2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

SIRF-1: Enhancing Reliability of Single Flash SSD through Internal Mirroring for Mission-Critical Mobile Applications

Michael S. MacFadden, Richard Shelby, Tao Xie
Computer Science Department
San Diego State University
San Diego, CA, USA

Abstract— Flash memory based solid state drives (SSD) are increasingly common in portable and mobile computing devices such as laptops, mobile phones, and tablets. Due to space, weight, and power constraints, portable devices are often restricted to a single storage device, which makes them susceptible to data loss from internal errors. On the other hand, mission-critical mobile applications like wireless healthcare always demand a high level of data reliability. This is mainly because data sampled from mobile and dynamic environments are most likely irreproducible. An effective approach to improving storage and data reliability is the RAID (redundant arrays of inexpensive disks) organization. However, the multiple disks required to implement RAID make it incompatible with the aforementioned restrictions of many portable devices. In this paper, we propose a SIRF (single internally redundant flash) architecture that leverages the internal hierarchical structure and parallelism of SSDs to provide redundancy similar to RAID in a single drive configuration. The initial effort focuses on implementing SIRF-1 (mirroring), which is the corollary to its RAID-1 counterpart. In SIRF-1, data is mirrored across SSD channels to optimally exploit parallelism for both read and write operations. Simulation results show that for read-dominant workloads SIRF-1 significantly outperforms a non-mirrored SSD by up to 39.5% in terms of mean response time. For write-intensive workloads, SIRF-1 pays a performance penalty no more than 5.5%.

I. Introduction

Industry trends show that flash memory based solid state drives (SSDs) are becoming the de facto standard for portable and mobile computing platforms. The rapidly decreasing cost, small physical size, low power consumption, mechanical simplicity, and durability of SSDs make them an attractive alternative to traditional hard disk drives (HDDs) [1][2]. In addition, SSDs are gaining ground in enterprise servers due to their low-latency read performance and power efficiency.

All persistent storage technologies strive to provide high levels of reliability in order to maintain both the availability and integrity of users' data. One well-known approach is RAID [16], which utilizes multiple HDDs in concert to provide redundancy to increase both performance and reliability. RAID has successfully been implemented using HDDs, SSDs, and hybrid systems [10][12][14]. The high cost associated with RAID organization has steered research and implementation of such systems towards server-class machines and enterprise applications. A large portion of RAID implementations are constructed in these scenarios since they are able to support the increased space, weight, and power requirements of multiple disks and controllers. RAID has proven effective in these applications by improving both reliability and availability of the data that they store.

At the other end of the spectrum, individual mobile computing platforms are becoming ubiquitous in users' everyday lives. The design of these devices typically focuses on reducing weight and size along with increasing battery life through the utilization of low power consuming components. Mobile storage applications are typically constrained to a small device, making it impractical to implement a traditional RAID architecture, which requires multiple disks, controllers, and connections. Storage in mobile devices is increasingly implemented using a single SSD, which currently offers little to no data redundancy. This lack of redundancy is at odds with the fact that mobile devices are becoming ever more critical in government, business, and personal computing environments.

For example, wireless healthcare utilizes mobile communication devices, such as mobile phones and PDAs, for collecting community and clinical health data, delivery of healthcare information to patients, real-time monitoring of patient vital signs, and direct provision of care [18]. Thus, a reliable SSD is essential for wireless healthcare systems where life-critical personal health data is collected, stored, and analyzed on a daily basis [18]. Corporate employees leverage mobile devices to develop and store intellectual property. Users also store irreplaceable personal data such as pictures, organizational information, and financial data. Moreover, government and defense agencies rely on mobile devices in intelligence and tactical operations scenarios, where the loss of data can impact national security, mission success and jeopardize the safety of soldiers [19].

Fortunately, the parallelism and the independent failure modes of the individual flash memory chips within an SSD [4] provide a unique opportunity to apply RAID style redundancy within a single device. In this paper, we develop a single internally redundant flash (SIRF) architecture that provides data mirroring comparable to RAID level 1 (RAID-1) [16] within a single SSD. SIRF level 1 (SIRF-1) is designed to

978-1-4799-8006-2/15 $31.00 © 2015 IEEE
DOI 10.1109/CCGrid.2015.28

IEEE computer society

improve the reliability and read performance of a single SSD by implementing an internal mirror through the enhancement of the flash transition layer (FTL) present in the software/firmware of SSDs. SSDs are composed of multiple flash memory chips arranged into a hierarchy exhibiting parallelism at multiple levels [4]. Figure 1 illustrates a notional internal mirroring organization for an SSD mirrored across channel boundaries. The RAID controller shields the users from knowing that two mirrors exist. The user only sees one channel (e.g., Channel 0). In SIRF-1, each write request goes to both channels (i.e., mirrors) in parallel. However, a read request is dispatched only to one of the two channels and is dispatched to the channel that is less busy.

If one of the channels (or chips, dies, etc.) fails, then all reads must be performed on the corresponding component on the surviving mirror. Much like a regular RAID-1 architecture, this relies on the device being able to detect failures. RAID-1 does not typically handle Byzantine failures where in a drive reports itself as healthy but returns invalid data. Modern HDDs and SSDs typically implement their own internal error detection schemes, such as cyclic redundancy checks (CRCs), which allow the drive itself to detect internal inconsistencies. SIRF-1 assumes to trigger the failover behavior. When a failure is detected, reading from the surviving mirror will maintain data integrity. However, users would no longer get the benefits of the distributed reads. Thus, there would be performance degradation in this failure mode.

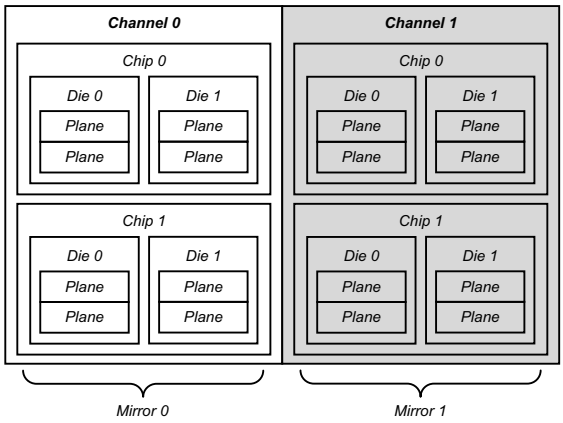

Fig. 1. SIRF-1 channel mirroring exmaple.

If redundancy similar to RAID can be introduced among the flash memory structures within an SSD, the reliability of a device can be increased without the need for multiple drives. While there may be many situations and classes of devices where the mirroring approach may not be ideal due to the reduction in effective storage, there are cases where performance and / or data integrity is paramount [18]. Additionally, the amount of storage per volume in SSDs is continually increasing. As of 2015, many mobile phones are available with upwards of 64GB of flash memory in a very small form factor. Even with a 50% reduction in effective storage, 32GB is more than enough storage for many use cases. While end users always want as much storage as possible, as storage capacities increase over time, the number of usage cases that can tolerate a 50% effective storage penalty will increase.

While the redundancy properties of mirroring are well understood [16], the performance implications of introducing mirroring across the various components of an SSD are not. As such, this paper makes the following contributions:

- The development of a SIRF-1 architecture that implements redundancy within a single SSD through internal mirroring. This approach improves both reliability and read performance. To the best of our knowledge, this is the first study on implementing mirroring within a single SSD to enhance reliability.
- A design roadmap for designing SIRF-1 within the FTL of an SSD, which encourages future research on multiple-parity-channel based SIRF-4 and SIRF-5.
- The enhancement of SSDSim, an open-source SSD simulator that has been validated against hardware. SSDSim will be used to evaluate the performance of the SIRF-1 architecture.
- An evaluation of SIRF-1 using several real-world workloads executed in a variety of SSD configurations. An average improvement in mean response time of 15% was observed with the best-case yielding a 39.5% improvement.

The remainder of this paper is organized as follows. Section 2 discusses related work that motivated this research. Section 3 gives the implementation details of SIRF-1. Section 4 presents the experimental results. Lastly, section 5 summarizes our efforts, insights and outlines our vision for continued research for internally redundant SSD architectures.

II. Related Work and Motivation

Current research in applying the RAID technologies on SSDs focus on using SSDs in place of, or in concert with, HDDs in an array fashion [2][10][13][15]. The proposed various RAID-like SSD arrays or HDD-SSD hybrid arrays largely focus on enhancing the RAID controllers by accounting for the unique performance and wear-out properties of SSDs to increase both the performance and longevity of SSDs [10][13]. Similar to RAID in HDDs, RAID structured SSD arrays attempt to exploit the parallelism provided by multiple SSDs. However, all of these SSD array techniques cannot be applied on mission-critical mobile applications where constraints in space, weight, and power consumption do not allow more than one SSD to be employed. Therefore, exploiting an SSD's internal structure to enhance its reliability becomes a feasible approach. Fortunately, several recent studies [4][5][15] reveal that the multi-level internal parallelism existed in the internal structure of an SSD significantly impacts its performance. The finding provides us with an opportunity to expand on the confluence of RAID and SSD concepts within a single SSD.

Existing research on exploring the SSD internal architecture mainly examines the allocation schemes, page size, and over-provisioning [1][4][5][7][15]. While most of them focus on enhancements to an FTL to increase

performance, little research has been done to address the inherent data reliability issues of SSDs from a system-level perspective. Although advances focused on wear leveling and overprovisioning generally aim to increase the longevity of the drive [1], they too do not directly address data reliability during the expected life of an SSD. Research has shown that while advancements in hardware manufacturing can increase the raw capacity and speed of the flash memory packages, FTL is still a key component for an SSD as it contributes noticeably to both performance and reliability [9][17].

Recent research has explored applying RAID-4 [7] and RAID-5 [10] style concepts within and/or between flash drives. This work demonstrated that applying RAID concepts to SSDs can be a viable mechanism to improving the reliablitiy of SSD. RAID-4/5 do offer attrative properties in terms of space efficiency and fault tolerance [7][10]. However, in traditional RAID implementations there are many tradeoffs when determining which RAID level to implement. For example, RAID-5 can have better effective storage efficiency, whereas RAID-1 will typically have better write performance. Additionally, the mirroring scheme of RAID-1 is much simpler to implement than the partiy schemes of RAID-5. This is why many integrated RAID controllers on nowadays motherboards only support RAID 0/1. Implmeenting RAID-1 can be a simpler extentions to existing FTLs and give implementers an additional alternative beyond RAID-4/5 concepts.

As stated above, the improved reliability and performance of RAID-1 may increase the utility of SSD's in mission-critical mobile application environments where data integrity and performance are paramount. Previous studies have demonstrated that the viability of enhancements to the FTL can be successfully evaluated through software-based simulations. By combining the proven data redundancy techniques in RAID-1 with the data access flexibility provided by SSD enhanced reliability can be achieved within a single device.

III. Design and Implementation

A. Architecture Overview

The FTL contained within an SSD hides the internal flash memory organization, data placement, and IO operations from the operating system. As depicted in Figure 2, the FTL is responsible for taking IO requests from the file system and translating them into appropriate internal operations required to service the request. For example, when a write request arrives at the SSD, the FTL must identify one or more available flash pages, write the data to those pages, and record the location of the written data in the mapping table. This process highlights the fact that there is no permanent relationship between the logical addresses that the file system uses to identify data and the locations of the physical pages that the SSD uses to store it. Each page within the SSD is assigned a fixed physical page number (PPN) that uniquely identifies the page. The PPN used to store a particular logical sector, as identified by the file system, will likely change each time the sector is written. The mapping table, contained in Dynamic Random Access Memory (DRAM), allows the SSD to keep track of where each logical page is physically stored in flash memory.

In traditional RAID implementations the controller presents multiple drives as a single storage device, thereby hiding the implementation details of the configuration [16]. The operating system sees a single storage device and is unaware of the data placement and redundancy properties of the RAID. Similarly, the SIRF-1 mirroring scheme must also be implemented in a transparent manor if existing device drivers and file systems are to be used. The FTL is the ideal place to implement mirroring, as it is already responsible for hiding the details of the SSD organization from the operating system. To implement mirroring within the FTL, five main high-level functions were enhanced: 1) page allocation, 2) write request processing, 3) page mapping, 4) buffer management, and 5) read request processing. Each of these will be discussed in turn.

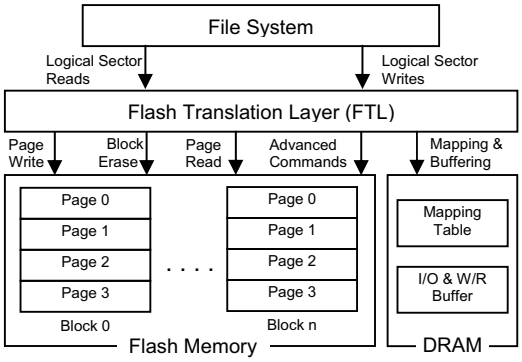

Fig. 2. SIRF-1 architecture.

B. Write Request Processing

Page allocation is the process of determining where to place data within the flash memory. The allocation algorithm within the FTL is responsible for selecting which physical pages will be used to service a write request. Allocation algorithms typically make use of all available flash memory to optimize performance and wear leveling [1]. To implement SIRF-1, the allocation technique must be enhanced to be cognizant of the mirroring scheme. To optimize parallelism, mirroring was implemented across channel boundaries. This required modifying the allocation algorithm to partition the SSD's channels into distinct mirror groups and to ensure that data assigned to a particular mirror group is only written to the channels belonging to that group.

When a write request arrives at the SSD, the FTL will leverage the allocation algorithm to find available pages to service the request. If this is the first time the logical page is to be written, the SSD writes the new data to the selected physical pages. However, since flash pages cannot be directly overwritten, if the requested logical pages had previously been written, the FTL may need to read the existing page data and merge it with the write operation's data before writing to the new location. Ultimately, the preexisting pages will be invalidated and be made available for subsequent garbage collection. In either case, the mapping table must be updated to record the physical location on the SSD that was used to store the data. The initial implementation for this research leverages

a page-level mapping table. This means each logical page affected by a write request will be mapped directly to the physical page used to store the data.

When a write request arrives from the operating system it specifics the starting logical sector number (LSN) of the data, the length of the data to be written, and the data itself. The FTL must create write operations to service the write request. If the data spans multiple logical pages, then the FTL will create multiple write operations. To implement SIRF-1, each write operation must be duplicated to store the requested data to both mirror groups. Since mirror groups are created by partitioning channels, the duplicate writes are guaranteed to execute on two different channels. This ensures that the mirrored write operations can happen in parallel, minimizing the overhead of processing the additional operations.

A *mirror* field was added to the write operation meta-data to indicate which mirror group the write operation is assigned to. As the FTL is creating the write operations to service the write request two identical write operations are created; one write request has the mirror field set to 0 and one write request has the mirror field set to 1. The updated allocation algorithm described above will use the mirror field to constrain the assignment of a physical page to the proper set of channels based on the specified mirror group. Figure 3 illustrates the conceptual flash memory state after a write request is issued on a 2-channel SSD that previously contained no data. The write operation illustrated is a request to write five pages worth of data starting from logical page number (LPN) seven on an SSD with two channels. This will result in ten individual write operations, with five operations executing on each channel.

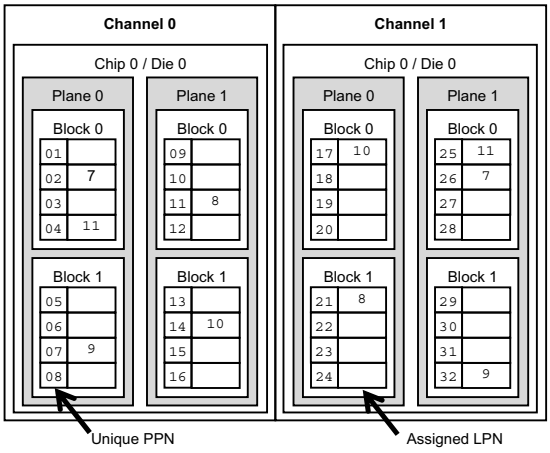

Fig. 3. Data placement after mirrored writes.

Figure 3 shows that LPNs seven through eleven have been written to both channels. However, the relative location that a particular LPN is stored within each channel is not the same. This is a major difference between the RAID-1 and SIRF-1 implementations. In a traditional RAID the drives are managed as identical mirrors. In SIRF-1, enabling the allocation algorithm to independently place data within each mirror group allows greater flexibility to optimize placement based on the current IO state of the drive when the write operations are actually carried out. To facilitate this flexibility, the page mapping mechanism must be enhanced to record the location of each logical page within each mirror group. Table 1 shows the enhanced mapping structure resulting from the write illustrated in Figure 3. Rather than a single mapping entry, every LPN has a record of the associated PPN within each mirror group.

To update a preexisting flash page, the FTL uses an out-of-place write strategy that may first read an existing page from flash memory in order to merge the existing data with the new data to be written. Most SSDs include an IO buffer to improve performance. This buffer leverages temporal locality to minimize the number of read-before-write operations required. When a write request arrives, the FTL will check if the physical page associated with the LPN is in the cache. If the page is not present in the cache, it must be read from flash memory. When implementing mirroring, an update request will result in pairs of associated write operations that operate on the same data. Both write operations can be serviced by a single cache entry. The buffer management algorithm was enhanced to detect if either page is present and, if required, only perform a single read operation from whichever mirror group is less busy to service both mirrored write operations. This reduces the amount of data required in the cache and lowers the overhead of mirrored update operations.

TABLE I. MAPPING TABLE AFTER MIRRORED WRITES

LPN	Mirror 0 PPN	Mirror 1 PPN
7	02	26
8	11	21
9	07	32
10	14	17
11	04	25

C. Read Request Processing

The final high-level enhancement required to complete the SIRF-1 mirroring scheme is the modification of the read request processing algorithm. By distributing read requests evenly across mirror groups the FTL can take increased advantage of channel level parallelism. The approach implemented in this research is a simple round-robin distribution of read operations across mirror groups. As multi-page read requests are processed, they are broken down into individual page reads. When the requested LPN is resolved to a PPN, the algorithm alternates between the two mirror group entries in the mapping table. Table 2 illustrates the resolution of PPNs to service a read request of LPNs seven through eleven using the mapping table shown in Table 1. The current approach guarantees that read operations will be evenly distributed across mirror groups over time. However, a potential enhancement would be to choose a channel based on its state when the read is actually carried out.

The design for implementing SIRF-1 requires modifying several components of the FTL. In order to implement SIRF-1 the read/write request processor, mapping table, and allocation scheme of SSDSim were enhanced. By implementing this design within the FTL, an SSD can effectively create and

manage mirrored data without requiring substantial changes to its hardware. By limiting changes to the software/ firmware of the SSD, our approach to SIRF-1 can be more easily integrated into existing SSD designs.

TABLE II. SEQUENTIAL READ REQUEST PPNS

LPN	PPN	Channel
7	02	0
8	21	1
9	07	0
10	17	1
11	04	0

D. Garbage Collection

The enhancement of the FLT allows it to make read and write decisions based on parallelism across channels. For write operations, writes are mirrored to both mirror groups. For read operations, reads are distributed to maximize parallelism. A potential issue is that for the same effective storage, the SSD has twice the physical storage it must perform garbage collection on. If not accounted for, these extra garbage collection operations could hinder the performance of the drive. As noted above the mirrored writes, will be distributed across channels, but the two mirrors may not be identical. For this reason, garbage collection operations may also be asymmetrical so that each mirror must be handled independently. However, garbage collection operations are constrained to operate within their mirror group. This allows the garbage collection of both mirrors to execute simultaneously in parallel. This ensures that a performance penalty is not incurred due to the larger amount of physical storage.

IV. PERFORMANCE EVALUATION

This section presents an assessment of the performance of the SIRF-1 architecture, as well as the methodology for performing the evaluation. Three primary metrics were considered when evaluating performance. The first of these is the mean response time (MRT), which is the mean time the SSD required to service all requests of a single trace. The MRT gives an overall assessment of how the SSD performed for a given workload. The second performance metric is the mean read response time (MRRT), which measures the average time required to service all read requests. The final metric is the mean write response time (MWRT), which measures the average time required to service all write requests. While the MRT describes overall performance metric, the MWRT and MRRT give additional insight into how SIRF-1 will affect read and write performance under different workloads

A. Simulation Environment

Analyzing the performance of the SIRF-1 mirroring architecture requires the collection of performance data under various enterprise-level workloads and configurations. Since it would be impractical to build a custom hardware implementation for each configuration, the evaluation was carried out using an SSD simulator. Rather than developing a new simulator, SIRF-1 was implemented within SSDSim. SSDSim is an IO event driven SSD simulator previously validated by comparing its performance predictions against actual hardware devices [5]. Given a trace file, SSDSim simulates the operation of a real SSD by calculating the time required to service each request based on the configuration.

SSDSim was extended to support SIRF-1 by modifying several functional areas of the simulator including the trace file pre-processor, read request handler, write request handler, mapping table, allocation algorithm and buffer management. In total, approximately 750 lines of code were added and/or modified in the existing codebase of approximately 15,000 lines of code. While, SSDSim was modified according to the implementation strategy outlined in Section 3, there are several items of consideration specific to the implementation within SSDSim. First, SSDSim's preprocessing mechanism scans the entire trace file for all read operations and performs "pre-simulation" write operations to simulate the presence of preexisting data on the drive. This process was enhanced to mirror the preprocessing write operations. SSDSim uses a dynamic allocation scheme, which selects free pages based on which channels and chips are either busy or idle. This algorithm was enhanced to limit the choice of channels based on the mirror group specified by the write request. Since SSDSim implements a page-level mapping table, every page write operation must update the mapping table for all mirror groups. The buffer management modifications also were made at the page level, since individual pages are stored in the buffer, impacting both cache hit/miss evaluation and read-before-update operations.

B. Experimental Setup

Simulations were performed using traces captured from real-world systems. Table 3 illustrates the properties of the various traces files used. The Exchange trace was obtained from a Microsoft Exchange Email server [6]. The Build trace was captured from a Microsoft Windows build server [3]. Both of these traces contain a low request arrival rate and a fairly even distribution of read versus write requests. The TPC-C trace was collected from a server connected to a Microsoft SQL server over a storage area network. This trace contains a relatively high request arrival rate, includes more read operations than write operations and contains predominantly random read/write access [11]. The Financial1 (hereafter, Fin1) and Financial2 (hereafter, Fin2) traces both contain data collected from on-line transaction processing systems. Both traces provide a moderate request arrival rate and are characterized by random read/write access. Finl contains predominately write operations whereas Fin2 contains predominately read operations [17].

TABLE III. CHARACTERISTICS OF THE TRACES

Trace	Requests	Write %	Read %	Avg. Reqts. / Sec	Avg. Reqts. Size
Exchange	67,087	69.7%	30.3%	14	12 KB
Build	209,020	28.1%	71.9%	28	8 KB
TPC-C	368,606	33.5%	66.5%	301	8 KB
Fin1	5,334,948	76.8%	23.2%	123	3 KB
Fin2	3,698,863	17.7%	82.3%	92	4 KB

Each workload was evaluated under a variety of SSD configurations to analyze the impact of implementing SIRF-1 mirroring on drive performance. SSDSim provides a wide array of configuration parameters. Many parameters were fixed for all simulations, while others were varied to analyze the scalability of the SIRF-1. Table 4 shows several of the key simulation parameters.

TABLE IV. SIMULATION PARAMETERS

Parameter	Value
Dies per Chip	4
Planes per Die	4
Blocks per Plane	1024
Pages per Block	64
Page size (KB)	4 – (2, 4, 8)
DRAM per GB of Storage	0.5MB
Block erase latency	1500µs
Page write latency	200µs
Page read latency	20µs

Table 5 shows the test configurations that were used during simulation and the values of the parameters that were varied for each test. All tests used the fixed parameters shown in Figure 4. Test configurations are named by the number of channels and a 'C' followed by either an 'M' if mirroring was enabled, or an 'NM' if mirroring was disabled, followed by a "-" and the total amount of flash memory in gigabytes. For example the 1CNM-64 configuration used 1 channel, no mirroring, and 64GB of flash memory. Similarly, the 2CM-128 configuration used 2 channels, with mirroring enabled and 128GB total flash memory.

Test configurations were executed in pairs to evaluate the performance impacts of adding mirroring to various configurations. Each pair contains one non-mirrored and one mirrored test configuration. Both tests in a given pair yield the same effective storage capacity to the user. A test pair (TP) is identified with a test pair ID (TP ID) that is simply the letters "TP" followed by a single digit (e.g. TP1). TP1 through TP3 explore the scalability SIRF-1 as the number of channels is increased in an SSD with a fixed effective storage capacity. TP4 through TP6 explore the scalability of SIRF-1 as the capacity of the SSD is increased while fixing the amount of flash memory per channel. The latter is achieved by fixing the number of channels contributing to the effective storage capacity within each test set. For all tests, the number of channels is doubled when mirroring is enabled.

C. Experimental Result Analysis

To evaluate the performance of SIRF-1, each workload was simulated using all configurations listed in Table 5. Several statistics were captured that were used to calculate the three key metrics used to evaluate performance. The most important of these metrics is the *mean response time* (MRT), which measures the average time to serve all requests in the trace file. By analyzing the MRT between the two tests in a test pair for a particular workload, a comparison of the overall performance of the two configurations can be made. Additionally, the *mean read response time* (MRRT) and *mean write response time* (MWRT) were also computed. The MRRT and MRWT measure the average time required to service all read requests and write requests, respectively, in the trace file. These two metrics give greater insight into how the different configurations affect the read and write operations independently. The MRRT and MWRT metrics are especially useful since it is well understood how existing RAID level impact read and write performance. The MRRT and MWRT metrics allow for a more natural comparison between the corresponding levels of SIRF and RAID. Additionally, by comparing the MRRT and the MWRT with the read/write composition of the workloads, more insightful conclusions can be drawn as to how future workloads might perform based on the predicted read/write characteristics.

TABLE V. VARIED SIMULATION PARAMETERS

TP ID	Test ID	# of Channels	Flash Memory	Storage Capacity	DRAM
TP1	1CNM-64	1	64GB	64GB	32MB
TP1	2CM-128	2	128GB	64GB	64MB
TP2	2CNM-64	2	64GB	64GB	32MB
TP2	4CM-128	4	128GB	64GB	64MB
TP3	4CNM-64	4	64GB	64GB	32MB
TP3	8CM-128	8	128GB	64GB	64MB
TP4	4CNM-32	4	32GB	32GB	16MB
TP4	8CNM-64	8	64GB	32GB	32MB
TP5	4CNM-64	4	64GB	64GB	32MB
TP5	8CM-128	8	128GB	64GB	64MB
TP6	4CNM-128	4	128GB	128GB	64MB
TP6	8CM-256	8	256GB	128GB	128MB

Figure 4 illustrates how SIRF-1 performs in terms of MRT across the five workloads as the number of channels is increased. TP1 contains 1 channel per mirror, TP2 contains 2 channels per mirror, and TP3 contains 4 channels per mirror. Table 6 shows additional detail by providing the percent change in performance between the two tests in each pair for the MRT, MRRT, and MRWT. The percent change stated is the performance of the SIRF-1 configuration relative to the non-mirrored configuration.

Several general trends can be seen in these results. First, performance improves across all tests as the number of channels increases, because there is more channel-level parallelism for the drive to exploit. Second, the most benefit is achieved in TP1 where the non-mirrored configuration only has 1 channel. In this case the non-mirrored drive has no channel-level parallelism to take advantage of. Adding the mirror dramatically improves performance since it introduces channel-level parallelism where the non-mirrored configuration had none. Lastly, in general MRRT improved more than MRWT. Other test pairs show less improvement since the SSD already had some measure of channel-level parallelism to leverage. It is intuitive that read performance is improved since reads can be distributed across the channels in the two mirror groups. It is less obvious why write performance would improve at all given that the major change in write handling is to duplicate the write operations required to service a write request. On the surface, this approach would be expected to

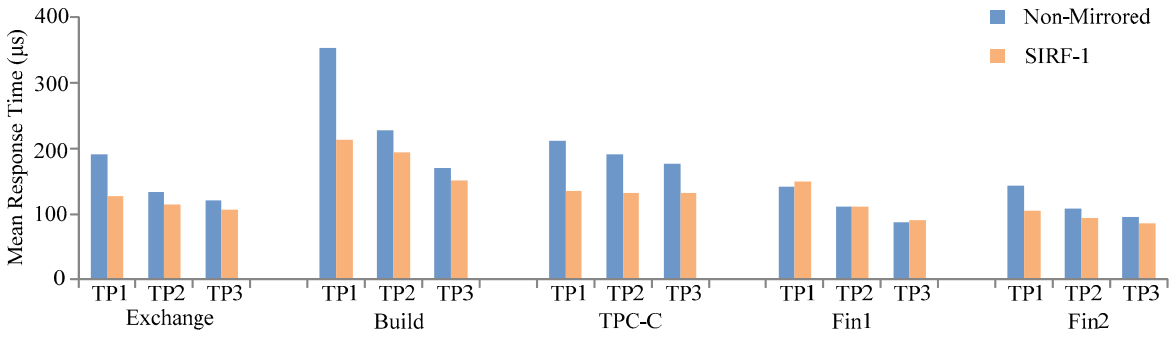

Fig. 4. Performance with channel scalability.

TABLE VI. RELATIVE PERFORMANCE IMPROVEMENTS FROM TEST PAIRS 1 THROUGH 3

	TP1 (1CNM-64 v. 2CM-128)			TP2 (2CNM-64 v. 4CM-128)			TP3 (4CNM-64 v. 8CM-128)		
Trace	***ΔMRT***	***ΔMRRT***	***ΔMWRT***	***ΔMRT***	***ΔMRRT***	***ΔMWRT***	***ΔMRT***	***ΔMRRT***	***ΔMWRT***
Exchange	33.6%	48.6%	17.2%	13.6%	9.9%	16.4%	12.3%	8.8%	14.4%
Build	39.5%	44.5%	3.9%	14.7%	17.0%	2.6%	11.4%	13.7%	2.0%
TPC-C	36.4%	46.9%	13.7%	30.8%	39.8%	12.7%	24.9%	31.8%	12.7%
Fin1	-5.5%	20.7%	-14.3%	0.4%	16.1%	-5.2%	-2.3%	10.4%	-7.3%
Fin2	27.3%	28.2%	15.7%	13.7%	13.6%	14.6%	10.0%	9.6%	14.1%

cause an overall decrease in write performance. However, the unique functionality of SSDs leads to a counterintuitive result.

Similar to read operations, duplicated mirror write operations are distributed across mirror groups and will therefore utilize different channels. This allows the mirrored write operations to execute in parallel. While there is some processing overhead required for the controller to create and issue the additional write operation, much of the two operations can execute simultaneously. Additionally, in update scenarios, a read operation must be performed in order to retrieve existing page data before the new page data can be written to flash memory. The write operation will be blocked until this read operation is completed. Given that the read performance of the SSD is increased dramatically due to mirroring, the amount of time the write request is blocked waiting for its read-before-update operation to complete is actually reduced. In addition, the FTL was modified to require only one page read to service both mirrored write operations, again reducing the overhead of duplicating the write operations to maintain the mirror.

Beyond the general trends, insight can be gained from the differences in the way each workload performed. Unlike all other workloads, Fin1 shows a decrease in overall MRT. Fin1 is a heavily write predominant workload with small, randomized IO operations. The benefit gained in MRRT is still 20.7%, however there are so few read operations as compared to write operations that the performance gain in MRRT alone does not significantly impact the overall MRT. Furthermore the write requests are received at a moderate pace and are mostly random, resulting in many more cache misses and therefore many more read-before-update operations. With more read-before-update and write operations consuming the drive, there is less opportunity to exploit parallelism during writes. Without the benefit of parallelism in writes, only the penalty incurred from duplicating write operations is left to influence the MWRT.

While the Exchange workload is also write dominant, it contains a less skewed distribution of operations and has a much lower average request arrival rate. With a lower request rate, the SSD has greater opportunity to exploit parallelism for all operations. The Build, TPC-C, and Fin2 workloads all contain a majority of read operations and therefore consistently show an overall increase in performance with mirroring enabled. Interestingly, the Fin2 workload gains less performance than the Build workload due to the smaller average request size of the Fin2 workload. Smaller requests are less able to leverage parallelism. For example if a request can be handled with a single page read operation, then the entire request can be handled with a single channel. The presence of a mirror channel will be of little benefit to that specific request. Furthermore, with only a moderate request arrival rate, the odds of a particular channel being busy when a request is received is also reduced, again minimizing the need for parallelism. These factors limit the amount of performance gain achieved in the Fin2 workload as compared to the other read dominant workloads.

The second set of tests, pairs TP4 through TP6, evaluate the scalability of SIRF-1 as the capacity of the drive is increased while holding the number of channels per mirror group fixed. Figure 5 illustrates how the MRT changes for each workload as the amount of effective storage is increased from 32GB to 64GB and finally to 128GB. Note that in the SIRF-1 configuration the actual capacity of the SSD is 64GB, 128GB, and 265GB respectively.

In general, the performance gains presented for the second set of tests are less pronounced than in the first set. The main contributing factor to this is that the channel configuration used

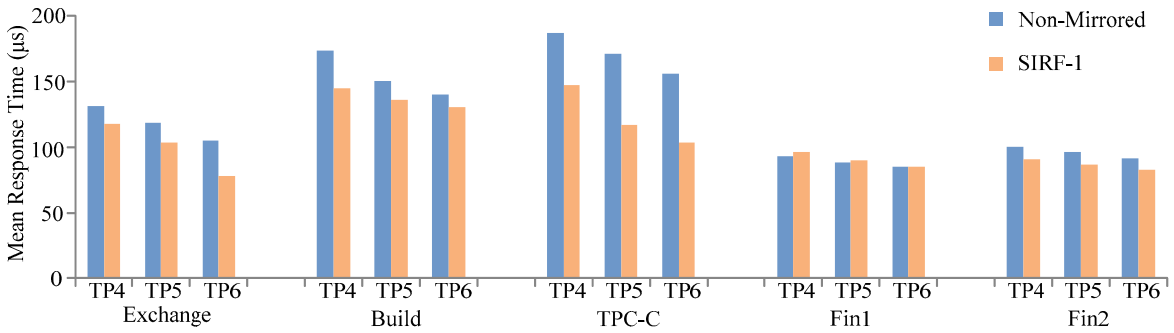

Fig. 5. Performance with capacity scalability.

TABLE VII. RELATIVE PERFORMANCE IMPROVEMENTS FROM TEST PAIRS 4 THROUGH 6

	TP4 (4CNM-32 v. 8CNM-64)			TP5 (4CNM-64 v. 8CM-128)			TP6 (4CNM-128 v. 8CM-256)		
Trace	***ΔMRT***	***ΔMRRT***	***ΔMWRT***	***ΔMRT***	***ΔMRRT***	***ΔMWRT***	***ΔMRT***	***ΔMRRT***	***ΔMWRT***
Exchange	10.2%	13.7%	8.2%	12.7%	9.0%	14.8%	25.3%	8.6%	36.4%
Build	16.8%	20.3%	4.0%	9.5%	11.8%	1.9%	7.2%	8.5%	2.8%
TPC-C	21.1%	29.5%	6.1%	31.5%	31.5%	12.7%	33.8%	33.8%	27.5%
Fin1	-3.5%	10.1%	-8.7%	-2.3%	10.4%	-7.3%	-0.1%	12.1%	-5.0%
Fin2	9.8%	9.8%	10.4%	10.0%	9.6%	14.1%	9.6%	8.8%	19.5%

for the second set of tests is identical to those used in TP3 above, which yielded the least amount of performance gain. This configuration was selected because the 4-channel and 8-channel configurations used in TP3 are more representative of the number of channels utilized in the SSDs on the market at the time this paper was written. Furthermore, the relative performance of the five workloads is consistent with the first three test pairs. Fin1 still pays an overall MRT penalty due the duplication of write request while the other four tests see an overall improvement in the MRT. Another general trend is that the performance of both the non-mirrored and SIRF-1 configurations increases as the capacity of the SSD increases. This is expected, as the SSD will have more room to perform out-of-place updates. This will reduce the amount and frequency of the garbage collection.

Several insights can be observed from Figure 5. For the Fin1, TPC-C and Exchange traces, the performance between the non-mirrored and SIRF-1 configurations actually improves as the SSD's capacity grows. While initially unintuitive, this can be understood by examining the use of the cache for these traces. These workloads are predominately random access in nature. This means that the cache will be strained to provide performance benefits due to the weaker locality of the data access. The simulation configurations of the SSD proscribe that the amount of DRAM scales linearly with the capacity of the drive, as is the case in most commercially available drives. However, since the amount of data in the trace file remains the same, as the SSD capacity increases, the size of the cache with respect to the amount of the data in the trace increases, providing more cache coverage. In the SIRF-1 configuration, the DRAM is doubled from the non-mirrored configuration since the capacity of the SSD is doubled. However, the effective capacity of the SSD remains the same, further widening the margin of extra DRAM available for caching. These factors lead to many more cache hits and far fewer read-before-update operations, which greatly improve the MWRT and MRT of the SIRF-1 configurations.

The purpose of the last group of tests is to understand the performance impacts of page size on SIRF-1. We only show page size impacts in TP1 configuration. Figure 6 demonstrates that the overall results are consistent with previous results with TP1. The results for the Exchange, Build, TPC-C, and Fin2 show significant improvements whereas the Fin1 shows only minimal improvements or a reduced overall performance for the same reasons as discussed earlier. It should be noted that all other parameters were held constant while the page size was varied. As the page size is increased, the capacity of the SSD is increased while the amount of DRAM cache is held constant. This results in an overall relative performance reduction as the page size is increased. Another interesting discovery is that while the results are still positive overall, the improvement is reduced as the page size is increased from 2KB to 8KB. The reason for this is that the larger page size actually reduces the opportunity for parallelism. Consider a request for 6KB of data from the operating system. With a page size of 2KB the request will be decomposed in to 3 individual page requests, which can be spread across the mirrored channels. However, if the page size is set to 8KB, then the request from the operating system will be handled by a single read request, which reduces the gain from parallelization.

V. CONCLUSIONS

The goal of this research was to evaluate the potential to improve the reliability and performance of SSDs by introducing internal RAID-1 style mirroring. SIRF-1 leverages the internal parallelism afforded by the flash memory organization of an SSD to mirror data in an efficient manner.

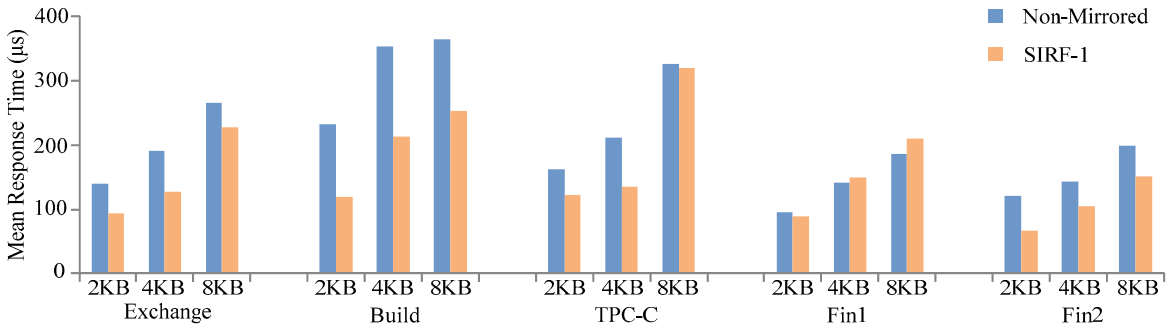

Fig. 6. Performance impacts of page size.

Improving the reliability of SSDs will allow them to support the vital need for reliable storage in mobile devices as they continue to support functions in business, mission critical, and personal use cases.

To evaluate SIRF-1, an existing validated SSD simulator SSDSim has been extended to implement it. The performance of SIRF-1 is evaluated by running several real-world workloads using the modified simulator. Finally, the performance implications of SIRF-1 are collected, presented, and analyzed. An average performance increase in MRT of 15% is observed across all tests and workloads. The best-case improvement observed for configurations that contained 2 or more channels was shown to be 39.5%. Workloads consisting of high rates of random reads benefit the most due to additional parallelism that overshadowed cache gains due to weak request locality. In the worst-case, a 5.5% decrease in performance is shown for a write-intensive trace. Enabling SIRF-1 will halve the effective capacity of the SSD. Therefore a penalty will be paid in either capacity, if the cost must be held constant, or in cost, if capacity must be maintained. This is the same trade-off scenario that is well understood in RAID implementations. Fortunately, the price of flash memory is dropping quickly, which makes SIRF-1 become increasingly practical.

In the future, SIRF-1 will be compared to other SIRF / RAID levels such as RAID-4 and RAID-5[7]. Each of the additional SIRF levels should be analyzed to determine the corresponding reliability, performance, and cost trade-offs. Future research will also investigate the interplay between SIRF methods and other SSD design choices.

ACKNOWLEDGMENT

This work was supported by the U.S. National Science Foundation under grant CNS (CAREER)-0845105.

REFERENCES

[1] N. Agrawal, et al., “Design Tradeoffs for SSD Performance,” Proc. USENIX Annual Technical Conference, pp. 57-70, 2008.

[2] M. Balakrishnan, A. Kadav, V. Prabhakaran, and D. Malkhi. "Differential RAID: rethinking RAID for SSD reliability." ACM Trans. on Storage, Vol. 6, No. 2 (2010): 4.

[3] Build Server Trace, SNIA IOTTA Repository, http://iotta.snia.org/traces/158, Accessed 2010-04-20.

[4] F. Chen, R. Lee, and X. Zhang, “Essential roles of exploiting internal parallelism of flash memory based solid state drives in high-speed data processing,” HPCA, 2011.

[5] Y. Hu, et al. "Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity." In Proceedings of the international conference on Supercomputing, pp. 96-107. ACM, 2011.

[6] Exchange Trace, SNIA IOTTA Repository, http://iotta.snia.org/traces/130, Accessed 2010-04-20.

[7] K. Greenan, et al. "Building flexible, fault-tolerant flash-based storage systems," The Fifth Workshop on Hot Topics in Dependability (HotDep’09), 2009.

[8] L.M.Grupp, A.M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P.H. Siegel, and J. K. Wolf. "Characterizing flash memory: anomalies, observations, and applications." In Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium, pp. 24-33. IEEE, 2009.

[9] A. Gupta, Y. Kim, and B. Urgaonkar. “DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings”, ASPLOS, 2009.

[10] S. Im, D. Shin, "Flash-Aware RAID Techniques for Dependable and High-Performance Flash Memory SSD," IEEE Trans. on Computers, vol.60, no.1, pp. 80-92, Jan 2011

[11] S.T. Leutenegger and D. Dias, “A modeling study of the TPC-C benchmark,” in Proc. ACM Int’l Conf. Management of Data, 22(2), pp. 22-31, 1993.

[12] S. Moon, N. Reddy, "Don’t Let RAID Raid the Lifetime of Your SSD Array," Proc. of HotStroage 2013, 2013.

[13] W. Pan, F. Liu, T. Xie, Y. Gao, Y. Ouyang, and T. Chen, “SPD-RAID4: Splitting Parity Disk for RAID4 Structured Parallel SSD Arrays,” Proc. 15th IEEE Int’l Conf. High Performance Computing and Communications (HPCC), Zhangjiajie, China, November 13-15, 2013.

[14] K. Park, D. Lee, Y. Woo, G. Lee, J. Lee, and D. Kim. “Reliability and performance enhancement technique for SSD array storage system using RAID mechanism.” Proc. 9th IEEE Int’l Symposium on Communications and Information Technology, pp. 140-145, 2009.

[15] S. Park, E. Seo, J.Y. Shin, S. Maeng, and J. Lee, “Exploiting Internal Parallelism of Flash-based SSDs,” IEEE Computer Architecture Letters, Vol. 9, No. 1, pp. 9-12, 2010.

[16] D.A. Patterson, G. Gibson, and R.H. Katz, “A Case for Redundant Arrays of Inexpensive Disks (RAID),” ACM SIGMOD 88, Vol. 17, No. 3, pp. 109-116, June 1988.

[17] J. Shin, et al., “FTL design exploration in reconfigurable high-performance SSD for server applications,” Proc. of the 23rd Int’l Conf. Supercomputing, 2009.

[18] J.M. Smith, “The doctor will see you ALWAYS,” IEEE Spectrum, Vol. 48, No. 10, pp. 56-62, October 2011.

[19] Department of Defense Mobile Device Strategy, http://www.defense.gov/news/dodmobilitystrategy.pdf, 2012.