PDB: A Reliability-Driven Data Reconstruction Strategy Based on Popular Data Backup for RAID4 SSD Arrays

Feng Liu[1], Wen Pan[1], Tao Xie[2],Yanyan Gao[1], and Yiming Ouyang[1]

[1] Computer and Information College, Hefei University of Technology, Hefei, P.R. China
[2]Computer Science Department, San Diego State University, San Diego, USA

{fengliu089,wenwen412}@gmail.com, txie@mail.sdsu.edu,
littlek.gao@gmail.com, oyym@hfut.edu.cn

Abstract. With the application of MLC (multi-level cell) and TLC (triple-level cell) techniques, the price of NAND flash memory based SSD (solid-state drive) decreases rapidly with increasing capacity. However, these techniques negatively influence the reliability of SSD as they lead to a larger number of raw flash memory errors. When multiple such reliability degraded SSDs organized in a RAID structure SSD failures could occur. Thus, a reliability-aware data reconstruction mechanism that can quickly recover the data of a failed SSD onto a replacement SSD becomes essential. In this paper, we propose a reliability-driven data reconstruction strategy called PDB (Popular Data Backup) for RAID4 and SPD-RAID4 (Splitting Parity Disk - RAID4), a variant of RAID4. PDB collaboratively backups popular data among data SSDs to achieve a shorter "window of vulnerability". Experimental results demonstrate that compared with the traditional SOR (Stripe Oriented Reconstruction) method PDB can shorten reconstruction time up to 31.3%.

Keywords: SSD, reliability, data reconstruction, popular data backup, window of vulnerability

1 Introduction

In order to decrease the price and increase the capacity of NAND flash memory based SSD (hereafter, SSD) [1], manufacturers are aggressively pushing flash memory into smaller geometries and letting each flash memory cell store multiple bits by employing either MLC (multi-level cell) or TLC (triple-level cell) technique [2]. Unfortunately, these techniques negatively influence the reliability of SSD as they lead to a larger number of raw flash memory errors compared with SLC (single-level cell) technology, in which each cell stores only one bit. As flash memory density increases, it becomes less reliable for it is more subject to various device and circuit level noises as well as retention errors [2]. Besides, a flash memory cell can only be reprogrammed in a limited number of times (called "program/erase cycles"), after which data can no longer be guaranteed to be correctly written into the cell [1,3]. These scaling down techniques also substantially reduce the endurance of flash memory. For

example, the available P/E (program/erase) cycles of MLC NAND flash memory has dropped from ~10K for 5x nm flash to around ~3K for current 2x nm flash [2].

Since a single SSD cannot satisfy the performance and reliability requirements demanded by data-intensive applications like video processing and bioinformatics, an array of SSDs organized in some RAID (Redundant Array of Independent Disk) [4] structures has been proposed to serve such applications [5,6]. However, when individual SSDs tend to be increasingly unreliable, the reliability of an SSD array becomes a severe problem. In particular, when an SSD fails a data reconstruction mechanism must be able to quickly recover its data onto a replacement SSD so that the length of the reconstruction time (or "window of vulnerability") is sufficiently short [7]. A shorter "window of vulnerability" can alleviate performance degradation caused by data recovery. More importantly, it enhances SSD array reliability by lowering the probability of a subsequent SSD failure during an ongoing data reconstruction process [8,9]. It is understood that a second SSD failure during a data reconstruction process could cause permanent data loss, which brings enormous economic loss in industry [8]. For instance, 50 percent of companies that lose critical business systems for more than 10 days never recover [8]. Apparently, a reliability-driven data reconstruction strategy that can shrink the "window of vulnerability" for a RAID structured SSD array is much needed. To the best of our knowledge, very little research about SSD array data reconstruction has been reported in the literature.

A RAID4 (block-level striping with dedicated parity) structured SSD array stores parity information on a dedicated SSD drive (i.e., the parity SSD) and distributes data among multiple data SSDs. It can tolerate one drive failure due to data redundancy. Among various RAID formats, RAID4 has not been popular because the dedicated parity drive becomes a performance bottleneck as parity data must be written to it for every block of non-parity data. Nevertheless, we recently proposed a new variant of RAID4 architecture called SPD-RAID4 (Splitting Parity Disk - RAID4) for parallel SSD arrays [10]. It splits the parity SSD into a configurable number of smaller ones. Thus, multiple small capacity parity SSDs can operate in tandem with the data SSDs to achieve a high performance [10]. For example, SPD-RAID4 turns a standard RAID4 array with five 512 GB SSDs (four data SSDs plus one parity SSD) into a new SSD array with four 512 GB data SSDs and two 256 GB parity SSDs. Note that the total cost of the two SSD arrays is almost the same as at the time of this writing the price of a 256 GB Intel SSD is about half of that of a 512 GB Intel SSD [10]. Experimental results from [10] demonstrate that in terms of mean response time SPD-RAID4 outperforms the widely used RAID5 (block-level striping with distributed parity) by up to 20.3%. As a result, in this paper we propose a reliability-driven online data reconstruction strategy called PDB (Popular Data Backup) for SPD-RAID4.

PDB divides each data SSD into a large user zone and a small mirroring zone. While the user zone serves outside user I/O requests, the mirroring zone of a data SSD backups its immediate neighbor data SSD's popular read data in real-time. Assume that an SPD-RAID4 SSD array has four data SSDs (from left to right: S0, S1, S2, and S3). PDB dynamically replicates S0's most popular read data onto S1's mirroring zone. If S0 fails S1 can speed up the data reconstruction process by dumping the replica of S0's most popular data onto a new replacement SSD. Similarly, S2

backups S1's most popular read data. Lastly, S0's mirroring zone is used to backup S3's most popular read data. PDB makes data SSDs help each other in a circular linked list format. Essentially, it is a data reconstruction strategy based on a collaborative popular data real-time backup scheme. PDB exploits the temporal and spatial locality of workloads and dynamically keeps track of the popularity changes of each data region. Section 3 explains the PDB strategy in details.

To evaluate the performance of PDB, we first largely extend a validated single SSD simulator called SSDsim [11] to an SSD array simulator, which can simulate an SSD array in RAID4, RAID5, and SPD-RAID4 formats. Next, PDB and a conventional data reconstruction mechanism named SOR (Stripe-Oriented-Reconstruction) [12] are implemented into the SSD array controller of the simulator. Finally, we use 3 real-world traces to conduct a comprehensive simulation study. Experimental results show that in terms of reconstruction time on RAID4 and SPD-RAID4, PDB outperforms SOR by up to 20.9% and 31.3%, respectively.

The remainder of this paper is organized as follows. Related work and motivation is presented in the next section. We describe the PDB strategy in section 3. In section 4, we evaluate the performance of SOR and PDB based on real-world traces. Section 5 concludes this paper with a summary.

2 Related Work and Motivation

2.1 SSD Basics

An SSD is a data storage device that uses NAND flash memory to store persistent data [1]. Main parts of an SSD include flash controller, internal cache, and flash memory [1]. Flash controller manages the entire SSD including error correction, interface with flash memory, and servicing host requests [1]. The flash memory part of an SSD consists of multiple identical packages. Each package has multiple dies that share one serial I/O bus and common control signals [1]. Each die contains multiple planes with each having thousands of blocks and one data register as an I/O buffer. Each block has multiple pages (e.g., 64 pages in one block). The common size of a page ranges from 2K to 8K. Flash memory offers three basic operations: program or write, read, and erasure [1]. While reads and writes are page-oriented, erasure can be conducted only at block granularity [11]. Flash memory does not allow in-place updates as a write operation can only change bits from 1 to 0 [2]. On the contrary, an erasure operation changes all bits of a block to 1 and a block must be erased before being programmed (written) [2].

2.2 Existing Data Reconstruction Approaches

When a disk fails, a parity-encoding-based RAID-structured disk array can restore to the normal operating mode by successively rebuilding each block of the failed disk onto a replacement drive while continuing to serve I/O requests from users [13]. This process is called data reconstruction or data recovery, which is normally performed by

a background process activated in either the host or the disk array controller [13]. Existing data reconstruction approaches are all dedicated to HDDs (hard disk drives). They can be generally divided into three categories: (1) reorganizing data layout [14]; (2) optimizing reconstruction workflow [7,8,12,13,15-17]; (3) cache assisted reconstruction [9, 18].

Approaches in the first group improve reconstruction performance by reorganizing the data layout of the replacement disk or parity data units during data recovery [14]. One drawback of these approaches is that changing data layout incurs a high overhead. Mainstream data reconstruction approaches fall in the second category. They can improve disk array reliability and alleviate performance degradation by optimizing reconstruction workflow. SOR (Stripe-Oriented Reconstruction) [12] is one representative approach in this category. SOR creates a set of reconstruction processes associated with stripes so that multiple reconstruction processes can run in parallel.

Since reducing user I/O traffic directed to a degraded RAID set is an effective approach to simultaneously reduce reconstruction time and alleviate user performance degradation, the Workout approach [7] exploits the temporal locality of workloads to reduce user requests during reconstruction. However, its cost is very high as a surrogated RAID set is required to help the degraded disk array. Cache has been widely used in data reconstruction strategies [9,18]. CORE [3] was developed on top of a hybrid disk array where HDDs and SSDs collaborate to optimize reconstruction.

Our PDB strategy concentrates on SSD array data reconstruction by using an approach completely different from the existing ones. By dividing each data SSD into a large user zone and a small mirroring zone, PDB collaboratively backups immediate neighbor data SSD's popular read data in real-time to significantly reduce reconstruction time, and thus, further enhances the reliability of system.

2.3 SPD-RAID4 Scheme

We recently developed a new SSD data reconstruction strategy called SPD-RAID4 [10], which is a variant of a standard SSD RAID4 structure. To the best of our knowledge, it is the first data reconstruction approach devoted to an SSD array.

SPD-RAID4 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. In a standard SSD RAID4 array, only one parity SSD undertakes all parity updates, which makes it wear out quickly. This problem can be largely solved in SPD-RAID4 because multiple parity SSDs evenly receive parity updates. In addition, when a request spans across two or more stripes, the parity SSDs can work in parallel, and thus, significantly boosts the SSD array performance. If a data SSD fails, multiple parity SSDs can serve requests in parallel when recovering data in the fault data SSD. Experimental results demonstrate that the performance of SPD-RAID4 is better than that of SSD RAID5 [10]. The scope of this research is to develop a reliability-driven data construction strategy for SPD-RAID4.

2.4 Workload Locality

In many applications, 80% accesses are directed to 20% of the data, a phenomenon that has long been known as Pareto's Principle or "The 80/20 Rule" [19]. It indicates the existence of temporal locality and spatial locality in various workloads. Temporal locality, on the time dimension, refers to the repeated accesses to specific data blocks within relatively small time durations. Spatial locality, on the space dimension, refers to the clustered accesses to data objects within small regions of storage locations within a short timeframe. Previous studies observe that 10% of files accessed on a web server approximately account for 90% of the requests and 90% of the bytes transferred [8]. Such studies also found that 20%-40% of the files are accessed only once for web workloads [8]. The two types of localities have been frequently exploited to boost system performance. PDB also exploits the two access localities. It dynamically tracks each data SSD's popular read data. And then each data SSD backups the most popular read data of its immediate neighbor data SSD in real-time. Although the size of popular data is small, it can serve a large number of user requests.

2.5 Motivation

Modern large capacity SSDs become less reliable due to a spectrum of aggressive scaling down techniques. Meanwhile, RAID-structured SSD arrays are replacing traditional HDD based disk arrays in various data-intensive applications. Thus, the reliability of SSD arrays becomes a critical issue. Especially, when one SSD fails in an SSD array, a reliability-aware data reconstruction approach is desperately needed. Unfortunately, to the best of our knowledge, little research has been done in SSD array data recovery. Motivated by our observations on the facts mentioned above and the insights on workload locality characteristics provided by other researchers, in this paper we propose a new reliability-driven data reconstruction strategy PDB to enhance RAID-structured SSD arrays' reliability during data recovery. PDB achieves reliability enhancement during reconstruction with a minimum performance penalty.

3 The PDB Strategy

3.1 Architecture Overview

Fig. 1 shows the architecture of the PDB strategy on an SPD-RAID4 structured SSD array with 4 identical data SSDs (i.e., SSD0, SSD1, SSD2, SSD3) and 2 parity SSDs (i.e., SSD4, SSD5). The capacity of each parity SSD is a half of a data SSD. SSD0 is in shadow, which indicates that it becomes a failed SSD after running for a while (see Fig. 1). Each data SSD's popular read data is replicated in real-time onto its corresponding immediate neighbor data SSD's mirroring zone in normal mode. For simplicity, the immediate neighbor SSD is named as a buddy SSD. For example, SSD1 is the buddy SSD of SSD0 and SSD0 is the buddy SSD of SSD3 (see Fig. 1). When SSD0 suddenly fails PDB does not need to reconstruct its popular data as it has been stored on SSD1. Rather, PDB simply dumps it to a new replacement SSD, which can

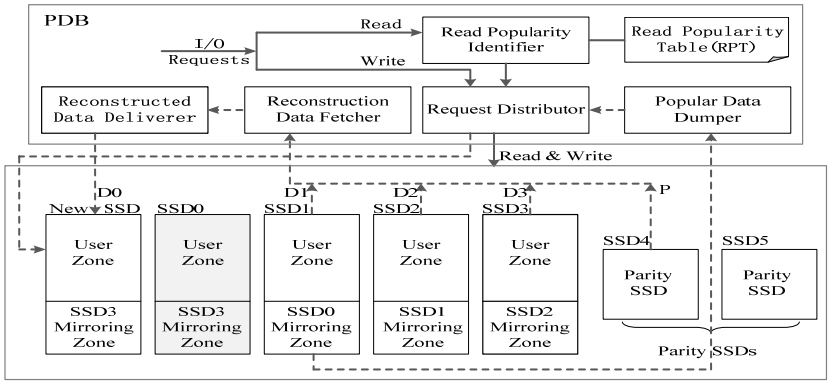

Fig. 1. Architecture overview of PDB

save data reconstruction time. Note that the requests that target on the popular data of SSD0 during the data recovery period can be served by SSD1. Obviously, when all SSDs are working correctly, the burden of each data SSD increases because each buddy SSD needs to backup its sponsored data SSD's popular data in real-time. Thus, PDB causes performance degradation when all SSDs are fine. Our experimental results presented in Section 4.2 show that compared with SOR the performance degradation of PDB in normal mode is no larger than 6.8% in terms of mean response time. However, PDB shrinks data reconstruction time by up to 31.3%. We argue that it is worthwhile to trade a slight performance degradation for a substantial improvement in data reconstruction time because PDB noticeably improves SSD array reliability during a data recovery process. The shorter a "window of vulnerability" is, the more reliable an SSD array is.

PDB consists of five key modules: read popularity identifier (RPI), request distributor (RD), popular data dumper (PDD), reconstruction data fetcher (RDF), and reconstructed data deliverer (RDD). The RPI module is responsible for identifying the popular data based on the recent access times of incoming user read requests in normal mode. The RD module directs I/O requests into either a user zone or a user zone and its corresponding mirroring zone. PDD dumps the popular data from the buddy SSD's mirroring zone to a new replacement SSD. Multiple RDFs are launched during reconstruction. Each of them reads one data block or a parity block from a surviving data SSD or a parity SSD. Next, the rebuilt data block D0 can be computed by D0=XOR(D1, D2, D3, P). Finally, RDD writes the reconstructed data D0 onto the replacement SSD.

In the normal mode, when a read request arrives PDB first identifies whether its associated data is popular by consulting the Read Popularity Table (RPT) (see Fig. 1). If it is popular, PDB backups it onto the mirroring zone of the corresponding buddy SSD once it cannot be found in the mirroring zone. If it is unpopular, PDB directly reads it from the right data SSD. Upon receiving a write request, PDB first checks whether it resides in the mirroring zone of the corresponding buddy SSD. If it does, PDB writes it into the right SSD and the mirroring zone of the corresponding buddy SSD simultaneously through the RD module. If not, PDB only writes it onto the right

data SSD. When an SSD fails, the entire SSD array enters into the recovery mode. In this mode, when a read request arrives, if it targets on the failed SSD and the data has not been rebuilt onto the replacement SSD, PDB checks whether the data has been stored in the mirroring zone of the failed SSD's buddy SSD. If yes, PDB will directly read it from there, and then, PDD dumps it to the replacement SSD (see Fig. 1). If not, PDB will launch multiple RDF processes to fetch data from surviving SSDs. For example, in Fig. 1 data blocks D1, D2, D3 and the corresponding parity block P are fetched by 4 RDF processes in parallel from SSD1, SSD2, SSD3, and SSD4, respectively. Note that the 3 data blocks and the parity block P belong to one stripe. After conducting an XOR operation (see Fig. 1), the reconstructed data block D0 will be delivered to the replacement SSD by the RDD module. The replacement SSD now can serve the read request. When a write request that targets on the failed SSD comes, it will be re-directed to the new replacement SSD.

3.2 Key Data Structures and Algorithm

PDB relies on two key data structures RPT and RL (restore list) to identify popular read data and replicate it into corresponding buddy SSD's mirroring zone. Fig. 2 demonstrates their implementation details and the algorithm of restore function is shown as below. PDB evenly divides each user zone into multiple non-overlapping but consecutive logical data regions. Each node in RL represents a data region and keeps track of region information that will be replicated to corresponding buddy SSD. The closer the node towards the head of RL, the more popular it is. The size of each region is equal to the size of a flash memory block (e.g., one flash memory flock has 64 pages and each page is 2 KB). RPT keeps track of the popularity of all logical data regions by using variables: *id* and *visit_count*. The *id* of an incoming read request is calculated by the equation below:

$$id = \text{int}\,(LPN/block\_size), \tag{1}$$

where *LPN* is the logic page number of a request and *block_size* is set to 64, which is the number of pages in a flash block. Equation (1) indicates that an *id* contains a block size data region. For example, request *LPN* 0 to *LPN* 63 all belong to *id* 0. The value of *visit_count* represents the popularity of a data region. Its value is incremented by 1 when a read request hits the data region. If the value of *visit_count* is equal to or greater than the popularity threshold, PDB takes the corresponding data region as a popular region. When the RPT table is full, the least popular data region with the fewest access times will be kicked out to accommodate a new popular data region. Then PDB inserts the corresponding node into RL and write the data in popular region into mirroring zone of indicated buddy SSD.

Data consistency in PDB includes two aspects: (1) The key data structures must be safely stored; (2) The real-time backup data in buddy SSD must be updated timely to guarantee data consistency. Firstly, to prevent the loss of key data structures in the event of a power supply failure or a system crash, PDB stores them in a non-volatile RAM (NVRAM). Since the size of RPT is generally small (1,024 entries with each entry 4 bytes in our experiments), it will not incur significant extra hardware cost.

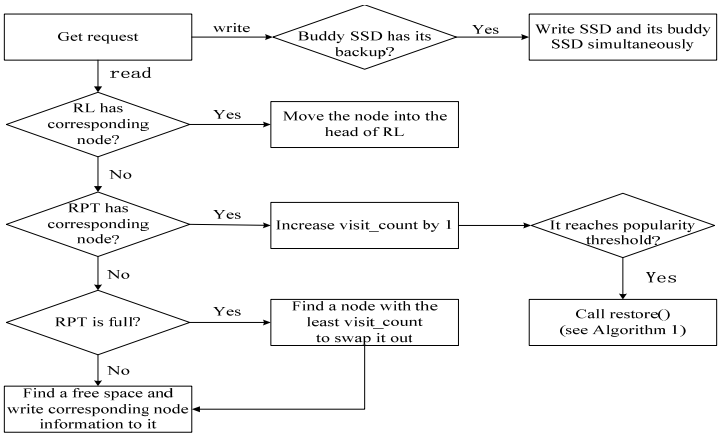

Fig. 2. Workflow of PDB in normal mode

Algorithm 1

restore()
Write data in indicated zone into corresponding buddy SSD and create a node
if restore list does not reach the maximum length **then**
    Directly insert the node into the head of restore list
else
    Delete the node in the queue tail and invalidate data of tail node
end if

Secondly, the popular read data must be safely stored in corresponding buddy SSD. To ensure the popular data recovered to replacement SSD is not out-of-date, when the corresponding buddy SSD exists identical LPN of a write operation, the backup data must be updated simultaneously to make it is always up to date.

3.3 Implementation

The original SSDsim [11] can only simulate a single SSD. We significantly extend it so that SSD RAID4 and SPD-RAID4 are also supported. On this basis, we also implement a baseline reconstruction mechanism SOR and our proposed strategy PDB on both RAID4 and SPD-RAID4. The size of a flash page is set to 2 KB in our experiments. When updating the parity data on the parity sub array, we also employ the round-robin way to write parity data evenly across all parity SSDs.

Once the host sends a request, the RAID controller gets the device number and its 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 (flash translation layer) implemented inside each SSD. When a read request comes, PDB identifies the popularity of corresponding logical data region. Once it is taken as a popular page (we take a page in a popular data region as a popular page), PDB redirects the data to mirroring zone of corresponding buddy SSD while it returns to host at the same time if it has not been

in buddy SSD. If the read operation is directed to the failed SSD, PDB checks that whether the indicated data has been stored in corresponding buddy SSD. If it has, PDB directly reads it from corresponding buddy SSD and dumps data in indicated data region to the replacement SDD. Otherwise, RDF and RDD are launched. When the popular data has been successfully dumped, the system can directly respond the subsequent identical requests. Although the size of popular data is small, it can serve a large number of user requests, and thus, significantly reduces reconstruction time.

4 Performance Evaluation

4.1 Experimental Setup

Among existing data reconstruction schemes, SOR is a widely used data recovery mechanism with the lowest overhead. Therefore, we implemented SOR and PDB on both SSD RAID 4 and SPD-RAID4 structures. Our simulator is built based on a validated SSD simulator called SSDsim [11], which is an event-driven, modularly structured, and highly accurate simulator for a single SSD. We added about 2,700 lines of C codes to extend SSDsim to an SSD array simulator and implement the two data reconstruction schemes. The RAID controller fetches a request from a trace file and splits it into multiple sub-requests.

We use three real-world traces [20] to compare the performance of SOR and PDB. The three traces and their characteristics are summarized in Table 1. The three Web-search (hereafter, Web) traces were collected from a machine running a web search engine. The read-dominated Web trace exhibits a strong locality.

The number of data SSDs and parity SSDs are both configurable for SPD-RAID4. We conducted our performance evaluation of the two strategies on SPD-RAID4 with 2 parity SSDs as we found that 2 is the optimal choice of the number of parity SSDs. The default number of data SSDs is 4. The number of RAID4's data SSD is also set to 4 and its parity SSD's capacity is twice of a parity SSD in SPD-RAID4. Due to the limited footprints of traces, the capacity of each data SSD is set to 16 GB in our experiments and the capacity of each parity SSDs in SPD-RAID4 is 8 GB. The capacity of the only parity SSD in RAID4 is 16 GB as well.

4.2 Real-World Trace Experimental Results

In this section, we evaluate the performance of PDB by comparing it with a classical data reconstruction method SOR on both RAID4 and SPD-RAID4 structured SSD

Table 1. Real-world traces characterstics

Trace Name	Read Ratio (%)	Avg. Size (KB)	Intensity (reqs./s)	Duration (minute)
Web 1	99.98	15.14	335	52.5
Web 2	99.97	15.07	297	256.6
Web 3	99.98	15.41	16	4543.9

arrays. Fig. 3 shows the read, write and overall mean response times of PDB and SOR before data reconstruction in the normal mode. The marks "_R", "_W", and "_O" represent read, write and overall mean response time, respectively. For PDB, the logical data region size is set to 128 KB, which is equal to a flash block size. The goal of this group of experiments is to measure PDB's performance degradation during the normal mode before an SSD failure happens.

From Fig. 3, we can see that the performance of PDB on both RAID4 and SPD-RAID4 is consistently and slightly worse than that of SOR before reconstruction in the three traces. Compared with SOR, PDB degrades performance in terms of overall mean response time on SPD-RAID4 by 4.7%, 3.8% and 3.1% on the three traces, respectively (see Fig. 3b). Note that the maximum overall performance loss is only 4.7% in Web1 trace case on both RAID4 and SPD-RAID4 structures. The reason for PDB's performance degradation is that it has extra work to do comparing with SOR. In particular, PDB needs to dynamically keep track of popularity changes of read requests and stores popular data in the mirroring zone in real-time, which inevitably increase its burden, and thus, enlarges the mean response time of user requests.

An SSD array enters the data recovery mode after one data SSD fails. To understand the performance and reliability enhancement of PDB during reconstruction, we

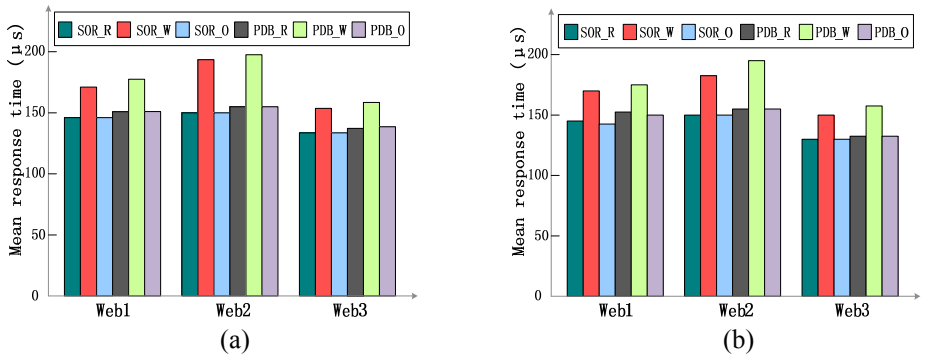

Fig. 3. Performance comparisons before reconstruction on (a) RAID4; (b) SPD-RAID4

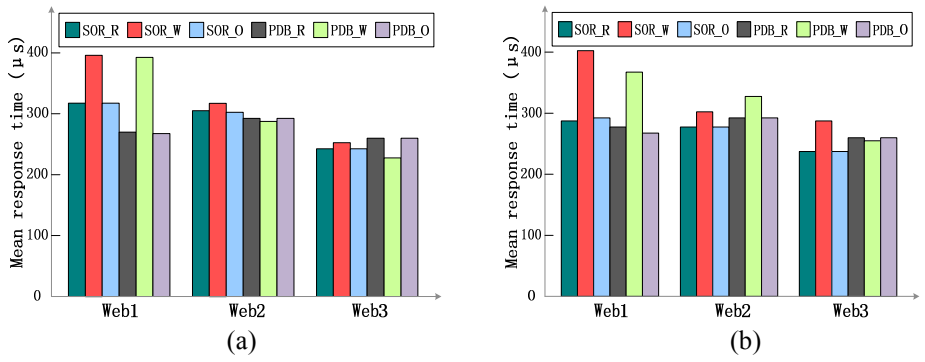

Fig. 4. Performance comparison during reconstruction on (a) RAID4; (b) SPD-RAID4

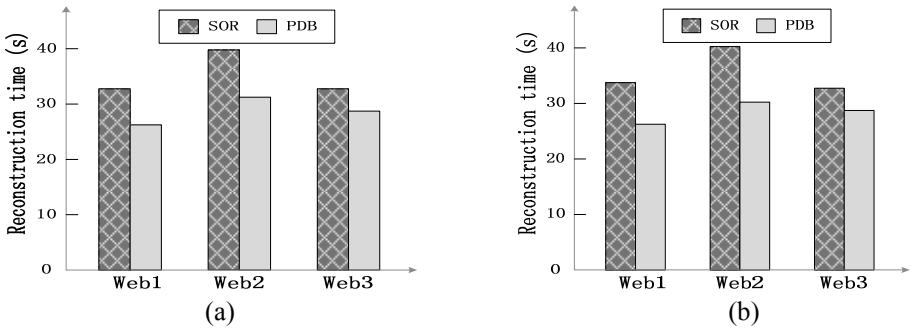

Fig. 5. Reconstruction time on (a) RAID4; (b) SPD-RAID4

measure mean response time during reconstruction and reconstruction time in a group of experiments whose results are demonstrated in Fig. 4 and Fig. 5. From Fig. 4a, we can clearly see that in the data recovery mode PDB even performs better in terms of mean response time during reconstruction than SOR in RAID4 format on Web1 and Web2 traces by 13.9% and 2.9%, respectively. It only performs a little bit worse in Web3 trace. However, on SPD-RAID4 structure, PDB only performs better on Web1 trace (see Fig. 4b). Still, on average PDB's performance degradation in data recovery mode is only 2.1% compared with SOR. The almost negligible performance degradation during reconstruction is because that when the reconstruction time decreases (see Fig. 5) the number of reconstruction requests per time unit obviously enlarges, which prolongs the mean response time of user requests. Therefore, the performance of PDB during reconstruction is lowered down.

The reconstruction times of SOR and PDB on RAID4 and SPD-RAID4 are illustrated in Fig. 5. It shows that on both RAID4 and SPD-RAID4 structures the reconstruction time of PDB is consistently less than that of SOR. In case of SPD-RAID4, PDB shrinks reconstruction time by 21.8%, 22% and 13.3% on the three traces, respectively (see Fig. 5b). PDB manifests a similar improvement in reconstruction time in RAID4 scenario (see Fig. 5a). Note that the much shorter reconstruction time shown in Fig. 5 compared with HDD array situations comes from two facts: (1) SSD has a much faster read and write speed than HDD [1]; (2) the footprint of the three traces is relatively small, and thus, the amount of data need to be rebuilt is not large. PDB's improvement in terms of reconstruction time stems from its ability to directly respond popular read requests targeting on the failed SSD from the mirroring zone of its buddy SSD. In this way PDB does not need to launch a standard reconstruction mechanism like SOR does. In addition, all three workloads are read-dominant and have strong locality, which can be effectively exploited by PDB to substantially reduce reconstruction time. Although the stored popular data in a buddy SSD only takes a small percentage of total data amount, a large percentage of requests may access them consecutively due to workload locality explained in Section 2.4. Therefore, PDB can significantly reduce reconstruction time to shrink the "window of vulnerability", and thus, can enhance SSD array reliability. We argue that scarifying slightly in performance in the normal mode to obviously shrink reconstruction time is a good trade-

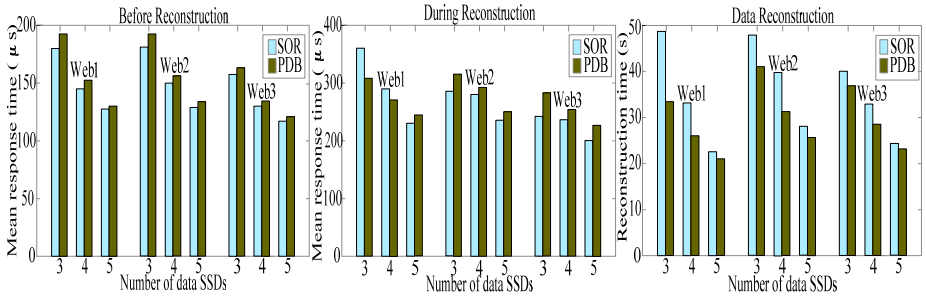

Fig. 6. The impacts of the number of data SSDs on performance

off for modern SSD arrays organized in a RAID structure.

The purpose of our last group of experiments is to study the impacts of the number of data SSDs in an SPD-RAID4 SSD array on the performance of PDB. Several interesting observations can be made from the results shown in Fig. 6. First of all, for all 3 traces, we can see that the performance of both SOR and PDB before reconstruction (i.e., in the normal mode) increases when the number of data SSDs is enlarged. It is easy to understand that with more data SSDs the SSD array can process user requests faster. Next, SOR consistently outperforms the PDB strategy on the three traces when the number of data SSDs is increasing. However, the performance gap between the two strategies shrinks with an increasing number of data SSDs (see Fig. 6), which is true for all 3 traces. Take the Web1 trace for example, the performance difference between the two reduces from 6.8% to 2.9%. The small performance degradation stems from the fact that PDB needs to backup the popular data in normal mode, and thus, enlarges its mean response time. The larger the number of data SSDs is, the less performance degradation PDB has. The rationale behind is that with more data SSDs and a fixed amount of total workloads each data SSD needs to replicate less popular data for its sponsored data SSD.

Furthermore, we can see that in the data recovery mode, PDB only performs better when the number of data SSDs is 3 or 4 in Web1 trace. PDB's worse performance during reconstruction is because it gives data reconstruction task a higher priority than processing user requests during reconstruction. Finally, for all 3 workloads and all data SSD numbers, the PDB strategy consistently outperforms the SOR mechanism in reconstruction time (see Fig. 6). When the number of data SSDs increases, the reconstruction times for both SOR and PDB decrease. The reason for the decreased reconstruction time on a larger size SSD array is that fewer user requests arrive on an individual data SSD in the recovery mode, which in turn reduces the reconstruction time. In particular, in Web1 case PDB can shrink the reconstruction time by up to 31.3% when there are 3 data SSDs in a SPD-RAID4 SSD array (see Fig. 6). However, the reconstruction time decreases the most when the number of data SSDs is 4 in the case of Web2 and Web3 traces. It is clear that SOR also lowers down its reconstruction time when the number of data SSDs increases from 3 to 5. More importantly, in Web1 trace, its improvement in terms of reconstruction time becomes more noticeable than that of PDB situation when the number of data SSDs enlarges from 3 to 5. The reason is that the amount of data that need to be rebuilt by SOR reduces faster than PDB.

5 Conclusions

With larger and more affordable yet less reliable SSDs, developing an efficient data reconstruction strategy with reliability-awareness for emerging SSD arrays organized in some RAID structures becomes a critical problem to be solved. Unfortunately, very little research about SSD array data reconstruction has been reported in the literature. To the best of knowledge, this research is the first step towards solving the critical issue. In this paper, we develop and evaluate a novel online data reconstruction strategy called PDB for both conventional RAID4 and SPD-RAID4. The PDB strategy exploits the workload locality, which has been frequently observed in a spectrum of real-world workloads like the three web search traces. It utilizes a collaborative popular data backup mechanism among all data SSDs to largely shrink the "window of vulnerability", and thus, enhances SSD array reliability. Our Experimental results demonstrate that compared with a traditional reconstruction method SOR the PDB strategy can further shorten reconstruction time by up to 31.3% on SPD-RAID4.

PDB, however, in its current format, only applies the popular data backup scheme among all data SSDs. When a parity SSD fails, it still uses a conventional reconstruction method, which is currently used by SOR. The main reason is that parity data generally does not exhibit obvious locality, which makes PDB inefficient. Considering that normally the number of data SSDs is larger than that of parity SSDs, the probability of a data SSD failure is higher than a parity SSD failure. Thus, PDB could enhance SSD array reliability in majority cases. Also, PDB may not work well under write-dominated workloads. One direction of the future of this research is to extend it to incorporate write-dominated. Finally, we only integrate PDB into RAID4 and SPD-RAID4 SSD arrays. In our future work, we are going to integrate it into different RAID architectures such as RAID5.

Acknowledgments

This work is sponsored in part by the U.S. National Science Foundation under grant CNS-(CAREER)-0845105 and Key Technologies R & D Program of Anhui Province (China)-11010202190.

References

1. Agrawal, N., Prabhakaran, V., Wobber, T., Davis, J.D., Manasse, M., Panigrahy, R.: Design Tradeoffs for SSD Performance. In: USENIX Ann. Technical Conference, pp. 57-70. USENIX Association, Berkeley (2008)
2. Cai, Y., Haratsch, E.F., Mutlu, O., Mai, K.: Threshold Voltage Distribution in MLC NAND Flash Memory: Characteristization, Analysis, and Modeling. In: the conf. on Design, Automation and Test in Europe, pp. 1285-1290. EDA Consortium, San Jose (2013)
3. Xie, T., Sharma, A.: Collaboration-Oriented Data Recovery for Mobile Disk Arrays. In: 29th Int'l Conf. on Distributed Computed Systems, pp. 631-638. Montreal (June 2009)

4. Patterson, D.A., Gibson, G., Katz, R.H.: A Case for Redundant Arrays for Inexpensive Disks (RAID). In: Boral, H., Larson, P.A. (eds) 1988 ACM SIGMOD Int'l Conf. on Management of Data, pp. 109-116. ACM, New York (1988)
5. Im, S., Shin, D.: Flash-Aware RAID Techniques for Dependable and High-Performance Flash Memory SSD. J. IEEE Transaction on Computer. 6(1), 80-92 (2011)
6. Lee, Y., Jung, S., Song, Y.H.: FRA: A Flash-aware Redundant Array of Flash Storage Devices. In: 7th IEEE/ACM Int'l Conf. on Hardware/Software Codesign and System Synthesis, pp. 163-172. ACM, New York (2009)
7. Wu, S.Z., Jiang, H., Feng, D., Tian, L., Mao, B.: Workout: I/O Workload Outsourcing for Boosting RAID Reconstruction Performance. In: 7th USENIX Conf. on FAST, pp. 239-252. USENIX Association, Berkeley (2009)
8. Tian, L., Feng, D., Jiang, H., Zhou, K., Zeng, L.F., Chen, J.X., Wang, Z.K., Song, Z.L.: PRO: A Popularity-Based Multi-Threaded Reconstruction Optimization for RAID-Structured Storage Systems. In: 5th USENIX Conf. on FAST, pp. 277-290. USENIX Association, Berkeley (2007)
9. Xie, T., Wang, H.: MICRO: A Multilevel Caching-Based Reconstruction Optimization for Mobile Storage Systems. J. IEEE Transactions on Computers, 57(10), 1386-1398 (2008)
10. Pan, W., Liu, F., Xie, T., Gao, Y.Y., Ouyang, Y.M., Chen, T.: SPD-RAID4: Splitting Parity Disk for RAID4 Structured Parallel SSD Arrays. In: 15th Int'l Conf. on High Performance Computing and Communications, IEEE Press, Zhangjiajie, China (Nov. 2013)
11. Hu, Y., Jiang, H., Feng, D., Tian, L., Luo, H., Zhang, S.P.: Performance Impact and Interplay of SSD Parallelism through Advanced Commands, Allocation Strategy and Data Granularity. In: Int'l Conf. on Supercomputing, pp. 96-107. ACM, New York (2011)
12. Holland, M.: On-line Data Reconstruction in Redundant Disk Arrays. In: PhD dissertation CMU-CS-94-164, Carnegie Mellon Univ., Pittsburgh (1994)
13. Holland, M., Gibson, G.A., Siewiorek, D.P.: Fast, On-Line Failure Recovery in Redundant Disk Arrays. In: 23rd Ann. Int'l Symp. on Fault-Tolerant Computing, pp. 422-443. IEEE Press, Toulouse, France (1993)
14. Hou, R.Y., Menon, J., Patt, Y.N.: Balancing I/O Response Time and Disk Rebuild Time in a RAID5 Disk Array. In: 26th Hawaii Int'l Conf. on Systems Sciences, pp. 70-79. IEEE Press, Hawaii (1993)
15. Lee, J.Y.B., Lui, J.C.S.: Automatic Recovery from Disk Failure in Continuous-Media Servers. J. IEEE Transaction on Parallel and Distributed Systems. 13(5), 499-515 (2002)
16. Wu, S.Z., Feng, D., Jiang, H., Mao, B., Zeng, L.F., Chen, J.: JOR: A Journal-Guided Reconstruction Optimization for RAID Structured Storage Systems. In: 15th Int'l Conf. on Parallel and Distributed Systems, pp. 609-616. IEEE Press, Shenzhen (Dec. 2009)
17. Wu, S.Z., Jiang, H., Mao, B.: IDO: Intelligent Data Outsourcing with Improved RAID Reconstruction Performance in Large-Scale Data Centers. In: 26th Int'l Conf. on Large Installation System Administration, pp. 17-32. USENIX Association, San Diego (2012)
18. Wan, S., Cao, Q., Huang, J.Z., Li, S.Y., Li, X., Zhan, S.H., Yu, L., Xie, C.S., He, X.B.: Victim Disk First: An Asymmetric Cache to Boost the Performance of Disk Arrays Under Faulty Conditions. In: USENIX Annual Technical Conference. USENIX Association, Berkeley (2011)
19. Gomez, M.E., Sontonja, V.: Characterizing Temporal Locality in I/O Workload. In: Int'l Symp. on Performance Evaluation of Computer and Telecommunication Systems, San Diego (July 2002)
20. SPC, Storage Performance Council I/O traces, http://traces.cs.umass.edu/index.php/Storage/Storage