MC-RAIS: Multi-chunk Redundant Array of Independent SSDs with Improved Performance

Suzhen Wu[1], Weijian Yang[1], Bo Mao[2(✉)], and Yanping Lin[1]

[1] Computer Science Department, Xiamen University, Xiamen 361005, China
[2] Software School of Xiamen University, Xiamen 361005, China
{suzhen,maobo}@xmu.edu.cn

Abstract. Big Data Analytics is a big challenge for the performance of the computing and storage systems. With the rapid development of multi-core and GPU processors, the performance of HDD-based storage system becomes much more serious. The flash-based Solid State Disks (SSDs) have become an emerging alternative to HDDs and received great attentions from both academia and industry. However, a single SSD cannot satisfy the capacity, performance and reliability requirements of a modern storage system supporting increasingly demanding data-intensive computing and applications. Redundant Array of Independent SSDs (RAIS) is an effective way to build high-performance, high-reliability, and high-capacity SSD-based storage systems. In RAIS, the chunk size is an important parameter that affects the system performance. However, the existing studies are mainly focused on the efficiency of chunk size of RAID. Because of the different performance characteristics between HDDs and SSDs, the results of these studies could not be applied to the RAIS. In this paper, we first conducted extensive experiments on the efficiency of chunk size on the RAIS performance. Based on the experimental results, we proposed a Multi-Chunk RAIS (short for MC-RAIS) to improve the performance of the SSD-based storage systems. Evaluation results show that MC-RAIS outperforms the existing fix-chunk-size SSD-based disk arrays in the I/O performance measure by more than 50 %.

Keywords: Solid state drive · Redundant array of independent SSDs · Chunk size · Garbage collection · Performance evaluation

1 Introduction

Hard Disk Drives (HDDs) have become the performance wall of storage systems. Recently, the flash-based Solid State Disks (SSDs) have become an emerging alternative to HDDs and received great attentions from both academia and industry [1,6,8,19]. Different from HDDs, SSDs are based on semiconductor chips and have no mechanical parts. SSDs can provide many benefits, such as low power consumption, high robustness to vibrations and temperature, and

G. Wang et al. (Eds.): ICA3PP 2015, Part IV, LNCS 9531, pp. 18–32, 2015.
DOI: 10.1007/978-3-319-27140-8_2

most importantly, high small-random-read performance. Unfortunately, SSDs also have some disadvantages, such as high cost, erase-before-write problem, low small-random-write performance and limited lifetime [1].

Besides the deployment of SSDs on desktop computers, some studies also use SSDs in the high-performance computing and enterprise environments [5]. In these environments, a single SSD cannot satisfy the performance, capacity and reliability requirements. Thus, applying the RAID (Redundant Array of Independent Disks) technique [13,17] to SSDs is necessary and likely promising to build large-scale high performance and highly reliable SSD-based storage systems [2,14,20,21]. In this paper, Redundant Array of Independent SSDs is short for *RAIS*. The different levels of RAIS are also short for *RAIS0*, *RAIS5* and so on.

By distributing user data across multiple disks in an array, RAID offers high performance, high reliability and high capacity. The chunk size is an important parameter that defines the granularity of data distribution in an array and has been traditionally determined based on characteristics of HDDs to balance throughput and response time [7]. Due to the characteristics of HDDs, the suggested chunk size by the enterprise data storage systems vendors such as IBM, HP, and EMC varies between 16 KB and 1 MB. The suggested chunk sizes can be possibly far from the optimal configuration for SSD-based disk arrays with respect to the I/O throughput and response time. Additionally, the conventional chunk size used for RAID should be revised due to the limited endurance of SSDs. To the best of our knowledge, the analysis and performance studies on the optimization of the chunk size of RAIS are missing in the previous work.

In this paper, we first conducted extensive experiments on the efficiency of the chunk size on the RAIS performance. The evaluation results show that the optimal chunk size is different for read-intensive workloads and write-intensive workloads. Based on the observations, we proposed a Multi-Chunk RAIS (short for MC-RAIS) to improve the performance of the SSD-based storage systems by exploiting the workload characteristics. To evaluate the efficiency of our proposed MC-RAIS scheme, we have implemented a prototype of MC-RAIS by integrating it into an open-source SSD simulator developed by Microsoft Research [1] and conducted extensive experiments to evaluate the performance of the MC-RAIS scheme by a wide variety of enterprise realistic workloads. The experimental results show that the MC-RAIS scheme outperforms the fix-chunk-size RAIS schemes by more than 50 %. Moreover, the MC-RAIS scheme also significantly reduces the performance variability simultaneously.

More specifically, this paper makes the following contributions:

(1) To the best of our knowledge, MC-RAIS is the first study to evaluate the impact of the chunk size on the performance of RAIS system.
(2) MC-RAIS fully exploits the parallelism characteristics of SSDs in RAIS to improve the RAIS performance and alleviate the GC impact on the performance variability.
(3) We evaluate the performance of MC-RAIS in a RAIS simulator developed by Microsoft Research [1] and Carnegie Mellon University (CMU) [4] driven by a wide spectrum of workloads. The experimental results show that the

MC-RAIS scheme outperforms the fix-chunk schemes in terms of average user response times.

The rest of this paper is organized as follows. Background and motivation are presented in Sect. 2. We describe the design detail of the MC-RAIS scheme in Sect. 3. The performance evaluation is presented in Sect. 4. The related work is presented in Sect. 5. We conclude this paper and point out the directions for the future research in Sect. 6.

2 Background and Motivation

In this section, we first describe the key characteristics of flash-based SSDs compared with magnetic HDDs. Then we elaborate how the configuration of the chunk size affects the RAIS performance. Based on these observations, we provide the motivation for our new multi-chunk size optimization scheme for RAIS.

2.1 Characteristics of Flash-based SSD

Similar to HDDs, data in SSDs is persistent when the power supply is turned off. However, unlike mechanical HDDs, flash-based SSDs are made of silicon memory chips and do not have moving parts (*i.e.*, mechanical positioning parts). SSDs can provide many benefits, such as low power consumption, high robustness to vibrations and temperature. Figure 1 illustrates a logical overview of a typical SSD with n independent channels. Each channel is shared by multiple flash chips. SSDs are inherently highly parallelized architectures, comprising different units, including page, block, plane, channel and package. The different constituent operational units can operate in parallel, thus providing the potential to achieve better performance.

Besides the advantages of high parallelism and high energy efficiency, flash-based SSDs have two main characteristics compared with HDDs, as follows.

First, the current generation of SSDs suffers from the poor performance of small random writes. The reason is that in the flash-based SSDs, each block of size 64–128 KB must be erased before any page in it can be re-written, which is known as the characteristic feature of "erase-before-write". Seriously, an erase operation typically takes several milliseconds that is one or two order of magnitude higher than the completion time of a read operation.

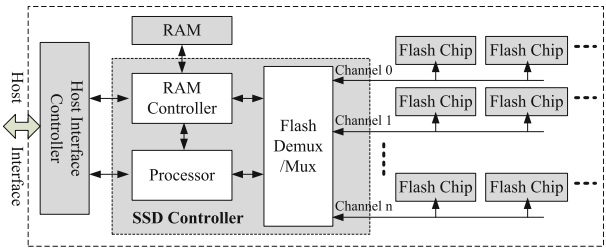


Fig. 1. Internal overview of a typical SSD.

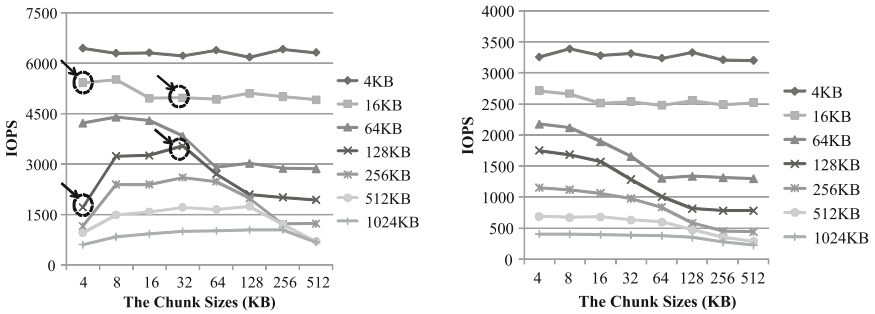
Second, the flash wear-out problem caused by a large number of repeated write-erase operations affects the reliability of SSDs. Generally, the expected number of erasures per block is 100,000 for the single level cell (short for SLC) NAND flash memory while the expected number is reduced to 10,000 for the multi level cell (short for MLC) NAND flash memory.

These two above limitations of SSDs must be taken into consideration when designing SSD-based storage systems, especially SSD-based disk arrays.

2.2 Chunk Size

The chunk size of RAIS is an important parameter that affects the RAIS performance. To investigate the effect of different chunk sizes on the performance of SSD-based disk arrays, we conduct experiments on a RAIS5 consisting of 4 Intel DC S3700 200GB SSDs, driven by the fio benchmark.

Figure 2 shows the performance of random accesses on the RAIS5 with different chunk sizes. From Fig. 2(a), we can see that the performance is increased with the increasing chunk size of RAIS. The experimental results also shown that when the request size is 3 times of the chunk size, the read performance is the best. The reason is that the access is the full stripe read and all SSDs in the RAIS5 are involved in one request. Therefore, the access parallelism among the SSDs is fully exploited so that the RAIS performance is improved accordingly. In contrast, as shown in Fig. 2(b), when the chunk size is the smallest, the write performance is the best. The reason is that with a smaller chunk size, more write requests could be performed in all or most SSDs in the RAIS5 by exploiting the access parallelism and avoid the small-write penalty problem.



(a) The random read to RAIS5 (b) The random write to RAIS5

Fig. 2. The performance of random accesses on a RAIS5 with different chunk sizes.

Based on the evaluation results, we can see that the optimal chunk sizes of SSD-based disk array in different workload conditions are diverse. To provide the best performance of RAIS, the workload characteristics must be fully exploited to guide the data layout, *i.e.*, distributing the data in different zones with different chunk sizes. These important observations motivate us to propose a multi-chunk

RAIS scheme to improve the performance of SSD-based disk arrays by exploiting the workload characteristics.

3 MC-RAIS

In this section, we first present the system overview of MC-RAIS, followed by a description of the request processing workflow and the data consistency in MC-RAIS.

3.1 System Overview of MC-RAIS

Figure 3 shows the system overview of our proposed MC-RAIS. As shown in Fig. 3, MC-RAIS interacts with the *RAIS Functional* module and can be incorporated into any existing RAIS5 schemes, including hardware and software RAIS systems. In general, MC-RAIS consists of two important functional modules added into the RAIS controller in the existing storage system: Workload Monitor and Request Redirector. The *Workload Monitor* module is responsible for identifying the user I/O requests based on the access history. The *Request Redirector* module is responsible for issuing the user I/O requests to the corresponding zones based on results of the Workload Monitor module.

In Fig. 3, the SSD-based disk array is divided into three data zones with different chunk sizes, 4 KB, 8 KB and 32 KB. The 4 KB zone is designed to store

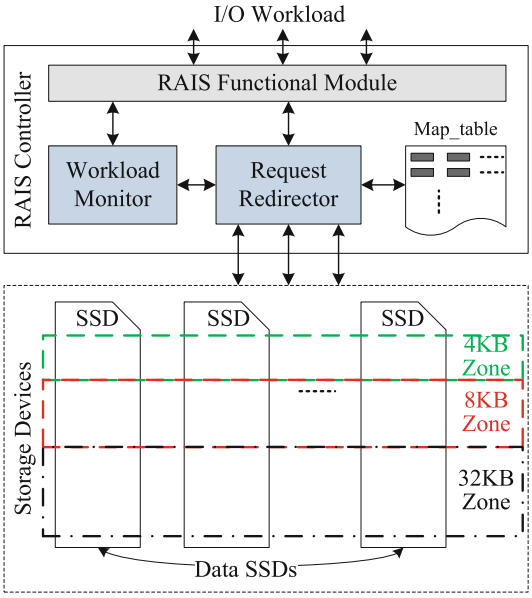


Fig. 3. System overview of MC-RAIS consisting of three data zones with different chunk sizes.

write-intensive data blocks (write-intensive zone), the 32 KB is designed to store the read-intensive data blocks (read-intensive zone) and the 8KB is designed for mixed data blocks. Moreover, the data blocks are classified into read-intensive, write-intensive and mixed types based on the access history. When a data block is accessed, its access frequency will be increased to track the popularity of the data block, which is a simple way to track the data popularity. However, other classic policies are also applicable in MC-RAIS to further track the data popularity to improve the efficiency of MC-RAIS.

In addition to the above two functional modules, MC-RAIS uses an important data structure, *i.e.*, Map_table, to record the mapping information of the redirected write data. As shown in Fig. 4, the *Map_table* is used to log all the write data that is stored on the different data zones. MC-RAIS uses the non-volatile memory to store the content of the Map_table to prevent data loss in case of power failure. The Map_table is similar to the mapping information within the FTL in a single SSD. Thus, the protection schemes of mapping information within FTL are also applicable to the Map_table. Since each incoming request should be checked in the Map_Table, MC-RAIS use the bloom filter scheme to improve the query efficiency which further reduces the space overhead.

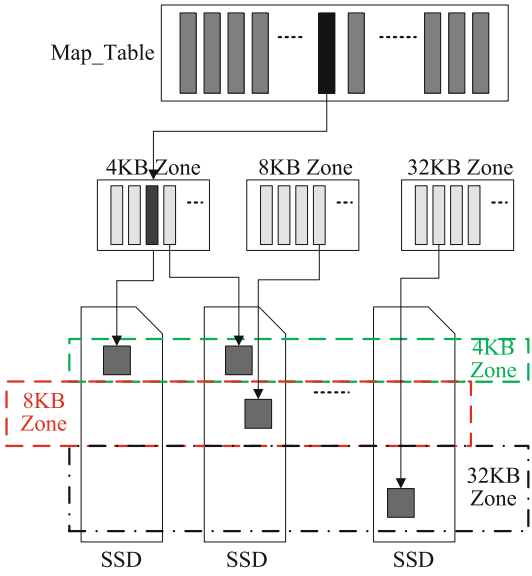


Fig. 4. Mapping table of MC-RAIS.

3.2 Request Processing Workflow

When processing the incoming requests, an important issue must be addressed for the new data blocks because no access history has been kept for the new data block. To be simple and effective, the new data blocks are first stored on

the read-intensive zone. If the data blocks are read-intensive, no migration is needed. Otherwise, the data blocks will be migrated to other zones when they are updated again, thus involving on extra write operations. By doing this, the frequency of the data migration is reduced.

Upon receiving the write request, MC-RAIS first checks the Map_table to determine which data zone to service the request. If the write request hit the Map_Table, the index will be retrieved to locate the data zones to service the write request. Moreover, the corresponding *w_count* value in the Map_table will be increased to record the write access frequency. Otherwise, the write data will be written on the read-intensive zone and be recorded in the Map_table. For read requests, MC-RAIS also checks the Map_Table firstly to determine which data zone to service the request. If the requested data are stored on multiple data zones, the read request will be replaced by multiple read requests to the different data zones in the same RAIS. After all the these read requests have completed, the requested read data is reconstructed and returned to the upper layer. Otherwise, the read request will be serviced as usual and the corresponding *r_value* in the Map_table will be increased to record the read access frequency.

Once the data blocks on the read-intensive zone are identified to be write-intensive, these data blocks will be migrated to write-intensive zones in the MC-RAIS. In order to reduce the migration overhead, the migration is processed when updating the same data blocks. In other words, when the data is updated, these data blocks will be stored on the write-intensive zone rather than the original read-intensive zone. To ensure the data consistency, the log entry of the write data is deleted from the original zone in the Map_Table after the migrate process completes. Moreover, to improve the efficiency of the migration process, the data blocks are stored sequentially on the targeted data zones in MC-RAIS.

3.3 Data Consistency

Data consistency in our MC-RAIS design includes the following two aspects: (1) The key data structure must be safely stored, (2) The user read requests must fetch the right data.

First, to prevent the loss of the Map_table in the event of a power supply failure or a system crash, MC-RAIS stores the contents of the Map_table in the non-volatile RAM (NVRAM). Since the size of the Map_table is general small, it will not incur distinct extra hardware cost to the RAIS system. A small battery or capacitor may delay shutdown until the contents in the RAM are safely saved to an area of flash memory reserved for the purpose. On the other hand, in order to improve the write performance by using the write-back strategy, NVRAM is commonly deployed in the RAIS controller or the RAIS system. Therefore, it is easy and reasonable to use the NVRAM to store the contents of the Map_table.

Second, since the up-to-date data for a read request can be stored on different data zones, each incoming read request is first checked in the Map_table to determine whether it should be serviced by the multiple data zones. If so, the read request will be split into multiple read requests to keep the fetched data be always up-to-date. Otherwise the read request will serviced as it is.

4 Performance Evaluations

In this section, we first describe the experimental setup and methodology of this paper. Then we evaluate the performance of our proposed MC-RAIS scheme and the comparisons of the fix-chunk-size scheme through trace-driven simulations with realistic enterprise workloads.

4.1 Experimental Setup and Methodology

To evaluate the efficiency of our proposed MC-RAIS scheme, we have implemented a prototype of the MC-RAIS scheme by integrating it into an open-source SSD simulator developed by Microsoft Research (MSR) [1]. The MSR SSD simulator, an extension of DiskSim from the Parallel Data Lab of CMU [4], has been released to the public and widely used to evaluate the performance of the SSD-based storage systems in many studies [12,16,18]. In this paper, we extended the original DiskSim and the MSR SSD simulator to implement our proposed MC-RAIS scheme. The specifications of each SSD in the SSD simulator are shown in Table 1.

Table 1. SSD model parameters.

SSD model	
Total Capacity	Configuable
Reserved Free Blocks	15 %
Minimum Free Blocks	5 %
Cleaning Policy	Greedy
Flash Chip Elements	64
Planes Per Package	4
Blocks Per Plane	512
Pages Per Block	64
Page Size	4 KB
Page Read Latency	25 us
Page Write Latency	200 us
Block Erase Latency	1.5 ms

In the evaluations, we compare the performance of the MC-RAIS scheme with that of the fix-chunk-size schemes, in terms of user response time. In the fix-chunk-size configuration, we also set different chunk sizes in different experiments: 4 KB, 8 KB and 32 KB (short for 4 KB-RAIS, 8 KB-RAIS and 32 KB-RAIS). For the MC-RAIS scheme, each zone size is configured to be 60GB by default. We use three realistic enterprise-scale workloads to study the impact of our proposed MC-RAIS scheme. The three realistic enterprise-scale workloads were collected from the Microsoft Cambridge Research [3]. The main workload parameters of these traces are summarized in Table 2. Each experiment is run three times and the average results are collected.

Table 2. The trace characteristics.

Traces	Read ratio	IOPS	Ave. Read Req. Size (KB)	Ave. Write Req. Size (KB)
proj_0	85.5 %	479	6.4	9.2
stg_0	20.6 %	354	9.3	8.6
hm_0	68.5 %	134	9.7	6.5

4.2 Performance Results

We first conduct experiments on an 4-disk RAIS5 system for the different schemes driven by the three workloads. Figure 5 shows the comparisons of the normalized average response times for the fix-chunk-size schemes with different chunk sizes and our MC-RAIS scheme driven by the three workloads. From Fig. 5, we can see that compared with the fix-chunk-size RAIS schemes, MC-RAIS outperforms 4 KB-RAIS, 8 KB-RAIS and 32 KB-RAIS by 40.2 %, 43.4 % and 49.8 % in terms of the average response time, respectively. The reasons are twofold. First, for the write requests, MC-RAIS reduces the overhead of the parity update, which significantly alleviates the write amplification penalty. Second, the read requests could be serviced by fully exploiting the access parallelism to improve the performance. Therefore, MC-RAIS reduces the overall response time, compared with the fix-chunk-size schemes.

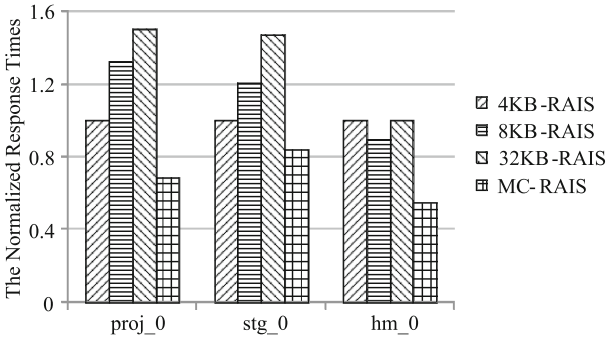


Fig. 5. Comparisons of the normalized average response times for the MC-RAIS scheme and three fix-chunk-size RAIS schemes driven by the three workloads.

To better understand the variance of response times for the three workloads in the experiments, Fig. 6 shows the comparisons of the standard deviation results for the MC-RAIS scheme and three fix-chunk-size RAIS schemes. From Fig. 6, we can first see that the 4 KB-RAIS scheme is capable of working with the most minimal variances. The reason is that the 4KB chunk size is the same as the page size, which makes the performance stable in the situation. Second, we can see that the MC-RAIS scheme is capable of working with more minimal variances

than 8 KB-RAIS and 32 KB-RAIS. In particular, the results show that the MC-RAIS scheme reduces the performance variance by 59.0 % and 45.8 % compared with the 8 KB-RAIS and 32 KB-RAIS schemes driven by the three workloads, respectively. In contrast, we can also see that the performance variability is a serious problem for the fix-chunk-size RAIS schemes.

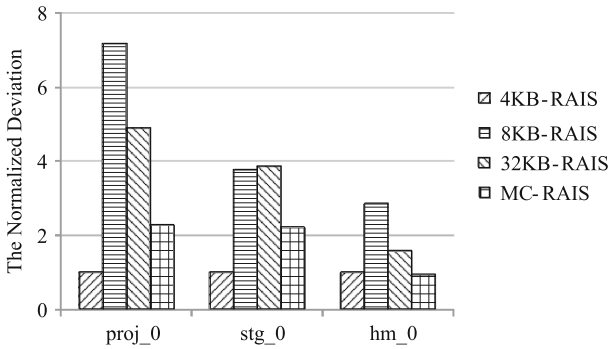


Fig. 6. Comparisons of the standard deviation results for the MC-RAIS scheme and three fix-chunk-size RAIS schemes driven by the three workloads.

4.3 Sensitivity Study

The performance of our MC-RAIS scheme is likely influenced by several important factors, including the number of SSDs in the RAIS and the zone size of the RAIS system.

Number of SSDs. To examine the sensitivity of MC-RAIS to the number of SSDs of the RAIS system, we conduct experiments on the RAIS5 systems consisting of different numbers of SSDs (3, 4 and 5) driven by the proj_0 workload. Figure 7 shows the comparisons of the normalized average response times for the MC-RAIS scheme and three fix-chunk-size RAIS schemes. From Fig. 7, we can see that the average response time decreases with the increasing number of SSDs in the RAIS system. The reason is that the I/O intensity on the individual SSD will decrease when increasing the number of SSDs in the RAIS system, thus reducing the service time for the user I/O requests. Moreover, a RAIS system consisting of more SSDs provides higher parallelism when processing the user I/O requests. Certainly, Fig. 7 also shows that the MC-RAIS scheme consistently outperforms the three fix-chunk-size schemes with different numbers of SSDs. We can also see that the MC-RAIS scheme is not sensitive to the number of SSDs in the RAIS system.

Zone Size. To examine the impact of the zone size of the RAIS system in the MC-RAIS scheme, we conduct experiments on an 4-disk RAIS5 system with zone sizes of 120 GB, 90 GB and 60 GB, respectively. Thus, the overall capacities

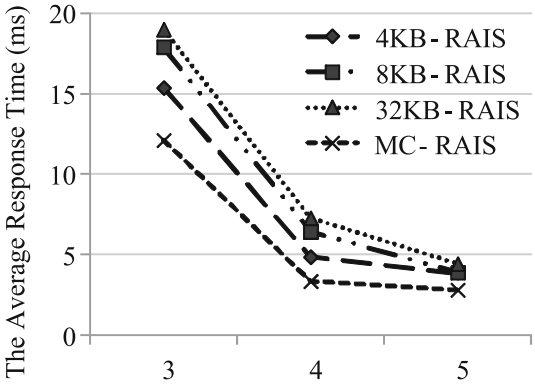


Fig. 7. Comparisons of the normalized average response times for the MC-RAIS scheme and three fix-chunk-size RAIS schemes with different numbers of SSDs (3, 4 and 5).

for MC-RAIS will be 360 GB, 270 GB and 180 GB. The capacity of the fix-chunk-size RAIS scheme is set to be 360 GB. Figure 8 shows the comparisons of the normalized average response times for the MC-RAIS scheme with different zone sizes and three fix-chunk-size RAIS schemes driven by the three workloads. From Fig. 8, we can see that the average response times of the user I/O requests decrease with the increasing zone size of the MC-RAIS system. The reason is that with a larger zone size, the GC operations will be fewer than that with a smaller zone size. Accordingly, the average response time is lower. Moreover, compared with the fix-chunk-size RAIS schemes, MC-RAIS reduces the average response times since it can adaptively store the data blocks with different access characteristics on the three data zones with different chunk sizes. In general, our proposed MC-RAIS scheme fully exploits the characteristics of the SSDs and the data blocks to improve the system performance.

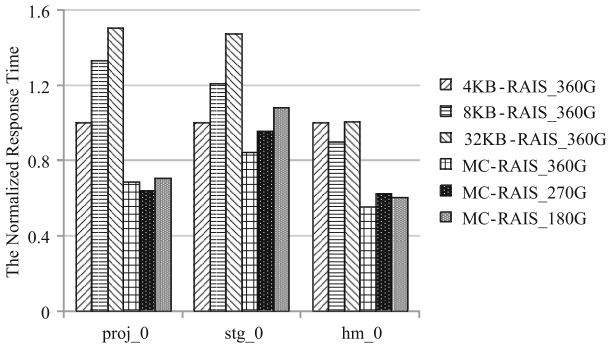


Fig. 8. Comparisons of the normalized average response times for the MC-RAIS scheme with different zone sizes and three fix-chunk-size RAIS schemes driven by the three workloads.

There are also some other important parameters that will affect the system performance, such as the numbers of the zone in the RAIS system and the frequency threshold. We will conduct much more experiments to evaluate their effectiveness in the future. Moreover, we will implement the MC-RAIS scheme in the real RAIS system, such as Linux MD, and evaluate it by using real applications.

5 Related Work

Studies conducted on SSD-based disk arrays fall into two categories, namely, pure SSD-based disk arrays that all the disks in the disk arrays are SSDs [2,9, 10,23,25] and SSD/HDD hybrid disk arrays that utilize the HDDs to assist the SSD-based disk arrays [11,14,22,24]. Our work belongs to the former.

Balakrishnan et al. [2] observed that in the pure SSD-based RAID5 disk array, the load balancing of write requests can cause correlated failures. Diff-RAID creates an age differential in an array of SSDs, distributes the parity blocks unevenly across the disk array and pre-replaces the faster degraded SSD. However, Diff-RAID does not reduce the number of parity updates on the SSDs. It only concentrates the parity updates on few SSDs to make the SSD failures be uneven in the disk array, thus increasing the reliability of the SSD-based RAID5 disk array. On the other hand, the performance of the Diff-RAID system is degraded due to the concentrated parity updates in their storage system. It essentially trades the performance for reliability for the SSD-based disk arrays.

Flash-aware RAID [10] reduces the number of internal write operations caused by the parity updates by using the delayed parity update strategy and the partial parity technique. It utilizes the cache in the upper layer to achieve this goal and then improve the performance of the RAIS system, which is orthogonal to our proposed MC-RAIS scheme. WeLe-RAID [9] introduces the Wear-leveling mechanism among the flash-based SSDs to enhance the endurance of the entire SSD-based disk arrays. It uses the age-driven parity distribution scheme to guarantee the efficiency of the wear-leveling mechanism among the flash-based SSDs and bring the performance benefit with better load balance.

On the other hand, since SSDs and HDDs have both advantages and disadvantages, none of them is the perfect choice for all applications. Some studies tried to construct hybrid disk arrays consisting of both SSDs and HDDs. Xie and Sun [22] proposed a hybrid disk array architecture, named HIT, for data-intensive applications. HIT periodically redistributes the data between SSDs and HDDs to adapt to the changing of the data access patterns. In addition, HIT balances the performance, reliability and energy efficiency to determine the appropriate data placement and migration strategy. However, HIT only considers the data placement scheme between the HDD-based disk array and the SSD-based disk array, but not in a real hybrid disk array. HPDA [14,15] is an enhanced hybrid RAID4 disk array composed of both HDDs and SSDs. It uses an HDD to service as the dedicated parity device, thus avoiding the parity updates on the SSDs. Moreover, it uses two HDDs to construct a mirroring write buffer to

absorb the incoming small write requests, thus improving both the performance and reliability of the hybrid disk arrays.

However, all the above schemes have not considered the efficiency of the garbage collection operations on the RAIS performance. Kim et al. [12] found that the uncoordinated GC processes on individual SSDs amplified the performance degradation of the RAIS system and proposed a RAID-level Global Garbage Collection (GGC) mechanism to alleviate the performance variability for the RAIS system. However, GGC forces all SSDs in the RAIS system to process the GC operation at the same time, which makes the RAIS unavailable to service the applications during the GC period. Moreover, GGC requires the SSDs to be RAIS-aware and only considers RAIS0 that has lower reliability than RAIS5. Differently, our proposed MC-RAIS scheme exploits the workload characteristics to place the data blocks on multiple data zones with different chunk sizes.

6 Conclusion

With the rapid development and wide applications of the SSD device, the SSD-based disk arrays become one of the most effective ways to solve the performance and energy bottlenecks of HDD-based storage systems. Due to the different characteristics between HDDs and SSDs, straightforwardly applying the RAID technique to SSDs is challenging. In this paper, we first conducted extensive experiments on the efficiency of the chunk size on the RAIS performance. Based on the experimental results, we proposed a Multi-Chunk RAIS (short for MC-RAIS) to improve the performance of the SSD-based storage systems. The evaluation results show that the performance of MC-RAIS outperforms the that of existing fix-chunk-size SSD-based disk arrays by more than 50 %.

Our proposed MC-RAIS scheme is an ongoing research project and we are currently exploring several directions for the future work. First, we will implement the MC-RAIS scheme in a real RAIS system, such as Linux MD, and evaluate it by using real applications. Second, we will investigate how the system reliability is affected by the MC-RAIS scheme and evaluate its efficiency.

Acknowledgments. This work is supported by the National Natural Science Foundation of China under Grant No. 61100033, No. 61472336 and No. 61402385, National Key Technology R&D Program Foundation of China under Grant No. 2015BAH16F02, Fundamental Research Funds for the Central Universities (No. 20720140515).

References

1. Agrawal, N., Prabhakaran, V., Wobber, T., Davis, J., Manasse, M., Panigrahy, R.: Design tradeoffs for SSD performance. In: Proceedings of the 2008 USENIX Annual Technical Conference (USENIX 2008), Boston, MA, June 2008
2. Balakrishnan, M., Kadav, A., Prabhakaran, V., Malkhi, D.: Differential RAID: rethinking RAID for SSD reliability. In: Proceedings of the 5th European Conference on Computer Systems (EuroSys 2010), Paris, France, April 2010

3. Block I/O Traces in SNIA. http://iotta.snia.org/tracetypes/3
4. Bucy, J., Schindler, J.S., Schlosser, S.W., Ganger, G.R.: The DiskSim Simulation Environment Version 4.0 Reference Manual, May 2008
5. Caulfield, A.M., Coburn, J., Mollov, T., De, A., Akel, A., He, J., Jagatheesan, A., Gupta, R.K., Snavely, A., Swanson, S.: Understanding the impact of emerging non-volatile memories on high-performance, io-intensive computing. In: Proceedings of the 2010 International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2010), New Orleans, LA, November 2010
6. Chen, F., Koufaty, D.A., Zhang, X.: Understanding intrinsic characteristics and system implications of flash memory based solid state drives. In: Proceedings of the 11th ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2009), Seattle, WA, June 2009
7. Chen, P.M., Lee, E.K.: Striping in a RAID level 5 disk array. In: Proceedings of the 1995 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 1995), Ottawa, Canada, May 1995
8. Dirik, C., Jacob, B.: The performance of PC solid-state disks as a function of bandwidth, concurrency, device architecture, and system organization. In: Proceedings of the 36th International Symposium on Computer Architecture (ISCA 2009), Austin, TX, June 2009
9. Yimo, D., Fang, L., Zhiguang, C., Xin, M.: WeLe-RAID: a SSD-Based RAID for system endurance and performance. In: Altman, E., Shi, W. (eds.) NPC 2011. LNCS, vol. 6985, pp. 248–262. Springer, Heidelberg (2011)
10. Im, S., Shin, D.: Flash-aware RAID techniques for dependable and high-performance flash memory SSD. IEEE Trans. Comput. **60**(61), 80–92 (2011)
11. Kim, Y., Gupta, A., Urgaonkar, B., Berman, P., Sivasubramaniam, A.: HybridStore: a cost-efficient, high-performance storage system combining SSDs and HDDs. In: Proceedings of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2011), Singapore, July 2011
12. Kim, Y., Oral, S., Shipman, G.M., Lee, J., Dillow, D.A., Wang, F.: Harmonia: a globally coordinated garbage collector for arrays of solid-state drives. In: Proceedings of the 27th IEEE Symposium on Mass Storage Systems and Technologies (MSST 2011), Denver, CO, May 2011
13. Mao, B., Feng, D., Wu, S., Chen, J., Zeng, L., Tian, L.: RAID10L: a high performance raid10 storage architecture based on logging technique. In: Proceedings of the 13th IEEE Asia-Pacific Computer Systems Architecture Conference (ACSAC 2008), Hsinchu, Taiwan, August 2008
14. Mao, B., Jiang, H., Feng, D., Wu, S., Chen, J., Zeng, L., Tian, L.: HPDA: a hybrid parity-based disk array for enhanced perfromance and reliability. In: Proceedings of 24th International Parallel & Distributed Processing Symposium (IPDPS 2010), Atlanta, GA, April 2010
15. Mao, B., Jiang, H., Wu, S., Tian, L., Feng, D., Chen, J., Zeng, L.: HPDA: a hybrid parity-based disk array for enhanced perfromance and reliability. ACM Trans. Storage, **8**(1), 1–20 (2012). Article No. 4
16. Park, S., Seo, E., Shin, J., Maeng, S., Lee, J.: Exploiting internal parallelism of flash-based SSDs. IEEE Comput. Archit. Lett. **9**(1), 9–12 (2010)
17. D.A. Patterson, G. Gibson, and R. H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In Proceedings of the International Conference on Management of Data (SIGMOD 1988), Chicago IL, June 1988

18. Wu, G., He, B.: Reducing SSD read latency via NAND flash program and erase suspension. In: Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST 2012), San Jose, CA, Febuary 2012
19. Wu, S., Chen, X., Mao, B.: GC-RAIS: garbage collection aware and redundant array of independent SSDs. J. Comput. Res. Develop. **50**(1), 60–68 (2013)
20. Wu, S., Jiang, H., Feng, D., Tian, L., Mao, B.: WorkOut: I/O workload outsourcing for boosting the RAID reconstruction performance. In: Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST 2009), San Francisco, CA, Febuary 2009
21. Wu, S., Jiang, H., Mao, B.: IDO: intelligent data outsourcing with improved RAID reconstruction performance in large-scale data centers. In: Proceedings of the 26th USENIX Large Installation System Administration (LISA 2012), San Diego, CA, December 2012
22. Xie, T., Sun, Y.: Dynamic data reallocation in hybrid disk arrays. IEEE Trans. Parallel Distrib. Syst. **21**(9), 1330–1341 (2010)
23. Yi, L., Shu, J., Ou, J., Zheng, W.: CG-Resync: conversion-guided resynchronization for a SSD-based RAID array. In Proceedings of the 31st International Conference on Computer Design (ICCD 2013), Asheville, NC, October 2013
24. Zeng, L., Feng, D., Mao, B., Chen, J., Wei, Q., Liu, W.: HerpRap: A Hybrid Array Architecture Providing Any Point-in-time Data Tracking for Datacenter. In: Proceedings of the 2012 IEEE International Conference on Cluster Computing (Cluster 2012), Beijing, China, September 2012
25. Zhang, Y., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H.: Warped mirrors for flash. In: Proceedings of the 29th IEEE Symposium on Massive Storage Systems and Technologies (MSST 2013), Long Beach, CA, May 2013

Springer

http://www.springer.com/978-3-319-27139-2

Algorithms and Architectures for Parallel Processing
15th International Conference, ICA3PP 2015,
Zhangjiajie, China, November 18-20, 2015,
Proceedings, Part IV

Wang, G.; Zomaya, A.; Martinez Perez, G.; Li, K. (Eds.)
2015, LI, 845 p. 353 illus. in color., Softcover
ISBN: 978-3-319-27139-2