RELIABILITY OF SSD STORAGE SYSTEMS

A Dissertation

by

SANGWHAN MOON

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	A. L. Narasimha Reddy
Committee Members,	Riccardo Bettati
	Paul Gratz
	Henry Pfister
Head of Department,	Miroslav M. Begovic

December 2015

Major Subject: Computer Engineering

ABSTRACT

Solid-state drives (SSDs) are attractive storage components due to their many attractive properties, however, concerns about their reliability still remain and this delays the wider deployment of the SSDs.

Many protection schemes have been proposed to improve the reliability of SSDs. For example, some techniques like error correction codes (ECC), log-like writing of flash translation layer (FTL), garbage collection and wear leveling improve the reliability of SSD at the device level. Composing an array of SSDs and employing system level parity protection is one of the popular protection schemes at the system level. 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. The high-end SSD cache equipped on a low-end SSD array enhances both latency and reduces write count of the SSD storage system at the same time.

This work analyzes the effectiveness of protection schemes originally designed for HDDs but applied to SSD storage systems. We find that different characteristics of HDDs and SSDs make integration of those solutions in SSD storage systems not so straight-forward.

This work, at first, analyzes the effectiveness of the device level protection schemes such as ECC and scrubbing. A Markov model based analysis of the protection schemes is presented. Our model considers time varying nature of the reliability of flash memory as well as write amplification of various device level protection schemes. Our study shows that write amplification from these various sources can

significantly affect the benefits of protection schemes in improving the lifetime. Based on the results from our analysis, we propose that bit errors within an SSD page be left uncorrected until a threshold of errors are accumulated. We show that such an approach can significantly improve lifetimes by up to 40%.

This work also analyzes the effectiveness of parity protection over SSD arrays, a widely used protection scheme for SSD arrays at system level. The parity protection is typically employed to compose reliable storage systems. However, careful consideration is required when SSD based systems employ parity protection. Additional writes are required for parity updates. Also, parity consumes space on the device, which results in write amplification from less efficient garbage collection at higher space utilization. We present a Markov model to estimate the lifetime of SSD based RAID systems in different environments. In a small array, our results show that parity protection provides benefit only with considerably low space utilizations and low data access rates. However, in a large system, RAID improves data lifetime even when we take write amplification into account.

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

To my father, my mother, and my sister.

ACKNOWLEDGEMENTS

I sincerely appreciate my advisor, Dr. A. L. Narasimha Reddy, for his guidance and consistent support with patience on this study. I am thankful to my committee members, Dr. Paul Gratz, Dr. Henry Pfister, and Dr. Riccardo Bettati, for their precious time to review and comment on this research.

And I especially thank to my family, my father, my mother, and my sister for their their continuing love and sincere support.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Solid state drives (SSDs) storage systems are receiving wide attention and their deployment is steadily increasing because of their higher performance and lower power consumption than hard disk drives (HDDs) storage systems. Advanced integration techniques such as multi-level cell (MLC) have considerably dropped cost-per-bit of SSDs such that wide deployment of SSDs is feasible [17, 18]. While their deployment is steadily increasing, the write endurance of SSDs still remains as one of the main concerns. Since the write and erase operations on an SSD wear it out gradually, after a certain number of operations, data could potentially be lost [19,58]. This write endurance problem is related to the physical features of flash memory, the most popular storage media for SSDs[1].

1.1 Device Level Protection Schemes

Flash memory can employ single-level cells (SLC) or multi-level cells (MLC). While MLC allows significant improvements of capacity over SLC, the lifetime of MLC is comparably lower, at 10,000 Program/Erase (P/E) cycles compared to 100,000 P/E cycles for SLC.

Many studies have investigated the bit error failure behavior of SLC and MLC. Two notable studies in this direction include [19,58]. These studies point out that the bit error rate of the flash memory increases with increased number of P/E cycles. In fact, these studies model the bit error rate as an exponential function of the number of P/E cycles the cell has gone through. This variable bit error rate requires further study to understand the implication on the flash memory.

Many approaches have been suggested to overcome the write endurance limi-

[1]We assume flash memory based SSDs only in this dissertation.

tation. An error correction code (ECC) [57] encodes data and stores the encoded data in order to detect and correct errors at a page level, of size, say 4KB. The ECC is checked and used to correct any detected errors whenever the page is read. Scrubbing [49, 54] actively scans data pages and detects/corrects latent errors in infrequently accessed pages using ECC. Flash memory employs a Flash Translation Layer (FTL) in order to provide wear-leveling of blocks. Flash cells have to be erased before they can be programmed, requiring a copy-on-write mechanism. This copy-on-write mechanism is exploited to spread the writes across all the memory such that frequent writes to one address do not result in some memory cells being worn out while other memory cells are not written.

Some of the protection methods mentioned above, however, require additional writes and these writes in turn can cause wear out of SSD. For example, log-like writing of FTL can cause fragmentation which requires garbage collection process, which results in writes and erases when it moves fragmented data to a different place in the memory. Pages are corrected and rewritten to the memory when ECC detects bit errors in the pages. The recovery process issues an additional write to write the corrected page. While ECC is beneficial to detect and correct the bit errors, the extra writes lead to higher bit error rates and potentially can lead to lower lifetimes. Scrubbing increases the chances to detect and correct errors in a page, before the page accumulates too many errors to become uncorrectable by ECC. However, frequent recovery from scrubbing could lead to extra writes and in turn could lower the SSD's lifetime.

Because of copy-on-write, garbage collection and other operations, write to one block may actually result in writes to more than one block. The ratio of actual number of writes to the number of writes issued to the device is termed *write amplification*. While these excess writes may be necessary, the writes cause wear out of

flash cells and hence can potentially reduce the SSD's lifetime.

In Section 2, we explore the relationship between the write amplification of the device level data protection schemes and the reliability of an SSD.

1.2 System Level Protection Schemes I: Parity Protection

When the device level protection schemes does not recover data corruption enough, system level protection schemes can be employed additionally. Composing an array of SSDs and employing system level parity protection is one of the popular protection schemes at the system level. In Section 3, we study popular system level protection schemes: striping (RAID0), mirroring (RAID1), and RAID5.

RAID5 has been employed to improve the lifetime of HDD based storage systems for decades [46]. However, careful decisions should be made with SSDs when the system level parity protection is employed. First, SSDs have limited write endurance. Parity protection results in redundant writes whenever a write is issued to the device array. Unlike HDDs, redundant writes for parity update can severely degrade the lifetime of SSDs. Parity data consumes device capacity and increases the space utilization. While it has not been a serious problem in HDDs, increased space utilization leads to less efficient garbage collection which in turn increases the write workload.

Many studies have investigated SSD based RAID systems. The notable study [26] points out the pitfalls of SSD based RAID5 in terms of performance. They discuss the behavior of random writes and parity updates, and conclude striping provides much higher throughput than RAID5. We consider the impact of write workload on reliability. Previous studies [16, 55] have considered different architectures to reduce the parity update performance penalty.

Section 3 explores the relationship between parity protection and the lifetime of

an SSD array. Different parameters such as the number of SSDs in an SSD array, the amount of data per device, and average write length of workload are extensively explored in the section.

1.3 System Level Protection Schemes II: A Mixed SSD Array

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 SLC flash memory whose write endurance is of the order of 100K write cycles, large enough to endure enterprise workloads for a few years. However, the high-end SSDs are expensive per gigabyte for deployment in large scale storage systems. On the other hand, a client class (low-end) SSD uses traditional serial ATA (SATA) interface and may employ MLC flash memory which is cheaper per gigabyte than SLC. However, the write endurance of MLC is an order of magnitude less than SLC, in the 10K-30K range [2].

Figure 1.1 shows the different classes of commercial SSDs with their cost and reliability. Each point shows cost per gigabyte (lower is better) on the y-axis and device writes per day (DWPD, higher is better) of recent SSDs from various vendors on the x-axis . 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 high-end SLC SSD as a cache and low-end MLC SSD as backend storage [47]. These systems try to improve performance at a lower

[2] 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 [18]. Phase change memory (PCM) has 10^8-10^{10} write endurance which is two orders of magnitude larger than SLC [32].

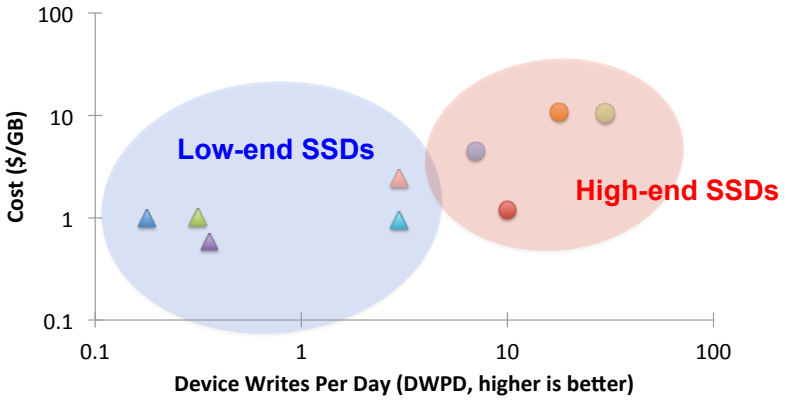


Figure 1.1: Different classes of SSDs

cost per byte. Not much work has been done in understanding the data lifetimes in such arrays. While the high-end SSD 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 of high-end SSD. 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.

In Section 4, we show that the high-end SSD cache can wear out faster than the low-end SSD main storage, thus, balancing the performance and lifetime of a mixed SSD array is important. In that section, we propose a sampling based approach to minimize the latency and maximize the lifetime of the mixed SSD array at the same

time.

This work focuses its attention on the reliability of a wide range of SSD storage systems including a single SSD device in Section 2, an SSD array at system level in Section 3, and a mixed SSD array in Section 4. Section 5 introduces the related work and Section 6 concludes this dissertation.

2. DEVICE LEVEL PROTECTION SCHEMES*

We categorize data protection schemes for Solid State Drives (SSDs) into two levels: device level protection and system level protection. In this section, we explore the relationship between the device level protection schemes such as error correction codes (ECCs) and the reliability of SSD. This section makes the following significant contributions:

- We provide a model for analyzing an SSD, taking variable bit error rate and the write amplification from the data protection schemes into account.
- We show that write amplification from ECC recovery and garbage collection can contribute up to a loss of 50% of data lifetime in an SSD.
- We propose a technique to reduce the write amplification from ECC recovery that improves the data lifetime significantly by up to 40%.

Section 2.1 introduces various device level protection schemes for SSDs. Section 2.2 builds reliability model of an SSD considering these protection schemes. Section 2.3 evaluates the reliability model. We show that write amplification from frequent ECC recovery results in less data lifetime in an SSD. Section 2.4 proposes a novel protection scheme to reduce the write amplification from frequent ECC recovery.

2.1 Device level protection schemes

A number of techniques have been developed to protect SSDs at the device level. In this section, widely used methods and their write amplifications are discussed.

* Part of this section is reprinted with permission from Sangwhan Moon and A.L.N. Reddy, Write Amplification due to ECC on Flash Memory or Leave those Bit Errors Alone, Mass Storage Systems and Technologies, 2012 IEEE 28th Symposium on, April 2012, ©2012 IEEE.

2.1.1 Error correction codes (ECCs)

ECC encodes data into check bits, and the encoded data is exploited to detect and correct errors in the data. The number of detectable and correctable errors is highly dependent on the complexity of the employed ECC. Since recent multi level cell (MLC) flash memory is prone to more errors than single level cell (SLC) flash, it requires stronger multi-bit correcting ECC like BCH code or Reed-Solomon code instead of single error correctable double error detectable (SECDED) Hamming code which is widely used in SLC. The storage and calculation overhead of ECC depends on the level of protection that is desired.

2.1.2 Scrubbing

ECC can be used to correct/detect errors only when data is accessed. In order to protect data that may not be frequently accessed in normal workloads, data on the SSD are actively scanned and errors found are *scrubbed*. Data scrubbing [49, 54] can be done during the idle periods of the SSD. Scrubbing rate can be either constant or exponentially distributed. In general, a large portion of data on the SSD are cold data and this makes scrubbing essential for data protection though it consumes energy for scanning the device.

2.1.3 Wear-leveling and garbage collection

The SSD blocks have to be erased before they can be rewritten. If blocks are overwritten with new data, hot blocks will wear out some locations faster than the rest of the device. The FTL tries to spread the writes over the entire device such that all the blocks wear out uniformly, in order to increase the lifetime of the device. Erase operations of SSD operate in units of a block, for example, 512KB, while write operations are done in units of a page, 4KB. Consequently, a number of valid pages

alive in a block have to be moved to another block before an erase. This recycling process is called *garbage collection*.

2.1.4 Write amplification

The protection schemes discussed in this section generate undesirable additional writes. This write amplification has been a just minor overhead to HDDs. On the other hand, for SSDs, the write amplification severely degrades reliability of devices, since the reliability is highly dependent on the number of writes issued to the SSD. Main sources of the write amplification are discussed in this section.

2.1.4.1 Garbage collection

NAND flash memory is typically written in a unit of a page, 4 KB, and erased in a unit of a block, e.g., 512 KB. It does not support in-place update and accumulates writes in a log-like manner. In such a log structured system, internal fragmentation and the garbage collection process to tidy the fragmented data are inevitable. The garbage collection process moves valid pages from one place to another place and this results in increasing the number of writes issued to the device. Write amplification due to garbage collection is dependent on the space utilization of the SSD. When the SSD is nearly full, garbage collection initiates quicker and results in being less efficient since a larger fraction of the blocks are still live.

Different algorithms are employed for garbage collection. These algorithms try to minimize the write amplification, for example, through selection of appropriate blocks with the least number of live pages and postponing garbage collection as long as possible. Write amplification due to garbage collection is strongly dependent on the space utilization of the SSD [24]. When the SSD is nearly full, garbage collection initiates quicker and results in being less efficient since a larger fraction of the blocks are still live.

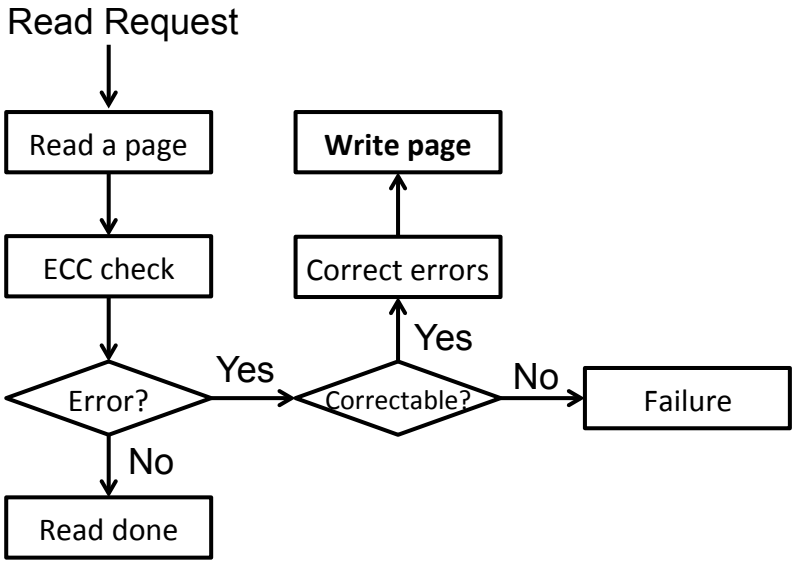


Figure 2.1: Write amplification from recovery process

2.1.4.2 Recovery process

Data recovery can be initiated due to ECC level error detection. In most of the recovery processes, at least one write is required to write corrected page back to the device. Figure 2.1 describes the process of write amplification from recovery process. This write amplification is inevitable and the number of amplified writes is highly dependent on page inspection rate. The term write amplification is known to be caused only by writes; however, we show that reads also lead to write amplification. This has a significant effect on the SSD's lifetime in modern computing environment where reads can be dominant compared to writes.

Figure 2.2 compares the traditional point of view of write amplification on the left side, where only writes amplifies writes, to our point of view of write amplification on the right side, where both reads and writes contribute to write amplification. In the figure, w_{ecc} represents the write amplification from ECC recovery process, and f_{ecc} is the fraction of the reads with an error detected by the ECC.

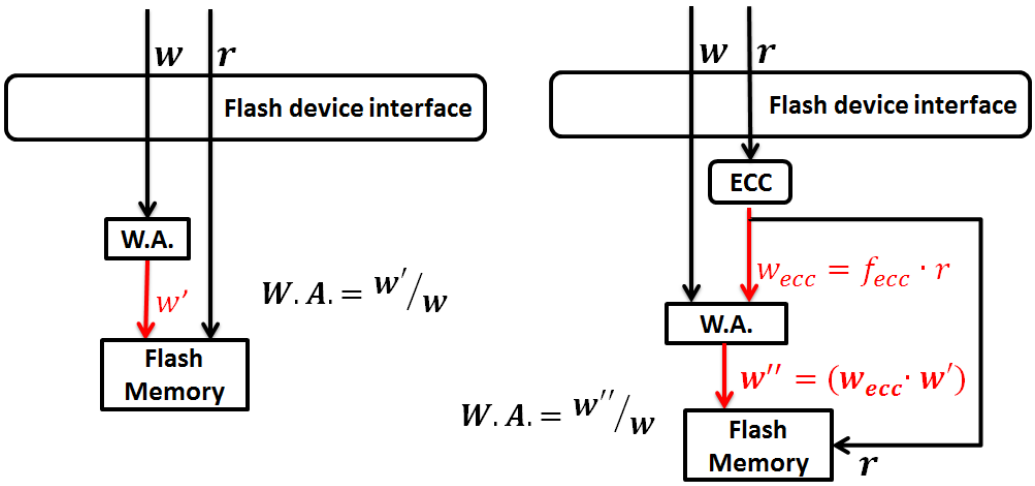


Figure 2.2: Write amplification from different point of views

ECC can correct a number of errors in a page simultaneously and it results in only one write. For instance, fixing a bit error in a page takes one redundant write, while fixing ten bit errors in a page also needs one additional write. Therefore, intensive recovery makes an SSD robust as well as it amplifies writes and hurts the device, making the design of a recovery process challenging.

Read-after-write mechanism is a popular technique to protect an SSD from write errors. Whenever write operation is done to a page, correctness of write is confirmed by reading the page immediately after write. It rewrites the page if write errors are detected in the page which leads to further write amplification.

2.2 Reliability model

We build a reliability model and analyze the various factors affecting the lifetime of an SSD. While there have been many studies on the reliability analysis of SSDs, to the best of our knowledge our work here provides the first model for SSDs considering write amplification. In this section, we assume that workload is random and uniformly distributed over the entire device such that the page access rate is

Error Type	A	B
Read disturb (per 1-1,000 reads)	7.73e-7	2.17e-4
Data retention (per month)	3.30e-6	1.83e-4
Write error (per write)	1.09e-7	3.01e-4

Table 2.1: Raw bit error rate $\lambda(x) = A \cdot e^{(B \cdot x)}$

constant on an average.

2.2.1 Raw bit error rate

According to many studies [5, 19, 58, 61, 62] on the behavior of flash memory, there are different sources of bit errors: read disturb, data retention failure, and write failure (write error). These studies model the bit error rate as an exponential function of the number of program/erase cycles (P/E cycles) the cell has gone through. We start with the assumption that bit errors are independent and their probabilities are exponentially distributed, and then we employ the data from the measurement study of [58] to model the rate of change of bit error rate. Table 2.1 shows the employed model. $\lambda(x)$ is the raw bit error rate at x P/E cycles.

Write error is immediately recovered by read-after-write mechanism of SSD; however, the recovery process requires extra write operation to write corrected data back which wears out the SSD faster. In this section, we assume that read-after-write scheme is not operational if it is not specified.

Read disturb error rate is a function of both P/E cycles and the number of reads on neighbors [58]. The number of read operations done to neighbors, k, can be estimated as a function of read ratio in the workload (R) and the number of neighbor pages (n):

$$
\begin{aligned}
k =& \sum_{i=0}^{\infty} p_i \cdot E(i, R, n) \\
=& n \cdot R/(1 - R^2)
\end{aligned}
\tag{2.1}
$$

where p_i is the probability of being read i times repeatedly without updating the target page, and $E(i, R, n)$ is the expected number of valid pages when the target page is read i times. By the time a page is accessed for the first time, $R \cdot n$ neighboring pages are expected to be read. A neighboring page that is written will be stored elsewhere and that physical neighbor becomes invalid. If the second access to a page is also a read (with a probability R^2), it is expected $R^2 \cdot R \cdot n$ neighboring pages would be read in that time. Likewise, we can estimate the number of additional reads when the page is read i times repeatedly. In the equation, the probability of a page being read i times is $p_i = R^i$, and the expected number of additional read operations done to neighbor pages when the page is read i times is $E(i, R, n) = R \cdot E(i - 1, R, n)$.

While read disturb error rate is expected to grow with the number of reads, the data in [58] shows that this error rate is nearly constant for the first 1,000 reads. Our evaluations are done within the range. For example, k is 218 when n is 127 (the number of pages per block is 128), and R is 0.75 (Read:Write=3:1)

2.2.2 Uncorrectable page error probability

ECC can correct errors up to a certain number of bits in a page. When higher number of bit errors occur within a page, the page fails or other techniques such as RAID have to provide data protection.

As shown in Figure 2.3, a canonical Markov model is typically used to build a statistical model of reliability of ECC for a page. In the figure, E is the number of correctable errors, S is the number of bits per page, $\lambda(x)$ is the bit error rate at x P/E cycles from the model of Section 2.2.1, and μ is the page recovery rate. The

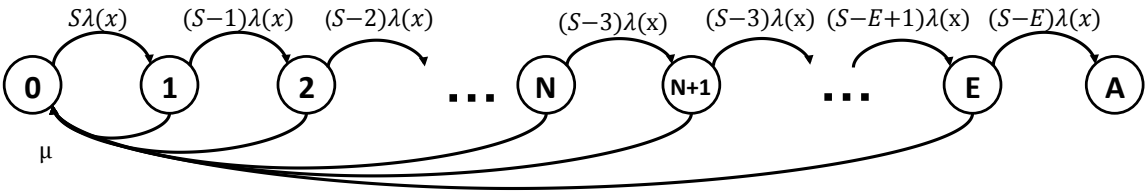


Figure 2.3: A Markov model of uncorrectable page error probability I

recovery rate is in fact the average page access rate, since the access interval time for the same page is much higher than ECC correction time. State A is the absorbing state where the number of errors exceed what ECC can correct and hence results in a page failure at the device level. From an analysis of the Markov model the probability of reaching the absorption state A can be obtained.

The bit error rate varies with the number of P/E cycles and it can be considered by modeling a series of Markov models with different bit error rates. We assume that the time varying nature of $\lambda(x)$ is relatively small because our raw bit error rate model increases error rate by about only 0.02% per P/E cycle. Also, we assume that the average time per P/E cycle is sufficiently large so that the Markov model converges in each P/E cycle. Since wear-leveling scatters write and erase operations over the whole device, an average time per P/E cycle is approximately the amount of time taken to write the whole device once. Under those assumptions, we can treat $\lambda(x)$ as constant at each P/E cycle x in the series.

We can estimate the uncorrectable page error rate at each P/E cycle using a steady state analysis of the Markov model. From the analysis, the probability of reaching the state A can be obtained.

2.2.3 Mean time to data loss

Since perfect wear-leveling is assumed, erases are uniformly spread out over the entire SSD; statistically, for an SSD of N pages, N writes to the device is equivalent to one write to each page on average. Therefore, the mean time to data loss (MTTDL) of the device is:

$$\mathrm{MTTDL_d} = \frac{\mathrm{MTTDL_p} \cdot \mathrm{N}}{\mathrm{N}} \tag{2.2}$$

where $\mathrm{MTTDL_p}$ is the MTTDL of a page shown in the following equation.

$$\mathrm{MTTDL_p} = \lim_{\mathrm{k}\to\infty} \sum_{\mathrm{j}=1}^{\mathrm{k}} \left(\mathrm{jg(j)} \prod_{\mathrm{i}=1}^{\mathrm{j}-1} (1 - \mathrm{g(i)}) \right) \tag{2.3}$$

2.2.4 Write amplification

Write amplifications are caused by garbage collection, recovery process and read-after-write which are denoted as α_{gc}, α_{rcv} and α_{raw}. These write amplifications increase the bit error rate and hence increase the uncorrectable page error probability and impact the data lifetime.

α_{gc} is dependent on space utilization and the range of it is estimated to range from 1.0 to 4.9 according to a recent study in [24].

α_{rcv} is estimated as

$$\alpha_{rcv} = \left(\mu \sum_{i=1}^{N} P_i \right) / w \tag{2.4}$$

where μ is the page recovery rate, w is the average write workload to a page and P_i is the probability of staying at state i of Markov model in Figure 2.3 in Section 2.2.2.

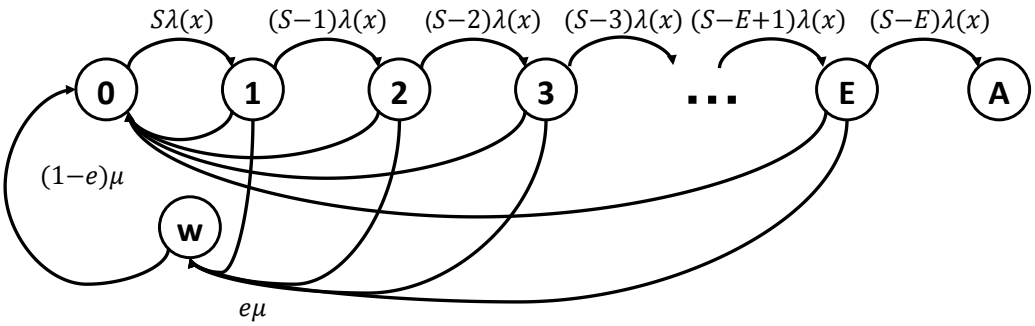


Figure 2.4: A Markov model of uncorrectable page error probability II

When write errors are considered and read-after-write mechanism is turned on, recovery process of read-after-write produces extra writes. Figure 2.4 shows the model of uncorrectable page error probability considering write errors and read-after-write mechanism.

In the figure, e is the probability of write error occurring. S is the number of bits per page, $\lambda(x)$ is the bit error rate at x P/E cycles from the model in Section 2.2.1, μ is the page recovery rate, E is the number of correctable bits by ECC and A represents an absorbing state when errors cannot be recovered by ECC. Write amplification from read-after-write α_{raw} is shown in the following equation.

$$\alpha_{raw} = \left(e \cdot \mu \cdot \sum_{i=1}^{N} P_i \right) / w \tag{2.5}$$

We assume that the sources of write amplification work independently and overall write amplification is $\alpha = \alpha_{gc} \cdot \alpha_{rcv} \cdot \alpha_{raw}$. In practice, they do not work independently, for example, while moving live pages for garbage collection, errors in the pages can be detected and corrected. We consider their dependency in Section 3.

The write amplifications are emulated by changing sampling of error probability function. Let $g(x)$ be an uncorrectable page error probability after x P/E cycles and

$\alpha = 1.3$, then new error probability $g'(x) = g(1.3x)$.

2.3 Evaluation

Various aspects of the expected lifetime of SSD are explored in this section. This includes lifetime changes under the consideration of ECC, scrubbing, device usage patterns such as space utilization and hot/cold dichotomy. As mentioned above, we exploited bit error rate mainly from [58], specifically 3x nm memory with the employment of 61-bit correctable (out of 4KB page) error correction code.

2.3.1 Practical issues

We built a reliability model for SSD in Section 2.2. When we evaluate this model, we are confronted with many practical issues.

2.3.1.1 Relative MTTDL

We evaluate the lifetime of SSD and normalize by the lifetime of reference device. For example, if the reference device's MTTDL is 1.5 years, and the evaluated device's MTTDL is 0.5 years, the normalized lifetime of the device is given as 0.33. We expect this to focus our attention on the relative strengths of various protection schemes, rather than the absolute lifetimes which are dependent on memory vendor. The reference device is set to be protected by 61-bit correctable ECC without any degradation from write amplification.

2.3.1.2 Environment

As we show later, the results are dependent on the space and throughput utilization of SSD. Unless otherwise mentioned, the results assume a space utilization of 0.5 and throughput utilization of 0.5. Specifically, we assume that capacity is 80GB, maximum throughput is about 120MB/s and read:write ratio is 3:1. The P/E cycle in time is about 80 minutes per page on average under the considered workload. Note

read:write	5000	10000	15000	20000	25000	30000
1:1	1.0302	1.0839	1.2125	1.4430	1.7011	1.8738
3:1	1.0308	1.0889	1.2475	1.6287	2.3165	3.0930
5:1	1.0309	1.0899	1.2560	1.6862	2.5968	3.9032
7:1	1.0310	1.0904	1.2598	1.7142	2.7571	4.4806
9:1	1.0310	1.0906	1.2619	1.7308	2.8609	4.9130

Table 2.2: Write amplification from ECC recovery at different P/E cycles

that these are arbitrary choices, and we vary some of these to study the sensitivity of results to these values. We assume that the TRIM command is being issued by the operating system whenever data is deleted on the device.

2.3.1.3 Sources of failure

Sources of bit errors evaluated in this section are read disturb and retention failure. Write failure errors are also evaluated, however, the result is not included if not specified since read-after-write mechanism can recover the write failure immediately and we find that its effect on lifetime of SSD is negligibly small. Other sources such as whole device failure and software errors are modeled in Section 3.

2.3.2 Sources of degradation

We look at the relative contribution to the loss of lifetime resulting from the various sources of write amplification.

2.3.2.1 ECC recovery

Table 2.2 reveals the relationship between workload and write amplification from ECC recovery. The amount of write workload is fixed and the read workload is varied based on the read:write ratio. The higher read:write ratio implies higher recovery rate from higher read rates and, hence, higher number of writes for recovery.

To analyze how much write amplification is harmful to lifetime, we evaluated the

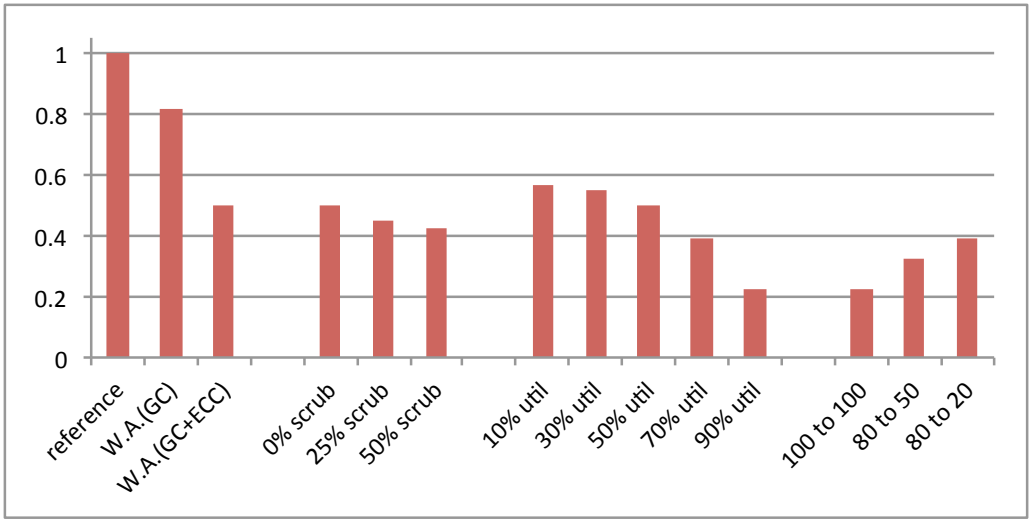


Figure 2.5: Impact of various factors on lifetime (Relative MTTDL)

lifetime of SSD. Figure 2.5 shows the impact of various factors on lifetime. When the write amplification from garbage collection is considered, the lifetime of a single SSD decreases by 20%. The write amplification from ECC recovery contributes to a loss of additional 30% of the lifetime. This shows the importance of devising data protection techniques that are less write intensive.

2.3.2.2 Scrubbing

Scrubbing is well-known to be useful for seldom referred blocks. For HDDs, scrubbing is beneficial if it works in the background and does not interfere with foreground I/O requests. However, for SSDs, intensive scrubbing may be harmful due to write amplification. The impact of scrubbing on lifetime of SSD is shown in Figure 2.5. We have investigated what exactly causes this degradation, and found that page error rate is increased by write amplification from frequent ECC recovery than it is reduced by higher recovery rate. Though our evaluation cannot find benefit from scrubbing SSDs in this section, we cannot argue that scrubbing is always harmful.

2.3.2.3 Garbage collection (space utilization)

Write amplification from garbage collection is highly dependent on space utilization and hot/cold distribution of data. Their impact on the lifetime of SSD is discussed here.

The amount of write amplification is strongly related to space utilization. Earlier work has studied the relationship between write amplification from garbage collection and space utilization [24]. We exploit this relationship to see the impact of space utilization on lifetime of SSD.

Figure 2.5 shows the change of lifetime as a function of space utilization. As we expected, lifetime is less at higher utilizations due to garbage collection. For example, increasing space utilization from 10% to 50% (increasing the amount of data by 5 times) results in 10% of loss in data lifetime, while increasing utilization from 50% to 90% (increasing the amount of data by 1.8 times) results in about 62% of loss in data lifetime. It is possibly better to employ new devices and keep the utilization at 50% even though more number of devices results in higher error probability, which is proportional to the number of devices. It is better to employ new devices and keep utilizations lower to maintain high data lifetimes. This observation motivates the study in Section 3 which investigates the relationship between space overhead of parity protection and the number of SSDs in an SSD array.

2.3.2.4 Read-after-write

Write failure (write error), in addition to read disturb error and data retention error, is one of three major sources of bit errors in SSD; however, write errors can be simply detected and corrected by read-after-write mechanism as described in Section 2.2.4. The Markov model in Figure 2.4 is evaluated in Figure 2.6. It shows that write amplification due to read-after-write reduces lifetime by less than 3% when write error

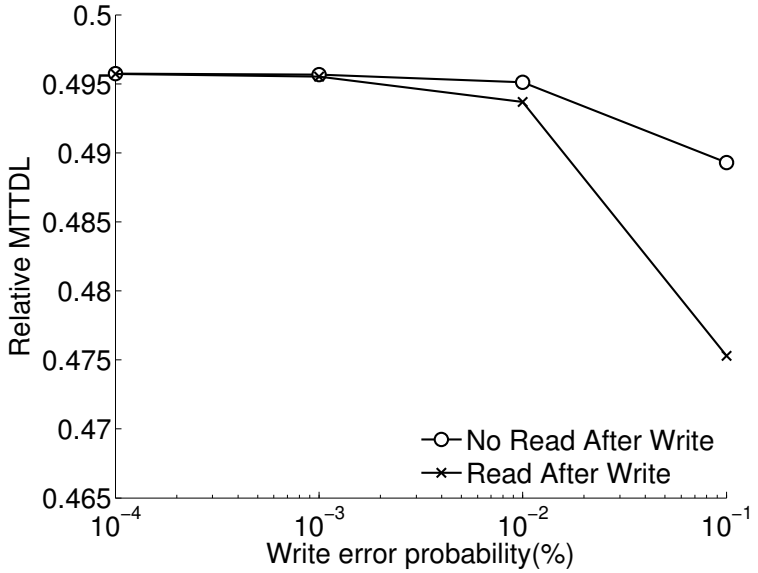


Figure 2.6: Read-after-write mechanism

probability is about 0.1% which is impractically high. In this section, write error is not considered and read-after-write mechanism is assumed to be turned off.

2.3.2.5 Hot/cold dichotomy

In real systems, data exhibit hot and cold behavior, where a few blocks receive significant fraction of requests while other blocks receive a smaller fraction of requests. We assumed that wear-leveling works well enough such that access rates for hot and cold blocks are fair over long time scales; however, in a short timescales, hot blocks are more likely to be overwritten, which results in more page invalidations before they are reclaimed. Due to this behavior of hot and cold blocks, garbage collection can gather invalid pages more efficiently by reclaiming hot blocks more often and this results in less write amplification.

We employed data about write amplification from the work in [9] which reveals the relationship of write amplification to hotness of data. The evaluation result is

shown in Figure 2.5. In the figure, x to y means x% of throughput is concentrated on y% of space. Space utilization is set to 90% in this evaluation since the work in [9] provides results of 85% and 90% of space utilization. The result shows that hot blocks increase lifetime of SSD because of more efficient garbage collection.

2.4 Threshold based ECC

Our analysis showed that, given sufficient workload intensity, most errors within a page are corrected rapidly so that only a few bit errors accumulate before an ECC action corrects and writes that page back to the SSD. Table 2.3 shows the probability distribution of the number of bit errors (n) when a page is accessed. When bit errors are found ($1 \leq n$), the bit errors are corrected by ECC recovery. For example, at 25,000 P/E cycles, about 83% (0.5824 / (1.0 - 0.3013)) of ECC recovery corrects less than or equal to 5 bit errors in 4 KB page while only 17% of ECC recovery corrects more than 5 bit errors in a page. As pointed out earlier, the writes from ECC can decrease data lifetimes. To avoid the disadvantage of extra writes of ECC recovery, we propose threshold based ECC correction. In this scheme, data is corrected by ECC and the correct data is returned to the requesting process, but the corrected data is not written back to the SSD until the number of bit errors within the page reach a threshold.

In other words, new scheme leaves page errors uncorrected on the device until the number of bit errors exceed some threshold. For instance, with 200MB/s workload at a 80GB SSD, 10^{-7} bit error rate means about one bit error is detected/corrected on average within a 4KB page when that page is accessed. Threshold based ECC correction waits until 10, 20, or a certain number of bit errors are accumulated, and then corrects the data to reduce the number of extra writes. As Table 2.3 shows, most of the bit errors are corrected before a page accumulates a significant number of

n	5000	10000	15000	20000	25000
$n = 0$	0.9705	0.9176	0.7904	0.5605	0.3013
$n = 1$	0.0286	0.0756	0.1657	0.2463	0.2105
$1 \leq n \leq 3$	0.0295	0.0823	0.2077	0.4022	0.4604
$1 \leq n \leq 5$	0.0295	0.0824	0.2096	0.4323	0.5824
$n \geq 5$	6.57e-10	3.12e-7	8.50e-5	0.0072	0.1163

Table 2.3: Probability distribution of the number of bit errors (n)

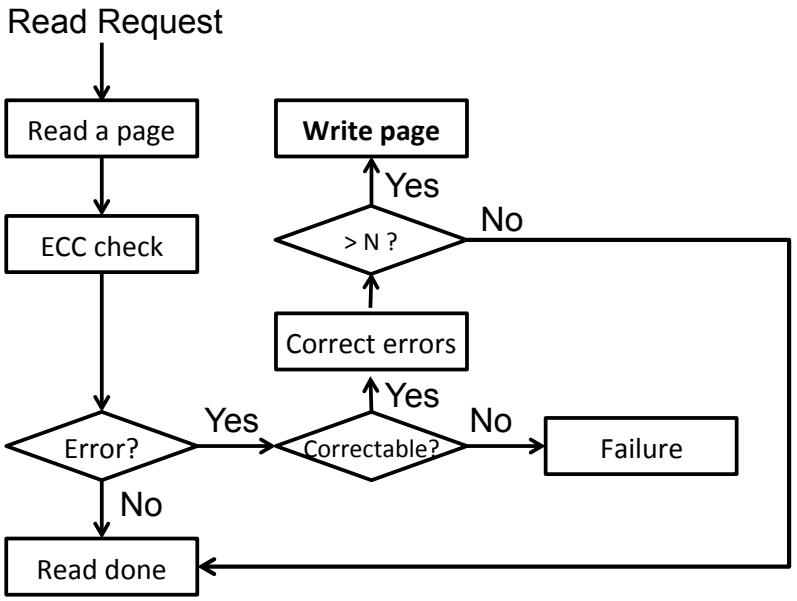


Figure 2.7: Write amplification and threshold based ECC

errors. Our new scheme significantly reduces write amplification from ECC recovery, since considerable portion of write amplification comes from repairing one or few errors rather than many errors at once.

Figure 2.7 describes how threshold based ECC works. For read requests to a page, threshold based ECC delays writing the corrected data on to the device until bit errors within the page reach a threshold of N.

Figure 2.8 shows a Markov model of a page when threshold based ECC is em-

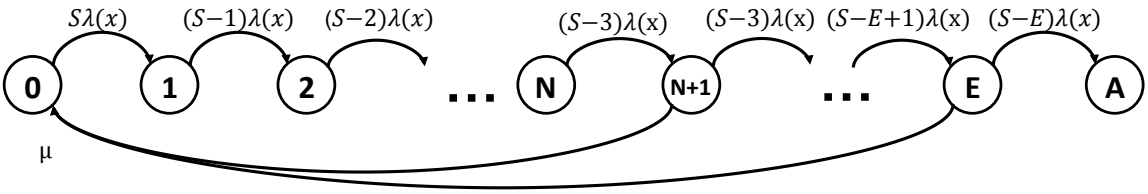


Figure 2.8: A Markov model of uncorrectable page error probability III

Threshold (the number of bits)	0	6	18	30	42	54
Relative MTTDL	0.496	0.614	0.671	0.694	0.702	0.696

Table 2.4: Relative MTTDL of threshold based ECC

ployed.

In the figure, S is the number of bits per page, $\lambda(x)$ is the bit error rate at x P/E cycles from the model in Section 2.2.1, μ is the page recovery rate, N is the threshold, E is the number of correctable bits by ECC and A is a state when errors cannot be recovered by ECC.

Table 2.4 shows an evaluation of threshold based ECC. A threshold of x means that ECC related writes are skipped until the accumulated bit errors in the page reach x out of 61 correctable bits by ECC within the page. Since leaving too many errors produces drastic growth of page error rate, there exists an optimal number for the threshold. Table 2.4 shows that leaving about 70% of correctable errors, 42 bit errors in a 4KB page, with 61-bit ECC is optimal. The optimal threshold is expected to be a function of the bit error rate and the page access rate and the power of the employed ECC. Deriving an optimal threshold as a function of these variables remains an open problem.

In this section, we made a number of assumptions to make the analysis tractable. Some of these assumptions are relaxed in Section 3.

3. SYSTEM LEVEL PROTECTION SCHEMES I: PARITY PROTECTION*

In the previous section, we explored different aspects of device level protection schemes such as ECC, wear leveling, and garbage collection. In this section, we will mostly focus on one of the system level protection schemes: parity protection such as RAID5 and mirroring.

We focus our attention on the reliability of an array of SSDs. We explore the relationship between parity protection and the lifetime of an SSD array. We make the following significant contributions:

- We analyze the lifetime of SSD taking benefits and drawbacks of parity protection into account.
- The results from our analytical model show that RAID5 can be less reliable than striping with a small number of devices because of write amplification.
- Our results show that RAID5 is effective at improving lifetime in large arrays and in small arrays with lower space utilizations and less intensive workloads.
- We explore device parameters that can improve lifetime and find effective configurations improving data lifetimes in SSD arrays.

Section 3.1 provides a background of system level parity protection and write amplification of SSDs. Section 3.2 explores SSD based RAID. Section 3.3 builds a reliability model of SSD based RAID. Section 3.4 describes our simulator to estimate the lifetime of an SSD. Section 3.5 presents results from our analytic model.

* Part of this section is reprinted with permission from Sangwhan Moon and A.L.N. Reddy, Does RAID Improve Lifetime of SSD Arrays?, in ACM Transactions on Storage (accepted), ©2015 ACM.

3.1 System level protection schemes

In many cases, device level protections are not enough to protect data. For example, when the number of bit errors exceeds the number of correctable bit errors using ECC, data in a page may be lost without additional protection mechanisms. A device can fail due to other reasons such as the failure of device attachment hardware. In this section, we call the former as a page error and the latter as a device failure. In order to protect SSD arrays against device failures, system level parity protection is employed.

3.1.1 Parity Protection

RAID5 is popular as it spreads workload well across all the devices in the device array with relatively small space overhead for parity protection. The group of data blocks and the corresponding parity block is called a page group. RAID5 is resilient to one device failure or one page error in a page group.

Mirroring is another popular technique to provide data protection at the system level. Two or more copies of the data are stored such that a device level failure does not lead to data loss unless the original and all the replicas are corrupted before the recovery from a failure is completed. When the original data is updated, the replicas have to be updated as well. Read operations can be issued to either the original or the replicas at the system level. When a device is corrupted, the replicas are used to recover the failed device.

3.1.2 Write amplification

Protection schemes for SSD often require additional writes and those writes in turn reduce the reliability of SSDs. Since higher write amplification can reduce the lifetime of SSD severely, protection schemes should be configured carefully to

maximize the lifetime improvement while minimizing write amplification. Write amplification severely degrades reliability, since the reliability is highly dependent on the number of writes done at the SSDs. Main sources of the write amplification are discussed in this section.

3.1.2.1 Recovery process

As discussed in Section 2.1.4.2 in Section 2, in most of the recovery processes, at least one write is required to write a corrected page. ECC can correct a number of errors in a page simultaneously with one write. For instance, fixing a bit error in a page takes one redundant write, while fixing ten bit errors in a page also needs one additional write. It is noted that ECC based write amplification is a function of read workload unlike other sources of write amplification.

3.1.2.2 Garbage collection

As discussed in Section 2.1.4.1 in Section 2, garbage collection amplifies the number of writes in SSDs and the write amplification due to garbage collection is dependent on the space utilization of the SSD. Recent study [10] reveals that the efficiency of garbage collection is also dependent on the hotness of data. Since hot data tends to be more frequently invalidated, garbage collection can be more efficient when hot workload concentrates on a small portion of data.

3.2 SSD based RAID

When data is distributed across many devices, the failure of any device can lead to data loss. As the number of devices increases, the failure rate increases and lifetime decreases, without additional measures for protecting data. To overcome this limitation, RAID is widely used in storage systems. Due to physical characteristics of SSD, RAID architecture needs to be modified for SSD based storage systems. Our

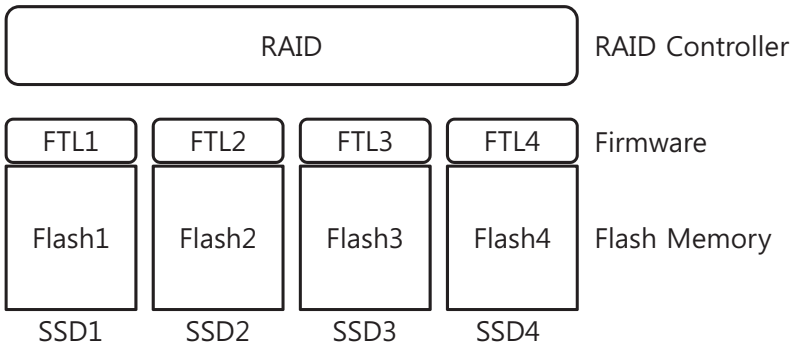


Figure 3.1: RAID architecture of an SSD array

analysis is based on an architecture shown in Figure 3.1 where a RAID controller operates on top of a number of SSDs. As a result, the set of pages within a page group have a constant logical address (before FTL translation) within the device. As the pages are written, the actual physical addresses of pages within the device change because of FTL translation. However, this does not impact the membership of the page group, based on the logical addresses. When the number of page errors in a page group exceeds the number of correctable page errors by RAID, the page group fails and the storage system loses data.

In RAID5, for a small write, the data block and the parity block need to be updated, potentially resulting in a write amplification factor of 2. However, when a large write that spans a number of devices is issued, the parity block can be updated once for updating $N-1$ data blocks, where N is the number of devices in the device array, resulting in a write amplification factor of $N/(N-1)$. Depending on the workload mixture of small and large writes, the write amplification will be between $N/(N-1)$ and 2. In a mirroring system, the write amplification is 2 regardless of the size of the write request.

Parity pages increase space utilization. Suppose that 120 GB of data is stored in

four 80 GB SSDs, RAID5 stores 30 GB of data and 10 GB of parity in each device while striping stores only 30 GB data per device. The increased amount of space utilization results in less efficient garbage collection and higher write amplification, which decreases the lifetime of SSDs.

It is possible to write to the array one stripe at a time to reduce parity update costs. However, this requires garbage collection to consider one stripe at a time, and not a block at a time within a device, requiring coordination across the devices. While this is possible when a RAID controller is built on top of raw flash memory, this is not feasible when building RAID on top of SSDs. In this section, we consider an array built on top of SSDs.

Additional measures can be employed to improve reliability. In order to protect data that may not be frequently accessed in normal workloads, data on the SSDs are actively scanned and errors found are *scrubbed*. Data scrubbing [49, 54] can be done during the idle periods of the SSD array. Scrubbing rate can be either constant or exponentially distributed. When a large portion of data on the SSD are cold data, scrubbing is essential for data protection though it consumes energy for scanning the SSD.

3.3 Lifetime model

In this section, we show a lifetime model based on a Markov model of Section 2. A number of symbols used in our model are shown in Table 3.1.

3.3.1 Uncorrectable page error rate

We employ a Markov model in Figure 3.2 to build a model of reliability of a page. In the figure, labels of states are the number of bit errors in a page, and K is the number of correctable bit errors by ECC. Bit errors accumulate within a page at a rate of $(S - i) \cdot \lambda(x)$, where S is the number of bits per page, i is the number of

Symbols	Description (G.C. stands for garbage collection)
k	the expected number of reads done to neighboring pages
R	the portion of reads in workload
W	the portion of writes in workload
S	the number of bits in a page
K	the number of correctable bit errors by ECC (per page)
$\lambda(x)$	raw bit error rate at x P/E cycles
$g(i, x)$	page access rate by G.C. (page has i bit errors at x P/E cycles)
μ	page access rate by workload
μ_r	page access rate by read workload
μ_w	page access rate by write workload
F_E	(state) page error: ECC cannot recover the corrupted page
N	the number of devices in an SSD array
$\nu(x)$	uncorrectable page error rate at x P/E cycles
$\nu^{(i)}(x)$	$\nu(x)$ of the i_{th} replaced device.
d	device failure rate
ξ	page group recovery rate
η	device recovery rate
v_i	(state) a page is corrupted (i: device replacement count)
d_i	(state) a device is corrupted (i: device replacement count)
F_R	(state) data loss: RAID cannot recover the system
$\sigma(x)$	data loss rate at system level
$p(x)$	the probability of data loss at x P/E cycles, induced by $\sigma(x)$
w	the number of writes issued per page per second
t_w	the average time to write whole storage system once
P_i	the probability of staying at state i in a Markov model
$P(x)$	the probability of seeing the first data loss at x P/E cycles
u_s	device space utilization
T	the number of pages in a device
a	over-provisioning factor, $a = 1/u_s$
B	the number of pages per block
M_v	the number of blocks full of valid pages found in G.C.
m_v	the probability of a block full of valid pages found in G.C.
A_F	the write amplification from G.C. with FIFO policy
$A_{F'}$	the write amplification from G.C. with modified FIFO policy
A_G	the write amplification from G.C. with Greedy policy
$A_F(a, r, f)$	A_F, a:over-provisioning (a), r:hot workload, f:hot space
$A_G(a, r, f)$	A_G, a:over-provisioning (a), r:hot workload, f:hot space
q	average write length in pages
W	exponential random variable of write length with average q

Table 3.1: List of symbols used in analysis models

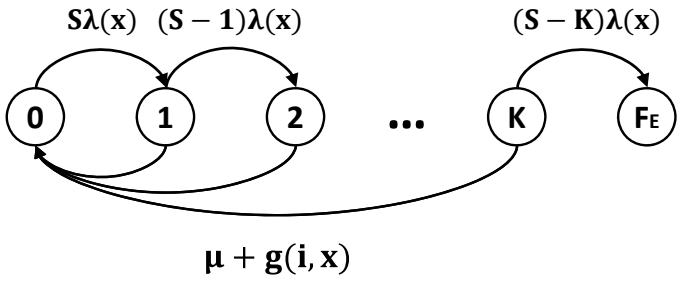


Figure 3.2: A Markov model of page error rate

errors accumulated in a page, and $\lambda(x)$ is the raw bit error rate at x P/E cycles. Once a state gets to the state F_E, bit errors cannot be recovered by ECC and it results in page error. ECC detects and corrects bit errors at page recovery rate μ. The recovery rate μ is a function of page access time (or error detection time) t_1 and page recovery time t_2:

$$\mu = \frac{1}{t_1 + t_2} \tag{3.1}$$

Like the page model in Section 2, we employ a series of Markov models with different bit error rates. The same assumptions are valid; the time varying nature of $\lambda(x)$ is small enough and the average time per P/E cycle is sufficiently large such that the Markov model converges in each P/E cycle. Under those assumptions, we can treat $\lambda(x)$ as constant at each P/E cycle x in the series.

From a steady state analysis of the Markov model, the probability of reaching the state F_E can be obtained.

The write amplifications from various sources are not independent of each other. For example, when garbage collection process moves valid pages to empty space in another block, bit errors in the pages are corrected by ECC during the move operation. We take such dependencies into account in our model. Each state of

Markov model in Figure 3.2 has its own probability of recovery by garbage collection. The garbage collection recovery rate $g(i, x)$ highly depends on garbage collection policy adopted.

3.3.2 Data loss rate

The reliability of an SSD array (RAID5) can be modeled through a Markov model shown in Figure 3.3.

There are mainly two sources of page error. Bit errors accumulate and result in a page error. The entire device can fail by various sources such as attached hardware error, software error, and human error. The device failure leads to the failure of all the pages in the device.

Figure 3.3 shows the Markov model for a RAID5 system. Two sources of failure are considered in this model. Bit errors accumulate and result in a page error. The page error results in data loss unless detected and corrected before another page error in the same page group or a device failure occurs. The device failure can lead to the failure of all the pages in the device. The device failure results in data loss when a page error in another device or another device failure occurs.

This is the list of cases when permanent data loss can occur:

1. Page error + page error in *the same page group.*
2. *Any* page error + device failure.
3. Device failure + *any* page error in *another device.*
4. Device failure + another device failure.

In Figure 3.3, case (1) and (2) are considered in the upper chain (from state i to $\mathrm{v_i}$ and then $\mathrm{F_R}$) of the Markov model, and (3) and (4) are considered in the lower chain (from state i to $\mathrm{d_i}$ and then $\mathrm{F_R}$).

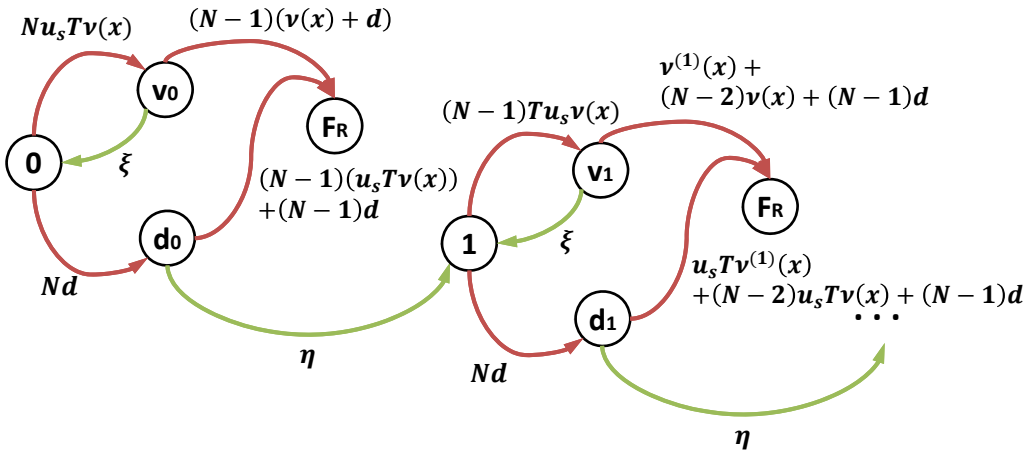


Figure 3.3: A Markov model of RAID5 system

It is noted that device failures are easier to detect and a recovery operation can be issued immediately soon after the failure. However, the device recovery time is considerably long. On the other hand, a page error is not detected until that page is read. However, a detected page error can be recovered quickly. Our analysis takes these issues into consideration with different value of ξ and η.

The reliability of SSD is highly dependent on the number of writes it has serviced. The Markov model in Figure 3.3 considers the number of device replaced after device failure recovery. The replaced devices have experienced relatively lower number of writes.

The data loss rate $\sigma(x)$ is the rate of reaching state F_R. The time series of these data loss rate σ(x) is integrated over time to obtain the expected time to data loss.

The data loss rate $\sigma(x) = u_s \cdot T \cdot v(x) + d$ for a single SSD. For N device striping, $\sigma(x) = N(u_s \cdot T \cdot v(x) + d)$. Under an assumption that each device pair is independent in a mirroring system, we can build a Markov model for the device pair, and then extend it to cover the entire system. The pair fails with a failure rate $\rho(x)$, and the mirroring system fails at the rate of $\sigma(x) = N/2 \cdot \rho(x)$.

$$\begin{aligned}
&\text{Single SSD}: \ \sigma(x) = u_s \cdot T \cdot v(x) + d \\
&\text{N device striping}: \ \sigma(x) = N(u_s T \cdot v(x) + d) \\
&\text{Mirroring}: \ \sigma(x) = N/2 \cdot \rho(x)
\end{aligned} \tag{3.2}$$

3.3.3 Mean time to data loss

We employ the mean time to data loss (MTTDL), which is one of the popular metrics in reliability analysis. The MTTDL is defined as the expected time to encounter the first data loss:

$$\begin{aligned}
MTTDL &= \left(\sum_{x=1}^{\infty} (x P_{DL}(x)) \right) \cdot t_w \\
&= \left(\sum_{x=1}^{\infty} \left(x p(x) \prod_{i=1}^{x-1} (1 - p(i)) \right) \right) \cdot t_w
\end{aligned} \tag{3.3}$$

In the equation, $p(i)$ is the probability of data loss at i P/E cycles such that the probability of seeing the first data loss at x P/E cycles $P_{DL}(x)$ is:

$$P_{DL} = p(x) \prod_{i=1}^{x-1} (1 - p(i)) \tag{3.4}$$

Thus, the MTTDL in Equation (3.3) is the sum of expectation of seeing the first data loss at a certain P/E cycle over the lifetime of an SSD array.

3.3.4 Write amplification

Write amplification can be caused by garbage collection, ECC recovery, and parity update which are denoted as $\alpha_{gc}, \alpha_{rcv}$, and α_{parity}.

3.3.4.1 Garbage collection

The write amplification from garbage collection α_{gc} relies on garbage collection policy. In this section, we consider two garbage collection policies: First In First Out

(FIFO) and Greedy. FIFO policy selects the least recently written block as a victim for garbage collection, while Greedy policy picks the block with the least number of valid pages as a candidate for garbage collection. We exploited the result from [10] to estimate garbage collection rate $g(i, x)$ which is dependent on the write workload.

3.3.4.1.1 FIFO This policy keeps a list of non-empty blocks in order of written time. We start from the simple case when the workload writes one page per write and it is random and uniformly distributed. The rate at which blocks are consumed is given by qA_F/B where q is average write length, A_F is the write amplification from garbage collection with FIFO policy, and B is the number of pages in a block. The write amplification A_F is the solution of the following equation. The details are described in [10].

$$A_F = \frac{1}{1 - (1 - \frac{q \cdot a}{T})^{\frac{T/B}{q \cdot A_F / B}}} \tag{3.5}$$

where T is the number of pages in a device and a is over-provisioning factor.

Real workload can write more than a page per operation, and this generates consecutive pages which are more likely to be invalidated together. When we write a large file sequentially and delete the file at once, for example, many blocks will be full of invalidated pages and garbage collection can get a number of empty blocks, free of write amplification. On the other hand, a large file can result in a series of blocks filled with all valid pages. It is unrealistic that FIFO policy selects a victim block filled with valid pages and move all of the valid pages to somewhere in the SSD during garbage collection process. (This will not generate empty blocks at all.) We modify FIFO model to pass over the blocks with 100% valid pages. In the modified FIFO model, we assume that the write length follows exponential distribution with average write length q such that the probability of write length is more than x is

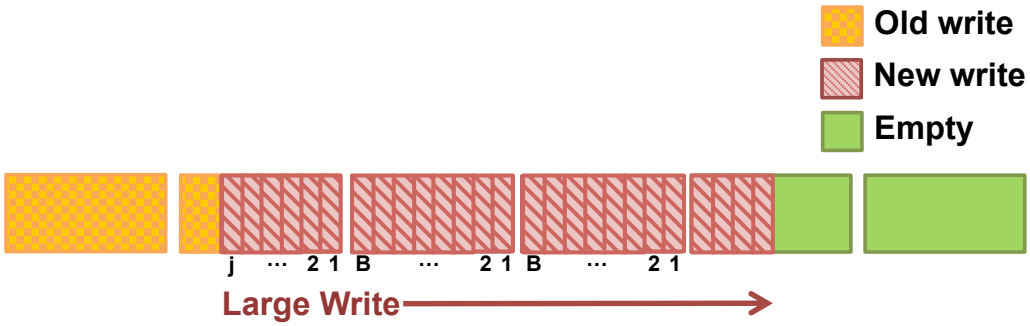


Figure 3.4: An example of a large write

$$P(W \geq x) = 1 - \frac{1}{q} e^{-\frac{x(N-1)}{q}} \tag{3.6}$$

Figure 3.4 shows an example of a large write when a write starts from page number j in the second block. In the figure, two blocks are fully written because of this one large write request.

As described in the figure, page number in a block starts from B and decreases to 1 as writes continue. When a write with size W starts at location j of a block, n blocks will be written fully with valid pages when $(n+1) \cdot B \geq W - j \geq nB$. The expected number of blocks filled only with valid pages is the sum of the expectation of n blocks being within a write.

$$\begin{aligned} &\sum_{n=1}^{\infty} n \cdot P((n+1) \cdot B + j \geq W \geq n \cdot B + j) \\ &= \sum_{i=1}^{\infty} P(W \geq i \cdot B + j) \end{aligned} \tag{3.7}$$

We assume that a write can start at any page with equal probability $1/B$. The estimated number of blocks full of valid pages is

$$M = \sum_{j=1}^{B} \frac{1}{B} \sum_{i=1}^{\infty} P(W \geq i \cdot B + j) \tag{3.8}$$

In a FIFO queue with blocks sorted by written time, blocks are moving at rate $q \cdot A_{F'}/B$ where $A_{F'}$ is the write amplification from garbage collection with modified FIFO policy. Then, surviving rate S of a block when it gets to the end of FIFO queue is

$$S = (1 - \frac{q \cdot a}{T})^{\frac{T/B}{q \cdot A_{F'}/B}} \tag{3.9}$$

And the number of blocks 100% filled with valid pages when garbage collection picks the blocks is given by

$$M_v = M \cdot S \tag{3.10}$$

And the probability of a block being filled with all valid pages is

$$m_v = M_v/(T/B) \tag{3.11}$$

Since M_v blocks are passed by garbage collection process, the write amplification from garbage collection is modified as

$$A_{F'} = \frac{1 - m_v}{1 - (1 - \frac{q \cdot a}{T})^{\frac{T/B}{q \cdot A_{F'}/B}}} \tag{3.12}$$

In this section, we call the modified FIFO as FIFO unless specified.

3.3.4.1.2 Greedy This policy chooses the victim block which has the least number of valid pages. It reduces the number of redundant writes for moving valid pages, while it requires additional data structures and operations to keep the information

on the number of valid pages in every block.

The write amplification from greedy garbage collection is estimated in Equation (3.13). In the equation, X_0 is the expected number of valid pages in the victim block.

$$\begin{aligned} X_0 &= \frac{1}{2} - \frac{2 \cdot B}{a} \cdot W(-(1 + \frac{1}{2 \cdot B}) \cdot a \cdot e^{-(1+\frac{1}{2 \cdot B}) \cdot a}) \\ A_G &= \frac{B}{B - (X_0 - 1)} \end{aligned} \tag{3.13}$$

Most workloads have nonuniform access patterns, where a significant fraction of the workload may concentrate on a small data space. Let fraction r of the workload access data in fraction f in space. Then the write amplification is given by

$$\begin{aligned} A_F(a, r, f) &= 1 + \frac{r}{e^{\frac{r}{f} \cdot \frac{a}{A}} - 1} + \frac{1 - r}{e^{\frac{1-r}{1-f} \cdot \frac{a}{A}} - 1} \\ A_G(a, r, f) &= \frac{A_F((1 + 1/2 \cdot B) \cdot a, r, f)}{1 + 1/2 \cdot B} \end{aligned} \tag{3.14}$$

The details of these equations are discussed in [10].

3.3.4.2 ECC recovery

In Section 2, we show that ECC recovery can be one of the sources of write amplification. When ECC detects and corrects bit errors in a page, the resulting write is an unintended write resulting from a read request. In Figure 3.2, the amount of the write amplification from ECC recovery is the ratio of the amount of write traffic moving the page from states with $1, 2, \cdots, K$ errors to a page with 0 bit errors to the actual write traffic issued to the device by the user. Equation (3.15) shows the write amplification.

$$\alpha_{rcv} = \left(\mu_r \cdot \sum_{i=1}^{K} P_i \right) / w \tag{3.15}$$

In Section 2, we have suggested threshold based ECC (TECC) to reduce the

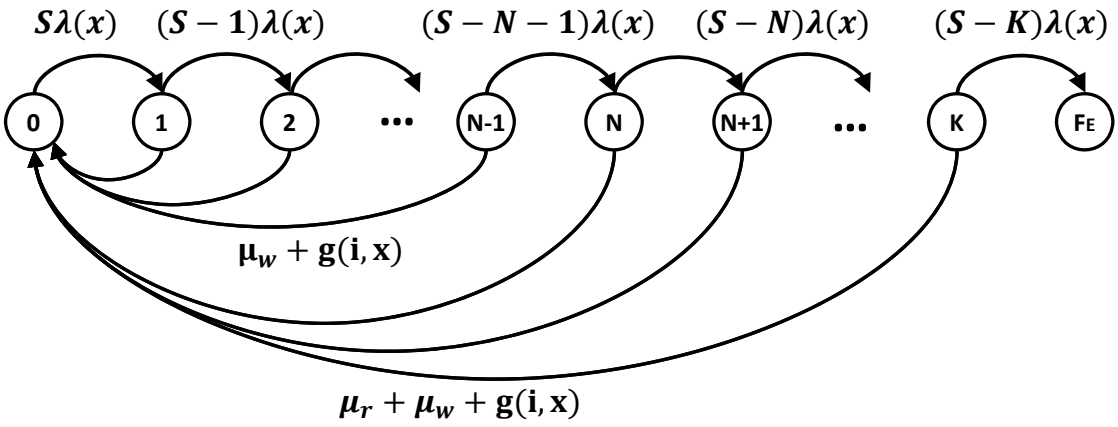


Figure 3.5: A Markov model of page error rate with TECC

write amplification from frequent ECC recovery by leaving bit errors in a page until it accumulates a certain number of bit errors. TECC can drastically reduce the write amplification from ECC recovery.

The Markov model of page error rate in Figure 3.2 is modified as Figure 3.5 when TECC is employed. In the model, the write amplification from ECC recovery is

$$\alpha_{rcv} = \left(\mu_r \cdot \sum_{i=N}^{K} P_i \right) / w \tag{3.16}$$

where N is a threshold and bit errors are left in storage media until the number of bit errors exceeds the threshold.

3.3.4.3 Parity update

As discussed in Section 3.2, the write amplification from parity update depends on workload characteristics and is somewhere in between $N/(N-1)$ and 2. The number of parity updates depends on an average on the length of write requests. Figure 3.6 shows an example of writing 6 pages from the 4th page in the page group 1 in an SSD array with 8 SSDs, which results in 2 parity page updates and the write

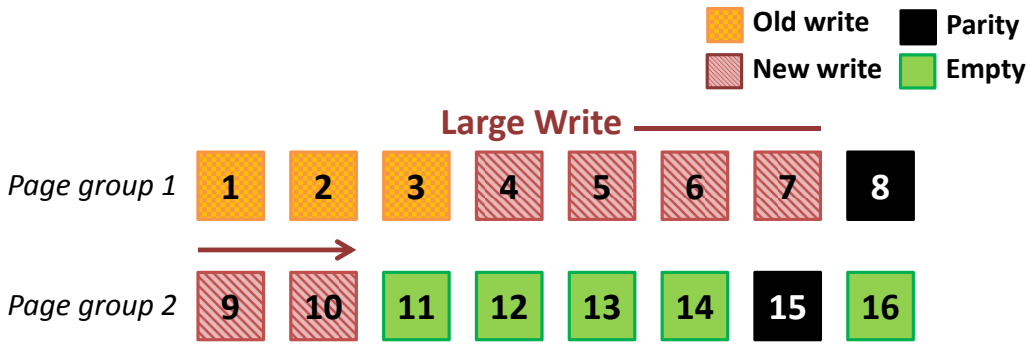


Figure 3.6: A large write over RAID5

amplification of 1.33. When the write starts from the 1st page, however, the resulting parity update is 1 and the write amplification will decrease to 1.16.

In general, when workload starts to write q pages from ith page in a page group, the number of parity update is $\lceil (q+i-1)/(N-1) \rceil$ when the average write length is q. [1] The average number of parity update q_p is:

$$q_p = \frac{1}{N-1} \cdot \sum_{i=0}^{N-2} \lceil \frac{q+i}{N-1} \rceil \tag{3.17}$$

And the write amplification from parity update is

$$\alpha_{parity} = 1 + \frac{q_p}{q} \tag{3.18}$$

Note that the write amplification from parity protection comes from two ways. The first is the write amplification from parity update, and the second comes from increased space utilization because the write amplification from garbage collection α_{gc} increases as the space utilization increases.

[1] $\lceil x \rceil$ is the smallest integer number larger than or equal to x.

3.3.4.4 Write amplification and lifetime

The write amplifications from ECC recovery and garbage collection are not independent. When garbage collection process picks a block with a page with many bit errors, ECC can detect and correct the bit errors while garbage collection moves the page to empty space for free of additional writes. We count it as write amplification from garbage collection, not from ECC recovery. Thus, the write amplification from garbage collection is

$$\alpha_{gc} = \sum_{i=0}^{K} \left(g(i,x) \cdot P_i \right) / w \tag{3.19}$$

where $g(i,x)$ comes from A_F, $A_{F'}$, A_G, $A_F(a,r,f)$, and $A_G(a,r,f)$ depending on garbage collection policy and hot/cold characteristics of workload and space.

Device level ECC recovery and system level parity update are independent. ECC recovery does not require parity update and vice versa. But garbage collection depends on the amplified writes from parity updates and ECC recovery:

$$\alpha = (\alpha_{rcv} + \alpha_{parity} - 1.0) \cdot \alpha_{gc} \tag{3.20}$$

The write amplification is modeled by factoring the additional writes and their impact on the data loss rate $\sigma(x)$ experienced at the devices.

3.4 Simulation

We built a simulator which takes bit error behavior of SSDs into account. The simulator estimates the lifetime of SSDs under different workloads. Figure 3.7 provides an overview of our simulator. In the figure, red boxes show processes and blue boxes show objects.

Workload generator produces I/O requests and sends them to the Device object.

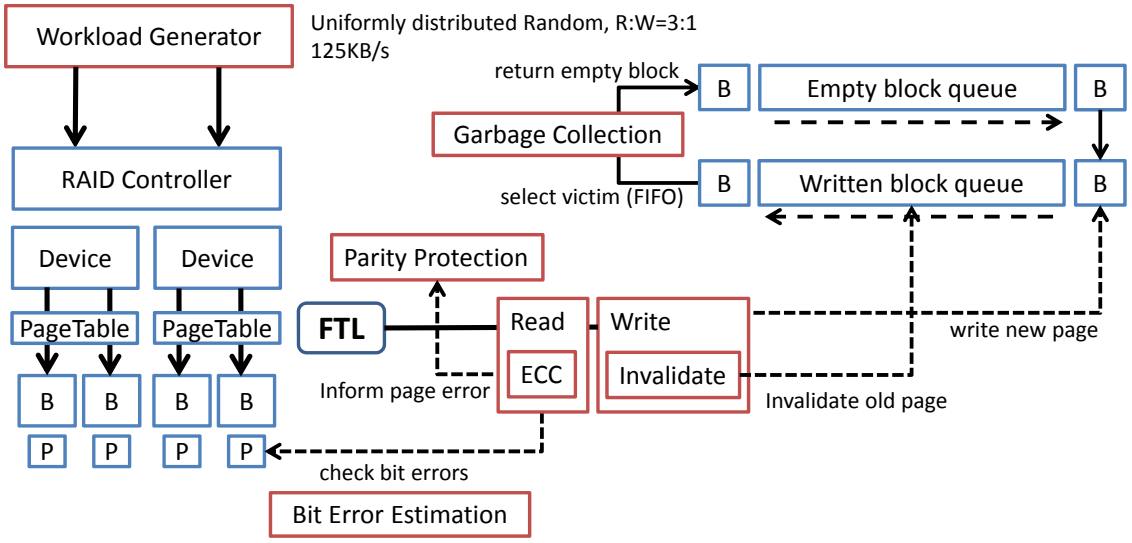


Figure 3.7: Simulator overview

The type of requests and addresses to read or write are decided following a probability distribution.

Bit error generator estimates the number of bit errors in the target page that is read or written. The number of bit errors depends on the amount of time passed since the last access to the page, erase count of the block where the page resides, and bit error rate data which is from a measurement study [58].

The Device object manages a number of Block objects and Page objects, and gives information such as FTL mapping or current status of the block and the pages for a read/write request.

Write requests invalidate old pages and write new data in empty (erased) pages. Read requests checks bit errors and call ECC process and the bit errors are corrected unless the number of bit errors are more than the number of correctable bit errors. Otherwise, ECC calls parity protection to recover the page error. If the parity protection fails, the simulation stops and issues a data loss report.

When the space utilization of SSD reaches an upper threshold, a garbage collection process is invoked and the garbage collection process cleans victim blocks until the space utilization goes below a lower threshold. Garbage collection policy can be different by the way it picks victim blocks in the written block queue. The garbage collection process updates erase count of victim blocks when its process completes.

3.5 Evaluation

The lifetime of an SSD array is evaluated in different environments. Various aspects of SSD arrays are explored.

3.5.1 Simulation environment

We exploited bit error rate mainly from [58], specifically the bit error rate of 3x nm MLC flash memory. We assume that 61-bit correctable ECC for a 4 KB page is employed. We assume a constant annual device failure rate of 3% over the device lifetime based on the survey results in [29]. We consider different device failure rates later in the section. Each SSD is assumed to have a capacity of 80 GB.

30 GB of data and 125 MB/s of workload, with 3:1 read:write ratio, per device as a default, are assumed. This allocates the same amount of data and the same amount of workload to each device. As a result, the total amount of data and workload in an SSD array is proportional to the number of SSDs. The actual space utilization can be more than 30 GB due to parity or replication. For example, in a 4-SSD RAID5 system, each SSD has 30 GB of data and 10 GB of parity which results in space utilization of 0.5 while the space utilization of 4-SSD system with striping is 0.375. Table 3.2 shows the change in important system environment variables according to the number of SSDs in an array. Vendors rate devices by maximum amount of data that can be written to a device per day due to write endurance concerns. For the devices, we consider in our study here, the workload of 125 MB/s translates into 33

SSD count	1	2	4	8	16
Workload (System, MB/s)	125	250	500	1000	2000
Workload (Device, MB/s)	125	125	125	125	125
Data (Striping, System, GB)	30	60	120	240	480
Data (Striping, Device, GB)	30	30	30	30	30
Space Utilization (Striping, Device)	0.375	0.375	0.375	0.375	0.375
Data (RAID5, System, GB)	30	120	160	274	512
Data (RAID5, Device, GB)	30	60	40	34	32
Space Utilization (RAID5, Device)	0.375	0.75	0.5	0.425	0.4

Table 3.2: Simulation environment

device writes per day (DWPD). Enterprise devices are expected and rated to endure 10-50 DWPD [13, 15, 21, 23, 51]. We consider different workload intensities later in the section.

Greedy garbage collection policy is assumed to be employed unless specified otherwise.

TRIM command issued by operating system indicates to SSD which data is invalidated. We assume that the TRIM command is exploited with zero overhead.

The device lifetimes vary from model to model and vendor to vendor. In order to account for these differences, we normalize the lifetime results to the lifetime of a single SSD in a default environment. We call the normalized lifetime as relative MTTDL.

3.5.2 Review of analysis of single SSD

Space utilization and the write amplification from garbage collection are strongly related to the lifetime. Figure 3.8 shows the relationship between space utilization and the lifetime of a single SSD. It shows an interesting point: the expected lifetime does not decrease linearly as space utilization increases. When we increase space utilization from 0.1 to 0.5, the amount of data is five times the original data, with

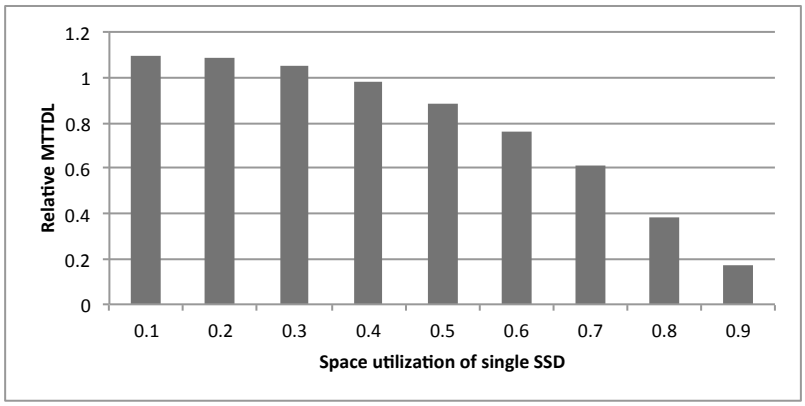


Figure 3.8: Lifetime of an SSD vs. space utilization

only 18% loss of lifetime. If we increase space utilization from 0.5 to 0.9, in contrast, the lifetime decreases by about 70%. The lifetime decreases faster at higher space utilizations.

3.5.3 Simulation

We built a simulator as described in Section 3.4 to verify our analytic models. Since lifetime simulation takes considerable amount of time to finish, we verify the simplest case when garbage collection policy is FIFO and workload is uniformly distributed. We limit the capacity of SSD to 80 MB with a workload of 128 KB/s/device which scales the 80 GB capacity and 125 MB/s/device workload for both analysis and simulation for fair comparison [2].

Figure 3.9 compares the results from analysis to that of simulation. The analytical model tracks simulation results fairly closely except when space utilization is very high. The error between the model and the simulation reaches 12% and 30% at utilizations of 80% and 90%, respectively.

[2]We use workload of 128 KB/s/device instead of 125 KB/s/device in scale down version and compared it to analytic model with 128 KB/s/device workload, since the unit of read and write operation is 4 KB. The rest of analytic models assume 125 MB/s/device workload with 30 GB of data in 80 GB device capacity unless specified.

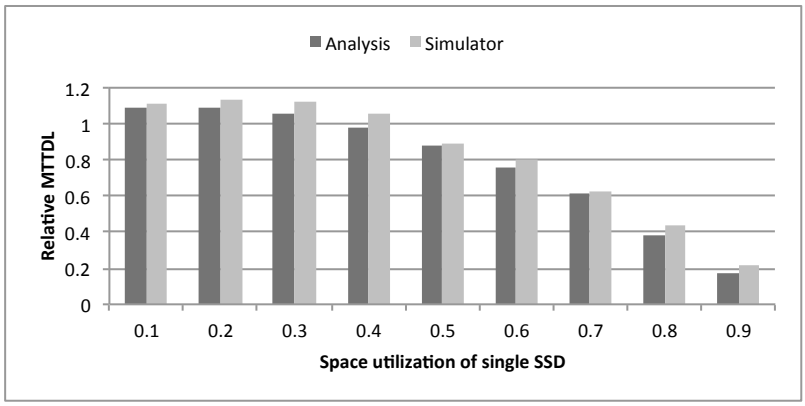


Figure 3.9: Simulation vs. analysis result of a single SSD (80 MB, 128 KB/s)

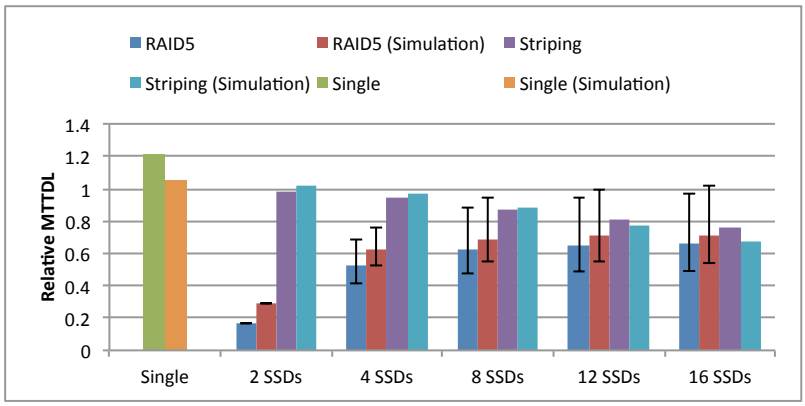


Figure 3.10: Simulation result of SSD arrays (80 MB, 128 KB/s)

We extended the simulator to multiple SSDs to verify our model further. The results are shown in Figure 3.10. The simulation results are obtained from a scaled down version of the system. The results show that the analysis is close to the simulation results at the considered system configuration parameters. In the rest of the section, we present results from our analysis model.

3.5.4 The number of devices

We compare the lifetime of RAID5 to the lifetime of striping for the rest of this section. Since the write amplification from parity update varies depending on small vs. large writes, we describe the lifetime of RAID5 as a range from its minimum to maximum achievable.

We first analyze the impact of write amplification to the lifetime of SSD based arrays. We vary parameters to find when RAID5 is superior to striping and when RAID5 may not be so beneficial in this section.

Figure 3.11 compares the lifetime of different systems with varying number of SSDs. In the figure, the error bar on RAID5 shows the possible range of the lifetime depending on the range of the write amplification from parity update. The lifetime of RAID5 is the lifetime write amplification from parity update is $1+N/(2\cdot(N-1))$. The results in Figure 3.11 bring out many interesting implications. First, mirroring [3] at 2 SSDs suffers extremely from higher write amplification. Its write amplification from parity update (or replica copy) is always 2. Its space utilization is twice as that of striping on 2 SSDs. The lifetime of RAID5 systems grows with the number of devices. The results show that with less than 8 SSDs the overhead from additional writes from parity overwhelms RAID5's reliability benefits.

3.5.5 The amount of data

Since the space utilization of RAID5 is higher than striping due to parity, it is more sensitive to space utilization than striping. This trend is shown in Figure 3.12. As we increase the amount of data, the lifetime of RAID5 drastically decreases and eventually its maximal lifetime is less than striping. This implies that the total amount of data should be less than a certain amount otherwise RAID5 will not

[3] RAID5 for 2 devices is same as mirroring for 2 devices.

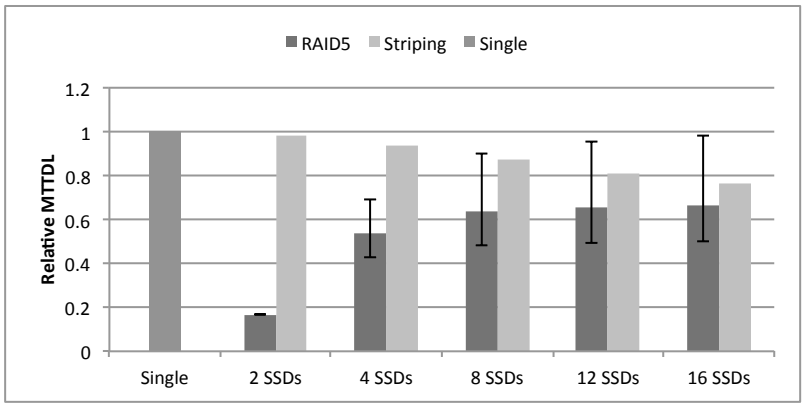


Figure 3.11: Lifetime of different number of SSDs

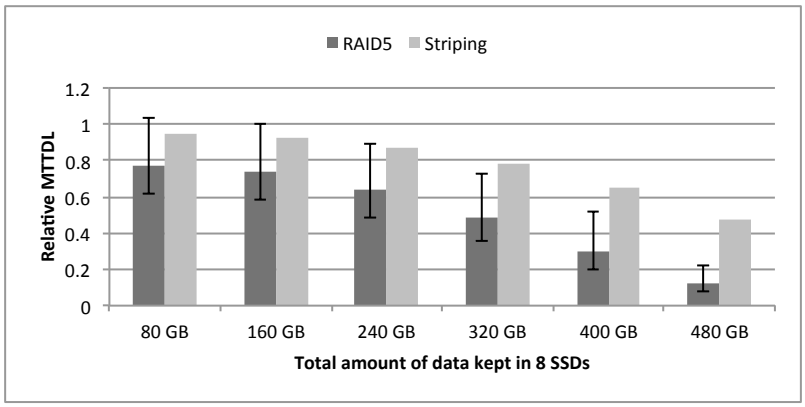


Figure 3.12: Lifetime of 8 SSDs with different amounts of data

be beneficial in terms of reliability. RAID5 is shown to be competitive when the amount of data is less than 240 GB or space utilization is less than about 40%. The 400 GB of data is equivalent to 71% space utilization or 40% over-provisioning. Considering that recent SSDs typically provide over-provisioning from 7% to 20% (93% to 83% of space utilization), additional measures should be taken to 1) increase over-provisioning considerably and/or 2) reduce write amplification further.

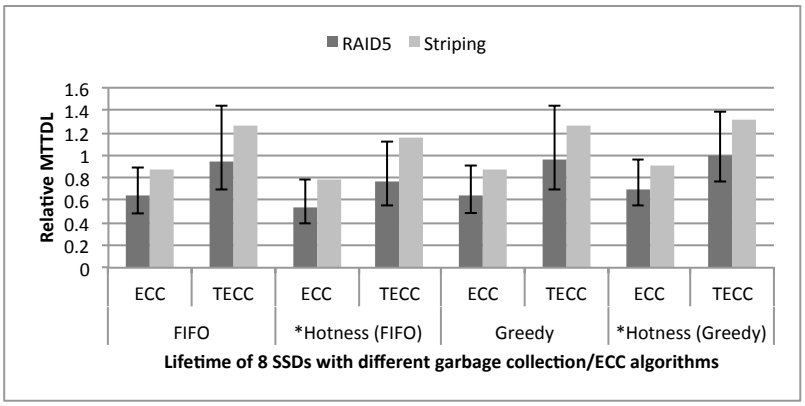


Figure 3.13: Lifetime of 8 SSDs with different garbage collection/ECC algorithms

3.5.6 Garbage collection policy

We exploited analytic model of garbage collection in [10] and extended its FIFO policy model considering non-uniform writes and write length. Figure 3.13 shows the impact of different cleaning policies and hotness of workload on data lifetime. With hot workload hot data blocks can be more efficiently cleaned than cold data blocks. However, FIFO policy picks cold blocks at a higher probability because of larger number of cold blocks than hot blocks. As a result, garbage collection efficiency decreases with hot workload and FIFO policy, while it increases with hot workload and greedy policy.

In contrast, we can see the data lifetime increases with greedy policy and hot workload when the space utilization of SSDs is higher as seen in Figure 3.14. In the figure, space utilization is higher than that in Figure 3.13. Specifically, the amount of data is 60 GB per SSD instead of 30 GB per SSD in Figure 3.13. The figure shows that the lifetime of SSD arrays is determined by garbage collection efficiency. When the space utilization of SSDs is higher, the write amplification from garbage collection is the dominant source of degradation. The gain from efficient garbage

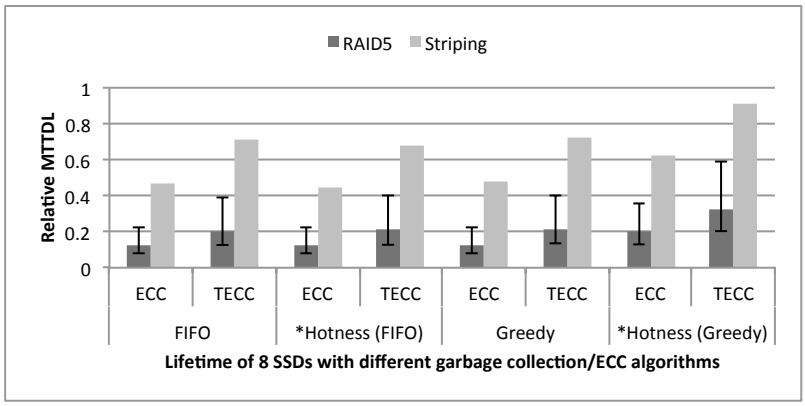


Figure 3.14: Lifetime of 8 SSDs with different garbage collection policies

collection is larger than the loss from higher read disturb error rate, and the data lifetime increases with greedy policy and hot workload.

3.5.7 Advanced techniques

It is noticed that the lifetime gain does not linearly increase as write amplification decreases. This is because write amplification from various sources are not independent. When the number of devices increases, for example, write amplifications from both parity update and garbage collection reduce at the same time. Moreover, when parity update overhead decreases, garbage collection rate also decreases and it results in less write amplification from garbage collection.

Many techniques have been recently proposed to reduce the write amplification. Among them, we evaluated, in Figure 3.15, threshold based ECC (TECC) which reduces the write amplification from ECC recovery. As expected, lifetime increases considerably. When TECC is employed, write amplification from parity update has a higher impact on the lifetime. As explained in Section 3.3.4.4, reducing write amplification from ECC recovery makes the influence of write amplification from parity update relatively larger.

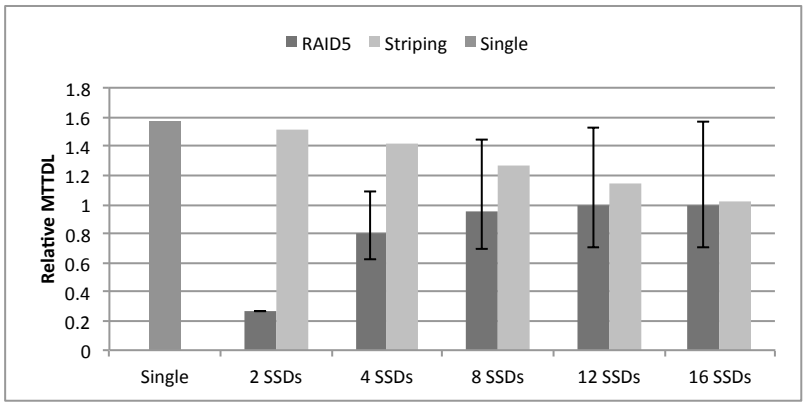


Figure 3.15: Lifetime of SSD arrays with TECC.

Another technique we evaluated is scrubbing, which actively scans cold data during idle time to prevent the accumulation of errors. Figure 3.16 shows the lifetime of 8 SSDs with scrubbing of cold data. We assume that 80% of hot workload is concentrated on 20% of hot space. The result shows that scrubbing is not beneficial in improving the lifetime of the device array. With both ECC and TECC, scrubbing is shown to decrease lifetime slightly. Hot pages enable efficient garbage collection and reduce write amplification. However, hot pages experience increased read disturb errors, canceling out the benefits of more efficient garbage collection. This implies careful consideration is required to employ advanced techniques in SSD based storage systems.

Once again, the flash characteristics, here the read disturb errors, are sufficiently different from magnetic disk drives, and hence warrant careful re-examination of traditional data protection mechanisms.

Advanced techniques to reduce the write amplification favor RAID5. RAID5 has higher space utilization due to parity resulting in higher write amplification from garbage collection. Parity updates also increase write amplification in RAID5 systems. These two factors amplify each other. When we compare the lifetime of

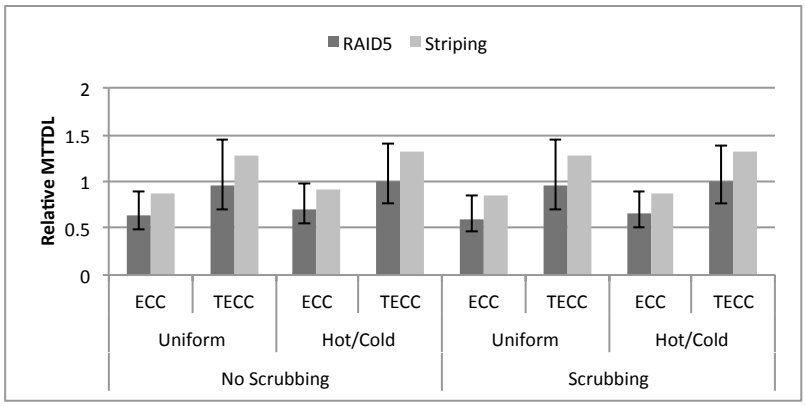


Figure 3.16: Lifetime of 8 SSDs with/without Scrubbing/TECC.

striping and RAID5 in Figure 3.11 and Figure 3.15, we can see that TECC improves lifetime in RAID5 systems more than that in systems with striping.

3.5.8 Read:write ratio

Read:write ratio in workload has an impact on many factors in reliability such as error detection rate and the write amplification from ECC recovery. We evaluate the lifetime of an SSD array with different read:write ratios in Figure 3.17. Under the same intensity of workload, higher read:write ratio implies less amount of write workload. Thus, the lifetime of both RAID5 and striping increase as read:write ratio increases. However, the lifetime of RAID5 improves faster than striping.

3.5.9 Workload intensity

We explore different workload intensities and their impact on the lifetime of RAID5 and striping. Figure 3.18 shows the lifetime of RAID5 and striping when the workload intensity is lower.

The lifetime of RAID5 goes up rapidly as the workload intensity decreases. The less intensive workload wears out SSD at a slower rate, resulting in lower page errors. As a result, device failure rate which is constant is more likely to cause data loss

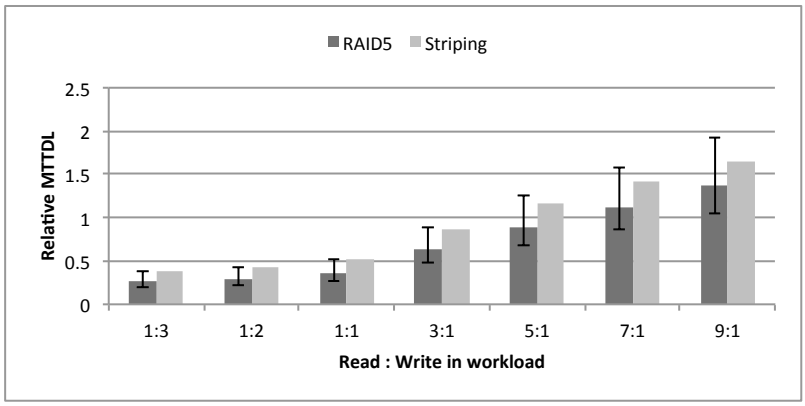


Figure 3.17: Lifetime of 8 SSDs with different read:write ratios in workload

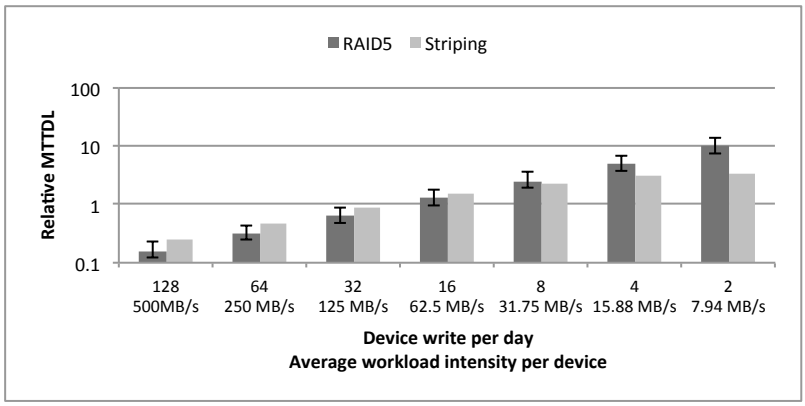


Figure 3.18: Lifetime improvement of 8 SSDs with different workload intensities

than bit/page errors under the less intensive workload. Since parity protection serves recovery from a device failure, RAID5 is more robust than striping under less intensive workloads. It is noted that when the workload intensity is less than 8 DWPD, RAID5 offers higher lifetimes than striping.

We compare the lifetime of RAID5 to striping under less intensive workload, 31.25 MB/s/dev, in Figure 3.19. We can clearly see that RAID5 wins over striping with smaller number of SSDs than RAID5 does under 125 MB/s/dev which is shown in Figure 3.11.

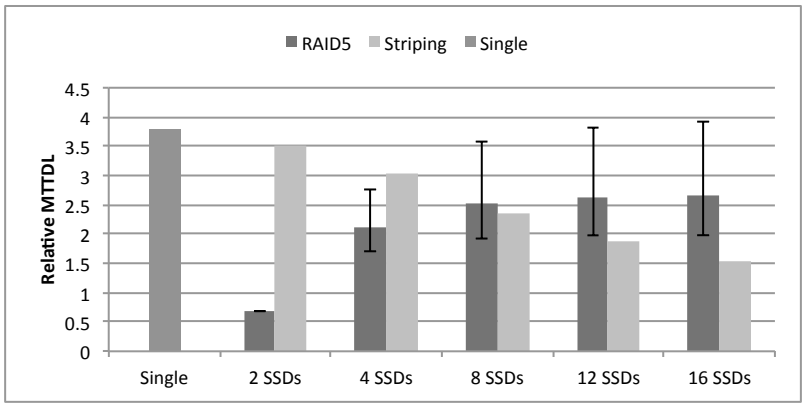


Figure 3.19: Lifetime of SSD arrays with 31.25 MB/s/dev workload

A (nonvolatile) cache can absorb significant part of the workload going to the storage system. The result in Figure 3.18 shows that with reduced workloads, RAID5 can provide higher lifetimes than striping. Moreover, RAID5 obtains a higher benefit from caching than striping. It is noted that in a system with cache, the cache may alter the workload distribution at the drives. We discuss different reliability issues when we employ a nonvolatile cache for an SSD array in Section 4.

3.5.10 Workload characteristics

We consider other parameters such as read:write ratios and data access rates in Figure 3.17 and 3.18. The workloads that are less write intensive result in SSDs that wear out at a slower rate while constant device failure rates contribute more to data loss. Since RAID5 is robust to device level failure, its lifetime grows faster than striping.

Read workload also needs to be considered because of its write amplification. For example, both the 62.5 MB/s/device workload in Figure 3.18 and the R:W=7:1 workload in Figure 3.17 have the same write request rate, but the 62.5 MB/s/device workload has lower read request rate which in part converts into extra writes from fre-

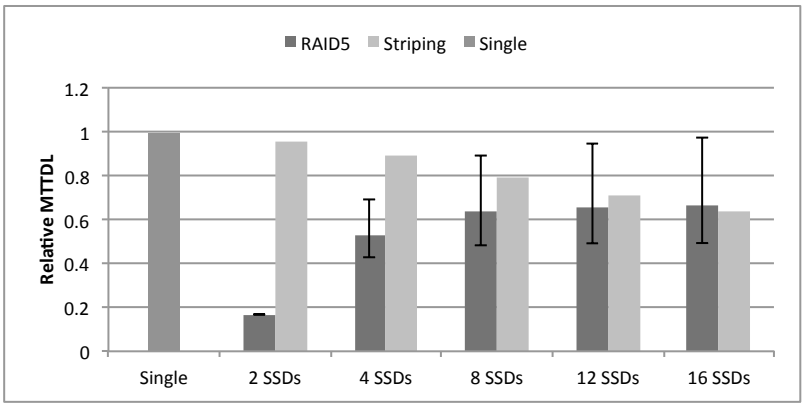


Figure 3.20: Lifetime of SSD arrays at higher annual device failure rate (5%)

quent ECC recovery. As a result, the lifetime of RAID5 under the 62.5MB/s/device workload is less than the lifetime under the R:W=7:1 workload (total 125.0 MB/s request rate).

3.5.11 Device failure rate

We evaluated the lifetime of RAID5 with higher failure rate (annually 5%) in Figure 3.20. In the figure, when the number of devices is more than 8, RAID5 provides higher average lifetimes than striping. This result shows that RAID5 is useful with larger device failure rate and larger number of devices in the array. The larger number of devices contribute to larger failure rates at the system level as well as to smaller overheads for parity.

Figure 3.21 shows the lifetime of 8 SSDs with different annual device failure rates (AFR). It clearly shows that RAID5 improves reliability at higher device failure rates. We assumed an AFR of 3% in this section, which comes from the average returning rate of SSDs in [29].

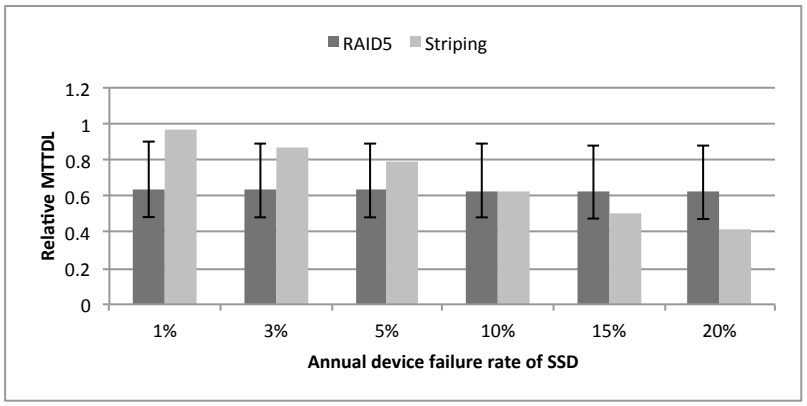


Figure 3.21: Lifetime of 8 SSDs with different annual device failure rates

3.5.12 Non-uniform workload

WE consider a more realistic workload where writes lengths are exponentially distributed. Figure 3.22 shows the shape of Equation (3.12): the write amplification from garbage collection with modified FIFO policy. It shows that the write amplification from garbage collection rapidly drops as the average write length increases. As shown in the figure, the write amplification from garbage collection drastically deceases at higher space utilization as the average write length increases thanks to sequentially written pages. This result is consistent with well-known observation that the performance under large sequential writes is better than that under small and random writes because of less frequent garbage collection process.

We evaluated the lifetime of striping and RAID5 with different average write lengths from 4 KB to 20 MB in Figure 3.23. It is observed that at low device space utilizations, write lengths do not have a significant impact on write amplification. But write lengths can impact write amplification significantly at higher space utilizations. Figure 3.23a shows the results with the default parameters used so far in this section and Figure 3.23b shows the results from a configuration with higher space

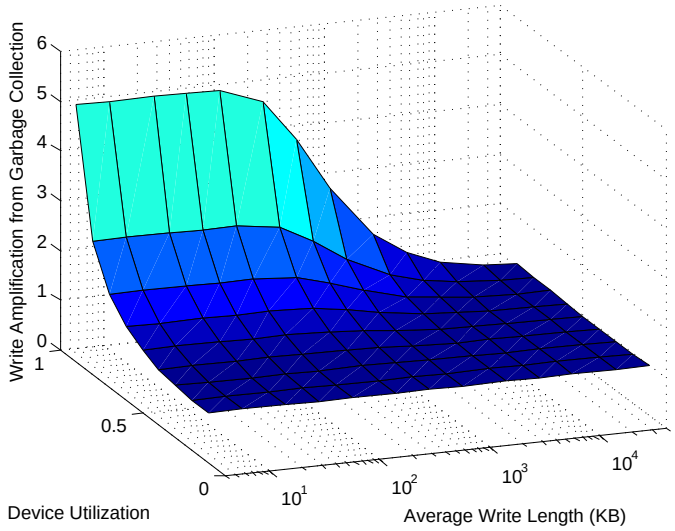
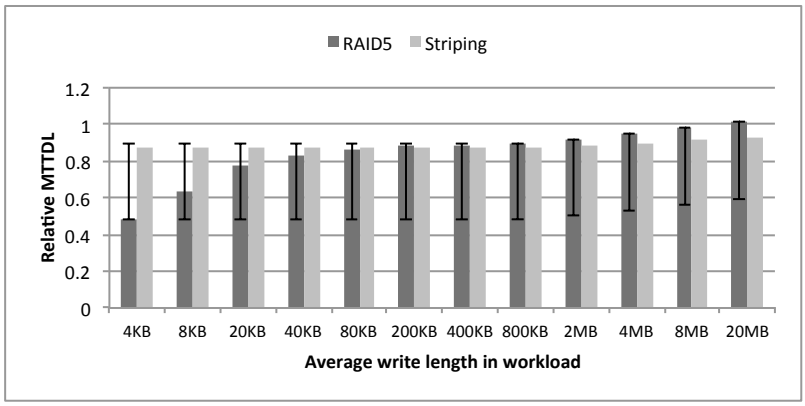


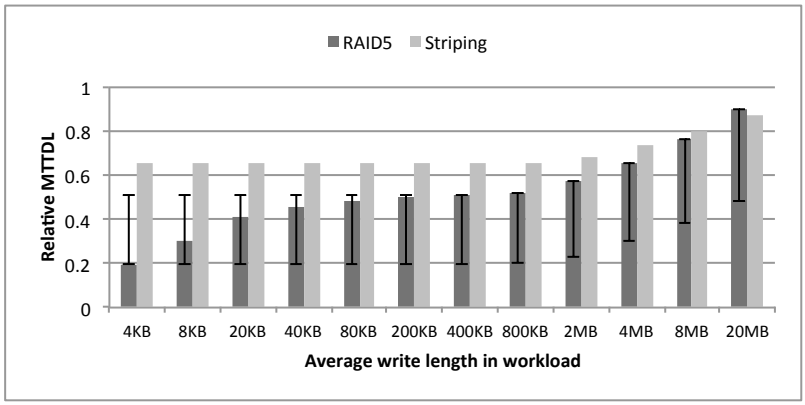
Figure 3.22: Write amplification from garbage collection (policy: modified FIFO)

utilization. In both cases, the lifetime improves as the write length increases from 4 KB to 20 MB. It is observed that the parity update overhead decreases with larger writes and the average data lifetimes get close to maximal lifetimes with increased write lengths. When we compare Figure 3.23a and 3.23b, it is observed that larger write lengths have higher impact on lifetime at higher space utilizations, consistent with the write amplification results in Figure 3.22.

Another observation is that parity update gets efficient when the write length increases. Figure 3.24 shows the change of lifetime as the average write length increases. It shows that the lifetime improves considerably at smaller write lengths, as we increase the write length from 4 KB to 40 KB and again at higher write lengths when the write length increases from 800 KB. At smaller write lengths, the improvement comes from increased efficiency in parity updates and at higher write lengths, the efficiency comes from improved garbage collection efficiency. It is also observed that write lengths play a larger role at higher capacity utilizations.



(a) 30 GB/device



(b) 50 GB/device

Figure 3.23: Lifetime of 8 SSDs with different average write lengths

3.5.13 Erase block size

The evaluation results in Figure 3.23 show that large write lengths make garbage collection more efficient by increasing the number of blocks with mostly invalid pages. Instead of increasing the write lengths in the workload, we can decrease erase block size to increase the number of blocks with many invalid pages. Figure 3.25 is the evaluation result when we employ smaller erase blocks. It shows that the lifetime of SSD arrays increases as erase block size decreases, similar to the results of different average write lengths in Figure 3.23, RAID5 lifetime improves faster than striping.

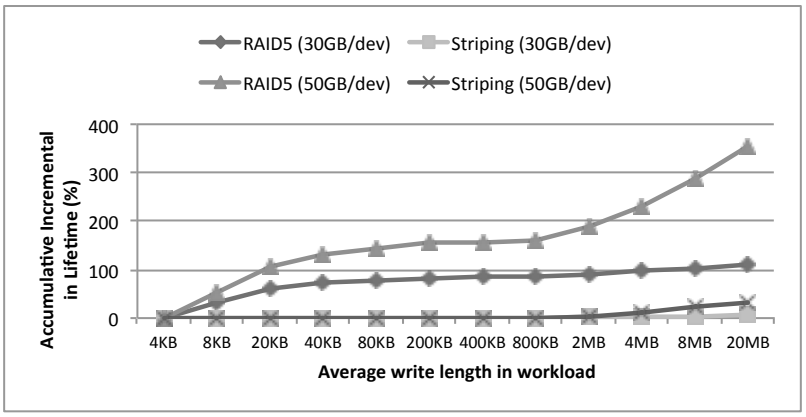


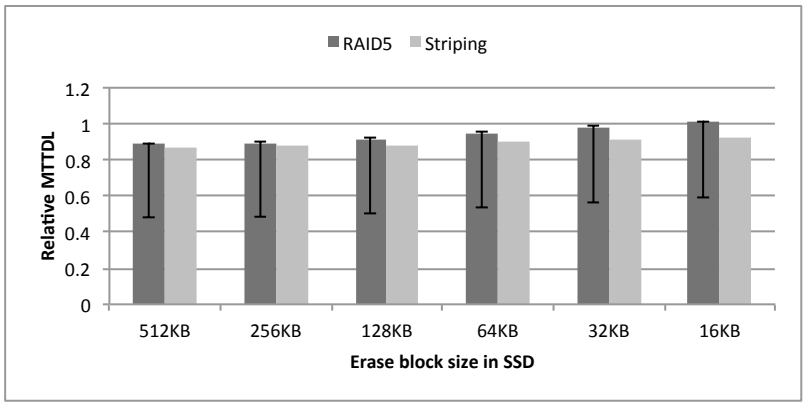
Figure 3.24: Accumulative increment in the lifetime of 8 SSDs

3.5.14 Spare SSD

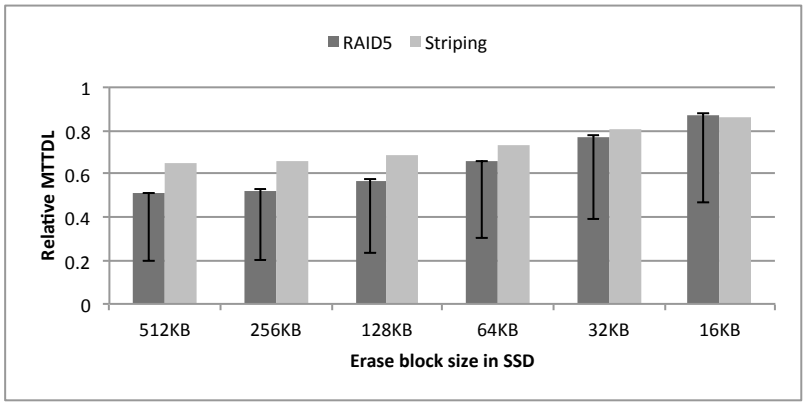
Since space overhead of parity is one of the main sources of degradation in lifetime of RAID5, we evaluated the impact of an additional SSD in a RAID5 system to keep space utilization of RAID5 same as striping. Figure 3.26 shows a comparison of N SSD system to that of an $N + 1$ SSD RAID5. The results show that the lifetime of RAID5 can be lower than striping at small number of devices even when we opt to employ an extra device. However, the lifetimes become closer and roughly comparable above 4 devices. These results show that simply adding an extra device to reduce space utilization may not be sufficient for RAID5 to have higher lifetime than striping at smaller number of devices.

3.5.15 Scalability

Our evaluation results so far have shown that RAID5 may not always be beneficial in improving the lifetime compared to striping when we consider a single array of 4-16 SSDs. However, when we scale out those results to many devices, we see a totally different story. We compare the lifetime of a system consisting of many 4-SSD arrays to the lifetime of striping in Figure 3.27. The results show that striping is very poor



(a) 30 GB/device



(b) 50 GB/device

Figure 3.25: Lifetime of 8 SSDs with different erase block sizes, $q = 512$ KB

at scaling out since only one failure in tens of devices results in data loss. We cannot argue to compose large-scale storage systems without parity protection at the system level, but our results show that RAID5 is not universally beneficial, especially with small number of devices.

3.5.16 Summary of evaluation

Most of our evaluation results imply that write amplification is key to understanding the reliability of SSD arrays:

(1) Write amplification can be rather harmful to the reliability of SSD arrays. The

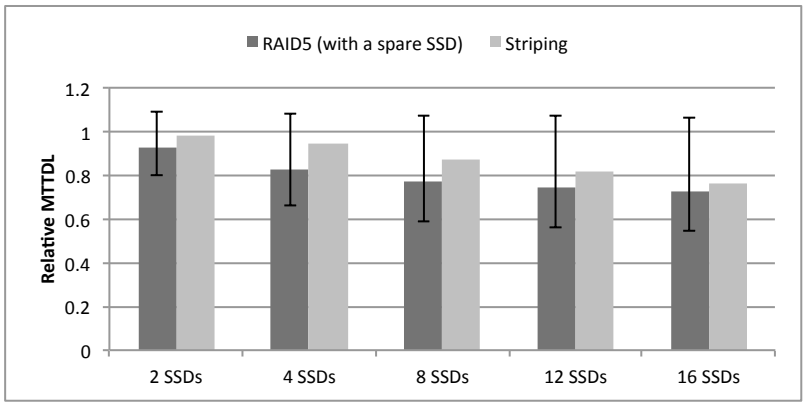


Figure 3.26: Lifetime improvement of SSD arrays (using one more SSD for RAID5)

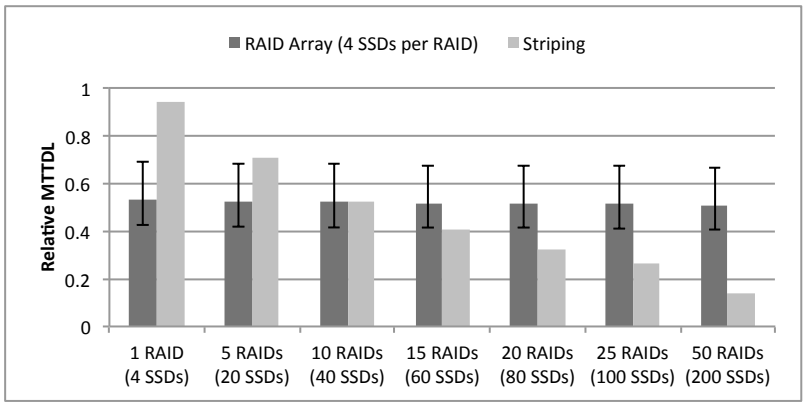


Figure 3.27: A series of 4-SSD RAID5s vs. striping

sources of write amplification should be carefully examined before employing RAID like protection schemes.

(2) Many factors can change write amplification like space utilization, hotness and sequentiality of workload. More efficient system level protection techniques need to be designed taking write amplification into account.

4. SYSTEM LEVEL PROTECTION SCHEMES II: A MIXED SSD ARRAY

In Section 1, we have observed that there are different classes of SSDs. Enterprise class (high-end) SSDs are expensive in cost-per-bit but they are faster and resilient while client class (low-end) SSDs are cheaper but they are slower and less reliable. In this section, we explore those classes of SSDs when they are composed in a hierarchical manner. In a mixed SSD array, high-end SSDs are employed as cache for low-end SSDs main storage to take advantage of both classes of SSDs at the same time.

SLC SSDs are typically employed as a cache because of their higher performance and higher lifetimes. MLC SSDs are typically employed as backend storage because of their lower cost per GB. Even though SLC caches have higher write endurance, since caches tend to absorb a very large fraction of the workload, it is feasible that the SLC caches can wear out faster than the backend MLC storage. In such SSD arrays, the data lifetime is dependent on the minimum of the lifetimes of the cache and storage. Hence, it is important that workload is appropriately distributed across the cache and storage to maximize data lifetimes. Traditionally cache policies have only considered performance as a metric and this section demonstrates the importance of considering lifetime as a metric along with performance. We propose policies for balancing performance and data lifetimes in such mixed SSD arrays.

4.1 System Level Protection Schemes : A Mixed SSD Array

In this section, we build an analysis model of workload distribution of a mixed SSD array. We also provide examples that show the weakness of mixed SSD arrays in terms of reliability.

Figure 4.1 overviews the architecture and workload distribution of a mixed SSD

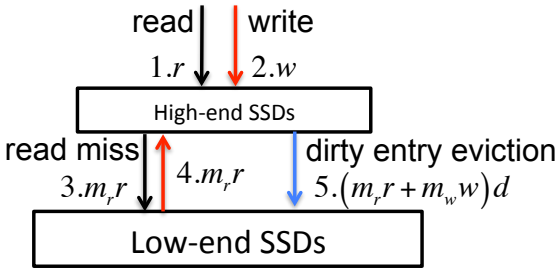


Figure 4.1: Overview of a mixed SSD array with LRU caching policy

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 4.3. In the figure, red line represents write workload in cache and blue line stands for write workload in main storage. Table 4.1 shows the list of symbols frequently used in this section. 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.

In Figure 4.1, 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 has to 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.

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

Table 4.1: List of symbols

Symbols	Description
l_c, l_s	Write endurance of flash in cache and 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 in cache
$t_{c,r}$, $t_{c,w}$	Read ($t_{c,r}$) and write ($t_{c,w}$) latency of cache
$t_{s,r}$, $t_{s,w}$	Read ($t_{s,r}$) and write ($t_{s,w}$) latency of storage

lifetime of the cache, but it can degrade the performance of the storage system. This section studies this tradeoff between performance and lifetime in an SSD array where the cache and backend storage employ different types of flash devices.

We model the workload distribution of the mixed SSD array in Figure 4.1. With LRU caching policy, flushing of dirty data results in writes at main storage as indicated by the arrow 5 in Figure 4.1. The writes served in storage per flash memory cell w_s is

$$w_s = \frac{(m_r \cdot r + m_w \cdot w) \cdot d}{N_s \cdot c_s} \tag{4.1}$$

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

In this section, we assume perfect wear leveling in both cache and main storage. According to Figure 4.1, 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.

$$w_c = \frac{m_r \cdot r + w}{N_c \cdot c_c} \tag{4.2}$$

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

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

$$\begin{aligned} f_c &= \frac{w_c}{l_c} = \frac{m_r \cdot r + w}{N_c \cdot c_c \cdot l_c} \\ f_s &= \frac{w_s}{l_s} = \frac{(m_r \cdot r + m_w \cdot w) \cdot d}{N_s \cdot c_s \cdot l_s} \end{aligned} \tag{4.3}$$

The achievable lifetime of a mixed SSD array T can be restated as

$$T = \frac{1}{\max(f_c,\ f_s)} \tag{4.4}$$

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. 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.

4.1.1 Example

Table 4.2 shows practical parameters of different classes of SSDs and enterprise workloads.

When we employ one high-end SSD as a cache for 3 low-end SSDs for Workload I in Table 4.2, 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:

Table 4.2: 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%

$$
\begin{aligned}
w_c &= \frac{160MB/s \cdot 0.1 + 200MB/s}{1 \cdot 100GB} \\
&= 2.16e\text{-}3 \ writes \ / \ cell \ / \ sec \\
f_c &= \frac{6.75e\text{-}3}{100K} = 2.16e\text{-}8 \ / \ cell \ / \ sec
\end{aligned}
\tag{4.5}
$$

This implies that the workload consumes $2.16e$-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 (4.6).

$$
\begin{aligned}
w_s &= \frac{(160MB/s \cdot 0.1 + 0.15 \cdot 200MB/s) \cdot 0.65}{600GB} \\
&= 4.98e\text{-}5 \ writes \ / \ cell \ / \ sec \\
f_s &= \frac{4.98e\text{-}5}{10K} = 4.98e\text{-}9 \ / \ per \ / \ sec
\end{aligned}
\tag{4.6}
$$

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

This simple example shows that SLC SSD cache can wear out faster and lose data before MLC SSD wears 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 4.2, however, we see a different workload distribution. Cache wears out at the rate f_c in Equation (4.7).

$$\begin{aligned} w_c &= \frac{100MB/s \cdot 0.5 + 250MB/s}{1 \cdot 100GB} \\ &= 3.00e\text{-}3 \ writes \ / \ cell \ / \ sec \\ f_c &= \frac{3.00e\text{-}3}{100K} = 3.00e\text{-}8 \ / \ cell \ / \ sec \end{aligned} \tag{4.7}$$

Meanwhile, storage wears out at the following rate f_s:

$$\begin{aligned} w_s &= \frac{(0.50 \cdot 100MB/s + 0.85 \cdot 250MB/s) \cdot 0.81}{600GB} \\ &= 3.54e\text{-}4 \ writes \ / \ cell \ / \ sec \\ f_s &= \frac{3.54e\text{-}4}{10K} = 3.54e\text{-}8 \ / \ per \ / \ sec \end{aligned} \tag{4.8}$$

The results in Equation (4.7) and (4.8) 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.

As seen above, the high-end SSD cache can wear out faster than low-end SSDs, because 1) majority of workload is served by the cache, and 2) read cache misses

turn into writes to the cache.

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

The major contributions of this section are:

- We find that high-end SSD cache can wear out faster than low-end SSD main storage and this could result in lower lifetimes of mixed SSD arrays.
- We introduce a new metric, 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 2.3 times.

The rest of the section is organized as follows. Section 4.2 states the problem Section 4 targets to solve. Section 4.3 discusses existing caching policies. Section 4.4 shows our sampling based approach for adaptive workload distribution. Section 4.5 evaluates different caching policies with our adaptive workload distribution algorithm.

4.2 Problem Statement

Storage systems could be designed to provide average latencies below a target performance metric. We call this latency as target latency in this section. 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 section, we have to operate with a given cache size.

Both performance (latency [1]) and lifetime are optimized such that a *latency over lifetime* is minimized. The goal of this section can be restated as an optimization problem to minimize the latency over lifetime subject to a latency constraint in Equation (4.9).

$$\begin{aligned} &\text{minimize} \quad L \;/\; T \\ &\text{subject to} \\ &\qquad L \leq L_{max} \end{aligned} \tag{4.9}$$

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

From Figure 4.4, we can estimate the latency of mixed SSD arrays. The average latency L can be measured at the storage system for a given workload.

4.3 Caching policies

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

4.3.1 Request size

Recent solutions [3, 12] propose to selectively cache requests whose size is less than a threshold. 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)

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

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 [3] uses a static threshold (4 MB) and caches I/O requests whose sizes are less than the 4MB threshold.

4.3.2 Hotness

Hotness of data has been used to determine which data should be cached and which data should be directly written to storage. Recent study [41] introduced a workload flow control system based on the hotness of write requests, where a portion of hot write requests are served by the faster device. This approach, however, controls only write requests. Unlike this earlier study, our algorithm considers the impact of data flows from both read and write requests (hits and misses) to properly account for writes into the cache and the storage.

Another recent study [11] employs a secondary cache to determine appropriate sizes of hot cache (frequency), LRU cache (recency), clean cache, and dirty 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 SSD cache which may wear out faster than low-end SSD 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.

Considering a variety of hotness based caching policies, we choose to design a simple policy with a second level cache, called a shadow cache in this section. The size of shadow cache is configurable and it is equal to the size of actual cache in this section. Figure 4.2 briefly describes the architecture of our hotness based caching

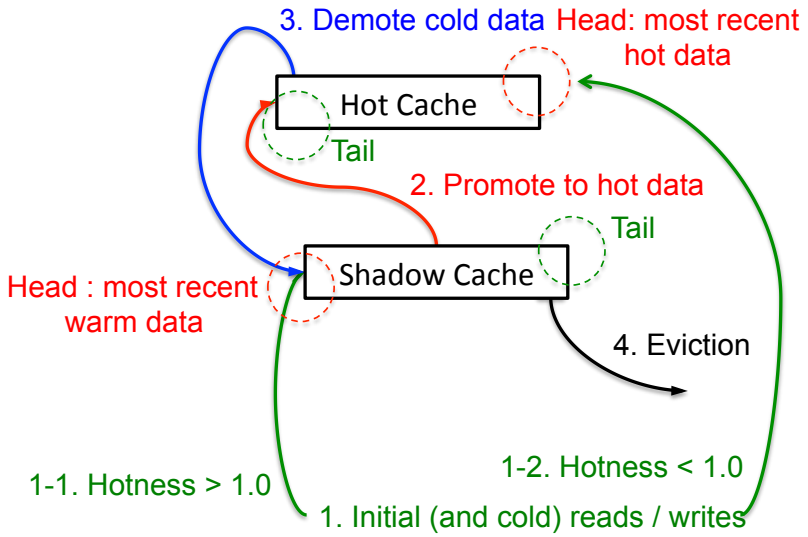


Figure 4.2: Overview of hotness based caching policy

policy employed in this section.

In the figure, initial and cold references are directly served by main storage and their metadata are stored in the shadow cache (arrow 1-1). In each observation period, our hotness based caching policy tracks reference count of used data by 4 KB blocks. The top 10% (configurable) of the recent reference count updates a hotness classifier. The hotness classifier is a threshold to distribute hot and cold workload across cache and storage in this policy. The data whose reference count is larger than the hotness classifier is treated as hot data. We call the cache accepting only hot data as hot cache in this section. Recent and frequent data (hot data) in shadow cache are promoted and relocated to hot cache from the shadow cache in the next reference (arrow 2). Only when the hotness classifier is less than or equal to 1.0 such that hot cache needs to accept all incoming workload, all references are sent to the cache directly (arrow 1-2). Both hot cache and shadow cache internally work like LRU caches. The least recently used data in the tail of hot cache is demoted,

removed and sent to main storage if the data is dirty, and its metadata is kept at the head of shadow cache (arrow 3). The metadata is held until it is evicted from shadow cache (arrow 4).

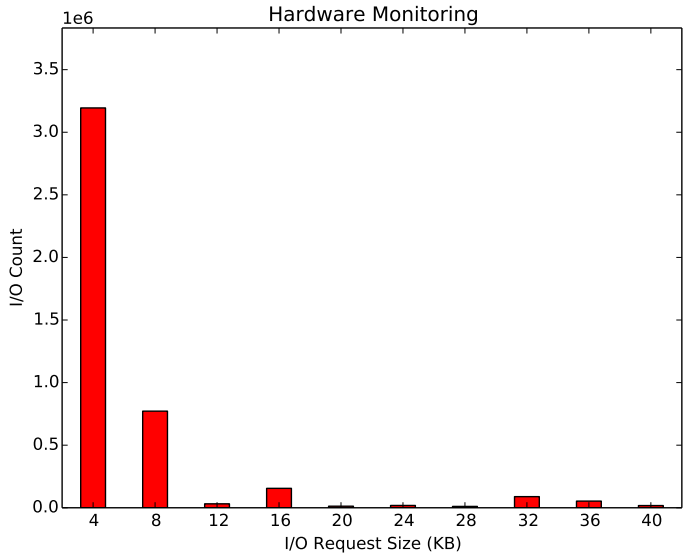
4.3.3 Probabilistic caching policy

Existing caching policies mentioned above do not allow precise control on workload distribution. For example, request size based caching policy cannot distribute workload well when one request size is dominating in the workload.

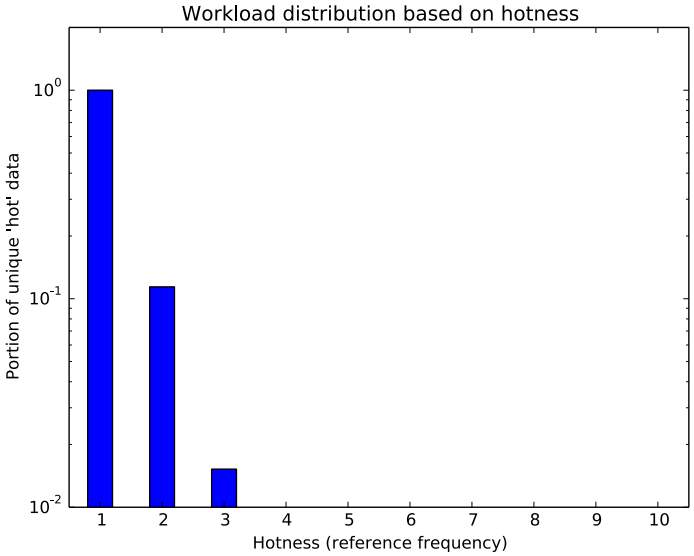
Figure 4.3a shows the distribution of request sizes of a workload trace of hardware monitoring server (hm) application from Microsoft Block I/O Trace [42]. It shows that 4 KB, 8 KB, and 16 KB are the dominant request sizes which make it difficult to choose an appropriate static size parameter for controlling the distribution of workload across the cache and main storage.

Figure 4.3b shows the distribution of reference frequency across logical addresses of data of media server (mds) application trace. It shows that reference frequency is extremely skewed to a few number of blocks and references of the rest of blocks are almost uniformly distributed. Specifically, about 90% of data are accessed once and never revisited, and about 9.5% of data are accessed only twice, and only 0.5% of accesses are repeated more than twice. Such skewed distribution makes it difficult to choose an appropriate hotness classifier to distribute workload appropriately across the cache and main storage.

Under such extreme workloads, existing caching policies cannot precisely control the workload distribution across the cache and storage components of the system. Consequently, a new caching policy is proposed which is able to distribute even an extremely skewed workload. In the new caching policy, requests are *probabilistically* cached. Unlike other policies mentioned above, the probabilistic caching policy does



(a) Hardware monitoring server, request size $\leq$ 40KB



(b) Media server, hotness $\leq$ 10 accesses

Figure 4.3: Workload distribution by (a) request size and (b) reference count.

not depend on skewness of workload in request sizes or hotness.

In addition, the probabilistic caching policy is one of the simplest ways to imple-

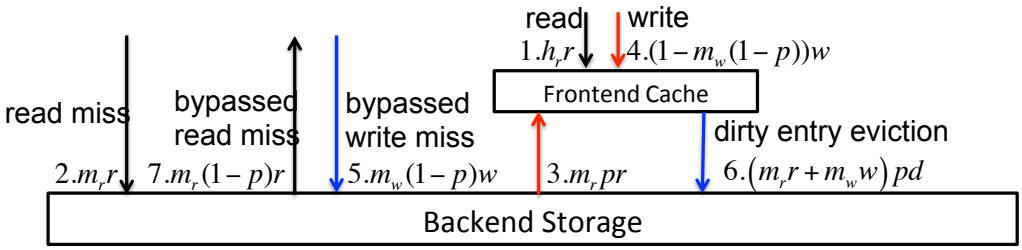


Figure 4.4: Workload distribution (probabilistic caching policy)

ment the workload flow control system. The policy periodically updates a threshold (the probability of caching) based on total write counts in cache and storage. In contrast, the hotness based caching policy in Section 4.3.2 requires keeping track of reference count for each recently used data in the cache and in the shadow cache.

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.

Figure 4.4 shows how the probabilistic caching policy is different from LRU. In the figure, red line shows write workload and black line shows read workload, respectively. 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. In addition, a portion $1-p$ of write cache misses go directly to main storage (arrow 5) which does not exist in the LRU policy in Figure 4.4. 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.

4.4 Adaptive workload distribution

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

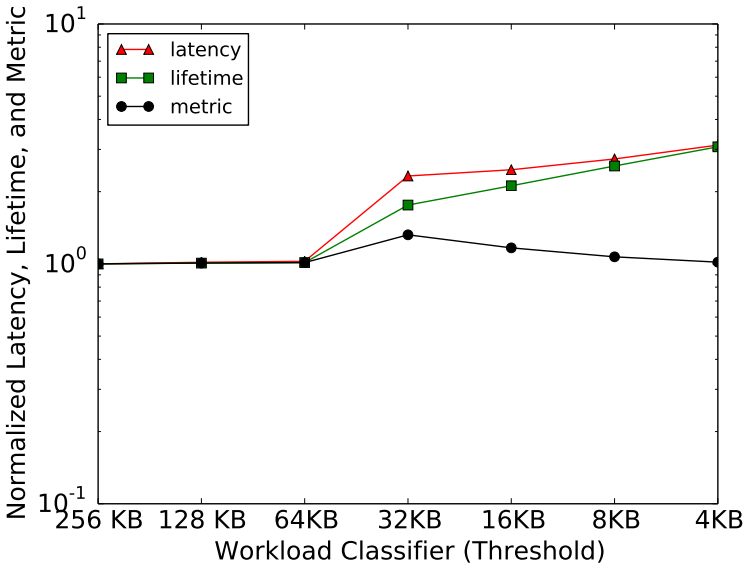
4.4.1 Static threshold based analysis

Static workload classifiers are recently used in many caching policies such as size based caching policy [3, 12]. 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. This policy has been advocated considering competitive sequential read/write performance of low-end storage device arrays.

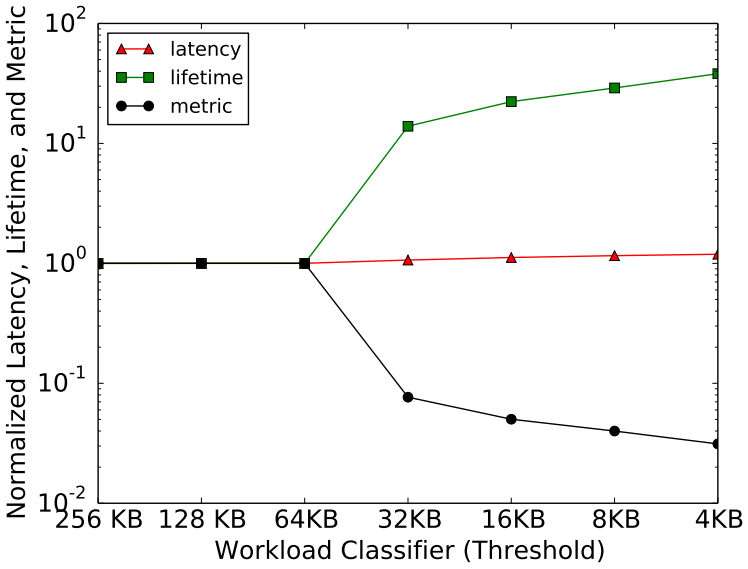
We explore the effectiveness of different static workload classifiers (thresholds) for different enterprise workloads. We use our own trace-driven simulator with enterprise workload traces in [42]. Details of the simulator are discussed later in Section 4.5.1. Figure 4.5 shows latency, lifetime, and latency over lifetime of a size based caching policy with different thresholds. In the figure, we normalize the results of different thresholds to that of 256 KB.

In the figure, 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 4.5a shows that 64KB is the optimal threshold for hardware monitoring server trace. Meanwhile, the optimal threshold of web server application in Figure 4.5b is 4 KB.

We also track latency, lifetime, and latency over lifetime of probabilistic caching policy with different static thresholds in Figure 4.6. 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



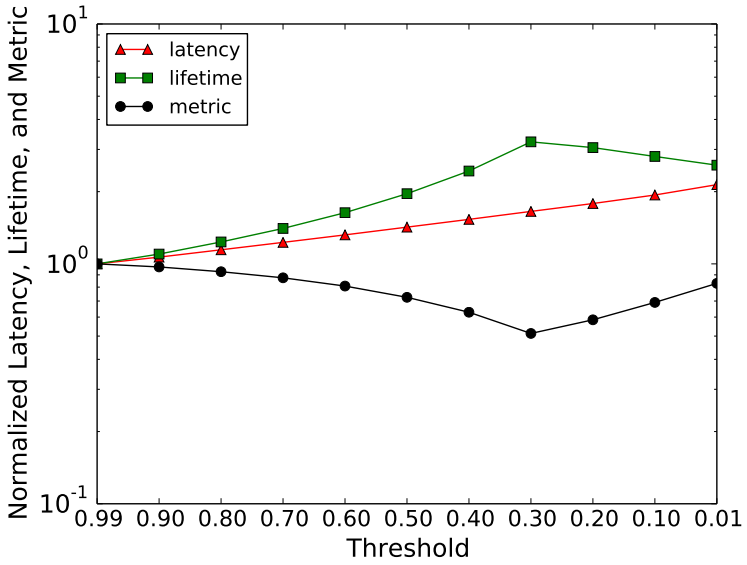
(a) Hardware monitoring server



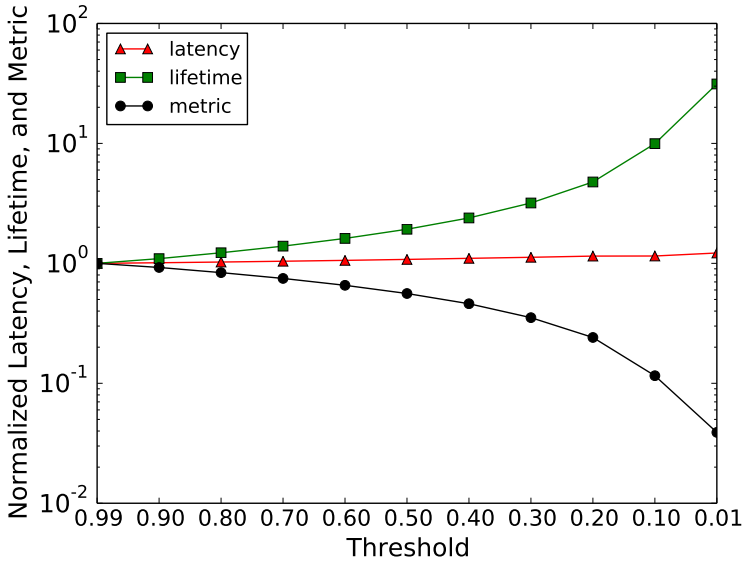
(b) Web server

Figure 4.5: Static threshold based analysis of size based caching policy

is normalized to the result of threshold of 0.99. The results in Figure 4.6a show that the optimal workload distribution is achieved at 0.3 where lifetime is maximized.



(a) Hardware monitoring server



(b) Web server

Figure 4.6: Static threshold based analysis of probabilistic caching policy

In Figure 4.6b, however, it is better to send most of the workload directly to main storage.

The results in Figure 4.5 and 4.6 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 4.6a.

4.4.2 Sampling based approach

Figure 4.5 and 4.6 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 a given workload, we propose a sampling based approach. 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 changes in the characteristics of the workload.

Figure 4.7 shows the architecture of the sampling based adaptive threshold algorithm applied to a probabilistic caching policy. It is noted that the sampling based approach can be employed with other caching policies in a similar way. 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 4.7 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 4.7, sample caches employ different thresholds of 10%, 20%, and so on and the probability 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) $T[s]$:

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

$$\begin{aligned} w'_c[s] &= \alpha w'_c[s-1] + (1-\alpha) \cdot w_c[s] \\ w'_s[s] &= \alpha w'_s[s-1] + (1-\alpha) \cdot w_s[s] \end{aligned} \tag{4.10}$$

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

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 the sampling rate and the size of the timeframe. We use $\alpha = 0.8$ for 1% sampling rate, $\alpha = 0.9$ for 5% and 10% sampling rate, in this section. 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:

$$p_c[s] = \alpha \cdot p_c[s-1] + (1-\alpha) \cdot p_s[s] \tag{4.11}$$

where $p_s[s]$ is the selected threshold in sample caches in timeframe s, and $p_c[s]$ is

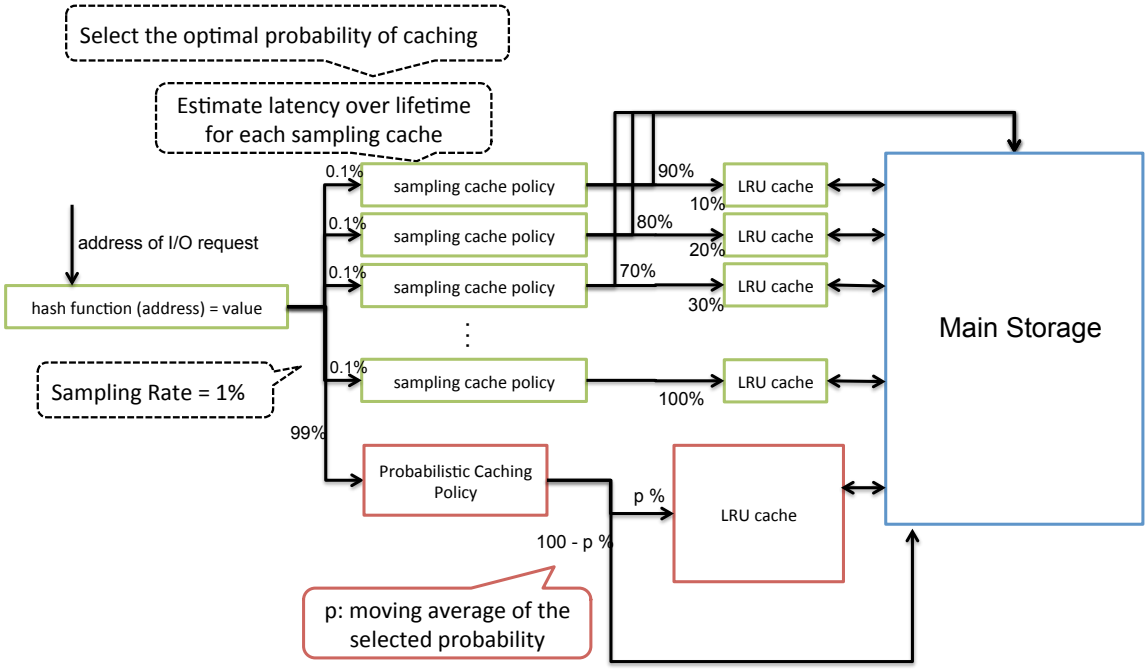


Figure 4.7: Probabilistic cache with sampling method (sampling rate: 1%)

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.

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

4.5 Evaluation

In this section, different caching policies in Section 4.3 are evaluated when the proposed adaptive workload distribution approach is employed.

4.5.1 Simulator

We built a trace-driven simulator based on the analysis in Section 4.1 to see the behavior of mixed SSD arrays with different caching policies in Section 4.3. Figure 4.8 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 4.3.2.

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 the threshold. The adaptive threshold algorithm periodically updates the threshold based on the information from the sample caches. Details of the adaptive threshold algorithm are discussed in Section 4.4.

4.5.2 Simulation environment

We employ enterprise workload traces from Microsoft Research Cambridge [42]. These 13 enterprise applications exhibit different characteristics; they have different cache hit rates vs. cache provisioning [2], read/write ratios, request size distribution, total unique data size, reference frequency (hotness) etc.

Among the 13 MSRC traces, we choose 2 applications and show their cache hit rate vs. cache provisioning in Figure 4.9. The characteristics of a full set of applications are discussed in a recent study [59]. It is observed from Figure 4.9 that cache provisioning impacts performance of different workloads differently. It is observed that the hit rates of the hardware monitoring application improve considerably with

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

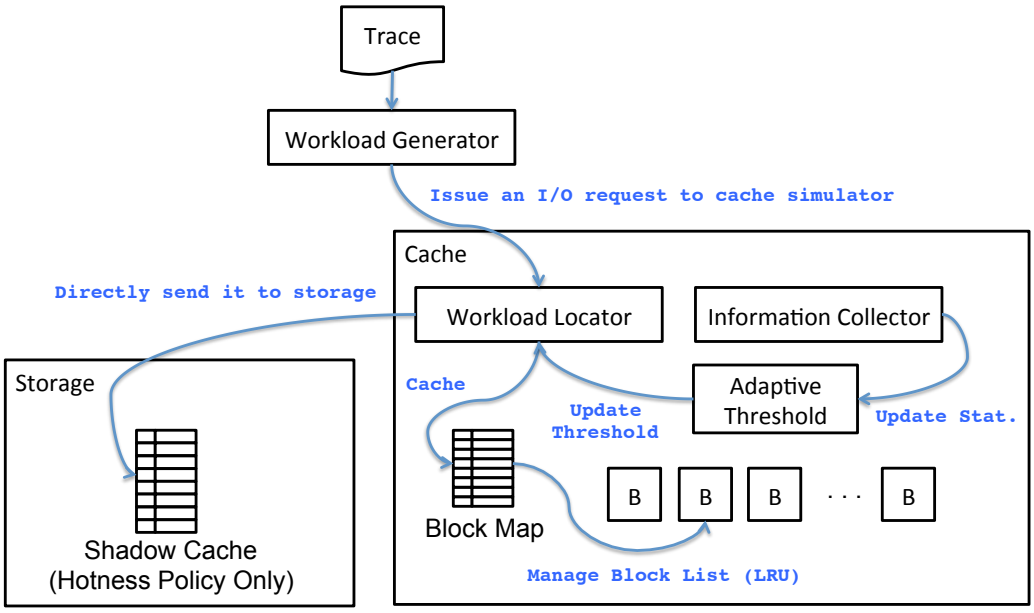
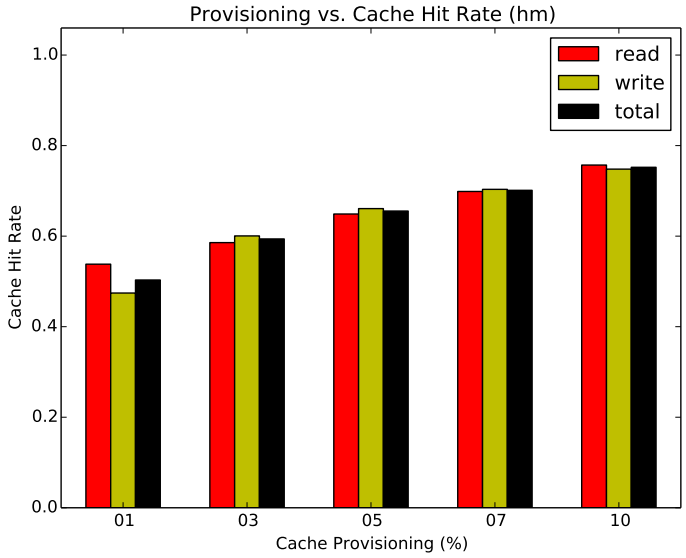


Figure 4.8: Trace-driven simulator overview

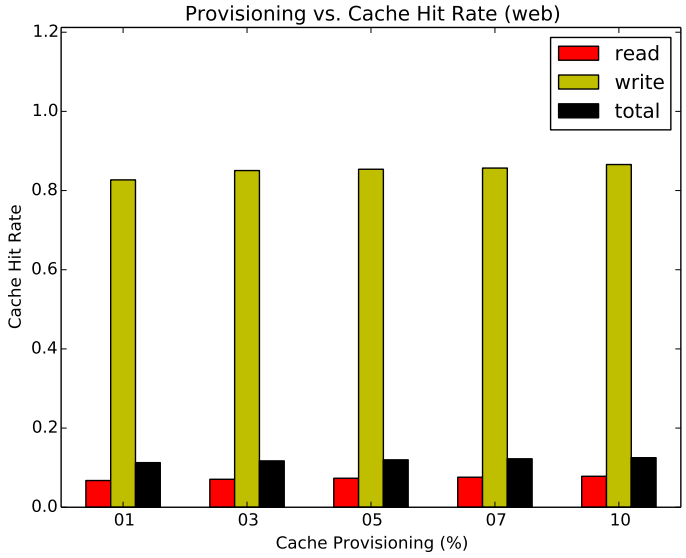
higher cache provisioning. However, the web server workload in Figure 4.9b doesn't benefit significantly from higher cache provisioning. In this case, increasing cache size is not efficient. An appropriate strategy therefore should be established for each workload. Due to the diversity of the characteristics of applications, static workload classifiers are overall less efficient than adaptive workload classifiers.

In this section, 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 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.



(a) Hardware monitoring



(b) Web server

Figure 4.9: Cache hit rates vs. cache provisioning

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 [1].

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

Some caching policies like [45] 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.

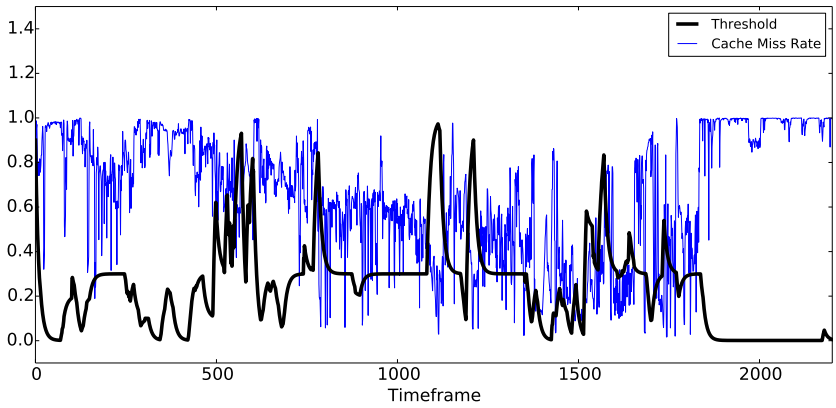
4.5.3 Adaptive threshold algorithm

Figure 4.10 shows how the proposed adaptive threshold algorithm tracks the changes in the characteristics of the workload.

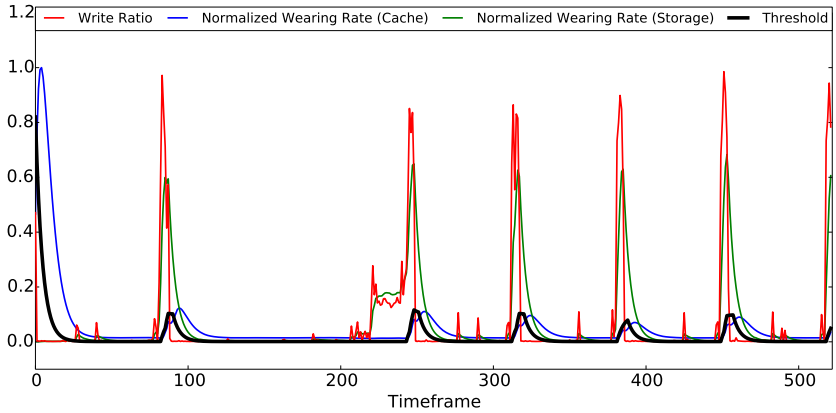
Figure 4.10a shows the change of threshold as a function of average cache miss rate of sample caches. In the figure, it is clearly shown that probability of caching decreases for workloads with weak locality. Higher cache miss rate in sample caches implies weak locality of workload, and caching such 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 4.10b 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



(a) Printer server



(b) Web server

Figure 4.10: Adaptive threshold in probabilistic caching policy

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 4.10a and 4.10b 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.

4.5.4 Different caching policies

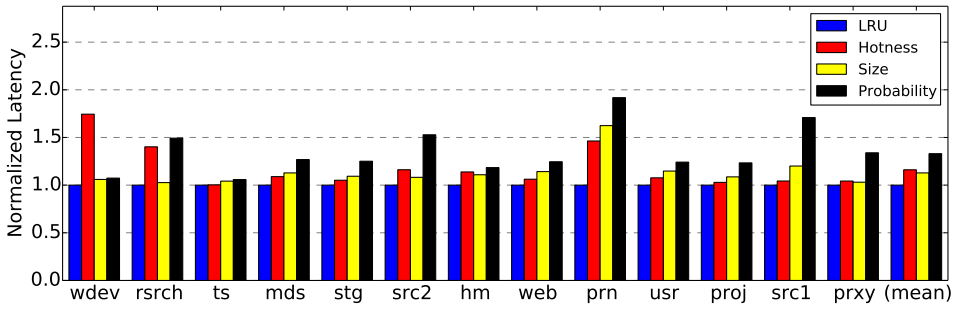
We evaluate different caching policies in Figure 4.11. In the figure, y-axis shows latency, lifetime, and latency over lifetime normalized to the results of LRU caching policy for each MSRC trace.

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.

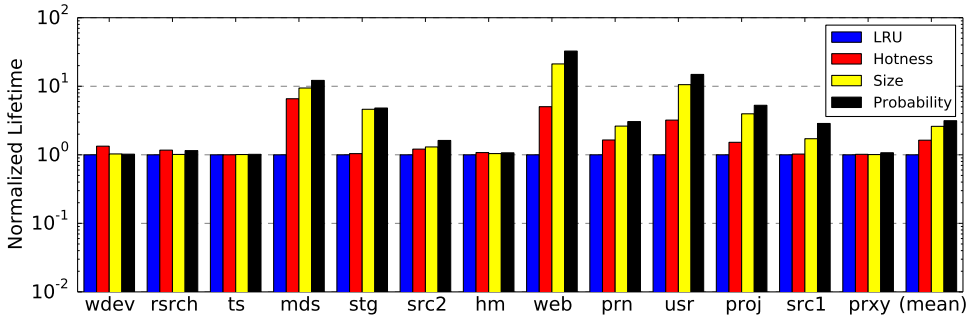
Figure 4.11a shows that latency can increase with the adaptive policies when compared to LRU policy. This is intentional as the adaptive policies are designed to tradeoff latencies for improving lifetimes. It is also noted that latencies are designed to stay below a target latency even with the adaptive policies. Figure 4.11b 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 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.

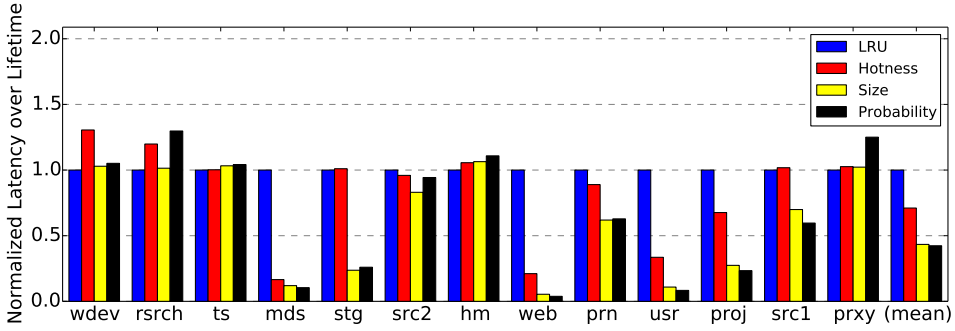
However, the proposed algorithm can be less effective than LRU caching policy



(a) Latency (lower is better)



(b) Lifetime (higher is better)



(c) Latency over lifetime (lower is better)

Figure 4.11: Different caching policies, target latency = 0.4 ms

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

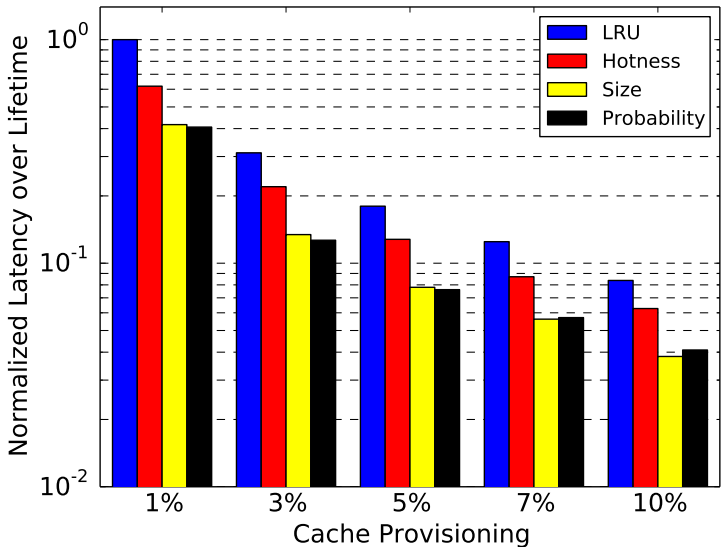


Figure 4.12: Latency over lifetime vs. cache provisioning

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.

4.5.5 Cache provisioning

The impact of the cache provisioning on latency over lifetime is shown in Figure 4.12. 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

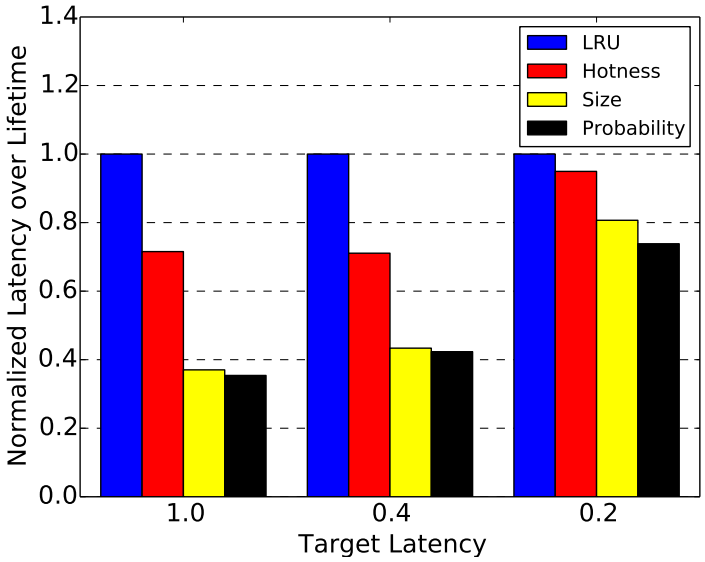


Figure 4.13: Latency over lifetime vs. target latencies

overall metric of latency over lifetime.

4.5.6 Target latency

The target latency prevents adaptive workload classifiers from sending too much workload directly to main storage and under-utilize high-end SSD cache. Figure 4.13 observes the behavior of cache as the target latency is varied.

The results in Figure 4.13 show the trade-off between latency and lifetime clearly. The latency over lifetime is smaller (or shows more improvement) 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.

4.5.7 Sampling rate

We used a sampling rate of 1% in this section. When we increase the sampling rate, we can expect more accurate optimal threshold estimation. However, increasing

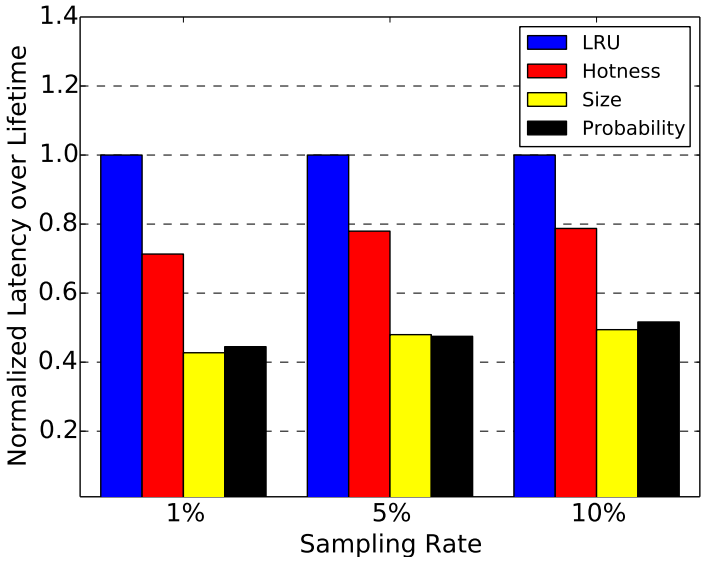


Figure 4.14: Latency over lifetime vs. sampling rates

sampling rate reduces the amount of workload (and the size of the cache) which benefits from the optimal threshold selection. For example, 99% of workload can be distributed with 1% sampling rate, while 90% of workload can be distributed by a more accurate threshold with 10% sampling rate.

The results from different sampling rates are shown in Figure 4.14. The results show that for the workloads considered in this section, 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.

4.5.8 Write amplification

This section assumed equal write amplification in cache and storage. When this is not the case, the lifetimes can be modified to appropriately account for them in Equation (4.1) and (4.2), based on observed statistics.

$$
\begin{aligned}
w'_s &= \beta_s \cdot \frac{(m_r \cdot r + m_w \cdot w) \cdot d}{N_s \cdot c_s} \\
w'_c &= \beta_c \cdot \frac{m_r \cdot r + w}{N_c \cdot c_c}
\end{aligned}
\tag{4.12}
$$

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

5. RELATED WORK

SSDs are promising storage components in modern storage systems while their reliability is still a concern for users. The reliability of SSD is widely studied from the design of powerful ECCs [7, 37, 39, 57], to FTL algorithms [8, 14, 20, 31, 45], to wear-leveling [6] and to RAID [46] over SSD arrays [2, 26, 36, 38, 40, 60].

Many studies [5, 19, 58, 61, 62] have investigated the error behavior of MLC flash memory. Among them, we exploited the results from [58] in modeling the raw bit error rates in Section 2 and 3.

The work in [37] looks at reliability of MLC flash memory with ECC, and estimates uncorrectable bit error rate using a binomial distribution. Unlike this study, we used a Markov model and considered the time-varying nature of bit error rates.

Several previous studies [4, 9, 10, 24, 25] analyzed the impact of write amplification on the performance of SSD storage systems. We broadened the sources of write amplification from garbage collection to include ECC recovery and parity protection, and we mainly focus our attention on the impact of write amplification on the reliability SSD storage systems.

Reliability analysis of memory or disk scrubbing is discussed in [49, 54]. They have already shown how scheduling of scrubbing impacts the reliability of DRAM or magnetic disk drives. Our work here considers the exponential increase of bit error rate of flash memory with increasing write counts to estimate the impact of scrubbing. Earlier work [53] has shown that when increased disk access can lead to lower lifetimes of HDDs, an optimal scrubbing rate may be desirable to increase data lifetime.

Commercial vendors are offering eMLC devices [50, 56] improving MLC lifetime

characteristics. They employ a number of protection techniques such as wear-leveling and parity protection which are mostly employed or assumed in our model.

Novel coding techniques for MLC flash memory are proposed to reduce the number of erase operations [27, 28].

System level protection schemes have shown their effectiveness in HDD storage systems for decades. These studies range from analysis [46, 60] to system level mechanisms [44, 48] for an array of independent disks (RAID).

Many studies have investigated SSD based RAID arrays. A notable study [26] considered SSD based RAID in terms of performance. They discuss the behavior of random writes and parity updates, and conclude striping provides much higher throughput than RAID5. We consider the impact of write workload on reliability.

Some studies [16, 55] have considered different architectures to reduce the parity update performance penalty. The work in [16] employed a cache for parity data. The work in [55] introduced a HDD write cache to reduce write counts in an SSD array. Another work [36] also employed HDDs as a parity storage to relieve the overhead of parity. These works require additional hardware while we consider RAID systems with SSDs only.

Harmonia [33] coordinated garbage collection processes of SSDs in idle time to prevent the garbage collection process from slowing down overall performance. The focus of the earlier study is on performance while we focus on reliability.

Problem of simultaneous wearing out of SSDs in SSD based RAID was considered in the study of differential RAID [2]. The diff-RAID tries to wear out SSDs unevenly to minimize the chance of simultaneous failures, based on the rated write cycles without considering the potential variability in those ratings. We have shown the possible ranges of lifetimes considering different characteristics of the workload.

Employing SSD cache for HDD storage systems is a practical configuration to

exploit lower latency of SSDs. Many studies [3, 12, 22, 30, 35, 52] have investigated different issues in employing an SSD cache for HDD based storage systems. The work in [30] studies the feasibility of flash as a disk cache and proposes to improve SSD cache's performance by separating read and write caches. Another work in [52] provided functionalities such as data protection and silent data eviction to SSD cache. The study in [22] examined interesting issues when SSDs are placed on the client side in a large scale storage system. The work in [35] introduces a deduplication technique and optimizes capacity usage of an SSD cache.

A recent work [43] 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 space utilization of the SSD cache where performance and lifetime of the SSD cache is maximized. Unlike this study, we balance those metrics of both SSD cache and SSD main storage at the same time.

Two recent studies [41] and [34] 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 [41] 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 [34] 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 [11] improves the lifetime of SSD main storage by efficient

usage of NVRAM cache. They increase the cache hit rate and reduce write workload in main storage by dividing cache space into four classes (clean, dirty, frequent, recent) and by adjusting those four spaces in an SSD cache. Unlike this study, we employ less reliable SLC flash as cache and consider the tradeoff between lifetime and performance in such an all SSD array.

The recent work [63] counts read cache misses as write amplification in high-end SSD 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 a performance metric while we use average latency over lifetime. In addition, their hotness and request size based caching policies use static classifiers while we employ adaptive variable classifiers.

6. CONCLUSION

Traditional protection schemes usually increase the number of writes internally to provide data protection. The increased number of writes has not been a problem for the reliability of the systems based on hard disks. However, when they are employed in SSD storage systems whose reliability is highly dependent on the number of writes, the efficacy of these protection schemes has not been clear. The objectives of this study are to understand the implications of the increased number of writes from the protection schemes on the reliability of the storage systems based on SSDs.

Section 2 studied the implications of write amplification at device level from ECC recovery. A Markov model was exploited to show that write amplification can have a significant impact on the lifetime of an SSD. Our analysis reveals that lifetime loss due to the write amplification is about 50% and a considerable amount of loss comes from frequent ECC recovery. We proposed threshold-based ECC which leaves errors on flash until it accumulates bit errors to a certain threshold. Our new scheme is shown to increase lifetime by up to 40%.

Section 3 analyzed the relation between parity protection and the lifetime of SSD arrays at system level. Parity update increases write workload and space utilization which can severely degrade the reliability of SSD arrays. According to our analytical model and evaluation, RAID5 is conditionally better in lifetime than striping due to the overhead of parity. Different factors such as the number of devices and the amount of data are explored, and the results imply that RAID5 is not universally beneficial in improving the reliability of SSD based systems. Our results show that the lifetime of RAID5 can be worse than that of striping in some cases.

In Section 4, we observed that mixed SSD arrays using different classes of SSDs

in a hierarchical manner should consider both latency and lifetime. We showed that high-end SSD caches can wear out faster than low-end SSD main storage under enterprise workloads. We argued that caching policies should balance the latency and lifetime of cache and storage at the same time. We proposed 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] 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`.

[2] Mahesh Balakrishnan, Asim Kadav, Vijayan Prabhakaran, and Dahlia Malkhi. Differential raid: Rethinking raid for ssd reliability. *ACM Transactions on Storage*, 6(2):4:1–4:22, July 2010.

[3] Bcache. `http://bcache.evilpiepirate.org/`.

[4] Simona Boboila and Peter Desnoyers. Write endurance in flash drives: Measurements and analysis. In *Proceedings of the 8th USENIX Conference on File and Storage Technologies*, FAST'10, pages 9–9, Berkeley, CA, USA, 2010. USENIX Association.

[5] Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai. Error patterns in mlc nand flash memory: Measurement, characterization, and analysis. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '12, pages 521–526, San Jose, CA, USA, 2012. EDA Consortium.

[6] Li-Pin Chang. On efficient wear leveling for large-scale flash-memory storage systems. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, SAC '07, pages 1126–1130, New York, NY, USA, 2007. ACM.

[7] Bainan Chen, Xinmiao Zhang, and Zhongfeng Wang. Error correction for multi-level nand flash memory using reed-solomon codes. In *Signal Processing Systems, 2008. SiPS 2008. IEEE Workshop on*, pages 94–99, Oct 2008.

[8] Feng Chen, Tian Luo, and Xiaodong 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*, FAST'11, pages 6–6, Berkeley, CA, USA, 2011. USENIX Association.

[9] Peter Desnoyers. Mathematical models of write amplification in ftls. Slides, presented in NVRAMOS 2011, `http://dcslab.hanyang.ac.kr/nvramos11fall/presentation/Desnoyers-NVRAMOS-2011.pdf`, 2011.

[10] Peter Desnoyers. Analytic modeling of ssd write performance. In *Proceedings of the 5th Annual International Systems and Storage Conference*, SYSTOR '12, pages 12:1–12:10, New York, NY, USA, 2012. ACM.

[11] Ziqi Fan, D.H.C. Du, and D. Voigt. In *Mass Storage Systems and Technologies, 2014 30th Symposium on*, MSST'14.

[12] FlashCache. `https://github.com/facebook/flashcache/`.

[13] Fujitsu. Fujitsu primergy servers solution-specific evaluation of ssd write endurance. `http://globalsp.ts.fujitsu.com/dmsp/Publications/public/wp-solution-specific-evaluation-of-ssd-write-endurance-ww-en.pdf`.

[14] Eran Gal and Sivan Toledo. Algorithms and data structures for flash memories. *ACM Computer Survey*, 37(2):138–163, June 2005.

[15] J. Gathman, A. McPadden, and G. Tressler. The need for standardization in the enterprise ssd product segment. Presentation in Flash Memory Summit, Santa Clara, CA, 2013.

[16] Kevin Greenan, Darrell D. E. Long, Ethan L. Miller, Thomas Schwarz, and Avani Wildani. Building flexible, fault-tolerant flash-based storage systems.

In *Proceedings of the Fifth Workshop on Hot Topics in System Dependability*, HotDep'09.

[17] E. Grochowski. Future technology challenges for nand flash and hdd products. Presentation in Flash Memory Summit, Santa Clara, CA, 2012.

[18] Laura M. Grupp, John D. Davis, and Steven Swanson. The bleak future of nand flash memory. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies*, FAST'12, pages 2–2, Berkeley, CA, USA, 2012. USENIX Association.

[19] L.M. Grupp, A.M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P.H. Siegel, and J.K. Wolf. In *Microarchitecture, 2009. 42nd Annual IEEE/ACM International Symposium on*, MICRO-42.

[20] Aayush Gupta, Youngjae Kim, and Bhuvan Urgaonkar. Dftl: A flash translation layer employing demand-based selective caching of page-level address mappings. In *Penn State University*, 2008.

[21] HGST. Ultrastar^{TM} ssd800mh. `http://www.hgst.com/tech/techlib.nsf/techdocs/07447FAD6BD1A4C288257B48000A8DCA/$file/US_SSD800MH_ds.pdf`.

[22] David A. Holland, Elaine Angelino, Gideon Wald, and Margo I. Seltzer. Flash caching on the storage client. In *Presented as part of the 2013 USENIX Annual Technical Conference*, USENIX ATC'13.

[23] HP. Hp enterprise solid state drive. `http://www8.hp.com/h20195/v2/GetPDF.aspx%2F4AA4-7186ENW.pdf`.

[24] Xiao-Yu Hu, Evangelos Eleftheriou, Robert Haas, Ilias Iliadis, and Roman Pletka. Write amplification analysis in flash-based solid state drives. In *Proceed-*

ings of SYSTOR 2009: The Israeli Experimental Systems Conference, SYSTOR '09, pages 10:1–10:9, New York, NY, USA, 2009. ACM.

[25] A. Jagmohan, M. Franceschini, and L. Lastras. Write amplification reduction in nand flash through multi-write coding. In *Mass Storage Systems and Technologies, 2010 IEEE 26th Symposium on*, MSST'10.

[26] Nikolaus Jeremic, Gero Mühl, Anselm Busse, and Jan Richling. The pitfalls of deploying solid-state drive raids. In *Proceedings of the 4th Annual International Conference on Systems and Storage*, SYSTOR '11, pages 14:1–14:13, New York, NY, USA, 2011. ACM.

[27] Anxiao Jiang, V. Bohossian, and J. Bruck. Rewriting codes for joint information storage in flash memories. *Information Theory, IEEE Transactions on*, 56(10):5300–5313, October 2010.

[28] Anxiao (Andrew) Jiang, Robert Mateescu, Eitan Yaakobi, Jehoshua Bruck, Paul H. Siegel, Alexander Vardy, and Jack K. Wolf. Storage coding for wear leveling in flash memories, October 2010.

[29] Solid state drives no better than others, survey says, 2012. `http://www.pcworld.com/article/213442`.

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

[31] Hyojun Kim and Seongjun 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*, FAST'08, pages 16:1–16:14, Berkeley, CA, USA, 2008. USENIX Association.

[32] Hyojun Kim, Sangeetha Seshadri, Clement L. Dickey, and Lawrence Chiu. Evaluating phase change memory for enterprise storage systems: A study of caching and tiering approaches. In *Proceedings of the 12th USENIX Conference on File and Storage Technologies*, FAST'14.

[33] Youngjae Kim, S. Oral, G.M. Shipman, Junghee Lee, D.A. Dillow, and Feiyi Wang. In *Mass Storage Systems and Technologies, 2011 IEEE 27th Symposium on*, MSST'11.

[34] Sungjin Lee, Keonsoo Ha, Kangwon Zhang, Jihong Kim, and Junghwan Kim. Flexfs: A flexible flash file system for mlc nand flash memory. In *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*, USENIX'09, pages 9–9, Berkeley, CA, USA, 2009. USENIX Association.

[35] Cheng Li, Philip Shilane, Fred Douglis, Hyong Shim, Stephen Smaldone, and Grant Wallace. Nitro: A capacity-optimized ssd cache for primary storage. In *2014 USENIX Annual Technical Conference*, USENIX ATC'14.

[36] Bo Mao, Hong Jiang, Dan Feng, Suzhen Wu, Jianxi Chen, Lingfang Zeng, and Lei Tian. In *Parallel Distributed Processing, 2010 IEEE International Symposium on*, IPDPS'10.

[37] N. Mielke, T. Marquart, Ning Wu, J. Kessenich, H. Belgal, Eric Schares, F. Trivedi, E. Goodness, and L.R. Nevill. In *Reliability Physics Symposium, 2008. IEEE International*, IRPS'08.

[38] Sangwhan Moon and A. L. Narasimha Reddy. Don't let raid raid the lifetime of your ssd array. In *Proceedings of the 5th USENIX Conference on Hot Topics in Storage and File Systems*, HotStorage'13, pages 7–7, Berkeley, CA, USA, 2013. USENIX Association.

[39] Sangwhan Moon and A.L.N. Reddy. Write amplification due to ecc on flash memory or leave those bit errors alone. In *Mass Storage Systems and Technologies, 2012 IEEE 28th Symposium on*, MSST'12.

[40] Sangwhan Moon and A.L.N. Reddy. Does raid improve the lifetime of ssd arrays? In *ACM Transactions on Storage (accepted)*, 2015.

[41] M. Murugan and D.H.C. Du. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems, 2012 IEEE 20th International Symposium on*, MASCOTS'12.

[42] Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron. Write off-loading: Practical power management for enterprise storage. *ACM Transactions on Storage*, 4(3):10:1–10:23, November 2008.

[43] Yongseok Oh, Jongmoo Choi, Donghee Lee, and Sam H. Noh. Caching less for better performance: Balancing cache size and update cost of flash memory cache in hybrid storage systems. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies*, FAST'12, pages 25–25, Berkeley, CA, USA, 2012. USENIX Association.

[44] John Ousterhout and Fred Douglis. Beating the i/o bottleneck: A case for log-structured file systems. *SIGOPS Operating System Review*, 23(1):11–28, January 1989.

[45] Seon-yeong Park, Dawoon Jung, Jeong-uk Kang, Jin-soo Kim, and Joonwon Lee. Cflru: A replacement algorithm for flash memory. In *Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, CASES '06, pages 234–241, New York, NY, USA, 2006. ACM.

[46] David A. Patterson, Garth Gibson, and Randy H. Katz. A case for redundant arrays of inexpensive disks (raid). In *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, SIGMOD '88, pages 109–116, New York, NY, USA, 1988. ACM.

[47] Pure Storage, FA-400. `https://www.purestorage.com/products/flash-array-m/hardware-fa-400.html`.

[48] Mendel Rosenblum and John K. Ousterhout. The design and implementation of a log-structured file system. *ACM Transactions on Computer Systems*, 10(1):26–52, February 1992.

[49] A.M. Saleh, J.J. Serrano, and J.H. Patel. Reliability of scrubbing recovery-techniques for memory systems. *Reliability, IEEE Transactions on*, 39(1):114–122, Apr 1990.

[50] Sandisk. Guardian^{TM} technology platform. `http://www.sandisk.com/assets/docs/WP003_Guardian_Technology_Platform_FINAL.pdf`.

[51] Sandisk. The why and how of ssd over provisioning. `http://www.sandisk.com/assets/docs/WP004_OverProvisioning_WhyHow_FINAL.pdf`.

[52] Mohit Saxena, Michael M. Swift, and Yiying Zhang. Flashtier: A lightweight, consistent and durable storage cache. In *Proceedings of the 7th ACM European Conference on Computer Systems*, EuroSys '12, pages 267–280, New York, NY, USA, 2012. ACM.

[53] Bianca Schroeder, Sotirios Damouras, and Phillipa Gill. Understanding latent sector errors and how to protect against them. *ACM Transactions on Storage*, 6(3):9:1–9:23, September 2010.

[54] Thomas J. E. Schwarz, Qin Xin, Ethan L. Miller, Darrell D. E. Long, Andy Hospodor, and Spencer Ng. Disk scrubbing in large archival storage systems. In *Proceedings of the The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, MASCOTS'04, pages 409–418, Washington, DC, USA, 2004. IEEE Computer Society.

[55] Gokul Soundararajan, Vijayan Prabhakaran, Mahesh Balakrishnan, and Ted Wobber. Extending ssd lifetimes with disk-based write caches. In *Proceedings of the 8th USENIX Conference on File and Storage Technologies*, FAST'10, pages 8–8, Berkeley, CA, USA, 2010. USENIX Association.

[56] Spec Inc. Engineering mlc flash-based ssds to reduce total cost of ownership in enterprise ssd deployments. Presentation in Flash Memory Summit, Santa Clara, CA, 2011.

[57] Fei Sun, Ken Rose, and Tong Zhang. On the use of strong bch codes for improving multilevel nand flash memory storage capacity. In *in IEEE Workshop on Signal Processing Systems: Design and Implementation*, SiPS'06.

[58] Hairong Sun, Pete Grayson, and Bob Wood. Quantifying reliability of solid-state storage from multiple aspects. In *Proceedings of the 7th IEEE International Workshop on Storage and Network Architecture and Parallel I/O*, SNAPI'11, 2011.

[59] Jake Wires, Stephen Ingram, Zachary Drudi, Nicholas J. A. Harvey, and Andrew Warfield. Characterizing storage workloads with counter stacks. In *11th USENIX Symposium on Operating Systems Design and Implementation*, OSDI'14.

[60] Qin Xin, E.L. Miller, T. Schwarz, D. D E Long, S.A. Brandt, and W. Litwin. In *Mass Storage Systems and Technologies, 2003 IEEE 20th Symposium on*, MSST'03.

[61] E. Yaakobi, L. Grupp, P.H. Siegel, S. Swanson, and J.K. Wolf. Characterization and error-correcting codes for tlc flash memories. In *Computing, Networking and Communications, 2012 International Conference on*, pages 486–491, Jan 2012.

[62] E. Yaakobi, Jing Ma, L. Grupp, P.H. Siegel, S. Swanson, and J.K. Wolf. Error characterization and coding schemes for flash memories. In *GLOBECOM Workshops, 2010 IEEE*, pages 1856–1860, Dec 2010.

[63] Jingpei Yang, Ned Plasson, Greg Gillis, and Nisha Talagala. Hec: Improving endurance of high performance flash-based cache devices. In *Proceedings of the 6th International Systems and Storage Conference*, SYSTOR '13, pages 10:1–10:11, New York, NY, USA, 2013. ACM.