Microelectronics Reliability 55 (2015) 2453–2467

Contents lists available at ScienceDirect

Microelectronics Reliability

journal homepage: www.elsevier.com/locate/mr

On endurance and performance of erasure codes in SSD-based storage systems

Saeideh Alinezhad Chamazcoti, Ziba Delavari, Seyed Ghassem Miremadi *, Hossein Asadi

Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 29 January 2015
Received in revised form 13 June 2015
Accepted 24 July 2015
Available online 11 August 2015

Keywords:
Erasure codes
Endurance
Performance
Solid-State Drive
Storage system

ABSTRACT

Erasure codes are widely used in data storage systems to protect against disk failures. Employing erasure codes in an array of Solid-State Drives (SSDs) in storage systems necessitates designers to revisit different characteristics in comparison to Hard Disk Drives (HDDs), due to non-mechanical property of SSDs. One of the most important characteristics of SSDs is their limitation on the number of Program/Erase (P/E) cycles. By taking into account the characteristics of SSDs, this paper presents a comprehensive analysis to investigate the effects of three well-known erasure codes on the endurance and performance of SSD-based disk subsystems. The three erasure codes, i.e., Reed–Solomon, EVENODD, and RDP are implemented on the SSD-extension of DiskSim simulator. The results show that the endurance and performance of Reed–Solomon are on average 90% and 60% higher than other erasure codes, respectively. Additionally, the three erasure codes are compared in terms of different stripe unit sizes, number of disks, and various request sizes. The results show that configuring a disk array with a 4 KB stripe unit size will improve the endurance and performance of EVENODD by 1.8× and 2.9×, respectively, as compared to 128 KB stripe unit size.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

Over the past decade, *Hard Disk Drives* (HDDs) have been gradually replaced by flash-based *Solid-State Drives* (SSDs) due to their higher performance, lower power consumption, and higher shock resistance as compared to HDDs. SSDs, however, suffer from a limited number of *Program/Erase* (P/E) cycles that can be applied to a block of flash memory. The number of P/E cycles in a block of flash memory is known as the write endurance of that block [1]. It has been reported that the endurance of a block in flash memories used in mainstream applications is lower than 10^4 cycles [2]. This means that the number of P/E cycles in write-intensive applications significantly affects the reliability of SSDs [2].

A commonly used scheme to enhance the reliability of HDD- and SSD-based storage systems is employing erasure codes. These codes are typically implemented in *Redundant Array of Independent Disks* (RAID) [3]. Erasure codes are stored in disks, called parity disks, to recover from failures occurring in data disks. Several erasure codes have been presented in the literature to improve both reliability [4–10] and performance of disk subsystems [11–13].

Due to non-mechanical characteristics of SSDs, the RAID arrays employing SSDs exhibit different reliability and performance levels as compared to HDD-based counterparts as follows: 1) The write endurance of flash memories affects the aging of SSDs while HDDs do not suffer from this limitation [1]. 2) SSD-based RAID arrays can provide improved performance as compared to HDD-based RAID arrays due to not employing mechanical components in the SSD structure. Applying erasure codes in SSDs and HDDs may change the figures of the write endurance and performance. There has been no effort in the literature to investigate the performance and endurance of different erasure codes on SSD-based RAID arrays.

This paper presents a comprehensive analysis of comparing the performance and endurance of different erasure codes applied in SSD-based storage systems. The comparison is done with respect to some characteristics of storage systems, such as the *Stripe Unit Size* (SUS) and the *Number of Disks* (NoD). Various real traces are run in the SSD extension of DiskSim simulator. The analysis in this study helps designers to effectively configure storage systems to adapt an appropriate erasure code for different applications. In particular, the contributions of this work are two-fold:

First, the write endurance of SSDs as a main concern in SSD-based storage systems is evaluated for various erasure codes. In the evaluation, different erasure codes are compared in terms of *Number of Cleans* (NoC) and *Number of Writes* (NoW). In the comparison, the impact of different erasure codes on the endurance of system is studied. Second, the performance of different erasure codes is compared in terms of the number of I/O operations per second. To this end, the effect of different erasure codes on the number of I/O operations is evaluated by reporting the *Average Response Time* (ART) of the

* Corresponding author.
E-mail addresses: alinezhad@ce.sharif.edu (S. Alinezhad Chamazcoti), delavari@ce.sharif.edu (Z. Delavari), miremadi@sharif.edu (S.G. Miremadi), asadi@sharif.edu (H. Asadi).

http://dx.doi.org/10.1016/j.microrel.2015.07.046
0026-2714/© 2015 Elsevier Ltd. All rights reserved.

requests. It is notable to mention that this paper presents a first study on the performance and endurance of different erasure codes applied in SSD-based storage systems. Although the use of erasure codes in storage systems is so common to provide reliability, applying these codes in SSD-based storage systems brings some reliability concerns such as endurance limit. The effect of erasure codes on the endurance of SSD-based storage systems is investigated in this paper, and also the configurations of disk subsystems such as the number of disks in a RAID array and the size of stripe unit are evaluated for the target erasure codes. The results of our evaluation will help storage vendors to choose the best erasure code with optimal configuration in storage systems.

Several observations are concluded from the proposed analysis as follows:

1) Unlike using large size of stripe unit in storage venders such as IBM and Hitachi (i.e., 256 KB in IBM [15] and 512 KB in Hitachi [16]), the best performance for erasure codes in SSD-based system is achieved by configuring the array to employ small SUS.
2) Considering the structure of erasure codes, Reed–Solomon and EVENODD provide the best and the worst performance among the under study erasure codes, respectively.
3) The impact of modifying SUS on the performance of different erasure codes is more significant than modifying NoD.
4) The variation of performance for different erasure codes in various system configurations has similar trend to the variation of endurance.

This paper extends our previous work presented in [14] in two-folds: a) as opposed to our previous work, where only the number of write cycles is explored, this work investigates the endurance of RAID arrays in the presence of erasure codes in terms of both the number of write cycles and the number of block erases. To accurately model the number of block erases, we have implemented the target erasure codes in DiskSim. b) In our previous work, only endurance of erasure codes has been studied while this paper investigates both endurance and performance of the target erasure codes in SSD-based storage systems.

The remainder of this paper is organized as follows: In Section 2, a background of SSD-based storage systems and a brief overview of erasure codes are presented. Section 3 reports an analytical comparison of different erasure codes applying in SSD-based storage systems. The experimental results of comparing the endurance and performance of target erasure codes are presented in Section 4. Next, Section 5 reviews the previous studies on endurance and performance of erasure codes. Finally, the paper is concluded in Section 6.

2. Background

2.1. SSD-based storage systems

The use of SSD technology is currently more attractive in storage systems due to its higher performance and lower power consumption, as compared to the HDD technology [1]. Besides the benefit of SSDs, they suffer from inherent characteristics such as erase-before-write problem and inherent limitation on endurance. SSDs endure a limited number of P/E cycles and after that the stored data is not reliable anymore.

Storage systems based on either SSD or HDD technologies, are protected against faults by two common approaches called complete replication [17,18] and encoding method [4–10]. The complete replication causes extremely high storage overhead, while the encoding methods provide redundancy with lower storage overhead. Erasure codes are a kind of encoding method implemented in RAID [3]; these codes are stored in the parity disks to recover possible failures in data disks. Although the first aim of designing these codes was enhancing reliability in storage systems, further research on design and implementation of these codes was with the aims such as improved performance or faster recovery.

2.2. Overview of erasure codes

Erasure codes are kind of *Forward Error Correction* (FEC) which protect data in storage systems against disk failures. In these codes, n blocks of data are encoded into $m + n$ blocks of data and parity (m parity blocks and n data blocks), tolerating up to m failed blocks. These codes are typically applied in RAID6 $n + 2$, which can tolerate concurrent failures of any two data or parity blocks. Several erasure codes have been proposed in literatures based on RAID6, including Reed–Solomon code [4], Cauchy Reed–Solomon code [11], EVENODD code [5], RDP code [12], Blaum–Roth code [6], Liberation code [7], Cyclic code [8], HDP-Code [13], X-Code [9], and P-Code [10]. The erasure codes can be classified into two main classes: 1) non-XOR based and 2) XOR based. Among the mentioned erasure codes, in this paper, we choose two sample codes from each class, namely "*Reed–Solomon*" and "*EVENODD*". These erasure codes are the most popular and applicable codes in its class. We also choose RDP as an improved version of EVENODD in terms of XOR complexity, as the last target erasure code. RDP code is chosen to investigate whether the improvement of the code in terms of computation complexity will improve the endurance or not. These codes are discussed briefly in the following subsections. The analysis provided in this paper can be further applied to other erasure codes as well.

2.2.1. Reed–Solomon

Reed–Solomon code [11] is the most popular and applicable non-XOR based erasure code, which has widely been used in communications and storage systems. The main advantage of this code is its scalability to recover up to m failed blocks including data and parity blocks (m greater or equal to 2). This code imposes complex computation in both encoding and decoding operations due to usage of Galois Field arithmetic during coding operation. Due to complicated operations used in Galois Field arithmetic, table-lookup is used for required operations to decrease computation intensity. Complex computation is the major drawback of Reed–Solomon, which prevents it from widely being used in enterprise applications. The layout of Reed–Solomon encoding is illustrated in Fig. 1.

2.2.2. EVENODD

EVENODD code [5] is an XOR-based array code which is employed in storage systems, reliable communication, and dynamic load balancing applications. The computational complexity of EVENODD is much lower than the Reed–Solomon code due to the use of XOR operations in its computation.

This code is defined as a $(p-1)\times(p+2)$ matrix, where p is a prime number. Data and parities are stored in the p first blocks (columns), and the last two blocks, respectively. This code uses two parity disks, and can tolerate up to two disk failures. The row and diagonal parity disks are constructed by applying XOR operation, over data in a row and diagonal, respectively. In this code, an adjusting factor (S) is computed by XORing data blocks in the main diagonal. This factor is used for computing diagonal parities. The construction of horizontal and diagonal encoding of this code is illustrated in Fig. 2.a and b, respectively. The cells with the same shape share the same parity. Data cells are placed horizontally in the same row in the horizontal layout, and the corresponding parity is placed in the same row. For example, in Fig. 2.a, cells with the "circle" shape located in (4,0), (4,1), (4,2), (4,3), (4,4), and (4,5) entries share the row parity unit in (4,6). In the diagonal layout, data cells which are placed in the same diagonal share the common diagonal parity. For example, the circle shape placed in (0,4), (1,3), (2,2), (3,2), and (4,0) entries share the shape placed in (4,7).

2.2.3. RDP

In *Row-Diagonal Parity* (RDP) code [12], a $(p-1)\times(p+1)$ matrix is defined where p is a prime number. Data is stored in the $p-1$ first blocks, while parities are stored in the two last blocks. RDP is the

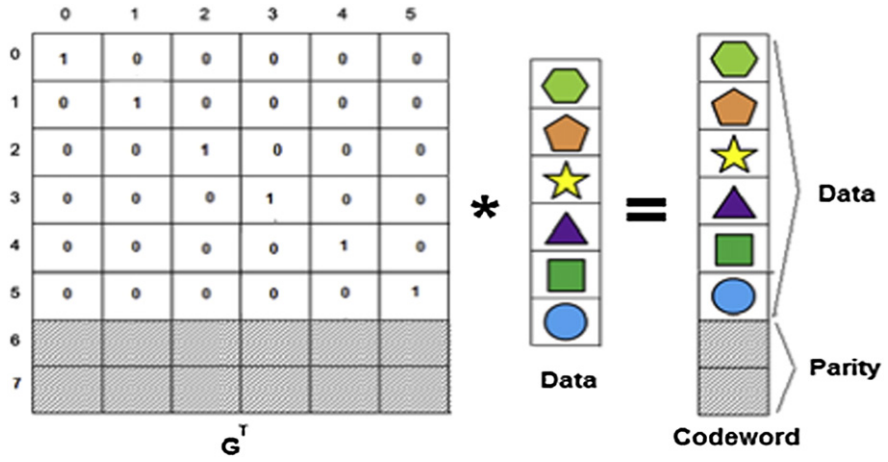

Fig. 1. Reed–Solomon encoding layout [14].

improved version of EVENODD in terms of computational complexity as it removes the calculation of the adjusting factor *S*, which is used in constructing diagonal parity. The RDP layout is illustrated in Fig. 3.

Among several proposed erasure codes, we just analyze and compare the abovementioned codes (i.e., Reed–Solomon, EVENODD, and RDP) with respect to the configuration of storage system and workload characteristics such as the number of disks, the size of stripe units, and the size of updated requests. The analysis is performed by using trace-driven simulation in the following sections.

2.3. Terminology

A brief description about the conceptual terms and abbreviation of this study is given in Fig. 4 and Table 1, respectively. As shown in Fig. 4, stripe unit size is the granularity of data distribution over disks. Each data unit in this figure is labeled by a 2-bit digit, which indicates the corresponding row and column (e.g., unit 03 belongs to row #0 and column #3).

Stripe Unit Size (SUS): It is the smallest access unit of disk in a RAID array by which any access to the stored data such as updating, decoding, and encoding procedures is scaled. For simplicity, the term 'unit' is used as an alternative to stripe unit in this paper.

Size of Request (SoR): This term indicates the length of request, which is specified in the trace file. This size determines the number of stripe units in the array that should be updated.

Number of Disks (NoD): This term refers to the number of data disks used in a RAID array, excluding the parity disks.

Code Pattern: This term refers to the structure of erasure codes in which the placement of data and parities in the array of disks are specified. The encoding of erasure codes is a key factor in design of code pattern. The code pattern of erasure codes under study are depicted in Figs. 1, 2, and 3.

3. Analytical comparison of erasure codes

In this section, we first explain our definition of performance and endurance of erasure codes in SSD-based RAID array. Then, we compare the erasure codes employed in SSD-based RAID arrays in terms of performance and endurance. Using SSDs in the configuration of RAID array brings new concern in evaluating erasure codes. The specifications of SSD and its limitations in the number of P/E cycles are the main distinction between our study and other studies which have examined erasure codes. The number of P/E cycles is the key factor in evaluating the performance and endurance of different erasure codes. The code pattern of erasure codes, i.e., the placement of data and parity in the layout of code array, affects the number of P/E cycles committed to the system by running applications. Additionally, the configuration of the RAID array such as stripe unit size and the number of disks impose different number of P/E cycles to the system. In this section, an analytical comparison on the performance and endurance of erasure codes is conducted by considering the above-mentioned issues, i.e., the code pattern of erasure codes and the configuration of RAID arrays.

3.1. Metrics of comparison

3.1.1. Performance

The performance metric used in the proposed analysis is defined as an average response time to perform the incoming requests. The computation complexity of erasure codes is, however, excluded in this definition. The novelty of our work is to perform comparison of erasure codes in SSD-based RAID arrays; thus we just consider those parameters that may be affected by the type of disk technology either SSDs or HDDs, in our evaluation. It is evident that the computation complexity of erasure codes is independent of the underlying disk technology, either SSDs or HDDs.

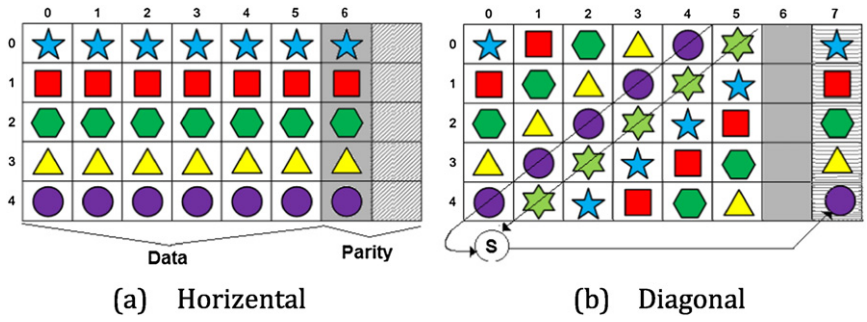

Fig. 2. Horizontal and diagonal layout of EVENODD code ($p = 6^1$).
P should be a prime number, but for the sake of fair comparison across different erasure codes, here we set p equal to 6.

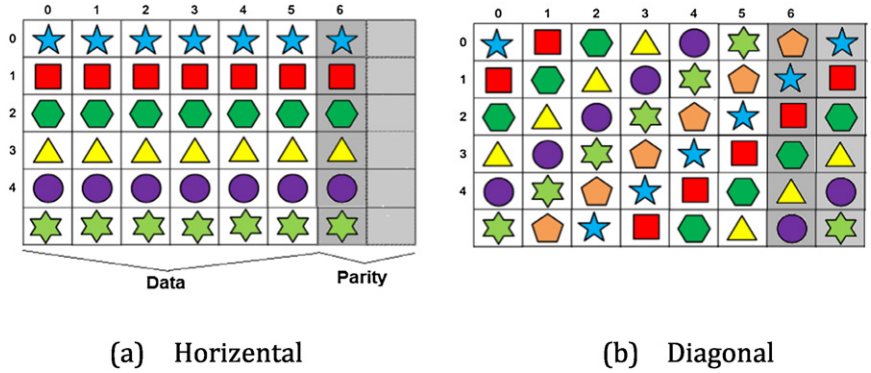

Fig. 3. Horizontal and diagonal layout of RDP code ($p = 7$) [13].

The average response time of different erasure codes is affected by IO operations, i.e., read, write, and erase operations. Due to the characteristics of SSDs, the erase time is almost one order of magnitude higher than read and write operations. Thus, the effect of NoC on performance is much higher than the other IO operations. Because of different code patterns of erasure codes in encoding and decoding, different numbers of I/O operations (i.e., read, write, and erase) would result in different performance of erasure code. The performance comparison in this paper is based on the I/O access performance, and the computation complexity has not been included in the evaluation. The reason is that the I/O operations are typically much more time-consuming than CPU operations (i.e., encoding and decoding time). While a single encoding or decoding can be accomplished in a CPU running at 2 GHz in order of hundreds of nanoseconds (or less than one microsecond), a single I/O operation typically takes from 100 microseconds to milliseconds. To this end, the computation time can be neglected in comparison to the I/O time.

A comparison on the computation complexity of target erasure codes is shown in Table 2. Reed–Solomon requires the minimum XOR operations for encoding; however, its decoding is the worst as compared to the other erasure codes. Reed–Solomon also offers the least complexity in updating while the other codes (EVENODD and RDP) impose variant complexity depending on the starting address of updating data blocks.

3.1.2. Endurance

In the proposed analysis, the *Number of Cleans* (NoCs) is used as a metric for comparing the endurance efficiency of erasure codes. In this analysis, we report the number of writes (NoWs) for each erasure code providing a comparison between NoCs and NoWs.

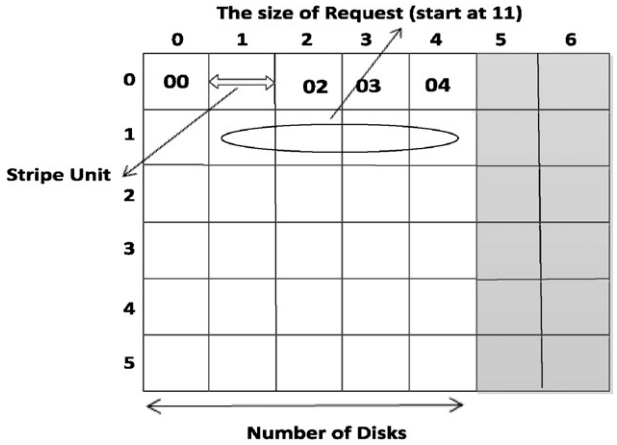

Fig. 4. Definition of conceptual terms used in this study.

Due to the characteristics of SSDs, the write and erase operations are performed on the granularity of page and block, respectively. Thus, by writing several pages of a block (in sequential write), only one erase operation should be performed. As a result, the number of writes is greater than the number of cleans. On the other, when a few number of pages of a block is written (trace with random size of request), the number of cleans is almost equal to the number of writes committed to the disks.

3.2. Analytical comparison

In this section, we analytically compare different erasure codes in terms of NoW. Modifying data units would require updating the corresponding parities (either row or diagonal parities). As such, parities should be recomputed, and rewritten to the parity disks. Since updating parities requires performing XOR calculation and I/O operations, this increases the number of P/E cycles admitted to system, affecting both system performance and endurance. The effect of code pattern, request size, and stripe unit size on the number of writes is evaluated as follows:

3.2.1. Impact of code pattern on NoW

The code pattern of erasure codes has the main impact on the endurance of different erasure codes. The number of parities for each data is determined in the code pattern. As shown in Fig. 1 through Fig. 3, Reed–Solomon, EVENODD, and RDP have different code patterns due to various encoding. The minimum dependency between data and parities results in smaller number of updated parities for each updated data unit. A comparison among different erasure codes for updating one data unit and their corresponding updated parity units is shown in Fig. 5. As can be concluded from this figure, Reed–Solomon imposes the smaller number of updated parities for each data update. As a result, one should expect higher endurance for Reed–Solomon in comparison to EVENODD and RDP.

A code with higher dependency between data and parities requires more XOR operations and more I/O operations to update the additional

Table 1
Abbreviation and its meaning.

Abbreviation	Meaning
SoR	Size of Request
NoD	Number of Disks
SUS	Stripe Unit Size
NoW	Number of Writes
NoR	Number of Reads
NoC	Number of Cleans
ART	Average Response Time
UD	Updated Data in terms of stipe unit
UP	Updated Parities (UP) = Updated Row Parity (URP) + Updated Diagonal Parity (UDP).

Table 2
A comparison on the computation complexity of erasure codes (n + 2 disks, i.e., n data + 2 parity disks).

Code	Computation (n, m = 2)		
	Encode	Decode	Update
Reed–Solomon code	O(mn)	$O(n^3)$	O(m)
EVENODD code	$O(n^2)$	$O(n^2)$	O(1)–O(n)
RDP code	$O(n^2)$	$O(n^2)$	O(1)–O(n)

parity units. The location of updated data units in the code pattern specifies which parity units should be updated. In Reed–Solomon, updating one data unit would update just two parity units irrespective to the place of the updated data. On the other hand, in EVENODD, the place of updated data does not affect the number of updated parity units. If the updated data is placed on the main diagonal, all the units of the diagonal parity disk should be updated. As an example, for a 4 KB write request, only one data unit should be updated. Depending on the place of the updated data unit within the array and the type of erasure codes, the number of updated parity units varies from one unit to five units.

In Fig. 5, we have illustrated cases where the minimum and the maximum number of parity updates can take place for the erasure codes under study. In erasure codes, depending on the place of updated data, the number of updated parity units is different. As a result, there are the minimum and maximum numbers of updated parity units for each erasure code. As shown in Fig. 5, Reed–Solomon only updates two parity units for any arbitrary updated data unit in the minimum and the maximum case. RDP in the minimum case updates just one parity unit, but in the maximum case, three parity units should be updated. EVENODD also updates two parity units in the minimum case but in the maximum case, when data units are placed in the main diagonal, five parity units should be updated. Consequently, as shown in Table 3, when SoR is equal to 4 KB, for updating one data unit, the average number of updated parity units in Reed–Solomon is lower than two other erasure codes.

Table 3
Effect of SoR on NoW for P1 and P2.

Code	SoR	SRI	SDI	NoD	SUS	NoW		
						P1	P2	P
Reed–Solomon	4 KB	X	D0	4 + 2	4 KB	1	1	2
		X	D3			1	1	2
	16 KB	X	D0			1	1	2
		X	D3			2	2	4
	32 KB	X	D0			2	2	4
		X	D3			3	3	6
	64 KB	X	X			4	4	8
EVENODD	4 KB	R1	D0	5 + 2	4 KB	1	1	2
		R1	D3			1	4	5
	16 KB	R1	D0			1	4	5
		R1	D3			2	4	6
	32 KB	R1	D0			2	4	6
		R1	D3			3	4	7
	64 KB	X	X			4	4	8
RDP	4 KB	R1	D0,D3	4 + 2	4 KB	1	1	2
		R2	D0			1	2	3
	16 KB	R1	D0			1	4	5
		R1	D3			2	4	6
	32 KB	R1	D0			2	4	6
		R1	D3			3	4	7
	64 KB	X	X			4	4	8

In Reed–Solomon, all data units in one row share the same parity units. This means that by increasing the number of updated data units in a row (i.e., sequential update), the number of updated parity units remain constant. In RDP and EVENODD, however, the corresponding diagonal parity units need to be updated when any arbitrary data unit is modified. For example, if a row is updated in an array of four disks (as shown in Table 3 when SUS is equal to 16 KB), only two parity units would be updated in Reed–Solomon. However, in the similar configuration using RDP and EVENODD codes, six and five parities would be

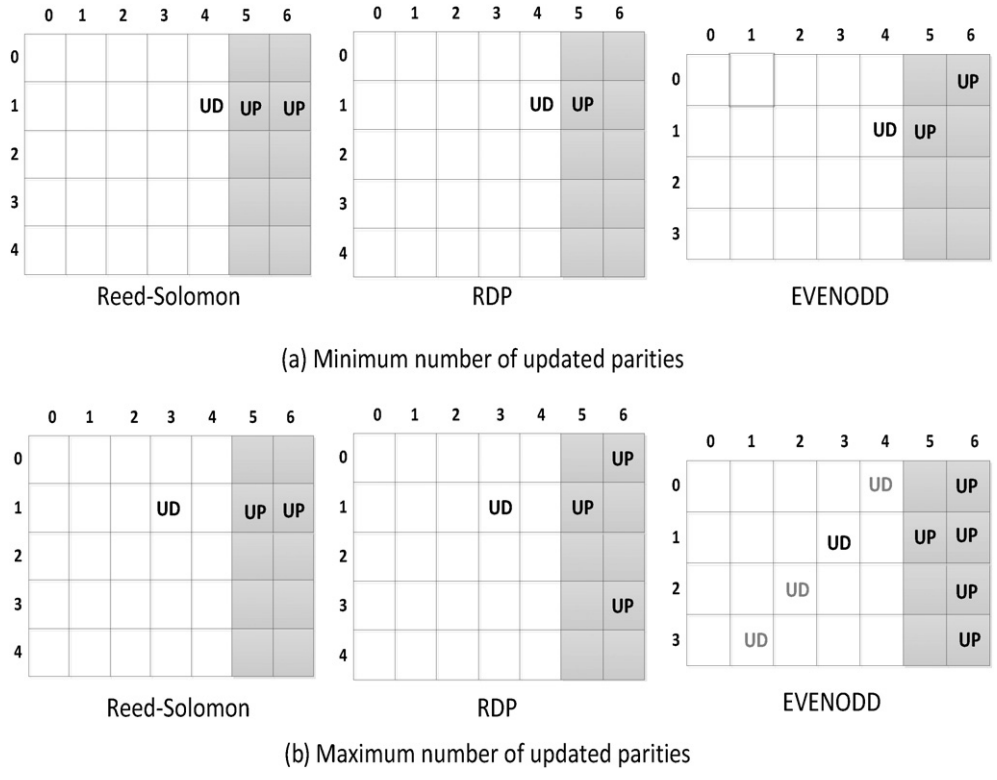

Fig. 5. Minimum and maximum number of updated parities when updating a data unit (*UD = Updated Data*, *UP*: *Updated Parity*).

updated, respectively. When updated data units are within more than one row, all the diagonal units should be updated in RDP and EVENODD. However, the number of updated parity units in the Reed–Solomon code only depends on the number of rows being updated.

Another scenario is when multiple rows of data units, which compose a full data array of the erasure code are updated. This has been indicated in Table 3 when SoR is equal to 64 KB. By updating a full data array, all parity units of the array should be updated as well. Considering the same size of the array for all erasure codes, the number of parity units being updated would be equal for all erasure codes under study.

3.2.2. Impact of request size on NoW

As mentioned earlier in Section 3.2.1, given a constant number of updated data units, different number of parity units would be updated in different erasure codes. The impact of request size on NoW can be examined in two cases considering the relative size of SUS as compared to SoR. This has been detailed as follows.

First, consider SUS is equal to SoR. As mentioned in the previous subsection, the number of corresponding parities for each data unit depends on the code pattern of erasure codes. This case has been illustrated in Fig. 5. As an example, for a 4 KB write request in a system with 4 KB stripe unit size, only one data unit should be updated. Depending on the place of the updated data unit in the array and the type of erasure codes, the number of updated parity units varies from two to five. As shown in Fig. 6, Reed–Solomon and EVENODD impose smaller and larger NoWs, respectively when SoR is equal to 4 KB.

In the second case, let's consider the request size is much larger than the stripe unit size. In this case, the updated data units are greater than a row, or may even be equal to the size of the array (SoR > NoD * SUS). As mentioned before, when the updated units are equal to the size of array, three codes under study almost impose the same number of updated parities; in this case, all parity units should be updated.

Table 3 illustrates the NoW of a disk system for different SoRs with a constant NoD and SUS. The NoW has been reported for each parity disks (i.e., P1 and P2) as well as the sum of NoWs committed to the parity disks. The starting disk number where data units are updated can affect the number of parity units being updated. Therefore, we consider two sample Starting Disk Index (SDI) in our analysis. This has been shown in Table 3 by D0 and D3. The index of the row in which the updated request is taken can also affect the number of updated parities. This is shown as Starting Row Index (SRI) in Table 3. To simplify our analysis here, we assume that all the updated data units are placed in the same array.

The difference of erasure codes in the number of updated parities can be explained by an example. Let's consider a system with the SUS of 4 KB, and an array with four data disks and two parity disks (Table 3). For a given request size of 16 KB, we have to update 4 (= 16 / 4) data units and their corresponding parity units. The locations of updated data units specify the number of updated parity units, which is different for various erasure codes. For Reed–Solomon, if the updated request starts from D0 (the first disk of the array), the updated data units are located in the same row and share the same number of parities. In this case, four data units and two parity units will be updated. For EVENODD and RDP, the number of updated parity units (P1, P2) is not equal. The parity units in the row parity (P1) are dependent on the number of rows being updated. In this case, all the updated data units are placed in the same row and share the same row parity unit, but their updated diagonal parities will be different.

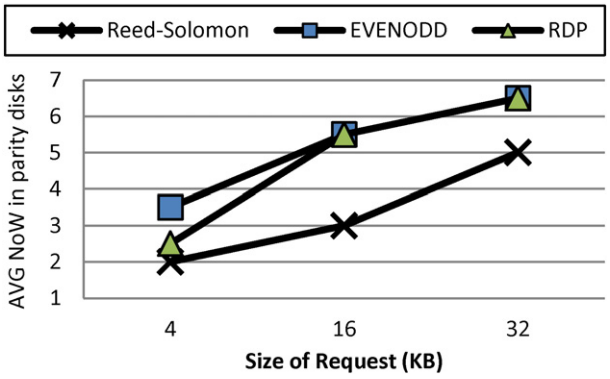

Fig. 6. Effect of SoR on average NoW in parity disks (NoD = 5, SUS = 4 KB).

According to Fig. 6 and Table 3, by increasing SoR in a system with a fixed NoD and fixed SUS, NoW is increased. According to the results reported in Fig. 6, the difference of Reed–Solomon and EVENODD codes for small (4 KB) and large (32 KB) SoR varies from 75% to 44%, respectively, while Reed–Solomon and EVENODD offer the minimum and maximum NoW among targeted erasure codes. As shown in Fig. 6, EVENODD and RDP behave in the same manner for SoR longer than a row (SoR > = SUS * NoD).

3.2.3. Impact of Size of stripe unit on the NoW

In this subsection, the impact of stripe unit size on the endurance of different erasure codes is investigated. Here we report the size of updated parity units (in KB) rather than the number of updated units to provide a fair comparison between two arrays with different stripe unit sizes. In addition, when the request size is smaller than the stripe unit size, considering the number of updated pages to calculate the number of writes improves the accuracy of our analysis.

As mentioned before, each data unit corresponds to two parity units (i.e., row and diagonal parity). Data units in the same row share the same row parity, however, the corresponding diagonal parities depend on the code pattern. The size of stripe unit is important in two aspects: 1) the size of stripe unit determines the size of updated parity units, and 2) with the same SoR, the number of *Updated Data* (UD) units is dependent on the SUS (UD = SoR / SUS).

Depending on SoR and SUS, different number of data units would be updated. If all data units are placed in the same row, the same row parity unit would be updated. The smaller stripe unit, the smaller size of parity unit would be updated. On other hand, the number of the corresponding diagonal parities depends on the number of updated data units. Hence, the smaller parity units, the larger number of data units (diagonal parities) should be updated. The total updated parity units include one row parity unit and few diagonal parity units. For fair comparison of two codes with different SUSs, the number of updated parity units should be multiplied by SUS (i.e., SoR = UD * SUS). Therefore, the difference of two stripe units in updating parity units is in the size of row parity unit. Consequently, the smaller number of parity units would be updated for small SUSs.

Eq. (1) shows how the *Size of Updated Parity Unit* (SUP) can be computed with respect to the SUS and NoD. As shown in this equation, the SUP is computed by the sum of *the Size of Updated Row Parity* (SURP) and the *Size of Updated Diagonal Parity* (SUDP). In this equation, the *Number of Row* (NoRO) stands for the number of updated parity rows

Table 4
Effect of SUS on NoW.

Code	SoR	NoD	SUS	NoW		
				p	AVG	Diff
Reed–Solomon	4 KB	5	4 KB	2 * 4	8	
	16 KB	5	4 KB	(2–4) * 4	12	166%
			16 KB	2 * 16	32	
EVENODD	4 KB	5	4 KB	(2–5) * 4	14	
	16 KB	5	4 KB	(5–6) * 4	22	155%
			16 KB	(2–5) * 16	56	
RDP	4 KB	5	4 KB	(2–3) * 4	10	
	16 KB	5	4 KB	(5–6) * 4	22	81%
		5	16 KB	(2–3) * 16	40	

in the array, and UD stands for the number of updated data units in the array. For example, let's consider the request size of 8 KB in an array with four disks. The size of updated parity units for 2 KB and 4 KB stripe unit size would be equal to 10 KB and 12 KB, respectively.

$$SUP = SURP + SUDP = NoRO \times SUS + \min\{SoR, UD \times SUS\} \quad (1)$$

Table 4 illustrates the NoW with different SUSs. According to the number of parity units corresponding to the updated data units, the NoW of erasure codes would be different. As an example, let's consider the request size of 16 KB for two different SUSs (i.e., 4 KB and 16 KB) in EVENODD. In this example, when the SUS is 4 KB, four data units should be updated. On the other hand, for a SUS of 16 KB, just one unit (16 KB/16 KB = 1) will be updated. The average size of updated parity units (NoW) is equal to 22 KB and 56 KB, for small and large stripe units, respectively. The NoW for large SUS is about 2.5× larger than NoW for small SUS.

It is important to note that if the SoR becomes smaller than the SUS, by increasing the SUS, the same NoW would be updated. It can be concluded from Table 4 that the small SUS provides lower NoW as compared to large SUS for all erasure codes. When the SoR is smaller or equal to SUS, the effect of SUS on the NoW is negligible; while for SoR much greater than SUS, the effect of SUS on NoW is significant. In this case, the difference in NoWs between small and large SUS in Reed–Solomon, EVENODD, and RDP are about 166%, 155%, and 81%, respectively.

3.2.4. Impact of number of disks on NoW

The size of array is denoted by NoD in this analysis. The effect of NoD on the NoW of different erasure codes is investigated in this subsection. We assume a fixed stripe unit size and request size for the array. When small number of data units is updated (especially one stripe unit), by increasing NoD, the difference of erasure codes in terms of updated parity unit is significant. In Reed–Solomon and RDP, the corresponding parity units for each data unit is independent of NoD, but in EVENODD, if the updated data units are placed in the main diagonal, all the parity units should be updated. The number of units in each disk is equal to the NoD in the array. Therefore, in EVENODD, the average number of updated data units increases by increasing NoD.

The impact of NoD on the NoW of different erasure codes is shown by an example given in Table 5 (when SoR = 4 KB). In this example, the request size of the updated data is 4 KB in an array with the strip unit size of 4 KB for two NoDs (NoD = 5 or 13). It is obvious from this table that Reed–Solomon imposes the minimum NoW in comparison to other erasure codes. This code also provides the same NoW for both large and small NoD. (In Reed–Solomon, NoD has no effect on NoW when the updated data is small). In EVENODD and RDP, two different values are reported for NoW, which indicate the minimum and maximum NoW for different cases.

Table 5
Effect of NoD on NoW.

	Code	SoR	NoD	SUS	NoW (p)
SoR = SUS	Reed–Solomon	4 KB	5	4 KB	2
			13		2
	EVENODD	4 KB	5	4 KB	2–5
			13		2–13
	RDP	4 KB	5	4 KB	2–3
			13		2–3
SoR > > SUS	Reed–Solomon	24 KB	5	4 KB	4
			13		2–4
	EVENODD	24 KB	5	4 KB	6
			13		7–15
	RDP	24 KB	5	4 KB	7
			13		8–10

Investigating the results reported for small SoRs in Table 5, NoD affects NoW only in EVENODD code. In this code, if the updated data units are placed in the main diagonal of the array, all the units of diagonal parity disks will be updated. For example, considering the small and large NoDs in Table 5, since the SoR and SUS are equal, only one data unit will be updated. Updating one data unit in an array of five disks in EVENODD would update two up to five parity units depending on the location of the updated data units (the average number of updated parities is 3.5). The number of updated parity units for the array of 13 disks varies from 2 to 13 units. The average number of updated parity units for NoD equal to 13 disks is 2.14× greater when NoD is equal to five.

The effect of NoD on NoW for large sequential request size is significantly different from small requests. In this case, the number of updated row units in the array is also important. Considering Reed–Solomon code in Table 5 (when SoR = 24 KB), two up to four parity units would be updated depending on whether the extension of data units are placed in one or two rows (each row corresponds to two parity units). For small NoDs, the updated data units are placed in two rows, but for large NoDs, one or two rows of data units may be updated depending on the starting disk. As can be seen in Table 5, the average number of updated parity units for small NoD is about 30% greater than large NoD. In EVENODD, besides the number of updated rows, the number of parity units in the diagonal disk is also affected by NoD. This means that in EVENODD, the impact of NoD on the updated parities is more significant than the other two codes.

As discussed earlier, the number of updated parity units in erasure codes is dependent on different parameters such as SoR, SUS, and NOD. In addition, SDI and SRI can also affect the number of updated parity units. One can formulate the number of updated parity units in terms of the above parameters. The number of updated parity units in Reed–Solomon (either P1 or P2) can be calculated according to Eq. (2). In this equation, SDI ranges from zero to NoD-1. The number of updated parity units for each request can be computed in two cases. The first case is when either SoR is less than or equal to SUS or SoR is not greater than the updated data units within the row. This means that all updated data units are placed in the same row. The second case is when the updated data units are distributed over multiple rows.

$$P1 = P2 = \begin{cases} 1, \text{if } SoR \le SUS \,||\, SoR \le SUS * (NoD - SDI) \\ \left(1 + \left\lceil \dfrac{SoR - (NoD - SDI) \times SUS}{N \times SUS} \right\rceil\right), \text{otherwise} \end{cases} \quad (2)$$

According to the analytical evaluation of Reed–Solomon and Eq. (2), if updated data units are placed within one row ($SoR \le SUS * (NoD - SDI)$), the number of updated parity units will be independent of NoD. This mainly happens for the traces with small random request size. On the other hand, for large sequential request size, NoD affects the number of updated parity units.

The main findings of the analytical evaluation presented in this section can be summarized as follows:

1) Considering the code pattern of erasure codes, Reed–Solomon and EVENODD impose smaller and larger NoW, respectively. As shown in Fig. 5, Reed–Solomon would update at most two parity units for each updated data, while EVENODD would update more than two parity units.
2) The difference of erasure codes in terms of NoW for small SoR is significant (up to 75%), while for large SoR all erasure codes impose almost the same NoW. The reason is that, for large size of request the updated data are covered a great number of data units in the layout, which would update all the parity disks. As the ratio between parity disks and data disks is almost the same in the layout of different erasure codes, for large SoR, the same number of parity units would be updated. As a result, the same NoW would be imposed to the system.

3) By increasing SUS when SoR is greater than SUS, NoW is increased in all target erasure codes. Smaller SUS provides the minimum NoW for the disk subsystem. The reason is that when SoR is larger than SUS, the updated data will in the same row and will share the same row parity unit. The smaller the size of SUS, the smaller size will be updated.
4) NoW in Reed–Solomon is independent of NoD for small request size, while in EVENODD, NoW is strictly dependent on NoD. This is because of the layout of Reed–Solomon, in which the corresponding parities for each data are placed in the same row. In Reed–Solomon, irrespective of the number of disks in the array, only two parity units would be updated for each data unit.

4. Experimental results

In this paper, in order to investigate the effect of erasure codes on the performance and endurance of storage systems, three erasure codes (Reed–Solomon, EVENODD, and RDP) are implemented using the SSD extension of DiskSim v4.0 simulator [19]. The implementation of these codes is appended to the source code of DiskSim, evaluating their effect on the parameters of systems (i.e., performance and endurance).

4.1. Experimental setup

The experimental setup used in DiskSim has been depicted in Table 6. The storage system includes an array of seven disks, with the size of 32 MB. Please note that the default value for NoD in the experiments is 7. Storage vendors usually do not use larger value for NoD since it significantly deteriorates the reliability of the RAID array. Five disks hold data and two disks store parities. As the main consideration in this paper is to evaluate the effect of different erasure codes on the performance and endurance of system, for the sake of simplicity in the implementation, dedicated parity disks (similar to RAID4) are used for this purpose. We believe that the structure of RAID (i.e., dedicated or distributed) has no effect on the relative comparison among different erasure codes, when these codes are implemented on the same structure. Employing distributed parity disks would enhance the performance and endurance of the disk subsystem for each erasure code individually, but we believe that the relative performance remains almost constant.

In the experiments, NoW, NoC, and ART are reported for three erasure codes. The simulation is done by running six I/O traces as representatives of two class of I/O traces, i.e., sequential-dominated workloads (Postmark [20], Financial1 [21], and Exchange [22]) and random-dominated workloads (Build08-10 [23], CAM03-lvm0 [24], and Iozone2 [25]). The characteristics of these workloads are reported in Table 7.

Three sets of experiments are analyzed in our simulations. Various request sizes, stripe unit sizes, and number of disks are evaluated for the three target erasure codes. In each set of experiments, the number of clean cycles and the average response time are reported as a metric of endurance and performance, respectively. The number of write cycles is calculated with respect to the effect of all the updated data units on the parity disks. The average response time is the average response time over all requests within a given trace. NoW is also reported for each set of experiments to provide a comparison with NoC. In the following subsections, the results of our experiments are presented.

Table 6
Experimental setup.

Parameter	Value	Parameter	Value
Number of Disks	7	Page Read Latency	25 μs
Disk Size	32 MB	Block Erase Latency	1.5 ms
Plane per Die	2	Page Write Latency	200 μs
Block per Plane	1024	The number of buses	1
Page per Block	4	Flash packages	1
Page size	4 KB	Dies per package	1
Reserve pages percentage	15%	Garbage Collection	Greedy
Garbage Collection Threshold	5%	Cache Policy	No cache

Table 7
Characteristics of the evaluated I/O traces.

Workload	Total I/O request	Write request	Avg. size write request	Std. dev. write request
Build08–10	100,000	28%	76.3 KB	1116.4
CAM03-lvm0	100,000	38%	9.7 KB	28.3
Exchange	200,000	64%	675 KB	952
Financial1	100,000	52%	818 KB	4294.8
Iozone2	94,641	99.9%	4 KB	4.8
Postmark	62,257	17%	833 KB	2395.8

4.2. Number of disks

Here, the effect of NoD on the performance and endurance of storage systems is analyzed. In this analysis, a constant size of stripe unit (i.e., SUS = 4 KB) is considered for storage system. The effect of the number of disks on NoW, NoC, and ART of target erasure codes are illustrated in Figs. 8, 9, and 10, respectively. In each figure, the results of six different traces are reported; the first row in each figure reports the results for small SoR, while the second row shows the results of large SoR.

As shown in Fig. 8, for all traces with SoR larger than SUS (Fig. 8.b, c, d, e, and f), by increasing NoD, the NoW in all erasure codes is decreased. As the average number of writes is reported in the evaluation section, it is generally expected that the average number of writes is decreased when the number of disks is increased. Furthermore, in the cases of small or large size of stripe unit, the corresponding parity units for updated data are different. For the case of small size of request, the code pattern of erasure codes significantly affects the NoW. As mentioned previously in Section 3.2.4, the number of updated parity units in EVENODD is a function of number of disks when the updated data is placed in the main diagonal. For the case of large size of request, by increasing the number of disks, a large number of disks share the same row parity unit. Thus, smaller number of parity units would be updated.

For example, NoW is decreased by 21%, 17%, and 17%, in Reed–Solomon, EVENODD, and RDP, respectively, in Fig. 8.f. On the other hand, as shown in Fig. 8.a, for traces with SoR equal to SUS, when the number of disks is doubled from 7 to 15 disks, NoW for EVENODD is increased by 11.7%, while NoW is decreased by 2.3% for RDP, and is remained constant for Reed–Solomon. As mentioned before, in Reed–Solomon, NoW is independent of NoD, while in EVENODD, the number of disks in each row can affect the number of updated parity units. It is clear from Fig. 8 that Reed–Solomon imposes the minimum NoW to the system compared to EVENODD and RDP.

As shown in Fig. 9, by increasing NoD from 7 to 15, NoC is decreased in the three erasure codes for traces with both small and large SoR. Because of distributing writes among larger NoD, the average number of writes for each disk is decreased, requiring smaller NoC for each

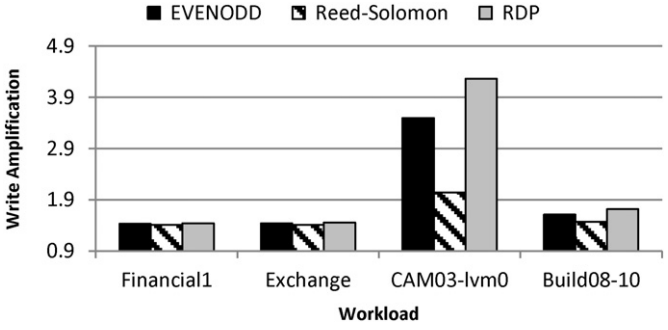

Fig. 7. write amplification of different erasure codes on various workloads (Financial1, Exchage: random workload, CAM03-lvm0, Build08-10: sequential workload).

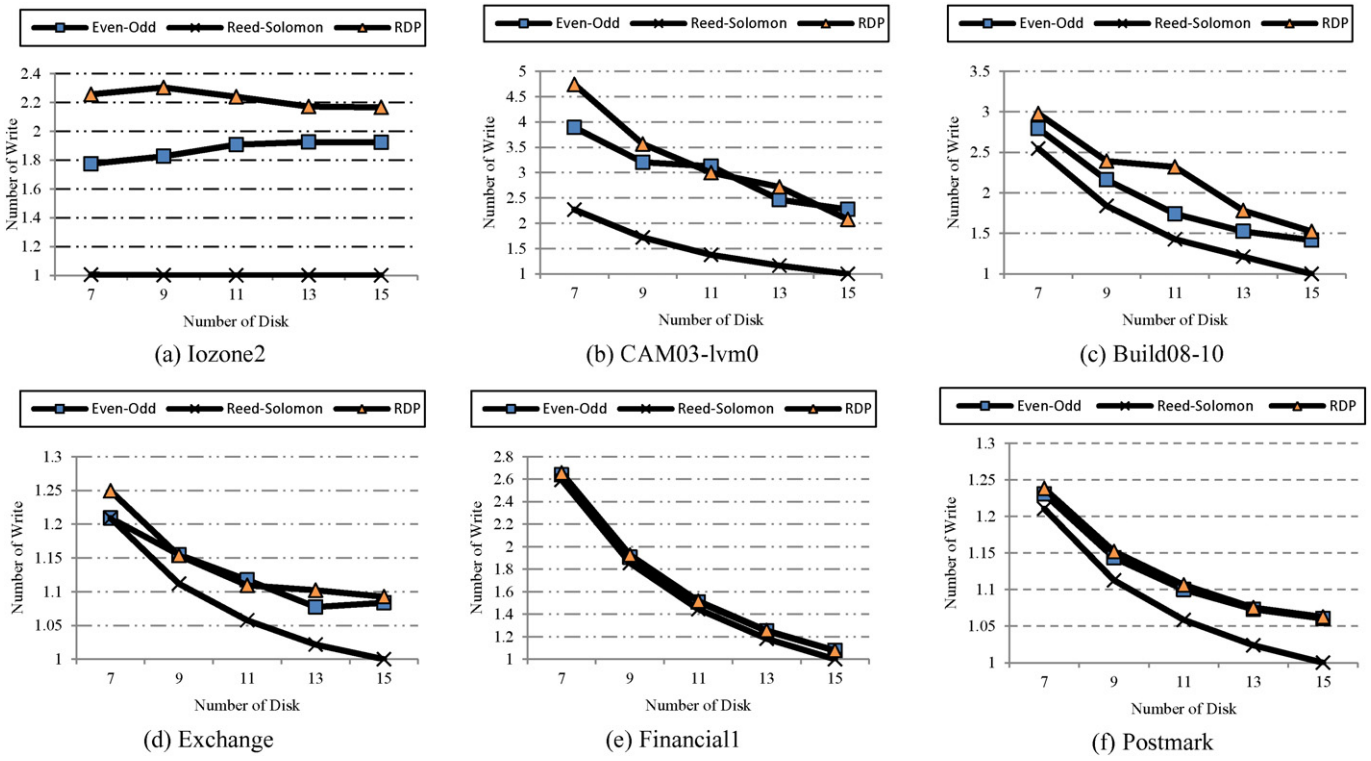

Fig. 8. Effect of number of disks on NoW for different erasure codes: (a) Iozone2, (b) CAM03-lvm0, (c) Build08-10, (d) Exchange, (e) Financial1, and (f) Postmark.{SUS = 4 KB}.

disk. For traces with large SoR, by increasing the NoD, the average NoC in all trace is decreased with significantly different rates when running various traces. For example as shown in Fig. 9.b, by doubling the NoD from 7 to 15 disks, NoC is decreased by 155%, 170%, and 175% in EVENODD, Reed–Solomon, and RDP, respectively. On the other hand, for traces with small SoR, all erasure codes have almost the same rate of decrement. For example, as shown in Fig. 9.a for a trace with small SoR, by doubling NoD from 7 disks to 15 disks, NoC is decreased

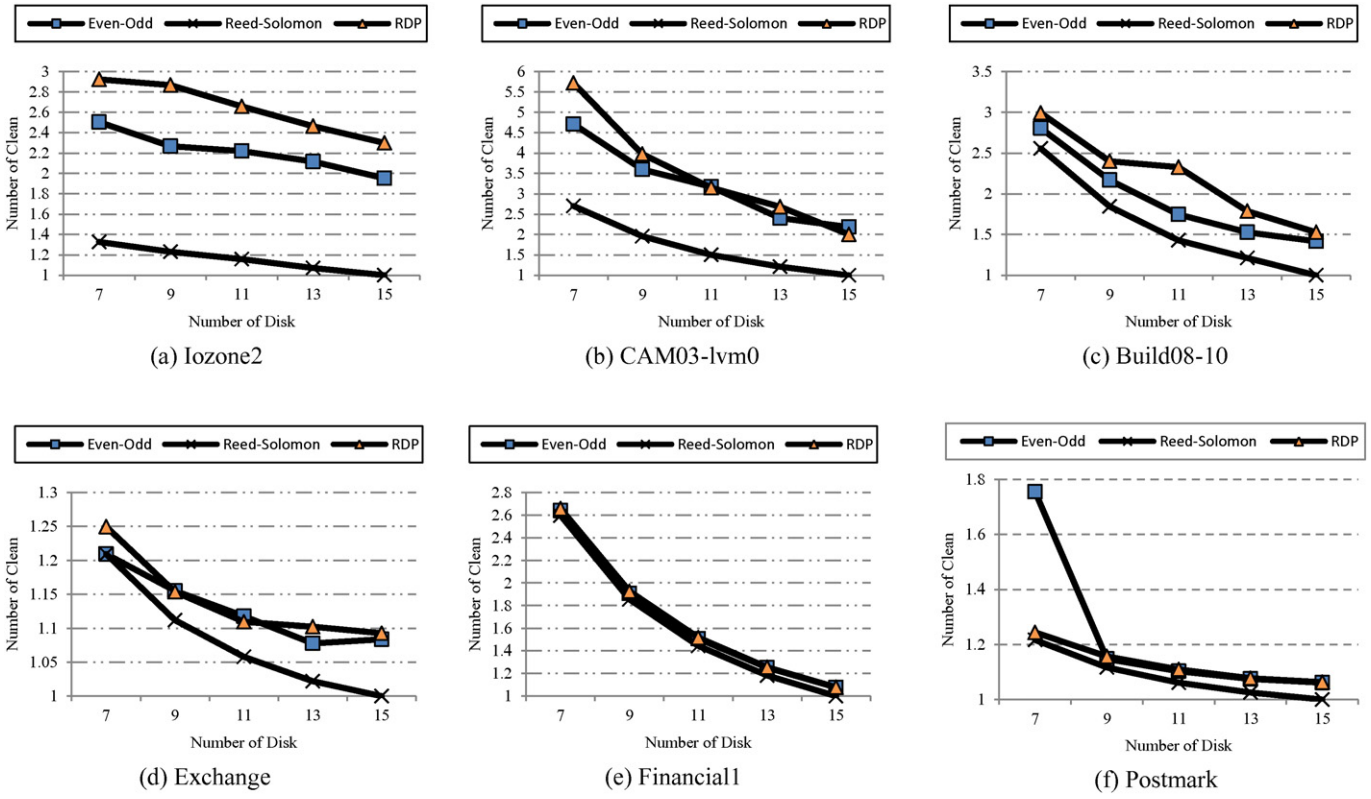

Fig. 9. Effect of number of disks on NoC for different erasure codes: (a) Iozone2, (b) CAM03-lvm0, (c) Build08-10, (d) Exchange, (e) Financial1, and (f) Postmark. {SUS = 4 KB}.

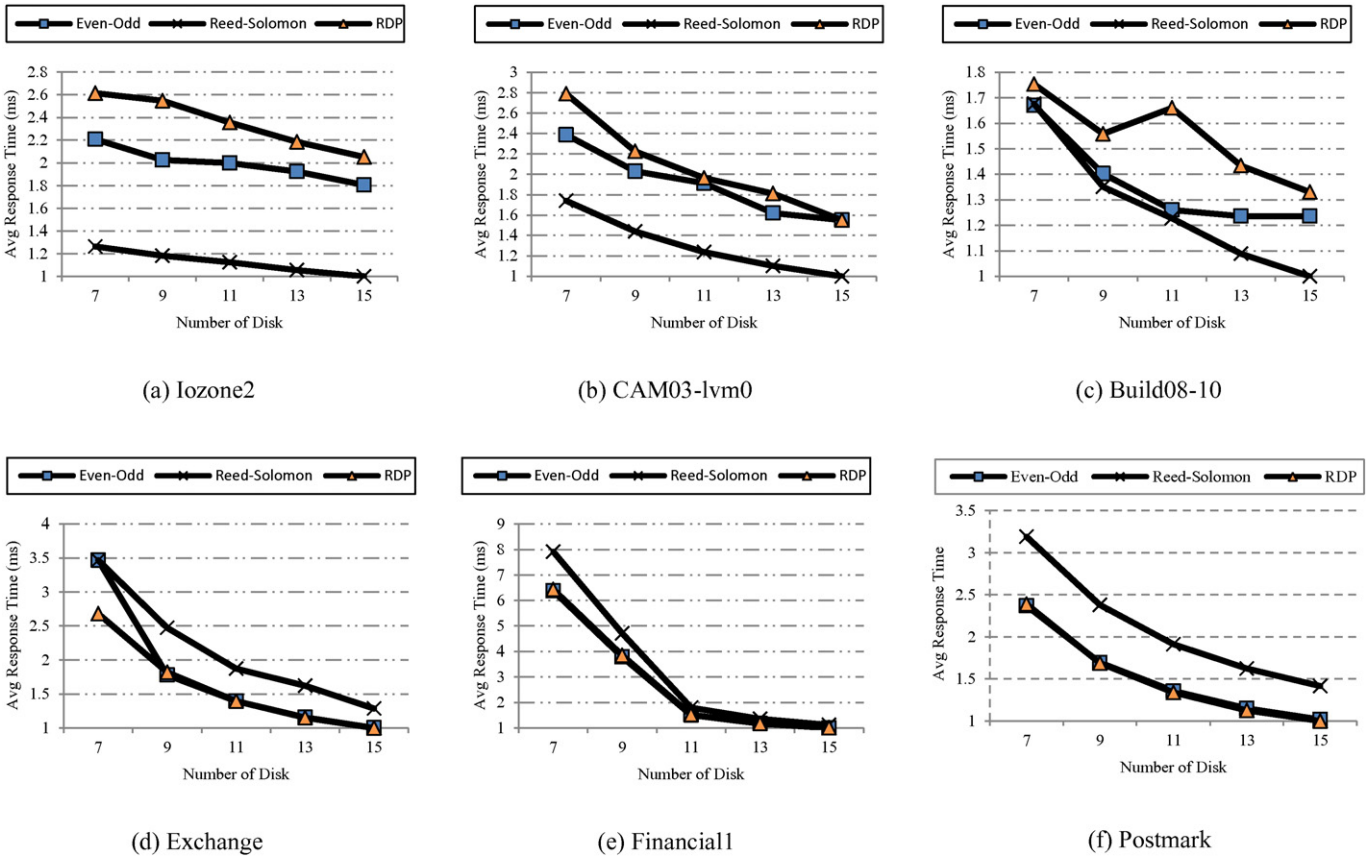

Fig. 10. Effect of number of disks on ART for different erasure codes: (a) Iozone2, (b) CAM03-lvm0, (c) Build08-10, (d) Exchange, (e) Financial1, and (f) Postmark. {SUS = 4 KB}.

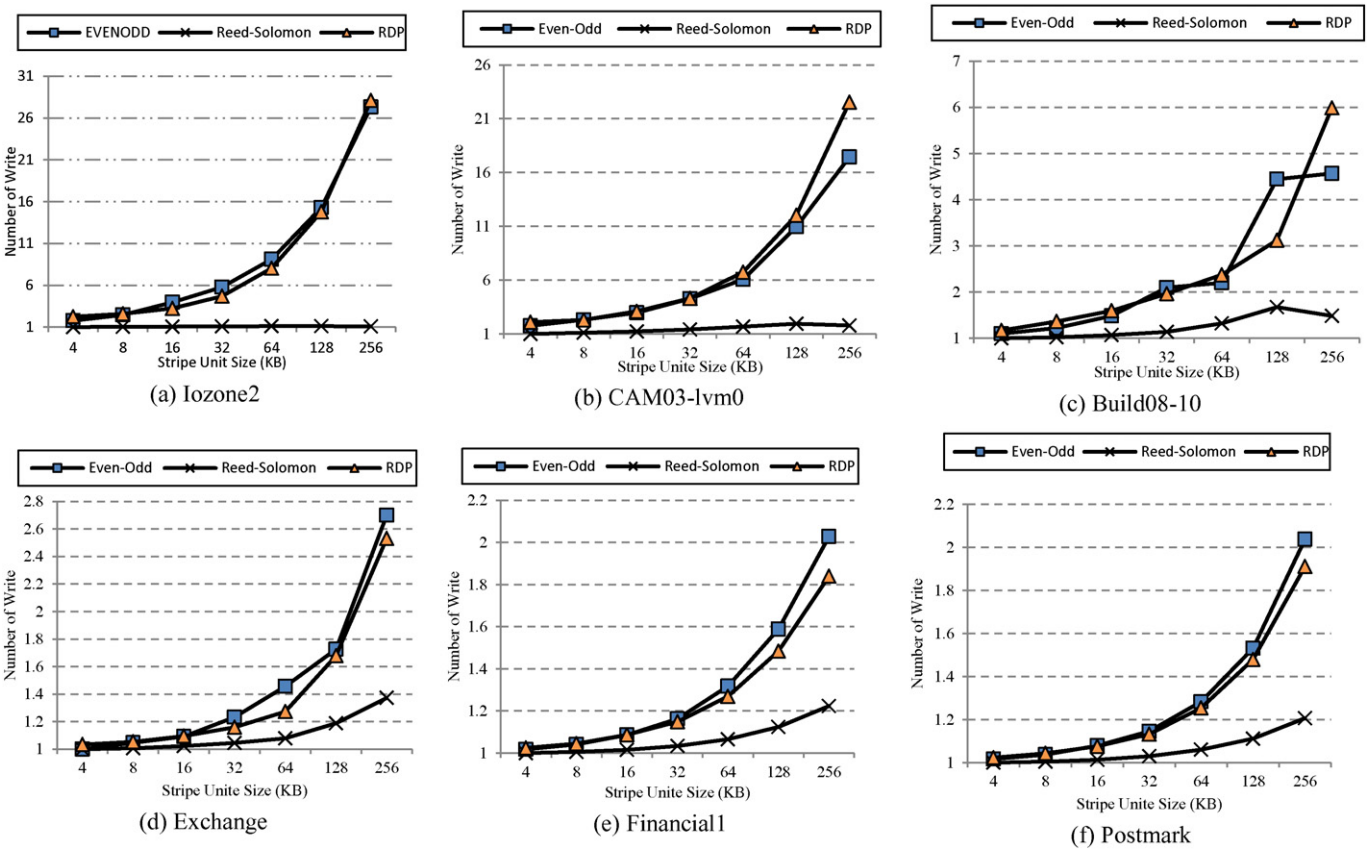

Fig. 11. Effect of strip unit size on NoW for different erasure codes: (a) Iozone2, (b) CAM03-lvm0, (c) Build08-10, (d) Exchange, (e) Financial1, and (f) Postmark. {NoD = 7}.

by 29.4%, 26.7%, and 25% for RDP, EVENODD, and Reed–Solomon, respectively.

As shown in Fig. 10, by doubling NoD from 7 to 15 disks, ART is decreased in all erasure codes. This means an improved performance for erasure codes in large NoDs. As mentioned earlier, the performance in this paper refers to the I/O access performance. Thus, the performance is extremely affected by the number of cleans, writes, and reads. The effect of NoD on the performance of erasure codes in traces with large SoR (Fig. 10.d, e, and f) is greater than traces with small SoR (Fig. 10.a, b, and c). For example as shown in Fig. 10.d and a, in Reed–Solomon, ART is decreased by 180% for large SoR, while this decrement is about 26% for a small SoR. In addition, the Reed–Solomon has the greatest ART in comparison to EVENODD and RDP for large SoR. On the other hand, the ART of Reed–Solomon is significantly smaller than the other erasure codes for small SoR. As shown in Fig. 10.a, the performance of Reed–Solomon for small SoR is greater than EVENODD and RDP by 80% and 100%, respectively.

As shown Figs. 10 and 9, Reed–Solomon outperforms the other erasure codes in terms of performance and endurance for small SoR. However, RDP has the lowest performance and endurance as compared to EVENODD and Reed–Solomon for small SoR. Additionally, the difference between EVENODD and RDP for small SoR in terms of NoW, NoC, and ART is considerable, while these two codes behave similarly for large SoR.

4.3. The size of stripe units

Here, the effect of various stripe unit sizes (i.e., 4 KB, 8 KB, 16 KB, 32 KB, 64 KB, 128 KB, and 256 KB) on the performance and endurance of erasure codes is evaluated. Increasing SUS has different effects on the NoW, NoC, and ART of different erasure codes, as shown in Figs. 11, 12, and 13. This analysis is performed on six different traces. By increasing SUS in traces with large SoR, the performance and endurance of erasure codes are decreased when SUS is increased for all three erasure codes. For instance, when the stripe unit size is set to 4 KB, the performance and endurance of SSD-based RAID array in EVENODD are improved up to 100% and 70%, respectively, as compared to the stripe unit size of 128 KB (Figs. 13.f and 12.f). This improvement in the performance and endurance of Reed–Solomon is 43% and 20%, respectively. On the other hand, considering traces with small SoR, by increasing SUS, the performance and endurance of EVENODD and RDP are decreased significantly (up to 21× for endurance and 17× for performance at Figs. 12.b and 13.b, respectively). However, the performance and endurance of Reed–Solomon remains almost constant for different SUSs in traces with small SoR.

Reed–Solomon demonstrates different performance in comparison to the other erasure codes in traces with small and large SUS, while its endurance is always better than two other erasure codes. For example, as shown in Fig. 13.f, considering a workload with large SoR, Reed–Solomon imposes 40% lower performance in comparison with RDP and EVEODD when SUS is equal to 4 KB. However, when SUS is equal to 256 KB, Reed–Solomon improves performance by 60% as compared to RDP and EVEODD. In addition, EVENODD degrades the performance as compared to other erasure codes for large SUS. Moreover, EVENODD has lower endurance as compared to the other erasure codes for any SUS.

For small SUS, the NoW, NoC, and ART of three codes are almost equal, but by increasing SUS, the difference of the erasure codes in terms of performance and endurance becomes considerably high. This shows that although different erasure codes perform similarly for small SUS, for large SUS, the difference of erasure codes in terms of performance and endurance is significant. In this case, choosing the

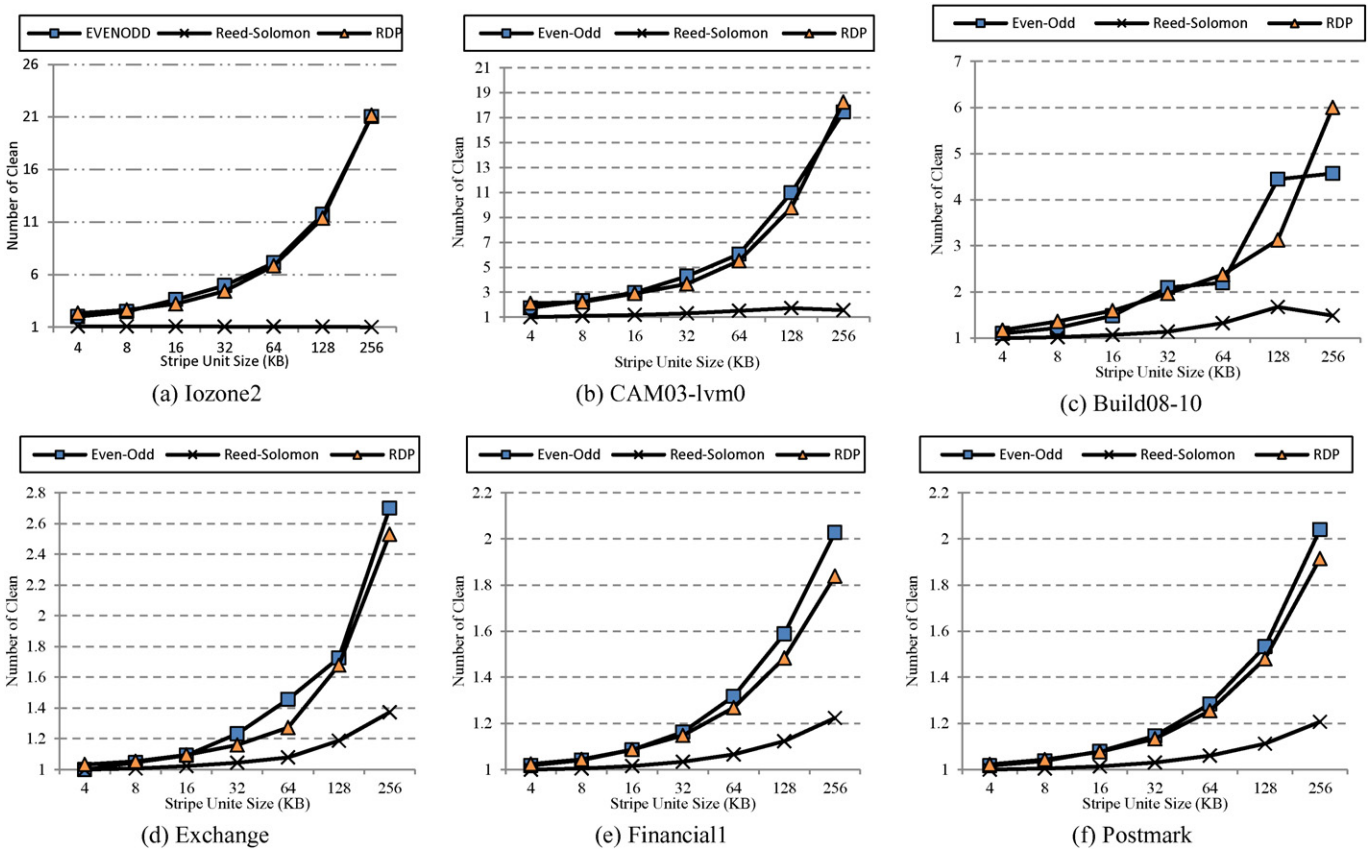

Fig. 12. Effect of stripe unit size on NoC for different erasure codes: (a) Iozone2, (b) CAM03-lvm0, (c) Build08-10, (d) Exchange, (e) Financial1, and (f) Postmark. {NoD = 7}.

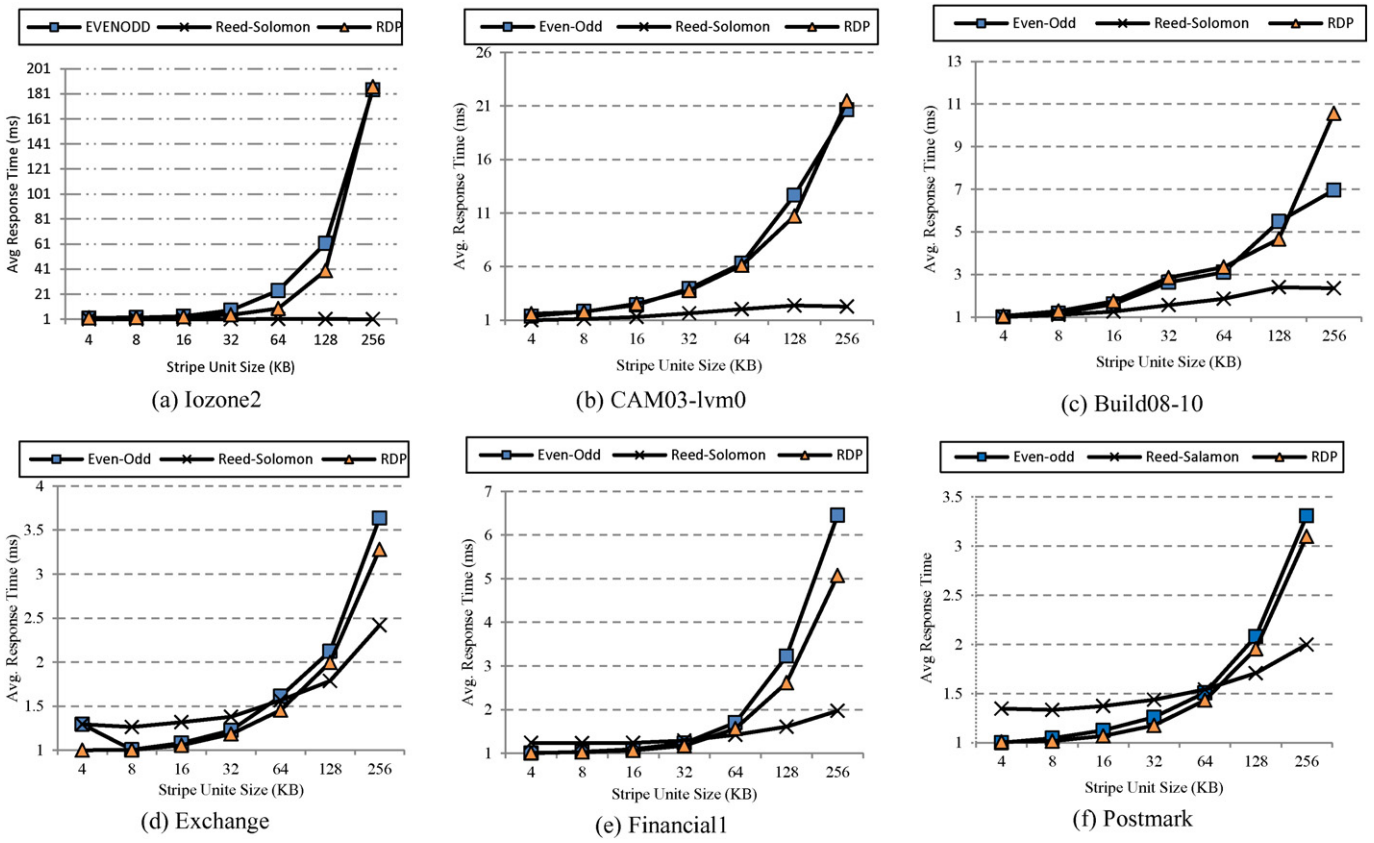

Fig. 13. Effect of stripe unit size on ART for different erasure codes: (a) Iozone2, (b) CAM03-lvm0, (c) Build08-10, (d) Exchange, (e) Financial1, and (f) Postmark. {NoD = 7}.

appropriate erasure code would significantly improve the performance and endurance of storage subsystems.

4.4. The average write amplification

Here, the effect of different erasure codes on the write amplification is evaluated. This evaluation determines the average write amplification for the considered erasure codes in terms of different page reservation percentages. We perform this evaluation on three workloads including random (i.e., CAM03-lvm0) and sequential (i.e., Exchange, and Financial1) dominated ones. The results of comparing the write amplification of different erasure codes are reported in Fig. 14. As shown in this figure, the reserve page has no effect on the write amplification of different erasure codes. As reported in this figure, in the case SUS and NoD are set to 4 K and 7, respectively, RDP imposes greater write amplification as compared with EVENODD and Reed–Solomon. It is admitted the results of comparing target erasure codes in terms of the number of writes as shown in Fig. 8, where RDP has the maximum number of writes as compared with other erasure codes. It can be concluded that a code with higher write amplification imposes the higher NoW to the disk subsystem. Fig. 7 reports a comparison among different erasure codes in terms of write amplification when running four different workloads (Financial1 and Exchange are random-dominated workloads) (note CAM03 and Build08 are sequential-dominated workloads). In this comparison, the default experimental setups are applied in the system. In both kinds of workloads, Reed–Solomon imposes the minimum write amplification compared to RDP and EVENODD. As shown in Fig. 7, the difference among erasure codes is considerable

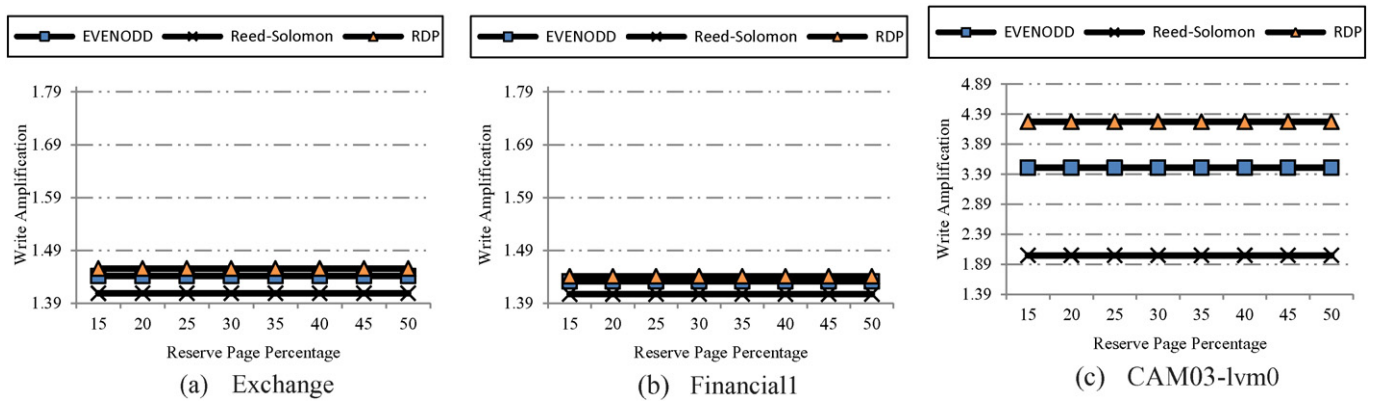

Fig. 14. Effect of page reservation on the write amplification for different erasure codes: (a) Exchange, (b) Financial1, (c) CAM03-lvm0 {NoD = 7, SUS = 4 K}.

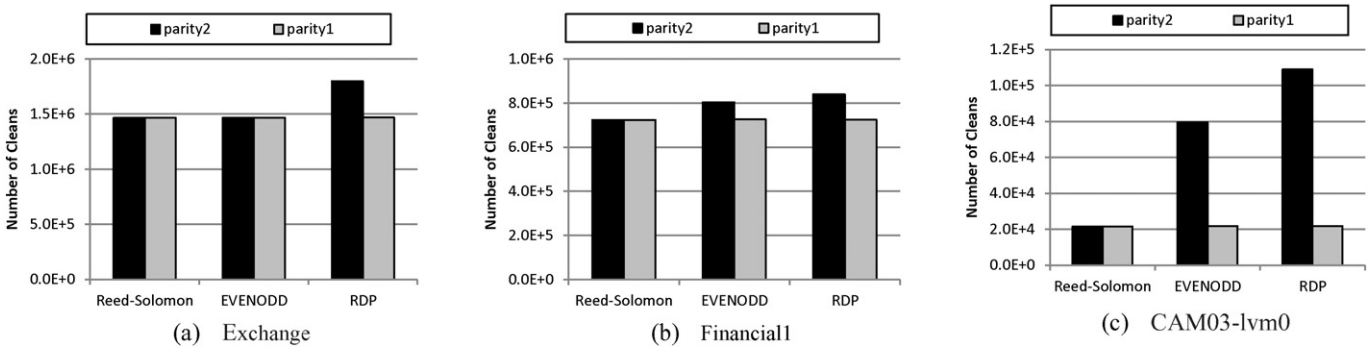

Fig. 15. The distribution of NoC between parity disks for different erasure codes: (a) Exchange, (b) Financial1, (c) CAM03-lvm0 {NoD = 7, SUS = 4 K}.

when running random workloads. For CAM03-lvm0 workload, RDP and EVENODD impose about 2.2× and 1.8× higher write amplification as compared with Reed–Solomon. The reason is that in a random workload, each updated data imposes extra writes to the system while in the sequential workloads some of data share the same parity unit as well as extra writes.

We also evaluate the effect of page reservation percentage on the performance and endurance of different erasure codes. The results of simulation show that for each of the erasure codes, the evaluated parameters, i.e., NoW, NoC, and ART remain unchanged by increasing the page reservation percentage. For the brevity of the paper, detailed results have not been included in the paper.

4.5. The distribution of number of cleans between parity disks

The distribution of NoC among disks is a key factor that should be considered in the endurance evaluation. As we use the dedicated parity disks in our evaluation, the distribution of NoC among data disks is same for all erasure codes, but the distribution of NoC among parity disks are different. As shown in Fig. 15, Reed–Solomon has evenly distributed NoC among parity disks for different workloads. It can be concluded from this evaluation that Reed–Solomon does not only impose the minimum number of cleans to the system, it also evenly distributes the NoC across parity disks.

The number of cleans on parity disks are not the same due to different patterns for computing each of parity disks in target erasure codes. As shown in Fig. 15, both parity disks impose the same number of cleans in Reed–Solomon, while the diagonal parity disk imposes higher NoC as compared to row parity disk in EVENODD and RDP. This is due to the layout of erasure codes. In Reed–Solomon, as show in Fig. 1, both parity disks are computed with the same pattern in the layout of erasure code. While, in EVENODD and RDP as shown in Figs. 2 and 3, respectively, the row and diagonal parity are computed with different patterns. The difference of NoC among parity disks is considerable for random workload (i.e., CAM03-lvm0), because in this case, updated data units do not share the same parity units. Hence, for each updated data unit, the difference between the corresponding row and diagonal updated parity disks should be considered individually. The effect of NoD and SUS are also evaluated for the parity disks in EVENODD in Figs. 16 and 17, respectively. By increasing SUS, the difference between parity disks is increased when running random or sequential workloads. By increasing NoD, the difference of NoC between parity disks is not the same for random and sequential workloads. This difference is decreased for sequential workloads, while it is increased for random workloads.

It can be concluded from this evaluation that Reed–Solomon not only imposes the minimum number of cleans to the system, it also distributes the NoC between parity disks as well.

4.6. Discussion

The main observations concluded from the results are summarized as follows:

1. System characteristics (i.e., SUS and NoD) can significantly affect the endurance and performance of erasure codes. The experimental results revealed that to achieve the best performance for erasure codes, small SUS and a large number of SSDs should be used in the configuration of disk subsystem. In applications with large SoR, the performance of a disk subsystem is improved by 295% when NoD is doubled (from 7 to 15), while this improvement is 51% in applications with small SoR (by averaging of all erasure codes). It is notable to mention that the customary size of stripe unit reported by storage venders such as IBM and Hitachi in SSD-based storage system is 256 KB–512 KB [15,16], which is not an optimal stripe unit size for the performance and endurance of disk subsystem. It is worthy to mention that the optimal configuration for SSDs may be different than for HDD-based disk subsystems due to the different characteristics of these two technologies. For example, the well-known storage vendors employ large size of stripe units since the arm positioning time in HDDs dominates the data

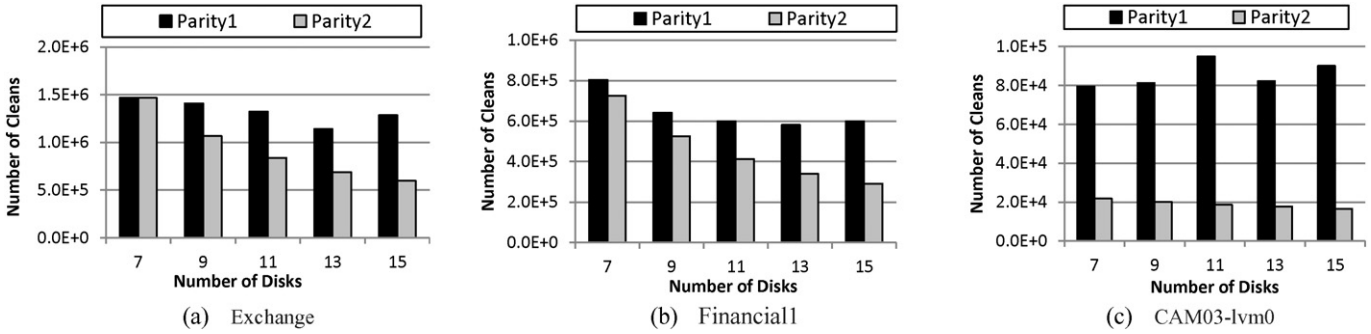

Fig. 16. The effect of NoD on the distribution of NoC between parity disks for EVENODD: (a) Exchange, (b) Financial1, (c) CAM03-lvm0 {NoD = 7, SUS = 4 K}.

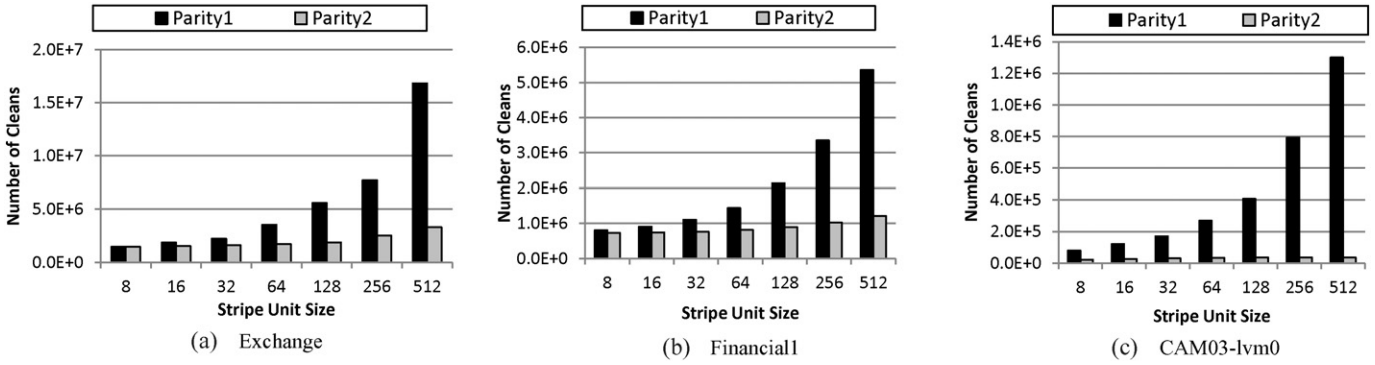

Fig. 17. The effect of SUS on the distribution of NoC between parity disks for EVENODD: (a) Exchange, (b) Financial1, (c) CAM03-lvm0 {NoD = 7, SUS = 4 K}.

transfer time in small stripe unit sizes. However, our study on SSD-based RAID shows that large stripe unit size is not optimal anymore for the SSD-based disk subsystems. The results of our evaluations also confirm the results of [29] for stripe unit size in SSD-based storage system.

2. When the stripe unit size is increased, the performance of XOR-based erasure codes (i.e., EVENODD and RDP) is decreased 3.7× faster than non-XOR based ones (i.e., Reed–Solomon). On the other hand, the performance of different erasure codes increases with almost the same rate when NoD is increased from 7 to 15.
3. With regard to SoR when running different applications, NoD has significantly different effect on the performance of erasure codes. For example, if Reed–Solomon is employed in a system with small SoRs (Fig. 9.a), by doubling the NoD from 7 to 15, the performance is improved by 26%, while the performance improvement is 125% for large SoRs (Fig. 9.f).
4. The experiments reveal that the characteristics of I/O traces have a significant impact on the behavior of erasure codes for different system characteristics. For example, by doubling NoD from 7 to 15, the endurance of Reed–Solomon is improved up to 170% when running *Finacial1*, while this improvement is 20% while running *Exchange*.
5. Considering the average of the results of all workloads, it can be concluded that: the endurance and performance of Reed–Solomon are on average 90% and 60% higher than other erasure codes, respectively. In addition, this code provides the even distribution of NoC between parity disks as compared with EVENODD and RDP.
6. The results show that configuring a disk array with a 4 KB stripe unit size will improve the endurance and performance of EVENODD by 1.8× and 2.9×, respectively, as compared to 128 KB stripe unit size.
7. As noted in [26], erasure codes are used to protect systems against the failures of entire disk or a sector, where the position of failed disks is specified. On the other hand, the wear-out of SSDs will lead to a permanent failure on the target flash chip, when the number of P/E cycles reaches to the endurance limit. One of the important open issues in the community is modeling the impact of endurance on the reliability of storage systems. There have been several models taking into account different failure mechanisms such as latent sector error or disk operation failure. However, there is no study to investigate the impact of limited endurance on storage reliability. Employing the erasure codes will increase the number of writes which in turn accelerate SSD wear-out. Investigation of such impact on overall SSD reliability is beyond the scope of this paper and is part of our ongoing research work.

5. Related works

Several studies on erasure codes have been done in recent decades. These studies include designing new erasure codes or analyzing and enhancing the performance of these codes. Different approaches for performance enhancement are also proposed including improving XOR complexity [7,27], optimizing I/O load balancing [8], and I/O optimal recovery [28].

A comparison of different erasure codes in terms of the NoW is performed in [14], indicating the impact of the structure of erasure codes on NoW imposed to the disk subsystem. A study on the impact of stripe unit size on the performance and endurance of SSD-based RAIDs reveals that unlike HDDs, the small stripe unit size provides the optimum performance in SSD-based system [29]. The authors of [30] compared the implementation of different erasure codes in terms of encoding and decoding. The main aim of their proposed study is to compare the implementations of erasure codes, to discern whether theory matches practice. They also attempt to demonstrate how parameter selection, especially that concerns memory, has impact on a code's performance.

To analyze the effect of erasure codes on the performance of the multiprocessor systems, two erasure codes, i.e., RDP and X-Code, are utilized in FT-MPI [31]. In their proposed study, the effect of each code on the checkpoint is evaluated. The checkpoint indicates the length of time for storing the state of processor in the memory. The shorter this time, the system has better performance. The results show that the checkpoint is shorter when using RDP in comparison with X-Code resulting in improved performance in a FT-MPI system.

In distributed systems, a file is divided into multiparts, each part is stored in one server. In these systems, erasure codes are used to protect files against failures. The study presented in [32] considers a network including 300 servers. Each 100 MB file is divided into N files and all N files along with M parity files are put on the server. Reed–Solomon and LDPC are employed in the servers to generate parity bits. This study considers the download time to evaluate the performance of each erasure code. The results show that Reed–Solomon imposes shorter download time to the server in comparison with LDPC.

In the work presented in [33], a developed redundancy placement algorithm is presented which determines the best placement for maximum reliability. For determining the best placement, an analytical model called Relative MTTDL Estimate (RME) is used to compare different placements of erasure codes.

In addition to the methods employed in the RAID arrays to enhance the endurance of storage systems [34,35], some papers concentrated on the reliability by modifying the architecture of storage systems and RAID structure [36]. Some studies also investigate the performance of SSD-based storage systems in terms of write-cycles [37], response time for parity update [38,39], and write buffer management [40,41].

Another major concern in storage systems is *Silent Data Corruption* (SDC) which refers to undetectable corrupted data in disk drivers [43,44]. SDCs could result in fatal errors in the application level if it is not properly handled in the disk subsystem. A conventional RAID array is designed to protect against only detectable errors with no protection against SDCs. To protect data against SDC, an integrity protection scheme is added to the RAID layer. Several schemes have been proposed to cope with silent data corruptions which are classified in four classes

[42]: self-check summing [43], physical identity [45], version mirroring [45], and checksum mirroring [44]. Using either of these schemes, an SDC in a block of data could be detected and can prevent manifesting such error to higher system levels. The detailed discussion of such techniques is beyond the scope of this paper.

6. Conclusion

Erasure codes are employed in RAID arrays to improve the reliability of storage systems. Applying SSDs in the configuration of RAID array brings new concerns in evaluating erasure codes. In this paper, a comprehensive analysis investigating the performance and endurance of erasure codes was conducted by considering the code patterns of erasure codes and the configuration of RAID arrays. Our experimental results showed that Reed–Solomon is the best choice for applications with small and random request sizes. In addition, modifying the number of disks in an array and the size of stripe unit would affect the performance and endurance of target erasure codes differently. Our results also revealed that the small size of stripe unit provides the best performance and endurance for SSD-based storage systems.

As a future work, the same comparison can be done on open-source implementation of erasure codes to investigate the effect of these codes on the performance and endurance of systems. Comparing the reliability of these codes in the SSD-based systems is part of our ongoing research.

References

[1] M.A.A. Sanvido, F.R. Chu, A. Kulkarni, R. Selinger, NAND flash memory and its role in storage architectures, IEEE Mag. 96 (11) (Nov. 2008) 1864–1874.
[2] Int. Technology Roadmap for Semiconductors, ITRS, 2007.
[3] D. Patterson, G. Gibson, R. Katz, The case for raid: redundant arrays of inexpensive disk, Proc. ACM SIGMOD Conf., 1988.
[4] I. Reed, G. Solomonm, Polynomial codes over certain finite fields, J. Soc. Ind. Appl. Math. (1960) 300–304.
[5] M. Blaum, J. Brady, J. Bruck, J. Menon, EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures, IEEE Trans. Comput. 2 (44) (1995) 192–202.
[6] M. Blaum, R. Roth, On lowest density MDS codes, IEEE Trans. Inf. Theory 45 (1) (1999) 46–59.
[7] J. Plank, The RAID-6 liberation codes, Proc. USENIX FAST 2008, pp. 97–110.
[8] Y. Cassuto, J. Bruck, Cyclic lowest density MDS array codes, IEEE Trans. Inf. Theory 55 (4) (2009) 1721–1729.
[9] L. Xu, J. Bruck, X-Code: MDS array codes with optimal encoding, IEEE Trans. Inf. Theory 45 (1) (1999) 272–276.
[10] C. Jin, H. Jiang, D. Feng, L. Tian, P-Code: a new RAID-6 code with optimal properties, Proc. Int. Conf. Supercomputing (ICS) 2009, pp. 360–369.
[11] J. Blomer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, D. Zuckerman, An XOR-based erasure-resilient coding scheme, Technical Report TR-95-048. Int. Computer Science Institute, 2004.
[12] P. Corbett, B. English, A. Goel, et al., Row-diagonal parity for double disk failure correction, Proc. 3rd USENIX Conf. File and Storage Technologies 2004, p. 1-1.
[13] Ch. Wu, X. He, G. Wu, HDP code: a horizontal–diagonal parity code to optimize I/O load balancing in RAID-6, 41st Int. Conf. Dependable Systems & Networks (DSN) 2011, pp. 209–220.
[14] S. Alinezhad, S.G. Miremadi, H. Asadi, On endurance of erasure codes in SSD-based storage systems, Proc. 17th CSI Int. Symposium on Computer Architecture & Digital Systems (CADS'13), Tehran, Iran October 2013, pp. 67–72.
[15] IBM Corporation, IBM Storwize family storage systems with SAS workloads on IBM power systems servers, Technical Report, 2013.
[16] Hitachi Data System, Hitachi Unified Storage VM Architecture Guide Hitachi, Technical Report, 2013.
[17] W. Bolosky, J. Douceur, D. Ely, M. Theimer, Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs, Proc. Int. Conf. Measurement and Modeling of Computer Systems 2000, pp. 34–43.
[18] P. Druschel, A. Rowstron, Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility, Proc. Eighteenth ACM Symposium on Operating Systems Principles 2001, pp. 188–201.
[19] SSD extension for DiskSim simulation environment [Online]. Available http://research.microsoft.com/en-us/downloads/b41019e2-1d2b-44d8-b512-ba35ab814cd4, 2009.
[20] J. Katcher, PostMark: a New File, System Benchmark, October 1997.
[21] Financial OLTP Application I/O, Available http://traces.cs.umass.edu/index.php/Storage/Storage, 2007.
[22] Exchange Trace, SNIA IOTTA Repository, http://iotta.snia.org/traces/130, Apr. 2010 (accessed).
[23] Build Server Trace, SNIA IOTTA Repository, http://iotta.snia.org/traces/158, Apr. 2010 (accessed).
[24] A. Narayanan, A.I.T. Donnelly, A.I.T. Rowstron, Write offloading: practical power management for enterprise storage, Proceedings of the File and Storage Technologies Conference (FAST) 2008, pp. 253–267.
[25] W.D. Norcott, IOzone, http://www.iozone.org.
[26] J. Plank, M. Blaum, Sector-disk (SD) erasure codes for mixed failure modes in RAID Systems, ACM Trans. Stor. 10 (1) (January 2014).
[27] C. Huang, M. Chen, J. Li, Pyramid codes: flexible schemes to trade space for access efficiency in reliable data storage systems, Proc. IEEE Int. Network Computing and Applications (NCA) 2007, pp. 79–86.
[28] O. Khan, R. Burns, J. Plank, C. Huang, In search of I/O-optimal recovery from disk failures, Proc. 3rd Workshop on Hot Topics in Storage and File Systems 2011, pp. 6–66.
[29] F.R. Salmasi, H. Asadi, M. GhasemiGol, Impact of stripe unit size on performance and endurance of SSD-based RAID arrays, Sci. Iran.Trans. D 20 (6) (Dec. 2013) 1978–1998.
[30] J. Plank, A performance evaluation and examination of open-source erasure coding libraries for storage, Proc. 7th USENIX Conf. File and Storage Technologies 2009, pp. 253–265.
[31] L. Xiaoguang, W. Gang, Z. Yu, L. Ang, X. Fang, The performance of erasure codes used in FT-MPI, IEEE Inf. Forum Inf. Technol. Appl. 3 (9) (May 2009) 360–363.
[32] R. Collins, J. Plank, Assessing the performance of erasure codes in the wide-area, IEEE Proc. Int. Conf. Dependable Systems and Networks (DSN), Yokohama, Japan, June. 2005 2005, pp. 182–187.
[33] K.M. Greenan, E.L. Miller, J.J. Wylie, Reliability of flat XOR-based erasure codes on heterogeneous devices, IEEE Proc. Int. Conf. Dependable Systems and Networks (DSN) 2008, pp. 147–156.
[34] D. Yimo, L. Fang, C. Zhiguang, M. Xin, WeLe-RAID: a SSD-based RAID for system endurance and performance, Proc of the 8th IFIP Int. Conf. Network and, Parallel Computing 2011, pp. 248–262.
[35] M. Balakrishnan, A. Kadav, V. Prabhakaran, D. Malkhi, Differential RAID: rethinking RAID for SSD reliability, Proc. 5th European Conf. Computer Systems 2011, pp. 15–26.
[36] K.M. Greenan, D.D. Long, E.L. Miller, T.J.E. Schwarz, A. Wildani, Building flexible, fault-tolerant flash-based storage systems, Proc. 5th Workshop on Hot Topics in Dependability (HotDep), 2009.
[37] S. Im, D. Shin, Flash-Aware RAID Techniques for Dependable and High-Performance Flash Memory SSD, IEEE Trans. Comput. 60 (1) (2011) 80–92.
[38] Y. Lee, S. Jung, Y.H. Song, FRA: a Flash-aware Redundancy Array of flash storage devices, Proc. 7th IEEE/ACM in Int. Conf. Hardware/Software Codesign and System, Synthesis 2009, pp. 163–172.
[39] S. Im, D. Shin, Delayed partial parity scheme for reliable and high-performance flash memory SSD, Proc. IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST) 2010, pp. 1–6.
[40] G. Wu, X. He, B. Eckart, An adaptive write buffer management scheme for flash-based SSDs, ACM Trans. Stor. 8 (1) (Feb. 2012).
[41] T. Xie, J. Koshia, Boosting random write performance for enterprise flash storage systems, Proc. IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST) 2011, pp. 1–10.
[42] Toward I/O efficient Protection Against Silent Data Corruptions in RAID Arrrays, 2014.
[43] L.N. Bairavasundaram, G.R. Goodson, B. Schroeder, A.C. Arpaci-Dusseau, R.H. Arpaci-Dusseau, An analysis of data corruption in the storage stack, Proc. USENIX FASTFeb. 2008.
[44] J.L. Hafner, V. Deenadhayalan, W. Belluomini, K. Rao, Undetected disk errors in RAID arrays, IBM J. Res. Dev. 52 (4/5) (2008) 413–425.
[45] A. Krioukov, L.N. Bairavasundaram, G.R. Goodson, K. Srinivasan, R. Thelen, A.C. Arpaci-Dusseau, R.H. Arpaci-Dusseau, Parity lost and parity regained, Proc. USENIX FASTFeb. 2008.