MixedStore: An Enterprise-scale Storage System Combining Solid-state and Hard Disk Drives *

Abstract

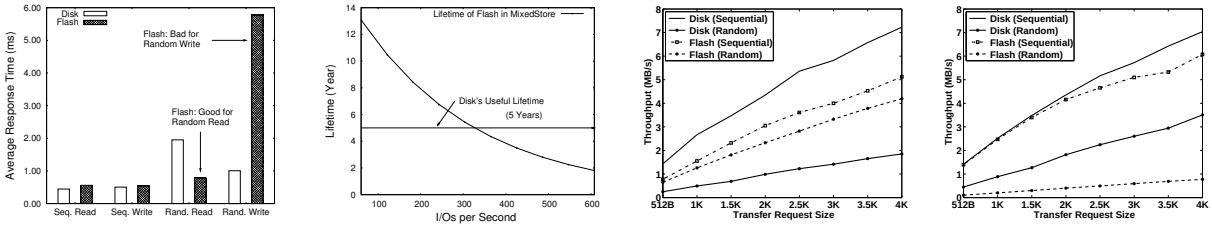
Flash memory overcomes some key shortcomings of HDDs including faster access to non-sequential data (when not degraded by garbage collection (GC) overheads) and lower power consumption. Given the complementary properties of HDDs and Solid State Disks (SSDs) in terms of cost, performance, and lifetime, the current consensus among several storage experts is to view SSD not as a replacement for HDD but rather as a complementary device within the storage hierarchy. Unlike the use of DRAM for caching/buffering, however, certain idiosyncrasies of flash make their integration into HDD-based systems non-trivial. Flash memory suffers from limits on its reliability, is an order of magnitude more expensive than the disk, and can be sometimes even slower than the HDD (due to excessive GC induced by high intensity of random writes). We design and evaluate a simplified hybrid system called *MixedStore* to provide: (a) improved capacity planning techniques to administrators of such hybrid systems with the overall goal of operating within *cost-budgets* and (b) improved performance/lifetime guarantees during episodes of deviations from expected workloads through innovative mechanisms such as *adaptive wear-leveling, write-regulation and fragmentation busting*. We implement a simulator for MixedStore and evaluate its efficacy using a variety of well-regarded enterprise-scale storage traces. As an illustrative example, MixedStore is able to reduce the average system response time by about 71% as compared to a HDD-based system for an enterprise scale random-write dominant Financial Trace [38]. A preliminary investigation of adaptive wear-leveling allows us to extend the useful lifetime of SSD by about 33% in the presence of unanticipated bursts in I/O, thus opening up new challenges in the design of efficient wear-leveling algorithms for the SSD controller.

1 Introduction

Hard disk drives (HDDs) have been the preferred media for data storage in enterprise-scale storage systems for several decades. The disk storage market totals approximately $34 billion annually and is continually on the rise [45]. Manufacturers of HDDs have been successful in ensuring sustained performance improvements while substantially bringing down the price-per-byte. As an example, during the past decade, the maximum internal data rate (IDR) for hard disks has witnessed a 20-fold increase resulting from improvements in rotational speeds (RPM) and storage densities; seek times have improved by a factor of 4 over the same period. However, there are several shortcomings inherent to HDDs that are becoming harder to overcome as we move into faster and denser design regimes. First, designers of HDDs are finding it increasingly difficult to further improve the RPM (and hence the IDR) due to problems of dealing with the resulting increase in power consumption and temperature [6, 16, 26]. Second, any further improvement in storage density—another way to increase the IDR—is increasingly harder to achieve and requires significant technological breakthroughs such as perpendicular recording [41, 32, 7]. Third, and perhaps most serious, despite a variety of techniques employing caching, pre-fetching, scheduling, write-buffering, and those based on improving parallelism via replication (e.g., RAID), the mechanical movement involved in the operation of HDDs can severely limit the performance that hard disk based systems are able to offer to workloads with significant randomness and/or lack of locality. Specific to our interest in this paper, in an enterprise-scale system, *consolidation* (e.g., as proposed/explored in [14]) can result in the multiplexing of unrelated workloads imparting/exaggerating the randomness and/or lack of locality in their aggregate [14, 15].

Alongside improvements in HDD technology, significant advances have also been made in various forms of solid-state memory such as NAND flash [2], magnetic RAM (MRAM) [39], phase-change memory (PRAM) [18], and Ferroelectric RAM (FRAM) [43]. Solid-state memory offers several advantages over hard disks: lower access latencies for random requests, lower power consumption, lack of noise, and higher robustness to vibrations and temperature. In particular, recent improvements in the design and performance of NAND flash memory (simply *flash* henceforth) have resulted in its becoming popular in many embedded and consumer devices. Small form-factor HDDs

*Name of storage system changed to ensure anonymity.



(a) Access Latency (b) Lifetime (c) Read Throughput (d) Write Throughput

Figure 1: A comparison of the performance and lifetime characteristics of representative SSD and HDD. although MTTFs for HDDs tend to be of the order of several decades, recent analysis has established that other factors (such as replacement with next, faster generation) implies a much shorter actual lifetime [42] and hence we assume a nominal lifetime of 5 years in the enterprise.

have already been replaced by flash in some consumer devices like music players, PDAs, digital cameras, etc. Flash has, however, only seen limited success in the enterprise-scale storage market [30]. Although (i) the aforementioned advances in flash technology and (ii) its dropping cost-per-byte [10] had led several storage experts to predict the inevitable demise of HDDs [11], flash has so far not been able to make inroads into the enterprise-scale storage market to the extent expected [30].

Solid-state Drives. Borrowing a few sentences from an excellent paper on this topic by Leventhal [30], *"The brunt of the effort to bring flash to primary storage has taken the form of solid-state disks (SSDs), flash memory packaged in hard-drive form factors and designed to supplant conventional drives. This technique is alluring because it requires no changes to software or other hardware components, but the cost of flash per gigabyte, while falling quickly, is still far more than hard drives. Only a small number of applications have performance needs that justify the expense"*. We will use the terms *SSD* and *flash* interchangeably in the rest of this paper. As evidence of this, major storage vendors producing flash-based large-scale storage systems such as RamSan-500 from Texas Memory Systems, Symmetrix DMX-4 from EMC, ioDrive from ioFusion, etc. are catering only a select class of applications such as large database servers rather than the general enterprise storage market.

Media	Access Time (μs)	Lifetime	Cost/GB ($)
DRAM	0.9	N/A	125
SSD	(45) Read, (200) Write	10K-1M Erase Cycles	25
HDD	5500	MTTF=1.2Mhr	3

Table 1: Performance, lifetime, cost comparison among different storage media. [30].

Table 1 presents a comparison of the performance, lifetime, and cost of representative HDDs, SSDs, and DRAM used in the enterprise. There are several important implications of how these properties of these devices (specifically, in this context, those of HDDs and SSDs) compare with each other. Flash technology possesses a number of idiosyncrasies that have hindered the SSD from replacing HDD in the general enterprise market. *First*, it is evident that there exists a huge gap between the Cost/GB of HDDs and SSDs. [1] *Second*, unlike HDD or DRAM, SSDs possess a huge asymmetry between the speeds at which reads and writes may be performed. As a result, the throughput a flash device offers a write-dominant workload is lower than for a read-dominant workload. *Third*, flash technology restricts the locations on which writes may be performed—a flash location must be *erased* before it can be written—leading to the need for a garbage collector (GC) for/within an SSD. We will elaborate on these properties of flash in Section 2. Certain workload characteristics (specifically, the presence of randomness), exacerbate GC overheads, thereby significantly slowing down the SSD—even to an extent where it operates slower than a HDD! [27]. *Finally*, to further complicate matters, unlike HDDs, SSDs have a life-time that is limited by the number of erases performed. Therefore, excessive writing to flash, while potentially useful for the overall performance of flash-based storage system, may limit its lifetime. This becomes an important concern in an enterprise-scale employing flash if its workload is write-intensive.

MixedStore: Motivation. From the above description, it should be clear that SSDs are fairly complex devices. Their peculiar properties related to cost, performance, and lifetime make it difficult for a storage system designer to neatly fit them between HDD and DRAM. To illustrate the complexity of the relationship between HDD and SSD, we present results from a simple experiment in Figure 1. As has been observed in other recent research, under certain workload conditions, an SSD can perform worse than the HDD [27]. A look at Figures 1(a),(c),(d) provides an illustration of such behavior and calls for careful de-

[1] A similar gap exists between SSD and DRAM. Furthermore, it is projected to worsen in the near future: up to a factor of 13 by 2010 [1]. This rules out major changes in the role played by DRAM in future systems that employ SSDs. DRAM will continue to retain both of its important roles related to caching and buffering. Therefore, we will not compare these two devices in the rest of this paper.

sign to gainfully utilize them in conjunction with HDDs in the enterprise. The degrading lifetime with increased write-intensity, as shown in Figure 1(b) [2] may result in premature replacement of these devices, adding to deployment/procurement/administrative costs. Finally, the low throughput offered by SSDs to random write-dominated workloads (Figure 1(d)), which are frequently encountered in enterprise-scale systems [27], necessitates intelligent partitioning of data in such hybrid environments while ensuring that the management costs do not overwhelm the performance improvements. As we shall see, unlike the HDD, flash-based devices require a longer history to be incorporated into a performance predictor. As a simple example, a large number of random writes may experience good response time but eventually the GC induced by the resulting fragmentation could result in requests coming much later to see degraded performance. Modeling these characteristics is an unexplored area and a significant part of our work as well as the foundation of the overall functioning of MixedStore.

Research Contributions. This paper makes the following specific contributions.

- We propose MixedStore, a simplified hybrid storage system containing only one each of an HDD and a SSD sharing the I/O bus. Besides this hardware, MixedStore comprises: (i) a *capacity planner* (*MixCP* henceforth) that makes long-term resource provisioning decisions for the expected workload; it is designed to optimize the cost of equipment that needs to be procured to meet desired performance and lifetime needs for the expected workload and (ii) a *dynamic controller* (*MixDC* henceforth) whose goal is to operate the system in desirable performance/lifetime regimes in the face of deviations at short time-scales in workload from those anticipated by MixCP.
- We develop simple statistical models that MixCP employs. These models are used in conjunction with MixedSim, [3] (a simulator we have developed for MixedStore by enhancing DiskSim [12]) to validate the efficacy of MixCP for a variety of well-regarded real-world storage traces. We expect MixCP to provide "rules-of-thumb" to administrators of hybrid storage systems when making provisioning decisions. As an illustrative result, MixCP is able reduce the cost of MixedStore by planning a well-provisioned system for a realistic random-write dominant workload (Financial Trace [38]) while ensuring similar performance as compared to an over-provisioned system.
- We implement MixDC in our simulator. In a MixedStore prototype, MixDC would have two components: (a) an enhanced block device driver that employs online statistical performance and lifetime models for SSD (and a performance model for HDD) to dynamically partition incoming workload among the SSD and HDD, and (b) two algorithms within the SSD controller (specifically, within the FTL layer) including reduction of fragmentation within the flash (fragmentation buster) and a novel concept of *adaptive wear-leveling*. As an illustrative result of our empirical evaluation of the efficacy of MixDC, it was able to prolong the life of flash device in MixedStore by about 33% in the face of an unexpected increase in I/O activity.
- Finally, we present preliminary ideas on how MixCP and MixDC could act in concert and present an initial validation of all components of MixedStore.

Road-map. The rest of this paper is organized as follows. In Section 2, we present the basics of flash memory technology and discuss relevant related work. Section 3 provides a bird's eye-view of the overall MixedStore architecture and how its two components, the Capacity Planner and the Dynamic Controller, interact. In Sections 4 and 5, we describe and evaluate these two components individually as well as when acting together. Finally, we present concluding remarks in Section 6.

2 Background and Related Work

2.1 Background on Flash

Basics of Flash Memory Technology. Flash is a unique storage device since unlike the HDD and volatile memories, which provide read and write operations, it also provides an *erase operation* [36]. Salient operational characteristics of these operations are as follows: Erase operations are performed at the granularity of a *block* which is composed of multiple *pages*. A page is the granularity at which reads and writes are performed. Each page on flash can be in one of three different states: (i) *valid*, (ii) *invalid* and (iii) *free/erased*. When no data has been written to a page, it is in the erased state. A write can be done only to an erased page, changing its state to valid. Out-of-place updates result in certain written pages whose entries are no longer valid. They are called invalid pages.

Erase operations (1.5ms) are significantly slower than reads/writes. Additionally, write latency can be higher than read latency by up to a factor of 4-5. The lifetime of flash memory is limited by the number of erase operations on its cells. Each memory cell typically has a lifetime of 10K-1M erase operations [9]. Thus, *wear-leveling* techniques [22, 24, 31] are used to delay the wear-out of the first flash block. The granularity at which wear-leveling is carried out impacts the variance in the lifetime of individual blocks and also the performance of flash. The finer the granularity, the smaller the variance in lifetime.

[2] We have picked a lifetime of 5 years for a HDD just for illustrative purposes. An excellent study of the useful lifetimes of disks based on data from real enterprise-scale systems appears in a paper by Schroeder and Gibson [42]

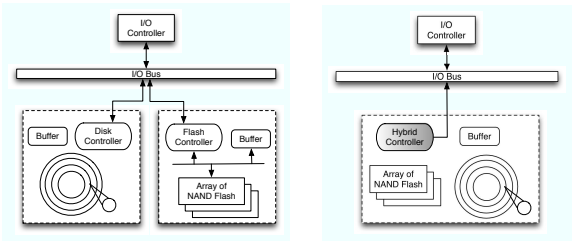
[3] Although our simulator is ready for sharing with other researchers, we are unable to provide its URL due to double-blinded review. The name of out simulator has been changed to preserve anonymity. However, reviewers interested in our code and data are welcome to approach us with the permission of the chairs.

The Flash Translation Layer (FTL). The FTL is a software layer which translates logical addresses from the file system into physical addresses between file system and physical flash memories. FTL helps in emulating flash as a normal block device by performing out-of-place updates which in turn helps to hide the erase operation in flash. The mapping tables and other data structures manipulated by the FTL are stored in a small, fast SRAM. These FTLs can be implemented at different granularities of how large an address space a single entry in the mapping table captures. Page-based FTLs map the logical page number of the request sent to the device from the upper layers such as file system to any physical page on flash. However, such translation requires a large mapping table to be stored in SRAM. At the other extreme, in a block-level FTL scheme, only the logical block number is translated into a physical block number whereas the logical page number offset within the block remains fixed. The size of the mapping table is reduced by a factor of *block size/page size* (128KB/2KB=64) as compared to page-level FTL. However, since a given logical page may now be placed in only a particular physical page within each block, the possibility of finding such a page decreases.

To address the shortcomings of the above two extreme mapping schemes, researchers have come up with a variety of alternatives. Although many schemes have been proposed [19, 8, 28, 23, 29], they share one fundamental design principle. They are a *hybrid* between page-level and block-level schemes. They logically partition their blocks into two groups - *Data Blocks* and *Log/Update Blocks*. Data blocks form the majority and are mapped using the block-level mapping scheme whereas the log blocks are mapped using a page-level mapping style. In related research , we have developed a novel page-based FTL scheme which utilizes temporal locality in workloads to overcome the shortcomings of the original page-based scheme by storing only a subset of mappings (likely to be accessed) on the limited SRAM and stores the remainder on the flash device itself. A paper describing this FTL is currently under review. We employ this FTL scheme in our current research.

2.2 Related Work

Flash as Cache and Write-Buffer. A lot of research has been conducted to improve performance of HDDs using non-volatile memory. eNVy [46] uses non-volatile memory for data storage wherein battery-backed SRAM is used to reduce the write overhead. HeRMES [35] uses magnetic RAM to reduce the overhead of frequently and randomly accessing meta-data. MEMS [44] has also been exploited to improve disk performance. Moreover, storage architecture in which flash memory is used as a conventional disk cache has already been proposed in [33] Our work goes beyond merely using flash as a cache/write-buffer—rather than treating flash as a *subordinate to the disk*, MixedStore views these as *complementary* storage media.



(a) MixedStore (b) Hybrid disk

Figure 2: Illustration of a MixedStore system and a hybrid disk.

Hybrid Disks. Samsung and Microsoft [40] have developed/deployed hybrid hard disks for laptops. Figure 2 shows two possible configurations of MixedStore. Although our focus in this paper is on the former configuration, we expect many ideas developed here to be of use in systems with hybrid disks. Booting time and resuming process from disk have been improved by overlapping the time for spinning up disk drive with the booting process from flash memory.

Bisson et al. [5] have explored the use of a flash-based NVRAM as a write buffer to reduce write latency of hard disks for desktop environments. They employ I/O redirection to reduce seeking overhead from disk by directing requests likely to incur long seeks to the on-disk NVRAM. We view the MixDP component of our system as conceptually close to Bisson et al.'s work and would be interested in comparing MixDP with their I/O redirection technique in the future. A key difference is that our flash model (developed in Section 4) additionally captures the fragmentation within flash (caused by random writes) and incorporates it into its redirection decision-making. This mechanism will be described in Section 5.

Flash-specific Improvements. Flash Translation Layer (FTL) is one of core-engines in a flash-based SSD. The state-of-the-art FTLs [8, 28, 23, 29] are based on log-buffer based approaches and optimize performance by trying to reduce costly GC overheads. Another orthogonal approach of exposing flash based devices to the file system has been proposed. JFFS2 [21] and YAFFS2 [47] are the most popular file systems optimized for flash memories. Kim et al. [25] have developed a flash device buffer management scheme to reduce fragmentation caused by random writes. Moreover, different SSD designs including interleaving requests to obtain parallelism and ganging etc. have been proposed to improve flash device performance [37].

Flash in the Enterprise. Sun Micro-systems has proposed a storage architecture incorporating flash-based SSDs as intent-log devices and read caches providing improved performance along with reduced power consumption [30]. They propose to use their ZFS file system as an interface to these SSDs. *We view Sun's proposed hybrid architecture as the closest in essence to MixedStore and believe that the models and techniques developed here are worth imple-*

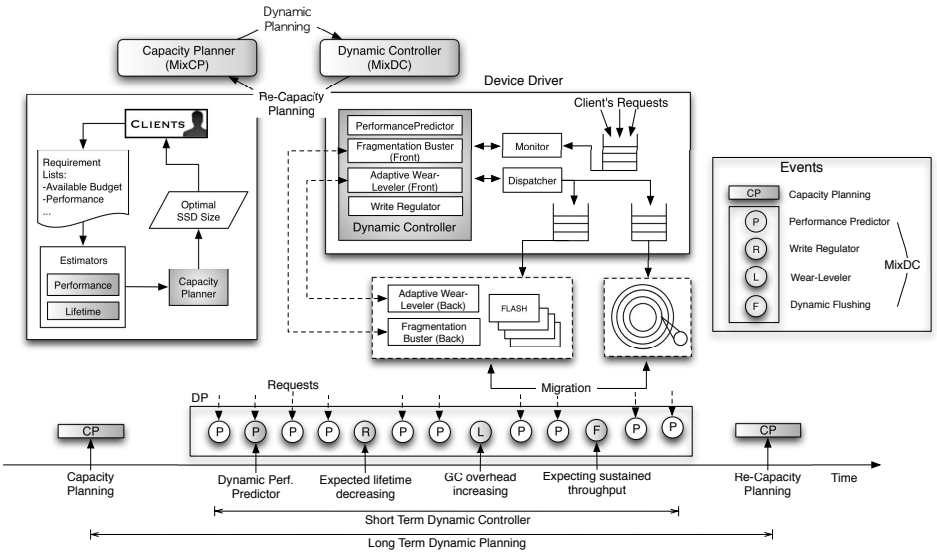


Figure 3: Depiction of various components of MixedStore and how they interact.

menting and evaluating in the context of their system.

Finally, to enhance flash-based DBMS, Lee et al., [27] proposed an in-page logging approach to reduce random write overhead by updating in-place in the database buffer and hence reducing garbage collection overhead. A key contribution in this paper is the observation that workloads with extensive randomness can cause an SSD to perform worse than a HDD. We find similar results in our evaluation and build models that can capture this aspect of an SSD's operation.

3 Overview of HybridStore

Figure 3 depicts the interaction between various components of MixedStore. In this study, we utilize a simplified model of an enterprise-scale storage system consisting of a single HDD and a single SSD connected to the same I/O bus. We will deal with more complex configurations consisting of RAID arrays etc. in the future work.

MixedStore consists of two major components: (i) a long-term resource provisioning tool (MixCP) for system administrators to optimize the procurement/deployment/maintenance costs while adhering to the performance budgets (specific to workloads) and lifetime budgets, and (ii) a short term dynamic controller (MixDC) which is part of the MixedStore internal structure.

MixCP utilizes statistical models for performance and lifetime to determine the optimal SSD capacity (we assume a static HDD size for our study) for different workloads. Figure 4 demonstrates the improvement in performance and reduction in cost using MixCP along with dynamism aware performance predictor. However, workloads are known to exhibit deviations from predicted behavior [] and hence we require MixDC to dynamically control the partitioning of

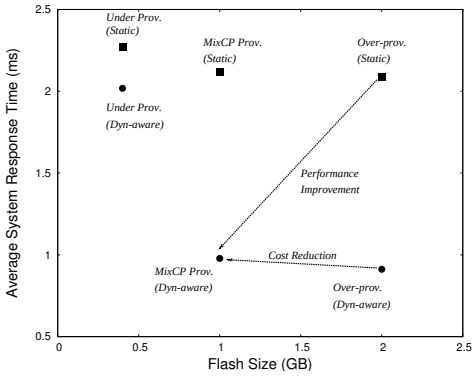


Figure 4: Capacity Planning for Financial Trace [38]. "Static" denotes a static data-partitioning policy where write requests larger than 4KB are assumed to be sequential and are serviced by the HDD and others are serviced by SSD. "Dyn. aware" denotes an intelligent data partitioning policy (described in Section 4 developed by us.

requests (using a performance predictor) as well as ensure high level of performance from the SSD (performance of SSD degrades with fragmentation of data on the device). Thus, MixDC is implemented at the I/O driver level and needs to interact with the SSD controller to perform *adaptive wear-leveling* and *fragmentation busting*.In the real prototype, these techniques will be implemented as split modules (similar to back-end and front-end drivers in Xen [4]) with the controller residing withing MixDC while the working thread(module) present in the SSD. Analogous to adaptive wear-leveling whose primary purpose is to improve SSD's lifetime, MixDC contains a write-regulator which can reduce the intensity of requests to SSD during periods

Workloads	Average Request Size (KB)	Read (%)	Sequentiality (%)	Average Request Inter-arrival Time (ms)
Financial (OLTP) [38]	4.38	9.0	2.0	133.50
Cello99 [17]	5.03	35.0	1.0	41.01
TPC-H (OLAP) [48]	12.82	95.0	18.0	155.56

Table 2: Enterprise-Scale Workload Characteristics.

of high I/O activity. We describe the functionality and preliminary evaluations of all these mechanisms in Section 5.

For evaluation purposes, we have developed our own simulator called *MixedSim* (name changed to ensure anonymity). The simulator has been built by enhancing Disksim [13], a well-regarded HDD simulator. MixedSim is designed with a modular architecture with the capability to model a holistic flash-based storage environment. It is able to simulate different storage sub-system components including device drivers, controllers, caches, flash devices, and various interconnects. In our integrated simulator, we add the basic infrastructure required for implementing the internal operations (page read, page write, block erase etc.) of a SSD. The core FTL engine is implemented to provide virtual-to-physical address s translations along with a garbage collection mechanism. We use a novel page-based FTL scheme for our evaluation (paper describing this scheme is under review).

We use realistic enterprise scale workloads (refer to Table 2 for their description) and some synthetic traces for experimentation. We simulate IBM Ultrastar 36Z15 as the HDD and a 32GB 2.5" SATA Solid State Drive from Super-Talent [3] in MixedSim. (Note that the specification available for SSDs are insufficient to completely model them. For example, the SRAM cache size and the FTL is unknown. Thus, we make suitable assumptions for these parameters.)

4 Capacity Planning

Given the large price gap between SSDs and HDDs, it is useful to be able to determine appropriate capacities of these devices for the workload the system expects to support. We define this process of determining the right size of devices in MixedStore as *capacity planning*.

As illustrated in Figure 4, both under-provisioning and over-provisioning of flash memory leads to inefficient storage utilization, thus adversely impacting the cost-to-benefit ratio for MixedStore. Therefore, the goal of capacity planning is to minimize this discrepancy so that overall storage investment cost can be optimized.

4.1 Problem Formulation

The objective of capacity planning is to minimize the cost of MixedStore (deployment, management, maintenance etc.) while meeting the service level agreements. These constraints can vary from guaranteeing some minimum performance requirements to reducing management and re-deployment costs, ensuring system reliability etc. For the purpose of our study, we try and minimize the deployment cost (in terms of $/GB) subject to a combination of both performance and re-deployment constraints. We use average system response time as a metric of MixedStore's performance and term this metric as the system's *Performance Budget*. As described in Section 2, the blocks in SSDs become unreliable beyond 10K-1M erase cycles. This poses a significant challenge for a system administrator whose objective is to keep system re-deployment frequency and costs under control. We capture these objectives in terms of a *Lifetime Budget* for the system, which is the time between successive capacity planning decisions and equipment procurement/installation.

We formulate our capacity planning problem as a means of minimizing the cost of acquiring/installing MixedStore while meeting the administrator/workload-specified performance (P_{Budget}) and useful lifetime budget (L_{Budget}). Let C_{SSD} indicate the cost of flash based SSDs and and C_{HDD} indicate the cost of HDDs in MixedStore. Then the total MixedStore cost $C_{MixedStore}$ is the sum of these individual costs.

Equation 1 shows the formal description of capacity planning.

$$\text{Minimize } C_{MixedStore} \text{ Subject to } \begin{cases} P_{MixedStore} \geq P_{Budget} \\ L_{MixedStore} \geq L_{Budget} \end{cases} \quad (1)$$

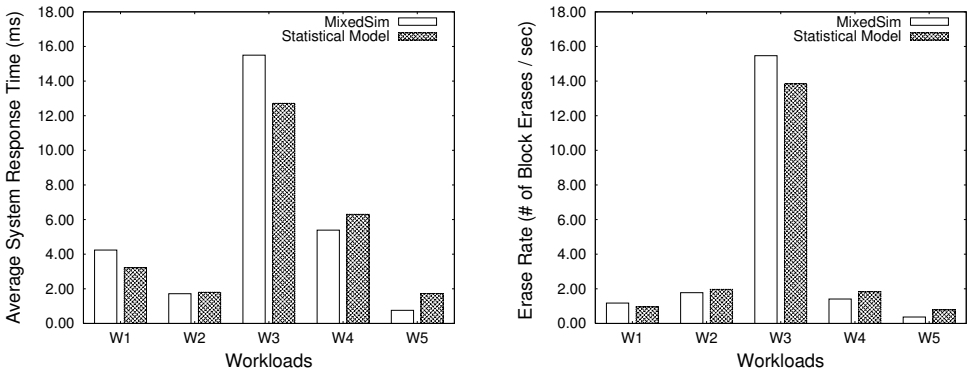
$$\text{Where } C_{MixedStore} = C_{SSD} + C_{HDD}$$

It is easily seen that the above optimization problem reduces to minimizing the cost of SSD for fixed size of HDD available in a MixedStore system.

However, the performance and lifetime of flash based SSD is highly dependent on not only the workload characteristics but also the internal intricacies of flash such as design of FTL, efficiency of GC etc. This provides a mandate for the design of a robust capacity planner (MixCP) tool for use by storage system designers. In the next sub-sections, we describe the statistical models utilized by MixCP to provision SSDs in MixedStore.

4.2 Modeling Performance and Lifetime of Flash Memory for MixCP

We employ a "black-box" modeling approach for estimating a given SSD's useful lifetime and performance. Our model makes no assumptions about the inner configurations (such as FTL employed, SRAM cache size etc.). We do find its



(a) Performance Model (b) Lifetime Model

Figure 5: Validation of performance and lifetime models compared with values measured using MixedSim.

efficacy varies depending on the internals of the SSD. For example, the predictor performs better with our page-based like FTL than other state-of-the-art hybrid FTLs. We do not elaborate on these here due to space constraints. For this purpose, we need to identify statistically significant workload characteristics that impact the SSD's lifetime and performance. Performance is directly impacted by data fragmentation caused by random writes which invoke costly GC operations. Moreover, high write intensity increases the number of erase operations required to reclaim invalid space on flash, thus reducing lifetime of blocks. Based on these observations, we consider the following workload characteristics as significant independent variables: (i) average read/write ratio, (ii) spatial locality captured in the form of average sequentiality among requests, (iii) average request inter-arrival time, (iv) average request size, and (v) flash utilization defined as the ratio of the working set size to the total flash size. We install probes in MixedSim to capture data relating to the above parameters.

4.3 Methodology: Regression Based Modeling

Using multiple linear regression, we first find significant predictor variables which affect the variables being predicted: (i) average system response time (ms) for performance budget, (ii) average block erase rate (erases/second) for lifetime budget. We start with the general approach in multiple regression of finding significant predictor variables while plugging in as many predictor variables as we can think of. In order to avoid multicollinearity problems, we also perform correlation analysis on predictor variables to ensure that they are all independent variables.

Performance Model. We use average I/O system response time (R_{avg}) as a predictor of flash performance. I/O system response time represents the time interval between the issuance of request to the SSD by the I/O driver and its completion notification to the driver. It includes queuing delay, bus delay and controller overhead in the device. We first experiment with a multiple linear regression based model. Upon finding this model unsatisfactory, we move towards a slightly more complicated multiple log-linear model [20]. It can be represented as

$$\log(R_{avg}) = a_0 + \sum_{i=1}^{n} a_i \cdot W_{avg}(i) + \epsilon \tag{2}$$

where (W_{avg}) represents the the average of a particular workload characteristic selected from a set of n parameters discussed earlier (Section 4.2) and ϵ is a small error. The coefficients (a_0, a_1, ... , a_n) are estimated during the learning phase of the experiments.

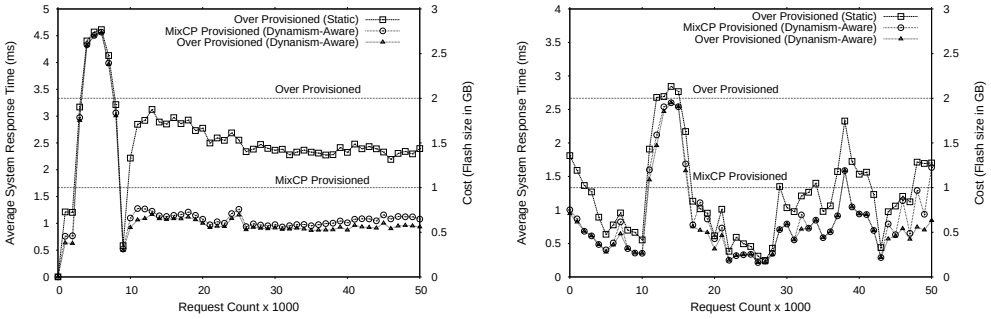
Lifetime Model. Erase rate (block erases per second) denoted by E_{avg}, represents the lifetime of a flash device since each block typically has a life of about 10K-1M erase cycles [9]. As in the case of performance modeling, we start by fitting a multiple linear regression model. Again, we observe that a multiple log-linear regression technique, similar to the one used for performance budget is able to model the lifetime budget. The similarity between the two models arises from the fact that higher response times are a function of garbage collection which require block erases and hence impact lifetime. Thus, the lifetime model can be represented as

$$\log(E_{avg}) = b_0 + \sum_{i=1}^{n} b_i \cdot W_{avg}(i) + \epsilon \tag{3}$$

where (W_{avg}) represents the the average of a particular workload characteristic selected from a set of n parameters discussed earlier (Section 4.2) and ϵ is a small error. The coefficients (b_0, b_1, ... , b_n) are estimated during the learning phase of the experiments.

4.4 Modeling Results and Validation

In this sub-section, we describe the experimental results with our modeling methodology. Then, we validate our



(a) Financial Trace (b) TPC-H

Figure 6: Capacity Planning

Index	Sequentiality (Ratio)	Request Size (Sectors)	Utilization (Ratio)	Inter-Arrival (ms)
W1	0.10	41.54	0.89	322.18
W2	0.70	16.90	0.89	79.90
W3	0.30	115.71	0.94	80.24
W4	0.70	115.44	0.58	319.74
W5	0.03	6.57	0.91	164.49

Table 3: Some of the synthetic write-only workloads (W1,W2,W3,W4) used to train the performance and lifetime models and a realistic Financial Trace workload [38](W5) used for evaluating the models.

models by comparing against the actual values measured using MixedSim. We generate a large number of synthetic traces by varying workload characteristics described in Section 4.2 to train the models and randomly select 900 of these traces to form our training set. The adjusted R-square [4] is found to be around 90% for both the multiple log-linear models [20]. The average error rate is about 25% for the training set.

Validation. We validate our performance and lifetime models by comparing their results with the corresponding values measured using MixedSim. Table 3 shows the salient characteristics of some of the synthetic and real workloads. We choose write-only synthetic traces for validation since flash performs very well for read dominant workloads. Moreover, lifetime is not an issue for such workloads since they encounter very few erase operations. For W2, the error in the performance model is only about 4% whereas it rises to about 21% for W3 which has the highest erase rate and response time values (owing to large request sizes and low inter-arrival times) in the traces shown. For the Financial trace [38], the observed performance as well as lifetime errors are about 55%. The major cause of this discrepancy is that our black-box model assumes no information about the internal state of the flash and hence is liable to errors. Arguably, by incorporating more information about flash internals we can improve our model further. However, as explained in Section 3, for MixedStore, having a reasonably accurate MixCP suffices so long as MixDC can handle the inaccuracies in the former models. To summarize our validation, we have demonstrated the possibility of developing a performance and lifetime estimation methodology with reasonable accuracy with simple linear regression models.

[4]Adjusted R-square defines the proportion of variability that is accounted for by a statistical model. Unlike R-square it only increases if a newly added predictor, statistically improves an existing model

4.5 Evaluation

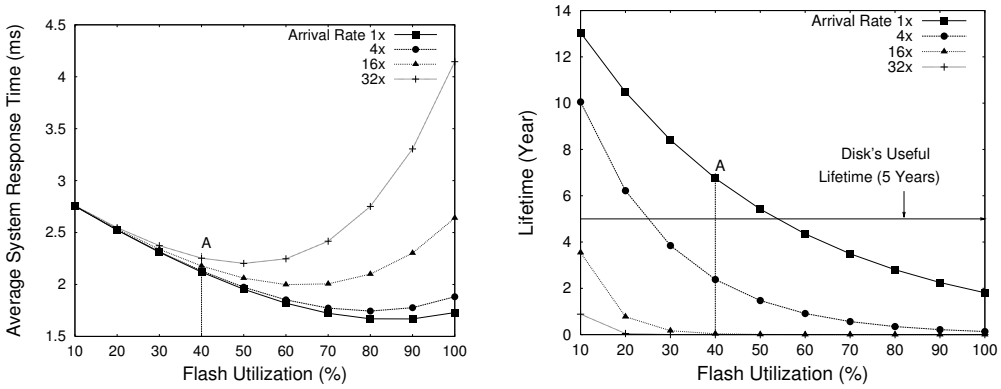
Workload	Lifetime (Yr)		
	Over Provisioned (2GB)	MixCP (1GB)	Under Provisioned (0.5GB)
Financial Trace	52.69	7.29	2.67
TPC-H	97.50	21.65	-

Table 4: Lifetime observations with different approaches. A block is assumed to possess 10K reliable erase cycles.

In this subsection we compare the performance of MixCP capacity planner with other generic capacity planning methodologies which either under-provision or over-provision the flash capacity in MixedStore. For evaluation purposes we use realistic enterprise scale workloads whose salient characteristics are shown in Table 2.

Table 4 shows the flash device lifetime for various capacity planning techniques with dynamism-aware data partitioning policy for different workloads. TPC-H is read dominant and hence performance budget is of greater concern than lifetime. For Financial trace which is a write-dominant workload, we observe that under-provisioning capacity would necessitate flash device replacement within 3 years and hence would impact the overall lifetime budget of MixedStore. We want the flash device to last till around the useful life of disk (approximately 5 years) and both over-provisioning and MixCP are able to achieve this mandate.

Over-provisioning flash capacity should reduce the request response times from flash device since the garbage collection overheads will be reduced and hence improve



(a) Overall Performance of MixedStore (b) Lifetime of flash in MixedStore

Figure 7: Capacity planning for financial-like trace [38]. We increase arrival rate by orders of 2 as shown in legends.

MixedStore performance as compared to MixCP. We observe that the performance benefits accrued with this extra flash are much less as compared to the increased cost due to larger flash. As shown in Figure 7(b), for read-dominant TPC-H [48], both MixCP and over-provisioned models provide similar performance. This can be directly attributed to the fact that read-oriented workloads have very small amount of writes, thus the garbage collector is invoked very infrequently and the service patterns remain similar for both the capacity planning methodologies. Even for the write-dominant Financial trace, Figure 7(a) shows that MixCP performs much better than a static over-provisioning scheme. However, if dynamism-aware data partitioner is utilized along with an over-provisioned flash, we observe a slight improvement in performance as compared to MixCP.

But this small improvement comes at additional costs of bigger flash memory. Thus, the cost-to-benefit ratio advocates the use of MixCP for capacity planning in enterprise scale systems.

4.6 Challenges in Capacity Planning

Workloads are known to exhibit variation from their predicted behavior. In such circumstances, capacity planning alone is not sufficient to meet the lifetime and performance budgets. Figure 7(a)-(b) show the impact of increased arrival rate on performance and lifetime budgets for a write-dominant workload. If the system designer had provisioned the system at point A to keep the flash lifetime around a disk's useful life while satisfying the performance needs, these guarantees do not hold if the workload changes. With higher intensity of writes, the garbage collector is invoked more often; thus degrading the system's performance. Moreover, it results in higher number of block erases, reducing the flash lifetime. Thus, we require additional sophisticated data partitioning mechanisms which can dynamically adapt to these changing workload environments. In the next section, we describe some techniques employed by our dynamic controller (MixDC) to meet the various budgets and thus work in synchronization with MixCP.

5 Dynamic Controller- MixDC

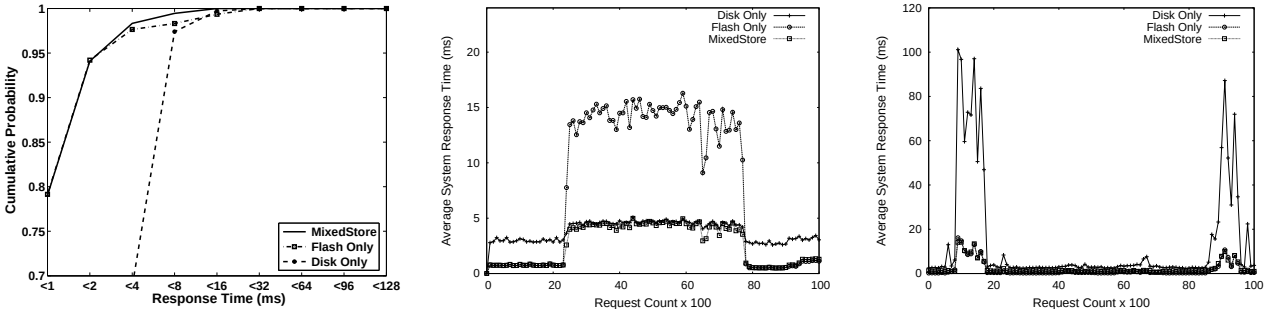
In this section, we investigate a variety of techniques which help operate MixedStore within or close to desirable performance, cost, lifetime budgets despite unanticipated changes in workloads.

5.1 Short-Term Performance Prediction Model for SSD

The performance of the SSD is highly dependent on the workload incident on it. Since out-of-place updates are performed on the flash, GC resulting from fragmentation has an important impact on response time. We build upon our learning from capacity planning and try to develop time-scale performance models suitable for MixDP.

Although the large-body of work on modeling disk performance is of use here, there are certain salient novel aspects of flash operation that MixDP's SSD model must capture. Perhaps the most important such feature is that unlike a disk, *an SSD performance model needs to incorporate a much* **longer history**, since a large enough number of random writes (that might themselves experience good performance) might cause fragmentation over time and the resulting GC invocation would then degrade the performance of requests that arrive much later the.

Again we start with identifying the crucial workload characteristics which play a major role. However, contrary to the earlier MixCP performance model here we work with a sliding window of requests. This sliding window acts as a short term history of requests and enable us to make fair short term decisions. The main workload characteristics used in the model are: (i) *Average Read to write ratio* of a window of requests, (ii) *Spatial locality*—average sequentiality of a window of requests, (iii) *Request inter-arrival time*, and (iv) *Current request size*.



(a) CDF (Financial) (b) Time series (Financial) (b) Time series (Cello99)

Figure 8: Performance of MixedStore compared with a disk-only and a SDD-only system.

Since this performance model needs to make predictions about the performance of requests in the immediate future, and as seen how performance depends on long-term history, we need to capture and preserve certain aspects of the *current state* of the flash device. However, this information about state of the flash device might require information about SSD internals that may not be feasible (e.g., in the SSD that MixedStore assumes).

In order to build a feasible as well as efficient black-box performance model, we use the history of previous device service times as an indicator of flash device state. For simplicity, we use the average of the service times (S_{avg}). Moreover, we use system response time ($R_{current}$) as a measure of flash device performance. Thus, our multiple linear regression model can be represented as

$$R_{current} = c_0 + c_1 W_{window} + c_2 S_{avg} + \epsilon$$

$$S_{avg} = \big(\sum_{j=1}^{w} S(j)\big)/w$$

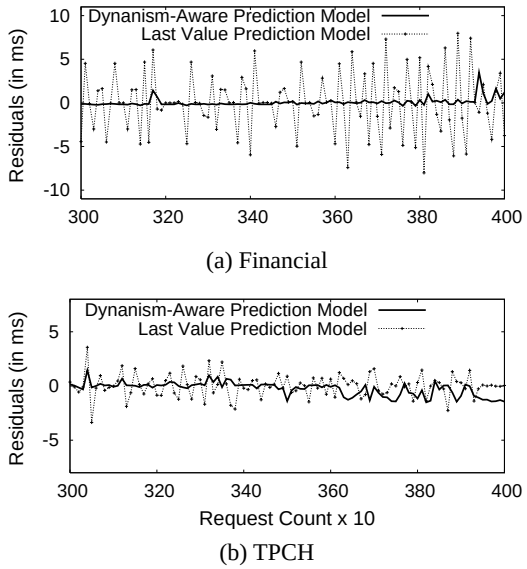
where, ϵ is a small error and

W_{window} is the workload during window$w$

The coefficients (c_0, c_1, c_2) are estimated during a learning/training phase of our experiment which consists of half of the workload. We believe converting our learning-based prediction technique can be easily adapted to operate online, although we do not evaluate that here.

5.2 Evaluation with Dynamism-Aware Performance Prediction Model

We use the Financial trace [38] and TPC-H [48] workload to validate our model. Contrary to our performance predictor for MixCP, our empirical evaluation suggests a simpler multiple linear regression to be satisfactory. For Financial trace, we observe the measured R-square value to be 98%. We compare the accuracy of our model with a simple baseline—a *last value-based* prediction model for SSD which uses the last service time value as its prediction. Figure 9 demonstrates the superior prediction quality of our model for both TPC-H and Financial trace. Our model is able to predict the state of the flash better than the last value predictor and hence shows much small error rate.

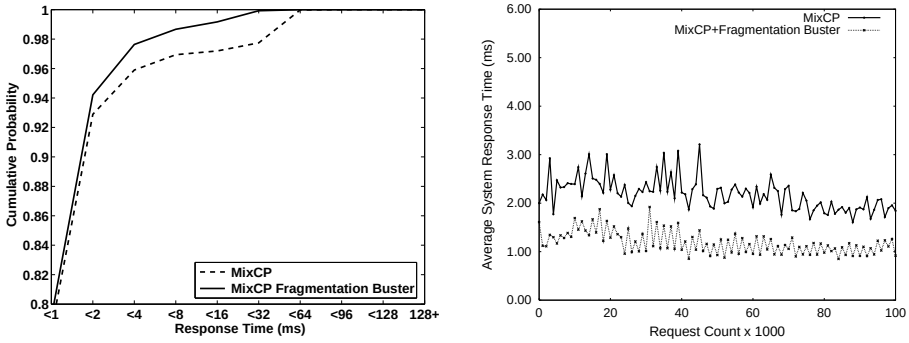


(a) Financial

(b) TPCH

Figure 9: Comparison of our dynamic SSD performance prediction model with a simple last value-based prediction model.

We integrate our SSD prediction model with an admittedly simple disk performance predictor. We use a model based on the average response time observed during the training phase to predict disk performance. The dynamic controller (MixDC) partitions write requests depending on the least response times predicted by the SSD and HDD models. MixDC maintains a table to store information about the current location of data (device id) and updates it whenever some data is migrated from one device to the other. The clean up of dirty data (old version) is scheduled during idle periods. Read requests are always serviced from the device which contains the data. Frequently read small sized data is migrated to the flash in the background during idle periods. We refer the reader to research on analyzing and predicting periods of idleness in storage workloads (specifically, findings of heavy-tailed inter-arrival times in enterprise-scale



(a) (b)

Figure 10: (a) Performance improvement of MixDC with fragmentation buster. (b) Sustained improved performance (consistently reduced response times) obtained using fragmentation buster.

workloads that imply the presence of significant idle periods along with those of intense activity) and do not incorporate a specific policy into our current MixedStore design. Figure 8(a) illustrates the performance of MixedStore incorporating the prediction models in MixDC with respect to a disk-only and flash-only system for the random write dominant Financial trace. Although flash is good for servicing most requests, but it shows extremely high response time for some requests (due to extensive garbage collection) which should to be serviced by the disk. MixedStore is able to reduce the average system response time by about 71% as compared to a disk-only system. The zoomed-in requests in Figure 8(b) reflect the performance of the prediction model as it is able to accurately distinguish the requests which could be serviced faster by flash as compared to disk. We believe that with a more sophisticated disk performance prediction model we can further improve the performance of MixDP and this is part of our future work.

5.3 Fragmentation Busting

We observe in Figure 10(b) that the response times SSD spike up in the region between requests 2000 to 8000. The major reason for this sudden poor performance is the high intensity of (mostly random write) requests that induces GC. Prior to this sudden burst we see a long activity of small random requests being quickly serviced from the SSD. This results in developing fragmentation on the device and hence requiring costly merge operations [25]. In order to prevent such fragmented zones on flash, we try to develop a flushing methodology called *Fragmentation Busting*.

Workloads are known to exhibit periods of idleness between bursts of requests [34]. Lot of research has gone into developing techniques to identify and utilize these idle periods. Specifically, Mi et al. [34] categorized workloads based on idle periods into tail-based, body-based and body+tail based. We utilize their work to schedule flushing of fragmented data from flash to the disk. The reader should note that this is unlikely to be gainfully implemented using a black-box approach. It requires co-operation from the device since the effective mapping tables are present within the device and are not exposed to outer systems. Thus, only a part of the flushing mechanism, specifically the scheduler can be implemented with MixDC. In order to decide which data needs to be flushed, the device controller needs to pin the pages causing this fragmentation. We maintain a LRU (Least Recently Used) list of the valid pages using the logical page number of the requests. This represents the cold data on flash and its migration to disk does not have any major impact on MixedStore's performance. When the idle period kicks in, the fragmentation buster directs the flash controller to start flushing the data fragments. A small DRAM-based buffer needs to be maintained so that any request to the data being migrated can be serviced. Since we flush mostly cold data, such requests are rare. Moreover, since this activity can be delayed until an idle period is available, in this work we consider it a pure background activity that does not interfere with the real workload and hence we ignore its possible degrading effects on overall performance.

In order to simulate fragmentation busting, we use an offline profiling approach in which we are able to identify the idle periods in the workloads and hence schedule flushing then. As shown in Figure 10(a), about 2% more requests get serviced with fragmentation buster as compared to a MixCP only dynamic controller. Although the gap in average values doesn't seem much, but fragmentation buster's real role is in *reducing the tail of the CDF*, i.e., reducing the number of requests that experience extremely poor performance. Figure 10(b) illustrates how fragmentation buster helps in smoothening the spikes in response times.

5.4 Write Regulation

We pointed out to one of the challenges in capacity planning as the unpredictability in workloads. In this section, we develop techniques for handling sudden unanticipated bursts in requests.

A prolonged and/or recurring period of unanticipated random writes detrimental impact on lifetime of flash We experiment with a simple write regulator that detects increased

Technique	Average Erase Rate (Erases/Sec.)	Ratio of Requests Serviced by Flash (Flash/(Flash+Disk))	Average System Response Time (ms)
MixCP	0.16	0.69	1.15
MixCP (Red_{25})	0.12	0.54	1.56
MixCP (Red_{50})	0.09	0.40	1.98

Table 5: Evaluation of Write Regulation.

I/O activity and consistently monitors the expected flash life through the lifetime model of MixCP. When violations are detected it starts to regulate the writes being sent to flash by over-riding the decisions made by the performance model in MixDC.

We experiment with two models of a static write rate regulator that pick 25% or 50% (uniformly at random) of the requests being sent to flash and redirects them to HDD during periods of higher-than-expected I/O intensity. Let us call these policies Red_{25} *and* Red_{50}*, respectively.* For this experiment, we synthesize a workload with similar characteristics as Financial Trace but with periods of reduced inter-arrival time between requests. Table 5 shows that we are able to reduce the flash block erase rate by about 25% while reducing the requests being serviced by flash by about 21% using Red_{25}. An additional 19% reduction in the erase rate is observed using Red_{50}. However, it results in an increase of 0.83ms in average system response time. Thus, the rate of write regulation must be chosen judiciously so as to meet the performance budget while ensuring that lifetime guarantees are satisfied.

5.5 Adaptive Wear-Leveling

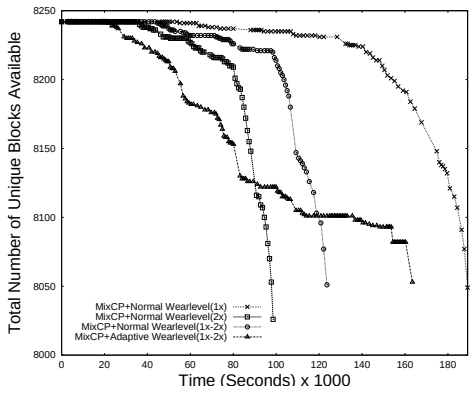


Figure 11: Adaptive Wear-Leveling

Wear-leveling (described in Section 3 requires swapping of data between blocks which have high erase count with blocks which have relatively lower erase count. This swapping operation results in additional erase operations which reduce the lifetime of blocks These extra erases start to play a significant role towards the end of a flash device's life and indeed accelerate its death.

We propose an *adaptive wear-leveling mechanism*—a novel idea to the best of our knowledge—which like the write regulator monitors the erase rate of blocks and during periods of prolonged unanticipated write activity, coordinates with the flash controller to prevent the extra erases caused by wear-leveling by temporarily halting the leveling algorithm. Once normal I/O activity starts, it allows the device to revert to its wear-leveling mechanism.

Figure 11 shows the impact of our adaptive wear-leveler on the financial trace with modified inter-arrival times to resemble a workload with periods of unpredicted high I/O activity. The SSD is assumed to be in a representative state with regions containing highly accessed data (hot regions) and others with infrequently accessed data (cold data) as found in prior experiments. This results in wear-leveler being called to uniformly maintain the erase level of all blocks. However, with MixDC equipped with our adaptive wear-leveler we observe an improved lifetime of about 33%, delaying the need for replacement and reducing re-deployment costs. This enables MixDC to achieve the lifetime guarantees as projected by MixCP; hence both our capacity planning and dynamic-controller tools act in tandem to achieve the lifetime and performance budgetary requirements imposed on them.

6 Concluding Remarks

This research was based on the emerging consensus among several storage experts that in the foreseeable future, with the exception of certain specialized domains, SSDs should be used as a complementary device to HDDs in enterprise-scale storage hierarchy. We attempted to address two problems in a simplified version of such a hybrid system consisting of one HDD and one SDD sharing the I/O bus. First, we developed an online capacity planner called *MixCP* that used statistical models to meet the performance and lifetime requirements of SSDs and HDDs to provide storage administrators with guidelines on provisioning such a system in a cost-effective manner. Second, we developed a dynamic controller, *MixDC*, that used shorter time-scale SDD and HDD models along with regulation of write rate to the SDD and a novel idea of adaptive wear-leveling within the SDD to operate the storage system within regions of desirable cost, performance, and lifetime budgets. We evaluated these systems using a simulator (MixedSim) developed by us using a variety of well-regarded benchmarks. We found that MixedStore is able to reduce the average system response time by about 71% as compared to a HDD-based system for a enterprise-scale Financial trace. Moreover, our innovative adaptive wear-leveling mechanism was able to prolong the life of SSDs by about 33% in the presence of unanticipated increase in I/O intensity. In essence, our research opened up new vistas for not only designing a well-provisioned enterprise-scale storage system consisting of SSDs and HDDs but also established a need for relooking at the design of SSDs to incorporate some innovative mechanisms such as fragmentation buster and adaptive wear-leveler.

References

[1] Flash vs DRAM Price Projections . http://www.storagesearch.com/ssd-ram-flash%20pricing.html.

[2] Samsung 256GB Flash SSD With High-Speed Interface . http://www.i4u.com/article17560.html.

[3] 2.5" Super-Talent SATA Solid State Drive. http://www.supertalent.com/products/ssd-commercial.php?type=SATA.

[4] BARHAM, P., DRAGOVIC, B., FRASER, K., HAND, S., HARRIS, T., HO, A., NEUGEBAUER, R., PRATT, I., AND WARFIELD, A. Xen and the art of virtualization . In *Proceedings of the nineteenth ACM symposium on Operating systems principles* (2003), pp. 164 – 177.

[5] BISSON, T., AND BRANDT, S. A. Reducing Hybrid Disk Write Latency with Flash-Backed I/O Requests. In *International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)* (2007), IEEE Computer Society.

[6] CHARRAP, S., LU, P., AND HE, Y. Thermal Stability of Recorded Information at High Densities. *IEEE Transactions on Magnetics 33*, 1 (January 1997), 978–983.

[7] CHEN, J., AND MOON, J. Detection Signal-to-Noise Ratio versus Bit Cell Aspect Ratio at High Areal Densities. *IEEE Transactions on Magnetics 37*, 3 (May 2001), 1157–1167.

[8] CHUNG, T., PARK, D., PARK, S., LEE, D., LEE, S., AND SONG, H. System Software for Flash Memory: A Survey. In *Proceedings of the International Conference on Embedded and Ubiquitous Computing* (August 2006), pp. 394–404.

[9] E. GAL AND S. TOLEDO. Algorithms and Data Structures for Flash Memories. *ACM Computing Survey 37*, 2 (2005), 138–163.

[10] Flash Price Drop Spurs Innovation, Feb 2008. http://www.washingtonpost.com/wp-dyn/content/article/2008/02/01/AR2008020101313.html.

[11] Can flash memory become the foundation for a new tier in the storage hierarchy?, Sept 2008. http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=547.

[12] GANGER, G., WORTHINGTON, B., AND PATT, Y. *The DiskSim Simulation Environment Version 2.0 Reference Manual*. http://www.ece.cmu.edu/~ganger/disksim/.

[13] GANGER, G., WORTHINGTON, B., AND PATT, Y. *The DiskSim Simulation Environment Version 3.0 Reference Manual*.

[14] GOYAL, P., MODHA, D. S., AND TEWARI, R. CacheCOW: Providing QoS for Storage System Caches. *ACM SIGMETRICS Performance Evaluation Review 31*, 1 (June 2003), 306 – 307.

[15] GULATI, A., MERCHANT, A., AND VARMAN, P. J. pClock: An Arrival Curve based Approach for QoS Guarantees in Shared Storage Systems. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems* (June 2007), pp. 13–24.

[16] GURUMURTHI, S., SIVASUBRAMANIAM, A., AND NATARAJAN, V. Disk Drive Roadmap from the Thermal Perspective: A Case for Dynamic Thermal Management. In *Proceedings of the International Symposium on Computer Architecture (ISCA)* (June 2005), pp. 38–49.

[17] HP Labs. Tools and Traces. http://tesla.hpl.hp.com/public_software/.

[18] Intel, STMicroelectronics Deliver Industry's First Phase Change Memory Prototypes. http://www.intel.com/pressroom/archive/releases/20080206corp.htm.

[19] J. KIM, J.M. KIM, S.H. NOH, S. MIN, AND Y. CHO. A Space-Efficient Flash Translation Layer for Compactflash Systems. *IEEE Transactions on Consumer Electronics 48*, 2 (2002), 366–375.

[20] JAIN, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley-Interscience, 1991.

[21] JFFS2: The Journalling Flash FileSystem. http://sources.redhat.com/jffs2/jffs2.pdf.

[22] JUNG, D., CHAE, Y., JO, H., KIM, J., AND LEE, J. A Group-based Wear-Leveling Algorithm for Large-Capacity Flash Memory Storage Systems. In *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)* (September 2007), pp. 160–164.

[23] KANG, J., JO, H., KIM, J., AND LEE, J. A Superblock-based Flash Translation Layer for NAND Flash Memory. In *Proceedings of the International Conference on Embedded Software (EMSOFT)* (October 2006), pp. 161–170.

[24] KAWAGUCHI, A., NISHIOKA, S., AND MOTODA, H. A Flash-Memory based File System. In *Proceedings of the Winter 1995 USENIX Technical Conference* (1995), pp. 155–164.

[25] KIM, H., AND AHN, S. BPLRU: A Buffer Management Scheme for Improving Random Writes in Flash Storage. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)* (Feburary 2008), pp. 1–14.

[26] KIM, Y., GURUMURTHI, S., AND SIVASUBRAMANIAM, A. Understanding the Performance-Temperature Interactions in Disk I/O of Server Workloads. In *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA)* (Febuary 2006).

[27] LEE, S., AND MOON, B. Design of Flash-based DBMS: An In-Page Logging Approach. In *Proceedings of the International Conference on Management of Data (SIGMOD)* (August 2007), pp. 55–66.

[28] LEE, S., PARK, D., CHUNG, T., LEE, D., PARK, S., AND SONG, H. A Log Buffer based Flash Translation Layer Using Fully Associative Sector Translation. *IEEE Transactions on Embedded Computing Systems 6*, 3 (2007), 18.

[29] LEE, S., SHIN, D., KIM, Y., AND KIM, J. LAST: Locality-Aware Sector Translation for NAND Flash Memory-Based Storage Systems. In *Proceedings of the International Workshop on Storage and I/O Virtualization, Performance, Energy, Evaluation and Dependability (SPEED2008)* (Feburary 2008).

[30] LEVENTHAL, A. Flash Storage Memory. *Communications of the ACM 51*, 7 (2008), 47–51.

[31] LOFGREN, K. M. J., NORMAN, R. D., THELIN, G. B., AND GUPTA, A. Wear Leveling Techniques for Flash EEPROM. In *United States Patent, No 6,850,443* (2005).

[32] MALLARY, M., TORABI, A., AND BENAKLI, M. One Terabit Per Square Inch Perpendicular Recording Conceptual Design. *IEEE Transactions on Magnetics 38*, 4 (July 2002), 1719–1724.

[33] MARSH, B., DOUGLIS, F., AND KRISHNAN, P. Flash memory file caching for mobile computers. In *To appare in Proceedings of the 27th Hawaii Conference on Systems Science* (1994).

[34] MI, N., RISKA, A., SMIRNI, E., AND RIEDEL, E. Enhancing Data Availability through Background Activities. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)* (2008), pp. 492–501.

[35] MILLER, E., BRANDT, S., AND LONG, D. HeRMES: High-Performance Reliable MRAM-Enabled Storage. In *HotOS* (2001), pp. 95–99.

[36] NIIJIMA, H. Design of a Solid-State File Using Flash EEPROM. *IBM Journal of Research and Developement 39*, 5 (1995), 531–545.

[37] NITIN, A., VIJAYAN, P., AND TED, W. Design Tradeoffs for SSD Performance. In *Proceedings of the USENIX Annual Technical Conference* (June 2008).

[38] OLTP Trace from UMass Trace Repository. http://traces.cs.umass.edu/index.php/Storage/Storage.

[39] S. TEHRANI AND J. SLAUGHTER AND E. CHEN AND M. DURLAM AND J. SHI AND M. DEHERREN. Progress and Outlook for MRAM Technology. *IEEE Transactions on Magnetics 35*, 5 (September 1999), 2814–2819.

[40] Samsung Hybrid Hard Drive. http://www.samsung.com/Products/Semiconductor/Support/ebrochure/hddodd/hybrid_hard_drive_datasheet_200606.pdf.

[41] SCHIRLE, N., AND LIEU, D. History and Trends in the Development of Motorized Spindles for Hard Disk Drives. *IEEE Transactions on Magnetics 32*, 3 (May 1996), 1703–1708.

[42] SCHROEDER, B., AND GIBSON, G. A. Understanding disk failure rates: What does an MTTF of 1,000,000 hours mean to you? In *Proceedings of the Annual Conference on File and Storage Technology (FAST)* (Feburary 2007).

[43] SHIMADA, Y. FeRAM: Next Generation Challenges and Future Directions. *Electronics Weekly* (May 2008).

[44] UYSAL, M., MERCHANT, A., AND ALVAREZ, G. A. Using MEMS-Based Storage in Disk Arrays. In *FAST '03: Proceedings of the 2nd USENIX Conference on File and Storage Technologies* (Berkeley, CA, USA, 2003), USENIX Association, pp. 89–101.

[45] White Paper: Datacenter SSDs: Solid Footing for Growth. http://www.samsung.com/global/business/semiconductor/products/flash/FlashApplicationNote.html.

[46] WU, M., AND ZWAENEPOEL, W. eNVy: a Non-Volatile Main Memory Storage System. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems* (1994), pp. 86–97.

[47] YAFFS: Yet Another Flash File System. http://www.aleph1.co.uk/yaffs.

[48] ZHANG, J., SIVASUBRAMANIAM, A., FRANKE, H., GAUTAM, N., ZHANG, Y., AND NAGAR, S. Synthesizing Representative I/O Workloads for TPC-H. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)* (2004).