Chip-Level RAID with Flexible Stripe Size and Parity Placement for Enhanced SSD Reliability

Jaeho Kim, Eunjae Lee, Jongmoo Choi, Donghee Lee, and Sam H. Noh, Member, IEEE

Abstract—The move from SLC to MLC/TLC flash memory technology is increasing SSD capacity at lower cost, but at the cost of sacrificing reliability. An approach to remedy this loss is to employ the RAID architecture with the chips that comprise SSDs. However, using the traditional RAID approach may result in negative effects as the total number of writes is increased due to the parity updates. In this paper, we describe Elastic Stripping and Anywhere Parity (eSAP)-RAID, a RAID scheme that allows flexible stripe sizes and parity placement. Using performance and lifetime models that we derive of SSDs employing RAID-5 and eSAP-RAID, we show that eSAP-RAID brings about significant performance and reliability benefits by reducing parity writes compared to RAID-5. We also implement these schemes in SSDs using DiskSim with SSD Extension and validate the models using realistic workloads. We also discuss policies such as dynamic stripe sizing and selective data protection that exploits the flexible nature of eSAP. We show that through such policies particular reliability enhancement goals can be met.

Index Terms—Flash memory, Reliability, RAID, SSD

1 INTRODUCTION

We make three contributions in this paper. First, we present reliability analyses of two RAID schemes; one, the well known RAID-5 scheme and two, eSAP (elastic Stripping and Anywhere Parity), a novel scheme that was previously proposed [1], [2] and which we describe in detail later. Next, through analyses and extensive experiments using the DiskSim with SSD Extension platform, we show that eSAP provides reliability better than RAID-5 by significantly reducing the parity overhead and, eventually, reduced P/E cycles for both MLC and TLC based SSDs. Finally, we show the flexible usability of eSAP through two examples, dynamic stripe sizing and selective data protection. Dynamic stripe sizing is a means to adjust the stripe size to improve reliability as wear increases, while selective data protection is a means to target particular data for parity protection. For the later, in particular, we devise a scheme, which we call Meta-eSAP, where reliability of metadata such as the superblock of the file system is enhanced by applying eSAP to just these types of data. We provide quantitative analyses that show that the two methods can improve reliability in particular ways.

1.1 Motivation: Does RAID-5 Improve Reliability?

Solid State Drives (SSDs) making use of multiple bits per cell technology such as MLC/TLC (Multiple/Triple-Level Cell) allow higher density at lower prices than those made with SLC (Single-Level Cell) devices. However, with it, the number of program/erase (P/E) cycles allowed drops one or two orders of magnitude being limited to as far as a few thousands depending on the technology [3]. Hence, the use of these MLC and TLC devices are generally limited to consumer products [4], [5], while SLC devices still prevail as enterprise SSDs. Flash memory devices also share a common characteristic that their bit error rates increase exponentially as P/E cycles increase [6], [7].

Error Correction Code (ECC), traditionally stored in the Out-Of-Band (OOB) area of each page in flash memory, has been used to cope with such bit errors [8], [9], [10]. However, higher density 3x nm and 2x nm NAND technologies are seeing an increase in the error rates requiring larger ECCs. This is a major constraint as with ECC taking up a major portion of the OOB area, less space remains for metadata that needs to be stored [11]. Moreover, ECCs cannot cope with bursty errors nor page-, block-, or chip-level errors. RAID-5-like structures formed with chips internal to the SSDs have been suggested to alleviate the current ECC limitations [12], [13], [14], [15].

Typical SSDs already employ the RAID-0 architecture as they stripe data over multiple chips, while relying on ECC for error recovery. By transforming RAID-0 SSDs to RAID-5 SSDs, we can supplement ECCs with parities with the hope of enhancing error recovery. However, employing RAID-5 within SSDs is a double-edged sword as it also increases the total number of page writes as these extra parity information take up storage space accelerating wear and the bit error rate.

Fig. 1(a) shows the Uncorrectable Page Error Rate (UPER) as more and more bytes are written to a conventional MLC SSD that uses BCH code with 4 bit redundancy per 512B for ECC and one that is enforced
1.2 Related Work

In this subsection, we review studies conducted on the reliability aspect of flash memory and reliability enhancing schemes that have been suggested. We omit discussions on the basics of flash memory and SSDs. We refer readers needing such information to the work by Grupp et al. [6], Agrawal et al. [17], and Gupta et al. [18] among others.

1.2.1 Flash Reliability

New flash memory technologies such as MLC (Multi-Level Cell) and TLC (Triple-Level Cell) increase density by storing multiple bits per cell, in particular, 2 bits for MLC and 3 bits for TLC. This, however, comes with sacrifices in reliability as the number of possible P/E (Program/Erase) cycles is reduced. While the typical P/E cycle for SLC (Single-Level Cell) flash memory is in the 100,000 range, the P/E cycle for MLC and TLC drops to the 10,000s and 1,000s range, respectively [3]. This decrease in P/E cycles exacerbates the reliability issue because the wear down of P/E cycles strongly affect the Bit Error Rate (BER) [3], [6], [7].

In particular, Mielke et al. measure the raw bit error rates of flash memory chips from various vendors and calculate the uncorrectable bit error rates observed after applying ECC [7]. Later, Sun et al. measure the raw bit error rates of 5$\times$nm, 4$\times$nm, and 3$\times$nm MLC flash memory chips and show that bit error rates rapidly increase as flash memory cells become smaller [19]. They also derive an exponential growth model of BER against P/E cycles.

Several wear-leveling schemes including static and dynamic ones have been proposed that try to distribute P/E cycles evenly among blocks, which eventually decreases BER [20]. More recently, Jimenez et al. propose a novel wear-unleveling scheme that identifies and skips the weak pages to lengthen the overall lifetime of the devices [21]. Yang et al. suggest a reliability-aware wear-leveling scheme that evenly out the error rate among blocks to maximize the number of good blocks [22]. Also, Jimenez et al. propose a technique called Phoenix that can switch from MLC to SLC mode to extend the lifetime of SSDs [23]. Liu et al. present a durable SSD cache that can dynamically employ stronger ECC to handle the increased BER [24].

In these studies, it is shown that ECC significantly lowers the bit error rate. However, ECC has its limitations. For example, the detection and correction level of ECC is limited and determined by the size of ECC stored in the OOB area, which is small in size and must be shared with other information like the logical block address as a safeguard for sudden power loss. Therefore, bursty errors beyond the ECC capacity cannot be corrected. In similar context, ECC does not help in cases of page-, block-, and chip-level errors. To cope with these kinds of bursty errors, SSDs with RAID architectures within have been proposed [12], [13], [14].

1.2.2 RAID within SSDs

Many studies on RAID storage systems composed of magnetic disks have been conducted (see [25] for annotated bibliography). Though our study starts off from the RAID concept, it is different from traditional RAID studies in two aspects. First, the medium of storage is flash memory, not magnetic disks. Second, RAID is employed on the components that comprise the storage device, that is, SSDs; that is, we are not composing a RAID system made up of SSD devices. In the following, we summarize some recent studies that consider RAID schemes within SSDs.

Lee et al. apply RAID architecture to their Flash-aware Redundancy Array (FRA) [14]. To reduce parity update cost of RAID-5, FRA retains the parity blocks in buffer memory postponing their writes until the buffer memory is full or no requests arrive for a specified amount of time. Im and Shin propose Partial Parity Cache (PPC), a method that generates partial parity for partial stripes [12]. The limitation of PPC, however, is that it requires non-volatile RAM (NVRAM) to keep these partial parities.

Based on the observation that BER increases as the number of P/E cycles increase, Lee et al. suggest dynamically adjusting the stripe size to maintain similar bit error rates as flash memory ages [13].
early usage of SSDs, they use a large stripe size as BER is low. However, as the SSD ages and hence, BER increases, the stripe size is reduced so that more parity blocks are recorded. This scheme also uses NVRAM to keep parities in memory to reduce the parity update overhead. Through simulation based experiments, they show how the reliability of SSDs employing RAID schemes fare against P/E cycles.

A recent study discusses the use of RAID-5 on proprietary SSDs on a large scale system [26]. Though RAID-5 is not the focus of the study, this study reveals an example of how redundancy is used in real systems.

Another recent work that is not about RAID within SSDs, but about RAID of SSDs is closely related to our work. Moon and Reddy discuss the potential benefits and drawbacks of SSD based RAID in terms of reliability [27]. A Markov model is presented to estimate the relation between parity protection and the lifetime of SSD arrays. Their findings are quite interesting in that they show that parity protection is not always profitable, especially in a single array, since parity updates increases write amplification and space utilization, which may degrade the reliability of SSD arrays. In other words, to reap the benefit of RAID, we have to decrease the parity management overhead, especially the read-modify-write overhead, which is one of the objectives of our study.

1.3 Organization of the Paper

ESAP-RAID was developed to overcome the limitations of RAID-5 [1], [2]. In the next section, we describe the operations of eSAP-RAID in detail. Before doing so, for comparison and analysis we also present the workings of RAID-5 within an SSD. Then, in Section 3, we derive the performance and lifetime models of RAID-5 and eSAP-RAID. We compare the RAID schemes and validate the models in Sections 4 and 5. In Section 6, we discuss extensions that make use of the flexible nature of eSAP, in particular, dynamic stripe sizing and Meta-eSAP, and finally, we conclude with Section 7.

2 RAID-5 and eSAP-RAID for SSDs

2.1 RAID-5 in SSDs

The internals of an SSD composed of five chips that supports the conventional notion of RAID-5 are shown in Fig. 2(a). Stripes 0 and 1 are composed of user data D0∼D3 and parity P0 and user data D4∼D7 and parity P1, respectively. The Stripe Map Table (SMT), whose number pairs represent the Physical Block Number (PBN) and the Physical Page Number (PPN) in the flash memory chip, maintains information about each stripe.

Now consider a case where D1∼D4 are modified. Conventional RAID-5 requires that the updated data be written to the location where the old data resides. Traditionally, with disks, this is a simple task as updated data will simply overwrite the old data. However, with flash memory, the updated data must be written to a new location within the same chip that the old data resides since overwrites are not possible, as shown in Fig. 2(b). The parity then has to be updated through either read-modify-write or reconstruct-write. In the example of Fig. 2, stripe 0 would use reconstruct-write and stripe 1 would use read-modify-write as they minimize the number of read and write operations. Finally, the SMT is modified to reflect the changes made in the stripes.

This approach has a number of limitations. First, the relation between the data and parity page and the particular chip that they are allocated to is fixed. This can lead to wide variance in the write counts depending on the popularity of particular pages. In particular, the chip in which the parity page resides will be more prone to wear out as it must be written to more frequently. These fixed relations not only has negative effects in terms of performance and lifetime of SSDs, but they also make employing flexible wear-leveling schemes and allocation policies to lower cleaning costs and improve lifetime of SSDs difficult. Second, when writes are not updates to existing data but writes of totally new data, data writes may have to stall until the stripe becomes full, leaving data vulnerable. For example, continuing with the example in Fig. 2, if only new data D8 and D9 arrive, parity cannot be calculated just with these pages, and thus these pages cannot be written until a full stripe is formed, that is, another two new pages arrive. Finally, irrespective of whether read-modify-write or reconstruct-write is used, reading of existing data must precede new parity calculations, as is done in traditional disk-based RAID-5 systems. We show, however, that with eSAP-RAID, even these reads are unnecessary.

2.2 eSAP-RAID for SSDs

In this section, we first describe the workings of Elastc Striping and Anywhere Parity RAID (eSAP-RAID or simply eSAP) that we propose. Then, a description
of how garbage collection is done with eSAP-RAID is provided.

The key feature eSAP is that the stripe size need not be fixed, that is, it may vary (up to the full stripe size) as need be, hence the name Elastic Stripping. This feature allows eSAP to construct partial stripes when the input data does not fill the stripe. This is possible as with eSAP parity does not need to be placed at a fixed location, but can be placed at any position within the (partial) stripe; hence, the name Anywhere Parity.

We use the example of Fig. 2(a) to explain these features. eSAP starts with all pages of PPN 0 and 1 comprising stripe 0 and 1, respectively, just like RAID-5. Again, assume that D1~D4 are modified. With eSAP, the controller simply calculates the new parity for the modified pages, writes them on the PPN 2 pages along with the parity value. The outcome is shown in Fig. 3(a). The old pages are simply marked obsolete once the new pages are written. Note that we do not purposely distribute parities as RAID-5 does. This is because eSAP dynamically reconstructs stripes and evenly utilizes all the chips.

Consequently, the first two benefits that eSAP brings about compared to conventional RAID-5 are as follows. First, performance improves as there is no need to read the old pages. Second, all chips are written to evenly helping even the wear out of the flash chips. This is because a stripe is constructed based on arrival order of write requests regardless of the Logical Block Number (LBN), and thus, every strip that comprises a stripe always has the same PPN number. This is regardless of whether the strip is data or parity, or whether write frequency is biased towards a particular strip or not.

We now consider the case where only part of the stripe is written. As with the previous example, assume that new data D8 and D9 are written. At this point, eSAP has two choices. First, it may choose to wait for more data to arrive to form a complete stripe, or second, it may choose to write the data as-is forming a partial stripe. In the first case, data D8 and D9 will be lost should a system crash occur as they are residing in volatile memory. Reliability can be enhanced with the second case, whose result is shown in Fig. 3(b). This figure shows the system state after D8, D9, and their parity have been written. In particular, D8 and D9 have been allocated to PBN 0, PPN 3 of Chips 0 and 1, respectively, while the parity for these data, denoted PSP (for Partial Stripe Parity), has been written to the same PBN, PPN of Chip 2. Should more data D10 arrive, it would be written to Chip 3 and the last parity would be calculated and written to Chip 4.

The eSAP scheme, at first glance, seems to reduce the effective capacity of the SSD compared to RAID-5 in case partial parities are generated. However, this is not the case as space used to store partial parities are consumed only temporarily as they are reclaimed when garbage collection occurs, as we will see later. Hence, usable space for eSAP is the same as that of RAID-5. The negative impact of partial parities is that they are prone to increase the frequency of cleaning as partial parities take up space, albeit temporarily. However, as we present later, the analytic models and the experimental results show that reliability of the SSD actually increases. Specifically, eSAP increases reliability in two aspects compared to RAID-5. First, two or more errors can be recovered if they fall into different partial stripes that are protected by different parities. Second, eSAP decreases the total number of write and erase operations, leading to slower wear and lower error rate.

In summary, eSAP brings about two more benefits compared to conventional RAID-5. One is that the window of vulnerability is minimized with eSAP as parity may be calculated for partial data stripes. The other is that eSAP provides stronger reliability.

To enable eSAP, more information than the SMT of conventional RAID-5 is needed. Recall that for RAID-5, an entry of the SMT consists of a PBN and PPN pair. Note that the chip number is not needed as the LBN of the data designates the chip number. However, for eSAP, since a data page may be allocated to any chip, the Chip ID must also be maintained as shown in Fig. 3. The L2P (Logical-to-Physical) Map Table, which eSAP maintains, translates the Logical Page Number (LPN) that is requested from the file system layer to PPN via the Chip ID, PBN, and PPN. Compared to the SMT that RAID-5 maintains, the extra overhead is the bits for the Chip ID, which is $\log_2 N$ bits, per logical page that the FTL maintains, where N is the number of chips that forms a RAID stripe. As a specific example, assume that we use 16 chips to form a stripe. Then, RAID-5 SMT would require 50 bits per entry, while eSAP would require 54 bits. If we take the 64GB and 128GB SSDs that we use in our experiments later, specified in Table 1 (with OPS set to 5%), then the overall space needed for the two SSDs is 95MB and 93MB, respectively, for RAID-5 and 102.6MB and 100.7MB, respectively, for eSAP. Overall, the result is a roughly 10% overhead in terms of space required for eSAP compared to RAID-5.

In terms of managing the L2P Map Table, the SSD controller could load the map partially as is done in DFTL [18] or in its entirety onto SDRAM during boot-up. When new data is written, the map is updated to reflect this change. To maintain consistency, the map is periodically destaged to flash memory. As this map information is kept in SDRAM, there is a chance that this...
information could be lost upon system failure. Failure recovery can be done similarly to other FTL schemes, and we do not elaborate on this issue any further as it is beyond the scope of this paper.

Let us now consider cleaning for eSAP. As flash memory does not allow in-place updates, updated data must be written to free pages. The cleaning operation is invoked when no free space is available. Fig. 4 contrasts the internals of the SSD before and after cleaning in eSAP. We define the notion of a Dynamic Stripe Group (DSG) to describe the cleaning process. A DSG is composed of physical blocks that comprise a stripe. This means that the physical blocks of a DSG are all of the same block number as shown in Fig. 4, where block 0 of all chips comprise DSG0, block 1 of all chips comprise DSG1, and so on.

The cleaning process of eSAP proceeds in two steps. First, a victim DSG is chosen. In our example in Fig. 4, the DSG with the smallest number of valid pages, DSG1, is selected. For cleaning purposes, at least one empty DSG must always be available. In our example, this is DSG15. In the second step, new stripes constructed from the valid data pages taken from the selected DSG and the parity calculated from these pages are written out to the empty DSG. Note that the old parity pages need not be copied to the new empty DSG, and the new parity pages are calculated with only the valid data pages. In our example, there are 5 valid data pages, namely, D1’, D2’, D3’, D31, and D32. Among them D1’, D2’, D3’, and D31 and their newly calculated parity page, P60, form a stripe, and are stored in Chips 0 through 4. D32, left by itself, forms a partial stripe with the PSP that is calculated for this page, and they are stored in Chips 0 and 1, respectively, as shown in Fig. 4(b).

3 Analytic Models of RAID Schemes

In this section, the analytic models of the RAID schemes are derived. We first review what a Write Amplification Factor (WAF) is as it is used in the analyses. Then, in the subsequent two subsections, we derive the analytic models for write performance and lifetime of an SSD that employs the traditional RAID-5 and the eSAP-RAID schemes, respectively.

Write Amplification Factor (WAF), defined as Eq. 1, where $u$ is the utilization of victim blocks selected for GC and $N_p$ is the number of pages in a block, is a notion that incorporates the cleaning (also referred to as garbage collection or simply, GC) cost in flash memory writes as flash memory storage recycles used space [28], [29].

$$WAF(u) = \frac{u \cdot N_p}{(1-u) \cdot N_p} = \frac{u}{1-u} \quad (1)$$

Eq. 1 tells us that $WAF(u)$ depends on utilization $u$, which is the ratio of valid pages in the block selected for garbage collection. Utilization $u$ can be obtained by measuring real SSDs or can be inferred from the ratio of the data size stored in the SSD and the initial size of the Over-Provisioned Space (OPS), which is space reserved for garbage collection [29], [30], [31]. The equation also tells us that, on average, the GC operation copies $u \cdot N_p$ pages from the victim block to an empty block, which is now left with $(1-u) \cdot N_p$ clean pages. The interpretation of this is that the $u \cdot N_p$ extra page reads and writes are amortized along the next $(1-u) \cdot N_p$ writes to the clean pages that were just generated. Another way to look at this is that when we write $N_{req}^p$ pages in fact, the actual number of page writes that occur in flash memory storage is $N_{req}^p \cdot (1 + WAF(u))$.

3.1 Analysis of SSDs with RAID-5

We make the following assumptions when deriving the analytic model of SSDs that employ RAID-5. The first assumption is that the starting position of write requests within a stripe, as well as the parities, are evenly distributed among the flash memory chips. Essentially, this is assuming that the flash memory chips are being evenly utilized. Second, we assume perfect wear-leveling within the flash memory chip comprising the SSD. The third assumption is that if the whole stripe does not fill up after data write requests arrive, the parity is not calculated immediately, but the stripe is left to wait for more requests to arrive for as much as $P_{wait}$ time, which is a predetermined parameter. If a new write request, whose logical block number is consecutive to the previous requests, arrives within $P_{wait}$, this request is consolidated with the previous requests and written to the same stripe (up to the full stripe), and the consolidated requests are counted as a single request. If, however, no request, whose logical block number is consecutive to the previous requests, arrives within $P_{wait}$, the SSD writes the requests so far as a
partial stripe through either the read-modify-write or 
reconstruct-write mechanism, whichever is more effec-
tive. Requests that are not consolidated are considered 
to be separate requests.

Let us assume that the average data size requested 
to the SSD is $S_{req}$, which can be measured from 
the workload. Then, if we take $S_P$ to be the page size of flash 
memory, the average number of page writes incurred by 
this write request can be calculated as follows:

$$N_{P\text{ req}} = \frac{S_{req}}{S_P}$$

Let us denote the number of pages that comprise a 
stripe as $N_{str}$. Then the stripe consists of $N_{str} = 1$ 
a data pages and one parity page. For simplicity, let us 
denote $N_{p, req} = (N_{str} - 1)N_{P} + 1$. Then, $N_{P\text{ req}}$, the 
average number of parity writes for writing $S_{req}$ data, 
which incurs $N_{P\text{ req}}$ page writes, can be derived as

$$N_{P\text{ req}} = \frac{N_{str} - 1}{N_{P}} + 1$$

(2)

To understand this equation, let us take an example 
shown in Fig. 5(a), where seven pages of data are to 
be written (denoted by the box with the letter ‘D’), five 
pages comprise a stripe and, for simplicity, the parity 
pages are not interleaved. If the number of data pages 
is greater than $N_{str}$, at least $\left\lceil \frac{N_{str} - 1}{N_{P}} \right\rceil$ full stripes are 
needed to write the data and, thus, that number of 
parities need to be written along with the data. Note 
that $\left\lceil \frac{N_{str} - 1}{N_{P}} \right\rceil$ is 1, in our example.

Now let us consider the case where the stripe is not 
completely filled, as we see in Fig. 5(a). The situation 
may be that the partially filled stripe just has the head 
part as in Fig. 5(a)(1), has both the head and tail parts as 
in Fig. 5(a)(2), or just has the tail part as in Fig. 5(a)(3). 
To see the influence of the head and tail formations on 
the number of parity writes, consider Fig. 5(b) where six 
data pages are being written. In the figure, there are four 
shaded data pages and two unshaded data pages. In this 
example, the four shaded data pages incur one parity 
write for all the cases and the remaining two unshaded 
data pages incur one or two parity writes depending on 
their starting position.

Generalizing this example, Fig. 5(c) shows the cases 
where the remaining data pages may be written to 
the four data pages, when there are one through four 
remaining pages. Note that Fig. 5(b) is simply the case 
where there are two remaining pages of Fig. 5(c), that is, 
Fig. 5(c)(2).

Let us now take the example shown in Fig. 5(c) to 
derive Eq. 2. Consider the cases (1) through (4) in the 
figure. As seen in Fig. 5(c)(1), if the remaining data size 
is only one page, then this page write always incurs 
only one parity write. As shown in Fig. 5(c)(2), when 
two data pages remain, with probability 3/4, one parity 
write is required, while with probability 1/4, two parity 
writes are required. Similarly, as shown in Fig. 5(c)(3), 
when three data pages remain, the probability of one 
parity write and two parity writes are both 2/4. Finally, 
when four data pages remain, as shown in Fig. 5(c)(4), 
the probability of one parity write and two parity writes 
is 1/4 and 3/4, respectively.

Let us now generalize these examples. Eq. 2, which 
represents the average number of parity writes for $N_{p\text{ req}}$ 
page write requests, depends on the remaining data size. 
If the remaining data size is $N_{p, req}$ pages, then, with 
probability $\frac{N_{str} - 1}{N_{P}}$, the data fits in a single stripe 
incuring only one parity write. Otherwise, with 
probability $\frac{N_{str} - 1}{N_{P}}$, the data will span two stripes incurring 
two parity writes. Eq. 2 is obtained by summing these 
two cases.

The overhead for writing a data page, $N_{P\text{ req}}$, is then

$$N_{P\text{ req}} = \frac{N_{p, req}}{N_{P\text{ req}}}$$

This shows that a write request of $N_{p\text{ req}}$ pages actually 
requires $N_{p\text{ req}}(1 + \frac{N_{p, req}}{N_{str}})$ page writes due to the parity 
writes. Moreover, as we must consider the garbage col-
collection cost incorporated in $WAF(u)$, the actual number 
of pages written for serving a write request in SSD 
using RAID-5 architecture, which we denote as $N_{p, req}$ 
becomes

$$N_{p, req} = \frac{N_{p, req}}{N_{P\text{ req}}} \left(1 + \frac{N_{p, req}}{N_{str}}\right)$$

Hence, to serve requests that average $N_{p\text{ req}}$ pages per 
request $n$ times, SSDs employing RAID-5 will incur 
$n \cdot N_{p, req}$ page writes. Then, assuming perfect wear-
levelling, $n$ write requests will erase $n \cdot N_{p, req}$ blocks. Then, we can see that a write request incurs 
$N_{p, req}$ number of erase operations where

$$N_{E\text{ req}} = \frac{N_{p, req}}{N_{P\text{ req}}}$$

(3)
In summary, this analytic model shows that the number of page writes and erase operations performed for RAID-5 configured SSDs can be estimated given the number of write requests, $n$, the average request size, $s^{req}$, and the average utilization, $u$, of victim blocks selected for GC.

### 3.2 Analysis of SSDs with eSAP-RAID

In this section, the analytic model of an SSD that employs the eSAP-RAID scheme is derived. To do so, we first compare eSAP-RAID and RAID-5 in terms of write request size and its performance impact.

Similarly to the RAID-5 configuration, eSAP does not write the parity immediately but waits for a subsequent request when the request does not fill the whole stripe. It waits for a maximum of $P_{\text{wait}}$ time, but if the subsequent write request arrives within $P_{\text{wait}}$, eSAP consolidates the two requests and counts them as a single request. Note, however, that, unlike RAID-5, where the subsequent request must be logically consecutive, in eSAP the subsequent request need not be logically consecutive. This is an important key difference.

If the subsequent request does not arrive within $P_{\text{wait}}$ time or the whole stripe is filled with data, then the requested data is written.\(^1\) If the whole stripe is filled with data and there is data still remaining, the parity for the whole stripe is first written. Then, the remaining data starts a new (non-full) stripe, at which point, eSAP again waits for subsequent requests for another $P_{\text{wait}}$ time period. If subsequent requests do not arrive within $P_{\text{wait}}$, a short partial stripe is constructed and written along with the partial parity for this sub-stripe. Once the parity is written, subsequent requests that follow are considered to be a separate request.

We emphasize again that eSAP can consolidate write requests even when their LBNs are not logically consecutive, that is, even if the data of the write request do not fall into the same logical stripe. Hence, eSAP consolidates many more requests compared to RAID-5, where only those requests falling into the same logical stripe are consolidated. Consequently, for the same workload, eSAP not only has a smaller number of write requests, but it also has a larger average request size compared to RAID-5.

We now derive the analytic model of an SSD employing eSAP-RAID. Similarly to RAID-5, if the data comprises more than $N_p^{req}$ pages as shown in Fig. 5(a), at least $\lceil N_p^{req} - 1 \rceil / N_p$ full stripes are needed to write the data. Thus, eSAP-RAID writes the same number of parities along with the data.

Similarly to the RAID-5 case, let us now consider the average number of page writes incurred by the remaining data. Fig. 5(d) shows the possible data and parity deployments for the four data pages and one parity page that comprise a stripe. In the figure, the pages of the current request are the rectangles denoted with D (data page) and P (parity page), while the shaded rectangles refer to the pages occupied by previous requests. The rectangles with X marks will be explained later. Similarly to the RAID-5 case, eSAP also has four possible remaining data size cases.

At first glance, if we compare Fig. 5(c) and 5(d), it looks as if the probabilities of the remaining data pages incurring one parity write and two parity writes for eSAP-RAID will be the same as that of RAID-5. However, there are two differences. First, the cases in the second row that are enclosed in the dashed circle in Fig. 5(d) cannot happen in eSAP-RAID because a parity write always follows a data write, which means that the previous write must have written two or more pages. Therefore, the cases in the second row of Fig. 5(d) should not be considered when deriving the probabilities.

Given this, and following similar derivation as that of RAID-5, when the remaining data is $N_p^{req, rem}$ pages, the probability of the remaining data fitting in a single stripe and incurring a single parity write is $P_{\text{rem}}^{\text{single}} = \frac{N_p - N_p^{req, rem}}{N_p - 1}$, while the probability of the data spanning two stripes and incurring two parity writes is $1 - P_{\text{rem}}^{\text{single}}$. Thus, the average number of parity writes for writing data of $N_p^{req}$ pages, which we denote $N_{p,c,\text{eSAP}}$, is as follows:

$$
N_{p,c,\text{eSAP}} = \frac{N_p - N_p^{req, rem} + \lceil N_p^{req, rem} \rceil}{N_p - 1}
$$

Then, the parity overhead for writing a data page is

$$
N_{p,o,\text{eSAP}} = \frac{N_{p,c,\text{eSAP}}}{N_p}
$$

A write request of $N_p^{req}$ pages, then, actually requires $N_p^{req}(1 + N_{p,o,\text{eSAP}})$ page writes due to the parity writes.

Let us now consider the second difference between eSAP-RAID and RAID-5 that has to do with the garbage collection cost incorporated in WAF($u$). Recall the rectangles marked with an X in Fig. 5(d). These are pages that are abandoned and wasted. Note that when only one page is left in a stripe in eSAP-RAID, this page cannot be written to as there is no room for parity for that data. In this situation, eSAP abandons this unused page and simply starts to constructs a new stripe. Such action does not incur a page write but wastes space resulting in increased garbage collection cost. Consequently, parity writes and space that is abandoned and wasted have the same effect on garbage collection.

Let $N_{p,\text{waste}}$ be the sum of the average number of parity writes and wasted pages, for writing $N_p^{req}$ pages.

---

\(^1\) Note that eSAP requires no more buffer space than does RAID-5 during the $P_{\text{wait}}$ waiting period as consolidating requests does not exceed the stripe size.
Also, as seen in Fig. 5(d), the space wasted per request, on average, is \( \frac{1}{P_{req}} \) pages. Thus, \( \bar{P}_{parity\_waste} \) is

\[
\bar{P}_{parity\_waste} = \bar{P}_{parity} + \frac{1}{\bar{P}_{req}}
\]

Then, \( \bar{P}^{PO\_WA}_{\_waste} \), the parity overhead and wasted space for writing a data page is

\[
\bar{P}^{PO\_WA}_{\_waste} = \frac{\bar{P}_{parity\_waste}}{\bar{P}_{req}}
\]

This tells us that for a write request of \( \bar{P}_{req} \) pages, garbage collection incurs \( \bar{P}_{req} \cdot (1 + \bar{P}^{PO\_WA}_{\_waste}) \cdot WAF(u) \) additional page writes.

Then, for an eSAP configured SSD, the actual number of written pages for serving a write request, \( \bar{P}^{PW\_eSAP}_{\_req} \) can be obtained by summing up data and parity writes and the additional writes for garbage collections.

\[
\bar{P}^{PW\_eSAP}_{\_req} = \bar{P}_{req} \cdot (1 + \bar{P}^{PO\_eSAP}_{\_waste}) + \bar{P}_{req} \cdot (1 + \bar{P}^{PO\_WA}_{\_waste}) \cdot WAF(u)
\]

Finally, the average number of erase operations incurred by a write, \( \bar{P}^{req\_eSAP}_{\_E} \), becomes

\[
\bar{P}^{req\_eSAP}_{\_E} = \frac{\bar{P}^{PW\_eSAP}_{\_req}}{\bar{P}_{req}}
\]

Again, in summary, this analytic model shows that the number of page writes and erase operations performed for eSAP configured SSDs can be estimated given the number of write requests, \( n \), the average request size, \( S_{req} \), and the average utilization, \( u \), of the victim blocks selected for GC.

### 4 Performance Evaluation

In this section, discussions of the experimental evaluations for the SSD RAID schemes considered in this paper are given. Specifically, we compare the I/O response time and the parity overhead of both MLC and TLC based SSDs employing RAID-0, RAID-5, and eSAP under a simulation environment. As current SSDs employ RAID-0, they are denoted as ECC in the experimental results as the ECC technique is the only means of coping with bit errors. We also note that the models that were described in Section 3 are validated with experimental results in Section 5.

#### 4.1 Experimental Environment

To evaluate our proposed scheme, we implement the ECC, RAID-5, and eSAP schemes within the DiskSim SSD extension [32]. The parameter settings of the simulator are presented in Table 1, which were obtained from data sheets of commercial MLC and TLC SSDs [33], [34], while the BCH numbers are from [11], [35]. In all configurations, we make use of 8 flash memory chips and thus, a physical stripe consists of eight pages that belong to different chips. Throughout the experiments, we set the over-provisioned space to be 5% of flash memory space.

Table 2 summarizes the workloads used for the experiments. The first column lists the name that we refer the workload as, the overall characteristic, the total request size, and the write ratio of the requests of each workload. The Financial workload is a trace that was collected at financial institutions and is mainly a random write intensive workload from OLTP applications [36]. The Exchange workload is a trace obtained from an Exchange server that serves 5000 corporate users collected over a 24-hour period. This workload shows random I/O characteristics, and of the 9 volumes that it is composed of, we use the trace of volume number 2 [37]. The MSN trace is that of a 6 hour duration collected from four RAID-10 volumes of the MSN storage back-end file store. Of the four volumes, we use the trace of volume 2 in our experiments [37].

The numbers in the fourth column and beyond of Table 2 are the postmortem numbers obtained for RAID-5 and eSAP with the MLC and TLC flash memory technologies. The parameters are defined in the model analysis of the previous section and used to explain the performance numbers presented in this section and Section 5. Note that, even for the same workload, RAID-5 and eSAP show different values regarding the number of write requests and the average size of writes since they use different methods to consolidate consecutive write requests, as discussed in Section 3. Also, note that under different MLC and TLC flash technologies, the values are different as well since they depend on the page and block size of the technology.

#### 4.2 Experimental Results

The \( P_{wait} \) value needs to be specified for our experiments. In our study, we considered two values of \( P_{wait} \):
TABLE 2: I/O workloads and their characteristics (first column) and the workload characterizing parameters obtained through experiments to be used in the analytic model (fourth column and beyond): u: observed utilization, stripe(x): x is stripe size

<table>
<thead>
<tr>
<th>Workload (Total Data Req.,Write Ratio)</th>
<th>Flash</th>
<th>Scheme</th>
<th># of Write Req.</th>
<th>Avg. size of Write</th>
<th>u for stripe(8)</th>
<th>u for stripe(16)</th>
<th>u for stripe(32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Financial (35.7GB, 0.81)</td>
<td>MLC</td>
<td>RAID-5</td>
<td>4317K</td>
<td>8.3KB</td>
<td>0.57</td>
<td>0.55</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>eSAP</td>
<td>436K</td>
<td>82KB</td>
<td>0.52</td>
<td>0.47</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td>TLC</td>
<td>RAID-5</td>
<td>4105K</td>
<td>12.7KB</td>
<td>0.45</td>
<td>0.43</td>
<td>0.42</td>
</tr>
<tr>
<td></td>
<td></td>
<td>eSAP</td>
<td>338K</td>
<td>145KB</td>
<td>0.4</td>
<td>0.34</td>
<td>0.32</td>
</tr>
<tr>
<td>Exchange (101.2GB, 0.46)</td>
<td>MLC</td>
<td>RAID-5</td>
<td>4055K</td>
<td>13.8KB</td>
<td>0.78</td>
<td>0.77</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td></td>
<td>eSAP</td>
<td>535K</td>
<td>104KB</td>
<td>0.74</td>
<td>0.71</td>
<td>0.69</td>
</tr>
<tr>
<td></td>
<td>TLC</td>
<td>RAID-5</td>
<td>3508K</td>
<td>19.7KB</td>
<td>0.66</td>
<td>0.65</td>
<td>0.64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>eSAP</td>
<td>364K</td>
<td>190KB</td>
<td>0.61</td>
<td>0.56</td>
<td>0.54</td>
</tr>
<tr>
<td>MSN (29.7GB, 0.96)</td>
<td>MLC</td>
<td>RAID-5</td>
<td>1903K</td>
<td>16.3KB</td>
<td>0.63</td>
<td>0.66</td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td></td>
<td>eSAP</td>
<td>406K</td>
<td>76.5KB</td>
<td>0.59</td>
<td>0.53</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>TLC</td>
<td>RAID-5</td>
<td>1527K</td>
<td>21.8KB</td>
<td>0.5</td>
<td>0.48</td>
<td>0.46</td>
</tr>
<tr>
<td></td>
<td></td>
<td>eSAP</td>
<td>341K</td>
<td>97.8KB</td>
<td>0.35</td>
<td>0.28</td>
<td>0.24</td>
</tr>
</tbody>
</table>

Fig. 7: Analysis of parity overhead for Financial, Exchange, and MSN for (a) MLC and (b) TLC technologies. The numbers on top of each bar represent the accumulated total (in millions) of the components that comprise the bar.

50 and 200 ms. These values were chosen based on the observation of inter-arrival time of requests for our workloads. We observed that essentially all inter-arrival times are 300 ms or less, hence, we chose a small value of 50 and a large value of 200 that lies between 0 and 300.

In this paper, however, we only present performance results for when $P_{\text{wait}} = 200 \text{ ms}$ in the interest of space and as the results for $P_{\text{wait}} = 50 \text{ ms}$ show similar trends. Interested readers should see the preliminary version of this paper where some results for $P_{\text{wait}} = 50 \text{ ms}$ with MLC SSD are reported [2].

Fig. 6 shows the average response times for the various RAID schemes for both MLC and TLC. In the figure, the x-axis denotes all the evaluated schemes per workload, while the y-axis represents the average response time in milliseconds. The stripe size for the schemes are given as values in parentheses after the name of the scheme. For example, RAID-5(8) and eSAP(8) represent the RAID-5 and eSAP schemes with stripe size 8. Note that stripe sizes 16 and 32 are logical stripe sizes that are composed of two and four physical stripes, respectively, as the experimental SSD configuration has 8 flash memory chips. This means that a logical stripe of, say 16, is composed of two physical stripes of size 8. Of these, one stripe would not have a parity, while the other would have one parity strip. Also, when a write of a logical stripe of size 16 is made, two separate physical writes are being made. Hence, the performance numbers reported for logical stripe sizes 16 and 32 are conservative values compared to their would-be physical stripe size counterparts. However, the disparity in the logical and physical stripes do not affect the P/E cycle or UPER numbers that we report later so long as the same amount of data is being written.

In Fig. 6, we observe that the ECC scheme shows the best performance among the three schemes. This is because it does not handle parities. RAID-5, on the other hand, shows the worst performance as the overhead for handling parity writes is manifested. Performance of the eSAP scheme lies in between ECC and RAID-5, but is significantly better than RAID-5. We also see that the performance of eSAP improves as the stripe size increases. Consequently, the average response time of eSAP with stripe size 32 is only slightly worse than that of ECC. In contrast, for RAID-5, increasing the stripe size has only marginal effects as it consolidates requests only when they fall into the same logical stripe.

The numbers in Table 2 explains why the performance between RAID-5 and eSAP differs. Note from the table that the average write request size for eSAP is considerably larger than that of RAID-5. The reason for this is because eSAP dynamically constructs a stripe with all the requests that arrive within $P_{\text{wait}}$, consolidating them to a single request. This results in eSAP having a significantly lower number of write requests than RAID-5. Between the MLC and TLC technologies, the major difference is the absolute response time, while the trend is similar.

The conclusion of the performance evaluation is that
elastic striping and anywhere parity can significantly reduce parity overhead. This results in eSAP performing in par with ECC, while providing RAID-5 reliability. Since the workings of the parity, we present a detailed analysis of parity management in the next subsection.

4.3 Analysis of Parity Overhead

The various components involved in managing the parity are shown in Figs. 7(a) and 7(b). For RAID-5, the parity overhead consists of page reads needed for parity calculations denoted PR and the parity writes for write requests denoted PW_WR. In contrast, eSAP does not require pages to be read during parity management. However, parity writes are invoked during the cleaning process. We distinguish these writes and denote them PW_GC. Additionally, eSAP may write multiple parities within a single stripe. Again, we distinguish these parity writes and the original parity write required per stripe. The former is denoted PPW (Partial-stripe Parity Writes), while the latter is denoted PW_WR. While each bar in Figs. 7(a) and 7(b) indicate the relative portions of the various components comprising the bar, the values on top of each bar represents the accumulated total of all such components (in million units). For example, for the Financial workload in Fig. 7(a) we see that for RAID-5(8) roughly 72% of the 17 million, that is, 12.24 million parity operations are page reads for parity calculations, while the other 28%, that is, 4.76 million are parity writes.

We observe distinct trends for each scheme. First, for RAID-5, as the stripe size increases, the PW_WR portion becomes smaller. This is because the number of parity blocks becomes smaller relative to the number of write requests. The PR portion, though, increases because more of the existing data has to be read to calculate the parity. Specifically for the Financial workload, we see that PW_WR and PR are affected less by the different stripe sizes. This is because the request size is much smaller than the other workloads.

Next, for eSAP, as the stripe size increases a smaller number of full stripe parities are written for writing the same amount of data. This results in decreased PW_WR, which in turn decreases PW_GC. In contrast, the PPW portion grows with increased stripe size. We also observe some partial parity overhead as the workloads include requests that have inter-arrival times larger than 200 milliseconds. Note that the substantially large PPW values for Financial and MSN are due to the relatively small number of GCs that occur, especially with TLC, not that it increases in absolute frequency.

5 Reliability Analysis

In this section, we validate our performance and reliability model. This is done by comparing values obtained through our model with those of the experiments. Based on the validation, we then use our model to project the long-term reliability of the various RAID schemes.

5.1 Performance and Model Validation

Fig. 8 shows the P/E cycles for the MLC and TLC flash memories for the various workloads and RAID schemes obtained through the experiments. We observe that Fig. 8(a) and Fig. 8(b) show similar trends, only the absolute values being different. While performing these experiments, data that represent the characteristics of the workloads are collected. Specifically, the number of write requests, the average request size, and the average utilization of victim blocks selected for garbage collection are obtained as shown in Table 2.

We estimate the P/E cycles and total number of page writes to process the given workload by applying the numbers in Table 2 to our model. Table 3 shows the accuracy ratio between the numbers obtained through the experiments and the numbers estimated from the model. From the table, we see that the model does a good job of estimating the P/E cycles and the total number of pages written by the various RAID schemes. The results show that, for the majority of the cases, the accuracy of the model versus the experiments is greater than 80%. The results here tell us that our model is a good indicator of the performance and wear out level of flash memory storage systems for the various RAID schemes.

5.2 Reliability Analysis

In this section, we discuss the bit error rate metric, which is a important reliability metric used for flash memory storage when a particular error recovery technique is applied. Before getting to the bit error rate, let us start our discussion with the assumption that ECC can correct \( k \) bits errors, while being able to detect \( 2k \) bit errors. Such an assumption is generally valid as coding theory suggests that the number of errors that can be detected by ECC is roughly twice the number of errors that can be corrected. Limiting our discussion to ECC, if ECC can correct \( k \) errors, then the rate that less than \( k + 1 \) bits fail, which is the Correctable Page Error Rate (CPER), can calculated as follows [7], [10], [13]:

\[
CPER(n, k) = \sum_{i=0}^{k} \binom{n}{i} \cdot RBER^i \cdot (1 - RBER)^{n-i}
\]

Then, the rate that \( k + 1 \) or more bits fail, that is, the Uncorrectable Page Error Rate (UPER) becomes
TABLE 3: Accuracy ratios of model compared to the experimentally obtained numbers (P/E cycles and the total number of page writes (TPW)).

<table>
<thead>
<tr>
<th>Workload</th>
<th>Flash</th>
<th>Type</th>
<th>ECC</th>
<th>RAID-5(8)</th>
<th>RAID-5(16)</th>
<th>RAID-5(32)</th>
<th>eSAP(8)</th>
<th>eSAP(16)</th>
<th>eSAP(32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Financial</td>
<td>MLC</td>
<td>P/E</td>
<td>0.94</td>
<td>0.83</td>
<td>0.85</td>
<td>0.86</td>
<td>0.82</td>
<td>0.87</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TPW</td>
<td>0.97</td>
<td>0.84</td>
<td>0.86</td>
<td>0.87</td>
<td>0.83</td>
<td>0.89</td>
<td>0.92</td>
</tr>
<tr>
<td>Exchange</td>
<td>MLC</td>
<td>P/E</td>
<td>0.96</td>
<td>0.84</td>
<td>0.86</td>
<td>0.87</td>
<td>0.84</td>
<td>0.9</td>
<td>0.92</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TPW</td>
<td>0.96</td>
<td>0.84</td>
<td>0.86</td>
<td>0.87</td>
<td>0.85</td>
<td>0.89</td>
<td>0.91</td>
</tr>
<tr>
<td></td>
<td>TLC</td>
<td>P/E</td>
<td>0.96</td>
<td>0.84</td>
<td>0.86</td>
<td>0.87</td>
<td>0.84</td>
<td>0.87</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TPW</td>
<td>0.96</td>
<td>0.84</td>
<td>0.86</td>
<td>0.87</td>
<td>0.84</td>
<td>0.89</td>
<td>0.91</td>
</tr>
</tbody>
</table>

(a) UPER (MLC)  
(b) UPER (TLC)  
(c) P/E cycles (MLC)  
(d) P/E cycles (TLC)

Fig. 9: Comparison of UPER and P/E cycles for ECC, RAID-5, and eSAP-RAID with MLC and TLC SSDs.

\[ \text{UPER}(n, k) = 1 - \text{CPER}(n, k) \]  \hspace{1cm} (7)

Before discussing the bit error rate of RAID schemes, we need to make clear the fundamental assumption behind RAID systems. That is, RAID systems work only when it can recognize that particular disks in a stripe have errors when they are at fault. Without this knowledge, it cannot correct errors even though it may have redundant information. Unfortunately, ECC does not guarantee error detection when there are more than 2k bit errors in a page. To get around this, we may use page read errors reported by the flash memory chips or employ supplementary techniques such as CRC checks. However, in this study, we make a conservative assumption that flash memory storage using RAID architecture can correct errors with parity only when ECC can detect the occurrence of errors, that is, only when a page has bit errors less than or equal to 2k. Uncorrectable Page Error Rate (UPER) is derived based on this assumption.

In a previous study, Lee et al. derive the UPER of an SSD for the RAID-5 scheme [13], which we denote UPERSTR, as follows.

\[ \text{UPERSTR}(N) = \frac{1 - \sum_{i=0}^{2k} \binom{n}{i} \cdot (1 - p)^{N - i}}{N} \]  \hspace{1cm} (8)

where \( F(x; N, p) = \sum_{i=0}^{x} \binom{n}{i} \cdot p^i \cdot (1 - p)^{N - i} \), which is the cumulative binomial distribution, and \( N \) is the number of pages in a stripe. This derivation, however, does not take into account the fact that bit errors of more than 2k may not be detected. To take this into account, we derive the Correctable Stripe Error Rate (CSER), \( \text{CSERSTR}(N) \), defined as \( F(1; N, \text{UPER}(n, k)) \) in Eq. 8.

Specifically, there are two cases where bit errors in a stripe can be corrected. The first case is when all pages comprising a stripe have less than or equal to \( k \) bit errors. For this case, errors can be corrected by the ECC, and the probability for this case to occur is \( \binom{N}{k} \text{PER}(n, k)^k \). The other case is when \( N - 1 \) pages in a stripe have less than or equal to \( k \) bit errors, which can be corrected by the ECC, and one page has more than \( k \) bit errors but less than or equal to \( 2k \) bit errors, which is detectable by the ECC. The probability of this case to occur is \( \binom{N}{k} \text{PER}(n, k)^{N-1} \cdot \sum_{i=k+1}^{2k} \binom{n}{i} \text{PER}^i(1 - \text{PER})^{n-i} \).

Note that in this derivation, we are excluding situations where there are more than \( 2k \) bit errors in a page. As noted previously, these errors may not be detected by the ECC and as a result, the RAID scheme cannot correct those errors.

Then, summing the two cases together, we get \( \text{CSERSTR}(N) \) as follows:

\[ \text{CSERSTR}(N) = \binom{N}{0} \text{PER}(n, k)^N + \binom{N}{1} \text{PER}(n, k)^{N-1} \cdot \sum_{i=k+1}^{2k} \binom{n}{i} \text{PER}^i(1 - \text{PER})^{n-i} \]

Then, we get \( \text{UPERSTR}(N) \) by replacing \( F(1; N, \text{UPER}(n, k)) \) in Eq. 8 with \( \text{CSERSTR}(N) \) as follows.

\[ \text{UPERSTR}(N) = \frac{1 - \text{CSERSTR}(N)}{N} \]  \hspace{1cm} (9)
5.3 Reliability Projection for MLC and TLC Flash

With our model, we now project the long-term reliability of SSDs employing MLC and TLC flash memory by extending the number of write requests for the given workloads. Using Eqs. 7 and 9, we depict, in Fig. 9(a) and 9(b) for MLC and TLC, respectively, the UPER values for the SSD that employ ECC, RAID-5, and eSAP. (For brevity, we denote UPER$_{STR}$ as UPER, hereafter.) Note that a subset of Fig. 9(a) was presented as Fig. 1(a).

In generating the graphs with Eqs. 7 and 9, different configurations are used for MLC and TLC as shown in Table 1 as the two technologies have different characteristics. Also, the parameters used for the analytic models of RAID-5 and eSAP are presented in Table 2. Recall again that RAID-5 and eSAP make use of different techniques to consolidate consecutive write requests resulting in eSAP providing average write sizes that are roughly 5 to 10 times larger than RAID-5. Recall also that this affects the number of write requests of the two schemes. Also, note that TLC SSD is of larger capacity and supports a larger page size, which results in different values between MLC and TLC. In Fig. 9, we depict the analytic results using the parameters obtained from the Financial workload.

The results in Figs. 9(a) and 9(b) show that ECC has the highest UPER. This is because it is not protected by any parity. However, the reliability gap between ECC and RAID-5/eSAP gradually diminishes as more data are written. This is because both RAID-5 and eSAP wear out faster as the number of P/E cycles to write parities increases. Among the three schemes, the eSAP scheme has the lowest UPER and is the slowest to reach the error rate of HDDs. Note that MLC and TLC show similar trends while MLC allows more bytes to be written before reaching the HDD error rate limit.

Making use of eSAP provides the lowest UPER and yet, the number of P/E cycles required is also far lower than that of RAID-5 for both MLC and TLC, as depicted in Figs. 9(c) and 9(d), respectively. These figures are obtained by making use of Eqs. 3 and 6. Fig. 1(b) is a subset of Fig. 9(c). The figures show that the number of P/E cycles for eSAP is slightly higher compared to ECC.

For MLC, except for eSAP(8), that is, eSAP with a small stripe size, the number of P/E cycles remains well below the MLC P/E cycle limit for up to 300TB of data writes. For TLC, however, the stripe size needs to be large for the eSAP P/E cycle to be in line with ECC. Compared to RAID-5, however, eSAP incurs far less P/E cycles.

6 FLEXIBILITY OF eSAP

An inherent merit of eSAP is that it allows for adaptive data protection through various policies as the stripe size is elastic and the parity can be placed anywhere. In this section, we present two policies that make use of this flexibility. First, we consider dynamic stripe sizing, a method that adjusts the stripe size as wear of blocks increases. The other, which we denote as Meta-eSAP, targets specific types of data, in particular, metadata, for higher reliability to reduce data loss probability.

6.1 Dynamic Stripe Sizing

Dynamic stripe sizing is a notion of altering the stripe size as needed. For example, we can reduce the stripe size so that reliability is enhanced as the P/E cycle of blocks in the SSD reaches a certain threshold.

To see the effect of dynamic stripe sizing, we consider two scenarios as examples and quantify the changes that occur on the UPER and P/E cycles for the TLC SSD. In the first scenario, which we refer to as eSAP(DYN1), eSAP starts with a stripe size of 32, then when 150TB of data is written the stripe size is reduced to 16 for 100TB of data writing. Then, starting at the 250TB point until the 300TB data write point the stripe size is reduced to 8. In the second scenario, eSAP(DYN2), eSAP starts with a stripe size of 32 until 250TB of data is written. From then on, for the next 50TB of data written, the stripe size is set to 8.

Figure 10 shows the effects of changing the stripe size on UPER and P/E cycles relative to eSAP with fixed stripe sizes. Specifically, in Figure 10(a) we see that at the 300TB data write point, the P/E cycles for eSAP(DYN1) is 17% lower compared to eSAP(8), while it is 6% higher relative to eSAP(32). For eSAP(DYN2), the P/E cycles is 20% lower relative to eSAP(8) and 4% higher relative to eSAP(32).

For UPER, Figure 10(b) shows the changes in both the linear and log scales. Overall, the (embedded) log
Fig. 12: Comparison of (a) Avg. response time and (b) P/E cycles.

(a) Avg. response time
(b) P/E cycles

Financial Exchange MSN
Avg. response time (ms)

1TB 50TB 100TB 150TB 200TB 250TB 300TB
Expected Avg. P/E cycles
ECC
Meta-Sep
Meta-eSAP
eSAP(8)
Written bytes
P/E cycle limit (TLC)

(a) P/E cycles
(b) UPER

Fig. 13: Comparison of (a) P/E cycles and (b) UPER.

6.2 Selective Protection via Meta-eSAP

In this subsection, we assess the capability of eSAP to selectively protect data. In particular, we present a derivation of eSAP that we call Meta-eSAP. Data stored in SSDs can be classified into metadata and user data. Metadata is a set of data used by storage management software such as a file system or FTL, while user data are those generated by users. As reliability of metadata is critical in maintaining consistency of a file system, Meta-eSAP treats metadata as more reliability-sensitive data and applies parity protection to metadata only. By eliminating the parity management overhead for user data, Meta-eSAP can significantly reduce P/E cycles, while still supporting reliability comparable to eSAP for the metadata.

We implement Meta-eSAP into our experimental platform by flagging the I/O traces in particular bands, selected based on actual observations, to indicate that they are metadata. We take this approach as metadata such as the superblock, group descriptors, inodes, and journals typically access particular bands. These traces are used as inputs for the Meta-eSAP simulator. This flagging approach is feasible in real systems as flags can be delivered to SSDs through the SCSI Group Number field of the SCSI commands [38] or the data tagging mechanism provided in the eMMC specifications [39].

Fig. 11 shows the data and metadata distributions for the three workloads that we use. In each graph, the dark bands represent the sector range for the metadata, while the gray area represents the user data. The white area corresponds to the sector ranges that the workloads do not access. We find that the sector ranges used by metadata are relatively small, that is, 23%, 12%, and 16% of all accessed ranges for Financial, Exchange, and MSN, respectively. Also, we observe that metadata references have strong spatial locality.

Fig. 12 shows the average response time and P/E cycles under the ECC and Meta-eSAP with stripe size 8 (and Meta-Sep, which we elaborate on later). While both MLC and TLC configurations show similar trends, this figure shows results for TLC. Compared to ECC, as Meta-eSAP incurs parity protection overhead for metadata, response time degrades slightly by within 8%.

As for P/E cycles, overhead for parity protection in Meta-eSAP should be cause for higher P/E cycles compared to ECC. However, we find in Fig. 12(b) that the results are counter-intuitive; Meta-eSAP actually outperforms ECC in terms of P/E cycles. To understand this, we performed sensitivity analyses revealing that Meta-eSAP brings about another positive feature, that is, natural separation of hot/cold data through metadata and user data segregation. This results in more efficient garbage collection, leading to reduced P/E cycles.

To validate our analysis, we implement a scheme that we call Meta-Sep (for metadata separation) that manages metadata and user data on separate blocks, while not applying parity protection. Comparing Meta-eSAP with Meta-Sep in Fig. 12(b), we see that Meta-Sep does best. (In fact, Meta-Sep does best in terms of response time as well.) Thus, we verify that Meta-eSAP not only enhances metadata reliability through parity, it allows for reduced P/E cycles compared to ECC through data segregation.

Finally, Fig. 13 shows the long-term reliability projection of Meta-eSAP in comparison to ECC, Meta-Sep, and eSAP. These results are obtained through the model described in Section 5 using parameters obtained in the simulation experiments. The results show that Meta-eSAP can reduce the UPER of metadata making file systems that use Meta-eSAP more reliable.

7 Conclusion

Though employing RAID-5 architecture inside SSDs increases the reliability of SSDs, the parity update mechanism in conventional RAID-5 introduces more writes...
resulting in increased P/E cycles and hence, higher bit error rates. To remedy this problem, we proposed a novel flash-aware RAID scheme, which we call Elastic Stripping and Anywhere Parity RAID or eSAP-RAID, that dynamically constructs stripes based on the arrival order of write requests and places the parity anywhere in the stripe so as to reduce parity update overhead and to increase reliability. In particular, in this study, we derive the performance and lifetime models of SSDs employing the RAID-5 and eSAP-RAID schemes. The models estimate write performance and P/E cycles for given workloads such that long-term reliability can be projected. Through experiments with various workloads, we evaluated the RAID schemes for both MLC and TLC based SSDs in terms of performance and reliability.

We validate the models that we derive with values obtained from the experiments, and our results show that the experimental results and the model results match each other well. Then, using the models we estimate the long-term reliability of SSDs employing RAID schemes. We also show that due to its flexible nature, eSAP can be deployed in particular reliability enhancing ways. Dynamic stripe sizing and selective data protection, in particular, Meta-eSAP, are presented as examples.

Our conclusion is that by making use of eSAP-RAID, SSD performance and wear out can be maintained at the current ECC based SSD level, while retaining reliability at a level substantially better than that of ECC throughout the lifetime of the SSD.

ACKNOWLEDGEMENT

We would like to thank the anonymous referees for their constructive comments that helped to improve the presentation of the paper. This research was supported in part by Seoul Creative Human Development Program funded by Seoul Metropolitan Government (No. HM120006), by the National Research Foundation of Korea (NRF) (No. 2012R1A2A2A01045733), and by Basic Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0025282).

REFERENCES


FURTHER READING


[34] Micron, “TLC NAND Flash Memory Features,” MT29F64G08EBAA, 2011.

Jaeho Kim received the BS degree in information and communications engineering from the Inje University, Gimhae, Korea, in 2004, and the MS degree in computer science from the University of Seoul, Seoul, Korea, in 2009. He is currently a PhD candidate in computer science, the University of Seoul. His research interests include storage systems, operating systems, and computer architecture.

Eunjae Lee received the BS degrees in computer science from the University of Seoul, Seoul, Korea, in 2011. He is currently a MS student in computer science, the University of Seoul. His research interests include flash memory and server storage technologies such as RAID, SSD cache.

Jongmoo Choi received the BS degree in oceanography from Seoul National University, Korea, in 1993 and the MS and PhD degrees in computer engineering from Seoul National University in 1995 and 2001, respectively. He is an associate professor in the Department of Software Science, Dankook University, Korea. Previously, he was a senior engineer at Ubiquix Company, Korea. He held a visiting faculty position at the University of California, Santa Cruz from 2005 to 2006 and Carnegie Mellon University, from 2104 to 2015. His research interests include micro-kernels, file systems, flash memory, and embedded systems.

Donghee Lee received the MS and PhD degrees in computer engineering, both from Seoul National University, Seoul, Korea, in 1991 and 1998, respectively. He is currently a professor in the Department of Computer Science and Engineering, University of Seoul, Korea. He held a visiting faculty position at the University of California, Irvine from 2008 to 2009. His research interests include operating systems, storage systems, and embedded systems.

Sam H. Noh received the BS degree in computer engineering from the Seoul National University, Seoul, Korea, in 1986, and the PhD degree from the Department of Computer Science, University of Maryland, College Park, MD, in 1993. He held a visiting faculty position at the George Washington University, Washington, DC, from 1993 to 1994 before joining Hongik University, Seoul, Korea, where he is now a professor in the School of Computer and Information Engineering. From August 2001 to August 2002, he was also a visiting associate professor with the University of Maryland Institute of Advanced Computer Studies (UMIACS), College Park, MD. He has served as General Chair, Program Chair, and Program Committee Member on a number of technical conferences and workshops including the ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES), IEEE International Conference on Parallel and Distributed Systems (ICPADS), USENIX Conference on File and Storage Technologies (FAST), and International World Wide Web (WWW) Conference. He also serves as Associate Editor of the ACM Transactions on Storage. His current research interests include operating system issues pertaining to embedded/computer systems. He is a member of the ACM, IEEE, USENIX, and KIISE.

0018-9340 (c) 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.