# Enterprise Storage Provisioning with Flash Drives

Ross Shaull
Brandeis University
rshaull@cs.brandeis.edu

Eitan Bachmat
Ben-Gurion University
ebachmat@cs.bgu.ac.il

Tal Ron
Ben-Gurion University

Adi Littman
Ben-Gurion University

Hadar Mor
Ben-Gurion University

Elad Shmidov
Ben-Gurion University

## Abstract

In the past, enterprise storage systems were configured with high-end disk drives supporting the SCSI or Fiber Channel protocol. In the last two years, flash drives and low cost SATA drives have entered the enterprise storage market as storage device options. In this paper we present an analytical tool for assessing the configurations formed from a mixture of all device types. Rather than relying on large-scale fine-grained traces, which are very rare, our tool uses ubiquitous coarse-grained logical volume statistics which are readily available in most production systems. We use our tool to analyze logical volume statistics collected from 120 large production systems. We show that mixing flash, SCSI, and SATA drives can lead in most cases to configurations which are better than using only SCSI devices in all key aspects: price, performance and energy consumption. This contrasts with other recent studies on smaller enterprise systems which are pessimistic about the advantages of flash drives in the enterprise setting.

## 1. Introduction

Traditionally, enterprise disk arrays were configured using expensive, fast, low capacity, power consuming enterprise drives, which supported either the SCSI or FC protocol. In the meantime, the storage world for personal computing was dominated by cheap, slow, high capacity, low power drives, supporting the SATA protocol. A few years ago, these SATA derives were introduced into the enterprise market. In a world with only these two options, combining both drive types in a single system necessitates a tradeoff between price/capacity and performance. Even more recently, solid state drives (SSDs), commonly known as flash drives, have been introduced into enterprise class disk arrays (e.g., [24]).

The addition of flash drives to the mix of current drives raises intersting new configuration possibilities which we explore in this paper.

With three components we are faced with the intriguing possibility that one configuration may be better than another in all aspects, faster, cheaper and less power consuming per equal capacity. This can occur when we replace a system configured with fast, expensive disk drives (SCSI drives), with a system comprised of a small amount of flash drives (SSD drives) with the rest of the system consisting of capacity oriented drives (SATA drives).

There are two ways to manage flash drives in a massive storage system. The first is to manage flash in the same way one manages DRAM cache [22]. The second is to manage it as a disk drive [21]. The first option is very appealing and in the long run will be the main method of application, however in most systems it will require many design changes. The second option is very appealing in that the flash drive interface is essentially the same as that of a disk drive, which allows use of flash with hardly any system changes, therefore the initial uses will be of the second type.

In addition, it is currently difficult to conduct a large scale study of the effectivness of using flash as a DRAM extension. The reason is that accurate assesment of cache hit ratios requires traces. As the cache becomes very large, the traces needed for the assesment become very large. Large real traces are extremely rare.

For these reasons, in this paper we will concentrate on the second type of use, i.e., of flash drives as a direct replacement of ordinary drives.

Several papers have suggested that flash drives can improve the performance of enterprise applications [14, 21], but these didn't consider it on an equal capacity and cost basis. When cost and capacity are taken into account, two studies [15, 25], concluded that SSD drives are not cost effective in the enterprise system. The studies used traces from a relatively small number of small enterprise systems.

In this paper we examine the issue of configuring enterprise storage systems with a mixture of drive types. We use data which is ubiquitous for such systems, namely, logical volume-level counter statistics. These statistics tell us about the activity of each logical volume in the system, every few minutes over an extended time period. This allows us to conduct a survey of many production arrays. Since we use coarse data we can only draw direct conclusions about the effectivness of using a small amount of flash drives, as disk drive replacements. However, as we noted, this is the less efficient mode of using flash, so using them as a DRAM extension will be equally or even more efficient.

To use the data we develop a simple analytical model for the cost, power consumption and average response time of an array which is configured using a particular drive mixture. The cost and power consumption models are straightforward linear models, while the performance model is a bit more involved and in particular is non linear. We also explain why our model tends to be conservative, an important property for models which rely on coarse data. Our end result is a simple capacity planning tool which suggests to the user a set of Pareto optimal configurations, i.e., configurations which beat any other configuration in at least one aspect, price, performance or power consumption. The user can then choose among those configurations, the one best suited for them.

We employ an implementation of our tool to analyze usage statistics from 120 production storage servers. Unlike the previous studies, we find that in a majority of cases, provisioning a mix of SCSI, SATA, and flash drives can improve performance and energy consumption without exceeding the cost to provision an all-SCSI system.

## 2. Storage System Characteristics

We provide a brief general description of the architecture of the type of storage systems which concern us in this paper. The main physical system components include directors, cache memory and secondary storage devices (disks).

### 2.1 Components

The system is comprised of two main types of components, directors and storage components. The storage components are further divided into primary storage (cache) and secondary storage (disks).

The computational heart of the storage system is a set of CPUs called directors, which manage incoming host requests to read and write data and direct these requests to the appropriate storage components, either cache or secondary storage, which actually keep the data.

### 2.2 Cache Memory

Cache memory (DRAM) is a fast and expensive storage area. The cache (DRAM) is managed as a shared resource by the directors. The content of the cache is typically managed by a replacement algorithm which is similar to FIFO or the Least Recently Used (LRU) algorithm. In addition, data can be prefetched in advance of user requests if there is a good probability that it will be requested in the near future. Additionally some data may be placed permanently in cache if it is very important, regardless of how often it is used. Whatever data is not stored in cache, resides solely on secondary storage. Typically, the cache comprises a very small portion of the total storage capacity of the system, in the range of $0.1$ percent. This is due to the prohibitive cost ratio of DRAM to disk.

### 2.3 I/O operations and caching

Four basic types of operations occur in a Storage system: Read Hits, Read Misses, Write Hits, and Write Misses. A *Read Hit* occurs on a read operation when all data necessary to satisfy the host I/O request is in cache. The requested data is transferred from cache to the host.

| Type | Capacity (#volumes) | Cost (Dollars) | Overhead per I/O (seconds) | R / W rate (MB / second) | Energy (watts) |
|---|---|---|---|---|---|
| SSD | 15 | 1000 | 0.0001 | 160 / 120 | 6 |
| SCSI | 30 | 100 | 0.0040 | 60 / 60 | 12 |
| SATA | 100 | 100 | 0.0100 | 50 / 50 | 12 |

**Table 1.** Estimated storage characterstics

A *Read Miss* occurs when not all data necessary to satisfy the host I/O request is in cache. A director stages the block(s) containing the missing data from secondary storage. The Director places the block(s) in a cache page. Simultaneously, a Director (possibly different from the first) reconnects to the host and sends the requested data.

The cache is also used for handling write requests. When a new write request arrives at a director, the director writes the data into one or more pages in cache. The storage system provides reliable battery backup for the cache (and may also employ cache mirroring to write the change into two different cache boards in case a DRAM fails), so write acknowledgements can be safely sent to hosts before the page has been written (destaged) to secondary storage. This allows writes to be written to secondary storage during periods of relative read inactivity, making writing an asynchronous event, typically of low interference. This sequence of operations is also called a *write hit*.

In some cases the cache fills up with data which has not been written to secondary storage yet. The number of pages in cache occupied by such data is known as the *write pending count*. If the write pending count passes a certain threshold, the data will be written directly to secondary storage, so that cache does not fill up further with pending writes. In that case we say that the write operation was a *write miss*. Write misses do not occur frequently, as the cache is fairly large on most systems. A write miss leads to a considerable delay in the acknowledgment (completion) of the write request.

Not every write hit corresponds to a write I/O to secondary storage. There can be multiple write operations to the same page before it is destaged to secondary storage, resulting in write *absorption*. Multiple logical updates to the same page are absorbed into a single destaging I/O to secondary storage.

### 2.4 Disks

Table 1 shows the estimated disk characteristics we use to calculate the performance of a particular provisioning, based on information from commercial system vendors. Our fundamental unit of storage is a volume (since our provisioning does not allow a volume to span multiple drives), so the capacity of a drive is measured in the number of volumes that fit on that drive. The cost of a drive is in US dollars, conservatively estimated from a recent search of prices for server-grade drives.

The I/O overhead is conservatively estimated from server-grade drive data sheets. The overhead is the average latency to complete an I/O, and depends on the drive firmware, interconnect, and rotational latency (for SCSI and SATA). Sequential I/O is less costly than random I/O in rotating disks, so for sequential read misses we decrease the estimated overhead by half. SSDs do not have a seek penalty, so the latency remains unchanged regardless of whether an I/O is sequential or random.

The read / write rate is the speed at which bytes can be read from the disk. For rotating disks, this speed is relatively stable and relatively close to the speed of flash drives (as compared to the difference in overhead between SSDs and rotating disks). Flash drives have the highest throughput, although it is lower for writes than reads. This is because SSDs do not support random rewrite operations, instead they must perform *erasure* on a large segment of memory (a slow process), and can only *program* (or set) an erased block. This tradeoff underscores the importance of analyzing the statistics of a storage system before provisioning.

The exact energy consumption of a drive depends on its capacity, the number of I/Os it receives, and its idle power consumption. We use a rule of thumb that ranks energy consumption SSD $<$ SATA $<$ SCSI. Flash drives use the least energy of any of the drive types, but because they have a significantly lower capacity, the energy consumption of a system comprised of SSDs

may be higher than, say, the same system comprised on SATA drives (because it would need many more SSDs than SATA drives to store the same amount of data). This is another tradeoff that makes provisioning heterogenous storage systems complex.

## 3. Logical volumes

The data in the system is divided into units, which are called logical volumes. A logical volume will typically span 5–10 GB. The volume is a unit of data which is referenced in the I/O communication between the host and the storage system. The logical volume is divided into blocks of fixed size (usually 512 Bytes). Typically an I/O request will consist of an operation, read or write, a logical volume number, a block offset within the logical volume and a size in blocks. For example, a request might be to read 32 blocks from logical volume number 2037, starting with block 1,825,345 within the logical volume.

### 3.1 Logical volume statistics

Logical volumes are the basic unit for reporting statistics. There are counters which record statistics for each logical volume. From such counters, our provisioning algorithm makes use of the following statistics:

- Read misses
- Write misses
- Bytes read
- Bytes written
- Percent of read misses that resulted in random I/O (a seek)
- Sequential read requests

We are particularly interested in read and write *misses*, since these actually incurr a cost from the storage system drives; read hits can be serviced by the cache, and write hits may be absorbed in the cache before any writes make it to a actual drive.

## 4. Data

Our data set was provided by EMC systems, and consists of the counter values described in section 3.1 (along with many other counters we did not use in this study), captured at intervals of 10 minutes during one day, for a total of 146 10-minute periods. Some counters represent averages; for example, the counter for read misses reports the average number of read misses

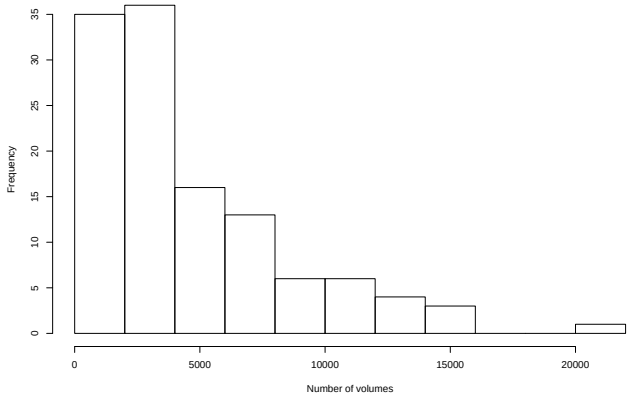



**Figure 1.** Histogram of volumes per machine

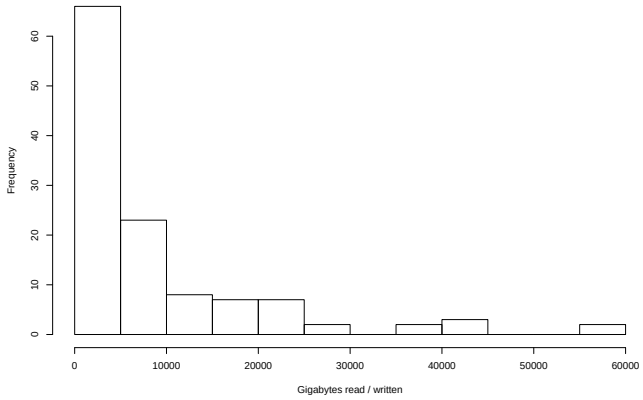



**Figure 2.** Histogram of activity per machine

per second during the entire period. Others are percentages, such as the percentage of read misses which required random I/O. The bytes read and written are summations of the sizes of each read and write I/O during the period (respectively). A small number of the machines in our data set appeared to be completely inactive, so we elided them from our study. In this paper we examine the activity patterns of 120 active production systems comprising a total of 604765 volumes.

The total size of in terms of the number of logical volumes in each machine varies, but is in general quite large. The smallest machine in our data set holds 350 volumes, while the largest holds 20590 volumes. Figure 1 shows a histogram of the number of volumes in each machine; most machines have between 350 and 5000 volumes. The number of bytes read and written on a single machine varies in our data set from very small

(order of megabytes) to very large (order of terabytes). Figure 2 shows a histogram of the number of gigabytes read / written. 84% of the machines in our data set read or write more than 100 GB total, with some machines transferring significantly more.

## 5. Provisioning Algorithm

To perform our calculations we need to make an assumption on the actual size of logical volumes (LVs), so we can compute the minimal number of storage devices which are needed to store them. We make the reasonable assumption that all LVs have a size of 10GB. The essence of our computations and our results is not dependent on this assumption, but our specific numerical computations are.

Using this assumption we can produce the basic capacity inequality that all legitimate configurations must satisfy.

We will attach the index 1 to flash drives, the index 2 to SCSI drives and the index 3 to SATA drives.

We let $n_1$ denote the number of flash drives in a suggested configuration, $n_2$ denote the number of SCSI drives and $n_3$ denote the number of SATA drives. We let $C_1$, $C_2$ and $C_3$ be, respectively, the capacities of the different types of devices. Table 1 shows the capacities we assumed for each drive type in this study, normalized by the smallest capacity drive type (SSD). Let $C$ be the total capacity of the system. Let $L$ be the number of LVs. For the system to have the required capacity it must satisfy

$$10 \times L \leq C = C_1 * n_1 + C_2 * n_2 + C_3 * n_3$$

If this inequality is not satisfied then we will not have enough capacity to store the data. We also note that we have not taken into account the added storage which is needed for redundancy, but this will have the same effect as declaring that a LV has a larger size than 10GB.

Let $power_1$, $power_2$, and $power_3$ denote the power consumption of the respective devices, say in units of Watts. The power consumption of a configuration is,

$$\begin{aligned} power &= power_1 * n_1 + \\ & \quad power_2 * n_2 + \\ & \quad power_3 * n_3 \end{aligned}$$

Similarly, if $price_1$, $price_2$, and $price_3$ are the prices of the respective devices, then the price of a configuration is,

$$\begin{aligned} price &= price_1 * n_1 + \\ & \quad price_2 * n_2 + \\ & \quad price_3 * n_3 \end{aligned}$$

A bit more challenging is the issue of performance. We assume that all the configurations will contain the same DRAM cache which is currently in the system from which statistics are measured; therefore, the secondary devices will only observe the read and write misses.

We sort the LVs with respect to their total number of misses (reads+writes), from highest to lowest. We place on the fastest devices, the flash devices, the LVs with the most misses until the total capacity of the flash drives in the configuration is filled. Then, we continue in the same manner, placing the remaining highest-activity LVs on the SCSI drives, until their capacity is also reached. The remaining LVs are placed onto SATA drives. This simple approach partitions the LVs such that faster drives will service a larger percentage of all I/Os than slower drives.

Explicitly, we place the $\frac{C_1 * n_1}{C} L$ LVs with the most misses on flash drives, the next $\frac{C_2 n_2}{C} L$ LVs with the most misses on SCSI drives and the rest on SATA drives.

Next, we compute the expected utilization $U_1(t), U_2(t), U_3(t)$ for each time slice $t$ and each device type. Utilization is the portion of time that the system spends on servicing I/O. A utilization of $0.7$ means that the system is working $70\%$ of the time while it is idle in the remaining $30\%$. For each device type $i$ and time slice $t$, we compute the utilization as follows. The total amount of work time, measured in seconds, available on all the devices of type $i$ during the time slice $t$ is

$$D_i(t) = n_i x D(t) \tag{1}$$

where $D(t)$ is the duration of a time slice in seconds. We recall that in all our data $D(t)$ is fixed to be $600$ (10 minutes). Let $KR_i(t)$ be the total amount of kilobytes read and $KW_i(t)$ be the total amount of kilobytes written to the devices of type $i$ during time slice $t$. The total amount of time it takes to read and write all the data is

$$(KR_i(t)/R_i) + (KW_i(t)/W_i)$$

where $R_i, W_i$ is respectively the number of kilobytes read or written per second on a device of type $i$. In addition there is an overhead for any I/O operation which is performed on the device. This overhead is negligible in SSD devices, but is substantial in disks and amounts to the seek and latency between I/O operations. We let $O_i$ denote the average overhead per I/O on a device of type $i$. The total amount of time required for overhead is $O_i miss_i(t)$. We distinguish between random and sequential operations. Random I/O are assesed the latency penalty given in the table, while sequential I/O are assesed only half the penalty. This is because two sequential I/O from the same sequential stream, which are excecuted one after the other will have only a small amount of overhead, while a sequential I/O preceeded or followed by a random I/O, or a sequential I/O from another stream will lead to latency.

The total time required for read and write operations is the sum of the above quantities, namely

$$T_i(t) = (KR_i(t)/R_i) + (KW_i(t)/W_i)O_i miss_i(t)$$

and the utilization is

$$U_i(t) = T_i(t)/D_i(t) \tag{2}$$

Once we have computed the utilization we use the M/M/1 queueing model for each device. In this model the total waiting time (response time) of requests during time slice $t$ on devices of type $i$ is given by

$$TW_i(t) = \frac{T_i(t)}{1 - U_i(t)} \tag{3}$$

We note that without the utilization factor the waiting time would not depend on the number of devices, therefore taking the utilization into account is crucial. In addition, we note that it may happen that we get a utilization number $U_i(t)$ which is above 1 for some device type $i$ and a time slice $t$. This means that for that during time slice $t$ the arrival rate of I/O requests is larger than the service rate and a very large queue of requests will form, leading perhaps to time outs, which are highly undesirable. To avoid such situations we consider a utilization above $0.4$ at any time slice, for any device type to be unacceptable. This leads to another constraint on the number of devices of a given type which is different from the capacity constraint.

In the computation we consider all configurations which minimally satisfy both the capacity constraint and the utilization constraint. By minimally we mean that no configuration which has fewer devices of each type satisfies both constraints. The computation of minimal configurations is done by first satisfying the capacity constraint, then assigning the LVs as described above. Next we compute the total activity $T_i(t)$ for each device type and time slice. We then compute the minimal number of devices needed to satisfy the utilization constraint using equation 2.

In general for a given set of counter data there are many (hundreds or more) such configurations, which we call *legitimate*. Our performance target function for legitimate configurations is the total response time of all device activity over all time periods

$$P = \sum_t \sum_i TW_i(t)$$

Since our assesment of performance numbers is based on coarse statistical data it is unlikely to be precise. Under such circumstances, it is important that the estimates will be conservative in an appropriate sense. We claim that our use of utilization in equation (3) favors disk drives over flash drives. The reason is that $U_i(t)$ is computed assuming uniform activity over devices of the same type and more importantly, under the assumption that the activity is uniformly spread throughout the time slice $t$. This later assumption is somewhat improbable, as I/Os tend to be bursty [28]. As a result, equation (3) tends to underestimate the actual response time delay due to utilization. Since flash drives handle peak loads much better than disk drives, they can handle bursty activity much better than disk drives. By using a uniformly spread model of activity we diffuse bursty behavior and hide to some extent the performance issues associated with handling bursty activity with disk drives. we conclude that in reality, we expect flash drives to be even more helpful than our estimates suggest. The issue of bursty activity explains why we have chosen the mild utilization limit of $0.4$. we simply expect that during some parts of the time slice the activity was even more intense and hence we should make a conservative choice for the utilization upper bound.

## 6. Evaluation of Provisioning Algorithm

When configuring a storage system, typically there is a cost ceiling based on available capital, and minimum

performance goals driven by business concerns. A relatively new concern is the energy consumed by a storage system, driven in part by the trend toward "green computing", but also by the bottom line, since energy consumption contributes to the ongoing cost of operating the storage system. In this evaluation, we take the baseline cost, performance, and energy consumption to be that of a provisioning of 100% SCSI drives. That is, we assume that the business need can be minimally met by homogenously provisioning a storage server to be all SCSI, on the assumption that SCSI is a compromise that has better per-unit performance than SATA and lower per-unit cost than SSD. We never allow a provisioning that fails to best the 100% SCSI provisioning at performance, cost, and energy consumption.

We find the pareto optimal provisionings iteratively using the algorithm described in section 5. Our tool can examine the possible provisionings for a given storage system, as well as find the best provisionings for each of our three criteria (performance, cost, energy consumption) among our entire data set. In section 6.1, we look at the output for a single storage system, and consider how it could be used to choose a provisioning based on business needs. In section 6.2, we examine the trends over all storage systems in our data set, and draw conclusions about the liklihood of flash being a beneficial part of storage system provisionings in current storage systems. Finally, in section 6.3, we look forward to see how these trends change as flash decreases in price.

### 6.1 Example: Provisioning a System

As described in section 5, for each system analyzed there will be a set of *legitimate* points which satisfy both the capacity and utilization constraints derived from counter data. Out of these legitmate points, many of them may be worse than a 100% SCSI configuration in either performance, cost, or energy consumption. In order to reduce the search space in practice, we eliminate as possibilities all legitimate points which are not better than a 100% SCSI configuration in all respects. We can visualize the remaining *acceptable* points in a 3-dimensional plot, and consider which one represents the best compromise between performance, cost, and energy consumption.

Figure 3 shows such a plot for an example system chosen from our data set. The x-, y-, and z-axes show the computed characteristics of each acceptable provisioning for this system (performance, cost, and energy use). Points are drawn as impulses to aid in visualization the location of the point in 3-space; each impulse is drawn from the location of the data point to the floor of the plot in the x-y plane. The color of each impulse indicates the percentage of SSDs that were used in the provisioning (1%, 2%, or 3%; provisionings with higher percentages of SSDs were not *acceptable*). The 3 "best" provisionings are marked with stars. Each of the best provisionings is either the best performing, least expensive, or lowest energy consumer out of all provisionings (respectively). Clockwise starting from the left-most star, the stars mark the best-performing, lowest energy consumer, and cheapest provisionings.

The system shown in figure 3 uses a small amount of flash drives (as section 6.2 will show, this is the common case). Most of the provisionings place 1% or 2% of volumes on SSDs. Only 3 configurations place 3% of volumes on SSDs; these are the most expensive. Interestingly, using a larger percentage of SSDs does not necessarily mean a better-performing system over all. While the best-performing system provisions the most SSDs, it is also the most expensive since it uses many SCSI drives as well.

As expected, the least-expensive configuration uses mostly SATA and SCSI drives, provisioning only 1% of volumes on SSDs. The lowest energy-consuming system increases the overall cost in order to provision 2% of volumes to SSDs, making it possible to eliminate SCSI drives entirely. Between the "best" configurations, numerous other provisionings can be made which are all improvements over a 100% SCSI configuration. For example, the point at (cost=17700, response time=2.68, energy use=984) is an interesting compromise; it is uses 12% more energy than the "best" configuration for energy use, yet it only costs 5% more and performs nearly twice as well, to within 60% of the "best" configuration for performance.

This simple approach reduces the space of possible configurations to a relatively small set of acceptable provisionings and estimates the characteristics of each. The "best" provisionings which optimize a specific axis can be identified, and a simple equation or even visual inspection can be used to identify a suitable compromise between the competing factors of performance, cost, and energy consumption. In the next section, we consider the best provisionings estimated from counter data for all systems in our data set in order to under-

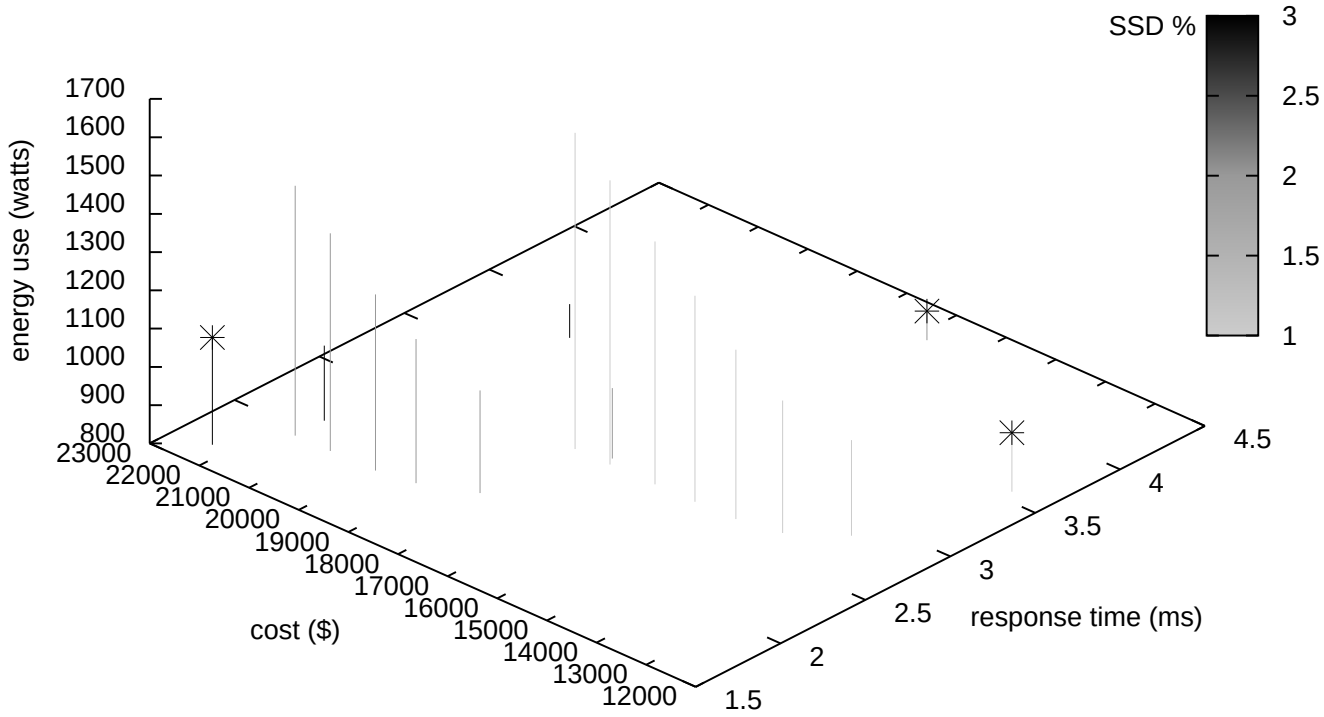



**Figure 3.** Example output from our tool: acceptable provisionings (smaller is better along all axes) for a representative machine from our data set

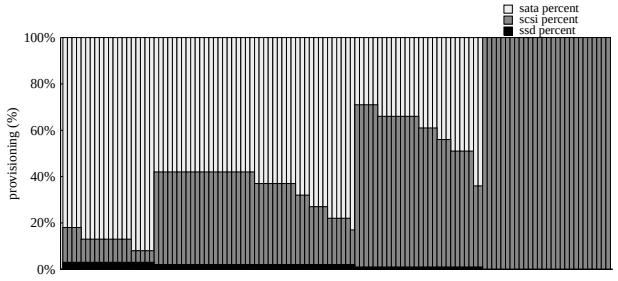
stand, in general, how SSDs will be provisioned in data centers in the near future.

### 6.2 Best Provisionings

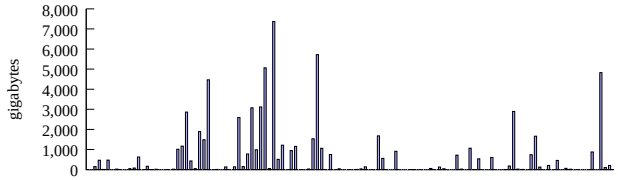
**Maximizing performance.** We first experimented with selecting the provisioning for each storage system which yielded the smallest average response time (figure 4). 76% of systems in our data set benefited from including SSDs in the storage provisioning. In each case where SSDs are provisioned, some SATA drives are also provisioned to compensate for the high cost of flash. As figure 4a shows, there is an inverse relationship between the number of SCSI drives and SSDs that need to be provisioned to maximize performance; SSDs take the place of SCSI drives for handling the busiest volumes, while the less-busy volumes can be placed on SATA drives. What seems to be a cap on the number of SSDs is due to the high cost of SSDs; since we do not select any configuration which exceeds the cost of an all SCSI system, the number of SSDs that can be provisioned in each system is limited.

Figure 4c shows the estimated response time of the best provisioning for performance relative to the performance of a all SCSI configuration (smaller is better). 25% of systems achieve a 20% or better improvement to average response time for the same or lesser cost as an all-SCSI provisioning when configuring for best performance. There is a weak inverse correlation between the total number of bytes read in the system (figure 4b) and the performance improvement enabled by provisioning SSDs. This is because the advantage of SSDs over rotating disks is less significant when costs are dominated by sustained transfers and not seeks. However, even the systems with the most bytes transferred can benefit from SSDs, suggesting that SSDs can be affordably employed to improve peformance in systems with a variety of workloads.

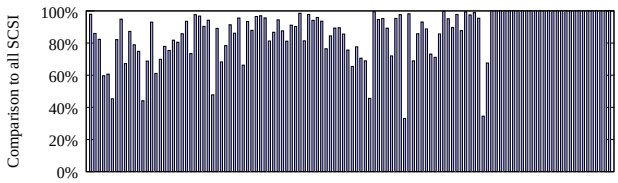
**Minimizing energy consumption.** Provisioning with energy in mind means storing the volumes on as




(a) Best provisioning

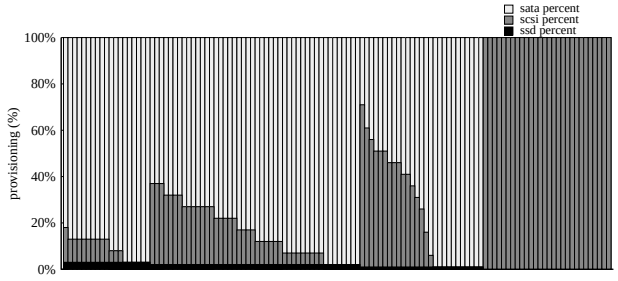



(b) Total read size

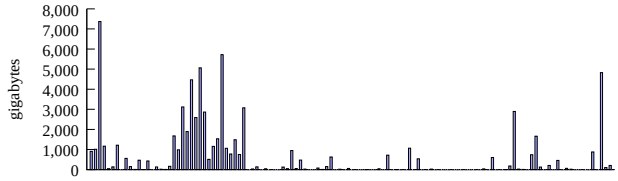



(c) Change in average response time (smaller is better)

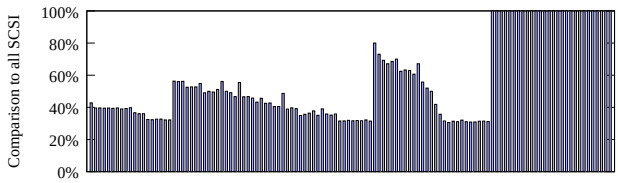
**Figure 4.** Maximizing performance




(a) Best provisioning




(b) Total read size




(c) Change in energy consumption (smaller is better)

**Figure 5.** Minimizing energy consumption

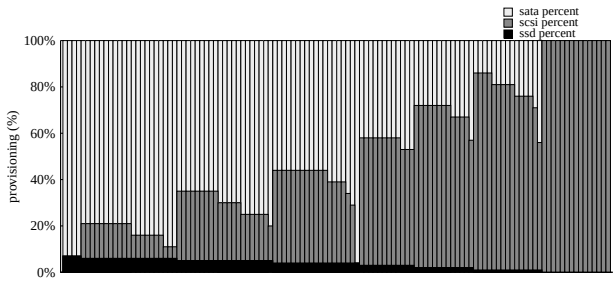
few disks as possible without causing performance to become worse than a 100% SCSI configuration (figure 5). SATA drives consume fewer watts per time unit; but, more importantly, they have the greatest capacity of our three drive types. This means that more volumes can be served from a single low-energy SATA drive than the other two drive types. In these provisionings, fewer SSDs are provisioned in most cases (figure 5a). Just enough SSDs are used to hold the most active volumes. Most of the rest of the volumes are served from SATA drives. The spikes of storage systems with more SCSI drives provisioned are those storage systems with that tend to have many large reads (see corresponding spikes in figures 5a and 5b). For these storage systems, SATA drives do not provide enough bandwidth, so more SCSI drives are provisioned.

The storage systems in our data set would receive a large decrease in energy consumption if they were provisioned with a combination of SSD, and SATA; many would use only 40% of the energy consumed by an all SCSI provisioning (figure 5c). This suggests that deploying SSDs into enterprise storage systems can affordably reduce energy consumption without sacrificing performance.

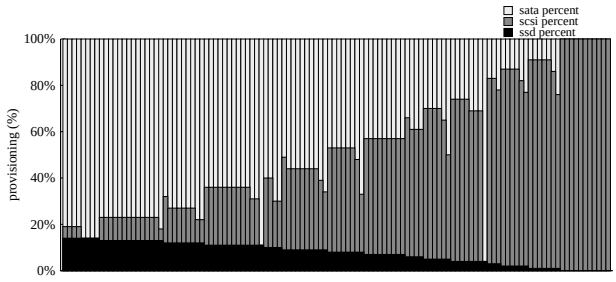
### 6.3 Provisioning as Flash Gets Cheaper

Looking forward, the cost of SSDs is expected to decrease as the technology becomes more mainstream, and manufacturing processes improve. As SSDs get closer to cost parity with SCSI drives, there will be less need to provision SCSI drives to hold busy volumes, since it will be affordable to place those volumes on SSDs instead. We consider now how the systems in our data set would be provisioned if the cost of SSDs decreases. Of course, the cost of rotating disks also decreases over time, but for our estimates we simply vary the cost of SSDs, since this models a decrease in the cost gap between SSDs and conventional rotating disks.

Figure 6 shows the same plot as figure 4a, but for an estimated SSD cost of $500 and $250, respectively. The general shape of the plots are very similar, with the percentage of volumes provisioned to SSDs increasing overall as SSD cost decreases. Also note that as the percentage of volumes on SSDs increases, the percentage of volumes on SCSI tends to decrease. This is




(a) SSD cost decreased by half




(b) SSD cost decreased by half again

**Figure 6.** Maximizing performance as SSDs become less expensive

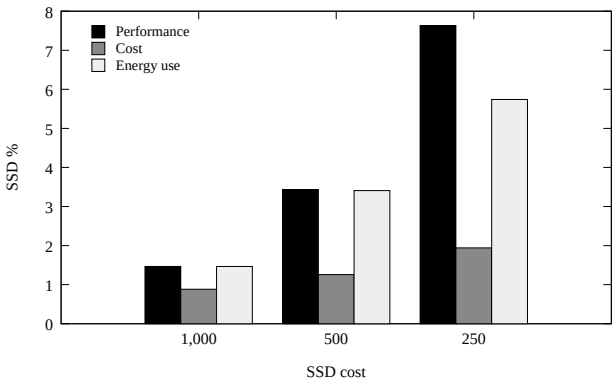



**Figure 7.** Average percentage provisioned to SSDs when optimizing for performance, cost, and energy use (respectively) as the cost of an SSD decreases

because volumes on SCSI drives can be split to either SSDs (if they are high-traffic) or to SATA (if they are low traffic). Overall, the trend is toward decreasing reliance on SCSI drives for high performance as SSDs grow cheaper, leading both to better performance and lower energy consumption.

Figure 7 summarizes the change in the average percentage of SSDs provisioned as the gap in cost between SSDs and conventional rotating disks increases. Three different SSDs costs are considered ($1000, $500, and $250). Each bar depicts the average percentage of volumes which are provisioned to SSDs across all systems in our data set. We consider each of our "best" provisionings, those which are optimized to either maximize performance, minimize cost, or minimize energy consumption.

Because of the significant performance difference between SSDs and rotating disks, the number of SSDs provisioned when maximizing system performance approximately doubles as the cost of SSDs decrease. This is because the configuration that maximizes performance always has the maximum number of volumes located on SSDs as possible given cost constraints. The number of SSDs provisioned also increases when minimizing energy use, although not as significantly as when maximizing performance. This is because SATA drives have lower energy consumption per unit than SSDs, so the number of SSDs provisioned depends on the number of volumes that cannot be located on SATA drives due to utilization constraints.

When maximizing for cost, on the other hand, the decrease in SSD cost has very little impact on the percentage of SSDs provisioned in each system. This is because only a small number of SSDs are needed in order to place the most active volumes on SSDs, leaving the remainder on low-energy SATA drives. We do not anticipate per-unit cost parity between SATA and SSDs in the foreseeable future, so it seems clear that even at their current prices, SSDs already provide a way to improve performance and reduce energy consumption without increasing price by placing the most active volumes on a small number of high-performance SSDs.

## 7. Related Work

There is a great deal of recent literature on flash drives. Writing is a more complicated operation than reading in flash drives, since it involves erasing the previous data as a preliminary step. In addition, writing causes serious media wear, therefore, writes have to be balanced across all device addresses. The issues involved in writing to flash drives are considered in [1, 10, 13, 17] among others. A comparison of SCSI and SATA drives appears in [4], it should be noted though, that the comparison predates the use of SATA drives in enterprise storage. Various applications which could profit from the enhanced performance of flash drives have been considered in [19–21, 23, 26], but, as pointed out in [25], they do not take price into consideration.

The configuration of storage systems with conventional disk drives only has been considered in the series of papers [3, 5–8, 11, 29]. This work uses traces as input data and is mostly concerned with the configuration of logical volumes to disks. the analysis is based on a mixture of modeling, extrapolation of device performance tables and bin packing heuristics. In contrast, we use the more common logical volume statistics, avoid the bin packing issues by assuming that data is striped across devices of the same type and use a thin queueing model which does not require traces.

A detailed analysis of performance and power in flash drives is given in [12]. It is shown that, depending on the specific workload (reads/writes, sequential/random), both the performance of power consumption of the flash drive may vary. A similar study of power consumption in disk drives [2] shows that the workload characteristics can also affect the power consumption of disk drives in ways which are similar to the way it affects power consumption in flash drives.

## 8. Future Work

An open question for flash drives concerns their reliability in enterprise storage settings. The limited number of writes tolerated by individual cells in flash drives is well-known, although sophisticated firmware already smoothes over this problem to a large degree. Recent analyses have carefully studied reliability of rotating disks and the surrounding storage infrastructure in enterprise settings using coarse-grained statistics [9, 16, 27], but more work is needed to understand the reliability of flash drives. As flash is adopted in the enterprise, more data will become available concerning its real-world reliability. These statistics could be included in our provisioning tool as a separate parameter or combined with the up-front cost of SSDs to improve enterprise storage planning.

Another opportunity to enhance our provisioning algorithm is to take into account the use of flash in the primary storage tier (cache) in addition to using flash as another type of secondary storage. Flash is slower than DRAM, but since it has no seek penalty there is an opportunity to deploy caching algorithms on flash storage. Doing so could significantly increase the size of the cache, thus improving its performance in some workloads. Logical volume counter statistics such as read and write hit ratios provide an opportunity to study the performance tradeoffs of using flash as cache and estimate the cost of deploying flash in the primary storage tier. However, large scale traces may be needed to accurately assess the full benefit of using flash in the cache. As explained in section 1, collecting large real-world traces from modern storage systems is difficult. If such traces become available, they could be combined with our existing approach to give a more complete picture of how to deploy flash in both the first and second tier of the storage system.

## 9. Conclusion

Enterprise storage systems require vast amounts of storage, need to meet high performance expectations, and consume a great deal of energy. The trend towards green computing and a renewed focus on the increasing cost of powering data centers has led to interest in reducing the energy consumption of enterprise storage arrays. Flash drives (SSDs) have recently been introduced as a new component in the enterprise, but little real-world analysis is available to understand how they can fit into the storage architecture.

Our study addresses this problem by analyzing data from 120 active enterprise storage systems to estimate the benefits and identify the trade-offs of provisioning flash drives. Instead of traces, which are often hard to collect and difficult to generalize, we use commonly-available storage counter statistics (e.g., read / write counts). We developed a novel provisioning algorithm that uses counter statistics to iteratively find provisionings which mix flash, SCSI, and SATA drives that improve performance, energy consumption, and cost as compared to the provisioning which includes only SCSI drives.

We found that in a majority of the systems we studied, provisioning a small number of flash drives along side conventional rotating disks can improve performance and lower energy consumption, without increasing the price, as compared to provisioning only SCSI drives. This suggests that many enterprises could easily “green” their storage systems while simultaneously improving performance by provisioning flash drives as part of their storage infrastructure.

## References

[1] N. Agrawal, V. Prabhakaran, T. Wobber, J.D. Davis, M. Manasse, and R. Panigrahy. Design tradeoffs for SSD performance. In USENIX Annual Technical Conference, pages 57–70, Boston, MA, June 2008.

[2] M. Allalouf, Y. Arbitman, M. Factor, R. I. Kat, K. Meth, and D. Naor, Storage modeling for power estimation, in proceedings of The Israeli Experimental Systems Conference (SYSTOR'09), May 4–6, 2009, Haifa, Israel

[3] G. Alvarez, E.Borowsky, S.Go, T. H. Romer, R. Becker- Szendy, R. Golding, A. Merchant, M. Spasojevic, A. Veitch, and J. Wilkes. Minerva: an automated resource provisioning tool for large-scale storage systems. ACM Transactions on Computer Systems, Vol. 19, 483–518, 2001.

[4] D. Anderson, J. Dykes, and E. 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.

[5] E. Anderson. Simple table-based modeling of storage devices. Technical Report HPL- SSP-2001-4, HP Laboratories, July 2001.

[6] E. Anderson, M. Hobbs, K. Keeton, S. Spence, M. Uysal, and A. Veitch, Hippodrome: Running rings around storage administration. In Proceedings of the USENIX Conference on File and Storage Technologies (FAST). USENIX, Monterey, CA, 175–188, 2002.

[7] E. Anderson, M. Kallahalla, S. Spence, R. Swaminathan, and Q. Wang. Ergastulum: an approach to solving the workload and device configuration problem. Technical Report HPL-SSP-2001-5, HP Laboratories, July 2001.

[8] E. Anderson, S. Spence, R. Swaminathan, M. Kallahalla, and Q. Wang. Quickly finding near-optimal storage designs. ACM Trans. Comput. Syst., 23(4): 337–374, 2005.

[9] L. Bairavasundaram, G. Goodson, B. Schroeder, A. Arpaci-Dusseau, R. Arpaci-Dusseau, An Analysis of Data Corruption in the Storage Stack. In Proc. of the 6th USENIX Conference on File and Storage Technologies (FAST), 2008.

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

[11] E. Borowsky, R. Golding, P. Jacobson, A. Merchant, L. Schreier, M. Spasojevic, and J. Wilkes. Capacity planning with phased workloads. In 1st Workshop on Software and Performance (WOSP98), pages 199–207, Santa Fe, NM, Oct 1998.

[12] F. Chen, D. Koufaty and X. Zhang Understanding intrinsic characteristics and system implications of flash memory based solid state drives, in Proceedings of SIGMETRICS 2009, 181–192, 2009.

[13] E. Gal and S. Toledo. Algorithms and data structures for flash memories. ACM Computing Surveys, 37(2): 138–163, 2005.

[14] J. Gray and B. Fitzgerald, Flash Disk Opportunity for Server Applications, ACM Queue, Vol. 6, Issue 4, 18–23, 2008.

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

[16] W. Jiang, C. Hu, Y. Zhou and A. Kanevsky, Are Disks the Dominant Contributor for Storage Failures? A Comprehensive Study of Storage Subsystem Failure Characteristics. In Proc. of the 6th USENIX Conference on File and Storage Technologies (FAST), 2008.

[17] H. Kim and S. Ahn, BPLRU: A buffer management scheme for improving random writes in flash storage. In Proc. of FAST08, article 16, 2008.

[18] T. Kgil and T.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.

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

[20] S. Lee and B. Moon. Design of flash-based DBMS: An in-page logging approach. In Proc. of SIGMOD07, 2007.

[21] S.W. Lee, B. Moon, C. Park, J. Kim, and S. 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.

[22] A. Leventhal. Flash storage memory. In Communications of the ACM, volume 51, July 2008.

[23] E. Miller, S. Brandt, and D. 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.

[24] M. Moshayedi and P. Wilkison, Enterprise Flash Storage, ACM Queue, Vol. 6, 32–39, 2008.

[25] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron. Migrating enterprise storage to SSDs: analysis of tradeoffs. In Proc. of EuroSys09, 2009.

[26] S. Nath and A. 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.

[27] E. Pinheiro, W. Weber and L. Barroso, Failure Trends in a Large Disk Drive Population. In Proc. of the 5th USENIX Conference on File and Storage Technologies (FAST), 2007.

[28] M. Wang, A. Ailamaki and C. Faloutsos, Capturing the Spatio-Temporal Behavior of Real Traffic Data, Performance 2002.

[29] J. Wilkes, Traveling to Rome: QoS specifications for automated storage system management. In Proceedings of the International Workshop on Quality of Service (Karlsruhe, Germany). Springer, Berlin, Germany, 75–91, 2001.