Adaptive policies for balancing performance and lifetime of mixed SSD arrays through workload sampling

Sangwhan Moon
Department of Electrical and Computer Engineering
Texas A&M University
College Station, Texas
Email: sangwhan@tamu.edu

A. L. Narasimha Reddy
Department of Electrical and Computer Engineering
Texas A&M University
College Station, Texas
Email: reddy@tamu.edu

***Abstract*—Solid-state drives (SSDs) have become promising storage components to serve large I/O demands in modern storage systems. Enterprise class (high-end) SSDs are faster and more resilient than client class (low-end) SSDs but they are expensive to be deployed in large scale storage systems. It is an attractive and practical alternative to exploit the high-end SSDs as a cache and low-end SSDs as main storage.**

This paper explores how to optimize a mixed SSD array in terms of performance and lifetime. This paper shows that simple integration of different classes of SSDs in traditional caching policies results in poor reliability. This paper also reveals that caching policies with static workload distribution are not always efficient.

In this paper, we propose a sampling based adaptive approach that achieves fair workload distribution across the cache and the storage. The proposed algorithm enables fine-grained control of the workload distribution which minimizes latency over lifetime of mixed SSD arrays. We show that our adaptive algorithm is very effective in improving the latency over lifetime metric, on an average, by up to 2.36 times over LRU, across a number of workloads.

I. Introduction

Solid-state drive (SSD) arrays are expected to accelerate a large volume of transactions in modern storage systems. Although they have shown their effectiveness in performance, concerns remain about their high cost per gigabyte and limited write endurance.

There are different classes of SSDs for different applications. Enterprise class (high-end) fast SSDs use I/O interfaces such as PCI express (PCIe). The high-end SSDs usually consist of single-level cell (SLC) flash memory whose write endurance is of the order of 100K write cycles which is large enough to endure enterprise workloads for a few years. However, the high-end SSDs are expensive per gigabyte to be deployed in large scale storage systems. On the other hand, a client class (low-end) SSD uses traditional serial ATA (SATA) interface and may employ multi-level cell (MLC) flash memory which is cheaper per gigabyte than SLC. However, the write endurance

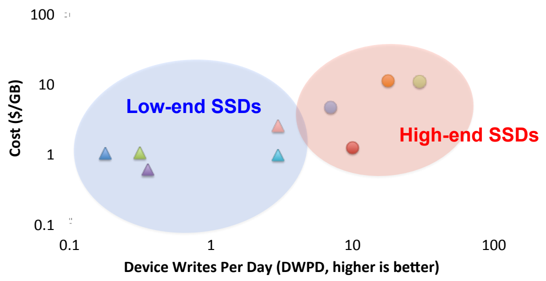


Fig. 1. Different classes of SSDs: cost per gigabyte (lower is better) vs. device writes per day (higher is better)

of MLC is an order of magnitude less than SLC, in 10K-30K [1].

Figure 1 shows the different classes of commercial SSDs with their cost and reliability. Each point shows cost per gigabyte and device writes per day (DWPD) of recent SSDs from various vendors. The DWPD is a widely used industrial metric for the reliability of an SSD. It means that the lifetime of the SSD is only guaranteed when the entire device is written less than DWPD times per day. The figure shows high-end SSDs provide higher reliability while low-end SSDs provide cost-efficiency.

Several vendors are offering SSD arrays combining these devices in a storage hierarchy. These systems employ SLC flash as a cache and MLC flash as backend storage [3]. These systems try to improve performance at a lower cost per byte. Not much work has been done in understanding the data lifetimes in such arrays. While the high-end SLC flash can improve lifetimes due to higher write endurance, they tend to absorb majority of the workload when employed as a cache. In an SSD array, the performance improves as cache hit rates go up and more and more requests are satisfied at the speed

[1]Recent triple-level cell (TLC) is cheaper than MLC, but it has only 3K write endurance which is two orders of magnitudes smaller than SLC [1]. Phase change memory (PCM) has 10^8-10^{10} write endurance which is two orders of magnitude larger than SLC [2].

of SLC flash. However, as a higher fraction of workload is absorbed in the cache, the SLC cells may wear out faster due to higher rate of writes than the MLC cells in the backend storage which absorbs only a small fraction of the write workload at higher hit rates. As a result, the lifetimes of data in an SSD array which equals the minimum of the data lifetimes in both the cache and the backend may be reduced as the cache absorbs a higher workload and wears out in time faster than the backend storage. Hence, it is important to balance the workload in such an SSD array considering both performance and data lifetimes.

The imbalance in lifetime between cache and main storage motivates the investigation of new approach that considers both performance (latency) and lifetime of different classes of SSDs.

Several studies have looked at the problem of distributing load across SSDs and HDDs based on hotness of data [4], size of the request [5], [6] and equalization of response times [7]. These approaches are focused on maximizing performance. Our approach here considers both performance and lifetimes in distributing loads across different types of SSDs.

We do not consider the issues of configurations in this paper. Several issues including performance, cost and other factors determine a configuration that may be employed. We consider the problem of balancing the performance and lifetime given a configuration. This problem is important since a machine may be employed to serve multiple workloads and since workloads evolve over time, making it necessary to balance the lifetime and performance over time irrespective of the decisions involved at the time of configuration.

The major contributions of this paper are:

- We find that high-end SSD cache can wear out faster than low-end SSDs main storage and this could result in lower lifetimes of mixed SSD arrays.
- We introduce a new metric, a latency over lifetime, to control the trade-off between the performance (latency) and lifetime. The metric is minimized when latency is smaller and lifetime is larger.
- We propose a sampling based approach to appropriately distribute workload across cache and backend storage for trading off performance and lifetime in mixed SSD arrays. We show that the proposed approach improves latency over lifetime in such arrays by up to 2.36 times.

The rest of the paper is organized as follows. Section II states the problem this paper targets to solve. Section III discusses existing caching policies. Section IV analyzes workload distribution of those different caching policies. Section V shows our sampling based approach for adaptive workload distribution. Section VI evaluates different caching policies with our adaptive workload distribution algorithm. Section VII introduces related work on cache and SSD storage systems and Section VIII concludes the paper.

II. Problem Statement

Storage systems could be designed to provide average latencies below a target performance metric[2]. We call this latency as target latency in this paper. When the estimated latency exceeds the target latency, caching policy would be required to load more data in the cache. When the cache is utilized on every request and the latency targets are not being met, the only alternative would be to increase the size of the cache. We assume, for this paper, we have to operate with a given cache size.

Both latency and lifetime are optimized such that a *latency over lifetime* is minimized. The goal of this paper can be restated as an optimization problem to minimize the latency over lifetime subject to a latency constraint in Equation (1).

$$\begin{aligned} &\text{minimize} \quad L \;/\; T \\ &\text{subject to} \\ &\qquad L \leq L_{max} \end{aligned} \tag{1}$$

where L is the expected latency, L_{max} is the target latency constraint, and T is the expected lifetime.

III. Background: Selective Caching policies

Traditional caching policies typically employ a static workload classifiers (thresholds). For example, Bcache [5] caches only requests below a certain static threshold. However, this could result in less efficient workload distribution, especially, when workload characteristics are skewed and/or changing over time.

In addition to traditional LRU caching policy, different caching policies can be applied in mixed SSD arrays for different characteristics of enterprise workloads. Caching policies based on request size and hotness of workload are introduced in this section. Additionally, a probability based caching policy is proposed to precisely control workloads across SSD arrays.

A. Request size based caching policy

Recent solutions [5], [6] propose to selectively cache requests whose size is less than a threshold. This policy has been advocated considering competitive sequential read/write performance of low-end storage device arrays. Although the target of the solution is an SSD cache for hard disk main storage, this is possibly effective in SSD arrays with an SSD cache because 1) large sequential I/O requests push out valid data in cache, and 2) sequential I/O performance of main storage is usually better than random read/write performance. However, the threshold for classifying sequential/random I/O should be determined carefully. Since the optimal size (threshold) depends on the characteristics of workload which changes over time. The work in [5] uses a static threshold (4 MB) and caches I/O requests whose sizes are less than the 4MB threshold.

[2]There have been many cache performance metrics such as cache hit rate, latency, and throughput. We use latency in this paper.

B. Hotness based caching policy

Hotness of data has been used to determine which data should be cached and which data should be directly written to storage. Many hotness based caching policies have been proposed. Among a variety of hotness based caching policies we choose 2Q replacement policy [4], as a representative. As its name indicates, 2Q employs two LRU queues of block references. Data returned on a cache miss is initially stored in the second level queue and the block reference is only promoted to the first level queue when it is referenced more than threshold (originally, this threshold is 1) while in the second level queue. Alternatively, we can track reference counts and promote block references whose reference count is in the top 10% of tracking history. In this policy, only blocks in the first level queue are actually cached. The size of the second level queue is configurable and it is equal to the size of actual cache in this paper.

C. Probability based caching policy

Traditional caching policies are not optimized to precisely control the workload distribution across the cache and storage components of the system. For example, under the workload mostly consisting of one size (4KB) requests, size based caching policy is not able to distribute the workload. Consequently, a new caching policy is introduced where requests are *probabilistically* cached. Unlike other policies mentioned above, the purpose of the probability based caching policy is to remove dependencies between workload distribution and workload characteristics.

The probabilistic caching policy is not optimal in terms of cache hit rate since it does not guarantee that cached data is more likely to be referenced in the near future than the bypassed data. Nevertheless, the caching policy is still effective since loss in performance is generally marginal because the bypassed hot data could be loaded on the next cache miss.

IV. Analysis of Workload Distribution

In this section, we analyze workload distribution with different caching policies discussed in the previous section.

Table I shows the list of symbols frequently used in this paper. For example, r and w are the read and write request rates, and m_r and m_w are read and write cache miss rates, respectively.

A. LRU Caching Policy

Figure 2 shows the architecture and workload distribution of a mixed SSD array. In the figure, least recently used (LRU) caching policy, one of the most popular caching policies, is employed. Other caching policies are discussed in Section III.

In Figure 2, read misses $m_r \cdot r$ in the cache become read requests at the storage (arrow 3). These requests will result in writes (arrow 4) at the cache. Read misses and write misses require space allocation in the cache. Clean data in the cache can be discarded while dirty data should be flushed to the storage at rate $d \cdot (m_r \cdot r + m_w \cdot w)$ on an average where d is the dirty ratio in the cache.

TABLE I
List of Symbols

Symbols	Description
l_c, l_s	Write endurance of flash (Cache, Storage)
c_c, c_s	Capacity of an SSD (Cache, Storage)
c_w	Unique data size of workload
N_c, N_s	The number of SSDs (Cache, Storage)
r	Read workload intensity
w	Write workload intensity
m_r	Read cache miss rate
m_w	Write cache miss rate
w_c, w_s	Actual writes in cache (w_c) and storage (w_s)
f_c, f_s	Wear out rate per flash cell (Cache, Storage)
d	The portion of dirty data (Cache)
$t_{c,r}$, $t_{c,w}$	Read ($t_{c,r}$) and write ($t_{c,w}$) latency (Cache)
$t_{s,r}$, $t_{s,w}$	Read ($t_{s,r}$) and write ($t_{s,w}$) latency (Storage)

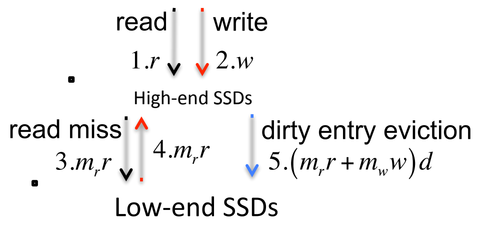


Fig. 2. A mixed SSD array with LRU caching policy (red line: writes in cache, blue line: writes in main storage)

When workloads have weak locality in reads, cache can wear out substantially with marginal benefits in performance. In this case, selective caching of read cache misses can save write endurance of cache with acceptable degradation in performance. However, bypassing read cache misses for hot data can result in severe degradation in performance. Write workload is entirely absorbed in cache first (arrow 2), and only a portion of writes is flushed to main storage (arrow 5). Write intensive workloads with strong locality therefore can wear out the cache faster than main storage. Appropriate distribution of writes between the cache and the storage improves the lifetime of the cache, but it can degrade the performance of the storage system. This paper studies this tradeoff between performance and lifetime in an SSD array where the cache and backend storage employ different types of flash devices.

With LRU caching policy, flushing of dirty data results in writes at main storage as indicated by the arrow 5 in Figure 2. The writes served in storage per flash memory cell w_s is

$$w_s = \frac{(m_r \cdot r + m_w \cdot w) \cdot d}{N_s \cdot c_s} \tag{2}$$

where N_s is the number of SSDs in main storage and c_s is the capacity of the SSDs.

In this paper, we assume perfect wear leveling in both cache and main storage. According to Figure 2, the writes served in cache per flash memory cell, w_c, is the sum of the read misses (arrow 4) and the write workload (arrow 2) per flash memory cell.

$$w_c = \frac{m_r \cdot r + w}{N_c \cdot c_c} \tag{3}$$

where N_c is the number of SSDs in cache and c_c is the capacity of the SSDs.

And cache and storage wear out at the rate of f_c and f_s, respectively.

$$\begin{aligned} f_c &= \frac{w_c}{l_c} = \frac{m_r \cdot r + w}{N_c \cdot c_c \cdot l_c} \\ f_s &= \frac{w_s}{l_s} = \frac{(m_r \cdot r + m_w \cdot w) \cdot d}{N_s \cdot c_s \cdot l_s} \end{aligned} \tag{4}$$

With SLC caches, data may reside in the cache for considerably long time before it is migrated to storage and hence the minimum of cache and storage lifetimes determine the data lifetime. Thus, the achievable lifetime of a mixed SSD array T can be restated as

$$T = \frac{1}{\max(f_c,\ f_s)} \tag{5}$$

Under the assumptions of equal write amplification in the cache and the storage, we can use these equations to estimate the lifetime of mixed SSD arrays.

The relationship between performance and lifetime is complex and non-linear. This implies that the performance benefits from cache potentially result in significant loss in lifetime of the mixed SSD arrays in practical configurations. Thus, the trade-offs between performance and lifetime of the storage systems should be tuned carefully.

B. Selective Caching Policies

In Section III, many selective caching policies are discussed. Figure 3 shows how probabilistic caching policy is different from LRU in terms of workload distribution. Similar analysis can be carried out for other policies that may be different from the ones studies in this paper. In Figure 3, only a portion p of read cache miss is recorded in cache for future accesses (arrow 3). The higher the p is, the more we write in the cache on a read miss. The caching factor p can be different for each caching policy. It is determined by the size, hotness and probability in the the size-based, hotness-based and probabilistic policies respectively. In addition, a portion $1-p$ of write cache misses go directly to main storage (arrow 5) which does not exist in LRU policy in Figure 3. As we increase the portion p, frontend cache serves a higher fraction of workload and wears out faster. The performance can also be tuned by changing the same probability parameter p, thus enabling an effective control mechanism to optimize for both metrics of performance and lifetime.

C. Write amplification

The above analysis assumed that write amplification is the same both in the cache and the storage. When this is not the case, the lifetimes can be modified to appropriately account for them in Equation (2) and (3), based on observed statistics.

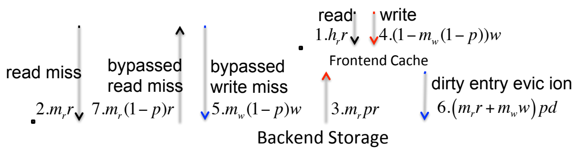


Fig. 3. Workload distribution of a mixed SSD array with selective caching policy (red: write, black: read)

TABLE II
PRACTICAL CONFIGURATION OF SSDS AND WORKLOAD

Item	Description	Specification
High-end (SLC) SSD	Capacity	100 GB
	Write endurance	100 K
	Read/write latency	0.04ms/0.2ms
Low-end (MLC) SSD	Capacity	200 GB
	Write endurance	10 K
	R/W latency	0.2ms/1.0ms
Workload I	Read / write (MB/s)	160 / 200
	R/W cache hit rate	90%/85%
	R/W length	16KB/16KB
	Dirty data in cache	65%
Workload II	Read / write (MB/s)	100 / 250
	R/W cache hit rate	50%/15%
	R/W length	4KB/64KB
	Dirty data in cache	81%

$$\begin{aligned} w_s' &= \beta_s \cdot \frac{(m_r \cdot r + m_w \cdot w) \cdot d}{N_s \cdot c_s} \\ w_c' &= \beta_c \cdot \frac{m_r \cdot r + w}{N_c \cdot c_c} \end{aligned} \tag{6}$$

where β_s and β_c are observed write amplification in the cache and the storage, respectively.

The write amplifications within the cache and the storage are determined by various factors. The amount of overprovisioning, the differences in the workloads seen at cache and storage and the management policies, among others, play a role in determining the write amplification. In a practical system, the write amplification can be measured and used appropriately as shown in Eq. (6).

D. Case Study

Table II shows practical parameters of different classes of SSDs and enterprise workloads. Those parameters are extracted from commercial SSD datasheet and block level I/O traces for enterprise workload.

When we employ one high-end SSD as a cache for 3 low-end SSDs for Workload I in Table II, for example, the high-end SSD cache absorbs all write workload as well as 10% of read workload (read cache miss) in writes. The amount of writes per flash memory cell in the high-end SSD is w_c and the SLC wears out at the rate f_c:

$$\begin{aligned} w_c &= \frac{160MB/s \cdot 0.1 + 200MB/s}{1 \cdot 100GB} \\ &= 2.16e\text{-}3\ writes\ /\ cell\ /\ sec \\ f_c &= \frac{6.75e\text{-}3}{100K} = 2.16e\text{-}8\ /\ cell\ /\ sec \end{aligned} \tag{7}$$

This implies that the workload consumes $2.16e\text{-}8$ of SLC's write endurance per second, or SLC's write endurance would be consumed in 1.47 years.

Cache miss initiates cache data eviction. While clean data is simply removed in cache, dirty data should be written back to main storage. The rate of writeback w_s, and resulting wear out rate of MLC f_s can be estimated in Equation (8).

$$\begin{aligned} w_s &= \frac{(160MB/s \cdot 0.1 + 0.15 \cdot 200MB/s) \cdot 0.65}{600GB} \\ &= 4.98e\text{-}5 \; writes \;/\; cell \;/\; sec \qquad (8) \\ f_s &= \frac{4.98e\text{-}5}{10K} = 4.98e\text{-}9 \;/\; per \;/\; sec \end{aligned}$$

In other words, the MLC will use up its write endurance in 6.37 years.

This simple example shows that high-end SSDs cache can wear out faster and lose data before low-end SSDs wear out, and the lifetime of the mixed SSD array is bounded to the shorter lifetime, 1.47 years in this case.

The average latency can be computed for each configuration. In this example, the average latency is 0.136 ms.

When the same configuration goes through Workload II in Table II, however, we can see the different workload distribution. Cache wears out at the rate f_c in Equation (9).

$$\begin{aligned} w_c &= \frac{100MB/s \cdot 0.5 + 250MB/s}{1 \cdot 100GB} \\ &= 3.00e\text{-}3 \; writes \;/\; cell \;/\; sec \qquad (9) \\ f_c &= \frac{3.00e\text{-}3}{100K} = 3.00e\text{-}8 \;/\; cell \;/\; sec \end{aligned}$$

Meanwhile, storage wears out at the following rate f_s:

$$\begin{aligned} w_s &= \frac{(0.50 \cdot 100MB/s + 0.85 \cdot 250MB/s) \cdot 0.81}{600GB} \\ &= 3.54e\text{-}4 \; writes \;/\; cell \;/\; sec \qquad (10) \\ f_s &= \frac{3.54e\text{-}4}{10K} = 3.54e\text{-}8 \;/\; per \;/\; sec \end{aligned}$$

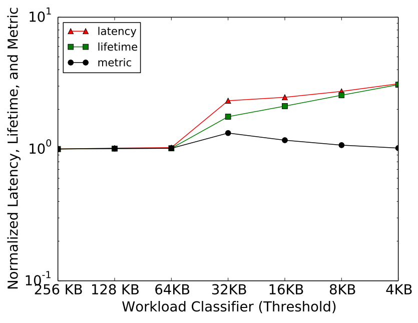
The results in Equation (9) and (10) show that the lifetime of SLC is expected to be 1.06 years, while MLC's lifetime is 0.90 years. The lifetime of the mixed SSD array is bounded to the lifetime of MLC (the lower), 0.90 years, in this example.

The latency of the mixed SSD array is 0.173 ms on an average.

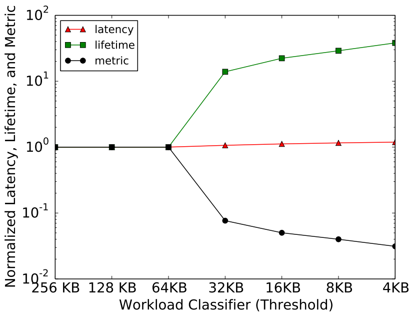
V. Adaptive workload distribution

Static workload classifiers have been used in many caching policies. For example, we can selectively cache requests whose size is less than or equal to 64 KB, and send requests larger than 64 KB directly to main storage. We can configure hotness based caching policy to promote block references (from second level queue to first level queue) whose reference is hotter than top 10% of reference count tracking history.

In this section, we show that those static workload classifiers can be less efficient. Based on the analysis, we propose a sampling based approach which makes existing selective caching policies adaptive to the characteristics of workloads.



(a) Hardware monitoring server



(b) Web server

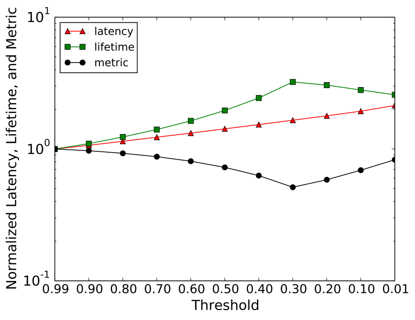
Fig. 4. Static threshold based analysis of size based caching policy.

A. *Static workload distribution*

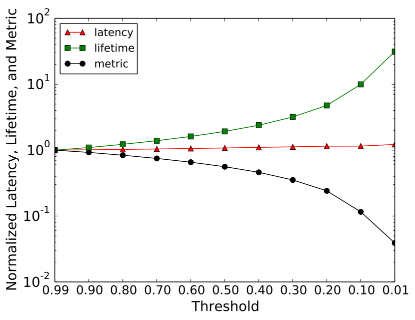
We explore the effectiveness of different static workload classifiers for different enterprise workloads. We use our own trace-driven simulator with enterprise workload traces from [8]. Details of the simulator are discussed later in Section VI-A. Figure 4 shows latency, lifetime, and latency over lifetime of a size based caching policy with different thresholds. These two traces are used to show the possible impact of static thresholds on different workloads. Performance of all the traces are shown later in the paper. In the figure, we normalize the results of different thresholds to that of 256 KB.

In Figure 5, we can clearly see that there is no ideal threshold which can be applied across both the workloads. Each workload has different optimal threshold in terms of latency over lifetime. Figure 4a shows that 64KB is the optimal threshold for hardware monitoring server trace. Meanwhile, the optimal threshold of web server application in Figure 4b is 4 KB.

We also track latency, lifetime, and latency over lifetime of probabilistic caching policy with different static thresholds in Figure 5. In the figure, static threshold is the probability of caching and it is applied from the beginning to the end of traces. The result shows the average latency over lifetime for a week, and the result is normalized to the result of threshold of 0.99. The results in Figure 5a show that lifetime increases as the threshold is decreased from 0.99 to 0.3 and the workload is distributed away from faster wearing high-end SSDs. However, as more wokload is moved to back-end SSDs, the back-end SSDs start to wear out faster and decrease lifetime below the



(a) Hardware monitoring server



(b) Web server

Fig. 5. Static threshold based analysis of probabilistic caching policy

threshold of 0.3. The optimal workload distribution is reached at a threshold of 0.3 when the wearing rates at both tiers are matched. In Figure 5b, however, it is better to send most of the workload directly to main storage. As mentioned before, the main purpose of the probability based caching policy is to precisely control workload distribution. The results in the figure shows that it can control the workload distribution fairly well.

The results in Figure 4 and 5 show that the latency increases as a smaller fraction of workload is served in the cache. However, the latency over lifetime does not always monotonically increase as the threshold is increased as shown in Figure 5a.

B. Sampling based adaptive approach

Figure 4 and 5 show that the latency over lifetime does not monotonically change and the control function may change from workload to workload. In order to adapt to each workload and to different phases of the workload, we propose a sampling based approach to adapt automatically to different characteristics of workloads. Our approach is to sample the workload and run them in separate sample caches with different workload classifiers (thresholds) to find the optimal threshold. This enables the estimation of the latency over lifetime vs. threshold curve. Based on the estimation, we can choose the optimal threshold. The optimal threshold can be adaptive to the change of the characteristics of workload.

Figure 6 shows the architecture of the sampling based adaptive threshold algorithm applied to a selective caching policy. It is noted that the sampling based approach can be employed with any selective caching policies. We control threshold in sampling based approach, and the threshold is size in size based caching policy while it is hotness in hotness based caching policy and probability in probability based caching policy, respectively. We use a hash function to sample the workload randomly. We maintain multiple sample caches that run the cache policy with different thresholds. The size of each sample cache is maintained proportional to the size of the sampled workload. Each sample cache works independently and the results from the sample caches are used to set the thresholds for the main cache. Figure 6 shows 10 sample caches with each cache supporting 0.1% of workload (with 0.1% of cache space in each) and the main cache supporting 99% of the remaining workload. In Figure 6, sample caches employ different thresholds and the threshold employed by the main cache is based on the observed results of the sampled caches.

The latency, lifetime, and resulting latency over lifetime of each sample cache is periodically updated. For each timeframe s, we estimate lifetime of each sample cache (considering the lifetime of corresponding storage) $T[s]$:

$$T[s] = min(\frac{l'_s[s]}{w'_s[s]}, \frac{l'_c[s]}{w'_c[s]}) \tag{11}$$

$$\begin{aligned} w'_c[s] &= \alpha w'_c[s-1] + (1-\alpha) \cdot w_c[s] \\ w'_s[s] &= \alpha w'_s[s-1] + (1-\alpha) \cdot w_s[s] \end{aligned} \tag{12}$$

$$\begin{aligned} l'_c[s] &= l_c - \frac{\sum_{k=1}^{s-1} w_c[k]}{N_c \cdot C_c} \\ l'_s[s] &= l_s - \frac{\sum_{k=1}^{s-1} w_s[k]}{N_s \cdot C_s} \end{aligned} \tag{13}$$

where $w_c[s]$ and $w_s[s]$ are write count per flash cell in sample cache and corresponding storage at timeframe s, and $l'_s[s]$ and $l'_c[s]$ are remaining lifetime of sample cache and corresponding storage, respectively. The smoothing factor α depends on sample rate and size of the timeframe. We use $\alpha = 0.8$ for 1% sampling rate, $\alpha = 0.9$ for 5% and 10% sampling rate, in this paper. The lifetime $T[s]$ shows how much time remains from timeframe s to reach the end of lifetime of the SSD array.

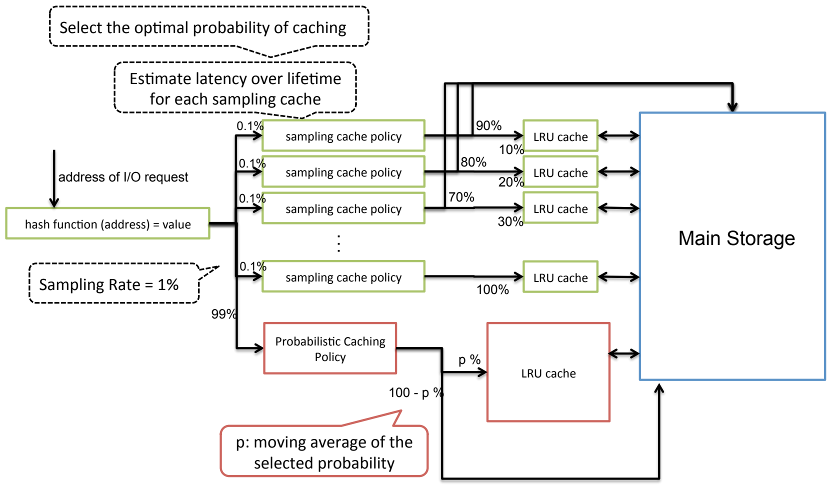
The optimal threshold is the threshold of the cache with the least latency over lifetime. The optimal threshold p_c is applied to the rest of the cache (except sample caches) in an adaptive way:

$$p_c[s] = \alpha \cdot p_c[s-1] + (1-\alpha) \cdot p_s[s] \tag{14}$$

where $p_s[s]$ is the selected threshold in sample caches in timeframe s, and $p_c[s]$ is the threshold applied to the rest of cache (except sample caches) at timeframe s.

The result of sample caches violating a latency constraint are excluded in the optimal threshold selection. When all sample cache violates a latency constraint, it choose minimum threshold and receives all workload in cache to work as traditional LRU cache which is the best in performance.

In this paper, default sampling rate is 1% unless specified.



Select the optimal probability of caching
Estimate latency over lifetime for each sampling cache
address of I/O request
hash function (address) = value
Sampling Rate = 1%
0.1%
sampling cache policy
90%
10%
LRU cache
80%
20%
70%
30%
100%
99%
Probabilistic Caching Policy
p %
100 - p %
p: moving average of the selected probability
Main Storage

Fig. 6. Architecture of probabilistic cache with sampling method (sampling rate: 10%)

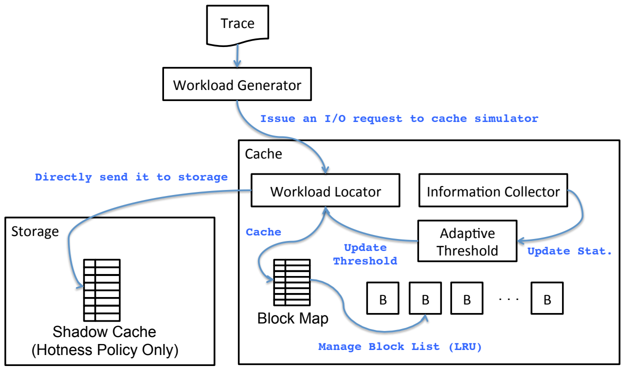


Fig. 7. Trace-driven simulator

VI. Evaluation

In this section, different caching policies in Section III are evaluated when the proposed adaptive workload distribution approach is employed.

A. Simulator

We built a trace-driven simulator based on the analysis in Section I to see the behavior of mixed SSD arrays with different caching policies in Section III. Figure 7 illustrates the details of our trace-driven simulator.

In the simulator, statistical information such as cache hit rate and actual read/write count in cache and storage is collected and periodically updated by information collector. Various metrics such as wearing out rate (and resulting expected lifetime) and average latency are estimated based on the information. For hotness based caching policy, both cache and shadow cache additionally maintain frequency (hotness) of references as described in Section III-B.

In the figure, the workload locator assigns appropriate SSDs to serve incoming I/O requests using a workload classifier (threshold) which is either a static constant or an adaptive variable. Each caching policy exploits the threshold in a different way. Request size based caching policy sends I/O requests whose size is less than a threshold to cache. Hotness based caching policy caches data whose reference count is more than a hotness parameter. Probability based policy handles only a portion p of I/O requests in cache where p is a threshold. The adaptive threshold algorithm periodically updates the threshold based on the information from the sample caches. Details of the adaptive threshold algorithm is discussed in Section V.

In the simulator, internals of SSD (FTL, garbage collection, and write amplification) are simplified and the simulation of its detailed behavior remains as future work.

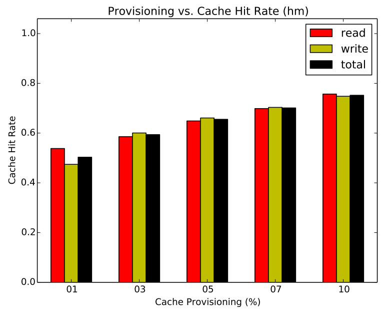
B. Simulation environment

We employ enterprise workload traces from Microsoft Research Cambridge [8]. These 13 enterprise applications exhibit different characteristics; they have different cache hit rates vs. cache provisioning [3], read/write ratios, request size distribution, total unique data size, reference frequency (hotness) etc.

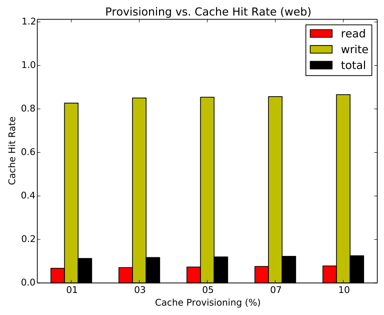
Due to lack of space in the paper, among the 13 MSRC traces, we choose 2 applications as representatives and show their cache hit rate vs. cache provisioning in Figure 8. The characteristics of a full set of applications are discussed in a recent study [9]. It is observed from Figure 8 that cache provisioning impacts performance of different workloads differently. It is observed that the hit rates of the hardware monitoring application improve considerably with higher cache provisioning. However, the web server workload in Figure 8b doesn't benefit significantly from higher cache provisioning. In this case, increasing cache size is not efficient. An appropriate strategy therefore should be established for each workload. Due to the diversity of the characteristics of applications, static workload classifiers are overall less efficient than adaptive workload classifiers.

In this paper, cache provisioning is 5% unless specified, and other provisioning numbers from 1% to 10% are explored as well. The capacity of main storage is twice the size of unique

[3]The cache provisioning is the ratio of cache size to storage size.



(a) Hardware monitoring



(b) Web server

Fig. 8. Cache hit rates vs. cache provisioning

data in the workload, i.e. the space utilization of main storage is 50%.

A default target latency of 0.4ms is considered while other latency constraints such as 0.2 ms and 1.0 ms are discussed. Many parameters such as queue depth and average I/O request size determine the relationship among performance metrics such as throughput, IOPS, and latency.

Different classes of SSDs have different read/write latencies. We use 0.04 ms and 0.2 ms of latencies for read and write operations in high-end SSDs, and 0.2 ms and 1.0 ms of latencies for read and write operations in low-end SSDs, respectively. Those numbers are from recent measurement results of commercial SSDs [10].

We use latency over lifetime as an effectiveness metric in this paper. The metric is lower (and desirable) when the expected latency is lower and/or the expected lifetime is higher.

Some caching policies like [11] favor dirty data in cache to reduce write workload in main storage. We assume that cache data eviction algorithm does not consider the dirty/clean status of data.

C. Adaptive threshold algorithm

Figure 9 shows how the proposed adaptive threshold algorithm tracks the change of the characteristics of workload online.

Figure 9a shows the change of threshold according to the change of an average cache miss rate of sample caches. In the figure, it is clearly shown that threshold decreases for workload with weak locality. Higher cache miss rate in sample caches implies weak locality of workload, and caching such workloads results in faster cache wear out with marginal benefits in performance. The threshold adapts to the cache miss rates in sample caches and saves write endurance of cache in this case.

Figure 9b shows how the proposed algorithm controls the threshold considering reliability. In the figure, red line shows write ratio of workload and blue and green lines show normalized wearing rates of cache and storage in the SSD array, respectively. Black line shows the smoothed value of threshold applied to cache.

Since web server application is read intensive and the read workload has weak locality, large number of read cache misses wear out the cache faster than the storage without appropriate workload distribution. In terms of performance, the cache doesn't improve latency significantly because of low hit rates. In this case, bypassing cache can save cache lifetime without significant penalty in performance. As a result, most of the read misses (99%) are served directly by main storage and this improves the latency over lifetime metric.

There are occasional bursts of writes in the workload. During these bursts, the estimated wearing rate of storage exceeds the wearing rate of cache and hence the adaptive algorithm steers some of these writes to cache by increasing the threshold.

Figure 9a and 9b illustrate how the sampling approach adapts to workloads and different phases in a workload for balancing both performance and lifetime.

For some observation periods, none of the sample caches may meet the target latency. In such cases, LRU policy is employed as a default and all the requests are sent through the cache.

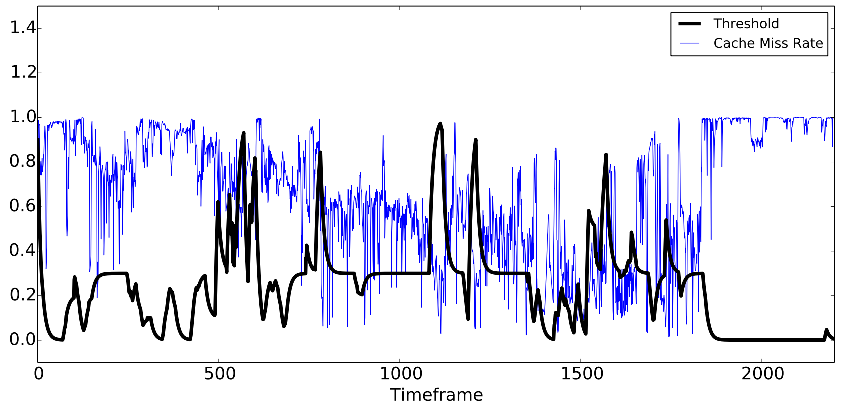
D. Different caching policies

We evaluate different caching policies in Figure 10. In the figure, y-axis shows latency, lifetime, and latency over lifetime normalized to the results of LRU caching policy for each MSRC trace.

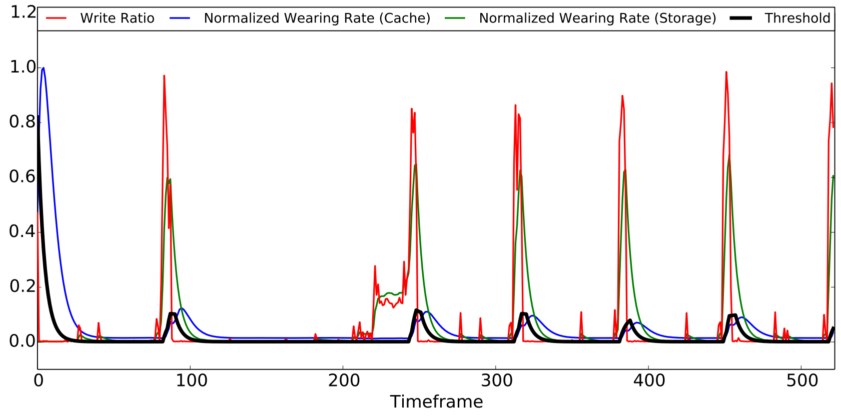
The results show that the adaptive caching policies work better than LRU when the latency over lifetime metric is considered. Probability based caching policy is on average 2.36 times better than LRU caching policy. Size based caching policy is 2.31 times, and hotness based caching policy is 1.41 times better than LRU, on an average across the 13 traces.

Figure 10a shows that latency can increase with the adaptive policies when compared to LRU policy. This is intentional as the adaptive policies are designed to tradeoff latencies for improving lifetimes. It is also noted that latencies are designed to stay below a target latency even with the adaptive policies. Figure 10b shows that adaptive policies improve lifetimes significantly by appropriately distributing the workloads. For all the 13 workloads, lifetimes are improved compared to LRU.

Trading latency for lifetime is especially beneficial for workloads with weak locality, because weak locality can wear out cache significantly while low hit rates don't contribute



(a) Printer server



(b) Web server

Fig. 9. Adaptive threshold (probability of caching) in probabilistic caching policy

significantly to improving performance. Among the 13 workloads, mds, stg, web, prn, usr, proj, and src1 have weak locality. Our solution is selectively caching data and avoids caching data with weak locality.

However, the proposed algorithm can be less effective than LRU caching policy for workloads with strong locality. Under such workloads, decreasing the probability of caching can result in significant loss in cache hit rates and performance. Among the traces, hm, prxy, rsrch, and wdev are such workloads. Even though LRU policy is employed in one of the sample caches, it may not always be selected. Caching history of adaptive cache is different from the history of a pure LRU cache. As a result, even when the adaptive algorithm adapts caching policy towards LRU, the performance may lag due to the differences in working sets in the cache when different policies are employed during the bursts.

E. Cache provisioning

The impact of the cache provisioning on latency over lifetime is shown in Figure 11. In the figure, the results are normalized to the result of LRU with 1% cache provisioning.

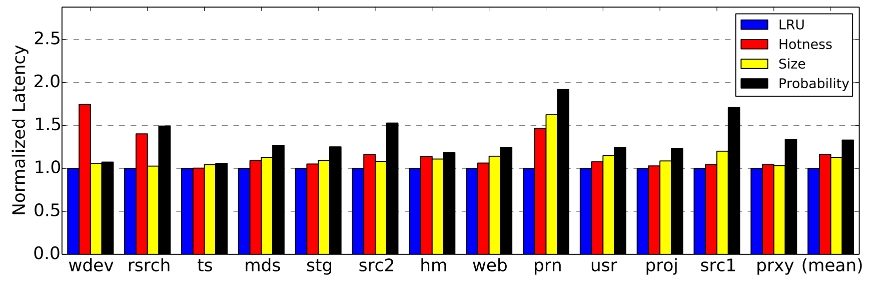
The figure shows that higher cache provisioning improves latency over lifetime by both reducing latency and enhancing lifetime. We find that the average latency across all the traces increases by less than 40% for all the adaptive policies compared to LRU at all the levels of cache provisioning. The adaptive policies improve lifetimes by significantly more than this at all the levels of cache provisioning to improve the overall metric of latency over lifetime.

F. Target latency

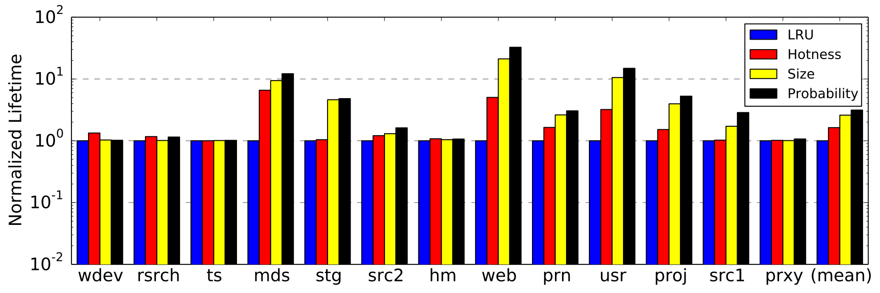
The target latency prevents adaptive workload classifiers from sending too much workload directly to main storage and under-utilize high-end SSDs cache. Figure 12 observes the behavior of cache as the target latencies is varied.

The results in Figure 12 show the trade-off between latency and lifetime clearly. The latency over lifetime is smaller (or improved) when the latency constraint is relaxed. Larger latency targets provide more opportunities for performance-lifetime trade-offs, thus, enhancing the benefits from the adaptive approach.

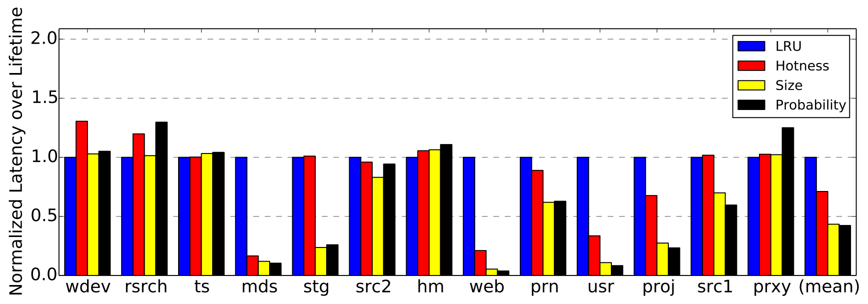
Given a configured machine, by extending its lifetime within given performance bounds, we reduce the cost of ownership



(a) Latency (lower is better)



(b) Lifetime (higher is better)



(c) Latency over lifetime (lower is better)

Fig. 10. Different caching policies on enterprise workloads, target latency = 0.4 ms

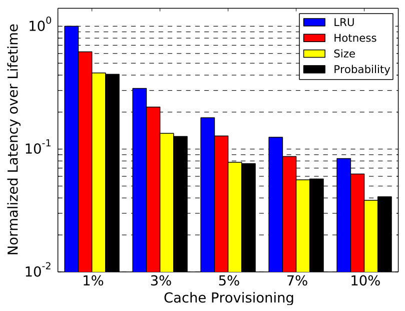


Fig. 11. Latency over lifetime vs. cache provisioning

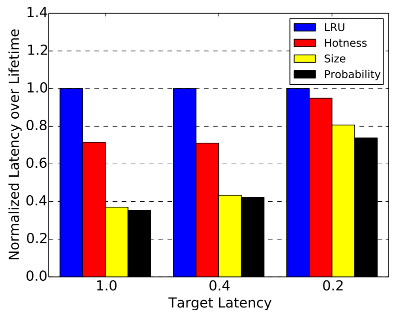


Fig. 12. Latency over lifetime vs. target latencies

of the storage systems. However, the maintenance costs may dominate the hardware purchase costs.

G. Sampling rate

We used a sampling rate of 1% in this paper. When we increase the sampling rate, we can expect more accurate optimal threshold estimation. However, increasing sampling rate reduces the amount of workload (and the size of the cache)

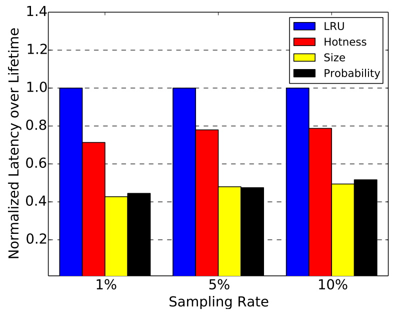


Fig. 13. Latency over lifetime vs. sampling rates

which benefits of the optimal threshold selection. For example, 99% of workload can be distributed with 1% sampling rate, while 90% of workload can be distributed by more accurate threshold with 10% sampling rate.

The results from different sampling rates are shown in Figure 13. The results show that for the workloads considered in this paper, 1% sampling rate provides better benefits.

It is noted that for higher sampling rates, we can use smaller observation periods such that the threshold can follow the optimal value faster. At higher sampling rates, shorter timeframes may suffice to provide sufficient sampled data to estimate the thresholds accurately.

It is also noted that the computation required is minimal (updating of counters) and hence will not have an impact on total performance.

VII. Related Work

SSDs are promising storage components in modern storage systems while their reliability is still a concern for its users. The reliability of SSD is widely studied from ECC [12–14], to FTL algorithm [11], [15–18], to wear-leveling [19] and to RAID [20] over SSD arrays [21–24].

Employing SSD cache for HDDs storage systems is a practical configuration to exploit lower latency of SSDs. Many [5], [6], [25–29] have suggested different issues on employment of SSD cache for HDDs based storage systems. The work [25] studies the feasibility of flash as disk cache and improve SSD cache's performance by separating read and write cache, and cache's reliability by employing strong ECC. Another work [26] implemented software interface of SSD cache and provides functionalities such as data protection and silent data eviction. The study in [27] examines interesting issues when SSDs are placed in client-side of large scale storage systems. The work in [28] introduces a deduplication technique and optimizes capacity usage of an SSD cache. The recent work [29] predicts working set size of virtual machine storage and minimize SSD cache usage based on the prediction.

The impact of write amplification on the performance and reliability of flash memory is widely studied in the work [24], [30–34]. In this paper, we assume write amplification is 1.0 for SSDs in mixed SSD arrays. However, different write amplification factors can be considered in our analysis as discussed in Section IV-C.

A recent work [35] controls the effective size of SSD cache considering write amplification from less efficient garbage collection when space utilization of the SSD cache is higher. They control the valid data size in SSD cache and find the optimal point of the SSD cache where performance and lifetime of the SSD cache is maximized. Unlike the study, we balance those metrics of both SSD cache and SSD main storage at the same time.

Two recent studies [36] and [37] propose a device controller to balance faster and more reliable (SLC) flash and slower and less reliable (MLC) flash in terms of both performance and lifetime. The work [36] controls workload distribution by sending a portion of frequent (hot) write workload to SLC and the rest to MLC in an SSD. They use a control system to adjust wearing and latency of SLC and MLC flash chips. Unlike those studies, this paper considers an all SSD array where faster SSDs are used as a cache for slower SSDs. We consider both reads and writes since read cache misses result in writes in the cache.

The work in [37] assumes that flash chips can be switched between SLC and MLC and controls the amount of flash in SLC mode and MLC mode considering the workload.

The related work [38] improves the lifetime of SSD main storage by efficient usage of NVRAM cache. They increase cache hit rate and reduce write workload in main storage by dividing cache space by four (clean, dirty, frequent, recent) and by adjusting those four spaces in an SSD cache using a secondary cache. The goal of the study is to reduce the number of writes to main storage while improving cache hit rate, assuming that the cache is robust enough. Unlike this study, our target system employs high-end SSDs cache which may wear out faster than low-end SSDs main storage. The goal of our study is to maximize lifetime of an entire storage system considering wearing of cache and main storage at the same time.

The recent work [39] counts read cache misses as write amplification in high-end SSDs cache, and proposed hotness based caching policy with a new garbage collection policy. Their focus is on the SSD cache and do not consider the latency and lifetime of main storage. They use cache hit rate as performance metric while we use average latency. In addition, their hotness and request size based caching policies use static classifiers while we employ adaptive variable classifiers.

Several studies [40–42] have investigated the problem of adapting cache sizes in multiple levels of caching.

VIII. Conclusion

SSD arrays are receiving wide attention as a storage component for enterprise storage systems. In this paper, we showed that mixed SSD arrays using different classes of SSDs in a hierarchical manner should consider both latency and lifetime.

We showed that high-end SSDs as a cache can wear out faster than low-end SSDs main storage under enterprise workloads. Based on the analysis, we argue that caching policies should balance the latency and lifetime of cache and storage at the same time.

We propose a sampling based method for adaptive workload distribution in mixed SSD arrays. The proposed solution enables fine-grained control of workload distribution and balances latency and lifetime effectively in such SSD arrays. Our trace-driven simulations show that the proposed method is adaptive to different workloads and can improve latency over lifetime metric by up to 2.36 times over a pure LRU policy.

REFERENCES

[1] L. M. Grupp, J. D. Davis, and S. Swanson, "The bleak future of nand flash memory," in *FAST*, Feb. 2012.

[2] H. Kim, S. Seshadri, C. Dickey, and L. Chiu, "Evaluating phase change memory for enterprise storage systems: A study of caching and tiering approaches," in *FAST*, Feb. 2014.

[3] Pure Storage, FA-400, http://www.purestorage.com/products/fa-400/.

[4] T. Johnson and D. Shasha, "2q: A low overhead high performance buffer management replacement algorithm," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 439–450. [Online]. Available: http://dl.acm.org/citation.cfm?id=645920.672996

[5] Google, Bcache, http://bcache.evilpiepirate.org/.

[6] Facebook, FlashCache, https://github.com/facebook/flashcache/.

[7] X. Wu and A. L. N. Reddy, "Exploiting concurrency to improve latency and throughput in a hybrid storage system," in *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on*, Aug 2010, pp. 14–23.

[8] D. Narayanan, A. Donnelly, and A. Rowstron, "Write off-loading: practical power management for enterprise storage," in *FAST*, 2008.

[9] J. Wires, S. Ingram, Z. Drudi, N. Harvey, and A. Warfield, "Characterizing storage workloads with counter stacks," in *OSDI*, 2014.

[10] Anandtech, "Intel ssd sd p3700 review: The pcie ssd transition begins with nvme," http://www.anandtech.com/show/8104/intel-ssd-dc-p3700-review-the-pcie-ssd-transition-begins-with-nvme/3.

[11] S.-y. Park, D. Jung, J.-u. Kang, J.-s. Kim, and J. Lee, "Cflru: A replacement algorithm for flash memory," in *Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, ser. CASES '06. New York, NY, USA: ACM, 2006, pp. 234–241. [Online]. Available: http://doi.acm.org/10.1145/1176760.1176789

[12] F. Sun, K. Rose, and T. Zhang, "On the use of strong bch codes for improving multilevel nand flash memory storage capacity," in *in IEEE Workshop on Signal Processing Systems (SiPS): Design and Implementation*, 2006.

[13] B. Chen, X. Zhang, and Z. Wang, "Error correction for multi-level nand flash memory using reed-solomon codes," in *Signal Processing Systems, 2008. SiPS 2008. IEEE Workshop on*, Oct 2008, pp. 94–99.

[14] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. Nevill, "Bit error rate in nand flash memories," in *Reliability Physics Symposium, 2008. IRPS 2008. IEEE International*, April 2008, pp. 9–19.

[15] E. Gal and S. Toledo, "Algorithms and data structures for flash memories," *ACM Comput. Surv.*, vol. 37, no. 2, pp. 138–163, Jun. 2005. [Online]. Available: http://doi.acm.org/10.1145/1089733.1089735

[16] F. Chen, T. Luo, and X. Zhang, "Caftl: A content-aware flash translation layer enhancing the lifespan of flash memory based solid state drives," in *Proceedings of the 9th USENIX Conference on File and Stroage Technologies*, ser. FAST'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 6–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=1960475.1960481

[17] H. Kim and S. Ahn, "Bplru: A buffer management scheme for improving random writes in flash storage," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies*, ser. FAST'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 16:1–16:14. [Online]. Available: http://dl.acm.org/citation.cfm?id=1364813.1364829

[18] A. Gupta, Y. Kim, and B. Urgaonkar, "Dftl: A flash translation layer employing demand-based selective caching of page-level address mappings," in *Penn State University*, 2008.

[19] L.-P. Chang, "On efficient wear leveling for large-scale flash-memory storage systems," in *Proceedings of the 2007 ACM Symposium on Applied Computing*, ser. SAC '07. New York, NY, USA: ACM, 2007, pp. 1126–1130. [Online]. Available: http://doi.acm.org/10.1145/1244002.1244248

[20] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (raid)," in *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '88. New York, NY, USA: ACM, 1988, pp. 109–116. [Online]. Available: http://doi.acm.org/10.1145/50202.50214

[21] Q. Xin, E. Miller, T. Schwarz, D. D. E. Long, S. Brandt, and W. Litwin, "Reliability mechanisms for very large storage systems," in *Mass Storage Systems and Technologies, 2003. (MSST 2003). Proceedings. 20th IEEE/11th NASA Goddard Conference on*, April 2003, pp. 146–156.

[22] B. Mao, H. Jiang, D. Feng, S. Wu, J. Chen, L. Zeng, and L. Tian, "Hpda: A hybrid parity-based disk array for enhanced performance and reliability," in *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, April 2010, pp. 1–12.

[23] M. Balakrishnan, A. Kadav, V. Prabhakaran, and D. Malkhi, "Differential raid: Rethinking raid for ssd reliability," *Trans. Storage*, vol. 6, no. 2, pp. 4:1–4:22, Jul. 2010. [Online]. Available: http://doi.acm.org/10.1145/1807060.1807061

[24] N. Jeremic, G. Mühl, A. Busse, and J. Richling, "The pitfalls of deploying solid-state drive raids," in *Proceedings of the 4th Annual International Conference on Systems and Storage*, ser. SYSTOR '11. New York, NY, USA: ACM, 2011, pp. 14:1–14:13. [Online]. Available: http://doi.acm.org/10.1145/1987816.1987835

[25] T. Kgil, D. Roberts, and T. Mudge, "Improving nand flash based disk caches," in *Computer Architecture, 2008. ISCA '08. 35th International Symposium on*, June 2008, pp. 327–338.

[26] M. Saxena, M. M. Swift, and Y. Zhang, "Flashtier: A lightweight, consistent and durable storage cache," in *Proceedings of the 7th ACM European Conference on Computer Systems*, ser. EuroSys '12. New York, NY, USA: ACM, 2012, pp. 267–280. [Online]. Available: http://doi.acm.org/10.1145/2168836.2168863

[27] D. A. Holland, E. Angelino, G. Wald, and M. I. Seltzer, "Flash caching on the storage client," in *Presented as part of the 2013 USENIX Annual Technical Conference (USENIX ATC 13)*. San Jose, CA: USENIX, 2013, pp. 127–138. [Online]. Available: https://www.usenix.org/conference/atc13/technical-sessions/presentation/holland

[28] C. Li, P. Shilane, F. Douglis, H. Shim, S. Smaldone, and G. Wallace, "Nitro: A capacity-optimized ssd cache for primary storage," in *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. Philadelphia, PA: USENIX Association, Jun. 2014, pp. 501–512. [Online]. Available: https://www.usenix.org/conference/atc14/technical-sessions/presentation/li_cheng_1

[29] D. Arteaga, J. Cabrera, J. Xu, S. Sundararaman, and M. Zhao, "Cloudcache: On-demand flash cache management for cloud computing," in *14th USENIX Conference on File and Storage Technologies (FAST 16)*. Santa Clara, CA: USENIX Association, Feb. 2016, pp. 355–369. [Online]. Available: https://www.usenix.org/conference/fast16/technical-sessions/presentation/arteaga

[30] X.-Y. Hu, E. Eleftheriou, R. Haas, I. Iliadis, and R. Pletka, "Write amplification analysis in flash-based solid state drives," in *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference*, ser. SYSTOR '09. New York, NY, USA: ACM, 2009, pp. 10:1–10:9. [Online]. Available: http://doi.acm.org/10.1145/1534530.1534544

[31] A. Jagmohan, M. Franceschini, and L. Lastras, "Write amplification reduction in nand flash through multi-write coding," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, May 2010, pp. 1–6.

[32] S. Boboila and P. Desnoyers, "Write endurance in flash drives: Measurements and analysis," in *Proceedings of the 8th USENIX Conference on File and Storage Technologies*, ser. FAST'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 9–9. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855511.1855520

[33] P. Desnoyers, "Mathematical models of write amplification in ftls," Slides, presented in NVRAMOS 2011, http://dcslab.hanyang.ac.kr/nvramos11fall/presentation/Desnoyers-NVRAMOS-2011.pdf, Korea, Republic of, 2011.

[34] ——, "Analytic modeling of ssd write performance," in *Proceedings of the 5th Annual International Systems and Storage Conference*, ser.

SYSTOR '12. New York, NY, USA: ACM, 2012, pp. 12:1–12:10. [Online]. Available: http://doi.acm.org/10.1145/2367589.2367603

[35] Y. Oh, J. Choi, D. Lee, and S. Noh, "Caching less for better performance: balancing cache size and update cost of flash memory cache in hybrid storage systems," in *FAST*, Feb. 2012.

[36] M. Murugan and D. Du, "Hybrot: Towards improved performance in hybrid slc-mlc devices," in *MASCOTS*, 2012.

[37] S. Lee, K. Ha, K. Zhang, J. Kim, and J. Kim, "Flexfs: A flexible flash file system for mlc nand flash memory," in *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*, ser. USENIX'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 9–9. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855807.1855816

[38] Z. Fan, D. Du, and D. Voigt, "H-arc: A non-volatile memory based cache policy for solid state drives," in *MSST*, 2014.

[39] J. Yang, N. Plasson, G. Gillis, and N. Talagala, "Hec: Improving endurance of high performance flash-based cache devices," in *Proceedings of the 6th International Systems and Storage Conference*, ser. SYSTOR '13. New York, NY, USA: ACM, 2013, pp. 10:1–10:11. [Online]. Available: http://doi.acm.org/10.1145/2485732.2485743

[40] N. Megiddo and D. S. Modha, "Arc: A self-tuning, low overhead replacement cache," in *Proceedings of the 2Nd USENIX Conference on File and Storage Technologies*, ser. FAST '03. Berkeley, CA, USA: USENIX Association, 2003, pp. 115–130. [Online]. Available: http://dl.acm.org/citation.cfm?id=1090694.1090708

[41] S. Huang, Q. Wei, J. Chen, C. Chen, and D. Feng, "Improving flash-based disk cache with lazy adaptive replacement," in *Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on*, May 2013, pp. 1–10.

[42] R. Santana, S. Lyons, R. Koller, R. Rangaswami, and J. Liu, "To arc or not to arc," in *7th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 15)*. Santa Clara, CA: USENIX Association, Jul. 2015. [Online]. Available: https://www.usenix.org/conference/hotstorage15/workshop-program/presentation/santana