2013 IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing

SPD-RAID4: Splitting Parity Disk for RAID4 Structured Parallel SSD Arrays

Wen Pan, Feng Liu
Computer & Information School
Hefei University of Technology
Hefei, Anhui, China
wenwen412@gmail.com
fengliu089@gmail.com

Tao Xie
Computer Science Department
San Diego State University
San Diego, California, USA
txie@mail.sdsu.edu

Yanyan Gao, Yiming Ouyang, Tian Chen
Computer & Information School
Hefei University of Technology
Hefei, Anhui, China
littlek.gao@gmail.com
{oyym, ct}@hfut.edu.cn

Abstract—Data-intensive applications like video processing and bioinformatics increasingly demand a high-performance and highly reliable storage system. Hard disk drive (HDD) has long been used as a standard storage device for most existing storage systems. Recently, NAND-flash memory based solid state drives (SSDs) are gradually exploited to replace HDDs in enterprise computing infrastructures due to their salient features such as high performance, low power consumption, and excellent shock-resistance. With rapid price decreasing and capacity increasing, flash SSD based disk arrays organized in some RAID structures become feasible and greatly needed. In this paper, we propose a new RAID4 architecture called SPD-RAID4 (Splitting Parity Disk - RAID4) for parallel SSD arrays. It splits the parity disk of a traditional RAID 4 array into configurable number of smaller ones. Thus, multiple small parity SSDs operate in tandem with data SSDs to achieve a high performance and high level of reliability. We compare the performance of SPD-RAID4 with conventional RAID4 and RAID5 architectures by using both real-world traces and synthetic benchmarks. Experimental results demonstrate that in terms of mean response time SPD-RAID4 outperforms standard RAID5 structured SSD arrays by up to 20.3%. Compared with standard RAID4, SPD-RAID4 achieves a performance gain up to 40.6%.

Keywords—flash memory; RAID; RAID4; SSD arrays

I. INTRODUCTION

Compared with modern CPU speed, the speed of a conventional hard disk drive (hereafter, HDD) is much slower. Due to its physical properties, HDD has some inherent disadvantages such as poor performance, low energy-efficiency, and weak robustness, which make it the bottleneck of a high-performance computing system [7][21]. As a result, NAND flash memory based solid state drive (hereafter, SSD) recently has attracted intensive attention and has gradually become an alternative storage device to HDD in enterprise storage systems [19][22][26]. Unlike rotating-based HDD, SSD is made up of semi-conductor chips without any moving parts. It possesses several extraordinary features such as low power consumption, excellent shock and temperature resistance, and extremely high performance in random read [1]. On the other hand, it also bears some obvious drawbacks like poor random write performance, erase-before-write issue, and wear-out problem [19][22][30]. With the application of MLC (multi-level cell) and TLC (triple-level cell) techniques, the price of SSD is decreasing while its capacity is increasing dramatically [2][25]. Thus, similar as a hard disk array, an array of SSDs organized in a RAID (Redundant Array of Independent Disk) [24] structure that can satisfy the performance, capacity, and reliability requirements of an enterprise storage system becomes both feasible and needed [12] [14][15][17][18].

Traditionally, the RAID technique improves the performance and reliability of HDD based storage systems by grouping a number of smaller disks together rather than building one large and expensive drive [24]. It has been successfully employed in almost all existing enterprise storage systems. It mainly employs the parallel I/O technique to improve performance and exploits data redundancy mechanisms to enhance the reliability of storage systems [24]. RAID 5 (block-level striping with distributed parity) distributes parity along with the data and it can tolerate a single drive failure. It is one of the most widely used disk array organizations. RAID 4 (block-level striping with dedicated parity) is equivalent to RAID 5 except that all parity data are stored on a single drive. Compared with RAID 5, it has been seldom used in HDD arrays. The main reason is that the use of a dedicated parity drive could create a performance bottleneck because the parity data must be written to a single, dedicated parity drive for each block of non-parity data [24]. Intuitively, the RAID technology can also be applied onto SSDs so that RAID 4 or RAID 5 structured SSD arrays can be built. Applying RAID 4 in an SSD array, however, is even more challenging as the dedicated parity SSD could wear out much faster than a data SSD [23]. To overcome this problem, current solution is to utilize an HDD as the dedicated parity drive while multiple SSDs are used as data drives [20][23].

Although existing SSD-HDD hybrid RAID 4 architecture can avoid parity SSD wear-out faster problem [20], it may also cause the following potential issues. First of all, the inherent performance gap between an SSD and an HDD could adversely affect the overall performance because the overall write performance still largely depends on the performance of the HDD parity drive, which is usually lower than that of an SSD. Next, data SSDs could wear out at similar rates, which can result in correlated failures as the data SSDs age in unison [15]. Therefore, the risk of irreparable more-than-one-SSD-failure becomes high. In addition, a hybrid RAID structure needs separated RAID controllers dedicated for SSDs and HDD, which increases the design complexity.

To alleviate the limitations of existing hybrid RAID 4 architectures, in this paper, we propose a new SSD RAID 4 structure named SPD-RAID4 (Splitting Parity Disk-RAID4),

978-0-7695-5088-6/13 $31.00 © 2013 IEEE
DOI 10.1109/HPCC.and.EUC.2013.12

IEEE computer society

which only utilizes SSDs. Moreover, it splits the parity disk of a traditional RAID 4 array into a configurable number of smaller ones. For example, SPD-RAID4 turns a standard RAID 4 array with five 512 GB SSDs (four data SSDs plus one parity SSD) into a new structure with four 512 GB data SSDs and two 256 GB parity SSDs. The new SPD-RAID4 structure will not increase the total cost as a 512 GB Intel SSD is currently $424.99 while the price of a 256 GB Intel SSD is $211.99 [13]. Using multiple smaller SSDs to replace the single parity SSD can enhance the parallelism of parity data processing, and thus, improve the overall performance. Our experimental results show that SPD-RAID4 performs much better than standard SSD RAID 4 and RAID 5.

The rest of the paper is organized as follows. Related work and motivation will be discussed in section II. We present the implementation of SPD-RAID4 in section III. In section IV, we evaluate the performance of SPD-RAID4 based on real-world traces and synthetic benchmarks. Section V concludes this paper with a summary and future direction.

II. Related Work and motivation

A. SSD Basics

Unlike HDDs, SSDs are made up of semiconductor memory chips, and thus, have no moving parts. They can avoid HDD's long seek time and rotational latency, which are in the order of magnitude of millisecond (ms). Despite of SSD's desirable properties like high energy efficiency, excellent temperature resistance, and high random read performance, it also has certain essential limitations.

Firstly, current SSDs suffer from poor performance of random write [30]. The reason is that in flash memory, each block must be erased before it can be written, a characteristic known as erase-before-write. For a Samsung's K9XXG08UXM series NAND flash memory, it takes 25 μs and 200 μs to read from and write into a flash page, whereas it needs about 1.5 milliseconds to execute an erase operation [1].

Secondly, flash memory wears out after repeated program/erase (P/E) operations, which affects the reliability of SSDs. Typically, for a single-level cell (SLC) NAND flash memory, its maximum available P/E cycles could reach 100,000, whereas for a multi-level cell (MLC) flash memory, its available P/E cycles are only 10,000 [20]. A block of flash memory could fail when it has been erased more than its available P/E cycles. These two limitations of SSDs must be taken into consideration when designing an SSD based storage system, especially for SSD based arrays.

B. Related Work

Applying SSDs in high-end server systems has attracted intensive attention in recent years due to both their excellent properties such as low-power consumption and high performance and their decreasing prices [1][8][19][22][26]. Typically, existing applications of SSDs in server domains can be categorized into two camps: a HDD-SSD hybrid storage architecture [11][16][20][28] and a RAID based pure SSD array [3][5][12][14][15][17][18][24]. Techniques in the first camp usually concentrate on how to combine one or multiple SSDs with either a single HDD or an array of HDDs to form various hybrid storage architectures. One example is a hybrid storage system called HybridStore proposed by Kim et al. [16]. It exploits the complementary properties of HDD and SSD to provide improved performance and service differentiation under a certain cost budget [16]. Mao et al. proposed an HPDA (Hybrid Parity-based Disk Array) architecture, which combines a group of SSDs and two HDDs [20]. In HPDA, the SSDs (data disks) and part of one HDD (parity disk) compose a RAID4 disk array. Meanwhile, a second HDD and the free space of the parity disk are mirrored to form a RAID1-style write buffer that temporarily absorbs the small write requests and acts as a surrogate set during recovery when a disk fails. Ren and Yang recently proposed a new hybrid storage architecture named I-CASH (Intelligently Coupled Array of SSD and HDD) [28]. The SSD stores seldom-changed and mostly read reference data blocks whereas the HDD stores a log of deltas between currently accessed I/O blocks and their corresponding reference blocks in the SSD so that random writes are not performed in SSD during online I/O operations [28]. Essentially, these hybrid storage architectures share one common idea: smartly utilize the complementary properties of SSD and HDD.

With rapid price decreasing and capacity increasing, pure SSD arrays organized in some RAID structures have become both feasible and needed. Pure SSD based RAID arrays normally can be classified into two categories based on the granularity of underlying storage device. While the coarse grain group directly applies RAID schemes on top of an array of independent SSDs [5][15][23], the fine grain group employs RAID structures on all flash chips within a single SSD [12][17][18]. In this research, we focus on the first category. When applying RAID schemes on SSDs, we need to adapt existing RAID strategies in order to make the best use of SSDs. Usually, a RAID mechanism on flash chips within a single SSD is integrated into the flash translation layer (FTL) of the SSD [1][8]. In other words, the FTL not only implements the function of address translation, garbage collection, and wear-leveling, but also provides a RAID mechanism.

Existing RAID based SSD array research focuses on the following two important aspects: improving performance [12][14][17] and enhancing reliability [15][18]. The main approach to improving SSD array performance is to reduce the number of write operations for the parity updates, which is identified as a performance bottleneck for RAID 5 SSD arrays [12][17]. Im and Shin proposed a scheme to delay the parity update that must accompany each data write in the original RAID technique [12]. Similarly, Lee et al. developed a new technique called FRA (Flash aware Redundancy Array) [17]. In this technique, parity updates are postponed so that they are not included in the critical path of read and write operations [17]. Instead, they are scheduled for when the device becomes idle. The above two [12][17] are the typical representations for flash chip based SSD arrays. Jeremic et al. discovered several pitfalls for deploying SSDs in common RAID level configurations, which can lead to severe performance degradation after conducting a deep analysis of SSD RAID configuration issues [14]. Unlike [12] and [17], their solution is to utilize over-provisioning, which offers a potential solution to

this problem.

Also, researchers have been working on improving the reliability and performance of pure SSD RAID mechanisms by inventing various SSD organizations, with several notable and effective outcomes. To enhance the reliability of an SSD array, Kadav et al. presented Diff-RAID, a new RAID variant that distributes parity unevenly across SSDs to create age disparities within arrays [15]. It can provide a great trade-off between throughput and reliability by dynamically regulating the distribution ratio of parity data. In this way, correlated failures as arrays age in unison can be reduced. Diff-RAID distributes more parity on some older devices to create an age gap to improve reliability by avoiding more than one SSD failures at one time [15]. However, parity devices that have taken more requests than others could receive an even heavier load, and thus, aggravating the load imbalance problem.

To avoid the problem that Diff-RAID bears, Du et al. proposed Wele-RAID [5], which introduces a novel wear leveling scheme among all flash SSDs. The wear leveling strategy is derived from the age-driven parity distribution mechanism to enhance the endurance of entire SSD RAID system [5]. Nevertheless, it requires replacing the entire array with new SSDs when the whole system approaches the end of its lifetime, which increases the hardware cost. Lee et al. found that the bit error rate of flash memory enlarges rapidly as the number of P/E (program/erase) cycles increases [18]. To relieve these problems, they proposed a lifespan-aware reliability scheme, which adopts RAID technologies together with ECCs (error correction codes) [18].

One obvious drawback of RAID 4 is that the parity disk could become a performance bottleneck even for workloads with a small percentage of writes because every write operation needs to update the parity disk. In order to overcome parity SSD's fast wearing out problem, Park et al. [24] proposed a heterogeneous RAID 4 array by replacing parity SSD with a parity hard disk drive. Qin et al. [27] proposed a flash memory redundant array that is similar to a RAID-4 array. In this scheme, they add an independent channel (parity channel) to an SSD to store specialized parity data and utilize a built-in NVRAM to cache the parity data updates for minimal write to flash memory in parity channel. Our SPD-RAID4 scheme concentrates on SSD array performance enhancement by using an approach different from existing ones [12][14][17]. SPD-RAID4 exploits the parallelism of multiple small capacity parity SSDs to significantly improve the performance.

C. Motivation

There are multiple types of RAID organizations such as RAID 0, RAID 1, RAID 4, RAID 5, and RAID 6 [24]. In addition, some RAID structures can be combined together to form a compound RAID architecture like RAID 10. Simply applying an existing RAID structure to an array of SSDs without taking their characteristics into consideration could cause problems. In what follows, we first analyze which RAID format is suitable for an SSD array. Next, we explain why we need to change the architecture of a standard RAID 4 to make it suitable for SSD arrays.

RAID 0 is not suitable for SSD arrays where data reliability has to be provided as it does not employ any data redundancy schemes although it can boost performance. RAID 1, which provides a mirroring disk to offer duplicated data redundancy, is too expensive for SSD arrays due to SSD's relatively high overhead in terms of the dollar cost per gigabyte [13]. In addition, each write request incurs two write operations, one on a data disk and one on its mirroring disk, which degrades the SSD reliability due to the wear-out problem. Although RAID 5, one of the most widely used RAID formats, could provide a decent performance and survive a single disk failure, it imposes some difficulties when it is applied on an SSD array. First, frequent parity update operations distributed across all SSDs could affect overall performance and SSDs' reliability due to the fact that parity data are mixed together with normal data on all SSDs in a RAID5 format. In other words, the stream of normal data operations and parity read/writes could interfere with each other, and thus, lowers the performance that a user can perceive. Besides, managing both normal data and parity data on one SSD increases the complexity of data management.

On the other hand, RAID 4, which uses a dedicated disk to store parity data, can eliminate the interference between normal data operations and parity data operations as it separates them onto different disks. As a result, the data management scheme could also be simpler than that of RAID 5. The only concern of applying a traditional RAID 4 on an SSD array is that the single parity SSD could become both a performance bottleneck and a potential failure point. This is because frequent updates could largely lower the performance of the single parity SSD. Also, they can wear it out rapidly. To overcome these problems while enjoying the simplicity of RAID 4, we construct a novel variant of a conventional RAID 4 organization for SSD arrays by splitting the single parity SSD into a configurable number of smaller SSDs. In this way, the two problems mentioned above can be alleviated.

III. THE SPD-RAID4 ARCHITECTURE

In this section, we first introduce the SPD-RAID4 architecture, which splits the parity SSD of a standard RAID 4 structure into a configurable number of smaller ones so as to achieve a higher performance. Next, implementation details are explained.

A. Architecture of SPD-RAID4

Fig. 1 shows the architecture of the proposed SPD-RAID4 structure. SPD-RAID4 has one data sub array and one parity sub array. All read/write requests from outside will be directed to the data sub array, whereas the parity sub array processes all parity updates. Basically, it is a variant of a standard SSD structure, which splits the parity SSD into a configurable number of smaller ones. It is composed of m data SSDs and n small capacity parity SSDs. When a request arrives, the RAID controller divides it into multiple one-page size sub-requests. Each of these sub-requests is dispatched to a data SSD in a round-robin fashion. The size of a flash page is set to 2 KB in our experiments. For a write operation, SPD-RAID4 also needs to internally create parity read and write operations to deal with parity data. When updating the parity data on the parity sub array, we also employ the round-robin algorithm to write parity data evenly across all parity SSDs (see Fig. 1). In addition, we

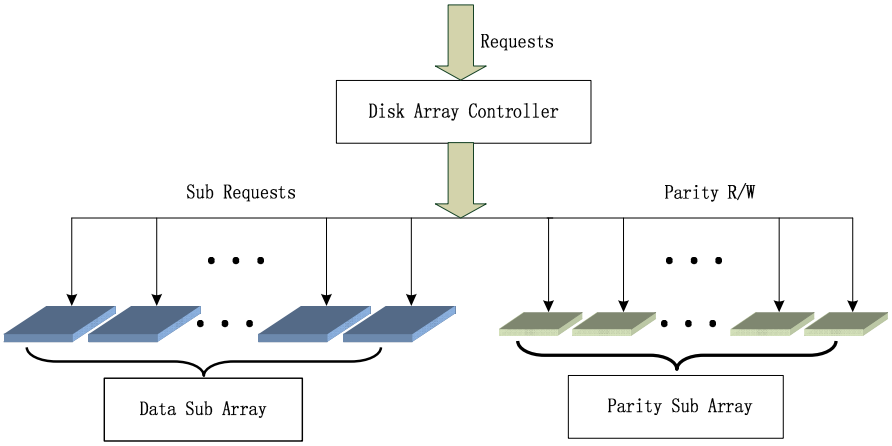

Figure 1. The architecture of SPD-RAID4.

adopt two methods to update parity data: RMW (Read-Modify-Write) and RCW (Read-Reconstruct-Write).

While the pre-read count of RMW equals to the sum of the number of devices which received write requests and one (i.e., parity read), the pre-read count of RCW is the number of devices which do not have write requests. After comparing the pre-read count of RMW and RCW, we will use the one that results in a smaller pre-read count in order to reduce pre-read overhead. If they are equal, we select the RCW method, which does not depend on the parity information so that the probability of coding errors becomes lower. When a request spans across two or more stripes, the parity SSDs can work in parallel, and thus, significantly boosts the SSD performance.

Since serving read request does not involve parity updates, the read performance of SPD-RAID4 almost maintains the same as that of standard SSD RAID 4 and RAID 5. On the other hand, in a standard RAID4 SSD array, only one parity SSD undertakes all parity updates, which makes it wear out quickly. This problem can be largely solved in SPD-RAID4 SSD array because multiple parity SSDs evenly receive parity updates. In addition, SSD failures becomes more common either in the data SSDs or in the parity SSDs and when it happens we should make sure that the recovery process satisfy the following three demands: function without taking the system off-line, rapidly restore the system to its fault-free state and have minimal impact on system performance as observed by users. When a parity SSD in an SPD-RAID4 array fails, only the data on this small capacity parity SSD needs to be rebuilt, which is much smaller than that of the parity SSD in a standard RAID 4 structure. Therefore, SPD-RAID4 can obviously decrease the reconstruction time and alleviate the performance degradation caused by data recovery. If a data SSD fails, multiple parity SSDs can serve requests in parallel when recovering data in the failed data SSD. It also can improve the performance during data reconstruction period when compared with standard SSD RAID 4 and RAID 5 structures.

B. *Implementation*

Fig. 2 shows the workflow of SPD-RAID4. Once the host sends a request, the RAID controller gets the device number and its logical page number (LPN) as well as the stripe number by a division operation. The mapping from a logical address to a physical address is controlled by FTL implemented inside each SSD. After the address mapping is finished, SPD-RAID4 selects multiple pages each from the requested pages in data SSDs and free pages from remaining unrequested data SSDs as well as a page derived from one of the parity SSDs to form a stripe. The request can be mapped to one or more stripes. When it spans across more than one stripe, we adopt the round-robin algorithm to write parity data in parity SSDs. Thus, the parity SSDs can work concurrently.

When a read request comes, the RAID controller first splits it into multiple one-page sub-requests, and then, puts each sub request onto the request queue of its corresponding SSD. After the reading process completes, the statistics like response time can be obtained. Upon receiving a write request, the RAID controller also first splits it into multiple one-page sub-requests, and then, it makes a selection between RMW and RCW based on pre-read count. Next, SPD-RAID4 pre-reads the corresponding data and puts the sub-requests into each involved SSD. It then waits for the write operation to complete. Lastly, it updates the parity data either by RMW or RCW approach. When a write request spans across more than one stripe, the parity SSDs can work in parallel to process parity updates. As a result, the write performance can be boosted compared with standard RAID4 SSD arrays.

IV. PERFORMANCE EVALUATION

A. *Experimental Setup*

We developed an SSD RAID simulator that can simulate standard RAID4, RAID5, and SPD-RAID4. Our simulator is built based on a validated single SSD simulator called SSDsim developed by Hu et al. [9][10]. SSDsim is an event-driven, modularly structured, and highly accurate simulator for single SSD. We added about 1,400 lines of C codes to implement a RAID controller on top of SSDSim [9]. The RAID controller fetches a request from a trace file and splits it into multiple sub-requests. And then it dispatches all sub-requests onto

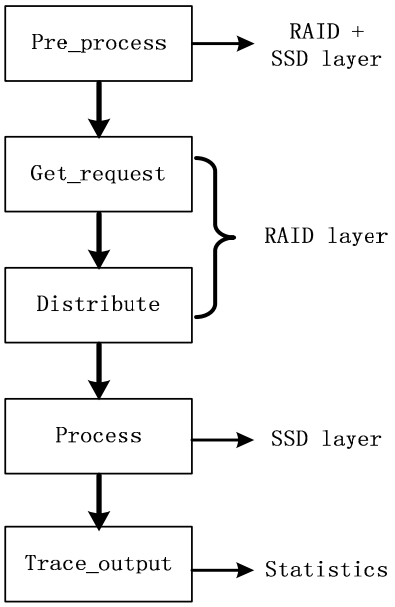

Figure 2. Process workflow of SPD-RAID4.

multiple SSDs. If the request is a write request, the RAID controller also needs to generate several read/write requests to update the parity data. Only after all sub-requests are finished can we say the original request has been served.

We use five real-world traces to compare the performance of RAID 4 and RAID 5 SSD arrays with the proposed SPD-RAID4 with different number of parity SSDs. The five traces and their characteristics are summarized in Table I. The Build trace [4] was collected from the Microsoft Build Server production traces. The Exchange trace [6] was from a collection of production traces collected over a period a 24 hours at Microsoft Exchange Server using the event tracing for Windows framework. The Financial1 and Financial2 [29] are I/O traces from OLTP application running at two financial institutions. The Radius trace [4] was collected for RADIUS authentication server. Table II illustrates the experimental parameters.

B. Real-World Trace Experimental Results

In this section, we only compare the overall performance of SPD-RAID4 and standard RAID 5 while ignoring standard RAID 4 because its performance is consistently lower than that of RAID 5. Furthermore, we evaluate the scalability of SPD-RAID4 by varying the number of parity SSDs. Due to the limited footprint of the five real-world traces, we measure the write counts instead of erase counts of each SSD to evaluate SPD-RAID4's wear-out degree. The number of data SSDs and parity SSDs are both configurable for SPD-RAID4. For simplicity, in our experiments, the number of data SSDs in SPD-RAID4 is fixed to 5 and the number of SSDs in a standard RAID 5 is set to 6. The default capacity of an SSD is set to be 32 GB. The capacity of each data SSD in SPD-RAID4 equals to each SSD in a standard RAID 5 array. We change the capacity of parity SSDs, whose total capacity is less than or equal to that of one SSD in RAID 5 array (i.e., 32 GB).

TABLE I. REAL-WORLD TRACES CHARACTERSTIC

Trace Name	Write Ratio	Ave. Size (KB)	Access Rate (reqs/sec.)	Duration (minute)
Build	45.71	6.5	372	15
Exchange	46.43	12.5	166	15
Financial1	77.88	3	122	782
Financial2	17.65	2	90	683
Radius	88.46	6.5	57	35

TABLE II. EXPERIMENTAL PARAMETERS

Parameters	Values
Page read	20 μ s
Page write	200 μ s
Block erase	1.5ms
Read one byte	25ns
Write one byte	25ns
Page size	2KB

Fig. 3 compares read mean response time and write mean response time of the two RAID architectures. All mean response times including read, write, and overall (see Fig. 4) are normalized to that of a standard SSD RAID 5 array, which always has six 32 GB SSDs. From Fig. 3a, we find that read response time of SPD-RAID4 does not improve remarkably. With the number of parity SSDs increasing from 2 to 5, read performance increases at most 0.3%. It performs best under the Exchange trace, because the average read request size is about 12 KB, which is the largest size in the five real-world traces and it equals to 6 logical pages. However, the read average request sizes of the other four traces are all less than 9 KB, which can not completely exploit the stripping and parallelism of SPD-RAID4. Under Financial2 and Radius trace, SPD-RAID4 performs a little worse than RAID 5. When the number of parity SSDs increases from 2 to 5, read performance almost remains the same. Still, compared to RAID 5, SPD-RAID4's read mean response time decreases 2.1% in the best case.

Fig. 3b demonstrates the best result in Exchange trace and the worst outcome in Financial2 case in terms of mean write response time. Since Financial2 is a read-dominant workload, the number of write requests SPD-RAID4 can serve is relatively small. Therefore, the write performance improves at a minimal percentage. In the best case of Build and Exchange traces, SPD-RAID4 improves 19.5% and 21.6% respectively. Besides, when configured with 5 parity SSDs, write mean response time under Financial1 and Radius cases decreases 5.4% and 5.1%, respectively.

As shown in Fig. 4, only in Financial2 does the overall mean response time of SPD-RAID4 perform a little worse than SSD RAID 5. It is because the read performance is poor under Financial2 trace, which impacts the write performance of SPD-RAID4. However, its mean response time increases at most 0.6% under the Financial2 trace. Similar to the case of write performance, the overall mean response time of SPD-RAID4

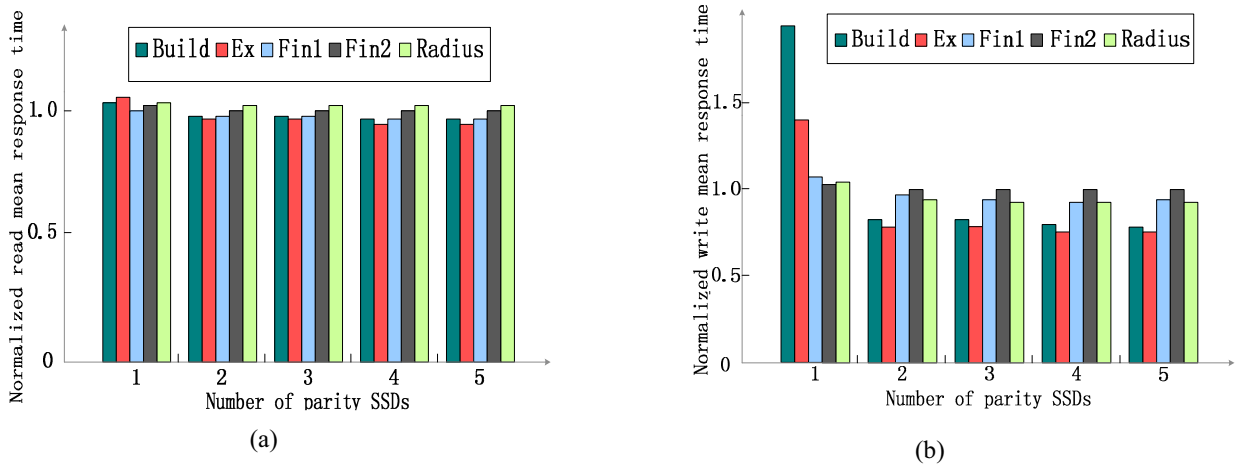

Figure 3. (a) Impacts of the number of parity SSDs on read performance; (b) impacts of the number of parity SSDs on write performance.

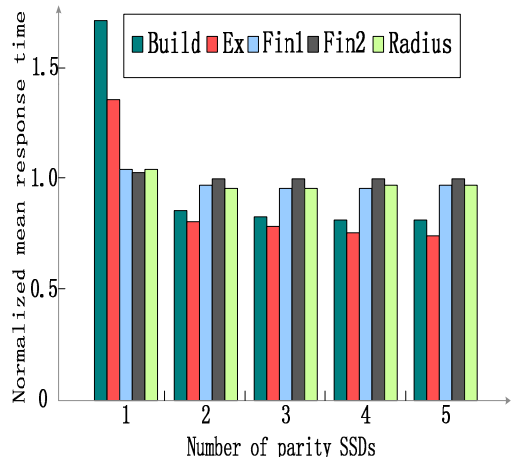

Figure 4. Impacts of the number of parity SSDs on overal performance.

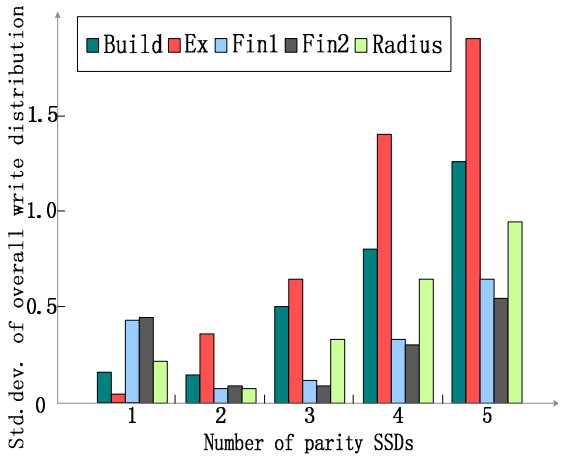

Figure 5. Write distribution comparison between RAID 4 and SPD-RAID4.

shows the best performance under Exchange trace.SPD-RAID4 improves 20.5% and 15.7% at best situations. SPD-RAID4 decreases overall mean response time by 5.0% and 4.4% under Financial1 and Radius trace, respectively.

Since SSDs serve read requests much faster than write requests, write requests dominate an SSD array's request waiting queue. Besides, write requests result in erase operations, which consume SSD's P/E cycle budget. Thus, the evenness of the distribution of writes among all SSDs in an array influences both overall performance and reliability.

Fig. 5 illustrates the standard deviation of write distribution in data SSDs and parity SSDs of an SPD-RAID4 array. A lower standard deviation of write distribution among all SSDs indicates a more even distribution of wear-out, which leads to a higher level of reliability. From Fig. 5 one can see that the standard deviation of writes in SPD-RAID4 with 2 parity SSDs is lower than that of RAID 4 (i.e., only one parity SSD) with the only exception in Exchange scenario. This is because the average request size of Exchange is large, which results in an even distribution of a write request across all data SSDs in a standard RAID 4. However, when there are 2 parity SSDs, the distribution gap between the data SSD and the parity SSD group becomes substantial. Also, Fig. 5 demonstrates that further increasing the number of parity SSDs (i.e., the number of parity SSDs is larger than 2) can only affect the reliability of the entire SSD array for the distribution of writes becomes increasingly uneven. Fig. 6 shows the impacts of the number of parity SSDs on write distribution across all parity SSDs in an SPD-RAID4 array. The write distribution in parity SSDs is very uniform. For example, the highest standard deviation is less than 0.105 and the lowest one is only about 0.005. The implication is two-fold. First of all, an even write distribution among all parity SSDs implies a load balancing, which is helpful for overall performance. Secondly, an even write distribution across parity SSDs makes them age at almost the same speed because their P/E cycles are consumed evenly, and thus, prevents premature failures on parity SSDs.

C. *Experimental Results from Synthetic Benchmarks*

We also use a set of synthetic benchmarks to evaluate our

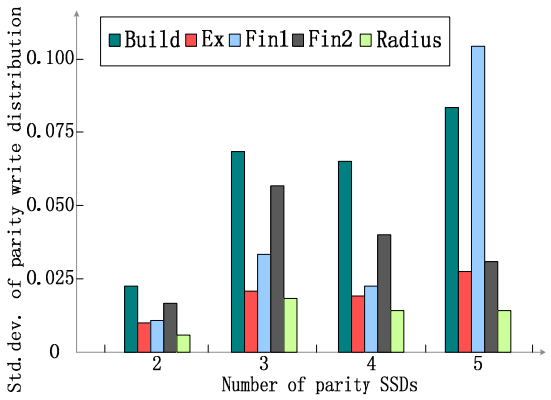

Figure 6. Write disitributions across parity SSDs in SPD-RAID4.

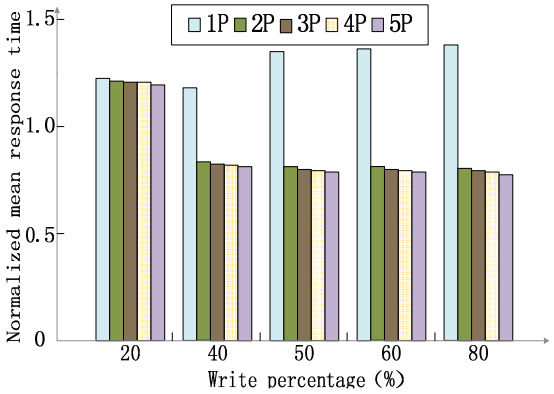

Figure 7. Impacts of percentage of write requests.

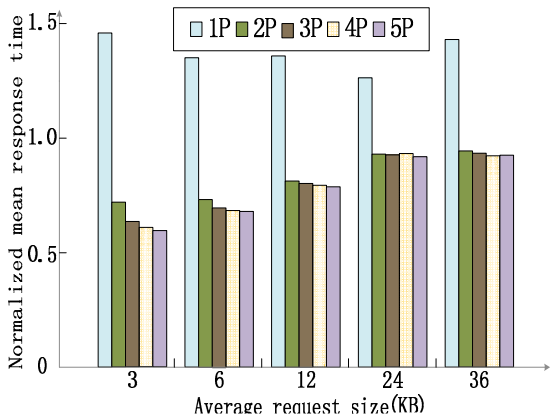

Figure 8. Impacts of average request size.

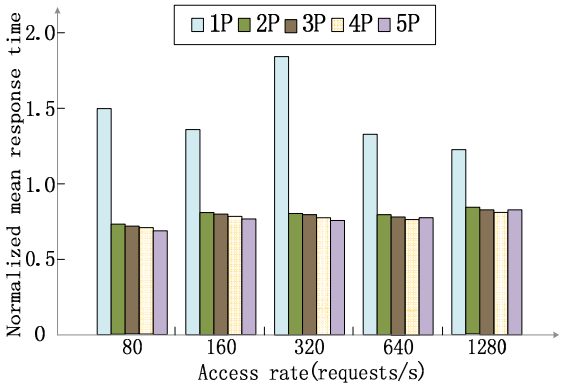

Figure 9. Impacts of aggregate access rate.

SPD-RAID4 architecture. In particular, we evaluate the performance impacts of write percentage, average request size and access rate on both standard RAID 5 and SPD-RAID4. Again, all the experimental results of SPD-RAID4 are normalized to that of a standard RAID 5 structure. The number in front of "P" in the legends of Fig. 7, Fig. 8, and Fig. 9 represents the number of parity SSDs.

Fig. 7 shows the impacts of write request percentage on SPD-RAID4's performance. The default average request size and request access rate are set to 12 KB and 160 requests per second, respectively. We vary the write percentage from 20% to 80%. SPD-RAID4 demonstrates a better performance except for the 20% write scenario. It outperforms RAID5 by up to 20.9%. When write percentage is low, the read performance behaves poor and negatively impacts the overall performance.

We also examine the impacts of average request size on performance. The default write percentage is set to 60% and the default access rate is configured to 160 requests per second. Fig. 8 shows that the smaller the average request size of the synthetic workload, the more improvement can be obtained by SPD-RAID4. In the best situation, SPD-RAID4 can gain 20.3% improvement (see Fig. 8). Clearly, when request size increases, mean response time increases as well.

The impacts of access rate on SPD-RAID4's performance are demonstrated in Fig. 9. The default write percentage and average request size are 60% and 12KB, respectively. The access rate varies from 80 requests per second to 1,280 requests per second. The access rate stands for the intensity of a workload. From Fig. 9, we can see that the average response time increases when the access rate enlarges. This is because the load of the SSD array becomes heavier. The largest performance improvement (i.e., 27.01% improvement with two parity SSDs) can be observed when the access rate is 80 requests per second. When access rate increases to 1,280 requests per second, SPD-RAID4 can only achieve 16.01% performance improvement when there are 2 parity SSDs. When request access rate increases, the queuing delay becomes much larger. Naturally, the average response time increases and performance improvement becomes less remarkable. However, SPD-RAID4 significantly outperforms RAID 4 in all cases.

V. CONCLUSIONS

Disk arrays built by SSDs recently attracted intensive attention from both industry and research communities [12][14][15][17][18]. Existing strategies on SSD based RAID arrays mainly concentrated on RAID 5 [12][17]. To reduce the number of write operations for the parity updates, they normally postpone the parity updates so that the interference caused by frequent updates can be minimized. In this paper, we explore a new approach to applying SSDs in a RAID-like structure where both performance and reliability can be offered.

In particular, we propose a novel SSD based RAID 4 structure called SPD-RAID4, which splits the parity SSD into a configurable number of smaller ones. In SPD-RAID4, the parity SSDs can function in parallel, and thus, can enhance the degree of parallelism and decrease mean response time. Comprehensive experimental results show that SPD-RAID4 can improve the performance of SSD based RAID storage system in terms of mean response times. Compared with a standard RAID5 structured SSD array, SPD-RAID4 can achieve a performance gain up to 20.3%.

In its current format, SPD-RAID4 does not provide a data reconstruction mechanism. One future direction of this research is to develop a new on-line data reconstruction algorithm for SPD-RAID4. To the best of our knowledge, very little research about SSD data reconstruction has been reported in the literature. We are going to develop a data reconstruction scheme for SSD arrays so that data reconstruction time can be reduced and performance degradation during data recovery can be alleviated.

VI. ACKNOWLEDGMENT

This work is sponsored in part by the National Natural Science Foundation of China (NFSC) under Grant No. 61204046 and U.S. National Science Foundation under grant CNS-(CAREER)-0845105.

REFERENCES

[1] N. Agrawal, V. Prabhakaran, T. Wobber, J. Davis, M. Manasse, and R. Panigrahy, "Design Tradeoffs for SSD Performance," Proc. USENIX Annual Technical Conference, pp. 57-70, 2008.

[2] R. Bilton, "Good News for Consumers: Solid State Drives prices are dropping," http://www.zdnet.com/blog/storage/good-news-for-consumers-solid-state-drive-prices-are-dropping/1706, June 2012.

[3] K. Bu, Q. Yin, X. Xu, H. Xu, Y. Zhang, and W. Yi, "Research and Design of Solid State RAID Storage System," Proc. Intelligent System Design and Engineering Application (ISDEA), pp. 143-145, 2010.

[4] Build Trace, Radius Trace, http://iotta.snia.org/traces/158..

[5] Y. Du, F. Liu, Z. Chen, and X. Ma, "Wele-RAID: a SSD- based RAID for System Endurance and Performance," Proc. 8th IFIP international conference on Network and parallel computing (NPC), pp. 248-262, 2011.

[6] Exchange Trace, SNIA IOTTA Repository, http://iotta.snia.org/traces/130.

[7] R. Fang, B. He, C. Mohan, and Y. Wang, "High Performance Database Logging using Storage Class Memory," Proc. 27th Int'l Conf. on Data Engineering(ICDE), pp. 1221-1231, 2011.

[8] A. Gupta, Y. Kim, and B. Urgaonkar, "DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings," Proc. 14th Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 229-240, 2009.

[9] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and S. Zhang, "Performance Impact and Interplay of SSD Parallelism through Advanced Commands, Allocation Strategy and Data Granularity," Proc. ACM Int'l Conf. on Supercomputing (ICS), pp. 96-107, 2011.

[10] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and C. Ren, "Exploring and Exploiting the Multi-level Parallelism Inside SSDs for Improved Performance and Endurance," IEEE Transaction on Computers, Vol. 62, No. 6, pp. 1141-1155, June 2013.

[11] J. Hui, X. Ge, X. Huang, Y. Liu, and Q. Ran, "E-HASH: An Energy-Efficient Hybrid Storage System Composed of One SSD and Multiple HDDs," Proc. Springer-Verlag Berlin Heidelberg, pp. 527-534, 2012.

[12] S. Im and D. Shin, "Flash-Aware RAID Techniques for Dependable and High-Performance Flash Memory SSD," IEEE Transaction on Computer, Vol. 6, Issue 1, pp. 80-92, 2011.

[13] Intel SSD Price - SSDs Starting from 64GB to 256 GB, www.crucial.com/, January, 2013.

[14] N. Jeremic, G. Muhl, A. Busse, and J. Richling, "The Pitfalls of Deploying Solid-State Drive RAIDs," Proc. the 4th Annual Int'l Conf. Systems and Storage, 2011.

[15] A. Kadav, M. Kalarishnan, V. Prabhakaran, and D. Malkhi, "Differential RAID: Rethinking RAID for SSD Reliability," ACM Transactions on Storage, Vol. 6, Issue 2, July 2010.

[16] Y. Kim, A. Gupta, B. Urgaonkar, B. Perman, and A. Sivasubramaniam, "HybridStore: A Cost-Efficient, High-Performance Storage System Combining SSDs and HDDs," Proc. 19th IEEE Int'l Symp. Modeling, Analysis & Simulation of Computer Telecommunication Systems, pp. 227-236, 2011.

[17] Y. Lee, S. Jung, and Y.H. Song, "FRA: A Flash-aware Redundant Array of Flash Storage Devices," Proc. 7th IEEE/ACM Int'l Conf. on Hardware/Software Codesign and System Synthesis, pp. 163-172, 2009.

[18] S. Lee, B. Lee, K. Koh, and H. Bahn, "A Lifespan-aware Reliability Scheme for RAID-based Flash Storage," Proc. ACM Symposium on Applied Computing, pp. 374-379, 2011.

[19] S. Lee, B. Moon, C. Park, J. Kim, and S. Kim, "A Case for Flash Memory SSD in Enterprise Database Applications," Proc. Int'l Conf. on Management of Data (SIGMOD), pp. 1075-1086, 2008.

[20] B. Mao, H. Jiang, D. Feng, S. Wu, J. Chen, L. Zheng, and L. Tian, "HPDA: A Hybrid Parity-based Disk Array for Enhanced Performance and Reliability," Proc. IEEE Int'l Symp. on Parallel & Distributed Processing (IPDPS), pp. 1-12, 2010.

[21] J. Matthews, S. Trika, D. Hensgen, R. Coulson, and K. Grimsrud, "Intel Turbo Memory: Nonvolatile disk caches in the storage hierarchy of mainstream computer systems," ACM Transactions on Storage, Vol. 4, Issue 2, May 2008.

[22] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron, "Migrating Enterprise Storage to SSDs: Analysis of Tradeoffs," Proc. 4th ACM European Conf. on Computer Systems), pp. 145-158, 2009.

[23] K. Park, D-H Lee, Y. Woo, G. Lee, J-H Lee, and D-H Kim, "Reliability and Performance Enhancement Technique for SSD Array Storage System Using RAID Mechanism," Proc. 9th Int'l Conf. on Communications and Information Technologies, pp. 140-145, 2009.

[24] D. A. Patterson, G. Gibson, and R.H. Katz, "A Case for Redundant Arrays for Inexpensive Disks (RAID)," Proc. ACM Int'l Conf. on Management of Data, Vol. 17, No. 3, pp. 109-116, 1988.

[25] D. Perry, "Intel Expected to Drop SSD Prices in August," CNET.NEWS, http://www.tomshardware.com/news/intel-ssd-prices-price-drop-cheap-discount,16276.html, July, 2012.

[26] M. Polte, J. Simsa, and G. Gibson, "Enabling Enterprise Solid State Disks Performance," Proc. Workshop on Integrating Solid-state Memory into the Storage Hierarchy, March, 2009.

[27] Y. Qin, D. Feng, J. Liu, W. Tong, Y. Hu, and Z. Zhu, "A Parity Scheme to Enhance Reliability for SSDs," Proc. 7th Int'l Conf. on Networking, Architecture, and Storage, pp. 293-297, June, 2012.

[28] J. Ren and Q. Yang, "I-CASH: Intelligently Coupled Array of SSD and HDD," Proc. 17th IEEE Int'l Symp. on High Performance Computer Architecture (HPCA), pp. 278-289, 2011.

[29] SPC, "Storage Performance Council I/O traces," http://www.storageperformance.org/.

[30] T.Xie, and J.Koshia. "Boosting Randon Write Performance for Enterprise Flash Storage System," Proc. IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1-10, 2011.