LDM: Log Disk Mirroring with Improved Performance and Reliability for SSD-Based Disk Arrays

SUZHEN WU, BO MAO, and XIAOLAN CHEN, Xiamen University, China
HONG JIANG, University of Texas at Arlington

With the explosive growth in data volume, the I/O bottleneck has become an increasingly daunting challenge for big data analytics. Economic forces, driven by the desire to introduce flash-based Solid-State Drives (SSDs) into the high-end storage market, have resulted in hybrid storage systems in the cloud. However, a single flash-based SSD cannot satisfy the performance, reliability, and capacity requirements of enterprise or HPC storage systems in the cloud. While an array of SSDs organized in a RAID structure, such as RAID5, provides the potential for high storage capacity and bandwidth, reliability and performance problems will likely result from the parity update operations. In this article, we propose a Log Disk Mirroring scheme (LDM) to improve the performance and reliability of SSD-based disk arrays. LDM is a hybrid disk array architecture that consists of several SSDs and two hard disk drives (HDDs). In an LDM array, the two HDDs are mirrored as a write buffer that temporally absorbs the small write requests. The small and random write data are written on the mirroring buffer by using the logging technique that sequentially appends new data. The small write data are merged and destaged to the SSD-based disk array during the system idle periods. Our prototype implementation of the LDM array and the performance evaluations show that the LDM array significantly outperforms the pure SSD-based disk arrays by a factor of 20.4 on average, and outperforms HPDA by a factor of 5.0 on average. The reliability analysis shows that the MTTDL of the LDM array is 2.7 times and 1.7 times better than that of pure SSD-based disk arrays and HPDA disk arrays.

Categories and Subject Descriptors: D.4.2 [**Operating Systems**]: Storage Management; D.4.8 [**Operating Systems**]: Performance

General Terms: Design, Performance

Additional Key Words and Phrases: SSD-based disk arrays, log technique, disk buffer, performance evaluation, reliability analysis

ACM Reference Format:
Suzhen Wu, Bo Mao, Xiaolan Chen, and Hong Jiang. 2016. LDM: Log disk mirroring with improved performance and reliability for SSD-based disk arrays. ACM Trans. Storage 12, 4, Article 22 (May 2016), 21 pages.
DOI: http://dx.doi.org/10.1145/2892639

This work is supported by the National Natural Science Foundation of China under Grant No. 61100033, No. 61472336, and No. 61402385; the U.S. NSF under Grant No. NSF-CNS-1116606 and No. NSF-CNS-1016609; and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry. This work is also sponsored by the Fundamental Research Funds for the Central Universities (No. 20720140515).
Authors' addresses: S. Wu and X. Chen, Computer Science Department of Xiamen University, Xiamen 361005, China; emails: suzhen@xmu.edu.cn, xiaolchen@qq.com; B. Mao (corresponding author), Software School of Xiamen University, Xiamen 361005, China; email: maobo@xmu.edu.cn; H. Jiang, Department of Computer Science & Engineering, University of Texas at Arlington, Arlington 76010, TX, USA; email: hong.jiang@uta.edu.

1. INTRODUCTION

In today's large-scale storage systems, two storage devices coexist: Hard Disk Drives (HDDs) and flash-based Solid-State Drivers (SSDs). The HDDs are mechanical devices that require head seeks and plate rotations to locate the requested data; thus, the data access latency is very high. With the rapid development of high-performance processors, the HDD-based storage system has become a performance bottleneck of the overall computer system. Thus, the emergent flash-based SSDs have received a great deal of attention from both academia and industry [Agrawal et al. 2008; Chen et al. 2009; Dirik and Jacob 2009]. In contrast, the flash-based SSDs are based on semiconductor chips with no mechanical parts, thus having many advantages over HDDs, such as low power consumption and high small-random-read performance. Besides the deployment of SSDs on mobile devices and desktop and laptop PCs, many have also proposed to use SSDs in the high-performance computing and enterprise environments [Narayanan et al. 2009; Mao and Wu 2015].

However, due to the flash characteristics, SSDs also have shortcomings, such as the low small-random-write performance and limited lifetime. Moreover, the cost/GB of SSDs is much higher than that of HDDs. These limitations, along with the high capacity and low cost advantages of HDDs, have made these two storage devices coexist in today's large-scale storage systems. Many creative ideas have been proposed for HDD-based storage systems, such as RAID (Redundant Array of Independent Disks) [Patterson et al. 1988], one of the most important techniques that has become a standard in storage systems. With the wide deployment of SSDs in enterprise and HPC environments, applying the RAID technique to SSDs is also a necessary and likely promising approach to building large-scale high-performance and highly reliable SSD-based storage systems [Balakrishnan et al. 2010a; Mao et al. 2010; Meza et al. 2015]. For brevity, throughout this article, *RAIS* stands for Redundant Array of Independent SSDs, along with different levels of RAIS being denoted by RAIS0, RAIS5, and so on.

Among these different RAIS levels, previous studies have demonstrated that RAIS5 is a better solution for SSD-based disk arrays [Balakrishnan et al. 2010a]. However, the inherent parity update problem will not only degrade performance but also affect the reliability of RAIS5 due to the write amplification problem [Moon and Reddy 2013]. Moreover, a recent study has revealed that updates in storage traces are frequent and small [Chan et al. 2014], thus making the parity update problem in parity- and SSD-based disk arrays an urgent issue to address. Our previous study HPDA [Mao et al. 2010] aims to address the parity update problem by replacing the parity SSD with an HDD in a RAIS4 system. Because the parity SSD is replaced by an HDD, the flash wearout and erase-before-write problems caused by the parity-update operations can be avoided. However, HPDA is only applicable to RAIS4 systems, and its parity HDD becomes a performance bottleneck under write-intensive workloads.

To address this problem, in this article, we propose an SSD-HDD hybrid disk array architecture, called Log Disk Mirroring (LDM), which combines an array of SSDs with two HDDs to improve the performance and reliability of SSD-based storage systems. In LDM, the SSDs (data disks) compose a RAIS5 disk array. The two HDDs are mirrored as a write buffer that temporarily absorbs small write requests and acts as a surrogate RAID1 set [Wu et al. 2011] during recovery when an SSD fails. The write data is reclaimed to the data disks during system idle periods. Since the small updates are temporarily and sequentially stored on the two HDDs, the write amplification and flash wearout problems on SSDs caused by the parity-update operations are avoided. Moreover, the mirroring buffer improves the small-random-write performance using a log-structured write scheme [Chan et al. 2014]. Our prototype implementation of the LDM array and the performance evaluations show that the LDM array significantly outperforms pure SSD-based disk arrays by a factor of 20.4 on average, and outperforms HPDA [Mao et al. 2012] by a factor of 5.0 on average. The reliability analysis shows that the MTTDL of the LDM array is 2.7 times and 1.7 times better than that of pure SSD-based and HPDA disk arrays, respectively.

The rest of this article is organized as follows. Background and motivation are presented in Section 2. We describe the design details of LDM in Section 3. The performance evaluation is presented in Section 4, and reliability analysis is presented in Section 5. We review the related work in Section 6, and discussion appears in Section 7. We conclude this article in Section 8.

2. BACKGROUND AND MOTIVATION

In this section, we describe the key characteristics of flash-based SSDs in contrast to those of magnetic HDDs. Then we elaborate on how the parity update process affects the performance and reliability of RAIS5. These observations motivate our proposed new hybrid disk array architecture for RAIS.

2.1. Characteristics of Flash-Based SSD

Like HDDs, data in SSDs remain 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). Besides of the high energy efficiency and high cost-per-GB advantages of flash-based SSDs, they have the following two main unique characteristics different from HDDs.

First, flash-based SSDs suffer from the poor performance of small-random-write requests. The reason is that, in the flash storage, each 64-128KB flash block must be erased in advance before any part of it can be rewritten, which is a characteristic feature of SSD known as "erase-before-write." Due to the sheer size of a block, an erase operation typically takes milliseconds to complete, one or two orders of magnitude slower than the read operation. Previous studies have shown that random writes can cause the data pages in NAND flash blocks to be copied elsewhere and erased, thus leading to internal fragmentation of SSDs and performance degradation by an order of magnitude [Agrawal et al. 2008; Chen et al. 2009; Min et al. 2012]. Many sophisticated FTL algorithms and embedded buffer management methods have been proposed to alleviate the flash small-random-write problem. According to the test specification drafted by SNIA SSSI [Solid State Storage Initiative 2010], the small-random-write performance of current flash-based SSDs is not stable and will drop after a varied amount of time depending on the venders. More importantly, small-random-write requests to SSDs will aggravate the flash wearout problem and reduce their lifetime accordingly.

Second, the flash wearout problem after repeated write-erase cycles impacts the reliability of SSDs. Generally, the expected number of erasures per block is 100,000 for the single-level cell (SLC) NAND flash memory, and it reduces to 10,000 for the multilevel cell (MLC) NAND flash memory. In recent years, the MLC or TLC (triple-level cell) NAND flash memory has been developed as an effective medium to increase the storage density and reduce the cost of flash-based SSDs. However, compared with the SLC-based flash, the MLC- and TLC-based flash has lower performance and less endurance [Grupp et al. 2012]. Thus, the reliability problem is still a critical issue when designing large-scale SSD-based storage systems using MLC- or TLC-based NAND flash memory.

Given their significant performance and reliability implications, these two limitations of flash-based SSDs must be taken into serious consideration when designing SSD-based storage systems, especially SSD-based disk arrays.

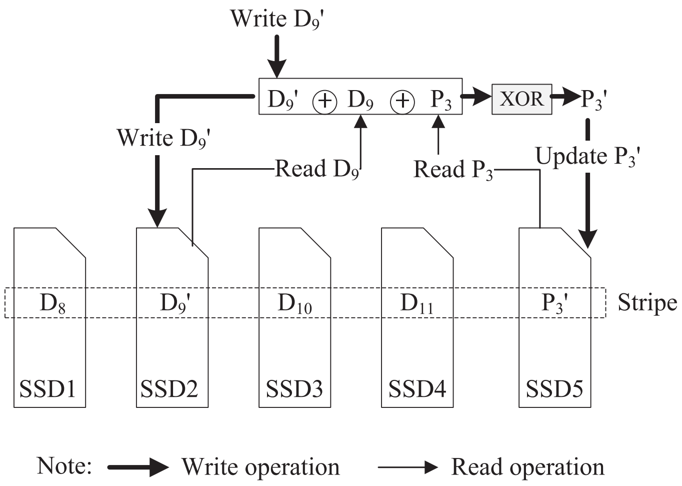


Fig. 1. The parity update process in RAIS5 consisting of five SSDs. Each small write to a data block, say, D_9, entails a total of two SSD reads (for old data block D_9 and old parity block P_3) and two SSD updates (writing new data block D_9' and new parity block P_3').

2.2. Parity Update Problem in RAIS5

In face of the high cost/GB of the current generation of SSDs and the reliability requirements of SSD-based disk arrays, RAID5 is arguably the best choice for SSD-based disk arrays [Balakrishnan et al. 2010a; Mao et al. 2010]. However, in order to apply RAID5 to SSD-based disk arrays, the two SSD-specific problems, namely, the *endurance problem* aggravated by hot parity updates and the *performance problem* due to small random writes, must be solved efficiently.

RAIS5 offers parity redundancy to protect data upon a disk failure. The parity update operations for small write requests exacerbate the flash wearout problem due to the additional and concentrated erase-before-write operations that occur in the parity blocks [Greenan et al. 2009; Balakrishnan et al. 2010a]. For example, Figure 1 shows a RAIS5 disk array consisting of five SSDs (SSD1–SSD5). Updating a data block D_9 on SSD2 will cause the updating of the corresponding parity block P_3' of the same stripe on SSD5, which incurs a total of two SSD reads (for old data block D_9 and old parity block P_3) and two SSD updates (writing new data block D_9' and new parity block P_3'). Obviously, the parity blocks are the hottest updated blocks because each of them is updated for each write to a data block in its corresponding stripe, thus receiving much more write traffic than their data counterparts for random write accesses. This, combined with the poor small-random-write performance and erasure limitations of SSDs, will severely degrade the overall I/O performance and reliability of the RAIS5 disk array.

A recent study has revealed that updates in storage traces are frequent and small [Chan et al. 2014]. For example, their analysis of the MSR Cambridge and Harvard NFS traces has found that more than 90% of the write requests are updates, and more than 60% of updates in the MSR traces are smaller than 4KB. These findings further indicate that the parity update in parity-based disk arrays, particularly SSD-based arrays, is an important and urgent issue to address.

2.3. Motivation

When deploying SSDs in the enterprise and HPC environments [Caulfield et al. 2010; Narayanan et al. 2009], a single SSD cannot satisfy the performance, capacity, and reliability requirements of storage systems. Moreover, failures of SSDs typically occur in the controller silicon rather than in the flash device as indicated by recent studies [Samsung report 2008; Meza et al. 2015]. Thus, it is necessary to apply the RAID technique to SSDs for applications that require high reliability, high performance, and high capacity [Mao et al. 2010]. However, applying the RAID technique to SSDs can be nontrivial due to SSDs' unique characteristics that are different from HDDs. In face of the high cost/GB of the current SSDs and the reliability requirements, RAIS5 is arguably the best choice [Balakrishnan et al. 2010a]. However, the parity-update and small-random-write problems in RAIS5, unabated, will likely cause RAIS5 to experience serious performance degradation and shortened lifespan [Mao et al. 2010].

On the other hand, previous studies and our own analysis have shown that the accesses exhibit a mixed pattern of burstiness and idleness in terms of I/O intensity in enterprise and HPC environments [Golding et al. 1995; Mao et al. 2012]. With the help of the upper-layer optimizing techniques such as buffer and I/O scheduling, the I/Os seen at the disk level are usually bursty and clustered. If I/O requests can be postponed or shifted from busy periods to less busy ones, the resource contention and queue length during the busy periods can be reduced, and the user-perceived system performance can be improved.

Based on the previous observations and analysis, we proposed LDM to exploit the high sequential performance characteristics of HDDs and the logging technique to eliminate or alleviate the negative impact of the parity update process on RAIS performance and reliability. By buffering and merging the small and random write requests into large write requests, and then flushing them to the SSD-based disk arrays during the system idle period, the performance and reliability of SSD-based disk arrays are significantly improved.

3. LOG DISK MIRRORING

In this section, we first outline the main principles guiding the design of LDM. Then we present a system overview of the LDM array, followed by a description of the data structure, the request processing workflow, and the data consistency in the LDM array.

3.1. Design Principles

The design of LDM aims to achieve high performance, high reliability, and high portability, as explained next.

—**High performance.** LDM strives to remove the user's small write requests that originally target the RAIS5 by first sequentially buffering them on the HDD-based RAID1 and then destaging them in a large block to the SSD-based RAIS5 during a later, idle period of the system.

—**High reliability.** By merging multiple small write requests into a large write request, LDM is able to significantly reduce the number of parity updates, thus noticeably alleviating the write amplification problem.

—**High portability.** Low random-write performance is a common problem for most commercial SSDs. LDM can be easily extended to any existing RAIS system, such as RAIS0, RAIS1, and RAIS6.

3.2. System Overview of LDM Array

Figure 2 shows a system overview of our proposed LDM array. As shown in Figure 2, LDM interacts with the *RAID Functional* module and can be incorporated into any existing RAIS5 schemes, such as hardware and software RAIS systems. In Figure 2, the disk subsystem is composed of four SSDs (RAIS5) and two HDDs (RAID1), where the HDD-based RAID1 serves as a write buffer for the SSD-based RAIS5. Noticeably, the write data is laid out sequentially as in a log file system (LFS) [Rosenblum and

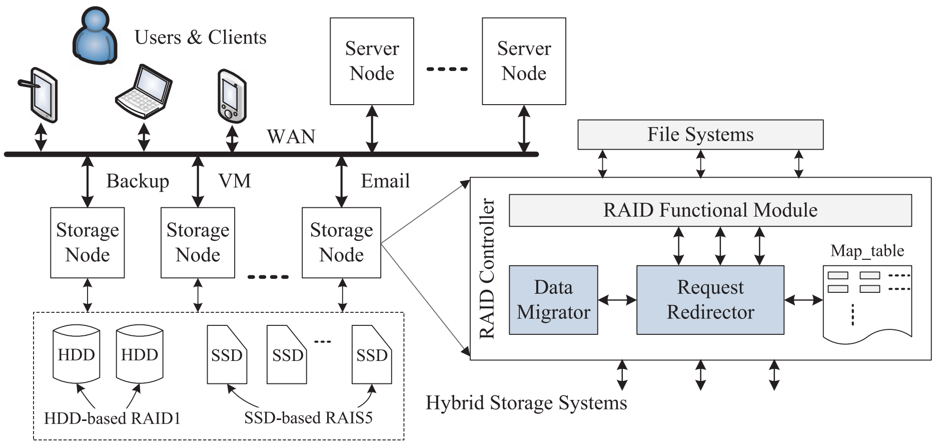


Fig. 2. System overview of an LDM array consisting of four SSDs and two HDDs.

Ousterhout 1992] in the HDD-based RAID1. Rewrites only render the old data invalid instead of overwriting the old data, which makes writing to the HDD-based RAID1 very fast.

LDM consists of two functional modules added into the RAID controller in an existing storage system: Data Migrator and Request Redirector. The *Data Migrator* module is responsible for moving data from the HDD-based RAID1 to the SSD-based RAIS5. The *Request Redirector* module is responsible for issuing a user I/O request to the appropriate location, either the SSD-based RAIS5 or the HDD-based RAID1, according to the *Map_table*. Besides the two functional modules, LDM uses an important data structure to record the mapping information of the redirected write data.

As shown in Figure 2, the *Map_table* is used to log all the redirected write data stored on the HDD-based RAID1. Each entry in the *Map_table* consists of three important parameters: the original address S-LBA on the SSD-based disk array, the temporally stored address H-LBA on the HDD-based disk array, and the request size length. Each entry needs 20 bytes. The size of the *Map_table* is 20 * (m/r), where m is the size of the RAID1-based disk buffer, and r is the request size. For example, if the size of the used RAID1-based disk buffer is 10GB and the average request size is 8KB, the total size of the *Map_table* is only around 25MB. LDM uses nonvolatile memory to store the *Map_table* to protect data in case of power failure. Since the redirected write data is only temporarily stored on the HDD-based RAID1, the nonvolatile memory used to store the mapping information is small. After all the write data on the HDD-based RAID1 is reclaimed, the nonvolatile memory is also released. Thus, the extra memory space can be further reduced by periodically reclaiming the write data from RAID1 to RAIS5 during system idle periods.

3.3. Data Migration

In the LDM array, the small write data from user applications is only temporarily stored on the HDD-based RAID1 and will be migrated to the SSD-based RAIS5 eventually, which is controlled by the Data Migration module. As discussed in Section 2.1, the small-random-write requests are the key bottleneck of performance and reliability for flash-based SSDs. For SSD-based RAIS5, the problem will be even more pronounced due to the write amplification problem that is further exacerbated by the parity update operations in the parity-based RAIS5. To address the problem, the LDM array uses a

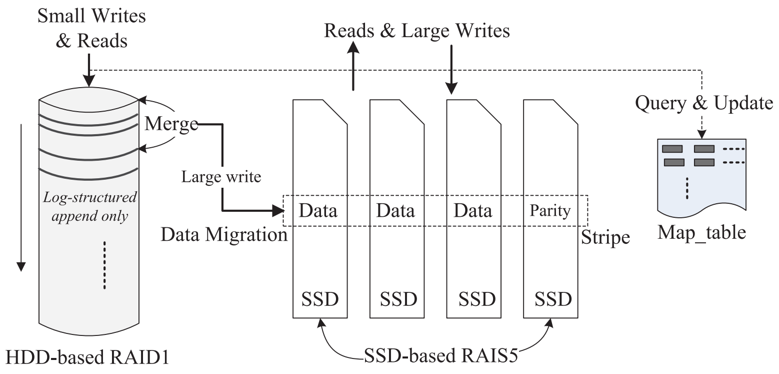


Fig. 3. The workflow of data migration and user request processing.

pair of mirroring log disks (i.e., the HDD-based RAID1) to absorb the small writes on HDDs and merge them into large writes to SSDs.

By using log-structured and append-only write strategies for the HDD-based RAID1, a good write performance is guaranteed. Figure 3 shows the data migration process. LDM first merges multiple adjacent write requests into a large write request. The size of the large write request is dependent on the stripe unit size and the number of SSDs in the SSD-based RAIS5. For example, given an SSD-based RAIS5 consisting of four SSDs and a chunk size of 16KB, the size of the large request is 48KB. In this way, a large write to the SSD-based RAIS5 constitutes a full-stripe write, which achieves the best write performance of an SSD-based RAIS5 with four SSDs. Moreover, by converting write requests into full-stripe writes in this way, the write amplification problem caused by the parity update operations is significantly alleviated, thus improving the write performance and reliability for the SSD-based RAIS5.

The data migration process is by default triggered during a system idle period to reduce the performance impact on the foreground user applications. However, during an IO-intensive period, if the user read requests addressed to the HDD-based RAID1 intensify, the data migration process is also triggered automatically to move the requested data from the HDDs to the SSDs to improve the read performance. Once the write data is migrated from the HDD-based RAID1 to the SSD-based RAIS5, the corresponding entries in the Map_table are deleted.

3.4. Request Processing Workflow

Upon receiving a read request, the Request Redirector first checks whether there is an entry corresponding to the request in the Map_table. If so, the data is read from the HDD-based RAID1, as illustrated in Figure 3. Otherwise, the request will be processed by the SSD-based RAIS5.

Upon receiving a write request, the Request Redirector first obtains the size of the write request. Based on the size, the Request Redirector will determine whether the request is serviced by the HDD-based RAID1 or the SSD-based RAIS5. For example, if the write request size is smaller than 48KB, the request is written to the HDD-based RAID1 sequentially, as illustrated in Figure 3. At the same time, the Map_table is updated to record the redirected write data. Thus, the subsequent read requests can be checked in the Map_table to determine whether the requested data is stored on the HDD-based RAID1. If the write request already exists in the Map_table, the previous log entry will be replaced by the new log entry, but the new write data is written by

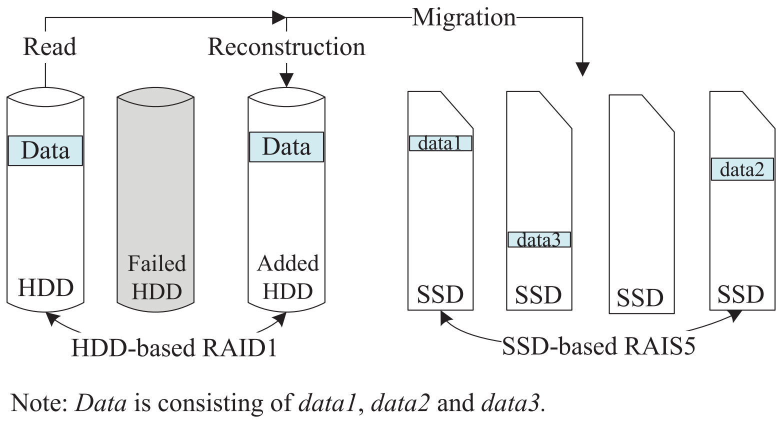


Fig. 4. The concurrent data reconstruction and migration operations in the data recovery process.

appending. Otherwise, a new log entry is created according to the request and inserted into the log list.

3.5. Recovery from a Disk Failure

Disk failure can occur among either the SSDs or the HDDs. If an SSD fails, the recovery operation is initiated in the SSD-based RAIS5. During recovery within the SSD-based RAIS5, all write requests are directed to the HDD-based RAID1 and read requests are served as usual. Since the two HDDs are mirrored as a write buffer that temporarily absorbs write requests and acts as a surrogate device [Wu et al. 2009], the recovery efficiency is improved significantly as evidenced in our experiments (see Section 4).

If an HDD fails, the Data Migrator is triggered to migrate the write data from the HDD-based RAID1 to the SSD-based RAIS5 according to the Map_table. Along with the data migration process, the data reconstruction operation within the HDD-based RAID1 is also in progress. Moreover, the data migration and data reconstruction operations are processed concurrently to reduce the data read overhead from the surviving HDD, as indicated in Figure 4. After the data migration and data reconstruction processes complete, the newly formed HDD-based RAID1 again acts as a write buffer for the SSD-based RAIS5 in the form of a pair of mirroring log disks. During the data migration period, all user requests are served by the SSD-based RAIS5. The Map_table is also checked to ensure data consistency; that is, if the write request is in the Map_table, the corresponding entry should be deleted after the data is written to the SSD-based RAIS5.

3.6. Data Consistency

Data consistency in the LDM array means that (1) the key data structure must be safely stored, (2) the redirected write data must be reliably stored on the HDDs, and (3) the user read requests must fetch the up-to-date data.

First, to prevent the loss of the Map_table in the event of a power supply failure or a system crash, LDM stores the Map_table in a nonvolatile RAM (NVRAM). Since the size of the Map_table is generally small, it will not incur significant hardware cost. In order to improve the write performance by using the write-back strategy, NVRAM is commonly deployed in the RAID controller. It is usually buildup of DRAM backed with a battery. Thus, it is easy and reasonable to use a small amount of NVRAM to store the Map_table. Moreover, in the high-end RAID-based storage products, the NVRAM is mirrored in the multiple RAID controllers to prevent a single point of failure.

Table I. Experimental Setup

Machine	Intel Xeon E5-2407, 8GB RAM
OS	Linux 2.6.21.1
Device adapter	PERC H710 SATA controller
Disk driver	Intel X25-E 64GB SATA SSD Seagate Savvio 73GB SATA HDD
Traces	OLTP [UMass Trace Repository 2010] MSR Traces [Microsoft Enterprise Traces 2009]
Trace replay tool	RAIDmeter [Wu et al. 2012]

Second, the redirected write data must be safely stored on the HDD-based RAID1. The LDM array uses two strategies to guarantee the data availability: mirroring and concurrent failure recovery. In the former, two HDDs are used to form a RAID1 to protect data loss caused by an HDD failure. For the latter, when an HDD fails, the LDM array first migrates the write data to the SSD-based RAIS5 and reconstructs the fetched data to the replacement HDD.

Third, since the up-to-date data for a read request can be stored on either the SSD-based RAIS5 or the HDD-based RAID1, each read request is first checked in the Map_table to determine whether it should be serviced by the former or the latter to keep the fetched data always up-to-date, until all redirected write data is migrated to the SSD-based RAIS5.

4. PERFORMANCE EVALUATIONS

In this section, we first describe the experimental setup and evaluation methodology. Then we evaluate the performance of the LDM array through trace-driven simulations.

4.1. Experimental Setup and Evaluation Methodology

We have implemented an LDM array prototype on top of the Linux Software RAID framework as an independent module. The performance evaluation is conducted on a Dell PowerEdge T320 with an Intel Xeon E5-2407 processor and 8GB DDR memory. In the system, there is a PERC H710 SATA controller card to house different SATA disks, including both SSDs and HDDs. The SSD module is the Intel X25-E 64GB SATA Solid-State Drive and the HDD module is the Seagate Savvio 73GB SATA Disk. A separate hard disk is used to house the operating system (Linux kernel 2.6.21.1) and other software (MD, mdadm, and RAIDmeter). The experimental setup is outlined in Table I.

The traces used in our experiments are obtained from the Storage Performance Council [UMass Trace Repository 2010] and realistic enterprise-scale environments [Microsoft Enterprise Traces 2009]. The two financial traces (Financial1 and Financial2) were collected from OLTP applications running at a large financial institution. The four enterprise-scale workloads (src1_2, usr_0, proj_0, and prxy_0) are collected from Microsoft enterprise platforms. These traces represent different access patterns in terms of read/write ratio, IOPS, and average request size, with the main workload parameters summarized in Table II. Performance evaluation uses the RAIDmeter [Wu et al. 2015] that is a block-level trace replay software capable of replaying traces and evaluating the I/O response time of the storage device.

To evaluate the LDM array, we compare it with other architectures of SSD-based disk arrays. The first disk array architecture is the basic RAIS5 of different sizes (e.g., 4-RAIS5 represents RAIS5 consisting of four SSDs and 6-RAIS5 indicates RAIS5 consisting of six SSDs). The second disk array architecture is HPDA, which consists of four SSDs and two HDDs [Mao et al. 2012], where one HDD serves as the parity disk for the SSD-based RAID4. The third disk array architecture is the same as the

Table II. The Workload Characteristics

Trace	Trace Characteristic		
	Read Ratio	IOPS	Aver. Req. Size (KB)
Fin1	32.8%	52	11.9
Fin2	82.4%	127	6.2
src1_2	21.5%	14	30.0
usr_0	49.4%	26	17.1
proj_0	46.7%	28	14.8
prxy_0	8.3%	224	2.7

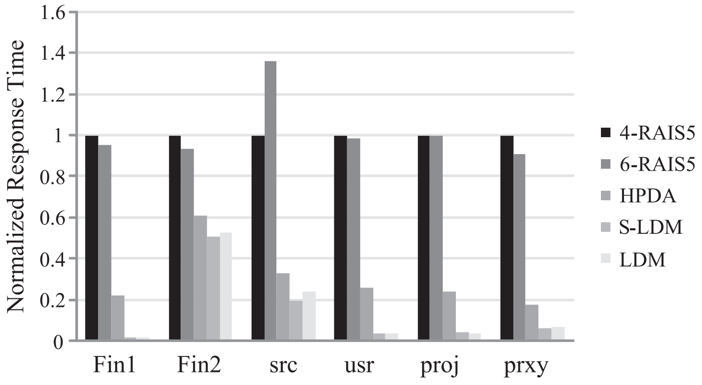


Fig. 5. Response time comparison among the different disk array schemes.

Table III. Average Response Time Improvement of LDM over 4-RIAS5 and HPDA

Response Time Improvement	Traces					
	Fin1	Fin2	src	usr	proj	prxy
Over 4-RAIS5	51.7	2.0	5.1	25.3	22.1	16.5
Over HPDA	12.1	1.2	1.4	6.7	5.8	2.6

LDM array except that the HDD-based RAID1 is replaced by SSD-based RAIS1 that consists of two SSDs (S-LDM). The purpose of this comparison is to show how the log disk mirroring scheme improves the system performance and reliability with the same number of SSDs.

4.2. Performance Results and Analysis

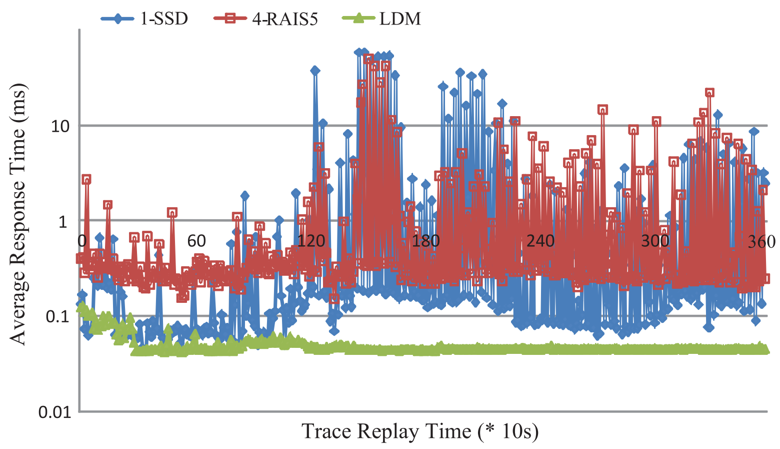
Figure 5 and Table III show the performance results in the normal mode where all disks perform normally. In the normal mode during the evaluations, 4-RAIS5 consists of four SSDs, and 6-RAIS5 consists of six SSDs. Both HPDA and LDM consist of four SSDs and two HDDs. S-LDM consists of six SSDs. In terms of the average response time, the LDM array outperforms the SSD-based disk array 4-RAIS5 by a factor of up to 51.7, 2.0, 5.1, 25.3, 22.1, and 16.5, and outperforms the HPDA by a factor of up to 12.1, 1.2, 1.4, 6.7, 5.8, and 2.6 for the Fin1, Fin2, src, usr, proj, and prxy traces, respectively. The reason is that for the OLTP and MSR workloads, the I/O requests are usually small and random. Moreover, the updates are common in these storage traces [Chan et al. 2014]. Thus, for the SSD-based RAIS5, the access latency for random small write requests is very long. In particular, since most requests of these workloads are smaller than the size of a flash page (as shown in Table II), these write requests incur substantial erase-before-write operations for SSDs, thus adversely impacting performance. However, the disk head seeks between the buffer area and the parity area on the parity HDD in

HPDA will degrade the system performance [Mao et al. 2012]. In contrast, both the LDM array and S-LDM array use the log-structured buffer to store the small writes, which significantly reduces the negative performance impact to the SSD-based RAIS5. Moreover, the LDM array and S-LDM array merge small updates into large write requests, further improving the storage performance.

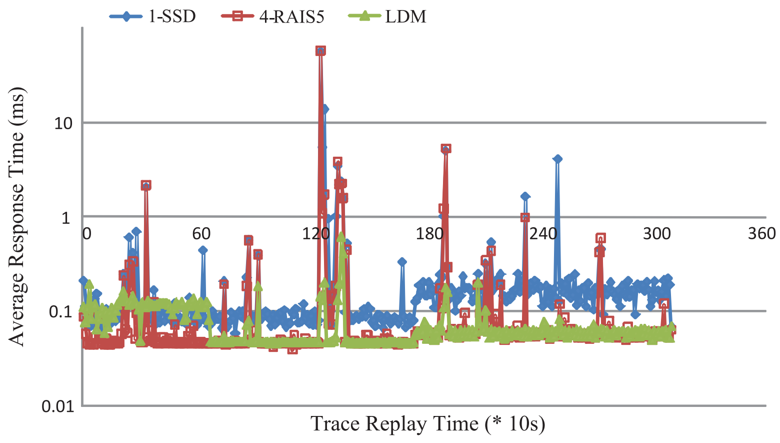
It is also clear that the S-LDM array performs better than 6-RAIS5, further validating the efficacy of the use of the log mirroring buffer architecture for small updates. Moreover, the LDM array performs comparably with the S-LDM array. The reason is that the writes to the mirroring buffer are sequentially appended, and the sequential performance gap between HDDs and SSDs is negligible, thus making the performances of sequential accesses to the SSD-based RAIS1 (in the S-LDM array) and the HDD-based RAID1 (in the LDM array) almost identical. It is interesting to note that, despite its two extra SSDs, 6-RAIS5 performs almost the same as 4-RAIS5 for all the traces. The reason is that, with a large stripe width (i.e., number of disks), the parity updates are much more frequent. Thus, the increased parity updates will offset the benefit from the increased parallelism by the two additional SSDs in 6-RAIS5. These results are also consistent with the previous studies [Mao et al. 2010; Soundararajan et al. 2010].

Figure 6 compares in more detail the user I/O performances of LDM, 4-RAIS5, and a single SSD during the trace replay period, and highlights the significant advantage of LDM over 4-RAIS5. Two aspects of the significant improvement in the user response time are demonstrated in these figures. First, LDM has much better stable user response times than that of a single SSD and 4-RAIS5. The reason is that for pure SSD-based storage systems, the garbage collection processes will significantly affect the system performance by overlapping with foreground user I/O requests. In contrast, LDM uses the disk mirroring buffer, which can absorb the foreground user I/O requests to alleviate the interaction, thus providing better stable performance. Though the onset of the user response times is much higher than 4-RAIS5, the stability of user response times is much better than 4-RAIS5. Second, LDM has fewer high-response time points than that of a single SSD and 4-RAIS5. In Figure 6, we can see that both a single SSD and 4-RAIS5 have many time points where the response times exceed 1ms and even go up to 10ms. The high user response times will significantly affect the QoS (Quality of Service) of the applications, thus violating the SLA (Service-Level Agreement). In contrast, the highest user average response time of LDM is less than 1ms, which guarantees the service qualities.

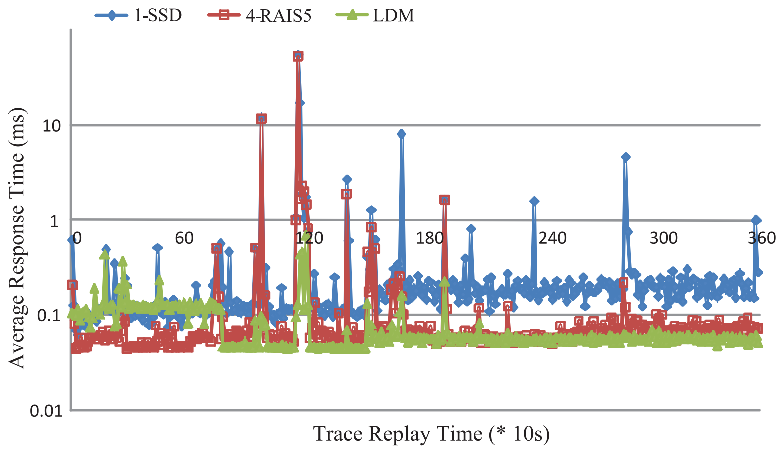
Besides the evaluation on response time (latency), we also compare the performance/cost ratio among the different disk array schemes, illustrated in Figure 7. For the performance/cost calculation, we use the evaluated response times as the performance values and the total cost of the storage devices as the cost values. For the current generation of flash-based SSDs, the cost/GB is about 10 times that of HDDs [Solid State Storage Initiative 2010]. That is, when we set the cost value of 1GB HDD space as 1, we will set the cost value of 1GB SSD space as 10. Figure 7 shows that both the LDM array and S-LDM array are more cost efficient than the SSD-based disk arrays (4-RAIS5 and 6-RAIS5) and HPDA. Moreover, the LDM array performs the best among all five disk array schemes. Though HPDA and LDM have the same devices, the performance of LDM is much better than HPDA with the different data layout schemes. Thus, the LDM array has a higher performance/cost ratio than HPDA. Compared with the S-LDM array, the LDM array achieves comparable performance but at a lower cost, as indicated in Figure 5, making the LDM array the most cost-efficient scheme among all five disk array schemes. This is also the key reason that we choose a 2-HDD-based RAID1 in the LDM array as the log mirroring buffer, rather than a 2-SSD-based RAID1.



(a) Fin1



(b) usr



(c) proj

Fig. 6. The user average response times for the different disk array schemes.

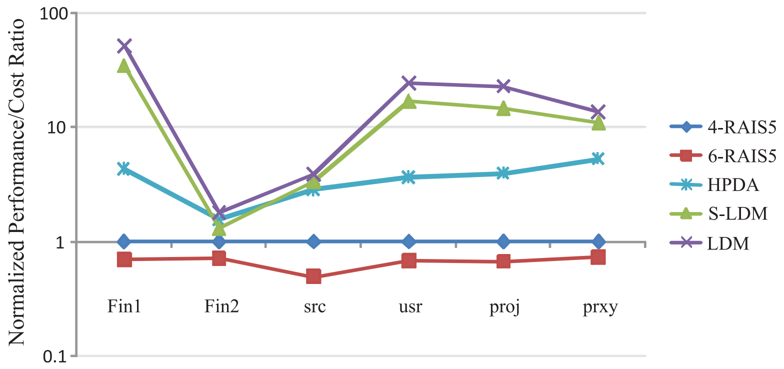
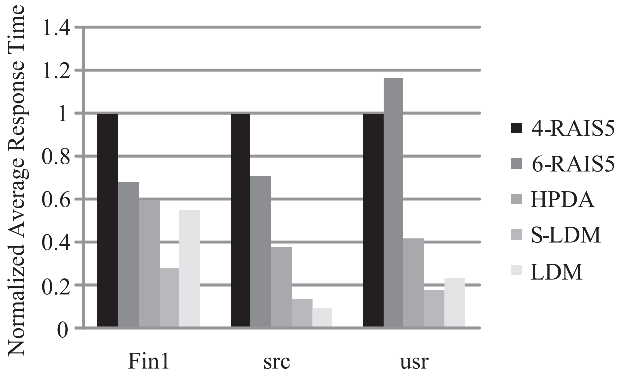
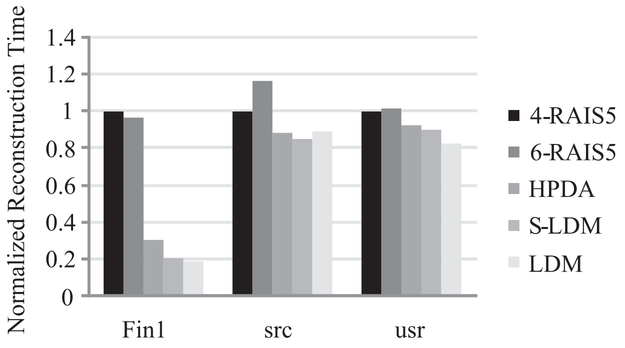


Fig. 7. Performance/cost ratio comparison among the different disk array schemes. Note that a higher ratio is better.



(a) The normalized average response times



(b) The normalized reconstruction times

Fig. 8. Normalized average response time and reconstruction time for the different disk array schemes.

To evaluate how effectively the LDM array handles failure recovery, we also conducted experiments on the recovery process of different disk array schemes. Figures 8(a) and 8(b) show respectively the average response time during recovery and the reconstruction time for the five disk array schemes driven by the three traces of Fin1, src, and usr. Similar to the normal mode, the LDM array and S-LDM array significantly outperform both 4-RAIS5 and 6-RAIS5 in terms of average response times and reconstruction time. The reason is that, in the cases of 4-RAIS5 and 6-RAIS5, the reconstruction I/Os

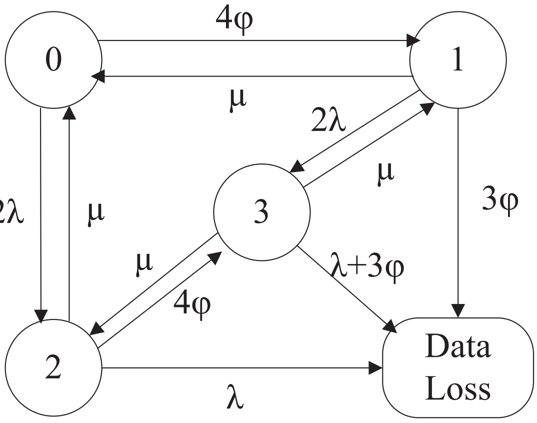


Fig. 9. State-transition diagram for an LDM array consisting of four SSDs and two HDDs.

and user I/Os compete for disk resources, thus increasing the reconstruction time and user response time simultaneously. In the LDM array and S-LDM array, the log mirroring buffer absorbs all write requests, thus significantly alleviating the contentions between the user I/Os and reconstruction I/Os. This results in the simultaneous reduction in the reconstruction time and user response time [Wu et al. 2012].

5. RELIABILITY ANALYSIS

In this section, we adopt the MTTDL metric to estimate the reliability of the LDM array, taking into account the impact of the reconstruction time. To the best of our knowledge, there is no consensus on an acceptable MTTDL in the industry for storage systems. What is generally accepted is that the higher the MTTDL, the higher the reliability of the storage system. Thus, when we design a reliable storage system, we should pursue as high an MTTDL as possible [Mao et al. 2015; Kao et al. 2013].

Our system model consists of a disk array with independent failure processes for all disks. When a disk fails, a reconstruction process is immediately initiated for that disk. We assume that HDD failures, SSD failures, and the reconstruction process are independent events following an exponential distribution of rate λ, φ, and μ, respectively. While some of these assumptions are not necessarily true for real systems, they are used in order to use stochastic models with finite numbers of states [Mao et al. 2010; Kao et al. 2013], and all the disk array schemes are based on the same assumptions.

According to the conclusion about the MTTDL results of RAIS5 [Kao et al. 2013], the MTTDL of RAIS5 consisting of four SSDs is

$$MTTDL_{4-RAIS5} = \frac{7\varphi + \mu}{12\varphi^2}, \tag{1}$$

and the MTTDL of RAIS5 consisting of six SSDs is

$$MTTDL_{6-RAIS5} = \frac{11\varphi + \mu}{30\varphi^2}. \tag{2}$$

Figure 9 shows the state transition diagram for an LDM array consisting of four SSDs and two HDDs. State <0> represents the normal state of the disk array when its six disks are all operational. A failure of any of the four SSDs would bring the disk array to state <1>. A failure of any of the two HDDs would bring the disk array to state <2>. A failure of any of the two HDDs in state <1> or a failure of any of the four SSDs in state <2> will bring the disk array to state <3>. A failure of a second

SSD in state <1> or a failure of a second HDD in state <2> would result in data loss. Any disk failure in state <3> would result in data loss. The reconstruction transition brings the disk array back from state <3> to state <1> or state <2>, then from state <2> to state <0> or from state <1> to state <0>. The failure of the other HDD does not incur data loss in the mirroring buffer. Thus, we omit the condition of other HDD failure events from the state-transition diagram, which does not impact the results.

The Kolmogorov system of differential equations describing the behavior of this LDM array is expressed in Equation (3):

$$\begin{cases} \dfrac{dp_0(t)}{dt} = -(4\varphi + 2\lambda)p_0(t) + \mu p_1(t) + \mu p_2(t) \\ \dfrac{dp_1(t)}{dt} = -(3\varphi + 2\lambda + \mu)p_1(t) + 4\varphi p_0(t) + \mu p_3(t) \\ \dfrac{dp_2(t)}{dt} = -(4\varphi + \lambda + \mu)p_2(t) + 2\lambda p_0(t) + \mu p_3(t) \\ \dfrac{dp_3(t)}{dt} = -(3\varphi + \lambda + 2\mu)p_3(t) + 2\lambda p_1(t) + 4\varphi p_2(t), \end{cases} \tag{3}$$

where $p_i(t)$ is the probability that the disk array is in state $<i>$ with the initial condition $p_0(0) = 1$ and $p_i(0) = 0$ for $i \neq 0$.

The Laplace transformation of Equation (3) is

$$\begin{cases} sp_0^*(s) - 1 = -(4\varphi + 2\lambda)p_0^*(s) + \mu p_1^*(s) + \mu p_2^*(s) \\ sp_1^*(s) = -(3\varphi + 2\lambda + \mu)p_1^*(s) + 4\varphi p_0^*(s) + \mu p_3^*(s) \\ sp_2^*(s) = -(4\varphi + \lambda + \mu)p_2^*(s) + 2\lambda p_0^*(s) + \mu p_3^*(s) \\ sp_3^*(s) = -(3\varphi + \lambda + \mu)p_3^*(s) + 2\lambda p_1^*(s) + 4\varphi p_2^*(s). \end{cases} \tag{4}$$

Observing that MTTDL of the disk array is given by Kao et al. [2013]:

$$MTTDL = \sum_i p_i^*(0). \tag{5}$$

Using Equation (5), we solve the Laplace transformation for $s = 0$ and use Equation (4) to compute MTTDL of an LDM array consisting of four SSDs and two HDDs:

$$MTTDL_{LDM} = \frac{9\lambda\mu + 21\varphi\mu + 2\mu^2}{6\lambda^3 + 4\lambda^2\mu + 24\varphi^2\mu + 42\varphi\lambda^2 + 108\varphi^2\lambda}. \tag{6}$$

Similarly, MTTDL of an S-LDM array consisting of four SSDs and two SSDs is

$$MTTDL_{S-LDM} = \frac{15\varphi\mu + \mu^2}{14\varphi^2\mu + 120\varphi^3}. \tag{7}$$

Figure 10 plots MTTDL as a function of MTTR (Mean Time To Repair/Reconstruct) for the different disk array architectures. The MTTDL of HPDA, which consists of four SSDs and two HDDs, is quoted from the analysis results [Mao et al. 2012]. The HDD failure rate, λ, is assumed to be one failure every 50,000 hours, which was derived from a recent study on real production disk array systems [Schroeder and Gibson 2007]. The SSD failure rate, φ, is assumed to be one failure every 200,000 hours, based on recent studies on SSD failure rates [Driver 2015]. For the SSD-based RAIS5 (4-RAIS5 or 6-RAIS5), due to the flash wearout of the parity update operation, the value of φ is doubled with respect to the basic reliability value of SSD (without frequent parity update operations) [Mao et al. 2012; Meza et al. 2015]. MTTR is expressed in terms

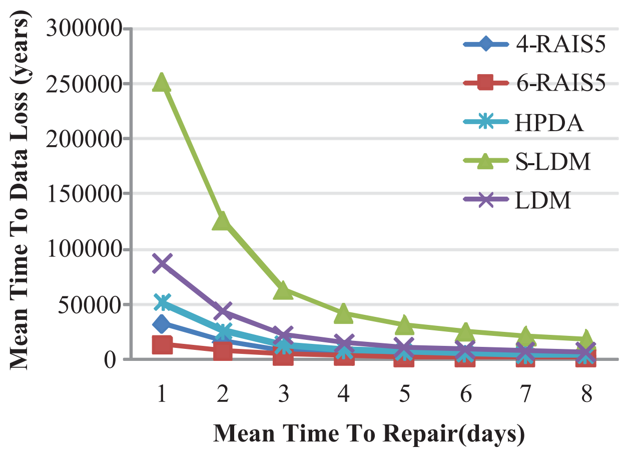


Fig. 10. MTTDL achieved by different disk array levels. Note that a higher MTTDL value indicates a higher reliability.

of days, and MTTDL is expressed in terms of years. From Figure 10, we can see that MTTDL of either the LDM array or S-LDM array is better than that of SSD-based disk arrays (4-RAIS5 and 6-RAIS5) and HPDA. As MTTR increases, the LDM array consistently outperforms the other two schemes in terms of MTTDL. The trend of relative results among the four schemes remains consistent as the total number of disks increases. In summary, compared with the SSD-based RAIS5, our proposed LDM array architecture is much more reliable.

6. RELATED WORK

Flash-based devices have been used in the embedded systems as the persistent storage device of choice for a long time. Along with the popularity and wide deployment of flash-based SSDs, a lot of studies have been conducted to improve the reliability, performance, and energy consumptions from the aspects of FTL design, buffer management, garbage collection, and so on. Recently, many studies have been focused on how to improve the utilization of and exploit the advantages of SSDs. Areas of focus include system optimizations, such as I/O scheduler, file systems, and disk array organizations. Our work falls within the latter category.

Since both HDDs and SSDs have advantages and disadvantages, a lot of studies try to combine them together effectively and avoid their disadvantages [Kim et al. 2011a; Mao et al. 2010; Xie and Sun 2010; Lin et al. 2011; Yang and Ren 2011; Zeng et al. 2012; Mao et al. 2014]. For example, HIT [Xie and Sun 2010] exploits the data access patterns to distribute the data between HDD-based disk arrays and SSD-based disk arrays. By periodically migrating data, it achieves a good balance among system performance, reliability, and energy efficiency. However, the write amplification problem of the SSD-based disk arrays are not addressed. Similar to HIT, HybridStore [Kim et al. 2011a] also employs both SSD-based disk arrays and HDD-based disk arrays in an enterprise-level storage system. The optimizations in HybridStore work above the two disk arrays and are not aware of the internal problems of the SSD-based disk arrays. In contrast, LDM works inside the SSD-based disk arrays; thus, it is orthogonal to and can further improve system efficiency of HIT and HybridStore.

HPDA [Mao et al. 2010] is an enhanced hybrid RAID4 disk array composed of both HDDs and SSDs. In HPDA, the SSDs (data disks) and part of one HDD (parity disk) compose a RAID4 disk array. The second HDD and the free space of the parity HDD are mirrored as a write buffer that temporarily absorbs small write requests and acts as a surrogate RAID1 set. It uses an HDD to serve as the dedicated parity device, thus avoiding the parity updates on the parity SSD. HRAID6ML [Zeng et al. 2012] extends the HPDA idea in a RAID6 storage architecture with an HDD-based parity device, thus improving both the performance and reliability of SSD-based disk arrays. However, since both HPDA and RAID6ML use the HDD storage space as a mirroring buffer that is shared with the parity HDD space, the contention between the parity area and the buffer area during the write-intensive and data migration periods will significantly degrade the system performance [Mao et al. 2012]. The parity HDD will also become a performance bottleneck under write-intensive workloads. Moreover, the read requests addressed to the shared HDD storage space will further degrade system performance. In contrast, the LDM scheme uses dedicated HDD space to absorb the small write data, thus avoiding the disk seek movement between disjoint regions of the shared HDD space, which significantly improves the system performance.

Griffin [Soundararajan et al. 2010] is a hybrid storage device that uses a hard disk drive as a write cache for an SSD. By maintaining a log-structured HDD cache and migrating cached data periodically, the hybrid design reduces writes to the SSD while retaining its performance advantages. CR5M [Wang et al. 2014] is also a hybrid RAID1/RAID5 scheme for the flash chips within a single SSD. Our study is related in spirit to Griffin and CR5M; it targets improving performance and reliability of an SSD-based disk array, not of a single SSD. More importantly, LDM exploits the parallelism and parity update characteristics of SSD-based disk arrays to merge the small writes into full-stripe writes to avoid the frequent parity-update operations, thus significantly improving system performance and reliability. As a result, the LDM array is more suitable for enterprise and HPC storage systems that require high capacity, performance, and reliability.

There are also some studies focused on pure SSD-based disk arrays consisting of exclusively SSDs [Balakrishnan et al. 2010a; Chung and Hsu 2014; Du et al. 2011; Im and Shin 2011; Yi et al. 2013; Zhang et al. 2013; Wu et al. 2015]. For example, Diff-RAID [Balakrishnan et al. 2010a] distributes the parity unevenly across SSDs in the SSD-based RAIS5 and pre-replaces the faster-degraded SSD to improve the system reliability. However, Diff-RAID does not alleviate the write amplification problem induced by the parity updates on the RAIS5. It deliberately makes the SSD failures uneven in the disk array, one by one, thus increasing the reliability of the SSD-based RAIS5. It essentially trades performance for reliability for pure SSD-based disk arrays. In contrast to Diff-RAID, WeLe-RAID [Du et al. 2011] enhances the endurance by using an even and global wear-leveling mechanism among the SSDs in the entire RAIS system. It improves the performance by achieving better load balance across the SSDs. The GGC scheme [Kim et al. 2011b] aims to alleviate the performance variability induced by the individual and uncoordinated garbage collections of SSDs in the RAIS system by using a global collaborated garbage collection. Flash-aware RAID [Im and Shin 2011] utilizes the cache in the storage system to reduce the number of internal write operations to alleviate the garbage collection overhead. Two optimizations, delayed parity update and partial parity techniques, are embedded inside the SSD-based disk arrays, which is orthogonal to our proposed LDM scheme.

As a summary, Table IV compares the characteristics between LDM and the state-of-the-art schemes. We can see that compared with HPDA, LDM is much more applicable to parity-based disk arrays. In order to address the applicable and scalable issues with LDM, we also provide a discussion section, which follows.

7. DISCUSSION

The idea of LDM is motivated by a simple goal of simultaneously reducing the small-random-write traffic and avoiding the write amplification problem through a log disk

Table IV. Comparison between LDM and the State-of-the-Art Schemes

Schemes	Characteristics			
	Hybrid	RAID Level	Parity Disk	Write Buffer
HIT [Xie and Sun 2010]	Yes	RAID10	No	No
Diff-RAID [Balakrishnan et al. 2010b]	No	RAID5	SSD	No
Griffin [Soundararajan et al. 2010]	Yes	Single	No	Yes
HPDA [Mao et al. 2012]	Yes	RAID4	HDD	Yes
LDM	Yes	RAID5	SSD	Yes

mirroring buffer for SSD-based disk arrays. However, our process of turning this design goal into a practical deployment reveals several interesting insights and considerations.

First, the log disk mirroring buffer in LDM is a RAID1 consisting of two HDDs. The write requests on the RAID1 are sequentially processed since they are organized in a log-structured manner with an append-only strategy. Besides RAID1, LDM also can use RAID5 or RAID6 acting as a log disk buffer for RAIS5. However, by using parity-based disk arrays, such as RAID5/6, the write accesses on the log disk buffer may be significantly slowed down due to the extra parity calculation and maintenance overhead. However, since the writes to the log disk buffer are sequential, the extra parity update overhead will be much less significant. Moreover, the RAID5/6-based log disk buffer will provide comparable or better read performance and reliability than the RAID1-based 2-HDD log disk buffer, though they may need many more disks.

Second, SSDs are able to deliver very high data rates. However, it changes as the system scales up to deal with the combined data rates of several drives that are accessed in parallel in an SSD-based disk array [Jeremic et al. 2011]. Thus, in SATA SSD-based disk arrays, the total number of disks is limited to four or six for performance and reliability considerations [EMC Symmetrix DMX Architecture 2010; Jeremic et al. 2011]. In these applications, the latency, rather than the throughput, is the desired objective. In order to achieve high throughput, they usually employ the PCIe SSDs in server platforms. In an HPC environment, they usually employ SAS/FC HDDs to satisfy both the high throughput and high capacity requirements. Though SATA SSD-based disk arrays may not scale up by adding more disks, they can provide high access bandwidth by using multiple storage nodes in a cluster or distributed storage environment. Each storage node is configured with a SATA SSD-based disk array. Thus, our LDM scheme also works well in these environments to provide fast data accesses. Moreover, in order to support better reliability, performance, and scalability, large-scale storage systems usually employ the LVM (Logical Volume Manager) technology, which provides storage virtualization capabilities [Virtualizing Storage for Scale, Resiliency, and Efficiency 2012; Logical Volume Manager (Linux) 2015]. LVM usually works on top of the multiple RAID sets. It allows the system administrators to combine the multiple physical storage elements into a collective storage pool, which can then be allocated and managed according to the application requirements, without regard for the specifics of the underlying physical disk systems.

8. CONCLUSION

The trick to maintaining the cloud as an active storage tier is compensating for latency. Flash-based SSDs are a promising assistant to HDDs to reduce the access latency. In this article, we propose an LDM scheme to improve the performance and reliability of SSD-based disk arrays. LDM is a hybrid disk array architecture that consists of several SSDs and two HDDs. In the LDM array, the SSDs (data disks) compose a RAIS5 disk array, whereas the two HDDs are mirrored as a write buffer that temporarily absorbs small write requests and acts as a surrogate RAID1 set during recovery when a disk fails. The write data is reclaimed to the data disks during system idle periods. Our prototype implementation of the LDM array and its performance evaluation show that the LDM array significantly outperforms other SSD-based disk arrays by a factor of 20.4 on average, and outperforms HPDA by a factor of 5.0 on average. The reliability analysis shows that the MTTDL of the LDM array is 2.7 times and 1.7 times better than that of pure SSD-based and HPDA disk arrays, respectively.

Our proposed LDM array architecture is an ongoing research project, and we are currently exploring several directions for the future work. First, we will evaluate LDM for other RAID levels of SSD-based arrays, such as RAIS0, RAIS1, and RAIS6. Second, we will investigate a garbage-collection-aware data layout and coding scheme for RAIS to further improve the performance and reliability simultaneously. Last, we will conduct many more experiments with benchmark tools to evaluate the bandwidth performance in the HPC environment. In these environments, not only response times but also the system throughput would be the main objective.

ACKNOWLEDGMENTS

The authors would like to thank ASTL members at Xiamen University for their active discussion about the project.

REFERENCES

N. Agrawal, V. Prabhakaran, T. Wobber, J. Davis, M. Manasse, and R. Panigrahy. 2008. Design tradeoffs for SSD performance. In *Proceedings of the 2008 USENIX Annual Technical Conference (USENIX'08)*. 57–70.

M. Balakrishnan, A. Kadav, V. Prabhakaran, and D. Malkhi. 2010a. Differential RAID: Rethinking RAID for SSD reliability. In *Proceedings of the 5th European Conference on Computer systems (EuroSys'10)*. 15–26.

M. Balakrishnan, A. Kadav, V. Prabhakaran, and D. Malkhi. 2010b. Differential RAID: Rethinking RAID for SSD reliability. *ACM Transactions on Storage* 6, 2 (2010), 1–22.

A. M. Caulfield, J. Coburn, T. Mollov, A. De, A. Akel, J. He, A. Jagatheesan, R. K. Gupta, A. Snavely, and S. Swanson. 2010. 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'10)*. 1–11.

J. C. W. Chan, Q. Ding, P. P. C. Lee, and H. H. W. Chan. 2014. Parity logging with reserved space: Towards efficient updates and recovery in erasure-coded clustered storage. In *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST'14)*. 163–176.

F. Chen, D. A. Koufaty, and X. Zhang. 2009. 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'09)*. 181–192.

C. Chung and H. Hsu. 2014. Partial parity cache and data cache management method to improve the performance of an SSD-based RAID. *IEEE Transactions on Very Large Scale Integration Systems* 22, 7 (2014), 1470–1480.

C. Dirik and B. Jacob. 2009. 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'09)*. 279–289.

SanDisk Solid State Driver. 2015. https://itblog.sandisk.com/truth-ssds-hdd-vendors-do-not-want-you-to-know/.

Y. Du, F. Liu, Z. Chen, and X. Ma. 2011. WeLe-RAID: A SSD-based RAID for system endurance and performance. In *Proceedings of the 8th IFIP International Conference on Network and Parallel Computing (NPC'11)*. 248–262.

EMC Symmetrix DMX Architecture. 2010. Retrieved from https://www.emc.com/collateral/hardware/solution-overview/c1011-symm-dmx-architecture-prod-desc-gd.pdf.

R. Golding, P. Bosch, and C. Staelin. 1995. Idleness is not sloth. In *Proceedings of the USENIX Technical Conference (USENIX'95)*. 201–212.

K. Greenan, D. D. E. Long, E. L. Miller, T. Schwarz, and A. Wildani. 2009. Building flexible, fault-tolerant flash-based storage systems. In *Proceedings of the 5th Workshop on Hot Topics in System Dependability (HotDep'09)*.

L. M. Grupp, J. D. Davis, and S. Swanson. 2012. The bleak future of NAND flash memory. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST'12)*.

S. Im and D. Shin. 2011. Flash-aware RAID techniques for dependable and high-performance flash memory SSD. *IEEE Transactions on Computers* 1, 60 (2011), 80–92.

N. Jeremic, G. Mühl, A. Busse, and J. Richling. 2011. The pitfalls of deploying solid-state drive RAIDs. In *Proceedings of the 4th Annual International Conference on Systems and Storage (SYSTOR'11)*. 1–13.

H. Kao, J. Paris, D. D. E. Long, and T. Schwarz. 2013. A flexible simulation tool for estimating data loss risks in storage arrays. In *Proceedings of the 29th IEEE Symposium on Massive Storage Systems and Technologies (MSST'13)*. 1–5.

Y. Kim, A. Gupta, B. Urgaonkar, P. Berman, and A. Sivasubramaniam. 2011a. HybridStore: A cost-efficient, high-performance storage system combining SSDs and HDDs. In *Proceedings of the 19th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'11)*. 227–236.

Y. Kim, S. Oral, G. M. Shipman, J. Lee, D. A. Dillow, and F. Wang. 2011b. 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'11)*. 1–12.

L. Lin, Y. Zhu, J. Yue, Z. Cai, and B. Segee. 2011. Hot random off-loading: A hybrid storage system with dynamic data migration. In *Proceedings of the 19th Annual Meeting of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOSTS'11)*. 318–325.

Logical Volume Manager (Linux). 2015. Retrieved from https://en.wikipedia.org/wiki/Logical_Volume_Manager_(Linux).

B. Mao, H. Jiang, D. Feng, S. Wu, J. Chen, L. Zeng, and L. Tian. 2010. HPDA: A hybrid parity-based disk array for enhanced performance and reliability. In *Proceedings of 24th International Parallel & Distributed Processing Symposium (IPDPS'10)*. 1–12.

B. Mao, H. Jiang, S. Wu, Y. Fu, and L. Tian. 2014. Read performance optimization for deduplication-based storage systems in the cloud. *ACM Transactions on Storage* 10, 2 (2014), 1–22.

B. Mao, H. Jiang, S. Wu, L. Tian, D. Feng, J. Chen, and L. Zeng. 2012. HPDA: A hybrid parity-based disk array for enhanced performance and reliability. *ACM Transactions on Storage* 8, 1 (2012), Article 4.

B. Mao and S. Wu. 2015. Exploiting request characteristics and internal parallelism to improve SSD performance. In *Proceedings of the 33rd IEEE International Conference on Computer Design (ICCD'15)*. 476–479.

B. Mao, S. Wu, and H. Jiang. 2015. Improving storage availability in cloud-of-clouds with hybrid redundant data distribution. In *Proceedings of the 29th IEEE International Parallel & Distributed Processing Symposium (IPDPS'15)*. 633–642.

J. Meza, Q. Wu, S. Kumar, and O. Mutlu. 2015. A large-scale study of flash memory failures in the field. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'15)*. 177–190.

Microsoft Enterprise Traces. 2009. Retrieved from http://iotta.snia.org/traces/list/BlockIO.

C. Min, K. Kim, H. Cho, S. Lee, and Y. Eom. 2012. SFS: Random write considered harmful in solid state drives. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST'12)*.

S. Moon and A. L. Narasimha Reddy. 2013. Don't Let RAID raid the lifetime of your SSD array. In *Proceedings of the 5th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage'13)*.

D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron. 2009. Migrating server storage to SSDs: Analysis of tradeoffs. In *Proceedings of the 4th European Conference on Computer Systems (EuroSys'09)*. 145–158.

D. A. Patterson, G. Gibson, and R. H. Katz. 1988. A case for redundant arrays of inexpensive disks (RAID). In *Proceedings of the International Conference on Management of Data (SIGMOD'88)*. 109–116.

M. Rosenblum and J. Ousterhout. 1992. The design and implementation of a log-structured file system. *ACM Transactions on Computer Systems* 10, 1 (1992), 26–52.

Samsung Report. 2008. Retrieved from http://news.cnet.com/8301-13924_3-9876557-64.html.

B. Schroeder and G. A. Gibson. 2007. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? In *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST'07)*. 1–16.

Solid State Storage Initiative. 2010. Retreived from http://www.snia.org/forums/sssi.

G. Soundararajan, V. Prabhakaran, M. Balakrishnan, and T. Wobber. 2010. Extending SSD lifetimes with disk-based write caches. In *Proceedings of the 8th USENIX Conference on File and Storage Technologies (FAST'10)*. 101–114.

UMass Trace Repository. 2010. Retrieved from http://traces.cs.umass.edu/index.php/Storage/Storage.

Virtualizing Storage for Scale, Resiliency, and Efficiency. 2012. Retrieved from http://blogs.msdn.com/b/b8/archive/2012/01/05/virtualizing-storage-for-scale-resiliency-and-efficiency.aspx.

Y. Wang, W. Wang, T. Xie, W. Pan, Y. Gao, and Y. Ouyang. 2014. CR5M: A mirroring-powered channel-RAID5 architecture for an SSD. In *Proceedings of 30th Symposium on Mass Storage Systems and Technologies (MSST'14)*. 1–10.

S. Wu, H. Jiang, D. Feng, L. Tian, and B. Mao. 2009. WorkOut: I/O workload outsourcing for boosting the raid reconstruction performance. In *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST'09)*. 239–252.

S. Wu, H. Jiang, D. Feng, L. Tian, and B. Mao. 2011. Improving availability of RAID-structured storage systems by workload outsourcing. *IEEE Transactions on Computers* 60, 1 (2011), 64–79.

S. Wu, H. Jiang, and B. Mao. 2012. 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'12)*. 17–32.

S. Wu, H. Jiang, and B. Mao. 2015. Proactive data migration for improved storage availability in large-scale data centers. *IEEE Transactions on Computers* 64, 9 (2015), 2637–2651.

S. Wu, W. Yang, B. Mao, and Y. Lin. 2015. MC-RAIS: Multi-chunk redundant array of independent SSDs with improved performance. In *Proceedings of the 15th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'15)*. 18–32.

T. Xie and Y. Sun. 2010. Dynamic data reallocation in hybrid disk arrays. *IEEE Transactions on Parallel and Distributed Systems* 21, 9 (2010), 1330–1341.

Q. Yang and J. Ren. 2011. I-CASH: Intelligently coupled array of SSD and HDD. In *Proceedings of the 17th International Symposium on High Performance Computer Architecture (HPCA'11)*. 278–289.

L. Yi, J. Shu, J. Ou, and W. Zheng. 2013. CG-Resync: Conversion-guided resynchronization for a SSD-based RAID array. In *Proceedings of the 31st International Conference on Computer Design (ICCD'13)*. 455–458.

L. Zeng, D. Feng, J. Chen, Q. Wei, B. Veeravalli, and W. Liu. 2012. HRAID6ML: A hybrid RAID6 storage architecture with mirrored logging. In *Proceedings of the 28th IEEE Conference on Massive Data Storage (MSST'12)*. 1–6.

Y. Zhang, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. 2013. Warped mirrors for flash. In *Proceedings of the 29th IEEE Symposium on Massive Storage Systems and Technologies (MSST'13)*. 1–12.

Received October 2014; revised November 2015; accepted February 2016