data structures for flash memories. *ACM Computing Surveys*, 37(2): 138–163, 2005.

[Golding 1995] Richard Golding, Peter Bosch, Carl Staelin, Tim Sullivan, and John Wilkes. Idleness is not sloth. In *Proc. USENIX Annual Technical Conference*, pages 201–212, New Orleans, LA, January 1995.

[Hetzler 2008] Steven R. Hetzler. The storage chasm: Implications for the future of HDD and solid state storage. `http://www.idema.org/`, December 2008.

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

[Jiang 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)*, pages 111–125, San Jose, CA, February 2008.

[Kgil 2006] 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)*, pages 103–112, Seoul, Korea, October 2006.

[Koltsidas 2008] Ioannis Koltsidas and Stratis Viglas. Flashing up the storage layer. In *Proc. International Conference on Very Large Data Bases (VLDB)*, pages 514–525, Auckland, New Zealand, August 2008.

[Lee 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)*, pages 1075–1086, Vancouver, BC, June 2008.

[Miller 2001] Ethan Miller, Scott Brandt, and Darrell Long. HeRMES: High-performance reliable MRAM-enabled storage. In *Proc. IEEE Workshop on Hot Topics in Operating Systems (HotOS)*, pages 95–99, Elmau/Oberbayern, Germany, May 2001.

[Narayanan 2008a] Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron. Write off-loading: Practical power management for enterprise storage. In *Proc. USENIX Conference on File and Storage Technologies (FAST)*, pages 256–267, San Jose, CA, February 2008.

[Narayanan 2008b] Dushyanth Narayanan, Austin Donnelly, Eno Thereska, Sameh Elnikety, and Antony Rowstron. Everest: Scaling down peak loads through I/O off-loading. In *Proc. Symposium on Operating Systems Design and Implementation (OSDI)*, pages 15–28, San Diego, CA, December 2008.

[Nath 2007] Suman Nath and Aman Kansal. FlashDB: Dynamic self tuning database for NAND flash. In *Proc. Intnl. Conf. on Information Processing in Sensor Networks (IPSN)*, pages 410–419, Cambridge, MA, April 2007.

[Panabaker 2006] 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.

[Popovici 2003] Florentina I. Popovici, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Robust, portable I/O scheduling with the Disk Mimic. In *Proc. USENIX Annual Technical Conference*, pages 297–310, San Antonio, TX, June 2003.

[Prabhakaran 2008] Vijayan Prabhakaran, Thomas L. Rodeheffer, and Lidong Zhou. Transactional flash. In *Proc. Symposium on Operating Systems Design and Implementation (OSDI)*, pages 147–160, San Diego, CA, December 2008.

[Rosenblum 1991] Mendel Rosenblum and John Ousterhout. The design and implementation of a log-structured file system. In *Proc. ACM Symposium on Operating Systems Principles (SOSP)*, pages 1–15, Pacific Grove, CA, October 1991.

[Samsung 2006] Samsung. MH80 SATA product data sheet, June 2006.

[Schroeder 2007] Bianca Schroeder and Garth 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)*, pages 1–16, San Jose, CA, February 2007.

[SNIA 2009] SNIA. IOTTA repository. `http://iotta.snia.org/`, January 2009.

[Strunk 2008] 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)*, pages 313–328, San Jose, CA, February 2008.

[US Department of Energy 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.

[Uysal 2003] Mustafa Uysal, Arif Merchant, and Guillermo Alvarez. Using MEMS-based storage in disk arrays. In *Proc. USENIX Conference on File and Storage Technologies (FAST)*, pages 89–102, San Francisco, CA, March 2003.

[Woodhouse 2001] David Woodhouse. JFFS: The journalling flash file system. `http://sources.redhat.com/jffs2/jffs2.pdf`, July 2001.

[Worthington 2008] Bruce Worthington, Swaroop Kavalanekar, Qi Zhang, and Vishal Sharda. Characterization of storage workload traces from production Windows servers. In *Proc. IEEE International Symposium on Workload Characterization (IISWC)*, pages 119–128, Austin, TX, October 2008.

[Wu 2004] Chin-Hsien Wu, Li-Pin Chang, and Tei-Wei Kuo. An efficient B-tree layer for flash-memory storage systems. In *Proc. Real-Time and Embedded Computing Systems and Applications (RTCSA)*, pages 409–430, Gothenburg, Sweden, August 2004.

[Wu 1994] 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)*, pages 86–97, San Jose, CA, October 1994.

[Zeinalipour-Yazti 2005] Demetrios Zeinalipour-Yazti, Song Lin, Vana Kalogeraki, Dimitrios Gunopulos, and Walid A. Najjar. Microhash: An efficient index structure for flash-based sensor devices. In *Proc. USENIX Conference on File and Storage Technologies (FAST)*, pages 31–44, San Francisco, CA, December 2005.