19th Annual IEEE International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems

HybridStore: A Cost-Efficient, High-Performance Storage System Combining SSDs and HDDs

Youngjae Kim†, Aayush Gupta‡, Bhuvan Urgaonkar‡, Piotr Berman‡, and Anand Sivasubramaniam‡
†Oak Ridge National Laboratory, ‡Pennsylvania State University
kimy1@ornl.gov, {axg354, bhuvan, berman, anand}@cse.psu.edu

***Abstract*—Unlike the use of DRAM for caching or buffering, certain idiosyncrasies of NAND Flash-based solid-state drives (SSDs) make their integration into existing systems non-trivial. Flash memory suffers from limits on its reliability, is an order of magnitude more expensive than the magnetic hard disk drives (HDDs), and can sometimes be as slow as the HDD (due to excessive garbage collection (GC) induced by high intensity of random writes). Given these trade-offs between HDDs and SSDs in terms of cost, performance, and lifetime, the current consensus among several storage experts is to view SSDs not as a replacement for HDD but rather as a complementary device within the high-performance storage hierarchy. We design and evaluate such a hybrid system called *HybridStore* to provide: (a) HybridPlan: improved capacity planning technique to administrators with the overall goal of operating within *cost-budgets* and (b) HybridDyn: improved performance/lifetime guarantees during episodes of deviations from expected workloads through two novel mechanisms: *write-regulation and fragmentation busting*. As an illustrative example of HybridStore's efficacy, HybridPlan is able to find the most cost-effective storage configuration for a large scale workload of Microsoft Research and suggest one MLC SSD with ten 7.2K RPM HDDs instead of fourteen 7.2K RPM HDDs only. HybridDyn is able to reduce the average response time for an enterprise scale random-write dominant workload by about 71% as compared to a HDD-based system.**

I. Introduction

Hard disk drives (HDDs) have been the preferred media for data storage in high-performance and enterprise-scale storage systems for several decades. However, there are several shortcomings inherent to HDDs. First, designers of HDDs are finding it increasingly difficult to further improve the RPM due to problems of dealing with the resulting increase in power consumption and temperature [6]. 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 [16]. Third, 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.

Alongside improvements in HDD technology, significant advances have also been made in various forms of solid-state memory such as NAND flash, STT-RAM, phase-change memory (PCM), and Ferroelectric RAM (FRAM). 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.

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 I: Performance, lifetime and cost comparison [13].

Table I presents a comparison of the performance, lifetime, and cost of representative DRAM, SSD, and HDD. There are several important implications of how these properties compare with each other. First, it is evident that there exists a huge gap between the Cost/GB of HDDs and SSDs. Second, unlike HDD, SSDs possess an asymmetry between the speeds at which reads and writes may be performed. As a result, the throughput a SSD offers for 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. Certain workload characteristics (in particular, the presence of randomness) increase the fragmentation of data stored in flash memory, i.e., logically consecutive sectors become spread over physically non-consecutive blocks on flash. This exacerbates GC overheads, thereby significantly slowing down the SSD [12]. Furthermore, this slowdown is non-trivial to anticipate. A given set of random writes may themselves experience good throughput, but increase fragmentation, thereby degrading the performance of requests (read or write) arriving much later in future. Finally, to further complicate matters, unlike HDDs, SSDs have a lifetime that is limited by the number of erases performed.

From the above description, it should be clear that SSDs are fairly complex devices [1]. 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. As has been observed in other recent research, under certain workload conditions, an SSD can perform worse than the HDD [12] and in certain SSDs, read throughput can be slower than write throughput for small random workload patterns [2]. The SSD's lifetime limit calls for careful design to gainfully utilize them in conjunction with HDDs in the enterprise. The

1526-7539 2011

DOI 10.1109/MASCOTS.2011.64

degrading lifetime with increased write-intensity may result in premature replacement of these devices, adding to deployment, procurement, and administrative costs. Finally, the low throughput offered by SSDs to random write-dominated workloads, which are frequently encountered in enterprise-scale systems [12], necessitates intelligent partitioning of data in such hybrid environments while ensuring that the management costs do not overwhelm the performance improvements.

This paper makes the following specific contributions.

- We propose a hybrid system containing HDDs and SSDs, called *HybridStore* that exploits the complementary properties of these two media to provide improved performance and service differentiation under a certain cost budget. Besides this hardware architecture, HybridStore comprises: (i) a *capacity planner* (*HybridPlan* 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 workload, and (ii) a *dynamic controller* (*HybridDyn* henceforth) whose goal is to operate the system in desirable performance/lifetime regimes in the face of deviations at short time-scales in workload.
- We develop models that HybridPlan employs to find the most economical storage configuration given devices and workloads using Mixed Integer Linear Programming (ILP). We expect HybridPlan to provide "rules-of-thumb" to administrators of hybrid storage systems when making provisioning decisions.
- We implement a simulator for HybridSim by combining FlashSim [11] with Disksim [3]. An implementation of HybridDyn 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) called *fragmentation buster* and *write regulator*.

The rest of this paper is organized as follows. Section II provides a bird's eye-view of the overall HybridStore architecture and how its two components, HybridPlan and HybridDyn, interact and discuss relevant related work. In Section III and IV, we describe the capacity planner and dynamic controller for HybridStore. Then we extensively evaluate HybridPlan and HybridDyn in Section V and VI respectively. Finally, we present concluding remarks in Section VII.

II. Overview of HybridStore

Figure 1 depicts the interaction between various components of HybridStore. Besides the storage hardware (HDDs, SSDs, and I/O buses) shown in the figure, HybridStore consists of two major software components. The first of these is a long-term resource provisioner called HybridPlan. We envision HybridPlan to be a tool that would enable storage administrators to provision both kinds of devices in cost-effective ways. The decision-making of HybridPlan would occur at coarse time-scales (months to years) corresponding to when procurement and deployment decisions are made. HybridPlan employs a ILP solver engine based on mathematical formulations to make its provisioning decisions. HybridPlan is intended to cost-effectively provision devices to allow HybridStore to (i) adhere to the performance needs of hosted workloads and (ii) meet useful lifetime requirements specified by the administrator, under these workload assumptions.

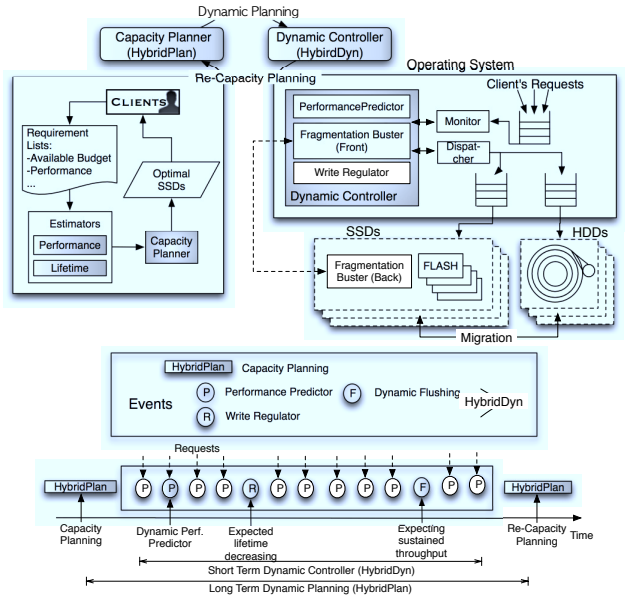


Fig. 1: Depiction of various components of HybridStore and how they interact.

The second component of HybridStore is a dynamic controller (HybridDyn) that operates at significantly finer time-scales (milliseconds to hours). HybridDyn employs statistical models for performance of SSD and HDD to make dynamic request partitioning decisions—these decisions are made at request-level granularity (milliseconds to seconds). Additionally, it employs novel techniques for data management within the SSD (write regulation, and fragmentation busting) that operate at the granularity of several minutes to hours. Intuitively, the components of HybridDyn operate collectively to take corrective data management decisions in HybridStore to adhere to desired performance and lifetime needs despite (i) provisioning errors made by HybridPlan and (ii) deviations in workload characteristics and device behavior.

A. Related Work

In a recent work from Microsoft Research, Narayanan et al. [15] examined the role of SSDs in enterprise storage systems using a number of real data center traces available to them. Their work explores the cost-benefit trade-offs of various SSD and HDD configurations flash and disk capacities/configurations for these real traces. However, there are several key differences between our contributions. First, our work, in particular HybridPlan, is much more general and can be used to target any type of devices including STT-RAM and

PCM. In this work, we focus only on flash since it is the only mature technology with concrete and meaningful numbers for cost and performance. Second, we have developed a data classification strategy which can be used to decide partitioning of workloads amongst the chosen devices. Third, while they admit that flash wear-out needs to be considered while using it as a write buffer, they do not explore any specific ways of doing this. We incorporate this in the form of lifetime budgets in HybridPlan and our dynamic workload partitioning (HybridDyn) employs a variety of techniques to adhere to these budgets. Finally, our study goes beyond capacity planning—HybridDyn employs a combination of model-driven as well as reactive techniques to operate our hybrid system under given performance/lifetime budgets despite varying workloads. Closest to our work is a recent paper by Guerra et al. [4] and we consider it highly complementary with similar results and insights. There are differences in our performance modeling approaches. Additionally, we consider lifetime constraints and include power costs in our formulation.

III. HybridPlan: Capacity Planning

A. Problem Formulation: Objective and Constraints

For the purpose of our study, we try and minimize the deployment and operation cost (in terms of $) subject to a combination of both performance and re-deployment constraints (due to lifetime of flash memory). We use IOPS as a metric of HybridStore's performance and term this metric as the system's *Performance Budget*. In addition, we need to consider lifetime issues in the flash because 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* (years) 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 HybridStore while meeting the workload-specified performance ($Perf_{Budget}$) and useful lifetime budget ($Life_{Budget}$). $Cost_{Installation}$ indicates the installing cost of devices. Apart from these, costs associated with power consumption, thermal consumption (cooling), other maintenance and management activity form the recurring costs denoted by $Cost_{Recurring}$. However, information in the academic domain about the management/maintenance costs of these devices (- HDDs and SSDs) is still sparse and inconclusive. Furthermore, management costs vary with legal contracts and are highly subjective. Hence, we only consider electricity cost of operation due to power consumption as recurring cost in this study. In sum, the total HybridStore cost is the sum of these individual costs ($Cost_{HybridStore}$ = $Cost_{Installation}$ + $Cost_{Recurring}$).

B. Workload Requirements and Device Characteristics

We can extract workload requirements (space or bandwidth requirement) by analyzing their IO traces.

1) Data Classification: We describe the data classification methodology used to partition a workload into smaller subsets. A workload can be characterized on the basis of certain features such as total size, read/write ratio, request arrival rate etc. Furthermore, each workload is a collection of sub-workloads which exhibit similar features. Each of these sub-workloads are called as *classes*. Classes help in determining commonality within workload streams and are essential for accurately mapping workloads to devices for an effective capacity planning framework.

2) Finding workload attributes: The entire logical address space of the workload is divided into fixed-size chunks, then mapped to different classes. These fixed-size chunks are called as *records*. We use 1MB as record size roughly corresponds to the granularity of data prefetching done by HDDs/SSDs. As described above, we need to find the attributes for describing workloads. We capture temporal locality in workloads using the total number of accesses to the records. We use average read volume to describe the read/write ratio of each record. Similarly, we use the median of request sizes to ascribe the request size to each record. This parameter captures the spatial locality in workloads. The reason for using median instead of average request size is because our experimental evaluation showed that median proved to be a better metric as it negated the impact of outliers (very small or very big requests) and helped in distributing records across classes appropriately. Lastly, we use the total number of IOs in a record in the workload as a measure of the number of IO arrivals to the record over the entire life of the workload. Note that there may be other attributes which can be used for data classification. However, our empirical analysis binds these parameters to be effective in partitioning workloads across classes.

3) Hierarchical data classification: Now that we have defined the parameters for characterizing workloads, we develop a mechanism to segregate records across classes. Temporal locality of a class is defined using hot/cold regions. The records which are accessed at least once in the workload are considered *hot* whereas records which are never accessed are treated as *cold* records. Classes are further divided based on request sizes. Records with request size less than 16KB are part of highly-random request classes whereas records with request sizes greater than 64KB are part of highly-sequential data classes. We also have intermediate data classes depending on whether request sizes are greater than 32KB (partially sequential) or not (partially random). We use the lower(25th), middle(50th) and upper(75th) quartiles of the overall distribution of total IOs across the records to further segregate these records. The readers should note that all the data points for creating classes as described above are based on empirical evaluation as well as qualitative intuition. In this study we have considered 33 data classes. The number of data classes can be further optimized using merging and reduction techniques and is part of our future work.

The device characteristics can be obtained not only from their data sheets but also from performance tests.

Variable	Description
General Variable	
$K_\$$	Electricity Cost ($/KWH)
T	Total trace time
$LIFE$	Lifetime: The threshold (in years) for which provisioning is being done
Device	
i (i=1, 2, 3, ..., I)	Device Type
C_i	Capacity of device of type i
U_i	Utilization of device of type i
RB_i	Read bandwidth of device of type i
WB_i	Write bandwidth of device of type i
IT_i	Initiation time of device of type i i.e the time for initiating each IO (1/IOPS)
P_i	Average Power consumption of device of type i
L_i	Lifetime of device of type i
$D\$_i$	Cost of device of type i
$E-UNIT_i$	Block size of a device of type i (only for SSDs)
$W-UNIT_i$	Size of each write on a device of type i
Data Class	
j (j=1, 2, 3, ..., J)	Data Class
S_j	Volume of data class j in terms of KB of records
IO_j	Total IO count of data class j
R_j	Read volume of data class j (in KB)
W_j	Write volume of data class j (in KB)
Decision Variable	
x_{ij}	Data of class j on y_i devices of type i in KB
y_i	The number of devices of type i

TABLE II: Declaration of Variables.

C. HybridPlan Formulation

We formulate our provisioning problem as a Mixed Integer Program. We describe a tool which finds the most cost-effective storage configuration using available devices for the provisioned workloads by reducing our optimization problem to a Mixed ILP problem. Table II shows declaration of each variable for problem formulation of HybridPlan.

As described earlier, we consider installation cost and electricity cost for the total cost of the storage systems. Given the properties of I different types of devices, the overall installation cost of storage systems is highly dependent on the numbers of each device type $i \in I$, and its individual device cost is $D\$_i$: $Cost_{Installation} = \sum_{i=1}^{I} D\$_i \times y_i$

Given the electricity cost per time (= $K_\$$) and the power consumption of device type i (= $P(i)$), the energy consumption of overall storage system (= E) over time followed by the overall electricity cost of operation by the energy consumption can be calculated as: $E_{Operation} = \sum_{i=1}^{I} y_i \times \int_t P_i dt$, $Cost_{Recurring} = K_\$ \times E_{Operation}$

Putting these together, we get the dollar cost of installing storage system and its operation. The objective function to minimize is: $Cost_{HybridStore} = Cost_{Installation} + Cost_{Recurring} = \sum_{i=1}^{I} D\$_i \times y_i + (K_\$ \times \sum_{i=1}^{I} y_i \times \int_t P_i dt)$

The constraints as shown in Table III, are related to (i) data groups, (ii) devices capacity, (iii) devices bandwidth, and (iv) life- time of the SSD.

- Equation 1 is the capacity constraint for the data classes and states that the sum of all the data belonging to class j partitioned across all I devices should be the same as the size of data class.
- Equation 2 is the capacity constraint for devices and states that the sum of data belonging to J classes on devices of type i should be less than the effective capacity of all the y_i devices.
- Equation 3 is the performance constraint for devices and states that y_i devices of type i should be capable of handling the performance needs of J data classes placed on these devices. $Diff_{ij}$ refers to a difficulty factor which essentially computes the read bandwidth, write bandwidth and IOPS needs for data class j.
- Equation 4 is the lifetime constraint for devices and states that each device of type i should last at least the specified LIFE for which provisioning is being undertaken. Generally, storage re-provisioning is carried out every 3-5 years in a typical data center. HDDs are known to have more lifetime than this specified value and hence, this constraint typically degenerates into provisioning for SSDs which have limited erase cycles.
- Equation 5 specifies this lifetime constraint for SSDs. $Wear_{ij}$ represents the wear-out factor for SSDs i.e. the erase rate of blocks on a SSD. It is a function of the rate at which writes are done on a SSD and the amount of free space (pages) available after each erase. Amount of free space reclaimed depends on the amount of fragmentation prevalent on a SSD and α is used to capture this phenomenon. In the worst case, each block erase can result in only 1 free page whereas in the best case, we can reclaim all pages in a block. Thus the value of α varies from ($W-UNIT_i$/$E-UNIT_i$) to 1.

$$\sum_i x_{ij} = S_j \ (\forall\ j \in J) \quad (1)$$

$$\sum_j x_{ij} \leq (U_i \times C_i) \times y_i \ (\forall\ i \in I) \quad (2)$$

$$\sum_j Diff_{ij} \times x_{ij} \leq y_i \ (\forall\ i \in I),$$

$$\text{where } Diff_{ij} = \frac{(IT_i \times IO_j + \frac{R_j}{RB_i} + \frac{W_j}{WB_i})}{(S_j \times T)} \quad (3)$$

$$L_i \geq LIFE \ (\forall\ i \in I) \quad (4)$$

$$\sum_j (Wear_{ij} \times LIFE \times x_{ij}) \leq L_i \times y_i \ (\forall\ i \in SSD_I),$$

$$\text{where } Wear_{ij} = \frac{(W_j/S_j)}{(\alpha \times E-UNIT_i)/T} \quad (5)$$

TABLE III: Constraints of optimization formulation. Each equation in the above constraints illustrates different constraints: The declaration of variables used in the equations are described in Table II.

IV. HybridDyn: Dynamic Controller

HybridDyn consists of several components; a performance prediction module (the core of HybridDyn), Fragmentation Buster and Write Regulator.

A. 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 short time-scale performance models suitable for the HybridDyn. Although the large-body of work on modelling disk performance is of use, there are certain salient novel aspects of flash operation that HybridDyn'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.

Again we start with identifying the crucial workload characteristics which play a major role. We used 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*. 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 HybridStore assumes).

We developed a performance prediction model for SSDs; we use the history of previous device service times of SSDs 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_{curr}) as a measure of flash device performance. Thus, our multiple linear regression model can be represented as

$$R_{curr} = c_0 + c_1 \cdot W_w + c_2 \cdot S_{avg} + \epsilon, S_{avg} = \frac{\left(\sum_{j=1}^{w} S(j)\right)}{w} \quad (6)$$

where ϵ is a small error and W_w 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.

B. Fragmentation Busting

Small random writes increase data fragmentation on flash, thus exacerbating garbage collection overhead. We demonstrate this impact in Figure 2 by alternating sequential and small random write requests for synthetic workloads using FlashSim [11]. Both "A" and "C" are regions with sequential write activity. However, the presence of random writes in region "B" leads to data fragmentation on flash, thus increasing the average response time for requests in "C". In order to prevent such fragmented zones on flash, we develop a flushing methodology called *Fragmentation Busting*. As shown in Figure 2, flushing some portion of these small random writes to disk (periodically moving 25% of random writes for this experiment), we can reduce the variation in response times and improve the performance.

Flushing 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 HybridDyn. In order to decide which data

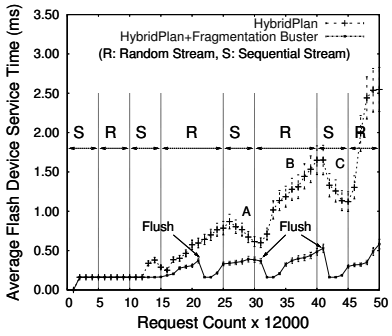


Fig. 2: Performance degradation due to fragmentation on flash and subsequent improvement with fragmentation buster. "Flush" indicates periods of migration activity from flash to disk.

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 HybridStore's performance. When the idle period kicks in, the fragmentation buster directs the flash controller to start flushing the data fragments. In this work we consider it a pure background activity that does not interfere with the servicing of requests and hence we ignore its possible degrading effects on overall performance.

C. Write Regulation

One of the challenges in capacity planning is the unpredictability in workloads. A prolonged and/or recurring period of unanticipated random writes detrimentally impact on lifetime of flash. In this sub-section, we develop techniques for handling sudden unanticipated bursts in requests.

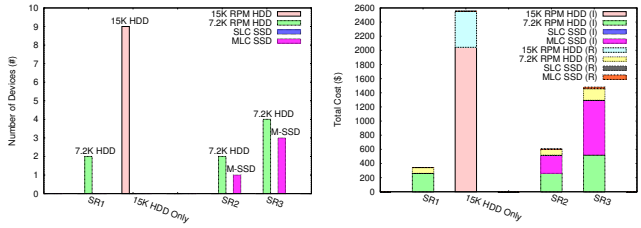
The projections made by HybridPlan are dictated by normal workload characteristics and are subject to violations during operation. The write regulator monitors the erase rate of blocks and comes into action if sustained violations (due to unanticipated write activity) are observed. This is essential to preserve the lifetime budget requirements. On detecting violations, it starts to regulate the writes being sent to flash by over-riding the decisions made by the performance model in HybridDyn. Currently, we use a policy which randomly picks the requests being sent to flash and diverts them to disk instead.

V. Evaluating HybridPlan

We developed the solver of HybridPlan using CPLEX, a well-regarded Integer Linear Programming (ILP) solver written in C. Also, we have written the trace analyser for data classification in C. The source codes are less than 500 lines of code. The Solver execution time is extremely short (in seconds), however the execution time of trace analyser for data classification is dependent on the trace size and can run into minutes for large traces. We use a variety of synthetic and real-world enterprise scale storage traces to evaluate the

	Index	Read (%)	Size (KB)	Inter-Arrival Time (ms)	I/O Bandwidth MB/s	I/O Bandwidth IOPs
Sequential Read	SR1	80	128	100 (L)	1.25	-
	SR2	80	128	2 (M)	62.5	-
	SR3	80	128	0.1 (H)	1,250	-
Random Read	RR1	80	4	100 (L)	-	10
	RR2	80	4	2 (M)	-	500
	RR3	80	4	0.1 (H)	-	10,000
Sequential Write	SW1	20	128	100 (L)	1.25	-
	SW2	20	128	2 (M)	62.5	-
	SW3	20	128	0.1 (H)	1,250	-
Random Write	RW1	20	4	100 (L)	-	10
	RW2	20	4	2 (M)	-	500
	RW3	20	4	0.1 (H)	-	10,000

TABLE IV: Description of synthetic workloads. The letter in parentheses denotes intensity of request's arrival rate, Low, Medium, and High.



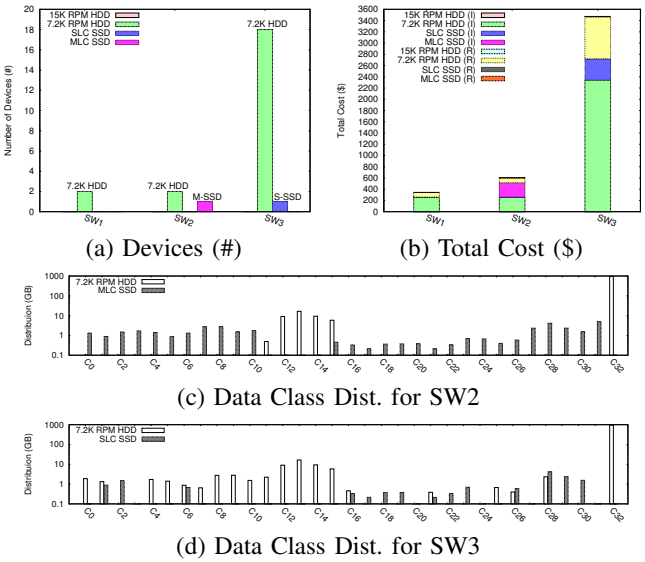
(a) Devices (#) (b) Total Cost ($)

Fig. 3: Study the impact of I/O intensity in the read dominant workloads. In (b) "I" and "R" respectively denote Installation Cost and Recurring Cost.

effectiveness of our solver and the data classification process in provisioning storage.

Table IV describes the characteristics of the synthetic workloads generated using Disksim [3], a well-regarded disk simulator capable of generating workloads based on certain input parameters. The synthetic workloads are divided into 4 categories, Sequential Read (SR), Sequential Write (SW), Random Read (RR) and Random Write (RW) with varying inter-arrival times. We used exponential distribution for varying the inter-arrival times and request sizes between subsequent requests. These workloads help in capturing the variations in the overall workload spectrum which is not possible using a limited number of real-world traces. However, in order to present the application of our solver in a realistic setting, we use the MSR Cambridge traces [14] and MSR Enterprise Traces [15].

We used four devices to evaluate HybridPlan. The details of devices are described in Table V. Device utilization ratio (ratio of amount of actual data stored in the device to its entire storage capacity) needs to be properly set in capacity planning. We set the expected utilization ratio of flash device as 50% while that of hard disk drive is set as 80%. This is based on the observation of Kgil et al. that garbage collection overhead in flash dramatically increases if the utilization exceeds 50% [10]. Also, we have a similar observation in experiment using our flash simulator. The expected disk utilization is set as 50% to provide sufficient storage space. For hard disk drive, since it is much cheaper in $/GB than flash device and most of cold data (rarely accessed) will be stored in the hard disk drive by HybridPlan, we set it as 80%. Moreover, we need to consider device use duration in order to consider the recurring cost of the storage system. We used this period as 5 years for our



(a) Devices (#) (b) Total Cost ($)

(c) Data Class Dist. for SW2

(d) Data Class Dist. for SW3

Fig. 4: Study the impact of I/O intensity in the write dominant workloads. (c) and (d) show data class distributions in SW2 and SW3 synthetic workloads.

evaluation. Note that 10 cents per kilowatt-hour (kWh) is used to estimate electricity cost in our evaluation.

A. *Impact of I/O Intensity*

Figure 3 and 4 show the impact of change in request arrival rate on the storage configurations provided by the solver. With increased I/O intensity, the number of devices increases as well as the type of devices needed to meet the I/O bandwidth requirements changes.

1) Read dominant workloads: In Figure 3(a), SR1 (sequential read only workload with low I/O intensity) only requires 2 slow HDDs. For fast 15K RPM HDDs, it needs nine HDDs to satisfy the capacity demand, which requires much higher cost than 7.2K RPM HDDs only (refer to the cost plot in Figure 3(b). As we increase the I/O intensity, we observe the need for MLC SSDs to satisfy the bandwidth requirements with increased I/O intensity. The choice of only slow HDD in SR1 clearly demonstrates that some workloads merely require storage for capacity and IOPS requirements for them are satisfied trivially. The same is corroborated by Narayanan et al. [15]. However, we contend that even in these situations our solver plays the critical role of determining the right devices to meet the capacity needs. This is demonstrated in Figure 3(b) (workload SR1) where choosing fast HDDs to meet the storage needs instead of slow ones would have resulted in 10 times increase in cost even though the system would have met the bandwidth requirements and not been over-provisioned. Furthermore, we observe that the recurring costs (in terms of power consumption by the storage devices) over the lifetime of the system are quite small as compared with the procurement costs of the devices. Thus, we observe that at least the direct power consumption by the devices is quite minuscule compared with other costs. The readers should note that we have not taken into account the indirect

Device	Type	Capacity (GB)	Per-GB ($)	Utilization	Read (MB/s)	Write (MB/s)	Latency (ms)	Erase (#)	Power (W)
Seagate Cheetah [18]	15K HDD	146	1.80	0.8	171	171	3.6	-	12.92
Seagate Baracuda [17]	7.2K HDD	750	0.17	0.8	125	125	4.2	-	9.4
Intel X25-E [8]	SLC SSD	32	11.96	0.5	230	200	0.125	100K	2
Intel X25-M [9]	MLC SSD	80	3.22	0.5	220	80	0.25	10K	2

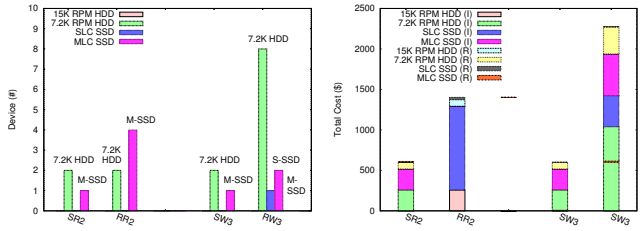
TABLE V: Storage device characteristics. SLC and MLC are denoted by Single-Level Cell and Multi-Level Cell respectively.

power consumption costs such as those due to cooling needs and other storage appliances (e.g: RAID controllers, SAN controllers etc).

2) Write dominant workloads: Similar to the read-dominant workloads, we again observe the need for SSDs for write-intensive workloads to service the IOPS needs of the workloads in Figure 4. However, there are certain subtle differences between the outputs for two workload categories. For write-dominant SW3, we observe the solver including an SLC SSD instead of the MLC ones for its read-intensive counterpart (SR3). This is because SLC SSDs are about 2.5 times faster than the MLC ones (refer to Table V) and hence more suitable for write-intensive workloads with high IOPS. Furthermore, we also observe a sharp increase in the number of slow HDDs with increased write intensity (SW3) in contrast to the rising number of MLC SSDs (SR3). This can be attributed to the vast $/GB difference between SLC SSDs and slow HDDs as shown in Table V. Figure 4(c) and (d) show data class distributions for write dominant workloads (SW2 and SW3)

B. Impact of Sequentiality

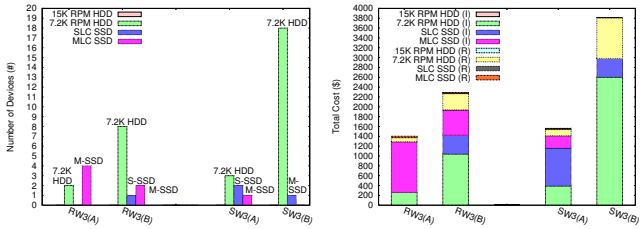
In this subsection, we explore the role of sequentiality on the decision making process of the solver for *iso-intensity* workloads. HDDs are known to perform better for sequential workloads because of reduced seek overhead as compared to the random workloads whereas SSDs are deemed to be primarily random access devices with good performance for both cases (especially for reads as random writes have been shown to have poorer performance comparatively [12], [5]).



(a) Devices (#) (b) Total Cost ($)

Fig. 5: Study the impact of sequentiality in the workloads.

This is confirmed in Figure 5(a) where we clearly observe the need for larger number of SSDs with increased randomness in requests even though the arrival rates remain the same. For read-dominant workloads, we see a 3-fold increase in the number of MLC SSDs to meet the IOPS requirements. This directly translates into a large increase in the overall cost of the storage system (Figure 5(b)). As a consequence, even though the performance constraints for both iso-intensity sequential and random workloads are the same, the cost as well as the type of devices required for provisioning storage are



(a) Devices (#) (b) Total Cost ($)

Fig. 6: Study the impact of lifetime constraints taken into account. "A" and "B" in the parenthesis denote "without lifetime constraint" and "with lifetime constraint" respectively.

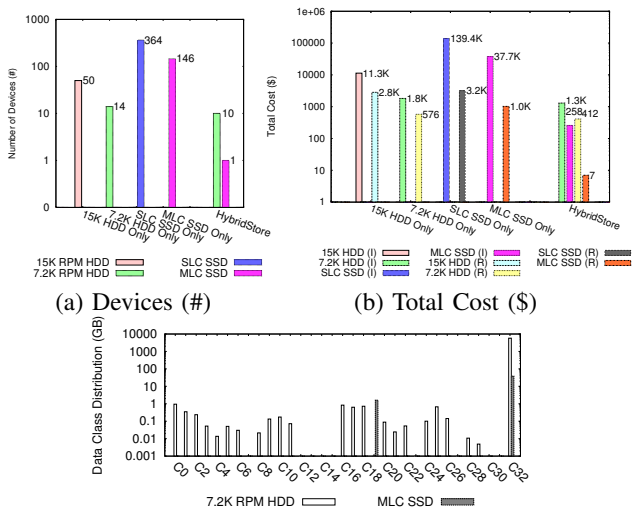
quite different. Hence, as a storage administrator it is highly advisable to increase the sequentiality of incoming workloads.

C. Impact of Lifetime Constraint

We have already established the importance of the lifetime constraint in capacity provisioning since its a long term decision made by a storage administrator. Figure 6 shows the difference in decision making with and without the lifetime constraint for both sequential and random write dominant workloads. The readers should note that we have only used write-intensive workloads because the lifetime of SSDs is directly dependent on block erases which are caused by writes. Without the lifetime constraint, we see a greater proportion of SSDs being used than with the lifetime constraint enforced. As shown in Figure 6(a) for SW3, an MLC SSD is used along with 2 SLC SSDs when the lifetime constraint is removed. However, as soon as the constraint is applied, the solver outputs only 1 SLC SSD since MLC SSDs have lower erase count (lower lifetime for same number of writes) than SLC SSDs (Table V) and would not be suitable in such an environment. Interestingly, the total number of devices as well as the overall costs (Figure 6(b)) are much lower without the lifetime constraint. This is because a relatively cheaper MLC SSD is able to meet the IOPS needs whereas a large number of slow HDDs are needed to meet the performance guarantees when the lifetime constraint is obeyed. However, the cheaper configuration with MLC SSD may not have the needed longevity and the storage administrator might need to re-provision prematurely, thus increasing the overall costs over the initial estimated provisioning period.

D. HybridPlan Study with Real World Traces

We use the MSR Cambridge traces [14] and Microsoft Exchange Server Traces [15] for realistic experiments. The MSR Cambridge traces are composed of several sub-traces that have been collected in different directory for 7 days. Since each of these sub-traces show very low I/O bandwidth demands, we consolidated the traces for aggregated bandwidth taken into account. Since the traces are too huge to run in



(a) Devices (#) (b) Total Cost ($)

(c) Data Class Distribution

Fig. 7: Results of MSR Cambridge Trace.

the solver, the day of the highest I/O arrival rate – 6th day trace has been run by the solver. Table VI summarizes the characteristics of these real workloads used.

Workload	Size (TB)	Read (%)	Request Size (KB)	IOPS
MSR Trace	5.7TB	68.1	23.32	823
Exchange Server	750GB	38.3	16.54	3,692

TABLE VI: Description of realistic traces.

1) Can SSDs replace HDDs?: We examine if SSDs can actually replace HDDs at current price points and if not, then at what price points does it become viable to use a SSD only storage system. Consider a storage system composed of only a single type of HDD. From the results in Figure 7(b), we see that employing 7.2K RPM HDDs is more economically efficient than employing 15K RPM HDDs in the MSR Cambridge traces (Refer to Figure 7(b)). 7.2 K RPM HDD only system requires lesser HDDs than when we consider 15K RPM HDD only system (Refer Figure 7(a)). It is because I/O bandwidth requirement of this trace is not much higher than the I/O bandwidth that HDDs can provide. In Figure 7(c), more than 99% data are classified into C32, a data class storing data rarely accessed. In case of SSD only system, it requires several hundreds of SSDs to satisfy the capacity requirement. we see again that a bounding factor for decision-making of HybridPlan is not I/O bandwidth requirement but storage capacity requirement. A similar observation can be found in [15]. that SSD only system is not an economically viable solution under current market prices of devices. We have similar observation for Microsoft Exchange Server Traces, however, their results are not shown because of lack of pages.

2) Efficacy of HybridStore: From Figure 7(a) we see that HybridPlan can find the most economic storage composition for a workload, given the available device characteristics and their prices. In Figure 7(a), HybridStore is composed of 10 x 7.2K RPM HDDs and 1 MLC SSD by HybridPlan. This is much more cheaper configuration than any of single device only systems See the total cost of HybridStore with those of other storage configurations in Figure 7(b). Total cost saving of HybridStore s about 85(%) compared to high-end HDD only system. HybridStore of this trace is composed of 2 7.2K RPM HDDs and 1 MLC SSD by HybridPlan. It also saves around 69% compared to the 15K RPM HDD only system. We see the data distribution of each data class for both traces respectively in Figure 7(c).

3) What if device prices fluctuate?: To allow price fluctuation, we varied the price of each device. Under current market price, 15K RPM HDD and SLC SSD are relatively more expensive than 7.2K RPM HDD and MLC SSD respectively, Thus, we conducted hypothetical experiments by reducing the device prices of 15K RPM HDD and SLC SSD from their prices in Table V and see how the HybridPlan operates for the Microsoft Exchange Server trace. Table VII show the results of HybridPlan in case of price variation. We consider the following cases for price variation of devices.

- "Base (Baseline)" is when we use current market prices for devices as shown in Table V.
- "A" is for when the price of SLC SSD becomes half while
- "B" is for when the price of 15K RPM HDD becomes 35% from their current market prices.
- "C" is for when both SLC SSD and 15K RPM HDD become 50% and 35% from the current market prices.

As clearly shown in Table VII, the price variation of each device can impact on the decision of HybridPlan. In case of "A", we see that SLC SSD is employed instead of MLC SSD. In case of "B", the price-down, 50% of 15K RPM HDD doesn't change the result of HybridPlan from the baseline, however, in cse of "C", we see that it completely changes it employs 1 15K RPM HDD in addition to 1 7.2K RPM HDD and 1 MLC SSD.

	HDD		SSD	
	15K	7.2K	SLC	MLC
Base	0	2	0	1
A	0	2	1	0
B	0	2	0	1
C	1	1	0	1

TABLE VII: Price fluctuation of device.

	HDD		SSD	
	15K	7.2K	SLC	MLC
Base	0	2	0	1
A	0	2	1	0
B	3	1	0	0
C	3	1	0	0

TABLE VIII: Recurring cost is not taken into account.

4) Impact of recurring costs: We study its impact on HybridPlan's decision making in Table VIII when the recurring cost is not considered in HybridPlan, comparing Figure VII which includes the recurring cost along with the installation cost, Table VIII which only includes the installation cost clearly demonstrates that recurring cost can play a significant role in the capacity planning process. In cases of "B" and "C", HybridPlan decides to use more 15K RPM HDDs that those results in VII. It is primarily because the recurring cost due to power consumption in HDDs is not taken into account by the decision-making of HybridPlan.

VI. Evaluating HybridDyn

We develop a simulation framework for integrated disk and flash based storage systems, called HybridSim. It is built by

integrating Disksim [3] (a well-regarded HDD simulator) and SSD simulator [11]. HybridSim is able to simulate different storage sub-system components including device drivers, controllers, caches, flash devices, disks, and various interconnects. It is capable of simulating multiple HDDs and SSDs. However, for our evaluation we consider a simple system consisting of a single HDD and SSD. Table IX illustrates the characteristics of enterprise-scale workloads used in our evaluation.

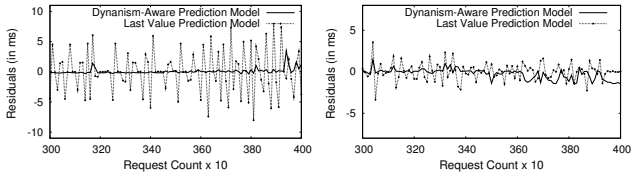
Workloads	Request Size (KB)	Read (%)	Sequentiality (%)	Inter-arrival Time (ms)
Financial (OLTP) [19]	4.38	9.0	2.0	133.50
Cello99 [7]	5.03	35.0	1.0	41.01
TPC-H (OLAP) [20]	12.82	95.0	18.0	155.56

TABLE IX: Enterprise-Scale Workload Characteristics.

We evaluate the performance of prediction models in HybridDyn along with our novel mechanisms such as (i) fragmentation busting and (ii) write-regulation.

A. Dynamism-Aware Performance Prediction Model for SSD

We use the Financial trace and TPC-H workloads in Table IX to validate our model. 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 8 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 smaller error rate.



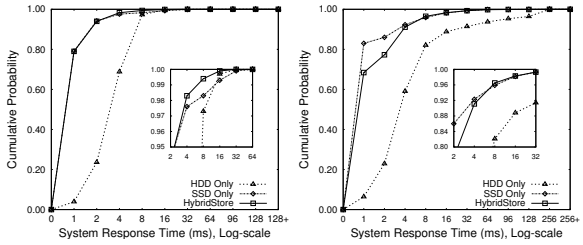
(a) Financial (b) TPC-H

Fig. 8: Comparison of our dynamic SSD performance prediction model with a simple last value-based prediction model.

B. Evaluation with Our Dynamism-aware 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 (HybridDyn) partitions write requests depending on the least response times predicted by the SSD and HDD models. HybridDyn 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. Read requests are always serviced from the device which contains the data.

Figure 9(a) illustrates the performance of HybridStore incorporating the prediction models in HybridDyn with respect to a disk-only and flash-only system for the random write



(a) Log-scale (b) Log-scale

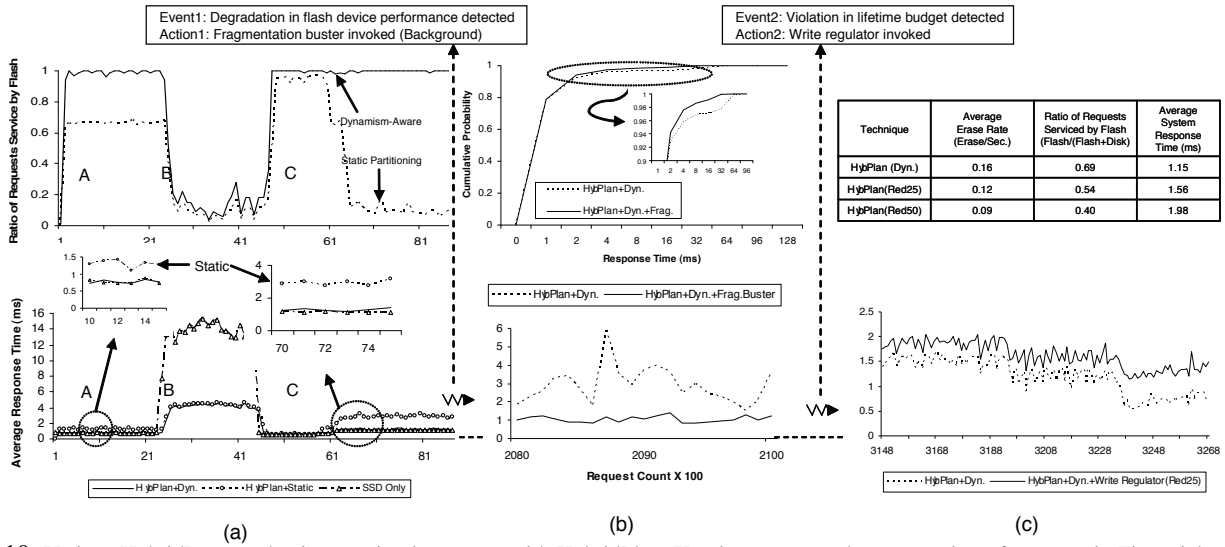
Fig. 9: Performance of HybridStore compared with a disk-only and a flash-only system. (a) and (b) show the CDF of system response time for Financial Trace and Cello99.

dominant Financial trace. Although we observe good performance from flash device for servicing most requests, but some requests suffer from extensive GC overhead thus exhibiting high response time on flash. Our prediction model is able to move these requests to the disk and achieve better performance for HybridStore. Moreover, HybridStore reduces the average system response time by about 71% as compared to a disk-only system. Similar performance improvement is observed for Cello. However, the limitation of simplistic disk prediction model is observed for Cello in Figure 9(b) where flash-only system improves response time by about 20% as compared to HybridStore. The disk prediction model (in HybridStore) is unable to capture the high intensity of random writes resulting in incorrect prediction by HybridDyn since some high latency requests are now inevitably wrongly serviced from disk.

C. HybridDyn at Work: A Microscopic Look

Figure 10 shows HybridDyn at work for parts of the financial trace. (a) compares the performance of hybrid system along with dynamism-aware data partioner(dyn.) with a static data partitioning policy and a SSD only system. Region B represents a period of intense, large, sequential write requests requiring GC on SSD and degrading the performance in a SSD only system. In regions "A" and C", the dynamism-aware data partitioner is able to make better performance predictions than a static policy, thus reducing the response time. (b) HybridDyn observes degradation in SSD performance and invokes Fragmentation buster (Frag. Buster) during idle periods, thus providing sustained improved performance. (c) HybridDyn observes violation in lifetime guarantees made by HybridPlan and invokes the write regulator causing small degradation in response time since some requests which could have been serviced from flash are now sent to disk. However, this helps in reducing the erase rate and meeting lifetime budget while still meeting performance guarantees. Red_{25} and Red_{50} represent different write-regulation policies.

Our dynamic data partitioner is able to intelligently partition the incoming requests, thus improving the system response time as compared to the static data partitioner (Figure 10-(b)). The fragmentation buster is able to reduce the tail of the CDF of response time (Figure 10-(b)), thus reducing the number of requests that experience high response time. We experiment with a write regulator that detects increased



Technique	Average Erase Rate (Erase/Sec.)	Ratio of Requests Serviced by Flash (Flash/(Flash+Disk)	Average System Response Time (ms)
HybPlan (Dyn.)	0.16	0.69	1.15
HybPlan(Red25)	0.12	0.54	1.56
HybPlan(Red50)	0.09	0.40	1.98

Fig. 10: Various HybridDyn mechanisms acting in concert with HybridPlan. X-axis represents the progression of requests in Financial trace.

I/O activity and consistently monitors the expected flash life through the lifetime model of HybridPlan.

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 redirect them to HDD during periods of higher-than-expected I/O intensity. Let us call these policies Red_{25} *and* Red_{50}*, respectively.* Figure 10-(c) 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.

VII. 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 complementary devices to HDDs in problems in such a hybrid system employing HDDs and SSDs. We developed an on-line capacity planner called HybridPlan that used an ILP technique to provide storage administrators with guidelines on provisioning a hybrid system in a cost-effective manner. HybridPlan tool can be used for systems that employ heterogeneous devices (not just limited to SSDs and HDDs). We then developed a dynamic controller, HybridDyn, that used shorter time-scale SSD and HDD models along with regulation of write-rate to the SSD. We demonstrated the efficacy of HybridDyn acting in concert with HybridPlan.

Acknowledgment

Most of this work was done during Youngjae Kim's graduate studies at Penn State. We would like to thank the anonymous reviewers for their detailed comments which helped us improve the quality of this paper. This research was funded in part by NSF grant CCF-0811670.

References

[1] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy. Design tradeoffs for SSD performance. USENIX ATC, 2008.

[2] F. Chen, D. A. Koufaty, and X. Zhang. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. SIGMETRICS, 2009.

[3] G. R. Ganger, B. L. Worthington, and Y. N. Patt. *The DiskSim Simulation Environment Version 3.0 Reference Manual*, 2003.

[4] J. Guerra, H. Pucha, J. Glider, W. Belluomini, and R. Rangaswami. Cost effective storage using extent based dynamic tiering. FAST, 2011.

[5] A. Gupta, Y. Kim, and B. Urgaonkar. DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings. ASPLOS, 2009.

[6] S. Gurumurthi, A. Sivasubramaniam, and V. K. Natarajan. Disk drive roadmap from the thermal perspective: A case for dynamic thermal management. ISCA, 2005.

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

[8] Intel. Intel X25-E Extreme SATA Solid-State Drive. http://www.intel.com/design/flash/nand/extreme/index.htm.

[9] Intel. Intel X25-M and X18-M Mainstream SATA Solid-State Drives. http://www.intel.com/design/flash/nand/mainstream/index.htm.

[10] T. Kgil, D. Roberts, and T. Mudge. Improving NAND flash based disk caches. ISCA, 2008.

[11] Y. Kim, B. Taurus, A. Gupta, and B. Urgaonkar. Flashsim: A simulator for NAND flash-based solid-state drives. SIMUL, 2009.

[12] S. Lee and B. Moon. Design of flash-based DBMS: An in-page logging approach. SIGMOD, 2007.

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

[14] D. Narayanan, A. Donnelly, and A. Rowstron. Write off-loading: Practical power management for enterprise storage. *ACM Trans. Storage*, 4(3):1–23, 2008.

[15] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron. Migrating server storage to SSDs: analysis of tradeoffs. EuroSys, 2009.

[16] N. Schirle and D. F. Lieu. History and trends in the development of motorized spindles for hard disk drives. *IEEE Transactions on Magnetics*, 32(3):1703–1708, May 1996.

[17] Seagate. Seagate Baracuda 7.2K. http://www.seagate.com/www/en-us/products/desktops/barracuda_hard_drives/.

[18] Seagate. Seagate Cheetah 15K.5. http://www.seagate.com/www/en-us/products/servers/cheetah/.

[19] Storage-Performance-Council. OLTP Trace from UMass Trace Repository. http://traces.cs.umass.edu/index.php/Storage/Storage.

[20] J. Zhang, A. Sivasubramaniam, H. Franke, N. Gautam, Y. Zhang, and S. Nagar. Synthesizing representative I/O workloads for TPC-H. HPCA, 2004.