An Empirical Study of Redundant Array of Independent Solid-State Drives (RAIS)

Youngjae Kim Sarp Oral Dave Dillow Feiyi Wang Douglas Fuller
Stephen Poole Galen Shipman
National Center for Computational Sciences
Oak Ridge National Laboratory, Oak Ridge TN 37831
{yk7, oralhs, dillowda, fwang2, fullerdj, spoole, gshipman}@ornl.gov

Abstract

Solid-state drives (SSD) are popular storage media devices alongside magnetic hard disk drives (HDD). SSD flash chips are packaged in HDD form factors and SSDs are compatible with regular HDD device drivers and I/O buses. This compatibility allows easy replacement of individual HDDs with SSDs in existing storage systems. However, under certain circumstances, SSD write performance can be significantly slowed by garbage collection (GC) processes. The frequency of GC activity is directly correlated with the frequency of write operations and the amount of data written. GC scheduling is locally controlled by internal SSD logic. This paper studies the feasibility of Redundant Arrays of Independent Flash-based Solid-state drives (RAIS). We empirically analyze the RAIS performance using commercially-off-the-shelf (COTS) SSDs. We investigate select RAIS configurations under a variety of I/O access patterns. Finally, we present our comparisons of RAIS with a fast, PCIe-based COTS SSD, in terms performance and cost. As an illustrative observation, based on current market prices, RAIS arrays built using multi-level cell (MLC) SSDs can be cost-effective solutions and perform reasonably well compared to fast, PCIe-based SSDs, such as the Fusion-io device. This is particularly the case for workloads that are small, random, and heavy read I/O dominant.

1 Introduction

Hard disk drives (HDD) are the leading media in storage systems. HDDs are widely deployed from embedded to enterprise-scale systems for the last several decades. HDD manufacturers were successful in providing a continuous improvement in total disk capacity by increasing the storage density while bringing down the price-per-byte using mass production. Perpendicular recording [28] has continued this trend but further advances will require new technologies such as patterned media [2] which present significant manufacturing challenges. On the other hand, HDD I/O performance increased at a slower pace compared to the storage density. Increasing the platter rotational speed (rotations per minute – RPM) was key to this progress. A recent single enterprise-class magnetic disk today can provide up to 204 MB/s at 15,000 RPMs [39]. However, we are now at a point where HDD designers conclude it is extremely hard to increase platter RPM beyond its current state because of power consumption and thermal dissipation issues [11].

Flash memory-based solid state disks (SSD), especially NAND Flash, are another leading media in storage systems. Unlike magnetic rotational disks, NAND Flash memory-based SSDs have no mechanical moving parts, such as spindles and voice-coil motors. Therefore, NAND Flash memory technology offers a number of benefits over the conventional hard disk drives, such as lower power consumption, lighter weight, higher resilience to external shocks, ability to sustain hotter operating regimes, and smaller I/O access times. Additionally, since SSD Flash chips are packaged in HDD form factors and SSDs are compatible with HDD device drivers and I/O buses, one-to-one replacement of HDDs with SSDs is possible.

The Oak Ridge Leadership Computing Facility (OLCF) at Oak Ridge National Laboratory (ORNL) has deployed several HPC systems including Jaguar XT5 [32], the Cray XT5 [7] simulation platform. Along with the core HPC systems at OLCF, a center-wide parallel file system has been deployed, called "*Spider*" [41]. The Spider storage system has been provisioned with 13,440 hard disk drives to support over 2 petaflops of computing infrastructure. Currently, Spider has employed a RAID (Redundant Array of Independent Disk) level 6 scheme in order to fulfill the need to provide a highly reliable and available storage system, as well as high I/O throughput.

Although Redundant Arrays of Inexpensive (or Indepen-

dent) Disks (RAID) [36, 43] can provide high I/O throughput by exploiting parallelism across disks, the mechanical movement involved in the operation of HDDs (seek operations in HDDs, moving the heads back and forth) can limit the performance that an HDD-based system can offer to workloads with significantly high non-sequential I/O patterns. We envision that SSDs can overcome such key shortcomings of HDDs with faster access to non-sequential data, and are investigating this area for future storage systems.

However, a storage system designer needs to carefully consider the use cases of SSDs. In addition to limited lifetime issues (10K-1M erase cycles per block) on Flash [3], the high price of SSDs ($/GB) is of significant concern. Since SSDs are much more expensive per gigabyte than HDDs, replacing HDDs with SSDs in large-scale storage systems is not a cost-effective approach under current market prices [29]. Thus, conventional wisdom suggests using SSDs in existing large-scale storage systems to partially replace HDDs using SSDs or augmenting the HDDs with SSDs as a caching area.

There are several possible design approaches to employ SSDs in our Spider storage system: (i) We can use SSDs as an I/O accelerator to boost the overall I/O bandwidth, especially for workloads with significantly high randomness or less locality. (ii) SSDs can be used as a fast write caching tier that can absorb bulk sequential I/O traffic such as checkpointing. (iii) In previous work [33] we examined the efficacy of SSDs as file system journal targets.

SSDs are priced according to their Flash type (single-level cell (SLC) and multi-level cell (MLC) Flash chips), host interface (SATA and PCIe), and packaging technologies. SLC Flash chips store one bit of data per memory cell, while MLC Flash chips store multiple bits of data per memory cell [1]. SLC Flash chips provide faster read and write latency than MLC Flash chips [1]. MLC-based SSDs tend to have lower lifetimes when compared to SLC-based SSDs. SLC Flash chips can achieve lower access latency and high I/O throughput, but are more expensive than MLC Flash chips.

In our study, we focus on RAID configurations built using COTS SSDs [1]. Here are the salient contributions of our work:

- We investigated the feasibility of RAIS levels in terms of their performance behavior. One of the main shortcomings of SSDs is the slowdown during the garbage collection (GC) process that is hastened by small, random writes [23]. This slowdown can even further impact future incoming requests, We term this "*pathological behavior*" of an SSD by delaying I/O request services.
- We conducted a RAIS efficiency analysis. We used COTS components (SSDs and RAID controllers) in our experiments. We measured the I/O performance of each RAIS configuration under various I/O access patterns such as, small or large, random or sequential writes or reads. We investigated each configuration in terms of:
  1. I/O operations performed per second per dollar
  2. Bytes per second per dollar
  3. Capacity (gigabytes) per dollar.

  We also compared our RAIS configurations with a PCIe-interfaced SSD, in terms of performance, capacity, and cost.

The rest of this paper is organized as follows. We first present an overview of the Flash memory technology and its applications in storage devices in Section 2. Section 3 provides a brief of overview of the RAIS configurations used in this paper. Our experimental methodology is described in Section 4. We provide a performance analysis of SSDs and RAIS in Section 5.1 and 5.2. SSD pathological behavior due to GC processes and its effects on aggregate RAIS performance is discussed in Section 5.3. We present a cost-based efficiency analysis of RAIS in Section 5.4. We conclude in Section 6.

2 Background and Related Work

Flash memory-based storage devices require an erase operation [30], unlike rotating media and volatile memories which only need read and write operations, Characteristics of these operations include: Erase operations are performed at the granularity of a block which is composed of multiple pages. A page is the granularity at which reads and writes are performed. Each page on Flash can be in one of three different states: (i) *valid*, (ii) *invalid* and (iii) *free/erased*. When no data has been written to a page, it is in the erased state. A write can be done only to an erased page, changing its state to valid. Erase operations (on average 1.5ms) are significantly slower than reads/writes. Therefore, out-of-place writes (as different from in-place writes in HDDs) are performed to existing free pages along with marking the page storing the previous version invalid. Additionally, write latency can be higher than the read latency by up to a factor of 4-5. The lifetime of Flash memory is limited by the number of erase operations on its cells. Each memory cell typically has a lifetime of 10^3-10^9 erase operations [9]. *Wear-leveling* techniques [15, 18, 26, 3] are used to delay the wear-out of the first Flash block by spreading erases evenly across the blocks.

The Flash Translation Layer (FTL) is a software layer in an SSD that translates logical addresses from the file system into physical addresses on a Flash device. The FTL helps

[1] In the rest of this paper, we will use the term, Redundant Array of Independent SSDs (RAIS) for describing the SSD-based RAID sets. We analyzed RAIS-0, 5, and 6 configurations in our study, analog to conventional RAID-0, 5, and 6 sets.

in emulating Flash as a normal block device by performing out-of-place updates which in turn helps to hide the erase operations in the Flash media. The mapping table is stored in a small, fast SRAM. These FTLs can be implemented at different granularities in terms of how large an address space a single entry in the mapping table captures. Many FTL schemes [20, 6, 24, 17, 25, 38, 5] and their improvement by write-buffering [19] have been studied. A recent page-based FTL scheme called DFTL [10] utilizes temporal locality in workloads to overcome the shortcomings of the regular page-based scheme by storing only a subset of mappings (those likely to be accessed) on the limited SRAM and storing the remainder on the Flash device itself. Also, there are several works in progress on the optimization of buffer management in NAND Flash based environments [34, 14].

There are several empirical studies that test SSD performance and power consumption with respect to different access patterns [40, 4]. However, these studies are limited by the analysis of single devices, and did not extend to RAID configurations. There is recent work from Microsoft [16] that addresses the reliability concern in RAIS that every SSD could wear out simultaneously when configured in RAID-4 or 5. They proposed a new RAID variant, called "Diff-RAID" that maintains differential ages among devices, reducing the probability of correlated failures. However, their work is different from our work in that their work is not only based on an analytical modeling and simulation approach but also concerned about reliability issues in an SSD array.

We identified a lack of research on empirical studies of RAIS configurations. In order to fill the void, we built an experimental RAIS testbed for using real COTS SSDs and conducted comprehensive studies of different RAIS configurations (0, 5, and 6, analogous to traditional RAID-0, 5, and 6 configurations). We evaluated the RAIS configurations in terms of performance and cost efficiency.

There is another study published by Microsoft Research examining SSDs in enterprise storage systems [29]. They explored the cost-benefit trade-offs of various SSD and HDD configurations and concluded that SSDs cannot replace the existing HDDs because of current prices of SSDs. Our work is different from their work in that we conducted a cost efficiency analysis of COTS SSDs-based RAIS configurations against against COTS PCIe-based SSDs. We will discuss this in Section 5.4.

3 RAIS Overview

In our study, we have focused in a RAID storage using solid-state disk drives (SSDs) instead of hard disk drives (HDDs), called RAIS. We defined RAIS-0, 5, and 6 analog to RAID-0, 5, and 6 as follows:

RAIS Scheme	0, 5, 6
Write Cache	Write Through
Read Ahead	No
Direct I/O	Yes
Stripe Size	64KB

Table 1: Default settings of a LSI MegaRAID Controller.

- **RAIS-0**: A request is striped across multiple SSDs. As there is no redundantcy in the storage, data loss will occur if an SSD fails.
- **RAIS-5**: A request is striped across multiple SSDs with parity data across multiple SSDs. In RAIS-5, there is no dedicated parity SSD. Instead, the parity is distributed over all SSDs in a round-robin fashion, enabling writing data and parity blocks all the SSD in the array, protecting from a single SSD failure.
- **RAIS-6**: Different than RAIS-5, a request is striped with dual parity blocks over all SSDs. It is logically a combination of $n - 2$ data SSDs and 2 additional parity SSDs among n number of SSDs. It can protect data against any two SSD failures.

4 Experimental Methodology

All experiments were performed on a single server with 24 GB of RAM and an Intel Xeon Quad Core 2.93GHz CPU [12]. The operating system was Linux and used an Oracle (nee Sun) Lustre-patched 2.6.18-128 kernel. The *noop* I/O scheduler that implements FIFO queue was used [37]. The testbed had seven 8x PCIe slots and two of these were installed with PCIe RAID controller cards. We used two LSI MegaRAID SAS 9260-8i KIT RAID Adapters [27], each of which can support up to 8 SATA drives.

We used three representative SSDs in our evaluation. We selected the Super Talent 128GB FTM28GX25H SSD [42] as a representative of MLC-based SSDs with SATA interfaces and the Intel 64GB SSDSA2SH064G101 SSD [13] as a representative of SLC-based SSDs. We used a Fusion-io 640GB ioDrive Duo [8] as a representative of PCIe-based SSDs. We denote the SuperTalent MLC, Intel SLC, and the Fusion-io MLC devices as SSD(A), SSD(B), and SSD(C) in the remainder of this study, respectively. Their details are presented in Table 2.

SSD(A) and SSD(B) use SATA-II interfaces that can provide up to 375 MB/s data transfer rates, while SSD C utilizes a PCIe interface, which provides data transfers up to 2 GBytes/s. To compare the performance of SSD(C) with RAISs configured with SSD(A)s and SSD(B)s [2], we used two PCIe interfaced hardware RAID controllers for each

[2]Hereafter, we call RAIS(A) and RAIS(B) for RAIS configurations using SSD(A)s and SSD(B)s respectively.

Label	SSD(A)	SSD(B)	SSD(C)
Company	Super-Talent	Intel	Fusion-io
Model	FTM28GX25H	SSDSA2SH064G101	ioDrive Duo
Type	MLC	SLC	MLC
Interface	SATA-II	SATA-II	PCIe x8
Capacity (GB)	120	64	640
Price ($)	415	799	13,990
Erase (#)	10-100K	100K-1M	10-100K
Power (W)	1-2	1-2	-

Table 2: Storage device characteristics examined in our study. SSD prices are based on current market values (January 2010).

configuration. Each RAID controller can be equipped with up to 8 SATA-II interfaced SSDs. The default settings of the RAID controller are given in Table 1.

In order to minimize the skew in our data due to start-up effects, we "warmed-up" each SSD device prior to collecting data. At the beginning of each evaluation, all SSDs are exercised with an I/O pattern identical to that of the experiment. We repeated every experiment five times for all test cases.

To measure the I/O performance, we developed a benchmark tool that uses the `libaio` asynchronous I/O library on Linux. `libaio` provides an interface that can submit one or more I/O requests in one system call (`io_submit()`) without waiting for I/O completion. It also can perform reads and writes on raw block devices. We used the direct I/O interface to bypass the operating system I/O buffer cache by setting the `O_DIRECT` and `O_SYNC` flags in the file open call – `open()`. We measured the performance of our test configurations with random and sequential I/O access patterns by varying the amount of reads in the workloads. Although the definition of a "sequential" I/O access can be debated, our definition is simple: If a request starts at the logical address immediately following the last address accessed by the previously generated request, we consider it a sequential request. Otherwise, we classify it as a random request.

5 Experimental Results

We conducted two phases of experiments. In the first phase, we examined the performance of individual, single SSDs of type A, B, and C given in Table 2 to obtain the individual baseline performance of each device. We present these results in Section 5.1. In the second phase, we evaluated RAIS configured SSD performance. These results are presented in Section 5.2. Based on data obtained in these two phases, we investigate the pathological behavior of RAIS configured SSDs due to locally controlled and scheduled GC processes on each individual SSD. In Section 5.4 we present our insight on the factors to be considered when building storage systems with SSDs, such as which SSDs to buy, how to configure them, etc., based on our storage efficiency analysis.

5.1 Individual SSD Performance

5.1.1 Sequential I/O Performance

In this subsection we present experimental results on individual SSD devices using sequential I/O requests. We varied the request size and number of outstanding requests in the I/O queue.

In Figure 1(a)(b), we present the read performance characteristics of SSD(A) and (B). As can be seen, read bandwidth for each device increases with queue depth and request size. Maximum read performance is around 235 MB/s and 260 MB/s for SSD(A) and SSD(B), respectively. For large requests, both devices scale well. However, SSD(A) does not scale as well as SSD(B) with respect to queue depth when the request size is small. (Refer to Figure 1(a)(b)).

For reads, we found that SSD(B) performs better than SSD(A). This can be explained by the fact that SSD(B) uses SLC Flash chips while SSD(A) uses MLC Flash chips. Read access time on SLC Flash is faster than on MLC Flash [22, 35].

For sequential writes, as shown in Figure 1(d)(e), we can see that SSD(A) has a maximum throughput of 140 MB/s while SSD(B) performs at 175 MB/s. These write values are smaller than the read values for both devices. Flash writes are slower than reads in part because writes can cause garbage collection events (GCs). Similar to reads, we see that SSD(A) does not scale well compared to SSD(B) for small write requests.

In Figure 1(c)(f), we see that SSD(C) achieves a maximum read bandwidth of 1.38 GB/s and its maximum write bandwidth is around 1 GB/s. These two values are much higher than their SSD(A) and (B) counterparts.

Overall, for reads as shown in Figure 1(a)(b)(c), we observe that variance is high at several measurement points. We speculate that following factors may have caused the

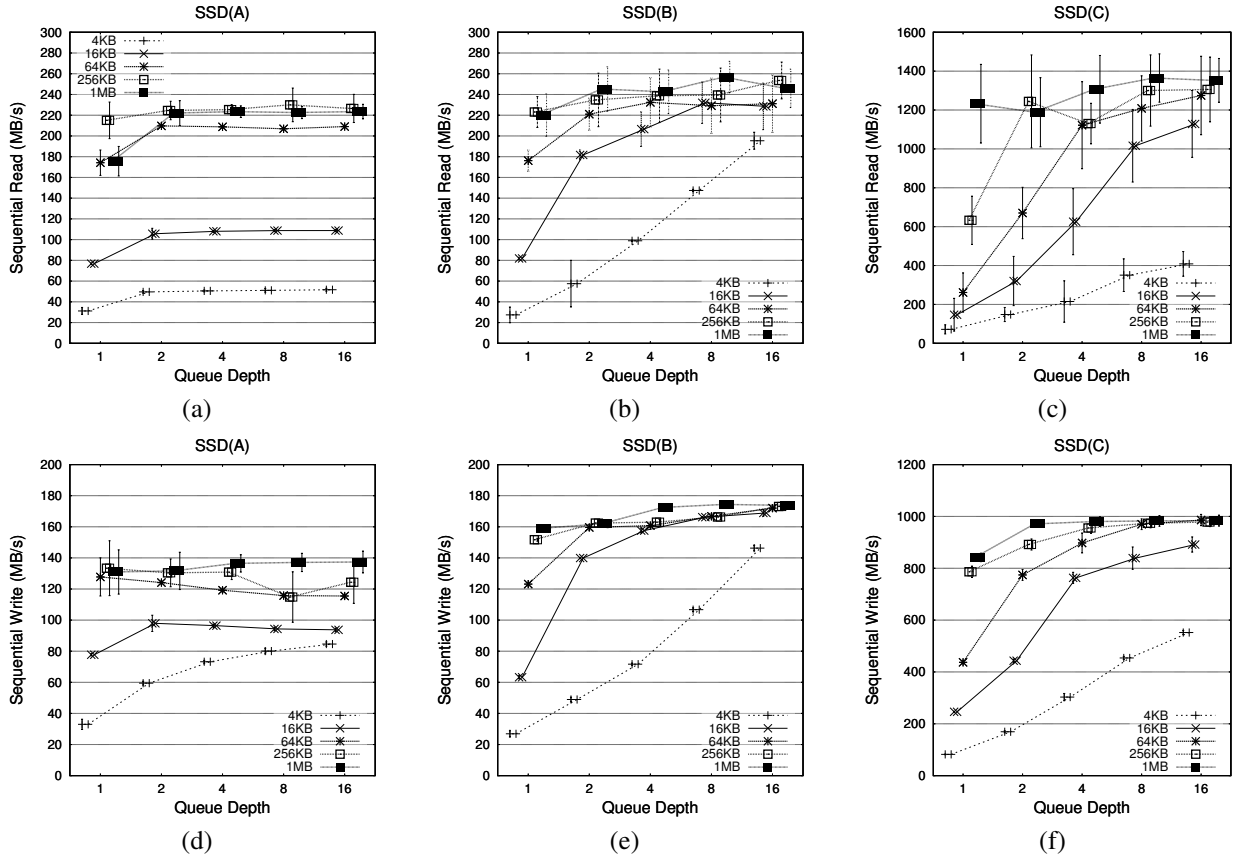


Figure 1: Observed sequential I/O throughput of single, individual SSD(A), (B), and (C) devices, using the libio benchmark. 'QD' (Queue Depth) denotes the number of outstanding requests in an I/O queue. The error bars show 95% confidence intervals. Note that some intervals are too narrow to be apparent here.

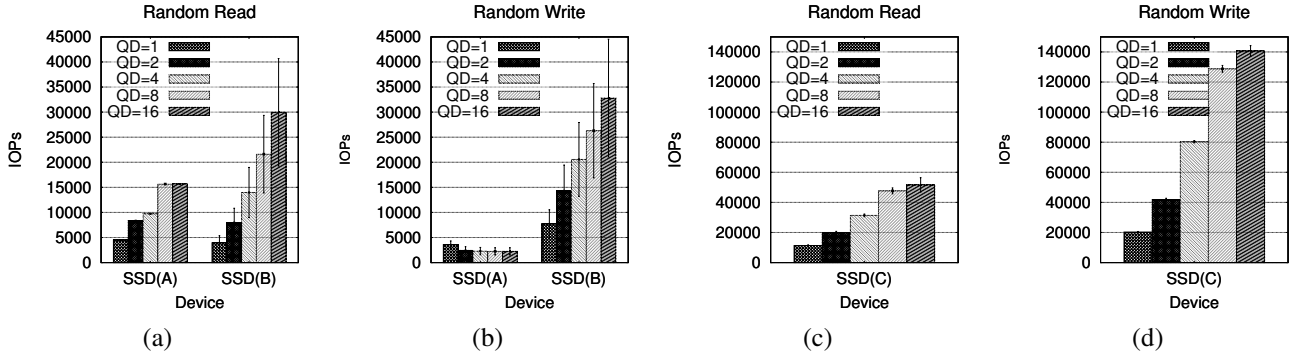


Figure 2: Observed random I/O throughput single, individual of SSD(A), (B), and (C) devices. Error bars show 95% confidence intervals. I/O Operations Per Second (IOPS) were measured by generating 4 KB read and write requests that were random in a 1 GB logical address space.

higher variance in read measurements:

- Garbage collection processes (GCs) can be triggered even in read-dominant workloads (possibly because of past write-dominant workloads). GCs can increase the queueing delay (the time a request waits in an I/O queue) of incoming requests, resulting high request service latencies.
- Devices might have different internal fragmentation lev-

els, which can directly impact the scheduling of GC processes [4].

Unfortunately, there is no easy way to query the internal fragmentation status of SSDs. We investigate this pathological behavior more in Section 5.3.

5.1.2 Random I/O Performance

Figure 2 shows the performance characteristics of SSD(A), (B), and (C) for 4 KB random I/O access patterns with respect to varying queue depth for each case. We use IOPs as a metric for comparing random I/O performance in the rest of this paper. IOPs is one of the widely used metrics in benchmarking storage systems for random I/O access patterns.

In Figure 2(a)(b), we see that SSD(B) outperforms SSD(A) for both random reads and writes. For 4 KB random reads on both devices, we see that I/O throughput increases with respect to the queue depth. On 4 KB random writes, surprisingly, we see that SSD(A) does not scale at all with respect to queue depth. This behavior was not observed for SSD(B). We also see that SSD(B) performs significantly better than SSD(A) in terms of IOPs, even though we observed a high variance in SSD(B) results. This high variance on random writes can be explained by: (i) In our experiment, we used a simple definition for sequential/random requests in workloads. However, random requests can be further divided by a distance of successive requests, which can differently impact the internal fragmentation level of an SSD and its scheduling of GCs. We have seen this similar behavior in our previous work [21]. We leave a more detailed investigation of this issue as future work. (ii) We believe internal caching policies (prefetching or write-buffering) can perform differently based on manufacturers' design choices.

Interestingly, SSD(B) and SSD(C) (Figure 2(c)(d)) perform better in terms of IOPs under a random write heavy workload compared to a random read heavy workload, while SSD(A) exhibits lower throughput on random writes than on random reads. Poor random writes has been one of the biggest problems to overcome in SSDs [23, 10].

We suspect that SSD(B) and (C) were designed to provide better performance on random writes. We speculate that the manufacturer of SSD(B) and (C) might have employed an enhanced write-buffering scheme [19] or employed an enhanced FTL scheme optimized for random writes [10].

5.2 RAIS SSD Performance

In our experiments, we used 4 SSDs for the RAIS-0 configuration. Since RAIS-5 and 6 configurations need extra parity drives, we used 5 SSDs (4 data drives and 1 parity drive) for RAIS-5 and 6 SSDs (4 data drives and 2 parity drives) for RAIS-6. All SSDs were connected to the RAID controller via SATA-II interfaces. Settings for the RAID controller are presented in Table 1.

5.2.1 Sequential I/O Performance

Here we present our results of performance testing on RAIS(A) and RAIS(B) with respect to a variety of I/O access patterns and I/O queue depths.

Overall, we see that bandwidth increases with respect to queue depth for all RAIS configurations. Figure 3(a)(b)(c) shows sequential read bandwidth for RAIS(A). Surprisingly, we observe that the bandwidth for the RAIS-5(A) configuration is higher than RAIS-0(A). The LSI RAID controller may be able to do RAID calculations at line rate, faster than the disk bandwidth. We also see that the RAIS-6(A) configuration performs even better than RAIS-5(A). From the figures, we see RAIS-0(A), RAIS-5(A), and RAIS-6(A) provide up to 850 MB/s, 985 MB/s, and 1190 MB/s, respectively. Based on our understanding in RAID theory, we did not expect the RAIS-5 and 6 configurations to perform better than RAIS-0 due to the extra computational power and bandwidth required in processing parity blocks, given that all three RAIS configurations have the same number of data disks. We had a similar observation on RAIS(B) setups. In Figure 3(g)(h)(j), we see that the bandwidth of RAIS-5(B) drops slightly down to 917 MB/s, which is lower than RAIS-0(B) by 144 MB/s, and reaches 1,088 MB/s, which is slightly beyond the 1,061 MB/s peak of RAIS-0(B).

We found that, unlike HDDs, more parallelism can be exploited for a given SSD as we increase the number of devices employed in RAIS sets [31]. Recently marketed SSDs use 4 or 8-way data channels, enabling them to process multiple requests simultaneously at different Flash chips. We infer that, in RAIS, individual SSDs can exploit more parallelism by interleaving requests over individual SSDs as the total number of SSDs drives in an array increases, regardless of the RAIS configuration. Moreover, the performance benefit due to this parallelism may outweigh the extra parity calculation overhead in RAIS-5 and 6 configurations.

5.2.2 Random I/O Performance

For random reads as shown in Figure 4(a)(c), we see both configurations' IOPs performance increase with respect to queue depth and request size, as we expected, from the individual SSD performance given in Figure 2. Among different RAIS schemes, we can not see any significant performance differences.

For random writes, Figure 4(b)(d), we see that RAIS-0(A) performs poorly compared to RAIS-0(B). This is congruent with our observations illustrated in Figure 2. We

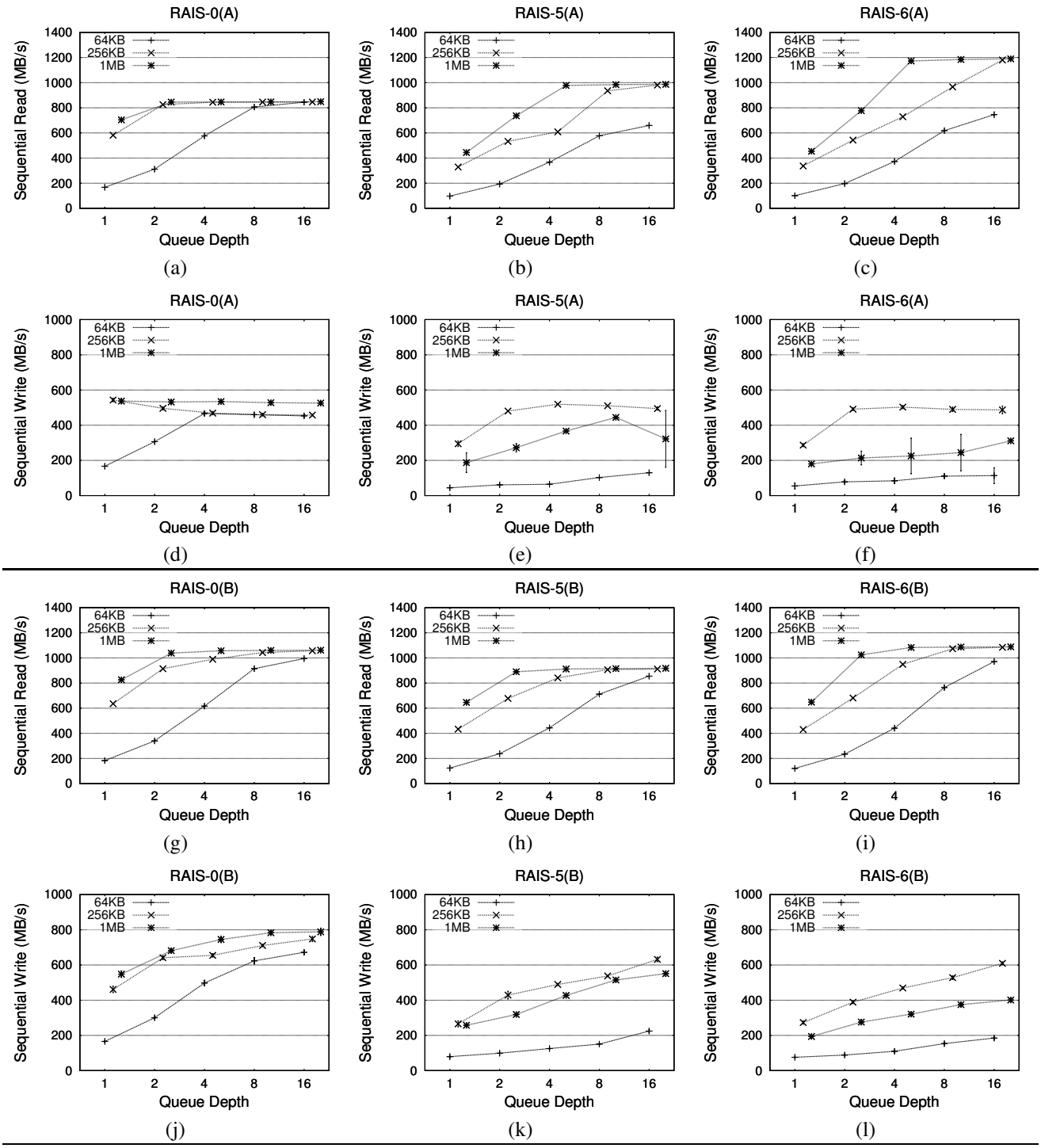
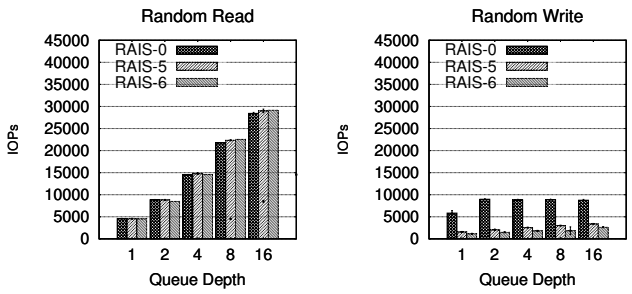


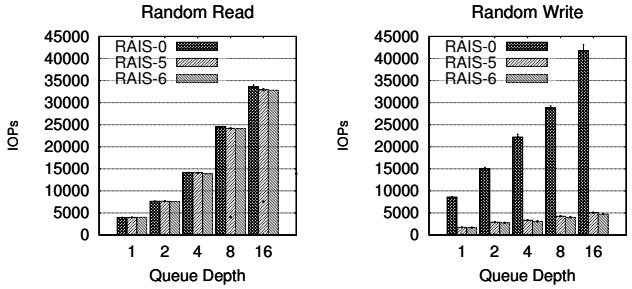
Figure 3: Test results of RAIS(A) and (B) for sequential I/O access patterns. Note that RAIS-0, 5, and 6 are configured with 4 SSDs, 5 SSDs including 1 parity drive, and 6 SSDs including 2 parity drives, respectively. The error bars show 95% confidence intervals. Note that some intervals are too narrow to be apparent here.

have seen that SSD(A) shows much lower bandwidth than SSD(B) for random writes dominant workloads. Also, interestingly, we see that RAIS-0(B) performs much better compared to RAIS-0(A) with respect to queue depth. How-



(a) RAIS(A)

(b) RAIS(A)



(c) RAIS(B)

(d) RAIS(B)

Figure 4: Performance of RAIS(A) and (B) sets under random I/O access patterns. The error bars show 95% confidence intervals.

ever, as can be seen in Figure 4(b)(d), RAIS-5 and 6 do not scale as we increase queue depth. Our knowledge about RAIS schemes coupled with these observations suggests that extra random writes due to parity block updates may be degrading performance.

5.3 Pathological Behavior

HDDs update data in-place, while SSDs are designed to provide out-of-place updates to hide the erase operation latency on their Flash devices. Out-of-place update operations trigger GC processes that can delay the servicing of incoming requests while invalid pages that are stale are collected and space is freed for new writes. In this section, we investigate the slowdowns on SSDs and RAIS performance due to GC processes. To illustrate this effect, we conducted analyses on bandwidth patterns in time series.

In this experiment, we used the following RAIS configurations: We configured 6 SSDs for RAIS-0 in order to observe the maximum achievable throughput. For RAIS-5, we used one parity drive out of 6 SSDs and the remaining 5 SSDs were used for storing actual data. In RAIS-6, 2 parity SSDs and 4 data storage SSDs were used.

Type	Metric	Write (%) in Workload			
		80	60	40	20
SSD(A)	Avg	162.6	179.1	205.6	225.9
	Stdev	10.9	7.2	4.2	3.1
SSD(B)	Avg	214.3	229.4	249.3	259.6
	Stdev	23.7	16.4	9.6	6.3

Table 3: Average and standard deviations of the values in Figure 5.

Type	Metric	RAIS Scheme		
		0	5	6
RAIS(A)	Avg	1109.4	609.4	612.5
	Stdev	33.6	60.8	37.6
RAIS(B)	Avg	1386.3	837.2	737.6
	Stdev	98.2	92.4	72.6

Table 4: Average and standard deviations of the values in Figure 6.

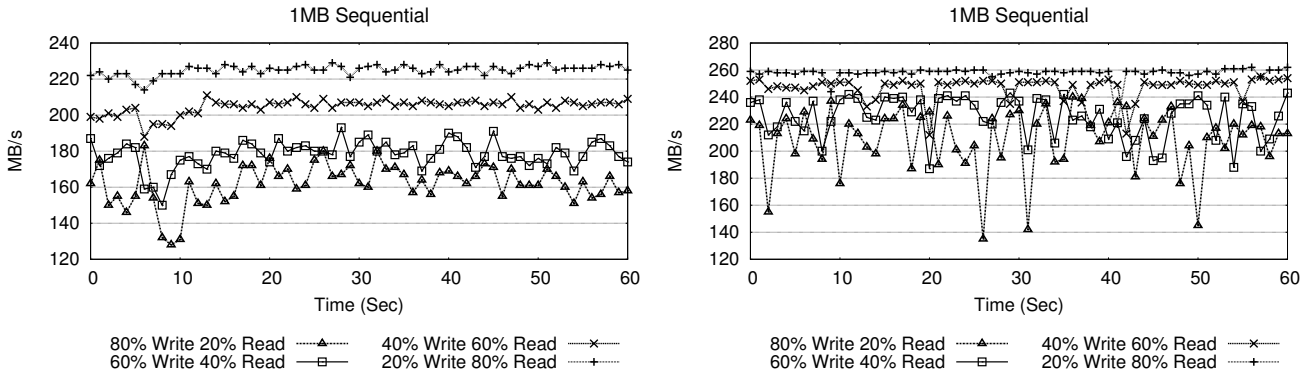
5.3.1 Performance Anomaly on Individual SSDs

In Figure 5(a)(b), we examine the large sequential I/O bandwidth responses of individual SSDs in time series. We varied the percentage of writes in workloads between 20% and 80% in increasing steps of 20%. We measured I/O bandwidth in one second intervals.

For write-dominant workloads, we observe that the bandwidth fluctuates widely due to excessive GCs . For example, the SSD(A) I/O throughput drops below 180MB/s at the 6^{th} and 7^{th} seconds under an 80% write workload. However, I/O throughput drops below 160MB/s for the 8^{th} second and then drops further to 130MB/s in the next 3 seconds. Overall SSD(B) shows higher bandwidth than SSD(A). Also, surprisingly, SSD(B) has a higher variance than SSD(A). (Refer to Table 3). For instance, SSD B's I/O throughput reached 240MB/s at the peak and dropped to 140MB/s (at 25^{th} to 27^{th} seconds). As we increased the amount of reads in the workloads from 20% to 80%, we observed that SSD(A)'s and B's I/O throughput increased around 50% and 18%, respectively.

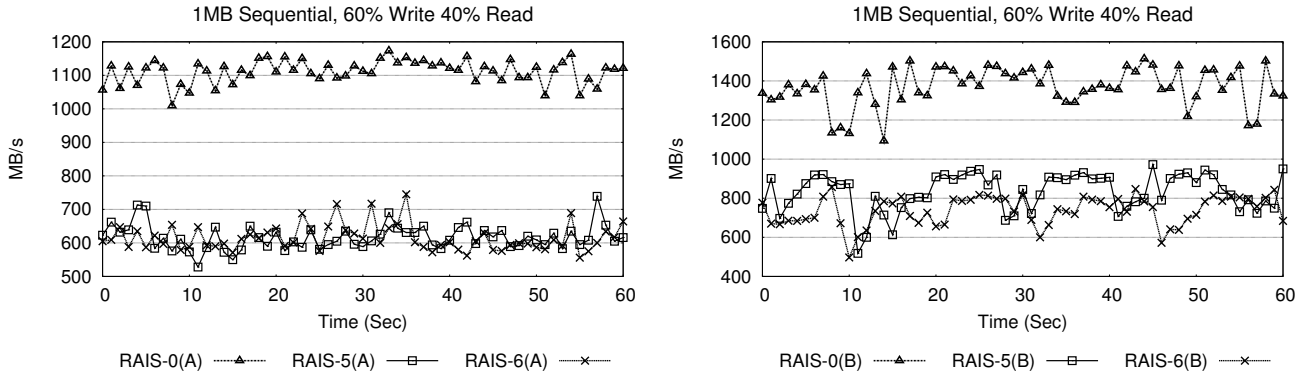
5.3.2 Pathological Behavior with RAIS

In Figure 6(a)(b) we show results for a workload mix of 60% writes and 40% reads. Similar to previous observations from test results on individual SSDs, we see that the bandwidth variance of RAIS(B) is higher than RAIS(A) (see Table 4), even though RAIS(B) provides higher bandwidth than RAIS(A). For example, we see that RAIS-0(B) reached 1.5 GB/s at the peak and then dropped to 1.2 GB/s. RAIS-0(A) varied between 1 GB/s and 1.15 GB/s. Also, we see from the results on RAIS-5 and RAIS-6 that their overall performance is much lower than RAIS-0 because of the



(a) SSD(A) (b) SSD(B)

Figure 5: Pathological behavior of single SSDs.



(a) RAIS(A) (b) RAIS(B)

Figure 6: Pathological behavior of RAIS.

extra operation for parity block updates. However, we did not observe a large difference between RAIS-5 and RAIS-6 performance, specifically for RAIS(A) (see the average and standard deviation values of RAIS(A) for RAIS-5 and 6 in Table 4). RAIS-5(A) and RAIS-6(A) performed identically as can be seen in Figure 6(a). Under current RAIS setups, GC scheduling is controlled locally by internal SSD logic. In RAIS, the lack of a global GC scheduling mechanism can cause overall synchronization problems across SSDs, reducing the efficiency of the RAIS set.

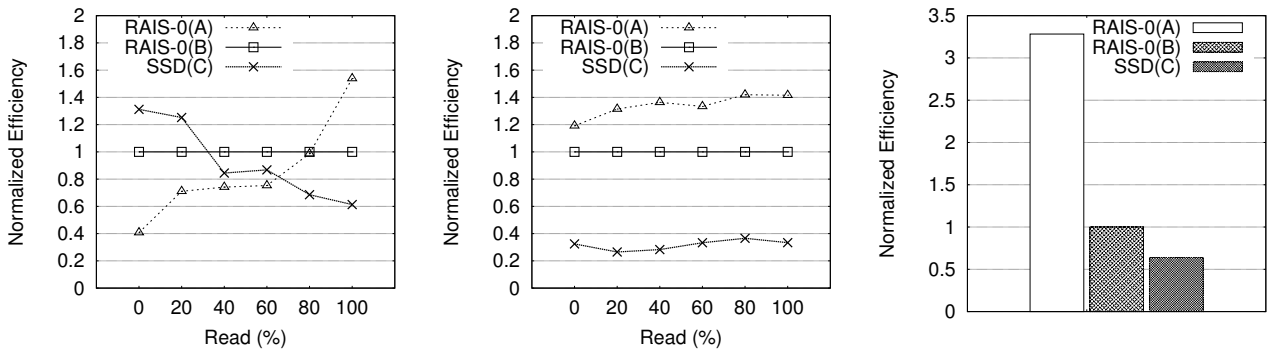
5.4 RAIS Efficiency Analysis

Unlike HDDs, SSDs are expose more flexible design spaces to manufacturers, as they do not require mechanical moving parts such as spindle and voice-coil motors. As described in Table 2, an SSD can be designed in different ways – from its internal design (e.g. internal memory size, firmware, and Flash chips) to external interface (SATA, PCI, etc.) [31]. Thus, SSDs show differences in market prices.

5.4.1 Efficiency Metrics

To do a complete comparison among RAIS configurations, performance (i.e. MB/s, IOP/s) and capacity (GB) are not sufficient metrics. We need additional efficiency metrics including performance and capacity per dollar. Performance should be measured as IOPs per second per dollar or bytes per second per dollar. Systems can also be compared based on total cost, total formatted capacity, and performance. Also, the primary metric for comparing different systems should be dependent on applications. In transaction processing applications, more requests are small and random. The right metric for this type of applications should be IOPs per second per dollar. On the other hand, media servers process sequential I/O requests to provide streaming services where bytes per second per dollar might be the correct metric.

We conducted experiments to answer the question, “can RAIS built on COTS SATA SSDs be competitive against a large, fast, PCIe SSD, such as the Fusion-io device?” To an-



(a) IOP/s per dollar (b) MB/s per dollar (c) GB per dollar

Figure 7: Storage efficiency comparison in terms of performance (in IOP/s per $ and MB/s per $) and storage capacity (in GB per $). SSD prices are presented in Table 2. In addition, we also included the RAID controller card price in RAIS configurations for a fair comparison. All values in the graphs were normalized with respect to RAIS-0(B).

IOPs	Read (%) in Workload					
	0	20	40	60	80	100
RAIS-0(A)	9,472	13,827	12,614	11,146	17,844	28,416
RAIS-0(B)	40,960	34,059	29,787	25,918	31,656	32,256
SSD(C)	140,544	111,050	65,509	58,571	56,499	51,456

MB/s	Read (%) in Workload					
	0	20	40	60	80	100
RAIS-0(A)	792	1,085	1,099	1,104	1,231	1,259
RAIS-0(B)	1,165	1,445	1,412	1,449	1,518	1,557
SSD(C)	983	998	1,040	1256	1,445	1,506

Table 5: Average performance values used in Figure 7(a)(b).

swer this question, we configured 6 SSD(A) and 6 SSD(B) devices in RAIS-0(A) and RAIS-0(B) configurations, respectively. We analyzed and compared their performance and storage efficiency against a single Fusion-io device (denoted as SSD(C)). Unlike SSD(C), RAIS-0(A) and (B) configurations require an additional RAID controller to configure the array. We also included the cost of this extra RAID controller in our calculations and metrics. The details of the test configurations are described in Table 6

5.4.2 Performance Cost Efficiency Analysis

Figure 7(a)(b) compares the efficiency of performance and storage capacity for our test configurations. Performance values that we considered are shown in Table 5. Since writes can be slower than reads for SSDs, we varied the amount of reads in the workload. In Figure 7(a), IOP/s per dollar has been used as a metric to compare the performance efficiency for workloads where small random requests are dominant. On write-dominant access patterns (e.g. 0-20% reads), we observed that SSD(C) outperforms RAIS-0(A) by 300% and RAIS-0(B) by 30%. However, we see that the performance efficiency of SSD(C) decreases while that of RAIS-0(A) increases as the mix of reads in the workload increases. Interestingly, we see that RAIS-0(A) and SSD(C) perform similarly around 60% reads and RAIS-0(A) is more economically efficient than SSD(C). Moreover, SSD(C) is less efficient than RAIS-0(B) as reads in workload exceed 40%. On read-dominant access patterns, RAIS-0(A) has twice the efficiency of SSD(C).

Figure 7(b) presents our comparisons on throughput efficiency of our storage configurations for large sequential access patterns. We used MB per second per dollar as a metric for estimating throughput efficiency. Unlike our previous observation, we see that the throughput efficiency does not change regardless of percentage of reads in the workload. Surprisingly, SSD(C) shows much lower efficiency than both RAIS configurations. Also we observe that the efficiency of RAIS-0(A) increases by 10-20% as reads become more dominant in the workload. Overall, we observed that RAIS configurations are more efficient than SSD(C) for workloads with large sequential I/O access patterns. For the

Type	RAIS Scheme	RAIS Cap. (GB)	Device (#)	RAID Controller (#)
RAIS-0(A)	(Striping)	768	6	1
RAIS-0(B)	(Striping)	384	6	1
SSDC()	-	640	1	0

Table 6: Test configurations. The price of the additional RAID controller is $579 (based on the current market price (January 2010)).

devices we considered, based on their current market prices, we see that I/O workload characteristics should be considered and for different workloads different storage solutions might be optimal.

5.4.3 Storage Capacity Cost Efficiency Analysis

In Figure 7(c), we compare the storage capacity efficiency based on current market prices shown in Table 2. We see that RAIS-0(A) is the most efficient storage configuration irrespective of I/O access patterns.

6 Summary and Future Work

We conducted a comprehensive empirical study on SSDs and RAIS configurations in terms of performance and cost efficiency. We also studied the performance anomalies in SSDs and RAISs that can be caused by the local garbage collection (GC) processes of individual SSDs. From I/O throughput analyses in time series, we identified a shortcoming that future RAIS sets should overcome: RAIS can not provide sustained bandwidth due to the independent, local GC processes of individual SSDs. Such local processes cause synchronization problems in RAIS configurations, degrading the overall I/O throughput. Moreover, we performed a cost efficiency analysis of various SATA RAIS configurations against a fast, but expensive PCIe SSD. We used mixes of four different major I/O access patterns (i.e. large sequential reads and writes, and small random reads and writes) in our tests and we observed that, under the current market prices of such devices, a RAIS-0 configuration using MLC based SSDs may currently be the most cost-effective option for workloads that are dominated by small random reads or large sequential reads.

Our work is still in progress. Our results are based on micro-benchmark tests and we performed performance tests on devices of interest for representative I/O access patterns. However, a more realistic approach will be to conduct file system level performance tests and cost efficiency analyses with higher-level benchmarks and applications. Our future plans include deploying RAIS configurations as a read and write cache that can be placed above HDD tiers in an experimental testbed.

Acknowledgements

We would like to thank Jason J. Hill for his technical support on the testbed setup and Raghul Gunasekaran for his corrections on the paper. Also, we are grateful to the anonymous reviewers for their detailed comments which helped us improve the quality of this paper.

This research is sponsored by the Mathematical, Information, and Computational Sciences Division, Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC. Also sponsored by the Department of Defense and used elements of the Extreme Scale Systems Center at ORNL.

References

[1] G. Atwood, A. Fazio, D. Mills, and B. Reaves. Intel StrataFlash^{TM} Memory Technology Overview. *Intel Technology Journal*, 1997.

[2] Z. Z. Bandic, D. Litvinov, , and M. Rooks. Nanostructured materials in information storage. *MRS Bulletin*, 33(9), 2008.

[3] Y.-H. Chang, J.-W. Hsieh, and T.-W. Kuo. Endurance Enhancement of Flash-memory Storage Systems: An Efficient Static Wear Leveling Design. In *Proceedings of the 44th Annual Conference on Design Automation*, pages 212–217, New York, NY, USA, 2007. ACM.

[4] F. Chen, D. A. Koufaty, and X. Zhang. Understanding Intrinsic Characteristics and System Implications of Flash Memory based Solid State Drives. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems (SIGMETRICS)*, pages 181–192, 2009.

[5] S. Choudhuri and T. Givargis. Performance improvement of block based NAND flash translation layer. In *Proceedings of the Fifth IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 257–262, 2007.

[6] T. Chung, D. Park, S. Park, D. Lee, S. Lee, and H. Song. System Software for Flash Memory: A Survey. In *Proceedings of the International Conference on Embedded and Ubiquitous Computing*, pages 394–404, August 2006.

[7] Cray Inc. Cray XT5. `http://cray.com/Products/XT/Systems/XT5.aspx`.

[8] Fusion-io 640 GB ioDrive Duo. `http://www.fusionio.com/products/iodriveduo/`.

[9] E. Gal and S. Toledo. Algorithms and Data Structures for Flash Memories. *ACM Computing Survey*, 37(2):138–163, 2005.

[10] A. Gupta, Y. Kim, and B. Urgaonkar. DFTL: A Flash Translation Layer Employing Demand-based Selective Caching of Page-level Address Mappings. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating System (ASPLOS)*, pages 229–240, March 2009.

[11] S. Gurumurthi, A. Sivasubramaniam, and V. Natarajan. Disk Drive Roadmap from the Thermal Perspective: A Case for Dynamic Thermal Management. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 38–49, June 2005.

[12] Intel. Intel Xeon Processor X5570 (8M Cache, 2.93 GHz, 6.40 GT/s Intel QPI). `http://ark.intel.com/Product.aspx?id=37111`.

[13] Intel. Intel X25-E Extreme 64GB SATA Solid-State Drive SLC. `http://www.intel.com/design/flash/nand/extreme/index.htm`.

[14] H. Jo, J. Kang, S. Park, J. Kim, and J. Lee. FAB: Flash-Aware Buffer Management Policy for Portable Media Players. *IEEE Transactions on Consumer Electronics*, 52(2):485–493, 2006.

[15] D. Jung, Y. Chae, H. Jo, J. Kim, and J. Lee. A Group-based Wear-Leveling Algorithm for Large-Capacity Flash Memory Storage Systems. In *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, pages 160–164, September 2007.

[16] A. Kadav, M. Balakrishnan, V. Prabhakaran, and D. Malkhi. Differential RAID: Rethinking RAID for SSD Reliability. In *Proceedings of the First Workshop on Hot Topics in Storage and File Systems (HotStorage)*, 2009.

[17] J. Kang, H. Jo, J. Kim, and J. Lee. A Superblock-based Flash Translation Layer for NAND Flash Memory. In *Proceedings of the International Conference on Embedded Software (EMSOFT)*, pages 161–170, October 2006.

[18] A. Kawaguchi, S. Nishioka, and H. Motoda. A Flash-Memory based File System. In *Proceedings of the Winter 1995 USENIX Technical Conference*, pages 155–164, 1995.

[19] H. Kim and S. Ahn. BPLRU: A Buffer Management Scheme for Improving Random Writes in Flash Storage. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, pages 1–14, Feburary 2008.

[20] J. Kim, J. Kim, S. Noh, S. Min, and Y. Cho. A Space-Efficient Flash Translation Layer for Compactflash Systems. *IEEE Transactions on Consumer Electronics*, 48(2):366–375, 2002.

[21] Y. Kim, A. Gupta, and B. Urgaonkar. MixedStore: An Enterprise-scale Storage System Combining Solid-state and Hard Disk Drives. In *Technical Report (TR 08-017), Dept. of Computer Science and Engineering, The Pennsylvania State University*, August 2008.

[22] S. Lee, K. Ha, K. Zhang, J. Kim, and J. Kim. Flexfs: A flexible flash file system for mlc nand flash memory. In *Proceedings of the USENIX Annual Technical Conference*, 2009.

[23] S. Lee and B. Moon. Design of Flash-based DBMS: An In-Page Logging Approach. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 55–66, August 2007.

[24] S. Lee, D. Park, T. Chung, D. Lee, S. Park, and H. Song. A Log Buffer based Flash Translation Layer Using Fully Associative Sector Translation. *IEEE Transactions on Embedded Computing Systems*, 6(3):18, 2007.

[25] S. Lee, D. Shin, Y. Kim, and J. Kim. LAST: Locality-Aware Sector Translation for NAND Flash Memory-based Storage Systems. *SIGOPS Oper. Syst. Rev.*, 42(6):36–42, 2008.

[26] K. M. J. Lofgren, R. D. Norman, G. B. Thelin, and A. Gupta. Wear Leveling Techniques for Flash EEPROM. *United States Patent, No 6,850,443*, 2005.

[27] LSI. MegaRAID SAS 9260-8i RAID Card. `http://www.lsi.com/channel/products/megaraid/sassata/9260-8i/index.html`.

[28] M. Mallary, A. Torabi, and M. Benakli. One Terabit Per Square Inch Perpendicular Recording Conceptual Design. *IEEE Transactions on Magnetics*, 38(4):1719–1724, July 2002.

[29] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron. Migrating Enterprise Storage to SSDs: Analysis of Tradeoffs. In *Proceedings of the ACM European Conference on Computer Systems (Eurosys)*, pages 145–158, March 2009.

[30] H. Niijima. Design of a Solid-State File Using Flash EEPROM. *IBM Journal of Research and Developement*, 39(5):531–545, 1995.

[31] A. Nitin, P. Vijayan, and W. Ted. Design Tradeoffs for SSD Performance. In *Proceedings of the USENIX Annual Technical Conference*, pages 57–70, June 2008.

[32] Oak Ridge National Laboratory, National Center for Computational Sciences. Jaguar. `http://www.nccs.gov/jaguar/`.

[33] S. Oral, F. Wang, D. Dillow, G. Shipman, and R. Miller. Efficient Object Storage Journaling in a Distributed Parallel File System. In *Proceedings of the Annual Conference on File and Storage Technology (FAST'10)*, February 2010.

[34] S. Park, D. Jung, J. Kang, J. Kim, and J. Lee. CFLRU: A Replacement Algorithm for Flash Memory. In *Proceedings of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, pages 234–241, New York, NY, USA, 2006. ACM.

[35] S. Park, J. Park, J. Jeong, J. Kim, and S. Kim. A Mixed Flash Translation Layer Structure for SLC-MLC Combined Flash Memory System. In *Proceedings of the 1th International Workshop on Storage and I/O Virtualization, Performance, Energy, Evaluation and Dependability (SPEED2008)*, 2008.

[36] D. Patterson, G. Gibson, and R. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *Proceedings of ACM SIGMOD Conference on the Management of Data*, pages 109–116, June 1988.

[37] S. L. Pratt and D. A. Heger. Workload Dependent Performance Evaluation of the Linux 2.6 I/O Schedulers. In *Linux Symposium*, July 2004.

[38] A. Rajimwale, V. Prabhakaran, and J. D. Davis. Block management in solid-state devices. In *Proceedings of the USENIX Annual Technical Conference*, 2009.

[39] Seagate. Seagate cheetah 15k.7 disc drive. `http://www.seagate.com/docs/pdf/datasheet/disc/ds_cheetah_15k_7.pdf`.

[40] E. Seo, S. Park, and B. Urgaonkar. An Empirical Analysis of the Energy Efficiency of Flash-based SSDs. In *Proceedings of the First Workshop on Power-Aware Computing and Systems (HotPower)*, December 2008.

[41] G. M. Shipman, D. A. Dillow, S. Oral, and F. Wang. The Spider Center Wide File Systems; From Concept to Reality. In *Proceedings of the 2009 Cray User Group (CUG)*, 2009.

[42] Super Talent. Super Talent 128GB UltraDrive ME SATA-II 25 MLC. http://www.supertalent.com/products/ssd_detail.php?type=UltraDrive%20ME.

[43] R. Youssef. RAID for Mobile Computers. Master's thesis, Carnegie Mellon University Information Networking Institute, August 1995.