Impact of Stripe Unit Size on Performance and Endurance of SSD-Based RAID Arrays

Farzaneh Rajaei Salmasi Hossein Asadi Majid GhasemiGol
rajaei@ce.sharif.edu asadi@sharif.edu ghasemigol@ce.sharif.edu
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran

***Abstract*—Over the past decades, *Redundant Array of independent Disks* (RAIDs) have been configured based on mechanical characteristics of *Hard Disk Drives* (HDDs). With the advent of *Solid-State Drives* (SSDs), such configurations such as stripe unit size can be far from the characteristics of SSDs. In this paper, we investigate the effect of stripe unit size on the endurance and the overall I/O performance of an SSD-based RAID array and compare the optimal stripe unit size with the suggested stripe unit sizes for HDD-based RAID. To this end, we first examine the number of extra page reads and writes imposed by write requests and then observe the corresponding impact on the overall throughput and the average response time of SSD-based RAID arrays. The effect of extra page writes for different stripe unit sizes and their impact on endurance has been also examined. To validate the analytical study, we have used I/O intensive traces and simulated an SSD-based RAID array using DiskSim simulator with different stripe unit sizes. The experimental results reveal that unlike HDD-based RAID arrays, a 4KB stripe unit size can significantly improve the throughput, response time, and endurance of an SSD-based RAID4 array (up to 67.6%, 52.2%, and 48.6%, respectively) as compared to 128KB stripe unit size.**

***Index Terms*—Solid-State Drive (SSD), Performance, Endurance, RAID, Stripe Unit Size.**

I. Introduction

In recent years, NAND flash-based *Solid-State Drives* (SSDs) have gained much attention as a suitable replacement for *Hard Disk Drives* (HDDs). By employing electronic parts instead of mechanical parts, SSDs offer appealing characteristics such as light weight, shock resistance, less power consumption, and higher I/O performance. Such advantages have made SSDs a promising storage media for small to large-scale applications [1; 2; 3; 4; 5]. SSDs, however, suffer from low write performance due to slow flash programming time, limited endurance caused by erase-before-write operations, and reduced reliability due to flash device aging effect. Additionally, SSDs impose higher per bit cost as compared to HDDs. Recently, *Multi Level Cells* (MLCs) have been introduced to reduce the per bit cost and to increase the flash capacity. This is achieved by storing more than a bit in each flash unit cell. MLCs, however, suffer from higher *Bit Error Rate* (BER) and more limited erase operations as compared to *Single Level Cells* (SLCs). An MLC block wears out by 5,000 to 10,000 erases while the maximum permissible erase operations is 10 times larger for an SLC block [2].

Several techniques have been proposed in the past to address the shortcomings of SSDs, namely, slow write performance, limited endurance, and reduced reliability [6; 7; 8; 9; 10; 11]. To enhance write performance, parallel writing on multiple NAND flash chips can be used [6]. Different wear-leveling algorithms have been also proposed and applied in the aim of improving the endurance of NAND flash-based SSDs [11]. The issue of reduced reliability, however, has not been widely addressed. Since BER progressively increases by erase-write transactions, the reliability of disks decreases by each erase operation [12]. In order to mitigate the issue of increasing BER, *Error-Correction Codes* (ECCs) can be employed in SSDs [13]. While SLCs mostly use single-bit ECCs such as hamming codes, MLCs deploy more complicated ECCs due to their higher BER [14]. This will further increase the access latency of MLCs as compared to SLCs [9]. Using page-level ECC codes mitigate the increasing BER, but they are unable to provide any protection in the event of page, chip, or whole device failure.

To achieve higher level of reliability, block- or device-level redundancy techniques such as mirroring, *Simple Parity Checking* (SPC), and erasure codes can be utilized [15; 16; 17]. Device-level redundancy can be implemented using *Redundant Array of independent Disks* (RAID) [18; 17]. RAID configurations, which are widely used in data storage systems, offer higher performance, reliability, and capacity [18; 17; 19; 20; 21]. This is achieved by distributing user data across multiple disks within an array. Each level of RAID array such as RAID4, RAID5, and RAID6 can be configured using different parameters such as stripe unit size and the number of disks participated in a RAID array. Stripe unit size, which defines the granularity of data distribution in a RAID array, has been traditionally determined based on characteristics of HDDs to balance throughput and response time. Due to characteristics of HDDs, the suggested stripe unit size by enterprise data storage systems vendors such as IBM, HP, and EMC varies between 16KB up to 1MB [22; 23; 24; 25; 26; 27]. The suggested stripe unit sizes can be possibly far from the optimal configuration for SSD-based RAID arrays with respect to I/O throughput and response time. Additionally, conventional stripe unit size used for HDD-based RAIDs should be revisited with respect to limited endurance of SSDs. To the best of our knowledge, such analysis for SSD-based RAIDs is missing in

the previous work.

This paper presents an analytical study to examine the effect of stripe unit size on the endurance and performance metrics of an SSD-based RAID array, i.e., I/O throughput and I/O response time. In the proposed study, we investigate the number of extra page reads and writes and the corresponding performance overhead incurred due to write requests. Our analytical study reveals four major observations.

- Regardless of HDDs or SSDs, the larger stripe unit size can result in significant number of extra read and write operations due to parity update.
- Due to the major contribution of positioning time to the overall HDD response time, the extra read and write operations do not lead to a significant performance overhead in HDD-based RAID arrays.
- The extra number of read and write operations can directly affect both response time and I/O throughput in SSD-based RAID arrays.
- The endurance of a parity disk in SSD-based RAID arrays can be significantly affected by larger stripe unit size.

Using I/O intensive traces and a disk subsystem simulator, we have experimentally validated the observations obtained by the proposed analytical study. Both analytical and experimental studies demonstrate that choosing an appropriate stripe unit size can significantly improve the performance metrics of an SSD-based disk subsystem. In particular, the experimental results for the studied benchmarks reveal that a 4KB stripe unit size can improve the throughput and response time of an SSD-based RAID4 array up to 67.6% and 52.2% respectively, as compared to 128KB stripe unit size. In addition to the performance improvement, the results reveal that the endurance of a 4KB stripe unite size significantly reduces the number of extra page writes and consequently enhances the endurance of RAID array, as compared to 128KB stripe unit size. We have also further validated the simulation results by conducting experiments on a system equipped with 40GB SSDs. To the best of our knowledge, this is the first comprehensive work which investigates the effect of stripe unit size on both endurance and performance of SSD-based RAID arrays.

The remainder of this paper is organized as follows. In Section II, a background on NAND-based flash and SSD-based RAID configurations is provided. Section III elaborates design challenges in SSD-based RAID configurations. Section IV investigates the effect of different stripe unit sizes on performance and endurance. In Section V, the experimental results will be presented. Section VI reviews related work on SSD-based RAID arrays and lastly, Section VII presents summary and conclusion.

II. Background

A. Flash Memory

An SSD consists of few flash chips, which are organized in an interleaved manner to provide parallel access to user data. A flash chip includes one or more dies and each die contains a set of banks or planes, which in turn are constructed by blocks. Typically one page size register and 2048 blocks are

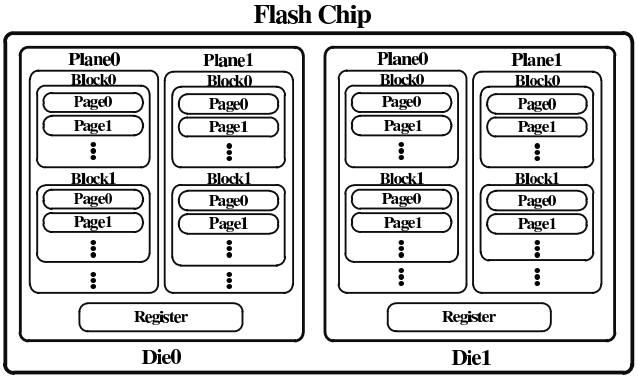

Fig. 1. A typical structure of a flash chip consists of two dies and four planes

organized in a plane and each block is composed of 64 or 128 pages, leads in hierarchical structure. An example of a flash chip with four planes is presented in Fig. 1. In a typical flash structure, dies in a plane can be accessed in parallel. The smallest parallel unit in a NAND flash chip is plane-pair.

NAND flash memory exhibits challenging characteristics such as asymmetric read/write latency, different granularity of read/write operations, and erase-before-write limitation. The asymmetric read/write latency implies that the latency of read and write accesses is not equal. Typically, a single write access takes about ten times longer than a unit-size read access. The smallest unit in both read and write accesses is a page. However, an erase operation is performed on a block level.

Another challenging characteristic of NAND flash memory is erase-before-write limitation, which implies a block should be erased before a page within the block is overwritten or updated. Therefore, updating a previously written data on the same page is not possible unless the entire bock is erased. Since the number of block erases is limited in the NAND flash technology, each block erase will reduce the total device lifetime. The limitation of the number of erase operations per block has been reported up to 10,000 and 100,000 for MLC and SLC flash, respectively [28]. To overcome the limited endurance of flash memory, wear leveling algorithms have been proposed in the literature [29; 30]. The main aim of wear leveling algorithms is to distribute write accesses across all blocks in an even manner to prolong the lifetime of a flash memory. Wear leveling algorithms are directed by a software named *Flash Translation Layer* (FTL). The main responsibility of FTL is mapping a logical page address received from the disk front-end to a physical page address in the disk flash chips.

B. Interleaving

Parallel access to user data provided by interleaving is one of the most prominent features of SSDs as compared to HDDs. Interleaving is provided in different layers in SSDs, which results in an improved performance and higher bandwidth. Fig. 2 shows building blocks of an SSD controller and a chip-level parallelism available in flash chips. In this figure, an SSD controller is connected to four flash chips using four parallel channels. Since one channel is dedicated to each flash chip, data transfer and read or write operations can be performed

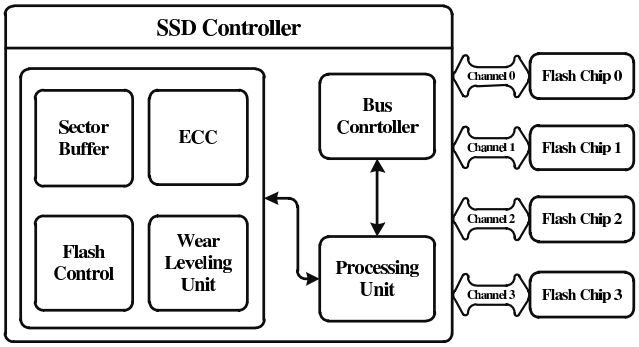

Fig. 2. Building blocks of an SSD controller and chip-level parallelism in a typical SSD

on different flashes at the same time, which resembles RAID0 configuration in the back-end flash chips. The RAID configuration, which is established on a number of flash chips within an SSD could be called *intra-RAID* as opposed to *inter-RAID*. Inter-RAID can be configured on a number of SSDs. Both architectures provide interleaving and parallelism, although there is a slight difference between these two architectures. In the intra-RAID configuration, data is serially transferred to an SSD and it is then interleaved among flash chips. In an inter-RAID array, however, data transmission from higher level to SSDs is performed simultaneously. Consequently, an inter-RAID array can provide better performance as compared to the intra-RAID configuration.

There is also another kind of parallelism among planes within a flash chip. Although there is one common channel for all planes in a flash chip, planes in a die can operate independently. In some circumstances, a pair of planes in a die can operate independent from another pair of planes in a same die. This means that data transfer is not performed simultaneously but data can be accessed in an interleaved manner on independent dies or planes [6].

C. RAID

Although RAID is commonly used for redundancy purposes, it is indeed used to enhance both reliability and performance [18; 17]. In particular, while RAID4, RAID5, and RAID6 configurations are aimed at improving both performance and reliability, RAID0 is only used to enhance performance. RAID0, which does a simple data striping across disks enhances performance and capacity while it does not improve reliability. On contrary to RAID0, RAID1 reaches higher reliability level by deploying mirroring but it does not improve performance as compared to other RAID configurations. RAID4 and RAID5 are two RAID schemes, where a space equal to one disk is allocated to parity stripes. In RAID4, the extra disk is dedicated to hold all parity bits while parity bits are evenly spread across all disks in a RAID5 array.

In RAID4 and RAID5 configurations, a part of interleaved data which is stored on a single disk, is called *stripe unit*. The *stripe unit size* defines the amount of data placed on a disk which represents the granularity of data distribution in a RAID array. Stripe unit size, which can be from a bit or a byte to multiple blocks of data, may influence the performance and/or the reliability of a RAID array [31].

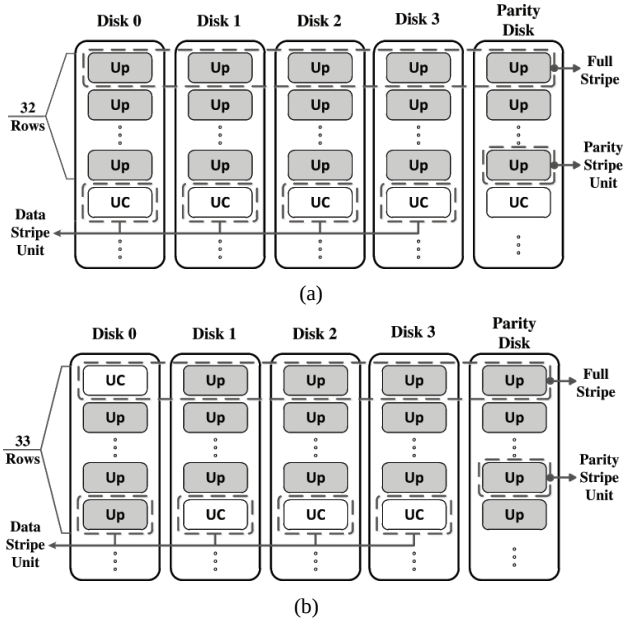

Fig. 3. Writing a 512KB request to a RAID4 4+1 with stripe unit size = 4K: (a) Full stripe update, (b) Partial stripe update (UP: Updated, UC: Unchanged)

A row of stripe units, which parity is computed over, is called a *stripe* or a *full stripe*. The computed parity is written on the parity stripe unit in the corresponding stripe. Therefore, each stripe includes data stripe units and a parity stripe unit. Data stripe unit, parity stripe unit, and full stripe have been shown in Fig. 3. In RAID4 and RAID5 configurations with N disks, there are $N-1$ data stripe units and one parity stripe unit. When an I/O request is started at the first data disk and accesses exactly an integer multiple of $N-1$ data stripe units, a full stripe access is accomplished. Otherwise, a partial stripe access within one row or multiple rows will happen [31; 19]. This will be further elaborated in Section IV.

To further achieve higher reliability levels in data storage systems, one can employ erasure codes in disk subsystems. Erasure codes such as Reed-Solomon [32; 33], X-codes [34], and Even-Odd [16] integrate m data disks and n redundant disks in order to tolerate up to n disk failures. These erasure codes are referred as RAID6 configuration. Majority of erasure codes are based on complex XOR and arithmetic operations [16; 32; 33]. There are several parameters such as word size, the number of words in a stripe unit, and the number of operations that are involved in data encoding of complex erasure codes. One important parameter of erasure codes is word size, where each stripe unit is partitioned into words. The effect of word size on the performance of erasure codes has been investigated in several studies, which will be discussed in Sec. VI. In this work, we only investigate the impact of stripe unit size in RAID arrays employing *Single Parity Checking* (SPC). Investigation of RAID6 configurations is beyond the scope of this work.

III. Challenges in SSD-based RAID Arrays

Despite significant reliability improvement of SSDs over HDDs, they still have limited *Mean Time To Failure* (MTTF) as reported by SSD venders [35; 36; 37]. Typical MTTF of

TABLE I
EXAMPLE: NUMBER OF PAGE WRITES IN A RAID4 4+1 ARRAY (PAGE SIZE=4KB)

	Disk Number					Total Size of Write Requests
	D0	D1	D2	D3	P	
Iozone	771K	767K	768K	767K	1439K	3,073K
Postmark	545K	546K	546K	546K	703K	2,183K
Vdbench	145K	145K	144K	145K	265K	579K

TABLE II
EXAMPLE: NUMBER OF PAGE READS IN A RAID4 4+1 ARRAY (PAGE SIZE=4KB)

	Disk Number					Total Size of Read Requests
	D0	D1	D2	D3	P	
Iozone	600K	410K	422K	498K	273K	0
Postmark	443K	416K	417K	445K	75K	1,272K
Vdbench	98K	71K	72K	97K	53K	5

SSDs varies from 1M hours up to 2M hours. Hence, a high available and reliable disk subsystem (e.g., 7- or 8-nine availability) is not achievable without using RAID configuration. Finding an efficient RAID configuration for SSDs can be challenging. Using RAID1 imposes higher cost while brings more reliability. RAID4 and RAID5 are more cost efficient but updating parity stripe units would result in fast disk aging. In RAID4 and RAID5 configurations, the parity stripe unit in each stripe is updated once its corresponding data stripe unit is modified. Consequently, parity stripe units are updated more frequently than data stripe units.

In a RAID5 array, since parity stripe units are distributed across all SSDs, each disk gets more updates and ages faster as compared to data disks within a RAID4 array. As a result, the average lifetime of SSDs in a RAID5 array is shorter than the average life time of data SSDs in a RAID4 array. It has been demonstrated in [10] that RAID5 may suffer from simultaneous disk failures. This is due to write requests are evenly distributed across all disks and as a result, all disks wear out approximately with the same rate. Similarly, disk aging with the same rate is also valid for RAID1 configuration. To alleviate this limitation, it has been suggested to use uneven parity distribution using RAID4 configuration [10; 38; 39]. In such schemes, SSDs experience different amount of writes in a RAID4 array, which results in differential aging of disk drives.

In a RAID4 array, frequent parity updates leads to fast aging of the parity disk while data disks in a RAID4 array wear out similar to data disks in RAID0. Table I shows an example for the number of page updates that data and parity disks receive in a RAID4 array for I/O intensive benchmark programs. In this example, the array includes five SSDs (four data disks and one parity disk) and the stripe unit size is set to 128KB. Note the total size of read and write requests and the other numbers reported in Table I and Table II are in terms of number of pages (a page size=4KB). It can be observed from Table I that the parity disk receives updates about twice as data disks do for Iozone and Vdbench benchmark programs. Consequently, the parity disk wears out with a higher rate and fails sooner than data disks.

As illustrated by an example in Table I, the parity disk fast aging in RAID4 is a major disadvantage, which should be properly addressed in SSD-based RAID arrays. Another shortcoming of SSD-based RAID4 configuration is that the parity disk may become performance bottleneck. Read accesses to the parity disk can be considerable since parity should be computed in each write access. When a write request is distributed across disks in a full stripe manner, no read access is required to compute the new parity. However, in case a partial stripe is overwritten, reading of unchanged stripe units or the parity stripe unit within a stripe is necessitated.

In the case of partial stripe update with the number of stripe units equal or less than half of data disks, it is more cost effective to read the old data and the old parity rather than the unchanged data within the stripe [9; 28]. To further clarify these statements, let's consider a RAID4 4+1 array with four data disks (D0, D1, D2, and D3) and one parity disk (P). In this array, a write to a full stripe (D0$\sim$D3) does not imply any read access to generate the new parity. However, a write to a partial stripe will require read accesses. For example, a write access to (D0$\sim$D2) will call to read data on D3 for the new parity computation. A write access to a single stripe unit (e.g., data on D0) can be followed by either of the following read accesses for parity generation. A straightforward solution is to read data on D1 through D3, which requires three extra read accesses. Another way is to read the old data on D0 and the old parity, which implies two extra read accesses. In the latter case, old data on D0 and the old parity are first read simultaneously. Then, the new parity is computed by performing exclusive OR between the old data on D0, the new data for D0, and the old parity. The new data for D0 and the new parity are written on disks at the same time.

In a RAID array, as explained before, some write requests incur extra read accesses to data or parity disks within the array. This overhead can increase the latency and degrade the performance of the RAID array. Table II shows the number of page reads due to write requests in an SSD-based RAID4 4+1 for sample I/O intensive workloads. For Iozone and Vdbench workloads, although there is no or few read requests in the input trace, the number of page reads is notable, which can result in throughput degradation.

IV. STRIPE UNIT SIZE ANALYSIS

In this section, we investigate the effect of different stripe unit sizes on the endurance and performance of an SSD-based RAID array utilizing a parity disk. In our analysis, we mainly focus on the number of page reads and page writes imposed by write accesses rather than those imposed by read accesses. This is due to the parity disk is not invoked in read accesses and as a result, no extra read or write operations take place on read accesses. Additionally, the endurance of RAID arrays is not affected by read accesses.

The analysis of the number of extra page reads and writes provided hereafter is independent of storage type (either HDD or SSD) used in a RAID array. The performance impact of the extra page reads and writes on SSD-based RAID, however,

can be very different from HDD-based RAID since HDDs and SSDs exhibit different characteristics. In particular, while the major contribution of positioning time to the overall response time in HDDs can alleviate the performance penalty of extra page reads and writes, this does not apply in SSDs as they do not have any moving parts. As an example, in case of write accesses to an SSD-based RAID, as it will be shown in the next subsections, both performance and endurance can be significantly affected with different stripe unit sizes.

In the rest of this section, we first investigate the impact of stripe unit size on the number of extra page reads in write requests. Then, the effect of stripe unit size on the number of extra page writes in write requests is presented next. The impact difference of extra page reads and writes between SSD and HDD RAID arrays will be discussed in the subsequent subsection. To provide better understanding of the impact of stripe unit size on extra page reads and writes, we use RAID4 in the analysis provided in this section. However, this analysis is also valid for RAID5 arrays as the extra number of reads and writes does not depend on the way parity stripe units are distributed across disks.

A. Impact of Stripe Unit Size on Number of Extra Page Reads

In a RAID configuration, a logical address from upper layer is converted to a physical address involving several parameters such as *Starting Disk Index* (SDI) and the number of data disks within the array, referred as N_d. SDI which refers to the data disk number holding the first stripe unit of a request, is calculated based on *Stripe Unit Size* (SUS), request address and the number of data disks within the array. SDI is equal to (ADD_l/SUS) modulo N_d, where ADD_l is the logical address of a request. When a request is striped across data disks, two possible situations may happen: i) a row involved in a write update forms a full stripe write, which all stripe units in the row are updated, ii) a row is not invoked in a full stripe manner. We call the former *full row update* against *partially row update* in the latter case. A write can fall into one of these two cases depending on logical address, stripe unit size, and request size. When a full row update happens, since all stripe units of a row exist in the RAID controller, there is no need to read any stripe unit from disks for parity stripe generation. In contrast, in case of partially row update, since some stripe units are missed in the RAID controller, read operations from data or parity disks should be undertaken.

In order to make the subject more clear, let's consider an example of a 512KB write request written into an SSD-based RAID4 4+1 array. Let's also assume that the stripe unit has the smallest granularity and its size is a page (typically 4KB). If the logical address of this request is mapped to the first disk (Disk 0), 32 full row updates will be accomplished as shown in Fig. 3(a). On the other hand, if the logical address is mapped to any data disk other than the first disk (Disk 0), 31 full row updates and 2 partially row updates would be emerged. Fig. 3(b) shows a situation in which the first stripe unit of a request is written to the second disk (i.e., SDI=1 or Disk 1).

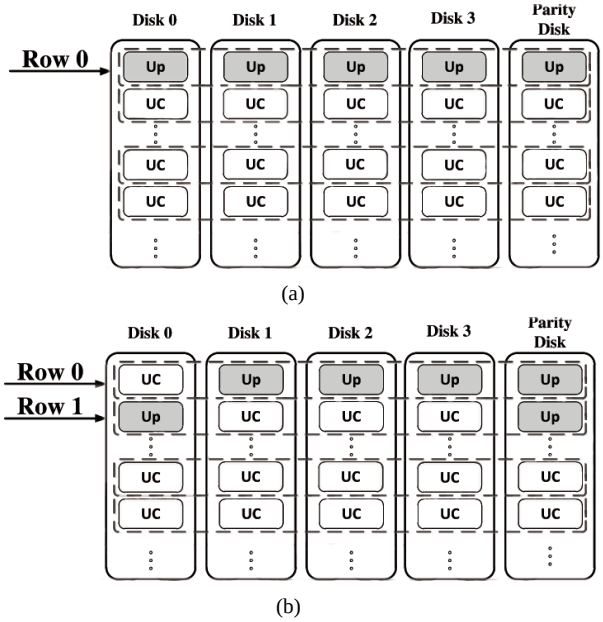

Fig. 4. Writing a 512KB request to a RAID4 4+1 with stripe unit size = 128KB: (a) Full stripe update, (b) Partial stripe update (UP: Updated, UC: Unchanged)

When data stripping leads to a full row update as shown in Fig. 3(a), since all rows are updated in a full stripe manner, no read operation is needed. In Fig. 3(b), only the first and the last rows imply read operations. In the first row, the first page is missed in the RAID controller and should be read from Disk 0. In the last row, only the first stripe unit is updated. Therefore, to compute the new parity, the old data stripe of Disk 0 and the old corresponding parity stripe should be read. Consequently, in this example, two page reads from Disk 0 and one page read from the parity disk are necessitated.

Now let us consider a larger stripe unit size equal to 128KB. In this situation, the request is divided into four stripe units, which may occupy one or two rows as shown in Fig. 4. As shown in Fig. 4(a), no read operation is needed when a write operation is performed on a full stripe. However, in case of partial stripe write, read operation from data disks and the parity disk is required. Fig. 4(b) shows an example when two data stripe units from Disk 0 and one data stripe from the parity disk should be read. Since the stripe unit size is 128KB, which is equivalent to 32 pages, totally 96 page reads is demanded. This number of page reads is significantly greater than 3 page reads that was requested when the stripe unit size was set to 4KB or one page size.

In Table III, we have analyzed the number of extra page reads performed for a 512KB write request with different stripe unit sizes and two sample starting disks (SDI=$D1$ or $D2$)[1]. It is intuitive that if the starting disk is $D0$ (not reported in Table III), no extra page reads will be incurred since all page updates will be accomplished in a full stripe manner. In case SDI is equal to $D1$, the first row of the write request will call to read from $D0$ and the last row will require to read from $D0$ and the parity disk. Hence, writing a 512KB request in a RAID4 4+1 array requires three extra stripe unit reads from

[1] $D1$ and $D2$ refer to Disk 1 and Disk 2, respectively.

$$NEPR_{(r=1)} = \begin{cases} (LDI - SDI + 2) \times NPS & \text{if } (LDI - SDI + 1) < N_d/2 \\ (N_d - LDI + SDI - 1) \times NPS & \text{otherwise} \end{cases} \quad (1)$$

$$NEPR_{(r\geq 2)} = \begin{cases} (2N_d - SDI - LDI) \times NPS & \text{if } (SDI > N_d/2) \;\&\&\; (LDI + 1 \geq N_d/2) \\ (N_d - SDI + LDI + 3) \times NPS & \text{if } (SDI > N_d/2) \;\&\&\; (LDI + 1 < N_d/2) \\ (N_d + SDI - LDI - 1) \times NPS & \text{if } (SDI \leq N_d/2) \;\&\&\; (LDI + 1 \geq N_d/2) \\ (SDI + LDI + 2) \times NPS & \text{otherwise} \end{cases} \quad (2)$$

TABLE III
NUMBER OF EXTRA PAGE READS TO WRITE A 512KB REQUEST IN A RAID4 4+1 ARRAY (PAGE SIZE=4KB)

		Number of Extra Page Reads (NEPR)					
SUS	SDI	D0	D1	D2	D3	Parity	Overall
4K	D1	2	0	0	0	1	3
	D2	1	1	1	1	0	4
8K	D1	4	0	0	0	2	6
	D2	2	2	2	2	0	8
16K	D1	8	0	0	0	4	12
	D2	4	4	4	4	0	16
32K	D1	16	0	0	0	8	24
	D2	8	8	8	8	0	32
64K	D1	32	0	0	0	16	48
	D2	16	16	16	16	0	64
128K	D1	64	0	0	0	32	96
	D2	32	32	32	32	0	128
256K	D1	64	0	0	64	0	128
	D2	64	64	0	0	0	128
512K	D1	0	128	0	0	128	256
	D2	0	0	128	0	128	256

$D0$, $D1$, and the parity disk (4KB stripe unit size: 3 pages, 8KB stripe unit size: 6 pages). In case SDI is equal to $D2$, the first row of the write request will call to read from $D0$ and $D1$ and the last row will require to read from $D2$ and $D3$. As a result, writing a 512KB request in this array requires to read four stripe units from the data disks (4KB stripe unit size: 4 page reads and 8KB stripe unit size: 8 page reads).

The analytical examples given above lead us to formulate the number of overall extra page reads caused by a write request. For simplicity, we assume that the request size is divided by stripe unit size. We also assume that the number of pages within a stripe unit is referred as *Number of Pages per Stripe* (NPS). This number is equal to $[SUS/Size_{page}]$, where $Size_{page}$ is the size of a page in flash chips. Let's also assume that the last disk in the last row accessed by a request, is referred as *Last Disk Index* (LDI). This index is equal to $[(Size_{req}/SUS)$ + SDI - 1] modulo N_d, where $Size_{req}$ is the size of the request. In case, the striping is performed only on one row (r=1), the *Number of Extra Page Reads* (NEPR), is calculated according to Equation 1. If the striping is performed on at least two rows (r $\geq$ 2), the number of extra page reads can be calculated as shown in Equation 2.

According to Equation 1, Equation 2, and the analytical results reported in Table III, as the stripe unit size boosts up, the number of page reads from data disks and the parity disk increases. The extra page reads obviously can impose performance penalty to the disk subsystem. As shown in Table III, the total number of extra page reads is increased from 3 to 256 pages when the stripe unit size is increased from 4KB to 512KB. As stated earlier, the extra number of page reads reported in this table is valid for both RAID4 and RAID5 configurations. Additionally, this analysis is independent of the type of storage device (either HDD or SSD).

B. Impact of Stripe Unit Size on Number of Extra Page Writes

Here, we analyze the effect of stripe unit size increase on the number of page writes in a write request. Since the request size is constant and it is distributed on data disks, the number of page writes on data disks does not change by the variation of stripe unit size. Therefore, no extra page writes is performed on data disks. However, the number of page writes on the parity disk can be affected by the stripe unit size.

Table IV reports the number of page writes imposed by a 512KB write request for different stripe unit sizes and different starting disk numbers. In this table, we report the number of page writes rather than the number of extra page writes. Similar to the analysis provided in the previous subsection, the number of extra page writes for the first starting disk (i.e., $SDI = D0$), would be equal to zero, since all page updates will be accomplished in a full stripe manner. Considering the data distribution shown in Fig. 3, if the granularity is a page or 4KB, there would be 33 page writes (equivalent to 516KB) on the parity disk when the starting disk is not $D0$. On the other hand, if the stripe unit size is 128KB and the starting disk is not the first disk (as shown in Fig. 4(b)), since two 128KB parity blocks will be written on the parity disk, the number of page writes on the parity disk will be increased to 64 pages. This number of page writes is about twice greater than what we had for 4KB stripe unit size. Similar to the number of extra page reads, we can extract the formulation from this example for the number of extra page writes. Here, we assume that the request size is divided by SUS and striping is performed over at least two rows of the array. Equation 3 can be used to compute the *Number of Extra Page Writes* (NEPW) on the parity disk.

$$NEPW = \lceil \frac{SDI + (Size_{req}/SUS)}{N_d} \rceil \times NPS \quad (3)$$

The total number of page writes committed to SSDs, reported in Table IV, would be equal to $NEPW$ + $(Size_{req}/Size_{page})$. The analytical results reported in Table IV, and the formulation given in Equation 3 demonstrates that the larger stripe unit size, the more write/erase operations on the parity disk, which results in

TABLE IV
NUMBER OF PAGE WRITES TO WRITE A 512KB REQUEST IN A RAID4 4+1 ARRAY (PAGE SIZE=4KB)

		Number of Page Writes					
SUS	SDI	D0	D1	D2	D3	Parity	Overall
4K	D1	32	32	32	32	33	161
	D2	32	32	32	32	33	161
8K	D1	32	32	32	32	34	162
	D2	32	32	32	32	34	162
16K	D1	32	32	32	32	36	164
	D2	32	32	32	32	36	164
32K	D1	32	32	32	32	40	168
	D2	32	32	32	32	40	168
64K	D1	32	32	32	32	48	176
	D2	32	32	32	32	48	176
128K	D1	32	32	32	32	64	192
	D2	32	32	32	32	64	192
256K	D1	0	64	64	0	64	192
	D2	0	0	64	64	64	192
512K	D1	0	128	0	0	128	256
	D2	0	0	128	0	128	256

fast aging of the parity disk. The total number of page writes is increased in this table from 161 to 256 pages when the stripe unit size is increased from 4KB to 512KB. In particular, the number of page writes on the parity disk is increased from 33 to 128 pages when the stripe unit size is increased from 4KB to 512KB. Since the parity disk in RAID4 is a performance bottleneck and page writes on flash memories are time-consuming operations, performance can be considerably degraded.

C. Extra Page Reads and Writes in SSD- vs. HDD-Based RAID Arrays

The main conclusion from the analytical study provided in the previous subsections is that as the stripe unit size becomes larger, the number of extra page reads and writes increases in both RAID4 and RAID5 configurations in either SSD- or HDD-based RAID arrays. The performance impact of extra page reads/writes needs to be further investigated. Intuitively, reading or writing $2n$ pages in SSDs takes twice as reading or writing n pages. Hence, it is expected that the extra number of page reads and writes in SSDs directly affects the performance. This will be validated in our results presented in Sec. V. Such statement, however, is not valid for HDDs due to the major contribution of positioning time in the overall response time. In general, the response time for an access to a HDD is calculated as follows:

$$T_{access} = T_{seek} + T_{rotate} + T_{transfer} \quad (4)$$

In this equation, T_{seek}, T_{rotate}, and $T_{transfer}$ are seek time, rotation time, and transfer time, respectively. We refer to positioning time as the sum of seek time and rotation time. Positioning time is generally independent from the request size and it depends on the characteristics of HDDs. Transfer time, however, depends both on HDD characteristics and the request size. Since positioning time will be a major contributor of response time for small requests, it is expected that imposing extra page page reads and page writes will not result into significant performance overhead in small requests. On the other hand when a request is large enough to be distributed on many stripes in an array, the number of extra page reads and page writes becomes negligible as compared to the total number of page reads and writes of user data. This will be validated in the results provided in Sec V-C.

V. EXPERIMENTAL RESULTS

In order to evaluate our analytical study, we have simulated SSD-based RAID4 and RAID5 4+1 arrays using DiskSim V4.0 simulator [40]. DiskSim is a disk subsystem simulator, which has been extended to support both HDDs and SSDs. In our experiments, we report the number of extra read and write pages for both RAID4 and RAID5 configurations. In this simulator, no cache memory has been used in the RAID controller for either SSDs or HDDs. The configuration of RAID controller in all experiments are the same for both HDD- and SSD-based RAID arrays. Additionally, we also report the performance metrics of SSD-based RAID arrays, i.e., throughput and response time, for both RAID4 and RAID5 arrays. In the experimental setup, we have used a common configuration previously used in other studies [4; 41; 42; 43]. This configuration has been outlined in Table V. In our simulations, the number of page reads and page writes on each disk and the average response time are measured. We have also extracted throughput by extending the source code of DiskSim. Six I/O traces have been used in our simulation, among which two are MSR Cambridge [44] block I/O traces and four are traces produced by Postmark [45], Iozone [46], and Vdbench [47] programs. These four latter traces are I/O intensive traces which were gathered by running the corresponding application programs. Table VI reports the characteristics of these traces. To further validate the proposed analytical study, we have also conducted experiments on a system equipped with three SSDs. Due to very time-consuming experiments on the physical system, we have only conducted a sample experiment on this system setup.

In the rest of this section, performance results extracted by DiskSim simulator for a SSD-based RAID array is presented in Sec. V-A and Sec. V-B. The performance simulation results for a HDD-based RAID array is presented in Sec. V-C. The endurance simulation results for a SSD-based RAID are reported in Sec. V-D. Lastly, experimental results for SSD-based RAID extracted from a system equipped with Intel SSDs are reported in Sec. V-E.

A. Effect of stripe unit size on number of page reads and writes

Fig. 5 shows the impact of varying stripe unit size on the number of extra page reads of data and parity disks due to parity generation in a RAID4 4+1 array. It can be seen that the number of extra page reads on data and parity disks significantly boosts up by increasing the stripe unit size from 4KB to 512KB. However, the number of extra page reads on data disks is declined when the stripe unit size is increased from 512KB to 1024KB. After this point, the number of extra page reads on all disks remains almost the same and it is

TABLE VI
STATISTICS OF TRACES EXTRACTED FROM IOZONE, POSTMARK, AND VDBENCH PROGRAMS; AND MSR CAMBRIDGE TRACES USED IN OUR EXPERIMENTS

Parameter	Iozone	Postmark	VdBench	Postmark2	CAMRESIRA01-lvm1	CAMWEBDEV-lvm2
Read Requests (%)	0	83.2	0.1	64	1	0
Write Requests (%)	100	16.8	99.9	36	99	100
Average Request Size (KB)	360	222	412	242	6231	4155
Average Read Request Size (KB)	–	99	4	146	6717	–
Average Write Request Size (KB)	360	833	413	405	6230	4155
Average Time Between Consecutive Write Requests (ms)	0.25	39.75	0.16	46.73	4.2	2.91
Average Time Between Consecutive read Requests (ms)	–	7.52	13.60	25.72	1.13	–
Max Write Request Size (KB)	512	4096	512	512	32768	32768

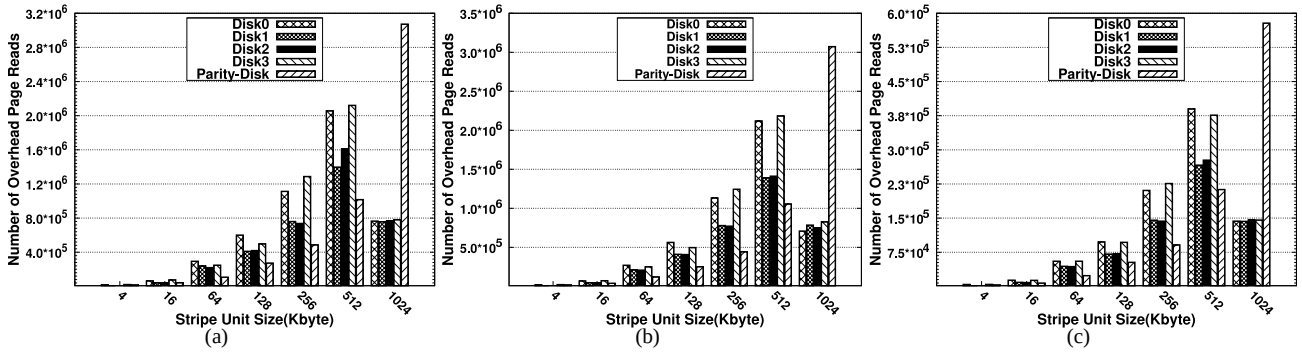

Fig. 5. Number of extra page reads by varying stripe unit size: (a) Iozone, (b) Postmark2, (c) Vdbench

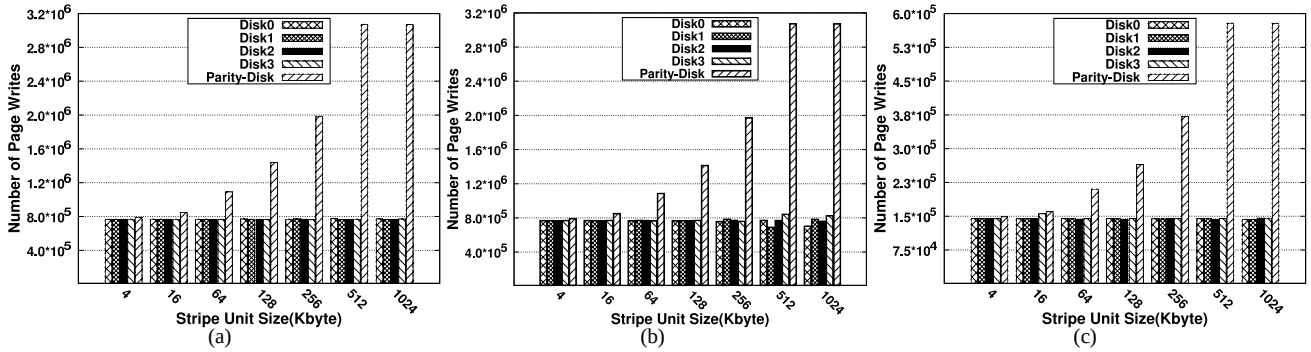

Fig. 6. Number of page writes by varying stripe unit size: (a) Iozone, (b) Postmark2, (c) Vdbench

TABLE V
EXPERIMENTAL SETUP USED IN DISKSIM

Parameter	Value	Parameter	Value
Page Size (KB)	4	Number of Pages per Block	64
Number of Blocks per Plane	2048	Number of Planes per Die	8
Number of Dies Per Flash Chip	8	SSD Size (GB)	32
Number of Parallel I/O Channels	1	Page Read Latency (ms)	0.025
Page Write Latency (ms)	0.2	Block Erase Latency (ms)	1.5
Cache Configuration		No cache used	

saturated. This point, where saturation happens, is dependent on the maximum write request size of a trace. From Table VI the maximum write request for traces used in Fig. 5 is 512KB. When the stripe unit size exceeds the maximum write request size, no distribution takes place and each write request is directed to only one disk. Consequently, for parity generation, one read operation from the target data disk and one read operation from the parity disk is requested. This means that every write request implies a page read from the parity disk and a page read from only one data disk. This will increase the number of parity page reads while it will decrease the number of data page reads. Here, one should expect that the total page reads from all data disks gets almost equal to the parity disk page reads. This has been confirmed by the results provided in Fig. 5.

We have also measured the number of page writes on data and parity disks in a RAID4 4+1 configuration to evaluate the

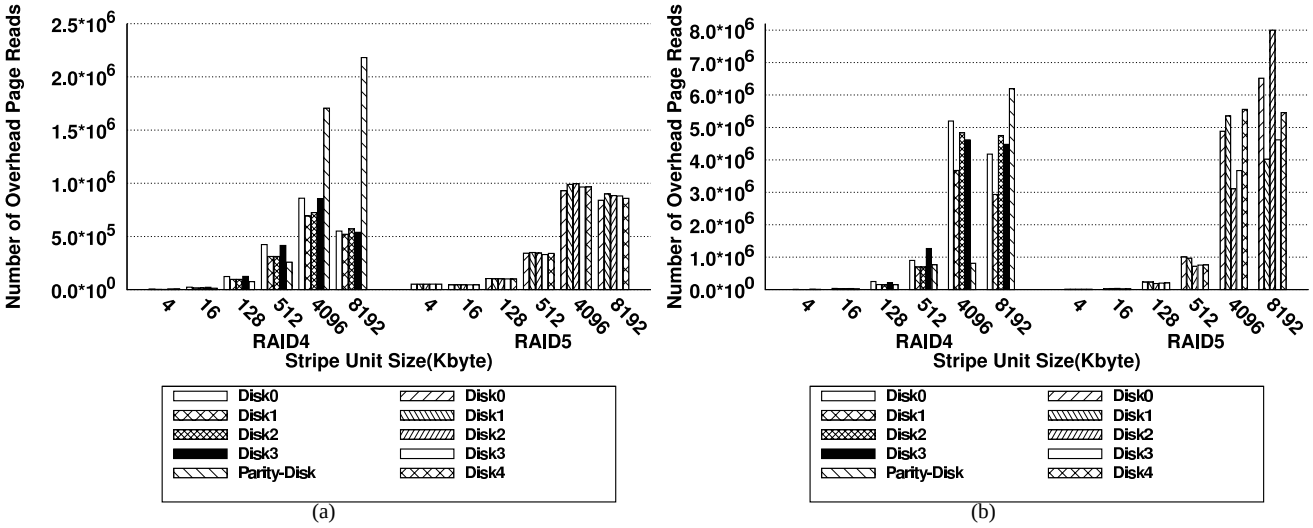

Fig. 7. Number of extra page reads by varying stripe unit size for RAID4 and RAID5: (a) Postmark, (b) CAMRESIRA01-lvm1

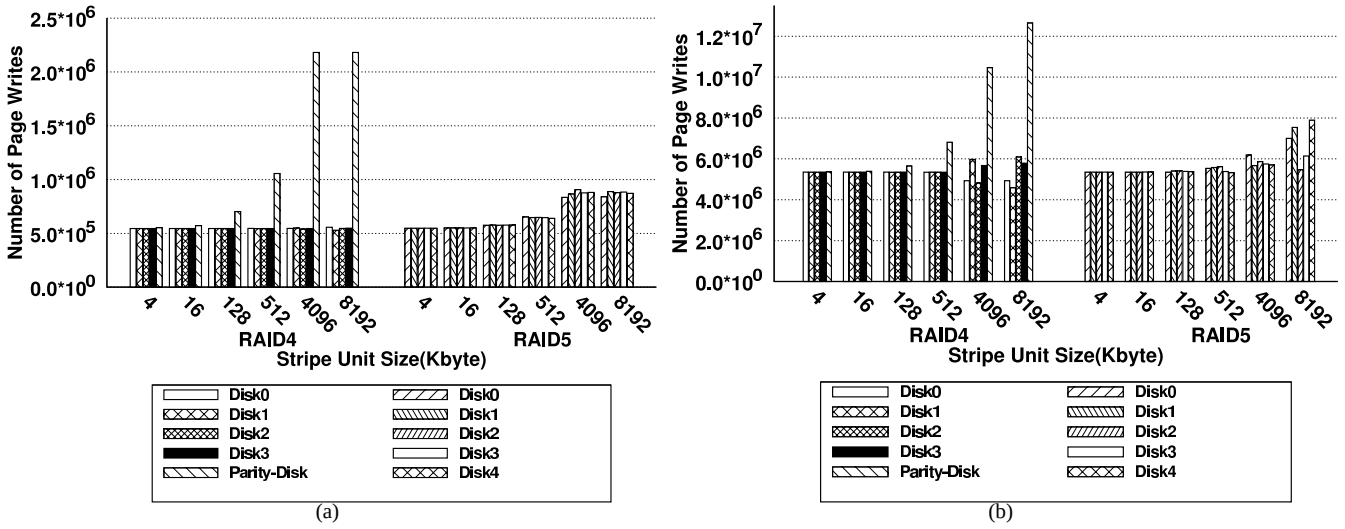

Fig. 8. Number of page writes by varying stripe unit size for RAID4 and RAID5: (a) Postmark, (b) CAMRESIRA01-lvm1

effect of varying stripe granularity on write performance and lifetime of SSDs as shown in Fig. 6. In this figure, we report the number of page writes rather than the number of extra page writes. It can be seen that the stripe unit size increment does not much affect the number of writes on data disks. This is because the amount of data that is supposed to be written on data disks is not affected by the variation of the stripe unit size. On the other hand, the number of page writes on the parity disk increases by enlarging the stripe unit size. This is in agreement with the analytical study presented in Section IV.

The incremental page writes on the parity disk deteriorates the lifetime of the parity disk since the greater number of page writes implies the greater number of erase operations on flash blocks. Additionally, the extra page writes can affect the overall performance as they increase the number of accesses to the parity disk, which is the performance bottleneck in a RAID4 configuration. For instance, when the stripe unit size is equal to 128KB, the number of writes on the parity disk is about twice greater than the number of page writes on a data disk. As it can be seen in Fig. 6, the number of page writes on the parity disk gets saturated when the stripe unit size exceeds the maximum write request size. This is due to the pattern of data distribution does not change once the stripe unit size becomes larger than the maximum size of write requests.

In Fig. 7, we compare the extra page reads for data and parity disks in RAID4 and RAID5 arrays. The total number of extra page reads in RAID5 increases similar to RAID4 with stripe unit size increment. Unlike RAID4, as it was expected, the extra page reads are evenly distributed in the RAID5 array. In a RAID4 array, page reads and page writes on the parity disk are directed to only one disk, whereas these extra transactions are distributed on all disks in RAID5. That is why one expects longer response time for a RAID4 array as compared to a RAID5 array. This will be demonstrated in the next subsection. Note the total number of page reads is almost similar in both RAID4 and RAID5 arrays. The slight difference is due to starting disk numbers in these two configurations can be different for I/O requests.

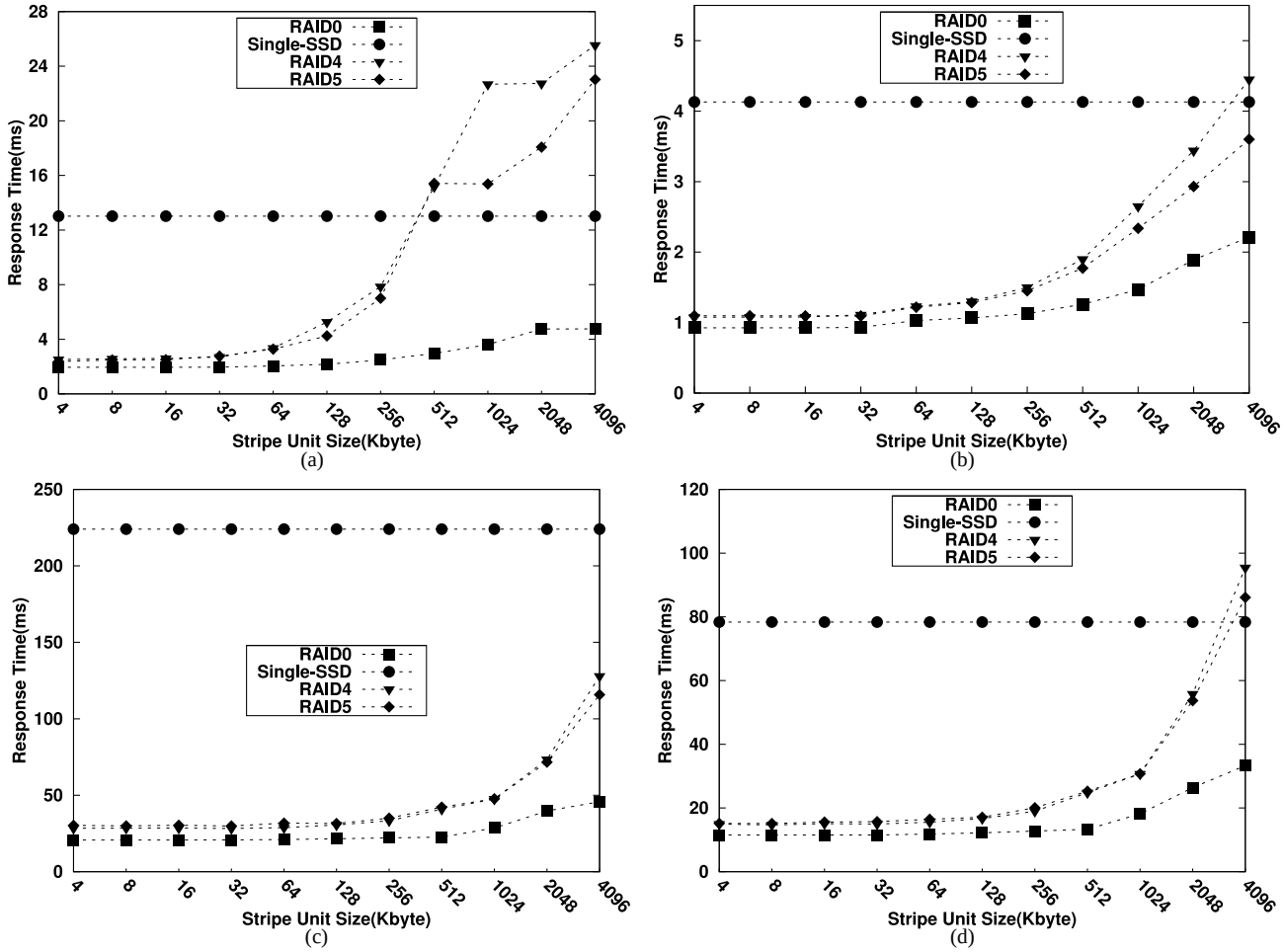

Fig. 9. Average response time by varying stripe unit size: (a) Iozone, (b) Postmark, (c) CAMRESIRA01-lvm1, (d) CAMWEBDEV-lvm2

Fig. 8 compares the number of page writes for data and parity disks in RAID4 and RAID5 arrays. In this figure, we report the number of page writes rather than the number of extra page writes. Similar to the discussion provided for Fig. 7, page writes in RAID5 is evenly distributed across all disks. Therefore, one should expect similar disk aging for all SSDs in a RAID5 array. From the results reported in Fig. 8, two observations can be concluded. First since the number of extra page writes is negligible in small stripe unit size (e.g., 4KB), we observe almost similar disk aging for both RAID4 and RAID5 configurations. Second, as the number of extra page writes becomes considerable in large stripe unit sizes (e.g., 1024KB), the disk aging in RAID5 array gets more pronounced than data disks in RAID4 array. As shown in Fig. 8, the number of page writes imposed to data disks in RAID4 is, on average, 20% less compared to RAID5 array in large stripe unit sizes.

Note that the number of writes in RAID4 and RAID5 arrays is almost similar. However, as pointed out in Sec. III, the main shortcoming of RAID5 array is that SSDs in a RAID5 arrays suffer from simultaneous disk failures as all disks get similar writes. That is all disks wear out approximately with the same rate. The results shown in Fig. 8 validate this observation.

B. Effect of stripe unit size on overall performance

As shown in the previous subsection, incrementing the stripe unit size leads to significant increase in the number of page reads and page writes from/to the parity disk and also increases the number of page reads from data disks. In this subsection, we investigate the effect of this observation on two major metrics of disk performance, namely, response time and throughput. The average response time of RAID4 and RAID5 arrays with different stripe unit sizes have been reported in Fig. 9 and it is compared with the performance of the following configurations: i) a RAID0 array including four SSDs and ii) a single SSD. The response time results have been reported for four I/O intensive benchmarks. As shown in Fig. 9, using RAID configuration with a small granularity of stripe unite size results in a significant improvement in the average response time compared to a single SSD. For instance, choosing a page-level stripe unit size in RAID0 will improve the response time up to six times compared to a single SSD. On the other hand, enlarging the stripe unit size to some extent in RAID4 and RAID5 can worsen the response time as opposed to a single SSD, as shown in Fig. 9. Additionally, it can be observed that RAID4 and RAID5 response times are very close to each other in small stripe unit sizes, however, RAID5 shows better response time due to distribution of parity

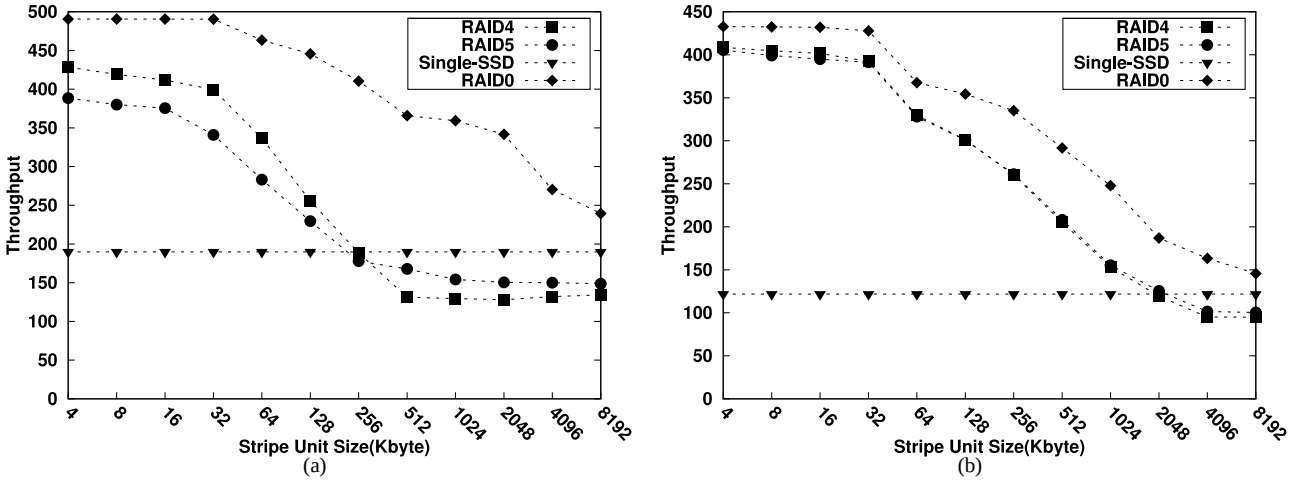

Fig. 10. Overall throughput by varying stripe unit size: (a) Iozone, (b) Postmark

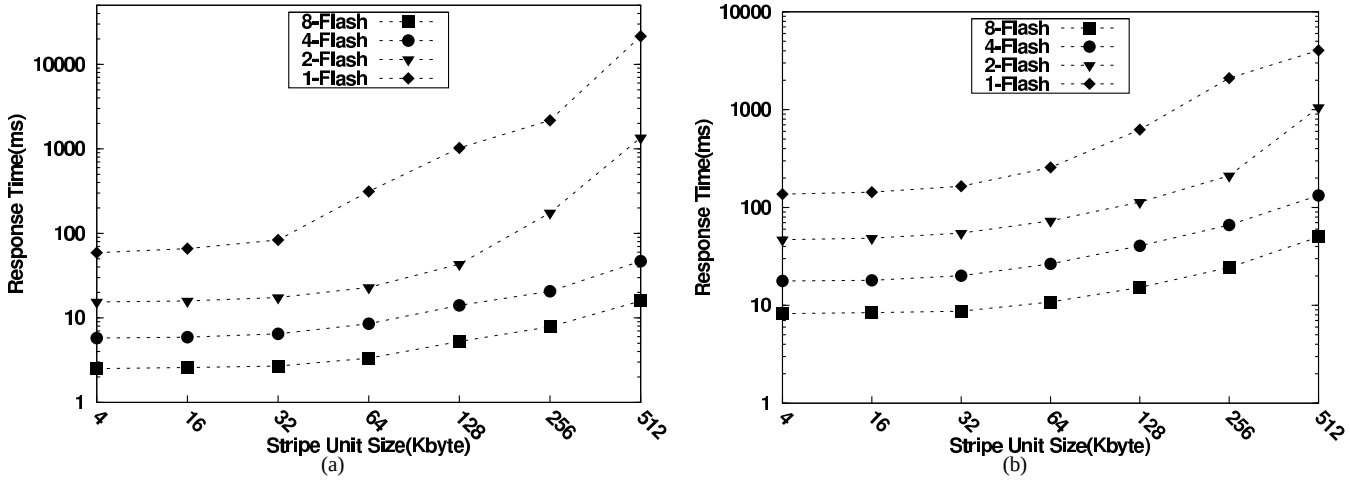

Fig. 11. Average response time by varying stripe unit size with different number of flash chips inside the SSD: (a) Iozone, (b) Vdbench

page reads and page writes. It should be noted that the larger stripe unit size decreases data distribution and consequently it worsens response time. However, as Fig. 9 depicts, more extra page reads and page writes has even more severe effect on performance degradation.

The overall throughput of RAID4, RAID5, and RAID0 arrays with different stripe unit sizes as well as a single SSD has been reported in Fig. 10. The overall throughput has been computed based on total time, idle time, number of requests, and average request size according to Equation 5. Such parameters are reported in DiskSim output simulation results.

$$Throughput = \frac{\sum Size(R_i)}{Total\ Simulation\ Time - Total\ Idle\ Time} \quad (5)$$

As it can be seen in Fig. 10, using 4KB stripe unit size in RAID4 and RAID5 improves throughput up to four times as compared to a single SSD. Fig. 10 also demonstrates that the throughput of RAID0, RAID4 and RAID5 arrays degrades as the stripe unit size enlarges. When stripe unit size exceeds 256KB and 2048KB, the throughput gets even worse than a single SSD in RAID4 and RAID5 configurations, for both Iozone and Postmark traces, respectively. The reason for this degradation is that when stripe unit size reaches the maximum request size in the trace, no data distribution takes place and extra page operations get saturated at its maximum number. The results show that both response time and throughput decline as the stripe unit size is increased. The experimental results reported in Fig. 9 and Fig. 10 are in agreement with the analysis provided in Section IV.

The way that the number of page reads and writes affects performance is also dependent on the number of parallel units used in each disk. For instance, when an SSD uses only one flash chip in serving a read or write request, it can only manage one request at a time. However, in an SSD with more number of flash chips, more requests can be simultaneously served due to parallelism between flash chips. We have evaluated the effect of number of flash chips on performance with different stripe unit sizes. Fig. 11 demonstrates that the less flash chips in an SSD, the more aggressive response time deterioration. Note the Y-axis in this figure is in logarithmic scale. The response time reported in this figure is comprised of service time and queue time. When more parallelism is established in an SSD, more extra reads and writes are served simultaneously.

TABLE VII
IMPACT OF STRIPE UNIT SIZE ON RESPONSE TIME IN RAID4 4+1 ARRAY USING SEAGATE-CHEETAH15K5 HDD

Stripe Unit Size (KB)	Response Time (ms)			Positioning Time (ms)			Transfer Time (ms)			Contribution of Positioning Time in Response Time (%)		
	Average Request Size (KB)											
	8	400	1600	8	400	1600	8	400	1600	8	400	1600
4	3.00	3.50	4.77	2.91	2.28	1.77	0.03	1.04	3.25	96.99	65.23	37.20
8	3.12	3.47	4.61	3.00	2.32	1.79	0.03	1.00	3.17	96.36	67.03	38.94
16	3.16	3.47	4.55	2.99	2.34	1.80	0.03	1.02	3.15	94.60	67.51	39.55
32	3.22	3.43	4.53	3.01	2.31	1.80	0.03	1.08	3.17	93.44	67.44	39.81
64	3.16	3.36	4.53	2.98	2.21	1.79	0.03	1.25	3.22	94.36	65.81	39.64
128	3.15	3.35	4.56	2.98	2.13	1.78	0.03	1.41	3.40	94.53	63.71	39.05

TABLE VIII
IMPACT OF STRIPE UNIT SIZE ON RESPONSE TIME IN RAID4 4+1 ARRAY USING SEAGATE-CHEETAH9LP HDD

Stripe Unit Size (KB)	Response Time (ms)			Positioning Time (ms)			Transfer Time (ms)			Contribution of Positioning Time in Response Time (%)		
	Average Request Size (KB)											
	8	400	1600	8	400	1600	8	400	1600	8	400	1600
4	5.07	18.21	59.71	4.16	4.01	4.10	0.22	13.63	55.82	82.05	22.04	6.87
8	5.13	17.60	56.50	4.06	3.89	3.99	0.27	12.96	52.57	79.20	22.11	7.06
16	5.20	17.94	55.58	4.02	3.76	3.84	0.37	13.09	51.48	77.31	20.98	6.91
32	5.13	19.40	56.27	4.03	3.87	3.91	0.28	13.76	51.43	78.61	19.92	6.95
64	5.10	22.75	58.45	4.02	3.68	3.67	0.28	15.72	52.50	78.72	16.18	6.28
128	5.13	28.90	64.97	4.02	3.84	4.00	0.29	19.08	55.86	78.44	13.28	6.15

TABLE IX
IMPACT OF STRIPE UNIT SIZE ON RESPONSE TIME IN RAID4 4+1 ARRAY USING QUANTUM-ATLASIII HDD

Stripe Unit Size (KB)	Response Time (ms)			Positioning Time (ms)			Transfer Time (ms)			Contribution of Positioning Time in Response Time (%)		
	Average Request Size (KB)											
	8	400	1600	8	400	1600	8	400	1600	8	400	1600
4	7.08	26.02	94.09	6.37	5.74	5.72	0.23	19.87	88.36	90.00	22.08	6.07
8	7.24	25.09	88.70	6.38	5.79	5.85	0.29	18.73	82.70	88.19	23.09	6.60
16	7.25	25.02	86.71	6.36	5.63	5.75	0.31	18.49	80.43	87.73	22.51	6.63
32	7.31	26.88	87.44	6.33	5.40	5.51	0.36	19.90	80.64	86.58	20.10	6.30
64	7.22	31.43	89.76	6.34	5.53	5.72	0.30	24.44	82.76	87.83	17.59	6.37
128	7.26	39.12	97.82	6.37	5.46	5.64	0.30	31.66	90.35	87.76	13.95	5.76

On the other hand, having less parallelism increases service time and queue time and it, in turn, harshly degrades the performance.

C. Extra Page Reads and Writes in HDD-Based RAID Arrays

Table VII, Table VIII, and Table IX provide three samples to elaborate the impact of stripe unit size on response time using different HDDs (Seagate-Cheetah15k5, Seagate-Cheetah9LP, and Quantum-AtlasIII [40]) whose characteristics are reported in Table X. In these sample experiments, we have used three synthetic traces with average request size equal to 8KB, 400KB, and 1600KB. The main observation from these three tables is that the response time does not exhibit significant variation with different stripe unit sizes. As an example, considering the synthetic trace with average request size equal to 8KB, the response time varies from 3.00ms to 3.15ms when increasing the stripe unit size from 4KB to 128KB in Table VII. As an another example, considering the synthetic trace with average request size equal to 1600KB, the response time varies from 59.71ms to 64.97ms when increasing the stripe unit size from 4KB to 128KB in Table VIII. In the results provided in these tables, positioning time contributes to more than 75% when the average request size is equal to 8KB. In this case, due to major contribution of positioning time in the overall response time, the extra number of page reads and writes does not lead to significant performance overhead in HDDs. This observation, however, is not valid in SSDs as it was demonstrated in the previous subsection. On the other hand, the results demonstrate that positioning time contributes to less than 10% in Table VIII and Table IX when the average request size is equal to 1600KB. In this case, since user data is distributed over several stripes, the number of extra read and write pages becomes negligible as compared to the total number of read and write pages.

D. Effect of stripe unit size on life time of SSD array

Here, we investigate the effect of different stripe unit sizes on the endurance of a RAID4 array. To do so, the number of erase operations performed on data and parity disks have been reported in Fig. 12 for few I/O benchmarks. It can be observed that although the stripe unit size increment does not have much effect on the endurance of data disks, it considerably affects the number of erases in the parity disk. For instance, when the stripe unit size reaches 128KB and 1024KB in Iozone and

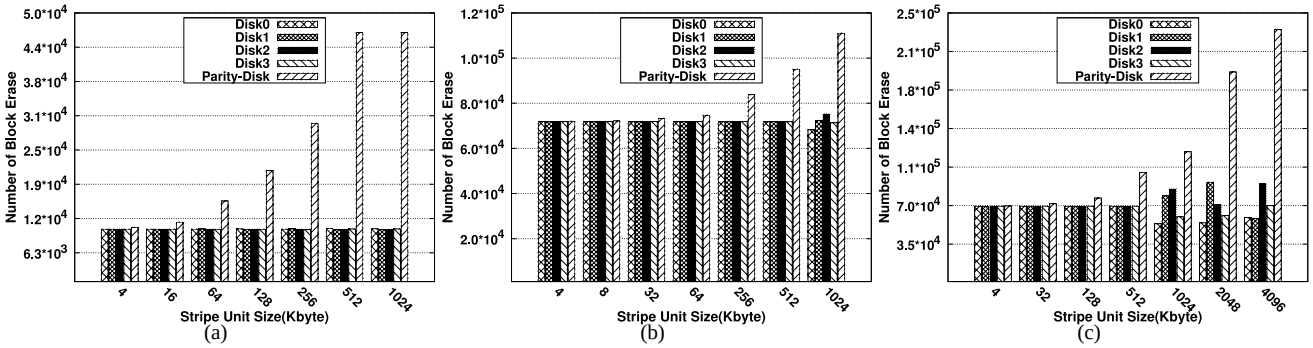

Fig. 12. Number of erases by varying stripe unit size : (a) Iozone, (b) CAMRESIRA01-lvm1, (c) CAMWEBDEV-lvm2

TABLE X
CHARACTERISTICS OF HDDS USED IN EXPERIMENTS

	Cheetah15k5	Cheetah9LP	Atlas *III*
Sector Transfer Time (ms)	0.0150	0.1050	0.1060
Max Seek Time (ms)	6.9139	10.627	15.3600
Min Seek Time (ms)	0.5525	0.8310	1.6630
Rotation Speed (rpm)	15020	10045	7200
Size (# blocks)	287M	18M	18M

CAMWEBDEV-lvm2 benchmarks, respectively, the number of erases on the parity disk is about twice greater than the number of erases in the data disks. This means that the parity disk ages about twice faster than data disks.

The issue of parity disk fast aging can be resolved by replacing the parity disk when it becomes too old. In advance replacement of parity disk, however, will impose cost in an inter-RAID4 array. In an intra-RAID4, it is not possible to replace only one chip within an SSD and the whole SSD should be discarded.

E. Performance experiments with a physical SSD-based RAID array

To further validate simulation results, we have conducted limited experiments on a physical system setup deploying RAID5 2+1 SSD array, using Linux software RAID. The evaluation is conducted on a hardware platform with an 1.8GHz AMD processor, employing an Nvidia ck804 SATA controller ©. The SSD module is the 40GB Intel SATA-*II* SSD (SSDSA2M040G2GC) ©. An 80GB HDD is used to house operating system (Ubuntu 12.04) and other applications. Fig. 13 demonstrates the results obtained by running Iozone trace on three SSDs configured as RAID5 2+1. Fig. 13.b shows that the number of extra page reads on different disks boosts up by stripe unit size enlargement. Fig. 13.c demonstrates slight increase in the number of page writes with increasing the stripe unit size. These two effects result in performance degradation as shown in Fig. 13.a, which admits our simulation results.

VI. RELATED WORK

Numerous studies have been performed on disk array storage systems characteristics and enhancement techniques. Among these works, studies which investigate striping methods and granularity effects on arrays performance and reliability, are more related to our study. In the following subsections, we review both HDD- and SSD-based RAID studies, which evaluate the effect of stripe unit size on performance, endurance, and reliability of RAID arrays. Since the scope of this work is on SSD-based RAID arrays, we elaborate such studies more in detail.

A. HDD-based RAID studies

There are a few studies which investigate the impact of stripe unit size on performance of HDD-based RAID arrays. In [48], Deng et. al. investigate the performance effect of stripe unit size on non-redundant HDD-based RAID integrated in *Network Attached System* (NAS). This study demonstrates that the stripe unit size has a negligible impact on performance of RAID0 arrays. This is explained by the fact that the sub-commands of different requests coming from file system are combined into one I/O command, which results in only one disk positioning in each drive. This effect decreases the overall positioning time and mitigates the impact of stripe unit size variation.

In [24], Chen et. al. study how data should be distributed on a redundant HDD-based RAID to optimize the disk array performance. This study investigates optimal stripe unit size for read and write intensive workloads in a RAID5 configuration. It is demonstrated that read requests in a RAID5 array behave similar to reads and writes in non-redundant disks when varying stripe unit size. However, write intensive workloads achieve optimal performance at smaller stripe unit size due to more full stripe writes with small granularity.

In [23], Jin et. al. introduce a performance model to analyze the effect of stripe unit size in a RAID array. This model can be applied to achieve optimal stripe unit size in a disk array. In [49], Hwang et. al. presented a distributed disk array architecture, named RAIDx, which is aimed at achieving higher I/O performance. By integrating orthogonal striping and mirroring architecture, it is shown that RAIDx outperforms RAID5 and RAID10 arrays. In this study, the effect of stripe unit size on aggregate bandwidth of RAIDx has been also investigated and it is compared with RAID5 and RAID10 arrays.

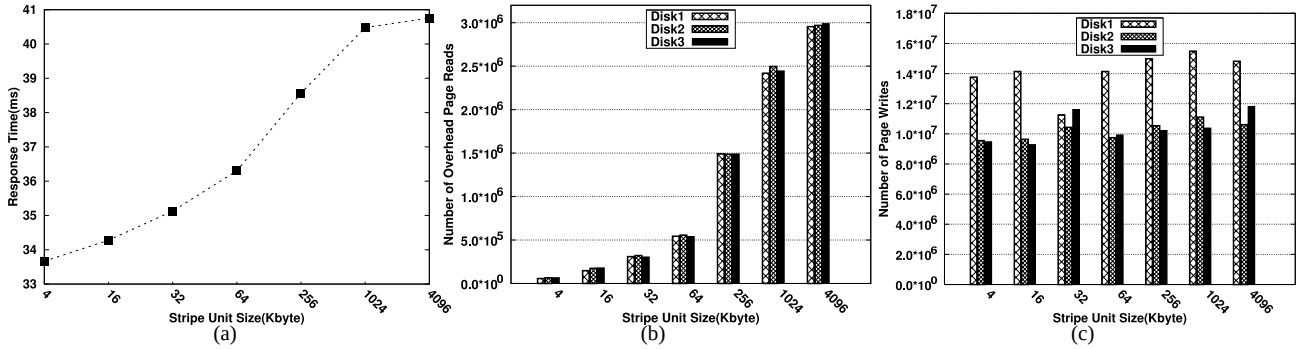

Fig. 13. Experiment results from executing Iozone trace on a physical SSD-based RAID array: (a) average response time, (b) number of extra page reads, (c) number of page writes

B. SSD-based RAID studies

Studies on SSD-based arrays have been performed considering different array architectures from simple level of RAIDs to more complicated array such as RAID6 utilizing erasure code techniques. A part of these studies investigate striping and granularity effects on different level of RAIDs or erasure coded arrays. Furthermore, there are several studies, which introduce techniques at the aim of performance and reliability enhancement. There are few methods, which investigate the impact of stripe unit size on performance of an SSD-based RAID array.

In [50], Petrov et. al. studied properties of an SSD-based RAID array and considered issues, which affect performance of such array. It is demonstrated that while a single SSD shows significant performance enhancement over HDD, an SSD-based RAID array suffers from performance degradation due to a) current non-SSD compatible RAID controllers, b) read/write asymmetry of SSDs, and c) fragmentation caused by erase operations. This study concludes that the performance degradation is mostly due to RAID controllers that are not designed for the characteristics of SSDs. The authors have also examined the effect of stripe unit size on the throughput of RAID0 or a non-parity-based array. Although the impact of stripe unit size on throughput was studied, the corresponding effect on other important metrics such as response time and endurance has not been analyzed. Additionally, parity-based RAID and the effect of parity computation on performance has not been studied in the proposed study.

In [51], Jeremic et. al. demonstrated the pitfall of SSD-based RAID arrays. They pointed out to the impact of stripe unit size in parity-based RAID arrays, however, no experimental results have been provided. The proposed study also investigates the impact of page-size alignment, synchronous aging, and workload history on performance of SSD-based RAID arrays.

In [52], Chang et. al. proposed an adaptive stripping method which enhances garbage collection process. They introduced dynamic stripping versus static stripping for choosing an element inside an SSD. In a static stripping scheme, a page write request is assigned to an element based on its logical block address and the number of elements reside in an SSD. It is explained that although the static scheme distributes data evenly, due to uneven access to data blocks, some elements might possess much more hot data blocks than others. Consequently, such elements may call garbage collection process more frequently, resulting in performance degradation. In order to mitigate this issue, dynamic stripping has been suggested, which distributes pages of write requests on idle banks that involve free pages. In this scheme, a hot data goes over a bank with the lowest number of erase count.

In [53], Chang et. al. presented a striping algorithm among banks in a flash memory. The proposed scheme uses load balancing and redundancy to improve both performance and reliability. In the proposed scheme, stripes of an intra-RAID are classified into data and code stripes. A code stripe maintains the *exclusive OR* of distributed information on the corresponding data stripe such that the exclusive OR of two banks in a data stripe will be stored on a different bank in a code stripe. Consequently, when a bank is busy, the corresponding data can be retrieved from other banks, which results in improved performance. In [54], Y. Deng et. al. examine flash-based storage system architectures and study optimization methods to improve SSDs performance and energy consumption. SSDs parallelism such as Intra-RAID and Inter-RAID and its different architectures are also investigated in this study.

Balakrishnan et. al. proposed a parity-based RAID, called differential RAID [10]. The proposed method is based on different aging rate of SSDs in an array. In this method, the reliability of SSD array is increased by decreasing the probability of data loss in the event of one disk failure. This method suggests an uneven parity distribution, which makes SSDs receive different percentage of parities and have different aging rate. For instance, when there are five disks in an array, the parity assignment such as (60, 10,10, 10 , 10) explains that the first disk stores 60 percent of parities while in the other disks, each holds 10 percent of parities. Therefore, the first disk ages faster than the other disks. The aging rate of disks also depends on write request distribution. Requests which impose full stripe writes, make disks age closely and decrease the reliability. In contrast, the reliability is the highest when all the requests are random writes. When the stripe unit size is small, many requests in a workload are distributed in a full stripe manner which increases the data loss probability. From previous subsections, we observed that smaller stripe unit size imposes less parity page writes. Hence, small stripe

unit size makes more full stripe writes and less parities which together make the reliability of the array to decline. When stripe unit size enlarges, the number of generated parity pages is augmented; however, the amount of full or partial stripe writes mostly depends on workload and request size.

There are a few studies that evaluate the effect of word size or the number of words in a stripe unit on the performance of disk subsystems configured as erasure coded arrays. A two dimensional stripe-based erasure code named GRID is proposed in [55]. This study analytically investigates the effect of stripe on storage efficiency in an array. The authors describe stripe size as multiplication of the number of stripes in a row stripe and the number of stripes in a column stripe. Although the proposed method is based on stripe-based erasure codes, the effect of stripe unit size, which is different from stripe size is not investigated.

Another study investigating word size has been presented in [56]. This study gives insights for erasure code deployment in cloud file systems. The impact of word size and the number of words on recovery time and degraded reads has been examined. It is demonstrated that larger word size results in larger recovery performance. It is also shown that lager word size depreciate the disk seek time overhead. Another study presented in [57] investigates the impact of data disks and word size on the proposed RAID6 scheme, called minimum RAID6 codes. The proposed experimental results show that larger word size would result in improved performance in the proposed RAID6 codes. This study has been proposed for HDD-based RAID arrays.

VII. Conclusion and Future work

In this paper, we explored the effect of stripe unit size on performance and endurance of SSD-based RAID arrays. We presented an analytical study to investigate the impact of stripe unit size on extra page reads and writes incurred by write requests. Although most previous studies have used larger stripe size (such as 64KB or 128KB) in SSD-based RAID arrays, our proposed analytical study revealed that a 4KB stripe unit size can significantly improve throughput, response time, and endurance in SSD-based RAID arrays as compared to large stripe unit sizes (e.g., 128KB). To validate the proposed analytical study, we used I/O intensive traces and evaluated the effect of stripe unit size using a disk subsystem simulator. The results obtained using experimental study using I/O intensive traces demonstrated that choosing a 4KB stripe unit size can improve throughput, response time, and endurance of SSD-based RAID arrays up to 67.6%, 52.2%, and 48.6% respectively, as compared to 128KB stripe unit size.

As a future work, one can extend this study to investigate the impact of stripe unit size on performance and endurance of complex erasure codes such as Reed-Solomon, Even-odd, and X-code. In particular, investigation of word size and the number of words within a stripe unit can be further examined in SSD-based RAID arrays employing complex erasure codes.

References

[1] Narayanan, D., Thereska, E., Donnelly, A., Elnikety, S., and Rowstron, A. "Migrating Server Storage to SSDs: Analysis of Tradeoffs". *4th ACM European Conference on Computer systems (EuroSys)*, pp. 145–158, (2009).

[2] Chen, F., Koufaty, D., and Zhang, X. "Understanding Intrinsic Characteristics and System Implications of Flash Memory Based Solid State Drives". *11th International Joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pp. 181–192. ACM, (2009).

[3] Kim, Y., Tauras, B., Gupta, A., and Urgaonkar, B. "FlashSim: A Simulator for NAND Flash-Based Solid-State Drives". *First International Conference on Advances in System Simulation*, pp. 125–131, (2009).

[4] Kim, Y., Oral, S., Shipman, G., Lee, J., Dillow, D., and Wang, F. "Harmonia: A Globally Coordinated Garbage Collector for Arrays of Solid-State Drives". *IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1–12, May (2011).

[5] Wong, G. "SSD Market Overview". *Inside Solid State Drives (SSDs)*, volume 37 of *Springer Series in Advanced Microelectronics*, pp. 1–17. Springer Netherlands, (2013).

[6] Agrawal, N., Prabhakaran, V., Wobber, T., Davis, J., Manasse, M., and Panigrahy, R. "Design Tradeoffs for SSD Performance". *USENIX Annual Technical Conference (ATC)*, pp. 57–70, (2008).

[7] Im, S. and Shin, D. "Flash-Aware RAID Techniques for Dependable and High-Performance Flash Memory SSD". *IEEE Transactions on Computers (TC)*, 60(1), pp. 80 –92, January (2011).

[8] Greenan, K., Long, D., Miller, E., Schwarz, T., and Wildani, A. "Building Flexible, Fault-Tolerant Flash-Based Storage Systems". *5th Workshop on Hot Topics in Dependability (HotDep)*, (, Lisbon, Portugal, 2009).

[9] Mao, B., Jiang, H., Feng, D., Wu, S., Chen, J., Zeng, L., and Tian, L. "HPDA: A Hybrid Parity-Based Disk Array for Enhanced Performance and Reliability". *IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, pp. 1–12, April (2010).

[10] Balakrishnan, M., Kadav, A., Prabhakaran, V., and Malkhi, D. "Differential RAID: Rethinking RAID for SSD Reliability". *ACM Transaction on Storage (TOS)*, 6, pp. 1–22, July (2010).

[11] Chang, Y. H., Hsieh, J. W., and Kuo, T. W. "Endurance Enhancement of Flash-Memory Storage Systems: an Efficient Static Wear Leveling Design". *44th Annual Design Automation Conference*, pp. 212–217, (2007).

[12] Mielke, N., Marquart, T., Wu, N., Kessenich, J., Belgal, H., Schares, E., Trivedi, F., Goodness, E., and Nevill, L. "Bit Error Rate in NAND Flash Memories". *IEEE International Symposium on Reliability Physics (IRPS)*, pp. 9–19, May (2008).

[13] Kang, Y. and Miller, E. "Adding Aggressive Error Correction to a High-Performance Compressing Flash File System". *7th ACM International Conference on Embedded Software (EMSOFT)*, pp. 305–314, (2009).

[14] Yaakobi, E., Ma, J., Grupp, L., Siegel, P., Swanson, S., and Wolf, J. K. "Error Characterization and Coding Schemes for Flash Memories". *IEEE Globecom Workshop on Application of Communication Theory to Emerging Memory Technologies (ACTEMT)*, pp. 1–5, December (2010).

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

[16] Blaum, M., Brady, J., Bruck, J., and Menon, J. "EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures". *IEEE Transaction on Computers*, 44(2), pp. 192–202, (1995).

[17] Chen, P., Lee, E., Gibson, G., Katz, R., and Patterson, D.

"RAID: High-Performance, Reliable Secondary Storage". *ACM Computing Surveys (CSUR)*, 26, pp. 145–185, June (1994).
[18] Patterson, D., Gibson, G., and Katz, R. "A Case for Redundant Arrays of Inexpensive Disks (RAID)". *ACM SIGMOD Record*, 17, pp. 109–116, June (1988).
[19] Chen, S. and Towsley, D. "A Performance Evaluation of RAID Architectures". *IEEE Transactions on Computers (TC)*, 45, pp. 1116–1130, October (1996).
[20] Lee, E. K. and Katz, R. H. "Performance Consequences of Parity Placement in Disk Arrays". *ACM SIGARCH Computer Architure News*, 19(2), pp. 190–199, (1991).
[21] He, X., Beedanagari, P., and Zhou, D. "Performance Evaluation of Distributed iSCSI RAID". *international workshop on Storage network architecture and parallel I/Os*, pp. 11–18, (2003).
[22] Lee, E. K. and Katz, R. H. "An Analytic Performance Model of Disk Arrays And Its Application". Technical report, (1991).
[23] Jin, H. and Hwang, K. "Optimal striping in RAID architecture". *Concurrency - Practice and Experience*, 12(10), pp. 909–916, (2000).
[24] Chen, P. M. and Lee, E. K. "Striping in a RAID Level 5 Disk Array". *SIGMETRICS - Performance Evaluation Review*, 23(1), pp. 136–145, May (1995).
[25] IBM Incorporation. "History of changes to software - ServeRAID". http://www-947.ibm.com/support/entry/portal/docdisplay?lndocid=MIGR-4QDMES.
[26] EMC Corporation. "EMC CLARiiON RAID 6 Technology". Technical report, July (2007).
[27] Hewlett-Packard Development Company. "HP Array Configuration Utility User Guide". Technical report, April (2011).
[28] Lee, S., Ha, K., Zhang, K., Kim, J., and Kim, J. "FlexFS: a Flexible Flash File System for MLC NAND Flash Memory". *USENIX Annual Technical Conference*, (2009).
[29] Chang, L. P. and Huang, L. C. "A Low-Cost Wear-Leveling Algorithm for Block-Mapping Solid-State Disks". *ACM Special Interest Group on Programming Languages Notices (SIGPLAN Not.)*, 46, pp. 31–40, May (2011).
[30] Chang, L. P. "On Efficient Wear Leveling for Large-Scale Flash-Memory Storage Systems". *ACM Symposium on Applied Computing (SAC)*, pp. 1126–1130, (2007).
[31] Kuratti, A. and Sanders, W. "Performance Analysis of the RAID 5 Disk Array". *International Computer Performance and Dependability Symposium (ICPDS)*, pp. 236–245, April (1995).
[32] Reed, I. and Solomon, G. "Polynomial Codes Over Certain Finite Fields". *Journal of the Society for Industrial and Applied Mathematics*, 8(2), pp. 300–304, (1959).
[33] Blmer, J., Kalfane, M., Karp, R., Karpinski, M., Luby, M., and Zuckerman, D. "An XOR-Based Erasure-Resilient Coding Scheme". Technical report, International Computer Science Institute, August (1995).
[34] Xu, L. and Bruck, J. "X-code: MDS Array Codes With Optimal Encoding". *IEEE Transaction on Information Theory*, 45(1), pp. 272–276, (2006).
[35] Tokyo-Toshiba Corporation. "Toshiba introduces high performance blade-type SSDs". http://www.toshiba.co.jp/about/press/2010_11/pr0801.htm, November (2010).
[36] Micron Technology Inc. "P400m Enterprise SATA SSD". http://www.micron.com, January (2013).
[37] Kaneko, H., Matsuzaka, T., and Fujiwara, E. "Three-Level Error Control Coding for Dependable Solid-State Drives". *14th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 281–288, (2008).
[38] Mir, I. and McEwan, A. "A Fast Age Distribution Convergence Mechanism in an SSD Array for Highly Reliable Flash-Based Storage Systems". *3rd IEEE International Conference on Communication Software and Networks (ICCSN)*, pp. 521–525, May (2011).
[39] Mir, I. and McEwan, A. "A Reliability Enhancement Mechanism for High-Assurance MLC Flash-Based Storage Systems". *17th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, volume 1, pp. 190–194, August (2011).
[40] Bucy, J., Schindler, J., Schlosser, S., and Ganger, G. "The DiskSim Simulation Environment Version 4.0 Reference Manual". Technical report, CMU-PDL-08-101, Carnegie Mellon University, May (2008).
[41] Lee, J., Kim, Y., Shipman, G., Oral, S., Wang, F., and Kim, J. "A semi-preemptive garbage collector for solid state drives". *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 12–21, April (2011).
[42] Lee, Y., Barolli, L., and Lim, S. "Mapping granularity and performance tradeoffs for solid state drive". *The Journal of Supercomputing*, pp. 1–17, (2012).
[43] Kim, J., Lee, J., Choi, J., Lee, D., and Noh, S. H. "Enhancing SSD reliability through efficient RAID support". *Proceedings of the Asia-Pacific Workshop on Systems*, pp. 4:1–4:6, (2012).
[44] Narayanan, D., Donnelly, A., and Rowstron, A. "Write Off-Loading: Practical Power Management for Enterprise Storage". *ACM Transaction on Storage*, 4(3), pp. 1–23, (2008).
[45] Katcher, J. "PostMark: a New File System Benchmark", October (1997).
[46] Norcott, W. D. "IOzone". http://www.iozone.org.
[47] Berryman, A., Calyam, P., Honigford, M., and Lai, A. M. "VDBench: A Benchmarking Toolkit for Thin-Client Based Virtual Desktop Environments". *IEEE Second International Conference on Cloud Computing Technology and Science*, pp. 480–487, (2010).
[48] Deng, Y. and Wang, F. "Exploring the Performance Impact of Stripe Size on Network Attached Storage Systems". *Journal of Systems Architecture*, 54(8), pp. 787–796, (2008).
[49] Hwang, K., Jin, H., and Ho, R. "Orthogonal Striping and Mirroring in Distributed RAID for I/O-Centric Cluster Computing". *IEEE Transaction on Parallel and Distribution Systems*, 13(1), pp. 26–44, (2002).
[50] Petrov, I., Almeida, G., Buchmann, A., and Graf, U. "Building Large Storage Based on Flash Disks". *ADMS*, September (2010).
[51] Jeremic, N., Mühl, G., Busse, A., and Richling, J. "The Pitfalls of Deploying Solid-State Drive RAIDs". *4th Annual International Conference on Systems and Storage*, pp. 14:1–14:13, (2011).
[52] Chang, L. P. and Kuo, T. W. "An Adaptive Striping Architecture for Flash Memory Storage Systems of Embedded Systems". *IEEE Real-Time and Embedded Technology and Applications Symposium (RTETS)*, pp. 187196, (2002).
[53] Chang, Y. B. and Chang, L. P. "A Self-Balancing Striping Scheme for NAND-Flash Storage Systems". *ACM Symposium on Applied Computing (SAC)*, pp. 1715–1719, (2008).
[54] Deng, Y. and Zhou, J. "Architectures and Optimization Methods of Flash Memory Based Storage Systems". *Journal of Systems Architecture*, 57(2), pp. 214–227, (2011).
[55] Li, M., Shu, J., and Zheng, W. "GRID Codes: Strip-Based Erasure Codes With High Fault Tolerance for Storage Systems". *ACM Transaction on Storage*, 4(4), pp. 15:1–15:22, (2009).
[56] Khan, O., Burns, R., Plank, J., Pierce, W., and Huang, C. "Rethinking Erasure Codes for Cloud File Systems: Minimizing I/O for Recovery and Degraded Reads". *10th USENIX Conference on File and Storage Technologies*, (2012).
[57] Plank, J., Buchsbaum, A., and Z. Vander, T. B. "Minimum Density RAID-6 Codes". *ACM Transaction on Storage*, 6(4), pp. 16:1–16:22, (2011).

Farzaneh Rajaei Salmasi received the BSc and MSc degrees in computer engineering from the Amirkabir University of Technology (Tehran Polytechnics) and the Sharif University of Technology (SUT) in 1999 and 2011, respectively. She has been working as a research assistant in the Data Storage Systems and Networks (DSN) laboratory, SUT, since 2010 till present. Her research interests include SSD reliability and security.

Hossein Asadi received the B.Sc. and M.Sc. degrees in computer engineering from the Sharif University of Technology (SUT), Tehran, Iran, in 2000 and 2002, respectively, and the Ph.D. degree in electrical and computer engineering from Northeastern University, Boston, MA, in 2007. He has been with the Department of Computer Engineering, SUT, since 2009, where he is currently an Assistant Professor. He has established and has been the director of the Data Storage Systems and Networks (DSN) laboratory, SUT, since 2009.

He was with EMC Corporation, Hopkinton, MA, as a Research Scientist and Senior Hardware Engineer, from 2006 to 2009. From 2002 to 2003, he was a member of the Dependable Systems Laboratory, SUT, where he researched hardware verification techniques. From 2001 to 2002, he was a member of the Sharif Rescue Robots Group. He has authored or co-authored more than 50 technical papers in reputed journals and conference proceedings. His current research interests include data storage systems and networks, solid-state drives, and reconfigurable and dependable computing. Dr. Asadi was a recipient of the Technical Award for the Best Robot Design from the International RoboCup Rescue Competition, organized by AAAI and RoboCup, and the Distinguished Lecturer Award from the SUT in 2010, one of the most prestigious awards in the university.

Majid GhasemiGol is currently a senior B.Sc. student in the Department of Computer Engineering at Sharif University of Technology (SUT). He has been working as a research assistant in the Data Storage Systems and Networks (DSN) laboratory, SUT, since 2011 till present. His research interests include SSD reliability and security.