

# An empirical study of redundant array of independent solid-state drives (RAIS)

Youngjae Kim

Received: 25 October 2013 / Revised: 22 September 2014 / Accepted: 2 January 2015 / Published online: 31 January 2015  
© Springer Science+Business Media New York 2015

**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 inside-SSD write operations and the amount of data written to it. GC scheduling is locally controlled by an 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 the performance of various RAIS configurations under a variety of I/O access patterns. Finally, we present our performance and cost comparisons of RAIS with a fast, PCIe-based COTS SSD, in terms of performance and cost.

**Keywords** Storage systems · Performance measurement and analysis · Flash memory · SSD · RAID

## 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 [35] has continued this trend but further advances will require new technologies such as patterned media [3] that 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 a key to this progress. A recent single enterprise-class magnetic disk today can provide up to 204 MB/s at 15,000 RPMs [45]. 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 storage media in storage systems. Unlike magnetic rotational disks, NAND Flash 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 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.

Although Redundant Arrays of Inexpensive (or Independent) Disks (RAID) [42, 50] 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

---

Y. Kim (✉)  
Department of Information and Computer Engineering,  
Ajou University, Suwon 443-749, South Korea  
e-mail: youkim@gmail.com

with faster access to non-sequential data, and are investigating this area for future storage systems.

Despite these benefits of SSDs over HDDs, storage system designer need to carefully consider the use cases of SSDs. In addition to limited lifetime issues (10K–1M erase cycles per block) on Flash [4], 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 [36]. 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 [24].

There are several possible design approaches to employ SSDs in a HDD in high performance storage systems:

- 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.
- 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 [38] we examined the efficacy of SSDs as file system journal targets.

In our study, we focus on RAID configurations built using COTS SSDs.<sup>1</sup> 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 [5, 28, 30, 41]. This slowdown can even further impact future incoming requests, We term this “*pathological behavior*” of an SSD by delaying I/O request services.
- We further studied the pathological behavior of the arrays of SSDs by extending the Microsoft SSD simulator, which we expect to see more interesting results with realistic workloads. Also we studied the effect of GC coordination for the aggregate performance of the SSD arrays. Our results show that the GC coordination can offer a significant performance improvement over the SSD arrays without the GC coordination.
- 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 Sect. 2. Section 3.1 provides a brief of overview of the RAIS configurations used in this paper. Our experimental methodology is described in Sect. 3.2. We provide a performance analysis of SSDs and RAIS in Sects. 4.1 and 4.2. SSD pathological behavior due to GC processes and its effects on aggregate RAIS performance is discussed in Sect. 4.3. In Sect. 4.4, we present preliminary simulation results of improvement by a GC coordination technique across SSDs in the array. We present a cost-based efficiency analysis of RAIS in Sect. 4.5. Related works are discussed in Sect. 5, and we conclude in Sect. 6.

## 2 Background

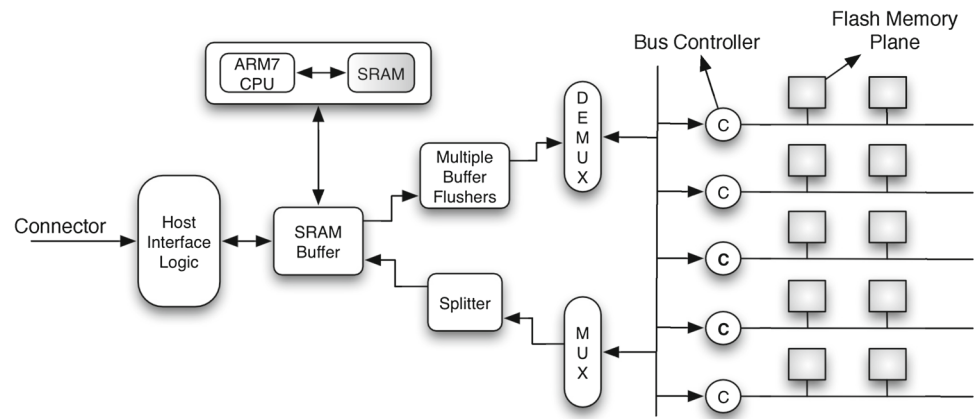
### 2.1 Flash memory technology

Flash memory-based storage devices require an erase operation [37], and page granularity based read and write operations, unlike rotating media and volatile memories. 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* [9]. 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 1.5 ms) 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 [4, 17, 20, 33] are used to delay the wear-out of the first Flash block by spreading erases evenly across the blocks.

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,

<sup>1</sup> 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 present our analysis results on comparing RAIS-0, 5, and 6 configurations in our study, analogous to conventional RAID-0, 5, and 6 sets.

**Fig. 1** A block diagram of flash based solid state disk drive [46]



while MLC Flash chips store multiple bits of data per memory cell [2]. SLC Flash chips provide faster read and write latency than MLC Flash chips [2]. 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.

## 2.2 Characteristics of flash memory operations

### 2.2.1 In-place versus out-of-place updates

In flash memory, in-place update operations are very costly. Since an erase occurs at the block granularity whereas writes are done to pages, an in-place update to a page entails (i) reading all valid pages of the block into a buffer, (ii) updating the required page, (iii) erasing the entire block and (iv) then writing back all the valid pages to the block. Instead, faster out-of-place updates are employed that work as follows: An out-of-place update invalidates the current version of the page being updated and writes the new version to a free page. This introduces the need to keep track of the current page version location on flash itself, which is maintained by implementing an address translation layer (FTL). The OOB area of invalid pages are marked to indicate their changed states.

### 2.2.2 Garbage collection

Out-of-place updates result in the creation of invalid pages on flash. A garbage collector is employed to reclaim invalid pages and create new erased blocks. It first selects a victim block based on a policy such as choosing a block with maximum invalid pages. All valid data within the block is first copied into an erased block. This data-rewrite operation can be quickly and efficiently processed by the special support of a *Copy-Back Program* operation where an entire page is moved into the internal data buffer first and then written [1]. Then the victim block is erased. The efficiency of garbage collector is one of the dominant factors affecting flash memory performance.

## 2.3 NAND Flash based SSDs

Figure 1 describes the organization of internal components in a flash-based SSD. It possesses a host interface (such as Fiber-Channel, SATA, PATA, and SCSI etc.) to appear as block I/O device to the host computer. The main controller is composed of two units—processing unit (such as ARM7 processor) and fast access memory (such as SRAM). The virtual-to-physical mappings are processed by the processor and the data-structures related to the mapping table are stored in SRAM in the main controller. The software module related to this mapping process is called FTL. A part of SRAM can be also used for caching data.

A storage pool in a SSD is composed of multiple flash memory *Planes*. The *Planes* are implemented in multiple *Dies*. For example, Samsung 4 GB flash memory has two *Dies*. A *Die* is composed of four planes, each of size is 512 MB [1]. A *Plane* consists of a set of blocks. The block size can be 64, 128, 256 KB etc. depending on the memory manufacturer. The SSD can be implemented multiple *Planes*. SSD performance can be enhanced by interleaving requests across the planes, which is achieved by the multiplexer and de-multiplexer between SRAM buffer and flash memories [1].

## 2.4 Flash translation layer

The 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 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 [6,7,19,22,31,32,44] and their improvement by write-buffering [21] have been studied. A recent page-based FTL scheme called DFTL [10,23] utilizes temporal locality in workloads to overcome the shortcomings of the regular page-based scheme by storing only a sub-

set 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 [16,39].

### 3 Experimental methodology

In this section, we present our experimental methodology for the performance analysis of individual SSD and various RAID configurations of SSDs.

#### 3.1 Redundant array of independent SSDs

Redundant arrays of inexpensive disks (RAID) [42] were introduced to increase the performance and reliability of disk drive systems. RAID provides parallelism of I/O operations by combining multiple inexpensive disks, thereby achieving higher performance and robustness than a single drive. RAID has become the de facto standard for building high-performance and robust HDD-based storage systems. Hypothesizing that RAID could provide similar benefits for SSDs, we investigate the performance behavior of SSD-based RAID systems for a variety of I/O workload patterns. As we will observe in Sect. 4, our comprehensive evaluation reveals that SSD-based RAID configurations exhibit serious bandwidth variability due to GCs of individual SSDs.

In our study, we have focused on a RAID storage using SSDs (RAIS), instead of HDDs. We defined RAIS-0, 5, and 6 analogous to RAID-0, 5, and 6 as follows:

- **RAIS-0** A request is striped across multiple SSDs. As there is no redundancy 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.

#### 3.2 Experimental setup

All experiments are performed on a single server with 24 GB of RAM and an Intel Xeon Quad Core 2.93 GHz CPU [13],

**Table 1** Storage device characteristics

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** Default settings of a LSI MegaRAID controller

RAIS scheme	0, 5, 6
Write cache	Write through
Read ahead	No
Direct I/O	Yes
Stripe size	64 KB

running Lustre-patched 2.6.18-128 kernel. The *noop* I/O scheduler that implements FIFO queue was used [43]. The test-bed has seven 8x PCIe slots and two of these are installed with PCIe RAID controller cards. We use two LSI MegaRAID SAS 9260-8i KIT RAID Adapters [34], each of which can support up to 8 SATA drives.

We use three representative SSDs in our evaluation. We select Super Talent 128 GB SSD [47] as a representative of MLC-based SSDs and Intel 64 GB SSD [14] as a representative of SLC-based SSDs. We use Fusion-io 640 GB ioDrive Duo [8] as a representative of PCIe-based SSDs. We denote SuperTalent MLC, Intel SLC, and the Fusion-io MLC devices as SSD(A), SSD(B), and SSD(C) respectively. Their details are presented in Table 1.

To compare the performance of RAISs of SSD(A)s and SSD(B)s<sup>2</sup> with SSD(C), we use two PCIe interfaced hardware RAID controllers for each 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 2. In particular, we only allow write-through cache and disable read-ahead in the RAID controller. This cache setting is intended to keep our evaluation setting the same as used in our production systems. Oak Ridge Leadership Computing facility hosts a peta-scale storage system, Spider-II for the fastest supercomputer Titan in US [49]. In the Spider storage system, all storage controllers are set to allow only write-through cache to not lose data in a power-failure and provide high data integrity in the storage system. They also

<sup>2</sup> Hereafter, we call RAIS(A) and RAIS(B) for RAIS configurations using SSD(A)s and SSD(B)s respectively.



disable read cache, because of an unexpected performance by the read cache.

In order to minimize the skew in our measurements due to start-up effects, we “warm-up” each SSD device prior to collecting data, which means SSDs are all filled so that they will start to trigger GCs when we run I/O workloads on them. At the beginning of each evaluation, all SSDs are exercised with an I/O pattern identical to that of the experiment. We repeat every experiment five times for all test cases.

To measure the I/O performance, we develop 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 without waiting for I/O completion (`io_submit()`). It also can perform reads and writes on raw block devices. We use 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 measure 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.

## 4 Performance analysis of RAIS

We present our performance analysis results for individual SSDs and various RAIS configurations with a variety of I/O access patterns.

### 4.1 Individual SSD performance analysis

In this subsection, we present experimental results on individual SSDs using sequential I/O requests by varying the request size and number of outstanding requests in the I/O queue.

#### 4.1.1 Sequential I/O performance

In Fig. 2a, 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 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 Fig. 2a, b). For reads, we observe 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 lower

than on MLC Flash [29,40]. For sequential writes, as shown in Fig. 2d, e, we 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 read values of both devices. Flash writes are slower than reads in part because writes can cause garbage collection (GC) events. Similar to reads, we see that SSD(A) does not scale well compared to SSD(B) for small write requests.

Figure 2c, f show the results of PCIe SSDs, we observe from the figure 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.

Interestingly, we observe high variances at several measurement points for read performance measurements. We conjecture the reasons for such high variance in measurements as follows:

- Devices might have different internal fragmentation levels of data blocks, which can directly impact the scheduling of GC processes [5].
- Garbage collection processes (GCs) can be triggered even in read-dominant workloads (possibly because of past write-dominant workloads and internal scrubbing operations in the SSDs). GCs can increase the queuing delay (the time a request waits in an I/O queue) of incoming requests, resulting in high request service latencies.

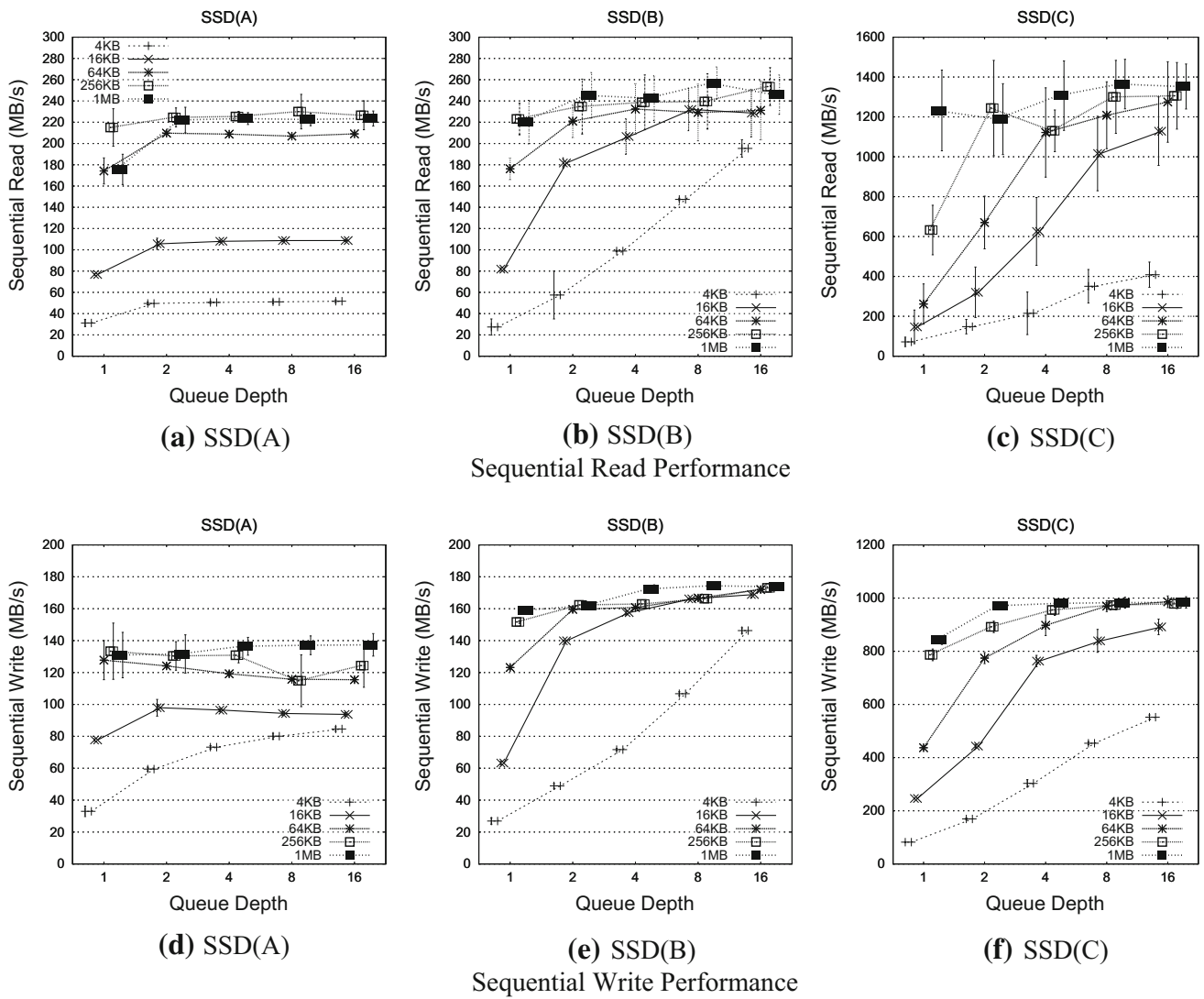
Compared to read performance, we do not see high variance from write performance measurements. It is because individual SSDs can use internal write buffers to minimize the write performance degradation due to GC.

*Lessons learned* SSD’s read performance can vary much higher than its write performance. SSD can use its internal write buffer to mitigate write performance degradation due to GC. Among SSDs with the same interface (e.g., SATA), their SSD performance behaviors can be different by manufacturers.

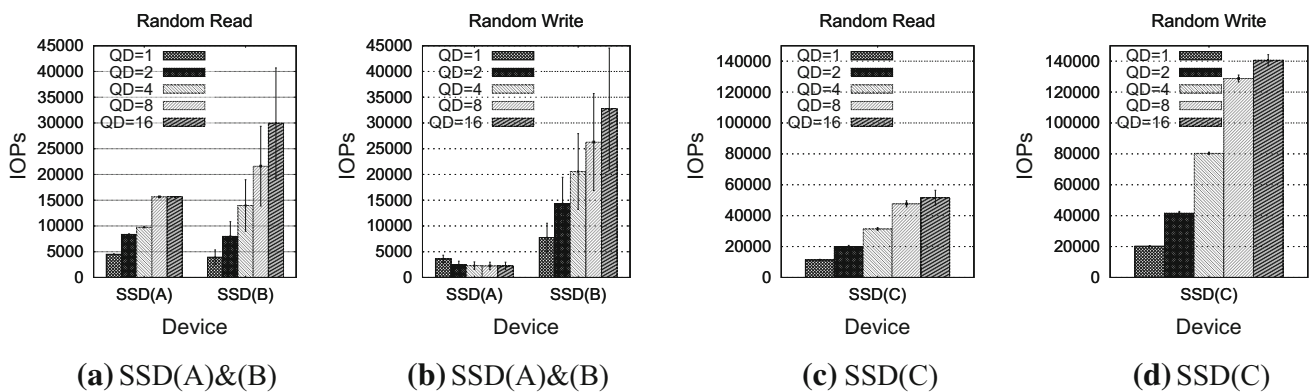
#### 4.1.2 Random I/O performance

Figure 3 shows the performance characteristics of SSD(A), (B), and (C) for 4 KB random I/O access patterns with respect to varying queue depth. These 4 KB I/O performance measurements are important because small I/O requests would happen by file system meta-data operations, which is critical for the file system performance. We use IOPs as a metric to comparing random I/O performance. IOPs is one of the widely used metrics in benchmarking storage systems for random I/O access patterns.

In Fig. 3a, 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 by the



**Fig. 2** Observed sequential I/O throughputs of single, individual SSD(A), (B), and (C) devices, using the libio benchmark. ‘Queue Depth’ (QD) 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



**Fig. 3** Observed random I/O throughputs of 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

increase of 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) significantly performs better than SSD(A) in terms of IOPs, even though we observe a high variance in SSD(B) results. This high variance on random writes can be explained by the following reasons. First, we used a simple definition for sequential/random requests in workloads in our experiments, 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. Second, we believe internal caching policies (prefetching or write-buffering) can perform differently based on manufacturers' design choices.

Interestingly, SSD(B) and SSD(C) (Fig. 3b, 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 have been one of the biggest problems to overcome in SSDs [10, 23, 30]. We suspect that SSD(B) and (C) must have been designed to providing a better performance on random writes. The manufacturer of SSD(B) and (C) might have employed an enhanced write-buffering scheme [21] or employed an enhanced FTL scheme optimized for random writes [10, 23].

*Lessons learned* SSDs can be tuned to present different performance. How the internal write cache is designed and implemented will determine the performance. SSD can poorly perform for the bursty workloads (referring to high queue depth), thus, system designers should consider these performance characteristics, when they purchase SSDs.

## 4.2 RAIS performance analysis

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.

For experiments, we use 4 SSDs for the RAIS-0 configuration. Since RAIS-5 and 6 configurations need extra parity drives, we use 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 2.

### 4.2.1 Sequential I/O performance

Figure 4a–c shows sequential read bandwidth for RAIS(A). Overall, we see that bandwidth increases by the increase of the queue depth and request size for all RAIS configurations. Surprisingly, we observe that the bandwidth for the RAIS-5(A) configuration is higher than RAIS-0(A) and the RAIS-6(A) configuration performs even better than RAIS-5(A).

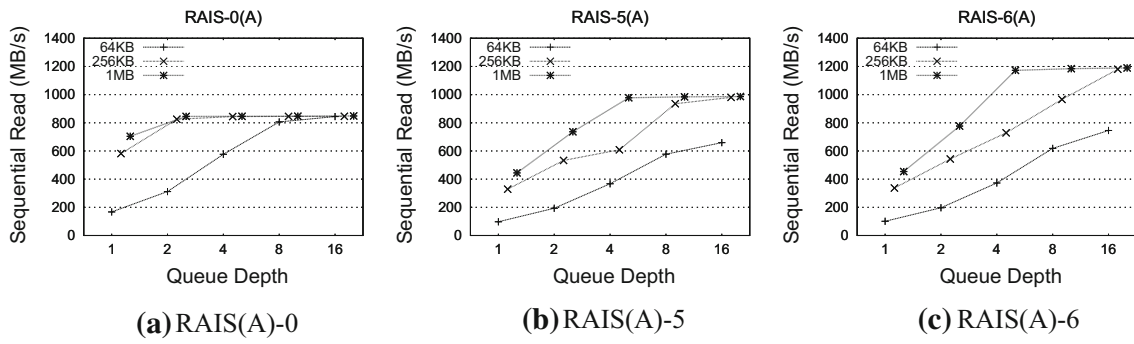
From the figures, we see RAIS-0(A), RAIS-5(A), and RAIS-6(A) provide up to 850 MB/s, 985 MB/s, and 1,190 MB/s, respectively. It is because RAIS-5 and RAIS-6 in our configuration used more disks by one and two respectively than RAIS-0. RAIS-5 and 6 does not read parity blocks for read requests and they could take advantage of more parallelism available over the disk arrays with RAIS-5 and 6 than RAIS-0. In Fig. 4g, h and 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). On the other hand, we have different observations from the experiments with SSA(B). We conducted repetitive tests over and over, however, we had the same trend. We attributed this to performance anomaly, which can be caused by a mismatched performance tuning of a HDD based RAID controller with SSD controllers. As a lesson from this experiment, we report that SSD array performance should be carefully measured because their performance expectation can be different from what can be observed from a HDD array setup.

Figure 4d–f shows the results of the RAIS performance for sequential write workloads by varying request size. We observe that there exists an optimal I/O request size where parity calculation overhead becomes minimal. In our evaluation, we see that when the request size is equal to the full strip size over data drives (in our experiment, 256 KB considering a stripe size of 64 KB), the parity calculation overhead is minimal. However, the request size is small than 256 KB or greater than that, we observe degraded throughputs. Especially, we observe that the performance degradation is more significant for small I/O requests than large I/O requests. And we observe that there is very little difference in bandwidth observed between RAIS-5 and RAIS-6 configuration for parity calculation overhead. It is because we used a hardware RAID controller for evaluating RAIS, and particularly, the hardware RAID controller does not increase parity calculation overhead for RAIS-6 (with two parity disks) than RAIS-5. We see the similar observation from Fig. 4j–l which are the evaluation results with RAIS(B).

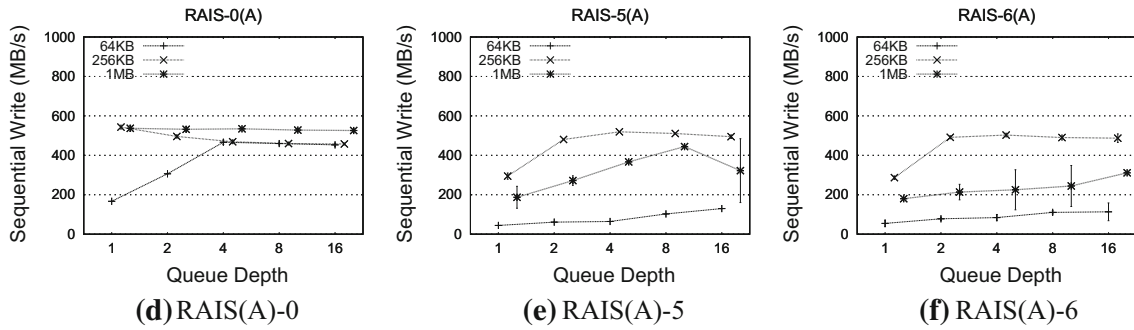
*Lessons learned* In RAIS-5 and RAIS-6, even if the number of data drives is the same as in their RAIS-0 configuration, the RAIS-5 and 6 configuration can offer a higher sequential bandwidth using more drives in RAIS-0, in particular for read workloads. There can exist an optimal request size to maximizing the write performance of RAIS-5 and 6 configurations.

### 4.2.2 Random I/O performance

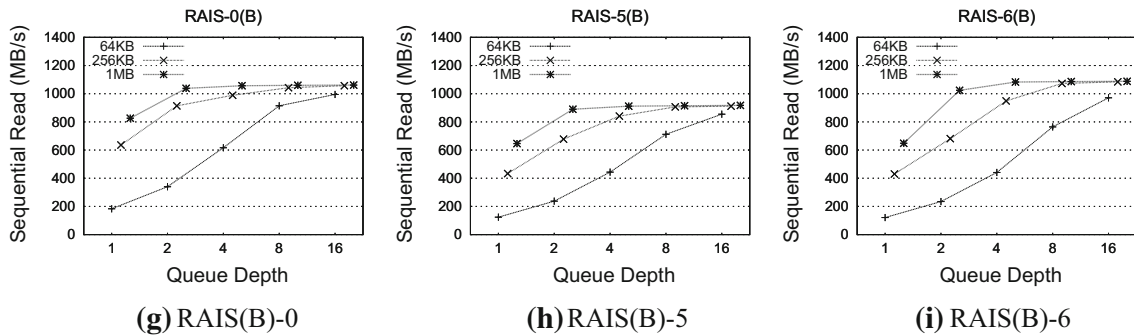
For random reads, as shown in Fig. 5a, c, we see both configurations' IOPs performance increase by the increase on the queue depth and request size, as we expected, from the individual SSD performance given in Fig. 3. Among differ-



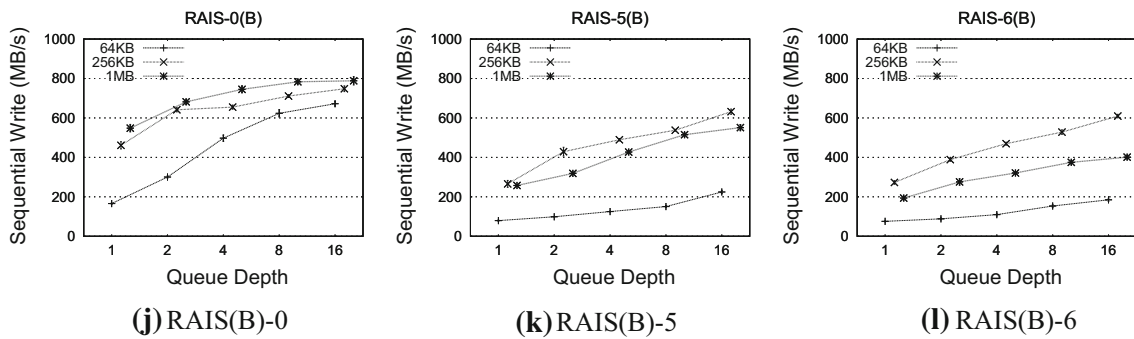
Sequential Read Performance for RAIS(A)



Sequential Write Performance for RAIS(A)



Sequential Read Performance for RAIS(B)

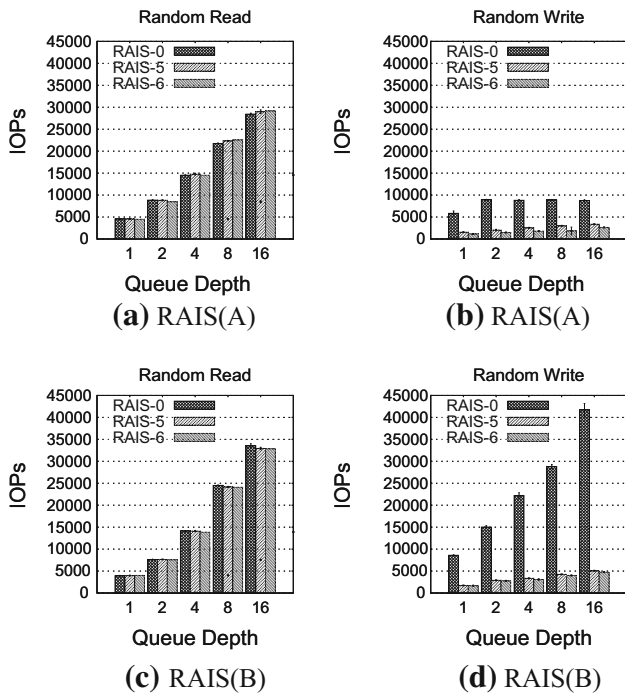


Sequential Write Performance for RAIS(B)

**Fig. 4** 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, respec-

tively. The *error bars* show 95% confidence intervals. Note that some intervals are too narrow to be apparent here





**Fig. 5** Performance of RAIS(A) and (B) sets under random I/O access patterns. The error bars show 95 % confidence intervals

ent RAIS schemes, we can not see any significant performance differences. For random writes, Fig. 5b, d, we see that RAIS-0(A) performs poorly compared to RAIS-0(B). We 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. These are congruent with our observations illustrated in Fig. 3. However, as can be seen in Fig. 5b, 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, however again, as we used a hardware RAID

controller, there is little difference between RAIS-5 and 6 for parity calculation overhead. Moreover, we observe that there is a huge performance gap between RAIS-0 and RAIS-5 & 6 with RAIS(B) compared with RAIS(A). We suspect that SSD(B)'s firmware implementation such as garbage collection could contribute more the performance degradation in RAIS(B) than in RAIS(A).

*Lessons learned* RAIS can slightly improve random read performance over individual SSDs however, their improvement does not linearly increase with respect to the number of drives in the RAID configuration. Particularly, we advise storage systems designers and architects should be careful when they configure the RAID based storage systems with SSDs, because individual SSDs' performance can lead different performance in their RAID configurations.

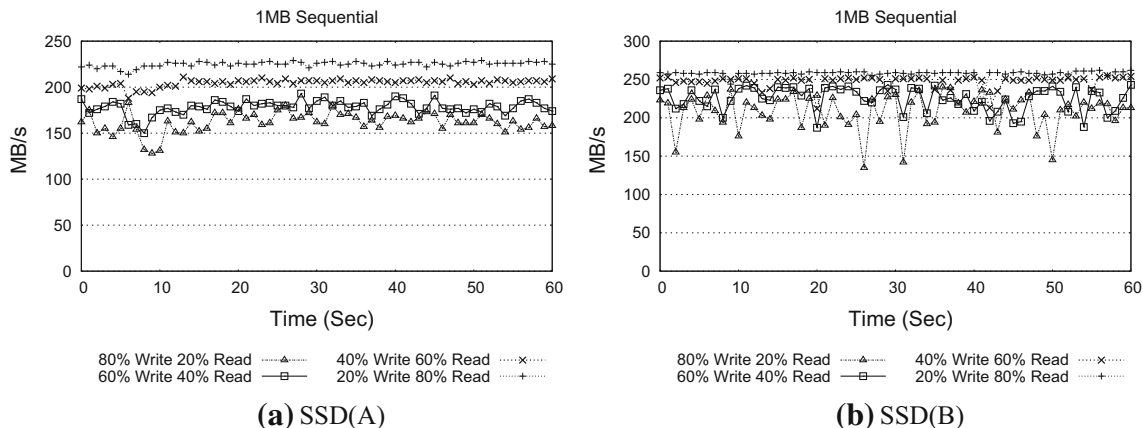
### 4.3 Pathological behavior

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.

#### 4.3.1 Performance anomaly on individual SSDs

In Fig. 6a, 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 180 MB/s at the



**Fig. 6** Pathological behavior of single SSDs

**Table 3** Average and standard deviations of the values in Fig. 6

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

6th and 7th s under an 80% write workload. However, I/O throughput drops below 160 MB/s for the 8th s and then drops further to 130 MB/s in the next 3 s. 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 240 MB/s at the peak and dropped to 140 MB/s (at 25th–27th s). As we increase the amount of reads in the workloads from 20 to 80%, we observe that SSD(A)'s and (B)'s I/O throughput increase around 50 and 18%, respectively.

#### 4.3.2 Pathological behavior with RAIS

In order to clearly understand the effect of data drives with parity drives in RAID configuration, we made the same number of drives for each RAIS configuration. In other words, we evaluated RAIS-6 of six SSDs, RAID-5 of five data SSDs and one parity SSD, and RAID-6 of four data SSDs and two parity SSDs.

In Fig. 7a, b, we show results for a workload with a 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) (refer to Table 4), even though RAIS(B) provides higher bandwidth than RAIS(A). For example, we see that RAIS-0(B) reaches 1.5 GB/s at the peak and then drops to 1.2 GB/s. RAIS-0(A)

**Table 4** Average and standard deviations of the values in Fig. 7

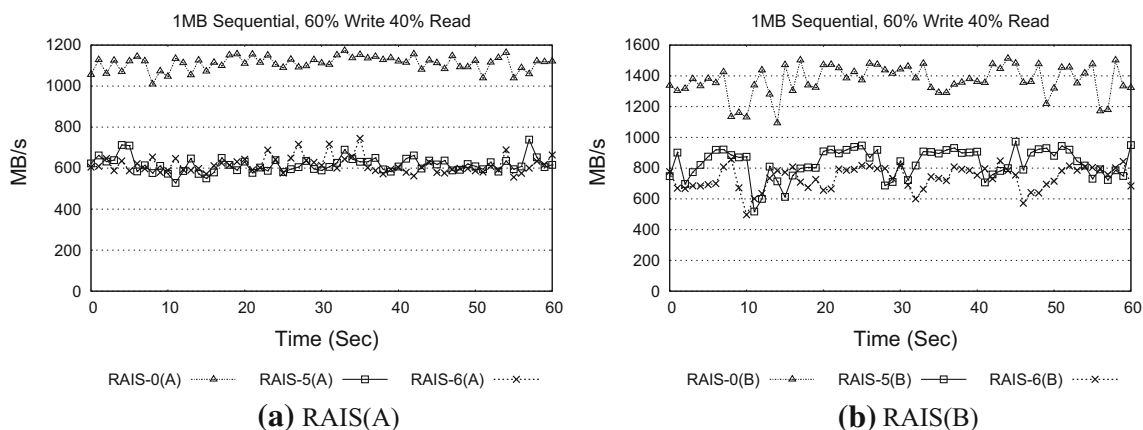
Type	Metric	RAIS scheme		
		0	5	6
RAIS(A)	Avg	1109.4	609.4	612.5
	SD	33.6	60.8	37.6
RAIS(B)	Avg	1386.3	837.2	737.6
	SD	98.2	92.4	72.6

varies 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 extra operations for parity block updates. However, we do not observe a large difference between RAIS-5 and RAIS-6 performance, specifically for RAIS(A) (refer to the average and standard deviation values of RAIS(A) for RAIS-5 and 6 in Table 4). RAIS-5(A) and RAIS-6(A) perform identically as can be seen in Fig. 7a. Under current RAIS setups, GC scheduling is controlled locally by an internal SSD logic. In RAIS, a lack of global GC scheduling mechanisms can cause overall synchronization problems across SSDs, reducing the efficiency of the RAIS set.

**Lessons learned** Compared with individual SSD's time-series analysis results, we observe that in RAIS, the performance can more severely fluctuate. As the hardware RAID does not incur extra overhead due to parity calculation for RAIS-6 over RAID-6, however, we could observe lower bandwidth on RAIS-6 than RAID-6 because, two parity updates can increase variability in bandwidth compared with one parity update in RAID-5.

#### 4.4 GC coordination

The overall performance of RAIS is limited by the slowest drive in the disk array. We suspect the performance fluctuates

**Fig. 7** Pathological behavior of RAIS

**Table 5** Descriptions of enterprise workloads

Workloads	Size (KB)	Read (%)	Arrival (IOP/s)
TPC-C	7.06	20.50	388.32
Openmail	9.49	63.30	846.62
TPC-H	31.62	91.80	172.73

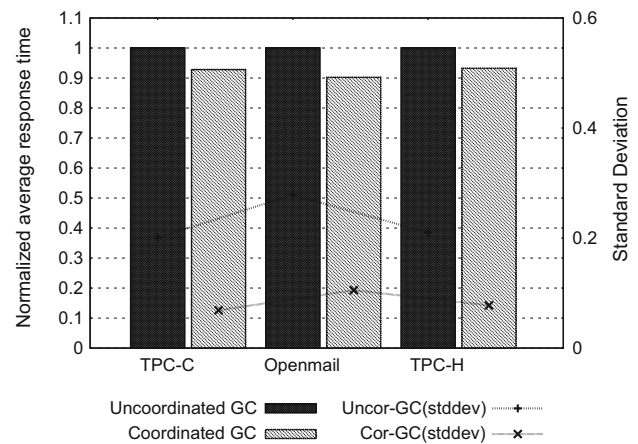
tuation we observed in the preceding sections is due to an uncoordinated GC process across SSDs in their array. Thus, we studied the effect of GC coordination between SSDs in RAID using simulation with realistic workloads.

For this experiment, we have implemented in the simulator, such that all GCs could be triggered at the same time, minimizing the performance-negative effect by the slowest disk in RAIS. We extended Microsoft Research SSD simulator [1] to present how much performance can be improved if the individual GCs are coordinated to be triggered all at the same time. We configure a RAIS-0 array of eight 32 GB SSDs. Each SSD simulates 64 flash chip elements, 4 planes per package, 512 blocks per plane, and 64 4 KB pages. 15% free blocks are reserved for GC management, and the Greedy GC algorithm is employed. For fair evaluation, prior to collecting performance data from the simulator, we fill the entire space on each SSD in the simulator with valid data by marking the flag that present the status of the page on every out-of-band (OOB) in the simulator as 1 s.

We evaluate five enterprise-scale workloads, whose characteristics are described in Table 5. We use write-dominant I/O traces from an online transaction processing application known TPC-C [48], made available by the Storage Performance Council (running at a financial institution). We consider e-mail server workloads referred to as Openmail [12]. We also examine TPC-H [51], which is a disk I/O trace collected from an online analytical processing application examining large volumes of data to execute complex database queries.

Figure 8 presents the improvement by coordinated GCs on SSDs, compared to the uncoordinated GCs. As clearly seen from the result, we could alleviate the performance concern in RAIS, by coordinating individual GC operations. We see that the performance improvement by the average response time could be achieved by about 10% but also, the variability of performance of the SSDs, could be minimized. We further studied for a variety of experimental scenarios, and those results can be referred in [25].

**Lessons learned** The overall performance of RAIS can be bound by the slowest drive. As we identified earlier, when RAID is built using SSDs, uncoordinated GCs of the individual SSDs can hinder the RAIS performance improvement. Thus, it is important to coordinate those GC operations on the SSDs to make full use of the RAIS advantages.



**Fig. 8** Normalized average response times (*left y-axis*) and normalized standard deviation (*right y-axis*) for uncoordinated GCs and coordinated GCs configured RAID arrays under various enterprise-scale workloads

#### 4.5 RAIS performance-cost efficiency analysis

SSDs, unlike HDDs can expose more flexible design spaces to manufacturers. As described in Table 1, 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.) [1], which is one of the reasons that SSDs show differences in market prices.

For fair comparisons, 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 conduct 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 answer this question, we experiment by configuring 6 SSD(A) and 6 SSD(B) devices in RAIS-0(A) and RAIS-0(B) configurations, respectively. Then we analyze 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 include the cost of this extra RAID controller in our calculations and met-

rics. The details of the test configurations are described in Table 7

Figure 9a, b compares the efficiency of performance and storage capacity for our test configurations. Performance values that we consider are shown in Table 6. Since writes can be slower than reads for SSDs, we varied the amount of reads in the workload. In Fig. 9a, IOP/s per dollar are 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 observe 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 9b presents our comparisons on throughput efficiency of our storage configurations for large sequential access patterns. We use 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. We also observe that the efficiency of RAIS-0(A) increases by 10–20% as reads become more dominant in the workload. Overall, we observe that RAIS configurations are more efficient than SSD(C) for workloads with large sequential I/O access patterns.

In Fig. 9c, we compare the storage capacity efficiency based on current market prices shown in Table 1. We see that RAIS-0(A) is the most efficient storage configuration, irrespective of I/O access patterns.

**Table 6** Average performance values used in Fig. 9a, b

	Read (%) in workload					
	0	20	40	60	80	100
<b>IOPs</b>						
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
<b>MB/s</b>						
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	1,256	1,445	1,506

The price of the additional RAID controller is \$579

**Table 7** Test configurations

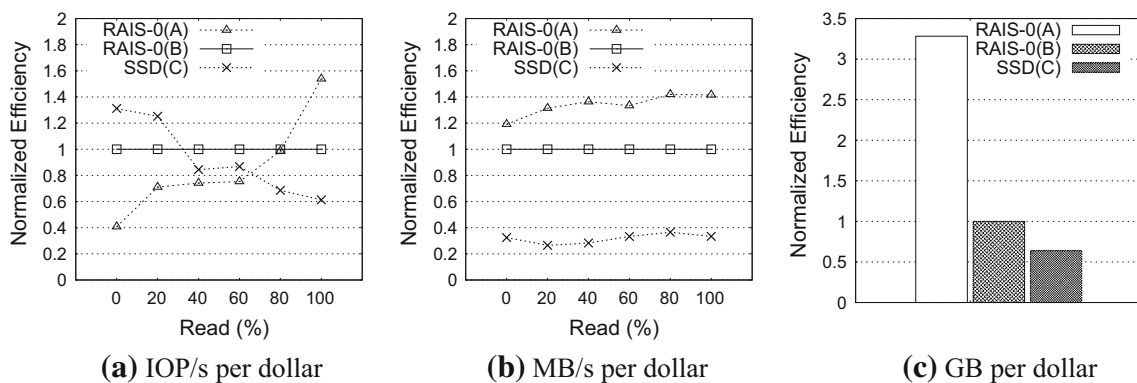
Type	RAIS Scheme	RAIS Cap. (GB)	Device (#)	RAID Ctrl (#)
RAIS-0(A)	(Striping)	768	6	1
RAIS-0(B)	(Striping)	384	6	1
SSD(C)	–	640	1	0

The price of the additional RAID controller is \$579

*Lessons learned* RAIS arrays built using multi-level cell (MLC) SSDs can be cost-effective solutions and perform reasonably well compared to fast, PCIe-based SSDs.

### 5 Related work

There are several empirical studies that test SSD performance and power consumption with respect to different access patterns [5,46]. However, these studies are limited by the analysis of single devices, and did not extend to RAID config-



**Fig. 9** 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 1. 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)



urations. There is a recent work from Microsoft [18] 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 the empirical studies with the arrays of SSDs.

Several recent studies have performed on the design of the arrays of SSDs [15, 25–27], and their performance optimization for various workloads. Kim et. al. [26] explored the way of improving the performance when SSDs are configured in RAID using a technique to coordinate individual SSD’s GC operations. Jian et. al. [15] proposed an optimized RAM buffer management on individual SSDs to improve the overall performance of their RAID configuration. Lee et. al. [27] enhanced the performance of RAID of SSDs by enhancing the buffer management algorithm on the RAID controller. All of these efforts have focused on improving the RAID performance of SSDs, however little has been done to thoroughly evaluate the performance of SSD RAIDs for the workloads with various I/O access patterns. There are several key different contributions compared to prior works.

First, we comprehensively characterized the performance of individual SSDs and their RAID configuration. Even if single SSD performance with respect to I/O access patterns have been already revealed, there are little performance study on SSD RAID. Our study will help understand better the results of performance of the RAID of SSDs, specifically about how they will scale, or to identify the performance bottleneck. Second, we address the challenges on building the RAID of SSDs, and open the opportunities to communities, to bring the ideas to solve the issues inherent when SSDs are configured in RAIDs. Finally, we have a cost-efficiency analysis with the sets of RAID of SSDs and the expensive yet powerful SSDs (PCIe Fusion-io SSD). Narayanan et. al., [36] 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. On the other hand, our work is more focused on conducting a cost efficiency analysis of COTS SSDs-based RAIS configurations against COTS PCIe-based SSDs. Our study reveals an idea of what kinds of products will be cost-effective to customers. All of these studies have been motivated to exploring the effectiveness of SSDs when they replace HDDs in our large-scale storage systems only populated with SSDs. From this study, we learned that variable performance from SSDs is a serious problem to building large-scale storage systems with SSDs.

## 6 Concluding remarks

In this paper, we conduct a comprehensive empirical study on SSDs and RAIS configurations in terms of performance and cost efficiency. We also study performance anomalies in SSDs and RAISs, which can be caused by local GC processes of individual SSDs. From I/O throughput analyses in time series, we identify shortcomings that future RAIS sets should overcome: RAIS can not provide sustained bandwidth due to independent and local GC processes of individual SSDs. Such local processes can cause GC synchronization problems in RAIS configurations, degrading the overall I/O throughput. We also investigated the chance of performance gains by coordinating GC operations of SSDs in their array sets. Moreover, we perform a cost efficiency analysis with various SATA RAIS configurations against a fast, yet expensive PCIe SSD. We used the mixes of four different major I/O access patterns (i.e. large sequential reads and writes, and small random reads and writes) in our measurements. We observe that, under the given market prices, a RAIS-0 configuration using MLC based SSDs is 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 plan include the effect of secure erase operations on the performance of the SSD arrays.

**Acknowledgments** I would like to thank Galen Shipman, Sarp Oral, David Dillow, and Jason Hill for their technical supports and comments. Also, I am grateful to the anonymous reviewers for their detailed comments which helped us improve the quality of this paper. This research is sponsored by the Office of Advanced Scientific Computing Research, U.S. Department of Energy and used resources of the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725. Most of this work was done while Youngjae Kim was with Oak Ridge National Laboratory.

## References

1. Agrawal, N., Prabhakaran, V., Wobber, T., Davis, J., Manasse, M., Panigrahy, R.: Design tradeoffs for SSD performance. In: Proceedings of the USENIX Annual Technical Conference, USENIX Association, Berkeley, CA, pp. 57–70, June (2008)
2. Atwood, G., Fazio, A., Mills, D., Reaves, B.: Intel StrataFlash™ Memory Technology Overview. Intel Technol. J. 1–8 (1997)
3. Bandic, Z.Z., Litvinov, D., Rooks, M.: Nanostructured materials in information storage. MRS Bull., **33**(9), 831–837 (2008)
4. Chang, Y.-H., Hsieh, J.-W., Kuo, T.-W.: Endurance enhancement of flash-memory storage systems: an efficient static wear leveling



- design. In: Proceedings of the 44th Annual Conference on Design Automation, pp. 212–217 (2007). ACM, New York
5. Chen, F., Koufaty, D.A., Zhang, X.: 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 '09) pp. 181–192 (2009). ACM, New York
  6. Choudhuri, S., Givargis, T.: 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), pp. 257–262 (2007)
  7. Chung, T., Park, D., Park, S., Lee, D., Lee, S., Song, H.: System software for flash memory: a survey. In: Proceedings of the International Conference on Embedded and Ubiquitous Computing, pp. 394–404, August (2006)
  8. Fusion-io 640 GB ioDrive Duo. <http://www.fusionio.com/products/iodriveduo/>. Accessed 2011
  9. Gal, E., Toledo, S.: Algorithms and data structures for flash memories. *ACM Comput. Surv.* **37**(2), 138–163 (2005)
  10. Gupta, A., Kim, Y., Urgaonkar, B.: 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), New York, NY, pp. 229–240, March (2009). ACM
  11. Gurumurthi, S., Sivasubramaniam, A., Natarajan, V.: Disk drive roadmap from the thermal perspective: a case for dynamic thermal management. In: Proceedings of the International Symposium on Computer Architecture (ISCA), Washington, DC, pp. 38–49, June (2005). IEEE Computer Society
  12. HP-Labs. The Openmail Trace. <http://tesla.hpl.hp.com/opensource/openmail/>. Accessed 2006
  13. Intel. Intel Xeon Processor X5570 (8M Cache, 2.93 GHz, 6.40 GT/s Intel QPI). <http://ark.intel.com/Product.aspx?id=37111>. Accessed 2009
  14. Intel. Intel, X25-E Extreme 64GB SATA Solid-State Drive SLC. <http://www.intel.com/design/flash/nand/extreme/index.htm>. Accessed 2009
  15. Jian, H., Hong, J., Lei, T., Lei, X.: GC-ARM: garbage collection-aware RAM management for flash based solid state drives. In: Proceedings of the Networking, Architecture and Storage NAS, pp. 134–143 (2012)
  16. Jo, H., Kang, J., Park, S., Kim, J., Lee, J.: FAB: flash-aware buffer management policy for portable media players. *IEEE Trans. Consum. Electron.* **52**(2), 485–493 (2006)
  17. Jung, D., Chae, Y., Jo, H., Kim, J., Lee, J.: 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), New York, NY, pp. 160–164, September (2007). ACM
  18. Kadav, A., Balakrishnan, M., Prabhakaran, V., Malkhi, D.: Differential RAID: rethinking RAID for SSD reliability. In: Proceedings of the First Workshop on Hot Topics in Storage and File Systems (HotStorage) (2009)
  19. Kang, J., Jo, H., Kim, J., Lee, J.: A superblock-based flash translation layer for NAND flash memory. In: Proceedings of the International Conference on Embedded Software (EMSOFT), New York, NY, pp. 161–170, October (2006). ACM
  20. Kawaguchi, A., Nishioka, S., Motoda, H.: A flash-memory based file system. In: Proceedings of the Winter 1995 USENIX Technical Conference, USENIX Association, Berkeley, CA, pp. 155–164 (1995)
  21. Kim, H., Ahn, S.: BPLRU: a buffer management scheme for improving random writes in flash storage. In: Proceedings of the USENIX Conference on File and Storage Technologies (FAST), USENIX Association, Berkeley, CA, pp. 1–14, February (2008)
  22. Kim, J., Kim, J.M., Noh, S.H., Min, S., Cho, Y.: A space-efficient flash translation layer for compactflash systems. *IEEE Trans. Consum. Electron.* **48**(2), 366–375 (2002)
  23. Kim, Y., Gupta, A., Urgaonkar, B.: A temporal locality-aware page-mapped flash translation layer. *J. Comput. Sci. Technol.* **28**(6), 1025–1044 (2013)
  24. Kim, Y., Gupta, A., Urgaonkar, B., Berman, P., Sivasubramaniam, A.: HybridStore: a cost-efficient, high-performance storage system combining SSDs and HDDs. In: Proceedings of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS'11, July (2011)
  25. Kim, Y., Lee, J., Oral, S., Dillow, D., Wang, F., Shipman, G.: Coordinating garbage collection for arrays of solid-state drives. *IEEE Trans. Comput. (TC)* **63**(4), 888–901 (2014)
  26. Kim, Y., Oral, S., Shipman, G., Lee, J., Dillow, D., Wang, F.: Harmonia: a globally coordinated garbage collector for arrays of solid-state drives. In: Proceedings of the Mass Storage Systems and Technologies (MSST), pp. 1–12 (2011). IEEE
  27. Lee, J., Kim, Y., Kim, J., Shipman, G.: Synchronous I/O scheduling of independent write caches for an array of SSDs. *IEEE Comput. Arch. Lett. (CAL)* **13**(1) (2015)
  28. Lee, J., Kim, Y., Shipman, G., Oral, S., Wang, F., Kim, J.: A semi-preemptive garbage collector for solid state drives. In: Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS '11, Washington, DC, pp. 12–21 (2011). IEEE Computer Society
  29. Lee, S., Ha, K., Zhang, K., Kim, J., Kim, J.: FlexFS: a flexible flash file system for MLC NAND flash memory. In: Proceedings of the USENIX Annual Technical Conference (2009)
  30. Lee, S., Moon, B.: Design of flash-based DBMS: an in-page logging approach. In: Proceedings of the International Conference on Management of Data (SIGMOD), New York, pp. 55–66, August (2007). ACM
  31. Lee, S., Park, D., Chung, T., Lee, D., Park, S., Song, H.: A Log Buffer based Flash Translation Layer Using Fully Associative Sector Translation. *IEEE Transactions on Embedded Computing Systems* **6**(3), 18 (2007)
  32. Lee, S., Shin, D., Kim, Y., Kim, J.: LAST: locality-aware sector translation for NAND flash memory-based storage systems. In: Proceedings of the International Workshop on Storage and I/O Virtualization, Performance, Energy, Evaluation and Dependability (SPEED2008), New York, pp. 36–42, February (2008). ACM
  33. Lofgren, K.M.J., Norman, R.D., Thelin, G.B., Gupta, A.: Wear Leveling Techniques for Flash EEPROM Systems, US Patent No. US6230233 B1 (2001)
  34. LSI. MegaRAID SAS 9260-8i RAID Card. <http://www.lsi.com/channel/products/megaraid/sassata/9260-8i/index.html>. Accessed 2009
  35. Mallery, M., Torabi, A., Benakli, M.: One terabit per square inch perpendicular recording conceptual design. *IEEE Trans. Magn.* **38**(4), 1719–1724 (2002)
  36. Narayanan, D., Thereska, E., Donnelly, A., Elnikety, S., Rowstron, A.: Migrating enterprise storage to SSDs: analysis of tradeoffs. In: Proceedings of the ACM European Conference on Computer Systems (Eurosys), pp. 145–158, March (2009)
  37. Nijjima, H.: Design of a solid-state file using flash EEPROM. *IBM J. Res. Devel.* **39**(5), 531–545 (1995)
  38. Oral, S., Wang, F., Dillow, D., Shipman, G., Miller, R.: 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)
  39. Park, S., Jung, D., Kang, J., Kim, J., Lee, J.: CFLRU: a replacement algorithm for flash memory. In: Proceedings of the International

- Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES), New York, pp. 234–241 (2006). ACM
40. Park, S., Park, J., Jeong, J., Kim, J., Kim, S.: 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)
  41. Park, S., Shen, K.: A performance evaluation of scientific I/O workloads on flash-based SSDs. In: Proceedings of the 2009 IEEE International Conference on Cluster Computing, New Orleans, August 31–September 4, 2009, pp. 1–5 (2009). IEEE
  42. Patterson, D., Gibson, G., Katz, R.: Case for redundant arrays of inexpensive disks (RAID). In: Proceedings of ACM SIGMOD Conference on the Management of Data, pp. 109–116, June (1988)
  43. Pratt, S., Heger, D.A.: Workload dependent performance evaluation of the linux 2.6 I/O schedulers. In: Proceedings of the Linux Symposium, July (2004)
  44. Rajimwale, A., Prabhakaran, V., Davis, J.D.: Block management in solid-state devices. In: Proceedings of the USENIX Annual Technical Conference (2009)
  45. Seagate. Seagate Cheetah, 15K.7 Disc Drive. [http://www.seagate.com/docs/pdf/datasheet/disc/ds\\_15k\\_7](http://www.seagate.com/docs/pdf/datasheet/disc/ds_15k_7). Accessed 2009
  46. Seo, E., Park, S., Urgaonkar, B.: 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)
  47. Super Talent. Super Talent 128GB UltraDrive ME SATA-II 25 MLC. [http://www.supertalent.com/products/ssd\\_detail.php?type=UltraDrive](http://www.supertalent.com/products/ssd_detail.php?type=UltraDrive). Accessed 2009
  48. Transaction-Processing-Performance-Council. TPC-C, an OLTP Benchmark. <http://www.tpc.org/tpcc/>
  49. Xie, B., Chase, J., Dillow, D., Drokin, O., Klasky, S., Oral, S., Podhorszki, N.: Characterizing Output Bottlenecks in a Supercomputer. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '12, Los Alamitos, CA, pp. 8:1–8:11 (2012). IEEE Computer Society Press
  50. Youssef, R.: RAID for Mobile Computers. Master's Thesis. Carnegie Mellon University Information Networking Institute, August (1995)
  51. Zhang, J., Sivasubramaniam, A., Franke, H., Gautam, N., Zhang, Y., Nagar, S.: Synthesizing representative I/O workloads for TPC-H. In: Proceedings of the International Symposium on High Performance Computer Architecture (HPCA), Washington, DC, pp. 142–151 (2004). IEEE Computer Society



**Youngjae Kim** received the BS degree in computer science from Sogang University, Korea in 2001, the MS degree from Korea Advanced Institute of Science and Technology in 2003, and the PhD degree in computer science and engineering from Pennsylvania State University in 2009. He worked at the Oak Ridge National Laboratory as a research scientist from 2009 to 2015. His research interests include operating systems, parallel I/O and file systems, storage systems, and emerging storage technologies. He is currently an assistant professor in the department of information and computer engineering at Ajou University, Suwon, South Korea.