On Endurance of Erasure Codes in SSD-based Storage Systems

Saeideh Alinezhad Chamazcoti, Seyed Ghassem Miremadi, and Hossein Asadi
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
alinezhad@ce.sharif.edu, miremadi@sharif.edu, asadi@sharif.edu

***Abstract*—The wear-out of flash-based *Solid-State Drives* (SSDs) is a main concern that significantly affects their reliability. One major parameter that accelerates SSD wear-out is the number of write-cycles committed to flash chips. The number of write-cycles in SSD-based disk subsystem is highly dependent on the erasure code implemented in *Redundant Array of Independent Disks* (RAIDs). In this paper, we investigate the impact of erasure codes and the configuration of storage subsystems (i.e., the number of disks participated in the RAID array and stripe unit size) on the endurance of storage systems. The number of write-cycles is considered as a metric to evaluate the endurance of storage system. We evaluate the endurance of four different well-known erasure codes, i.e., Reed-Solomon, EVENODD, RDP, and X-Code, employed in SSD-based RAID systems. In the evaluation, the number of write-cycles is measured with respect to the number of disks, stripe unit size, and request size using trace-driven simulation. The simulation results show that Reed-Solomon provides the lowest number of write-cycles due to the optimal dependency between data and parities in its coding. The results also demonstrate that EVENODD and RDP impose the highest number of write-cycles when using the high number of disks with large stripe unit size. These results recommend designing erasure codes with minimum dependency between data and parities as this minimum dependency provides optimal number of write-cycles.**

Keywords—Endurance; Erasure Code; Data Storage System; Solid State Drive.

I. INTRODUCTION

The use of *Solid State Drives* (SSDs) has become more attractive in data storage systems due to its higher performance and lower power consumption as compared to *Hard Disk Drives* (HDDs) [1]. Storage subsystems are typically protected against faults by two common methods called complete replication [2-3] and data encoding methods [4-9]. 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 methods implemented in *Redundant Array of Independent Disks* (RAID) [10]. These codes are stored in parity disks to recover possible failures in data disks.

Among different faults that may threaten the reliability of storage systems, wear-out is a common failure type in SSD-based storage systems. Wear-out is accelerated by the increasing number of write-cycles. Hence, the endurance of SSDs decreases as the number of write-cycles increases. Endurance is defined as the longevity of a flash media and is a function of the number of write/erase cycles [11]. Due to the relation between endurance and reliability, decreasing the endurance may result in lower reliability as well. In this paper, the terms life-time and endurance are used interchangeably.

The improvement of endurance in SSD-based storage systems has been addressed in [12-16] by handling wear-out problem. In this way, either a specific distribution of parities across disks of a storage subsystem is utilized [12-14] or a hybrid storage subsystem consists of SSDs and HDDs is employed [15-16]. A number of erasure codes for storage systems have been also presented in the literatures [4-9], where reliability and/or performance have been the main concern. To the best of our knowledge, no study has investigated the effect of erasure codes on the endurance of SSD-based storage systems.

This paper investigates the effect of erasure codes on the endurance of SSD-based RAID systems. In the investigation, four widely used erasure codes, i.e., Reed-Solomon, EVENODD, RDP, and X-Code are analyzed in terms of the number of write-cycles using trace-driven simulation. The endurance of these codes is examined with respect to the number of disks, stripe unit size, and the size of requests. We have implemented the four above mentioned codes in a simulation environment. These codes are provided with three input parameters, i.e., the number of disks in a RAID array, the stripe unit size, and the input trace. Based on the analytical analysis and simulation results, the following observations are revealed:

- A lower dependency between data and parities gives lower number of write-cycles, resulting in higher endurance. Reed-Solomon and EVENODD impose the minimum and maximum dependency, respectively.
- Parity distribution among disks is varied for different erasure codes. In Reed-Solomon, both parity disks impose equal number of writes while the diagonal parity disk in EVENODD imposes much higher number of writes than the row parity disk. Even distribution of parities among disks leads to similar rate of wear-out in all disks resulting in longer life and higher reliability in the disk subsystem.
- By increasing the number of disks in a RAID array, the number of writes is increased in all erasure codes except Reed-Solomon. Hence, when an array with greater number of disks is required due to performance requirements, employing Reed-Solomon can offer improved endurance as compared to other erasure codes.
- In all target erasure codes, the number of write-cycles increases with the increment in the stripe unit size.
- In terms of endurance, different erasure codes would be appropriate for variant applications depending on the target application. For example, to reach more reliable storage system, Reed-Solomon is the best choice for all the applications especially in a RAID array with high number of disks. It is not recommended using X-Code for the applications which request size is much larger than the stripe unit size. EVENODD and RDP are suitable for the applications which their request size is close to or less than stripe unit size.

The reminder of this paper is organized as follows. In Section II, a background on SSD characteristics and a brief introduction of four erasure codes, i.e., Reed-Solomon, EVENODD, RDP, and X-Code are presented. Section III compares these codes with respect to the number of disks, stripe unit size, and request size. In Section IV, the simulation results are reported. Section V reviews previous work on

endurance of SSD-based storage systems. Finally, Section VI concludes the paper.

II. BACKGROUND

A. Characteristics of Solid State Drives

In recent years, SSDs have been widely used in many applications due to their benefits such as low access latencies, low power consumption, and high resistance to vibrations and temperature. SSDs typically employ NAND flash memory to store user data. NAND-based SSDs, however, suffer from limitations such as *out-of-place update* and *asymmetric read-write operations*. The characteristics of flash memory do not permit to rewrite a data block unless the corresponding memory block is erased. Erasing before each write operation imposes significant erase cycles to SSDs. On the other hand, NAND flash memory suffers from inherent limited number of erase cycles. After specific number of erase cycles for each block, the block would wear out, and stored data may not be reliable any more (endurance limit).

Some advanced algorithms such as wear-leveling address the limited endurance of SSDs by evenly distributing erase cycles in all blocks. Some other managing methods such as garbage collection may intense the wear-out of blocks. By increasing the write-intensive applications, endurance limit becomes more pronounced, and requires more consideration.

B. Overview of Erasure Codes

Erasure codes 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 used in RAID6 with $n+2$ disk subsystems, which can tolerate concurrent failures of any two data or parity blocks. Several erasure codes have been presented based on RAID6, including Reed-Solomon code [4], EVENODD code [5], RDP code [6], HDP-Code [7], X-Code [8], and P-Code [9]. Among the mentioned erasure codes, we compare and analyze widely used erasure codes such as "*Reed-Solomon*, *EVENODD*, *RDP* and *X-Code*" in terms of the number of write-cycles. 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.

1) REED-SOLOMON

Reed-Solomon code is the most popular and applicable erasure code technique, which is widely being used in communications and storage systems. The main advantage of this code is its scalability to recover up to m data or parity blocks (m greater or equal to 2). This code, however, 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 to widely being used in enterprise applications. The layout of Reed-Solomon encoding is illustrated in Fig. 1.

2) EVENODD

EVENODD code is defined as a $(p-1)\times(p+2)$ column matrix, where p is a prime number. Data and parities are stored corresponding in the p first blocks (columns) and the last two blocks, respectively. This code uses two parity disks, thus can tolerate up to two disk failures. The row and diagonal parity disks are constructed by applying XOR operations over data blocks in row and diagonal, respectively. In this code, an adjusting factor (S) is computed by XORing data blocks in the main diagonal. This factor is applied in computing the diagonal parities. The construction of encoding of this code is illustrated in the Fig. 2.

3) RDP

In *Row-Diagonal Parity* (RDP) code, a $(p-1)\times(p+1)$ matrix is defined where p is a prime number. Data is stored in the *p-1* first blocks, and the two last blocks store the row parity and diagonal parity. RDP is the improved method of EVENODD in the terms of computational complexity as it removes the calculation of the adjusting factor S for diagonal parity and involves row parity in constructing diagonal parity. The RDP layout is illustrated in Fig. 3.

4) X-Code

X-Code is a $(p\times p)$ matrix, where p is also a prime number. In this matrix, data and parity blocks are stored in the first $p-2$ and the last two rows, respectively. Diagonal and anti-diagonal parities are calculated, as shown in Fig. 4. In spite of other horizontal codes, which each disk contains just data or parity, each disk in X-Code comprises both data and parity. Because of this structure, X-Code is called vertical codes. X-Code has an optimal computational complexity due to applying XOR operation for its coding.

III. CHARACTERIZATION OF ERASURE CODES

In this section, we characterize four main erasure codes, i.e. Reed-Solomon [4], EVENODD [5], RDP [6], and X-Codes [8] in terms of the number of write-cycles (including the updated data and parity data in a RAID array). These codes are used in RAID disk subsystems and can ensure to recover up to two disk failures. For the sake of fair comparison, we consider similar storage area for each of these codes and use the same number of disks within an array. We analyze these codes in terms of the number of write-cycles by using trace-driven simulation [17]. Fig. 5 gives a brief understanding about the conceptual terms of this study. As shown in this figure, stripe unit size is the granularity of data distribution over disks. The request size is determined by the trace and it indicates how many stripe units should be updated. Each data block in this figure is labeled by a 2-bit number, which indicates the corresponding row and column (e.g., block 03 belongs to row 0 and column 3). In the following characterization study, we have investigated the impact of request size, the number of disks participated in the RAID array, and the stripe unit size on the number of write-cycles for target erasure codes. In the following subsections, we analyze the impact of each of the above mentioned parameters on the number of write-cycles and report the corresponding observations accordingly.

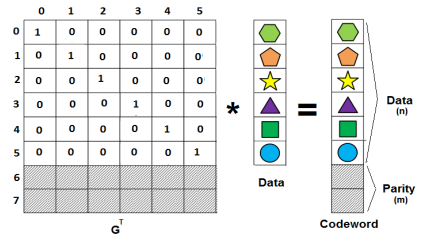

Fig. 1. Reed-Solomon encoding layout.

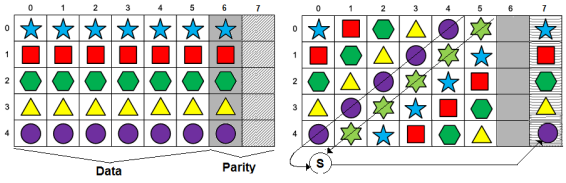

Fig. 2. Horizental and diagonal layout of EVENODD code (p=6[1]).

[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.

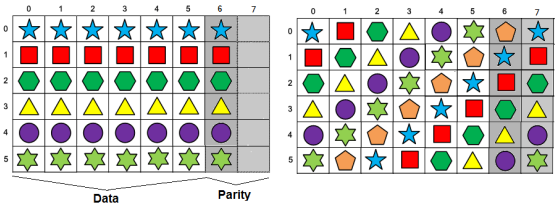

Fig. 3. Horizontal and diagonal layout of RDP code (p=7) [7].

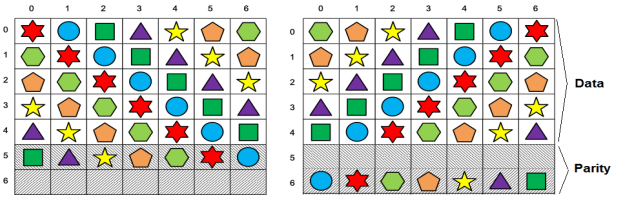

Fig. 4. Diagonal and anti-diagonal parity layout of X-Code (p=7) [7].

A. Request Size

In this subsection, we evaluate the effect of request size on the number of write-cycles. Request size is specified in the trace file and states the number of units that should be updated in the array. We compare four erasure codes using the same system setup (number of disks = 7, stripe unit size = 4K). As the size of requests is mainly determined by a running application, we have used sample traces with fixed request size and counted the number of write-cycles for each sample trace. Since the starting address of each request can affect the number of write-cycles, we investigate the minimum and maximum number of write-cycles for different starting addresses.

Table I reports the number of write-cycles and the starting address of requests for different erasure codes. This table also reports both the minimum and the maximum number of writes for target erasure codes. The starting address will help reader to easily understand and justify the correctness of calculated write-cycles for each case. As an example, the minimum and maximum number of write-cycles required updating one unit (when stripe unit size is equal to request size) has been shown for EVENODD in Fig. 6. As shown in Fig. 6.a, the minimum number of write-cycles for updating one stripe unit occurs when only two parity units are updated (three units update in total). Fig. 6.b demonstrates the maximum number of write-cycles for updating one stripe unit where five parity units are updated (six updated units in total). The maximum number of write-cycles is imposed to the RAID array if one of the data units in the main diagonal of the disk array is updated. In such case, one unit of row parity and all units of diagonal parity should be updated accordingly. This is due to the participation of data units on the main diagonal, in constructing diagonal parity; hence, updating a data unit on a main diagonal will also change the diagonal parity as well. This has been illustrated in Fig. 6.b.

Fig. 7 has reported the maximum number of write-cycles for different request sizes. Although, in all erasure codes, the number of write-cycles is increased by raising the request size, but the rate of increment are various for different erasure codes. For example, in X-Code, by doubling the size of request from 32K to 64K, the number of write-cycles is increased 83% but this increment is 71% for Reed-Solomon. In Reed-Solomon, the adjacent data in one row share the same parities. Hence, if the updated data is located in one row, just two parities for each row should be modified; however, in X-Code, each data in a row has two independent parities. As shown in Fig. 10, EVENODD imposes 100% more write-cycles than X-Code for 4KB requests. Considering 64KB requests, X-Code imposes 57% more write-cycles than EVENODD.

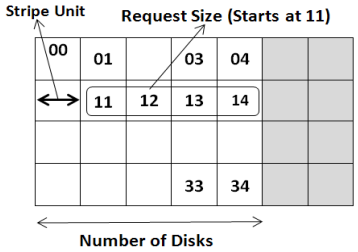

Fig. 5. Definition of conceptual terms used in this study.

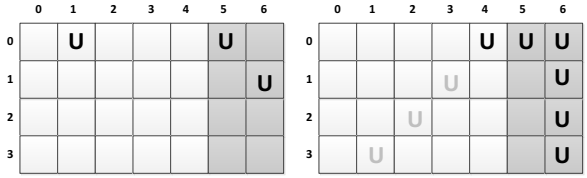

Fig. 6. EVENODD update for one page: a) min_no. of write-cycle, b) max_no. of write-cycle.

TABLE I. MINIMUM AND MAXIMUM NUMBER OF WRITES WITH DIFFERENT REQUEST SIZES IN A RAID6 5+2 ARRAY (PAGE SIZE=4KB)

Request Size (# of pages)		1	4	8	16
EVENODD	Min/ Starting Addr.	*3*/**01**	*9*/**01**	*14*/**01**	*24*/**01**
	Max/ Starting Addr.	*6*/**04**	*10*/**04**	*18*/**31**	*28*/**24**
RDP	Min/ Starting Addr.	*2*/**14**	*9*/**01**	*15*/**00**	*25*/**00**
	Max/ Starting Addr.	*4*/**11**	*11*/**10**	*20*/**43**	*30*/**30**
X-Code	Min/ Starting Addr.	*3*/**01**	*11*/**10**	*21*/**01**	*30*/**01**
	Max/ Starting Addr.	*3*/**01**	*12*/**01**	*24*/**40**	*44*/**36**
Reed-Solomon	Min/ Starting Addr.	*3*/**01**	*6*/**01**	*12*/**01**	*24*/**01**
	Max/ Starting Addr.	*3*/**04**	*8*/**04**	*14*/**04**	*24*/**04**

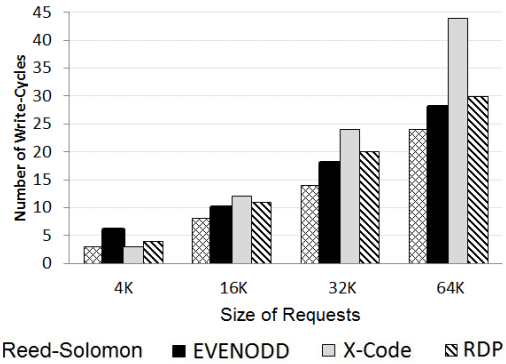

Fig. 7. Maximum number of write-cycles by increasing the request size (Number of disks=5 and stripe unit size=4K)

B. Number of Disks

In this subsection, we analyze the impact of increasing the number of disks in a RAID6 array on the number of write-cycles. The size of updated request is fixed (15KB) and the effect of the number of disks on write-cycles is evaluated for different stripe unit sizes. The results in Fig. 8 demonstrate that by increasing the number of disks, the total number of write-cycles increases in EVENODD, RDP, and X-Code; however, it decreases smoothly or remains almost constant in Reed-Solomon.

In all target erasure codes except Reed-Solomon, the number of disks in each row affects the number of updated parity units. Hence, greater number of disks leads to more parity updates and more write-cycles. In Reed-Solomon, the number of write-cycles is independent of the number of disks, and instead it depends on the number of rows participated in computing parity units.

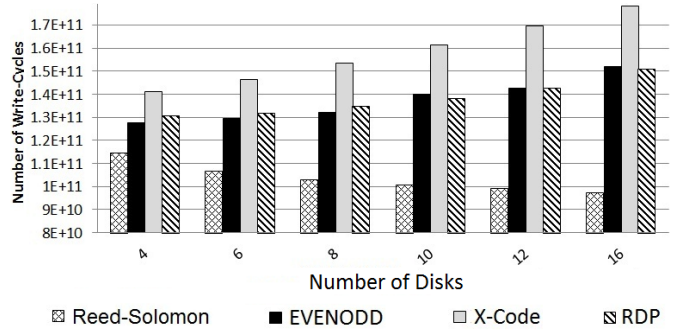

Fig. 8. Number of write-cycles for different number of disks (stripe unit size = 4K, request size=15KB).

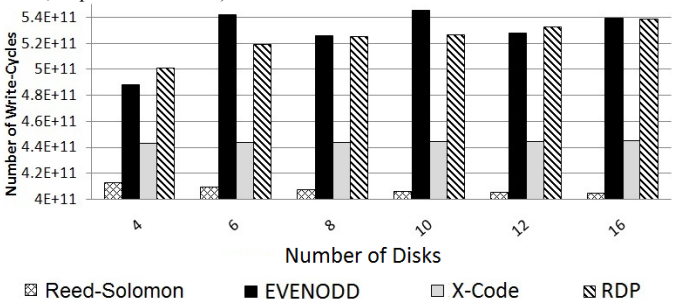

Fig. 9. Number of write-cycles for different number of disks (stripe unit size = 32K, request size=15KB).

The impact of the number of disks on the number of write-cycles with different stripe unit sizes has been also reported in Fig. 8 and Fig. 9. If the stripe unit size is equal or larger than the request size, then only one data stripe unit is updated (this has been shown in Fig. 9 where the request size and stripe unit size are equal to 15K and 32K, respectively). In this case, only two parity units are updated in Reed-Solomon and X-Code, but RDP and EVENODD can impose up to four and six updated units, respectively. In these two erasure codes, i.e., EVENODD and RDP, the location of updated data will determine the number of updated parity units. The alternative behavior of EVENODD in Fig. 9 (where the request size is larger than stripe unit size) is due to dependency of the number of write-cycles on the location of updated data.

C. Stripe Unit Size

In this section, the effect of stripe unit size on the number of write-cycles is investigated. Fig. 10 demonstrates the number of write-cycles for different erasure codes. As shown in this figure, the number of write-cycles increases with the stripe unit size increment. As shown in Fig. 10, X-Code imposes higher number of write-cycles than other erasure codes in smaller stripe unit sizes (4KB). However, considering larger stripe unit sizes (32KB), EVENODD and RDP impose higher number of write-cycles than the others. The Reed-Solomon imposes the least number of write-cycles for both small and large stripe unit sizes as shown in Fig. 10.

D. Advantages and Limitations of Existing Erasure Codes

The main features and limitations of the target erasure codes in terms of computation complexity and storage efficiency have been summarized in Table II. According to this table, Reed-Solomon suffers from computational complexity ($O(n^3)$); however, it imposes the minimum number of write-cycles in almost all configurations (with different parameters). In addition, this code can tolerate more than two disk failures in disk subsystem while the other codes, i.e., EVENODD, RDP, and X-Code, can tolerate up to two disk failures. The order of computational complexity in data encoding and data decoding in all erasure codes except Reed-Solomon is identical.

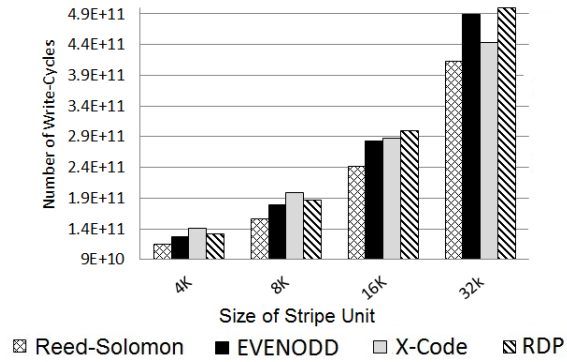

Fig. 10. Number of write-cycles for different stripe unit sizes. (number of disks =4 ,request size = 15KB)

TABLE II. COMPUTATION COMPLEXITY OF ERASURE CODES (5 DATA DISKS + 2 PARITY DISKS)

Code	Computation (n,m=2)			Storage Efficiency ($\frac{data}{data+parity}$)
	Encode	Decode	Update	
Reed-Solomon Code	O(mn)	$O(n^3)$	O(m)	$\frac{n^2}{n\times(n+2)}$
EVENODD Code	$O(n^2)$	$O(n^2)$	O(1)-O(n)	$\frac{n\times(n-1)}{n\times(n+2)}$
X-Code Code	$O(n^2)$	$O(n^2)$	O(1)	$\frac{n\times(n-2)}{n^2}$
RDP Code	$O(n^2)$	$O(n^2)$	O(1)-O(n)	$\frac{n^2}{n\times(n+2)}$

EVENODD and RDP impose the maximum number of write-cycles and offer less lifetime than other erasure codes. However, RDP is proposed as the improvement of EVENODD in order to enhance its performance. This code not only makes no amendment on the endurance, but also makes it worse in some conditions.

X-Code, which is based on vertical codes, has no dedicated disk for parity. Each disk includes both data and parity. This code evenly distributes write-cycles on all disks. Even-distribution is a significant feature of this code in terms of endurance and life-time. Moreover, in terms of storage efficiency, X-Code imposes the least storage penalty in comparison with other erasure codes.

IV. Experimental Result

In order to investigate the effect of erasure codes on the endurance of storage systems, four erasure codes (*Reed-Solomon, EVENODD, RDP,* and *X-Code*) are implemented by C programming language and compiled by gcc 4.3.3 compiler. In the simulation, these codes are provided with three input parameters, the number of disks, the stripe unit size, and the input trace. The size of requests for updated data is specified in the trace. The simulation reports the number of write-cycles for each updated data. Different numbers of disks and the stripe unit size are evaluated in the simulation. The number of write-cycles is calculated after considering the effect of all the updated data on the parity-disks. It is clear that the number of write-cycles in data disks is the same for all the methods except X-Code, which has no special disk for parity. The difference of these methods is in the number of write-cycles on the parity disks (Fig. 11). Fig. 11 illustrates the number of write-cycles in three disks of the array, i.e., the first data disk, parity disk1, and parity disk2. Other observations extracted from Fig. 11 are as follows:

- In Reed-Solomon, EVENODD, and RDP, the contribution of the parity disks in write-cycles is much higher as compared to the write-cycles of first data disks. Therefore, the parity disks will wear out much sooner than data disks.
- The parity disks in Reed-Solomon have the same number of write-cycles, but in EVENODD and RDP, the second

parity disk (which is called diagonal parity in these codes) receives more write-cycles than the other parity disk. The same number of write-cycles in parity disks means the same aging rate. Hence, they will wear out with the same rate, while more write-cycles in one disk lead to fast aging on that disk.

- In X-Code, since there is no dedicated disk for parities, they are distributed in all disks. This kind of distribution results in the same number of write-cycles to all disks. As a result, all disks wear out with almost the same rate.

In the evaluation, we have examined the request size, the stripe unit size, and the number of disks for each configuration. In each experiment, we have fixed two parameters and examined the impact of the third parameter on the number of write-cycles. We have implemented and executed different erasure codes using CAMBRIDGE traces (write request size=15KB, number of write request= 3,857,714) [17]. In each trace, the starting address of each request, the request size, and the request type (read or write) have been declared. To summarize, our observations and conclusions from the results illustrated in Fig. 8 through Fig. 11 are as follows:

- For the fixed stripe unit size, increasing the number of disks increases the total number of write-cycles in all target erasure codes except Reed-Solomon. (Fig. 8).
- For the fixed number of disks, by increasing the size of stripe unit, the number of write-cycles increases in all target erasure codes (Fig. 10).
- In most configurations and in particular for the larger stripe unit size (32K), RDP and EVENODD impose the maximum number of write-cycles, and Reed-Solomon provides the minimum one. The high number of write-cycles in EVENODD and RDP is because of high dependency of parity disks to the data disk in these methods. By each update in data, the corresponding parities should be updated as well. The impact of this dependency becomes much worse when just a few number of update is required. In the case of large stripe unit size with the same request size, larger stripe unit size leads to less number of data or parity updates.
- If the updated data blocks are in the same row, EVENODD, RDP, and Reed-Solomon would have the least number of write-cycles, but X-Code is almost independent of the location of the updated data block. Therefore, by each update in X-Code, the total number of write-cycles increases linearly until the number of updated data blocks becomes equal to the number of data blocks in a row.
- Reed-Solomon code has the minimum number of write-cycles in comparison with other erasure codes. The number of updated parities in Reed-Solomon remains constant as long as the updated data block is within the same row. Therefore, if the updated data is in the same row, this code is the best choice from endurance perspective.
- Although RDP is the improvement of EVENODD in computational complexity but it is as worst as EVENODD in endurance. The results illustrate that both codes impose the maximum number of write-cycles in comparison with other erasure codes.

V. RELATED WORK

The related work can be classified into two groups; first group consists of studies on erasure codes and storage systems; and the second group concentrates on the endurance of storage systems. The first group comprises both SSD and HDD in the structure of storage systems, but the second one just considers SSD-based storage subsystems.

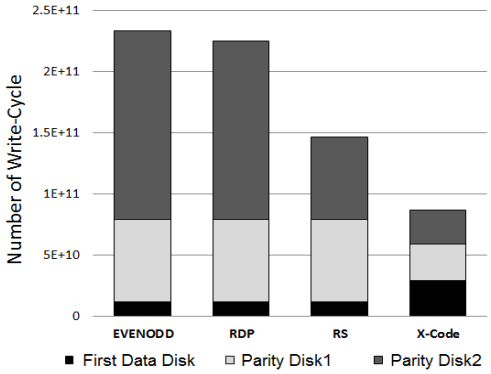

Fig. 11. Number of write-cycles for different erasure code. (stripe unit size=16K, number of disks=8, request size=15KB).

A. Erasure codes

Several studies on the erasure codes have been done in recent decades. These studies include proposing new erasure codes or analyzing and enhancing the performance of these codes. Different approaches for performance enhancement have been proposed such as improving existing codes [6, 18], optimizing I/O load balancing [7], or I/O optimal recovery [19]. Optimizing I/O operations in [7, 19] lead to performance improvement. As I/O operations, especially write operations, are time-consuming, reducing the number of write-cycles leads to improved performance. Besides the performance studies, there are some studies considering the parameters of storage system such as stripe unit size and word size. The investigation of the effect of *the number of words* in *a stripe unit* or *word size* on the performance of disk array has been presented in [20-22].

B. Endurance

Employing SSDs in storage systems associates with the wear-out problem, leading to short life-time and limited endurance. Some efforts have been done to enhance the endurance of SSD-based storage systems by handling the wear-out problems [12-16]. These efforts are mainly concentrated on the structure and configurations of storage subsystems.

Improvement on the structure of storage systems is focused on the RAID structure. In [12], a parity-based redundancy called Diff-RAID is proposed, which makes different ages for each disk in an array of SSDs. In Diff-RAID, parity blocks are distributed irregularly across the arrays in order to avoid the concurrent failures at different disks. After replacing the old devices by the new ones, the parity distribution is reordered on each device so the age differential will be maintained and the life-time of SSDs is improved. It is shown that Diff-RAID improves the life-time of SSDs much better than RAID5. However, variety of aging during the first few replacements is the main issue in Diff-RAID. This problem has been addressed in [13], by proposing a new reliability enhancement method for flash-based storage systems. This method claims seven times faster convergence in comparison with Diff-RAID. The adaption of the mentioned method to the Diff-RAID mechanism can enhance the reliability of flash-based storage systems without decreasing the numbers of device replacement. This method reaches to the same reliability at the first replacement which is achieved after several replacements in the Diff-RAID. Another SSD-based RAID method called WeLe-RAID is proposed in [14]. WeLe-RAID applies a wear-leveling method among flash SSDs to improve the endurance of SSD-based RAID system. This method employs age-driven parity distribution and guarantees wear-leveling among flash SSDs by assigning more parity to younger SSDs and less parity to the

older ones. This method enhances the life span and performance of SSD-based RAID as compared to the conventional RAID with low overhead.

A study investigating the impact of stripe unit size on performance and endurance of SSD-based RAID arrays has been presented in [23]. In this work, it is shown that choosing a 4KB stripe unit size can enhance both performance and endurance. The proposed analysis compares the impact of stripe unit size in SSDs as compared HDD-based RAID arrays. The proposed analysis has been presented for only parity codes employed in RAID4 and RAID5 arrays and has not been discussed for erasure codes employed in RAID6 arrays.

In addition to the methods used in RAID structures to enhance the endurance of storage subsystems, there are some studies on the algorithms and methods applied in SSDs [24-34] including wear-leveling algorithm [27], error correcting code [28], and writing buffer [29-30]. Meanwhile, some studies concentrate on the reliability by modifying the architecture of storage systems and RAID structure [31-32], and some studies focus on the performance of SSD-based storage systems in the terms of write-cycles [33] and response time of parity updates [34].

VI. CONCLUSIONS

In this paper, we analyzed the endurance of the most popular erasure codes, i.e., Reed-Solomon, EVENODD, RDP, and X-Code, by counting the number of write-cycles. In particular, we have examined the impact of stripe unit size, the request size, and the number of disks on the endurance of RAID arrays. The results show that despite computational complexity of Reed-Solomon, it offers the highest endurance level among the studied erasure codes. The results also revealed that lower dependency between data and parities will improve the lifetime of SSD-based RAID arrays.

REFERENCES

[1] M. A. A. Sanvido, F. R. Chu, A. Kulkarni, and R. Selinger, "NAND flash memory and its role in storage architectures," IEEE Magazine, vol. 96, no. 11, pp. 1864-1874, Nov. 2008.

[2] W. Bolosky, J. Douceur, D. Ely, and M. Theimer, "Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs," in *Proceedings of the International Conference on Measurement and Modeling of Computer Systems*, 2000, pp. 34-43.

[3] P. Druschel And A. Rowstron, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility," in *Proceedings of the eighteenth ACM symposium on Operating systems principles*, 2001, pp. 188-201.

[4] I. Reed and G.Solomon, "Polynomial codes over certain finite fields," Journal of the Society for Industrial and Applied Mathematics, pp. 300-304, 1960.

[5] M. Blaum, J. Brady, J. Bruck, and J. Menon, "EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures," IEEE Transactions on Computers, vol. 44, no. 2, pp. 192-202, Feb. 1995.

[6] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-Diagonal Parity for double disk failure correction," in *Proceedings of the the 3rd USENIX Conference on File and Storage Technologies*, 2004, pp. 1-1.

[7] Ch. Wu, X. He, G. Wu, "HDP code: A Horizontal-Diagonal Parity Code to Optimize I/O load balancing in RAID-6," in *Proceedings of the 41st International Conference on Dependable Systems & Networks* (DSN), 2011, pp.209–220.

[8] L. Xu and J. Bruck, "X-Code: MDS array codes with optimal encoding,".IEEE Transactions on Information Theory, vol. 45, no. 1, pp. 272-276, Jan. 1999.

[9] C. Jin, H. Jiang, D. Feng, and L. Tian, "P-Code: a new RAID-6 code with optimal properties,".in *Proceedings of the 23rd International Conference on Supercomputing* (ICS), 2009, pp. 360-369.

[10] D. Patterson, G. Gibson, and R. Katz, "The case for RAID: redundant arrays of inexpensive disks," in Proceedings of the ACM SIGMOD International Conference on Management of Data, 1988, pp. 109-116..

[11] International Technology Roadmap for Semiconductors, ITRS 2007.

[12] M. Balakrishnan, A. Kadav, V. Prabhakaran and D. Malkhi, "Differential RAID: rethinking RAID for SSD reliability," in *Proceedings of the 5th European Conference on Computer Systems*, 2011, pp. 15-26..

[13] I. F. Mir, "A reliability enhancement mechanism for high-assurance MLC flash-based storage systems," in *Proceedings of the IEEE 17th International Conference on Embedded and Real-Time Computing Systems and Applications* (RTCSA), 2011, pp. 190-194.

[14] D. Yimo, L. Fang, C. Zhiguang and M. Xin, "WeLe-RAID: a SSD-based RAID for system endurance and performance," in *Proceedings of the 8th IFIP international Conference on Network and parallel computing* (NPC), 2011, pp. 248-262

[15] B. Mao, H. Jiang, D. Feng, S.Z. Wu, J.X. Chen, L.F. Zeng, L. Tian, "HPDA: a hybrid parity-based disk array for ehnaced performance and reliability," in *Proceedings of the IEEE International Symposium on Parallel & Distributed Processing (IPDPS),* 2010, pp. 1-12.

[16] K. Park, D. H. Lee, Y. Woo, G.Y. Lee., "Reliability and performance enhancement technique for SSD array storage system using RAID mechanism,". in *Proceedings of the 9th International Conference on Communications and Information Technologies* (ISCIT), 2009, pp. 140-145.

[17] A. Narayanan, A. I. T. Donnelly and A. I. T. Rowstron, "Write offloading: practical power management for enterprise storage,".in *Proceedings of the File and Storage Technologies Conference* (FAST), 2008, pp. 253-267.

[18] C. Huang, M. Chen, and J. Li, "Pyramid Codes: flexible schemes to trade space for access efficiency in reliable data storage systems," in *Proceedings of the IEEE International Network Computing and Applications (NCA)*, 2007, pp. 79-86.

[19] O. Khan, R. Burns, J. Plank, and C. Huang, "In search of I/O-optimal recovery from disk failures," in *Proceedings of Hot Storage'11*, *3rd Workshop on Hot Topics in Storage and File Systems*, 2011, pp. 6-66.

[20] M. Li, J. Shu, and W. Zheng, "Strip-based erasure codes with high fault tolerance for storage systems," ACM Tranasaction on Storage, vol. 4, no. 15, pp. 1-22, 2009.

[21] O. Khan, R. Burns, J. Plank, W. Pierce, and Ch. Huang, "Rethinking erasure codes for cloud file systems: Minimizing i/o for recovery and degraded reads," in *Proceedings of 10th USENIX Conference on File and Storage Technologies*, 2012.

[22] J. S. Plank, A. L. Buchsbaum, and T. Bradley Z. Vander, "Minimum density RAID-6 codes," ACM Transaction on Storage, vol. 6, no. 16, pp. 1-22, 2011.

[23] F. Rajaei Salmasi, H. Asadi, and M. GhasemiGol, "Impact of Stripe Unit Size on Performance and Endurance of SSD-Based RAID Arrays", accepted for publication in the Scientia Iranica, 2013.

[24] Intel Corporation, "Understanding the flash translation layer (FTL) specification," AP-684, Dec. 1998.

[25] E. Gal and S. Toledo, "Algorithms and data structures for flash memories," ACM Computing Survey (CSUR), vol. 37, no. 2, pp. 138-163, June 2005.

[26] P. L. Wu, Y. H. Chang, and T. W. Kuo, "A file-system-aware FTL design for flash-memory storage systems," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE),* 2009, pp. 393-398.

[27] S. Sh. Rizvi and T. S. Chung, "AMI: and advanced endurance management technique for flash memory storage systems," *International Arab Journal of Inforamtion Technology*, vol. 8, no. 1, Jan. 2011.

[28] T. H. Chen, Y. Y. Hsiao, Y. T. Hsing, and C.-W. Wu, "An adaptive-rate error correction scheme for NAND flash memory," in *Proceedings of the 27th IEEE VLSI Test Symposium,* 2009, pp.53-58.

[29] Z. Chen, F Liu, Y Du, "Reorder the write sequence by virtual write buffer to extend SSD's lifespan," in *Proceedings of the 8th IFIP International Conference on Network and Parallel Computing*, 2011, pp. 263-276.

[30] G. Wu, B. Eckart, X. He, "BPAC: An Adaptive Write Buffer Management Scheme for Flash-based Solid State Drives," in *Proceedings of the IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST),* 2010, pp. 1-6.

[31] K. M. Greenan, D. D. Long, E. L. Miller, T. J. E. Schwarz, and A.Wildani. "Building flexible, fault-tolerant flash-based storage systems," in *Proceedings of The Fifth Workshop on Hot Topics in Dependability (HotDep)*, 2009.

[32] S. Im, Dongkun Shi, Flash-Aware RAID Techniques for Dependable and High-Performance Flash Memory SSD, *IEEE Transactions on Computers*, vol. 60, Issue.1, pp. 80-92, Jan. 2011.

[33] Y. Lee, S. Jung, Y. H. Song, "FRA: A Flash-aware Redundancy Array of Flash Storage Devices," in *Proceedings of the 7th IEEE/ACM in international conference on Hardware/software codesign and system synthesis*, 2009, pp. 163-172.

[34] S. Im, D. Shin, "Delayed partial parity scheme for reliable and high-performance flash memory SSD," in *Proceedings of IEEE 26th Symposium on Mass Storage Systems and Technologies* (MSST), 2010, pp. 1-6.