ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.11, No.23, 1–9

A new differential RAID for high reliable All Flash Array

Wei Yi[a)], **Hui Xu, Kai Bu, and Nan Li**

Department of Electronic Science and Engineering, National University of Defense Technology, No 109, Deya Road, Changsha City, Hunan 410073, China

a) *yiwei@nudt-esss.com*

Abstract: For high I/O performance, Solid State Drive (SSD)-based all Flash arrays (AFAs) are widely employed to larger scale storage systems. However, the aging problem of SSDs limits the lifetime and reliability of AFA. In order to extend the lifetime of AFA, we designed a capacity-based differential RAID (Redundant Array of Independent Disks) scheme (CDiff-RAID). In the proposed scheme, the array is initialized by a group of SSDs with unequal capacities. The smaller SSDs suffer more erase operations and age quickly. To maintain the age differential, the worn SSDs are continually replaced by new SSDs. Compared with parity-based Diff-RAID, the age differential in our scheme is independent of workload and the time-consuming of reconstruction is greatly reduced. We also evaluated the I/O performance of CDiff-RAID using SSD simulators. The proposed scheme outperforms Diff-RAID scheme in sequential and random traces in most cases.

Keywords: SSD, AFA, reliability, performance

Classification: Storage technology

References

[1] M. Balakrishnan, A. Kadav, V. Prabhakaran and D. Malkhi: ACM Trans. Storage **6** (2010) 4. DOI:10.1145/1807060.1807061

[2] M. Blaum, J. Hafner and S. Hetzler: IEEE Trans. Inf. Theory **59** (2013) 4510. DOI:10.1109/TIT.2013.2252395

[3] J. S. Plank and M. Blaum: ACM Trans. Storage **10** (2014) 4. DOI:10.1145/2560013

[4] J. S. Bucy, J. Schindler, S. W. Schlosser and G. R. Ganger: J. Parallel Data Laboratory (2008) 26.

[5] K. Takeuchi, T. Hatanaka and S. Tanakamaru: IEICE Electron. Express **9** (2012) 779. DOI:10.1587/elex.9.779

[6] K. Zhao, W. Zhao, H. Sun, T. Zhang, X. Zhang and N. Zheng: Fast (2013) 243.

[7] L. M. Grupp, J. D. Davis and S. Swanson: Fast (2012) 2.

[8] M. Li and P. C. Lee: Fast (2014) 147.

[9] N. Jeremic, G. Mühl, R. Haas and J. Richling: SYSTOR (2011) 14.

[10] X. Hu, E. Eleftheriou, A. Busse, I. Iliadis and R. Pletka: SYSTOR (2009) 9.

[11] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo and S. Zhang: ICS (2011) 96.

[12] Iometer (2003) http://www.iometer.org/.

DOI: 10.1587/elex.11.20141007
Received October 21, 2014
Accepted November 10, 2014
Publicized November 21, 2014
Copyedited December 10, 2014

1 Introduction

In recent years, Flash-based Solid State Drives (SSDs) prevail in enterprise storage systems owing to its high performance and low power consumption. However, SSDs are severely hamstrung by low endurance limits. NAND Flash chips (NANDs), as the basic components of SSD, are subject to bit errors. The bit error rate (BER) is growing with the aging of SSD. Although there are error correcting codes (ECCs) in SSD to cope with bit errors, the probability of unrecoverable sector errors (USEs) is increasing.

For both high performance and reliability, RAID (Redundant Array of Independent Disks) technology is commonly used to construct SSD-based All Flash Array (AFA). The RAID schemes with redundancy have the ability to cope with disk failures and USEs, such as RAID level 5 (RAID-5) and RAID level 6 (RAID-6). However, the two RAID schemes are unsuitable for AFA. All drives in them suffer almost the same number of writes and age simultaneously. The reliability at the aged stage is very poor.

Balakrishnan et al. [1] proposed a Diff-RAID scheme, a parity-based redundancy solution that creates an age differential in an array of SSDs by distributing parity blocks unevenly. The drive which holds more parity blocks ages fast. When old devices are replaced by new ones, Diff-RAID reshuffles the parity distribution to maintain the age differential. In this way, the fluctuations in reliability are diminished. However, the drive holding more parity blocks restricts the performance of array. The age differential is highly dependent on the workload running on the array. And the reconstructing process is complicated and time-consuming.

In this paper, we proposed a capacity-based Diff-RAID scheme to improve I/O performance and availability. In order to create an age differential, we use a group of SSDs with unequal capacities to initialize the array. The wear-leveling algorithm in SSD keeps all NANDs are programmed evenly. The NANDs in small SSDs suffer more erases and age fast. In this way, the capacity differential is transformed into an age differential. We can combine the popular RAID-5 or RAID-6 schemes with our scheme to maintain the age differential everlastingly. Compared with parity-based Diff-RAID scheme, our scheme has little impact on I/O performance and the age differential is independent of workload. When the oldest drive is replaced by a new drive, we just need to copy the valid data and parity to the new drive.

The reminder of this paper is organized as follows. Section 2 reviews relevant researches on the reliability problem of AFA and corresponding reinforcements. Section 3 presents our proposed capacity-based Diff-RAID. In section 4, we evaluate the reliability, performance and availability of the proposed scheme by experiments. At last, section 5 gives the conclusions and future directions of research.

2 Related research

In SSD-based AFA, both drive failures and USEs are likely to lead to data loss. Because there is no mechanical moving part in SSD, drive failures are relatively rare. The common failure type is the latent USE in which sectors on SSD are

DOI: 10.1587/elex.11.20141007
Received October 21, 2014
Accepted November 10, 2014
Publicized November 21, 2014
Copyedited December 10, 2014

unrecoverable from ECCs. To reduce the cost per gigabyte, manufacturers continue to improve NANDs' density by scaling down the feature size. The endurance of NANDs is decreased further [7]. To improve reliability, strong ECCs are recommended to SSD [5, 6]. However, they can only postpone the risk of data loss.

RAID technologies at disk-level have the ability to recover sector errors. They are either insufficient like RAID-5 or over provided like RAID-6 [2, 3, 8]. Design a series of Sector-Disk codes to enhance the AFA's ability to cope with USEs. However, those schemes just mitigate the threat brought from USEs. When all SSDs in AFA become aged synchronously, the security of the stored data is no longer guaranteed [9].

Balakrishnan et al. [1] create a age differential among all SSDs by distributing parity unevenly. The parity block suffers more updates in random trace. So the drive which holds more parity blocks ages quickly. However, the age differential is highly dependent on the workload running on it. Even if random writes are dominant in a workload, the drive with more parity blocks becomes a performance bottleneck for AFA. When the oldest SSD is replaced by a new SSD, the parity distribution needs to be reshuffled. But the reshuffling operation is time-consuming and complicated.

3 CDiff-RAID

In Diff-RAID scheme, the aged SSDs are more likely to produce USEs. The young SSDs with seldom USEs are able to protect the old SSDs. However, the method may lead to decrease in performance and availability. In this section, we introduce a new way to build a differential RAID by exploiting the characteristics of SSD.

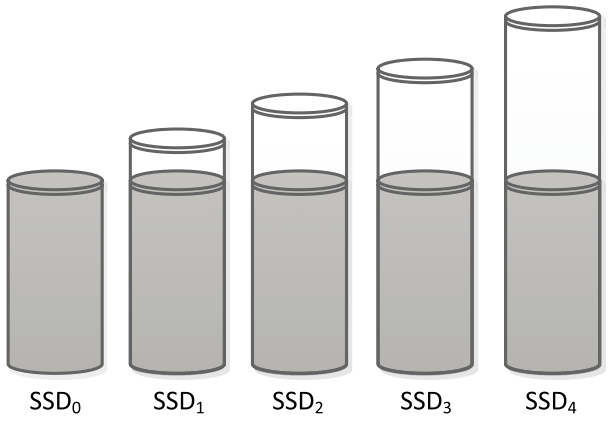

Fig. 1. The architecture of Capacity-based Diff-RAID.

3.1 Capacity-based Diff-RAID

In order to produce a differential in age, we design a Capacity-based Diff-RAID (CDiff-RAID) scheme to build AFA, in which all SDDs have different capacities. As shown in Fig. 1, the capacity of SSD_0 to SSD_4 forms an arithmetic progression. Each SSD provides the same logical storage space (LSS) for the RAID controller. If the array adopts a RAID-5 scheme, all SSDs nearly suffer the same number of write

DOI: 10.1587/elex.11.20141007
Received October 21, 2014
Accepted November 10, 2014
Publicized November 21, 2014
Copyedited December 10, 2014

operations. Though the physical space of all drives is unequal, the wear-leveling algorithm in SSD keeps all physical blocks are programmed and erased (P/E) evenly. As a result, the smallest SSD_0 suffers more P/E cycles and ages faster than other SSDs. Thus, the capacity differential is transformed into an age differential.

To exemplify our scheme, we assume that an array consists of five drives and the capacities are ranging from 256 GB to 512 GB. The capacity difference is 64 GB and the rated P/E cycles of all drives are equal to e. We define two parameters: equivalent spare space (ESS) and equivalent used space (EUS) to indicate the lifetime difference. ESS means the equivalent space has not been programmed and erased. EUS denotes the equivalent space wears off. The ESS and EUS of SSD_i are calculated by:

$$ESS_i = C_i \times \frac{e - e_{ei}}{e} \tag{1}$$

$$EUS_i = C_i \times \frac{e_{ei}}{e} \tag{2}$$

where e_{ei} is the number of P/E cycles executed on SSD_i and C_i is the capacity of SSD_i. Table I shows the ESS distribution of the array after three replacements. Each column represents the ESS of one drive. Each row indicates the ESS of the existing drives. At the beginning, the e_{ei} is 0 and ESS is equal to the initial capacity. When SSD_0 is used up, the EUSs of all drives computed from formula 2 is 256 GB. We calculate ESSs from formula 1 and find the age differential between all SSDs is equal to 64 GB. To maintain the disparity, 320 GB is provided into SSD_5 to replace SSD_0. The age differential and the total ESS are stabilized after the first replacement. We only need to insert 320 GB drives to substitute the worn drives in the next replacements. The new added drives are bold and appended at end of each row. As shown in Table I, we replace SSD_0 by SSD_5, SSD_1 by SSD_6, SSD_2 by SSD_7 in the 1^{st}, 2^{nd} and 3^{rd} replacement respectively.

Table I. ESS at replacing moments (GB).

Stage of Array	ESS of SSD_i								**Total ESS**
	0	1	2	3	4	5	6	7	
Initial stage	256	320	384	448	512				**1920**
1^{st} replacement	0	64	128	192	256	**320**			**960**
2^{nd} replacement		0	64	128	192	256	**320**		**960**
3^{rd} replacement			0	64	128	192	256	**320**	**960**
⋮				⋮	⋮	⋮	⋮	⋮	⋮

3.2 Analysis of age distribution

In fact, subsequent drives for replacements may tolerate less P/E cycles. In our CDiff-RAID scheme, the rated P/E cycles of SSDs in the array can be different. To achieve a steady age distribution, the SSD with small endurance has to provide more capacity. We use available storage space to denote the lifetime (l_i) of SSD_i, which is computed by multiplying the drive's capacity (c_i) and its rated P/E cycles (e_i) together. However, the write amplification factor (WAF) of drives which is

DOI: 10.1587/elex.11.20141007
Received October 21, 2014
Accepted November 10, 2014
Publicized November 21, 2014
Copyedited December 10, 2014

closely related to over-provision space varies with their capacities. It must be taken in account in the calculation of the lifetime. We use A_f and S_f defined in [10] to denote the WAF and spare factor. Then, the revised lifetime of SSD_i is given by:

$$l_i = \frac{C_i \times e_i}{1 + A_{fi}} \tag{3}$$

where the A_{fi} denotes the WAF of SSD_i.

To produce an equidifferent age differential, the lifetime of all drives needs to form an arithmetic progression. Suppose that an array consists of n drives, the lifetime of the first drive is equal to l_0 and the lifetime difference is Δl. Then, the lifetime distribution of the array is: $(l_0, l_0 + \Delta l, \ldots, l_0 + i \cdot \Delta l, \ldots, l_0 + (n-1) \cdot \Delta l)$. When the first drive is used up, a drive whose lifetime is equal to $n \cdot \Delta l$ is inserted for subsequent replacements. The lifetime of all drives in the array is equal to $n \cdot \Delta l$ after n replacements.

The initial capacity distribution of the array must be recalculated after taking the WAF into account. To exemplify the calculation process, we adopt the Fixed model of Fig. 4 in [10]. Suppose that A_{fi} is a function of S_{fi} ($A_{fi} = f(S_{fi})$), where S_{fi} equals $1 - LSS_i/C_i$ (LSS_i is the Logical Storage Space of the i^{th} SSD). Then, we can establish an equation to obtain the initial capacities of all drives. It's given by:

$$\frac{C_i \times e_i}{1 + f\left(1 - \frac{LSS_i}{C_i}\right)} = l_0 + i \cdot \Delta l \quad (i = 0, 1, \ldots, n-1) \tag{4}$$

Table II. Initial capacity distribution (GB) after considering WAF

Provided LSS (GB)	Physical Capacity (GB)				
	SSD_0	SSD_1	SSD_2	SSD_3	SSD_4
640 GB	281.7	320.0	362.9	409.4	458.3
800 GB	293.3	320.0	345.2	374.5	406.3
1024 GB	305.7	320.0	333.6	347.0	360.2

For example, we use the equation 4 to revise the initial capacity distribution in Table I. Suppose that all drives have the same rated P/E cycles and the capacity of drives for replacements is equal to SSD_1's capacity (320 GB). As shown in Table II, the initial capacities of all drivers are computed with different LSS configurations. The larger the LSS that the array provides, the smaller the capacity differential is. As smaller drivers have large WAFs and age fast, they have to provide more storage space to compensate write penalty. So the capacity differential increases progressively for a given LSS configuration.

To illustrate the alteration in aging differential after considering the WAF, we use the initial capacity (293.3 320.0 345.2 374.5 406.3) in Table II to recalculate the ESS distribution. As shown in Fig. 2, we draw the ESS of all drivers in the array as a function of the amount of write data. Every turning point in the figure indicates a drive replacement. Each curve represents an ESS distribution with a different initial capacity configuration. And the slope of the curves denotes the aging speed of each SSD. Although the aging speeds of all drives are different at

DOI: 10.1587/elex.11.20141007
Received October 21, 2014
Accepted November 10, 2014
Publicized November 21, 2014
Copyedited December 10, 2014

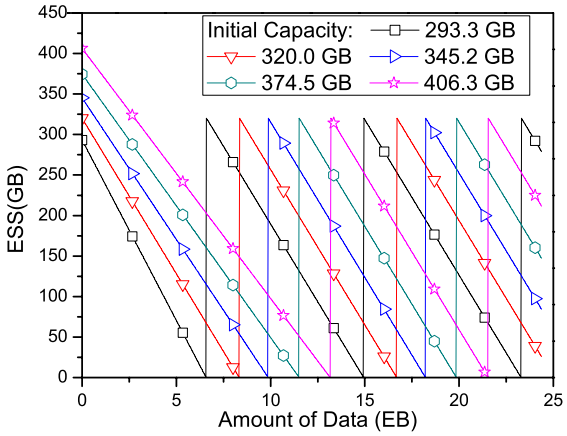

Fig. 2. ESS distribution after considering WAF.

the beginning on account of the varied WAFs, but they are equal after five replacements. Then, the disparity in ESS is stabilized everlastingly.

3.3 CDiff-RAID deployment and reconstruction

In order to generate a differential in lifetime, we have to produce a set of SSDs with different capacities. However, some capacities in Table II are unavailable. In general, Flash chips have fixed size, such as 8 GB, 16 GB, 32 GB etc. We provide a few more storage space to make the size of each drive as a multiplication of 8. The practical initial capacity distributions is shown in Table III.

Table III. Practical initial capacity distribution (GB) after considering WAF

Provided LSS (GB)	Physical Capacity (GB)				
	SSD_0	SSD_1	SSD_2	SSD_3	SSD_4
640 GB	288	320	368	416	464
800 GB	304	320	352	376	408
1024 GB	312	320	336	352	368

When the oldest SSD is replaced by a new one, the RAID controller needs to reconstruct the array. In the proposed scheme, the oldest SSD is aged but the data in it is still valid. We just need to copy the valid data and parity to the new SSD. In order to keep the availability during the reconstructing process, the copy operations are carried out when the array is idle. New write and update operations to the oldest SSD are redirected to the new SSD. After the reconstruction is finished, the new SSD plays the role of the replaced SSD.

4 Evaluation

4.1 Reliability

We adopted the same data and method in [1] to calculate the data loss probability of RAID-5, Diff-RAID and CDiff-RAID as a function of the amount of writes. In order to keep the aging distribution in Diff-RAID scheme, suppose that the adopted

DOI: 10.1587/elex.11.20141007
Received October 21, 2014
Accepted November 10, 2014
Publicized November 21, 2014
Copyedited December 10, 2014

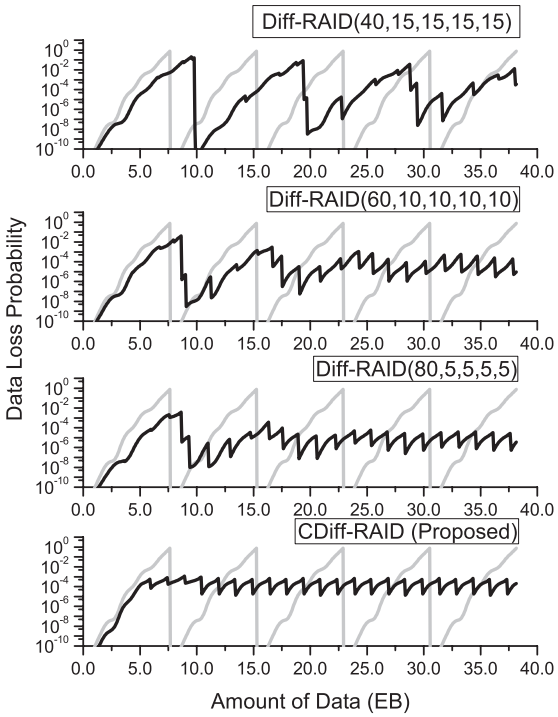

Fig. 3. Reliability distribution in different replacing schemes.

workload only consists of small random writes. The CDiff-RAID employs the initial capacity distribution (312, 320, 336, 352, 368) in subsequent tests. In RAID-5 and Diff-RAID schemes, the capacity of all SSDs in array is 320 GB. Since the reliability of Diff-RAID depends on parity assignments, we test Diff-RAID with three parity distributions: (40, 15, 15, 15, 15), (60, 10, 10, 10, 10), (80, 5, 5, 5, 5).

As shown in Fig. 3, the grayed curve with big spikes denotes RAID-5 scheme with full replacements. Both CDiff-RAID and Diff-RAID scheme are able to cut off the peaks in RAID-5 scheme. The reliability distribution of CDiff-RAID is stationary, except four small spikes at the beginning. However, the spike values in Diff-RAID schemes are inversely proportional to the ratio of parity in the first drive. If the age differential in Diff-RAID is diminished, the data loss probability will climb up. So the reliability of Diff-RAID scheme relies mostly on the workload running on it. The proposed scheme has no restriction on trace. The distribution of reliability is fixed after capacity differential is established.

4.2 I/O performance

4.2.1 Experiments setup

In order to evaluate the I/O performance, we implemented a miniaturized AFA based on DiskSim [4] and SSDsim [11]. The SSD simulator adopts the default parameters as [11]. To get different capacities for CDiff-RAID, we adjust the chips in SSD simulator. We choose RAID-5 scheme and Diff-RAID scheme with parity distributions (60, 10, 10, 10, 10), (80, 5, 5, 5, 5) for comparison. The capacity of drives in RAID-5 and Diff-RAID is 20 GB. The miniaturized initial capacity distribution of CDiff-RAID in Fig. 3 is (19.5, 20, 21, 22, 23). For all schemes, the pages of Flash memory act as the stripe units. The stripe size is 10 KB, containing a 2 KB parity unit.

DOI: 10.1587/elex.11.20141007
Received October 21, 2014
Accepted November 10, 2014
Publicized November 21, 2014
Copyedited December 10, 2014

4.2.2 Comparison of I/O performance

We tested the sequential and random I/O performance of RAID-5, Diff-RAID and CDiff-RAID while varying the request size from 0.5 KB to 64 KB. The traces for testing are all collecting from Iometer [12]. We consider every read and write operation as an independent request. To get standalone I/O performance, the cache size in RAID controller and SSD controller is set to zero.

As shown in Fig. 4(a) and (b), the read performance of CDiff-RAID and RAID-5 is very close. They both outperform Diff-RAID scheme. If the first drive undertakes more parity, other drives will bear more read operations. So the performance of Diff-RAID (80, 5, 5, 5, 5) is worse than Diff-RAID (60, 10, 10, 10, 10). However, the Diff-RAID scheme has to arrange more parity in the first drive to get high reliability.

Fig. 4(c) and (d) show the write performance of all schemes. As the smallest drive in CDiff-RAID drives down the total performance, the performance of CDiff-RAID scheme is poorer than the RAID-5 scheme. But it also outperforms the Diff-RAID scheme when the request size is small. Because each small request is deemed as a standalone request, it will induce a mass of parity updating operations in sequential trace. Therefore, all schemes perform badly at the beginning in sequential traces. When the request size is getting larger, data is written to array stripe by stripe. Since the number of parity updates is decreased, the write performance of all schemes converges. However, full-stripe writes make no contribution to the age differential for Diff-RAID scheme. Generally, the RAID controllers employ a large cache to reduce overwrites. They do their most to extract full-stripe writes from workload to extend the life of SSD. It contradicts with the age differential strategy in Diff-RAID scheme. Fortunately, the proposed scheme can cooperate friendly with those optimization strategies in RAID controller.

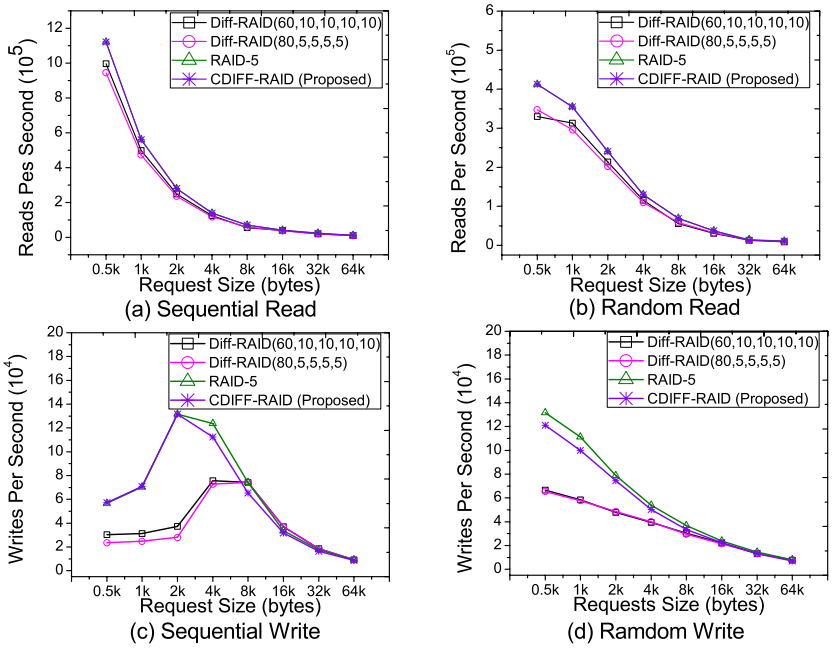

Fig. 4. I/O performance while varying request size.

DOI: 10.1587/elex.11.20141007
Received October 21, 2014
Accepted November 10, 2014
Publicized November 21, 2014
Copyedited December 10, 2014

4.3 Reconstruction

In RAID schemes, the reconstruction complexity is crucial to the availability of array. In differential RAID schemes, we need to continually replace the aged disks. The Diff-RAID scheme in [1] proposed two replacing methods: Naive and Minimal. In the Naive scheme, the RAID controller simply shifts the logical order of the devices to left after inserting the new device. But it has to copy the valid data and parity in the array to new logical locations. To save reconstruction time, the Minimal scheme changes the mapping function in RAID controller and only copies the surplus parity to the new sacrificial drive. Because the RAID controller has to keep mapping tables for the data and parity, the complexity of mapping algorithm will increase iteratively.

Table IV. The replacement overhead quantified by copies.

Schemes	Copies
Naive	5
Minimal (60,10,10,10,10)	1.5
Minimal (80,5,5,5,5)	1.75
CDiff-RAID	1

In comparison, the proposed scheme just copies the valid data and parity in the aged drive to the new drive. It doesn't need any modification in RAID controller. In this section, we only use the number of copies to measure the drive replacing overhead. If we quantify a copy of a drive as 1, the replacing overhead of all schemes can be deduced from it. We list the copy cost of all schemes in Table IV. The proposed scheme outperforms the Diff-RAID scheme even regardless of the overhead in RAID controller.

5 Conclusion

CDiff-RAID introduces a new way to construct the age differential for AFA. The proposed scheme exploits the characteristic that the NANDs in SSD are program and erased evenly even if the provided logical space is smaller than the physical space. It can cooperate with any RAID scheme which distributes parity evenly among all drives. We only need to initialize the array with a set of SSDs in different lifetime distribution. The modifications in RAID controller are negligible. The age differential in the proposed scheme is fixed after the capacity distribution is established. Compared with the parity-based Diff-RAID scheme, the I/O performance of CDiff-RAID is better in most cases and the time-consuming of reconstruction is greatly reduced. In future works, we plan to build a real AFA system to evaluate the performance of the proposed scheme in different RAID schemes.

DOI: 10.1587/elex.11.20141007
Received October 21, 2014
Accepted November 10, 2014
Publicized November 21, 2014
Copyedited December 10, 2014