

Evaluating Reliability of SSD-Based I/O Caches in Enterprise Storage Systems

Saba Ahmadian, Farhad Taheri, Hossein Asadi, *Senior Member, IEEE*

Abstract—I/O caching techniques are widely employed in enterprise storage systems in order to enhance performance of I/O intensive applications in large-scale data centers. Due to higher performance compared to *Hard Disk Drives* (HDDs) and lower price and non-volatility compared to *Dynamic Random-Access Memories* (DRAM), Flash-based *Solid-State Drives* (SSDs) are used as a main media in the caching layer of storage systems. Although SSDs are known as non-volatile devices but recent studies have reported large number of data failures due to power outage in SSDs. To overcome the reliability implications of SSD-based I/O caching schemes, RAID-1 (mirrored) configuration is commonly used to avoid data loss due to uncommitted write operations. Such configuration, however, may still experience data loss in the cache layer due to correlated failures in SSDs. To our knowledge, *none* of previous studies have investigated the reliability of SSD-based I/O caching schemes in enterprise storage systems.

In this paper, we present a comprehensive analysis investigating the reliability of SSD-based I/O caching architectures used in enterprise storage systems under power failure and high-operating temperature. We explore variety of SSDs from top vendors and investigate the cache reliability in mirrored configuration. To this end, we first develop a physical fault injection and failure detection platform and then investigate the impact of workload dependent parameters on the reliability of I/O cache in the presence of two common failure types in data centers, *power outage* and *high temperature* faults. We implement an I/O cache scheme using an open-source I/O cache module in Linux operating system. The experimental results obtained by conducting more than twenty thousand of physical fault injections on the implemented I/O cache with different write policies reveal that the failure rate of the I/O cache is significantly affected by workload dependent parameters. Our results show that unlike workload requests access pattern, the other workload dependent parameters such as request size, *Working Set Size* (WSS), and sequence of the accesses have considerable impact on the I/O cache failure rate. We observe a significant growth in the failure rate in the workloads by decreasing the size of the requests (by more than 14X). Furthermore, we observe that in addition to writes, the read accesses to the I/O cache are subjected to failure in presence of sudden power outage (the failure mainly occurs during promoting data to the cache). In addition, we observe that I/O cache experiences *no* data failure upon high temperature faults.

Index Terms—Flash-Based Solid-State Drives (SSDs), Storage Systems, I/O Cache, Reliability Analysis, Power Outage.

1 INTRODUCTION

Increasing the I/O intensive applications such as *Online Transaction Processing* (OLTP) and banking services makes storage subsystems built upon *Hard Disk Drives* (HDDs) as the performance bottleneck of enterprise systems. In order to alleviate the performance shortcomings of HDD-based storage subsystems, enterprise manufacturers such as Dell EMC, NetApp, and HP [1–3] and emerging storage architectures [4–6] employ high performance flash-based devices such as enterprise *Solid-State Drives* (SSDs) as a cache layer for disk subsystem, which is mainly composed from low-performance HDDs and mid-range flash-based SSDs (as depicted in Fig. 1). SSDs are non-volatile devices which because of their non-mechanical design provide higher performance and lower power consumption compared to HDDs [5, 7–10]. In addition, SSDs cost about 20X lower than volatile *Dynamic Random-Access Memories* (DRAMs) and also do not require additional peripherals such as backup batteries to retain data in case of power outage [11].

Employing SSD-based I/O caches in enterprise storage systems can enhance the performance of I/O intensive applications. In such SSD-based I/O cache architecture, however, the SSD cache becomes the single point of failure because of buffering write pending requests where any failure in the SSD device leads to data loss. Although SSDs are known as non-volatile devices but recent studies such as [12–14] have reported different types of failures such as data, metadata, and device failures in the SSDs under *power outage*. To enhance the reliability of I/O cache and reduce the probability of data loss, enterprise storage systems such as Dell EMC and HP employ *Redundant Array of*

Independent Disks (RAID) [15] in the configuration of I/O caches [1, 3]. In such architecture, multiple SSDs are typically configured as RAID-1¹ in the cache layer of enterprise storage systems (Fig. 1). Such configuration improves both performance and reliability but cannot completely resolve the reliability issues at the RAID level. The mirrored configuration keeps two copies of each data in two different devices known as *primary* and *secondary* disks (i.e., primary is the one that the RAID controller chooses to write first) and provides high level of reliability upon disk failures [15–21]. For each write request coming from the application, two identical write operations are performed in both primary and secondary disks. RAID-1 configuration doubles the read performance of the disk subsystem by involving *only* one of the disks which has a minimum queue size and service time for read operations. Such configuration tolerates *disk failures* in the subsystem while data failures such as shorn writes (i.e., incomplete writes), flying writes (i.e., misplaced writes), and unserializability (i.e., out-of-order writes) reported in [12–14, 22] cannot be tolerated completely [15–21]. However, a process namely *inconsistency check* or *scrubbing* runs in the background and regularly checks the consistency of primary and secondary disks in RAID configuration.² In RAID-1 configuration, in case of inconsistency (i.e., when the comparison of primary and secondary disks fails), the data of primary disk is copied to the secondary disk. Such operation may destroy the correct data stored in the secondary disk by a faulty data block in the primary disk (this issue happens when data failure is occurred in the primary disk) [23].

Recent studies such as [12–14] have *only* investigated the impact of power outage on the failures of SSDs while they have neglected the impact of other parameters affecting the reliability

• Saba Ahmadian, Farhad Taheri, and Hossein Asadi are with the Data Storage, Networks, and Processing (DSN) Lab, Department of Computer Engineering, Sharif University of Technology, Tehran, Iran.
E-mail: ahmadian@ce.sharif.edu; farhadtaheri@ce.sharif.edu; asadi@sharif.edu

1. RAID-1, also known as mirrored configuration, replicates data blocks in two or more paired devices.

2. In other types of RAID, consistency check operation checks the correctness of stripes parity.

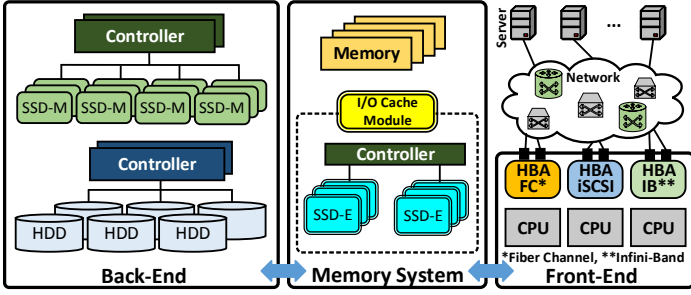


Fig. 1: Overview of an enterprise storage system (SSD-M: Midrange-SSD and SSD-E: Enterprise SSD).

of SSDs such as temperature. In addition, recent studies have *only* focused on the reliability of SSDs in non-mirrored (single) configuration under power outage while the impact of such failures on I/O cache architectures (commonly in RAID-1 configuration) has been neglected. Furthermore, the previous studies do not emulate the real power outage effect that occurs in data centers and ignore the discharge delay of large-size embedded capacitors in *Power Supply Units* (PSUs).

In this paper, we investigate the failures of SSD-based I/O caches in RAID-1 configuration, which is widely used in enterprise storage systems under two common failure types in data centers, i.e., *power outage* and *high temperature* faults. To do so, we develop a reliability test platform including Hardware-Software codesign which injects real faults that may occur in data centers such as power outage and high temperature.³ We integrate EnhanceIO as an open-source I/O cache module with the kernel to implement the I/O cache level and analyze the reliability of committed requests to the subsystem in presence of I/O cache. We classify different types of failures into 1) *False Write Acknowledge* (FWA) in which the data is not written in the SSD while the acknowledgment is received in the application level, 2) *Full Data Corruption* (FDC) in which entire data is corrupted. 3) *shorn writes* (i.e., incomplete writes) in which only a portion of data is written in the SSD, 4) *flying writes* (i.e., misplaced writes) where the write operation is performed in a wrong address, 5) *unserializability* issue which is due to the concurrent writes from different threads to an identical address, and 6) *I/O error* (i.e., failures due to disk unavailability) in the disk subsystem. Furthermore, the proposed test platform measures the temperature, current, voltage, and power consumption of under test SSDs.

Using the proposed physical fault injection platform, we conduct real experiments on more than 10 enterprise SSDs from different vendors. In our experiments, we first study the impact of workload dependent parameters such as 1) workload *Working Set Size* (WSS), 2) request size, 3) request type, 4) access pattern, and 5) sequence of the accesses on the ratio of different failures on the I/O cache (including SSDs in mirrored configurations). Second, based on the ratio of detected failures, we propose a comprehensive classification of failures that occur in I/O caches due to power outage.⁴ Our results show that several workload dependent parameters such as request size and workload WSS have significant impact on the failure rate (about 14X and 44%, respectively) while the others such as request access pattern does

3. Several recent works present test platforms and evaluate the reliability and lifetime of other emerging technologies such as *Thermally Assisted Switching-Magnetic Random Access Memory* (TAS-MRAM), *Resistive Random-Access Memory* (RRAM) [24], and memristors [25]. Our work mainly evaluates the reliability of SSDs.

4. In this work, we evaluate the impact of workload characteristic on the reliability of I/O caches. Our initial experiments reveal different levels of I/O cache reliability for various cache configurations such as write policy (evaluated in this work), block size, and replacement policy. These observations show the importance of assigning efficient cache configuration for the applications based on their workload characteristics to get the maximum reliability under power outage faults. The investigation of the impact of cache configuration parameters on the reliability of I/O caches will be considered as a future work of our study.

not considerably affect the failure ratio (only by 2%). In our experiments, we observe no data failure in the I/O cache in presence of high temperature faults. In addition, the experiments reveal that both read and write requests fail upon power outage in the I/O cache with different write policies such as 1) *Write Back* (WB), 2) *Write Through* (WT), and 3) *Read Only* (RO).

The main contributions of this work are as follows.

- To our knowledge, this paper is the first to investigate the impact of power outage and high temperature faults which commonly occur in datacenters on the SSD-based I/O cache architectures in enterprise storage systems.⁵
- We develop a physical reliability test platform to inject power outage and high temperature faults which can distinguish 1) *False Write Acknowledge* (FAW), 2) *Full Data Corruption* (FDC), 3) *shorn writes*, 4) *flying writes*, 5) *unserializable writes*, and 6) *I/O error* failures. The proposed test platform as the first proposed physical framework measures the current, temperature, and power consumption of the SSDs during test.
- By conducting a set of extensive workloads, we study the impact of workload dependent parameters such as 1) workload *Working Set Size* (WSS), 2) request size, 3) request type, 4) access pattern, and 5) sequence of the accesses on the reliability of the SSD-based I/O cache architectures with different write policies such as 1) *Write Back* (WB), *Write Through* (WT), and *Read Only* (RO). We observe a significant impact of workload dependent parameters such as request size and workload WSS on the failure ratio in presence of power outage fault. In addition, we observe that high temperature faults have no impact on the failure of SSDs in I/O cache layer of storage systems.
- We conduct real experiments on more than ten enterprise SSDs from different vendors by injecting more than twenty thousand of physical faults and examine the ratio of data failures under different faults.

The rest of the paper is organized as follows. Section 2 discusses related work. In Section 3, we characterize different types of failure that occur on the SSDs. In Section 4, we present our proposed test platform. Section 5 provides our proposed evaluations and observations. Finally, Section 6 concludes the paper.

2 RELATED WORKS

In this section, we first provide previous studies that investigate the reliability of SSDs and flash-based devices. Then we present the shortcomings of previous studies and show the key advantages of our study.

Based on the *source of failures*, the previous studies on reliability of flash-based devices can be investigated in two groups. The first group analyzes the failure of the flash-based memories that are mainly due to the internal structure of devices. Failures such as endurance, read disturbance, and write disturbance that are originated from the structure of flash-based devices (i.e., there is no external reason for such failures), are studied in the first group. The second group mainly focuses on the failures in the flash-based devices that are mainly due to external events such as power outage and other environmental reasons.

Previous studies such as [22, 26, 27, 30–36] mainly analyze the failure of flash-based devices such as read disturbance, write disturbance, and endurance. Such failures are commonly reported in chip and device levels. SSD failures in Google and Facebook datacenters during six and four years of operation are studied in [26, 35]. Meza et al. observed similar “bathtub curve” in the failure trend of SSDs and an additional phase namely “early

5. In this work, we target the impact of power outage and high operating temperature on the reliability of I/O caches. Other important parameters such as SSD aging can affect the reliability of data in SSDs, which is partially reported in [22, 26–31].

detection". Life cycle of SSDs consists of four phases: 1) early detection, 2) early failure, 3) usable lifetime, and 4) wear-out phase. Meza et al. conclude that SSDs in their wear-out phase do not experience a monotonic failure rate. In addition to [26], other studies such as [33] observed more realistic results of SSD failures in production environment.

The other parameters of flash-based memories such as performance, power consumption, and reliability of devices are measured in [32, 34]. The results reveal a significant difference between the measured and reported values in the datasheets of the products. The measured parameters are used to present the most reasonable trade-off in the design of storage systems.

The impact of power outage in embedded systems is investigated in [37]. A software-based test platform is proposed which mainly simulates the effects of power outage on the *Flash Translation Layer* (FTL) of SSDs and file systems at the *Operating System* (OS) layer. The proposed test platform is programmed to detect a group of previously defined types of failures that occurs during simulation. Such a platform neglects the effects of realistic power failures and only is able to detect a limited number of failures that may occur in simulation.

Recent studies such as [12, 13] have investigated the reliability of flash cells and SSDs under power outage. Such studies have proposed a test platform that injects realistic power outage faults to the devices. Tseng et al. in [12] have proposed an FPGA-based reliability test platform where the power of under test flash cells is cut by high-speed transistors with less than $3.7\mu s$ delay. The results of the experiments in [12] show several failures in the flash cells (i.e., in the chip level design) due to power outage. Such study investigates the reliability of flash cells in the chip level design which neglects several recovery mechanisms that is employed in device level designs such as SSDs.

In upper level designs of flash-based devices, most of chip level failures are masked and would not result in data failure in the application layer. However, such designs suffer from other types of failure that may not occur in chip level and hence, investigating the reliability of the SSDs (i.e., device level design) is required. To this end, later studies such as [13] have evaluated the reliability of SSDs under power outage.

Zheng et al. have proposed a reliability test framework that explores the impact of realistic power outage faults on SSDs. They have analyzed fifteen SSDs from five different vendors and reported several failures for thirteen out of fifteen SSDs [13]. Similar to previous studies, [13] neglects the impact of workload dependent parameters on the failure ratio where they only examine the reliability of SSDs under a fixed and simple workload. In addition, the power failure is performed by high-speed transistors where the input voltage of under test SSDs is dropped to zero in microsecond-order delay. In such condition, the impact of large size capacitors that are provided in *Power Supply Unit* (PSU) is neglected where the SSDs under test do not experience the real power outage process that occurs in the systems. Experimental results in [14] show that the fall delay of PSU output (voltage drop from 5v to 0) takes more than 900ms where the SSDs become unavailable once the input voltage drops to 4.5v that takes about 40ms.

The impact of workload dependent parameters on the failure ratio is examined in [14] by injecting realistic power outage failures. However, none of previous studies investigated the impact of workload dependent parameters on the reliability of SSD-based I/O caches in the enterprise storage systems in the presence of realistic faults. In addition, the proposed test platform in [14] does not provide the current, temperature, and power consumption of the under test SSDs. Furthermore, [14] neglects the impact of other common types of faults such as high temperature on the failure of SSDs in datacenters.

3 FAILURE CHARACTERIZATION IN SSDS

In this section, we elaborate different types of failures that occur

in case of either power outage or high temperature (in Section 4.2, we will elaborate on how we detect the failures using our proposed reliability test platform). Fig. 2 shows SSD failures and their relation. As shown in Fig. 2, the failures are categorized in three groups namely: 1) *data failure*, 2) *I/O error*, and 3) *dead device*. *Data failures* may occur in five types: 1) *False Write Acknowledge* (FAW), 2) *unserializable writes*, 3) *shorn writes*, 4) *flying writes*, and 5) *Full Data Corruption* (FDC). In the following, we describe each type of failure and demonstrate how such failures occur.

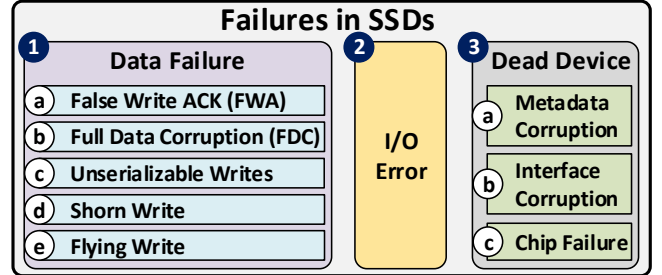


Fig. 2: Different types of SSD failures.

3.1 Data Failure

Data failure occurs when the correct data becomes unavailable after a successful write operation on the SSD. In the following, we investigate data failures in five different groups.

- 1) **False Write Acknowledge (FWA):** This failure occurs when we receive ACK in the application layer but the data is not actually written in the SSD due to power outage. This failure is mainly due to the volatile storage elements in the internal data path within the control layer of SSDs such as *host FIFO buffer* and *DRAM cache* [26, 28, 29, 38, 39].
- 2) **Unserializable Writes:** The most common *host controller interface* which schedules the requests in SSDs is *Advanced Host Controller Interface* (AHCI) [39, 40]. AHCI keeps I/O requests received from OS in a single command queue namely *Native Command Queue* (NCQ) in which the requests are queued out-of-order to provide higher performance [39]. Such condition raises challenging issues such as unserializable writes where the random and unknown order of committed writes on the SSD fail in presence of power outage which leads to data loss.
- 3) **Shorn Write:** The SCSI layer of the Linux kernel partitions each request into smaller sub-requests and commits them into the disk subsystem. In this case, the sub-requests are kept in *DRAM buffer* in the SSD internal data path where sudden power outage may corrupt one or more of the sub-requests. In this type of failure, some parts of an I/O request are written to the SSD (NAND flashes) while others are not.
- 4) **Full Data Corruption (FDC):** In this type of failure, although the write operation is completed in the SSD but the data block (including all sub-requests) is corrupted. Such failure differs from shorn writes since all sub-requests are failed. In addition, due to different initial data and final written data, such failure cannot be considered as FWA. This failure can be due to 1) errors in volatile storage elements in the SSD internal data path (such as *host FIFO buffer* and *DRAM cache* [26, 28, 29, 38, 39]) and 2) errors in NAND flashes within the SSD such as program errors [39].
- 5) **Flying Write:** This failure mainly occurs in HDDs, however, it may also occur in SSDs. Flying writes in SSDs may be due to the corruption of the SSD mapping tables which are kept in internal DRAM to provide higher performance [26, 38]. In other words, in such type of failure, the correct data is written successfully on a wrong address in presence of power outage.

3.2 I/O Error

This type of failure is due to disk unavailability through application during power outage. In this case, some of requests are committed to the disk and ACK is received while we do not receive ACK for the remaining requests. Such requests are blocked until the disk becomes available and then committed to the disk subsystem. In case of long unavailability, the application receives a timeout response from the disk subsystem. Data failure is occurred for the first group that the ACK is received but because of disk unavailability during power outage, the data is destroyed.

3.3 Dead Device

This type of failure occurs when the SSD becomes broken after multiple power outages. In case of dead device failure, the S.M.A.R.T report of the SSD provides details of the problem. Dead device failure can be experienced in three modes namely: 1) metadata corruption, 2) interface corruption, and 3) chip failure. In the following, we elaborate different types of dead device failure.

- 1) **Metadata Corruption:** This type of failure is due to the problems that occurs for FTL of the SSDs after multiple power outage. In such failure, the address map and mapping algorithms which are done by the FTL are disrupted. When this failure occurs, some address areas of the SSD become unavailable through application.
- 2) **Interface Corruption:** Multiple power outages affect the SCSI interface of SSDs (typically SATA or SAS) and disrupt the operation of this part. When this failure occurs, the SSD becomes unavailable through *scsi scan* commands from the OS. In some cases, such interface problem affects the controller of the motherboard where the other healthy disks become unavailable. Such problem is resolved by system reboot or disconnecting the faulty SSD from the system. In this type of failure, the S.M.A.R.T report of the SSD includes “*scsi error badly formed scsi parameters*” log.
- 3) **Chip Failure:** In this type of failure, the flash chips or internal connections of the SSD are disrupted where the SSD is not recognized by any controller.

4 PROPOSED RELIABILITY TEST PLATFORM

The proposed reliability test platform consists of *HardWare Module (HWM)* and *SoftWare Module (SWM)* working together where HWM is programmed and controlled by SWM. Fig. 3 shows an overview of our proposed reliability test platform. SWM is responsible for generating the I/O requests and scheduling fault injection time intervals. HWM is programmed to receive commands from SWM and inject physical faults to the SSDs. Finally, SWM is used to detect data failure and device failures that are occurred due to injected faults. In the following, we first describe SWM and its components in Section 4.1 and then in Section 4.2 we elaborate the proposed failure detection algorithm. Finally, we provide the structure of HWM in our proposed reliability test platform in Section 4.3.

4.1 SoftWare Module (SWM)

SWM generates various workloads with defined parameters such as request size, request type, access pattern, sequence of accesses, and WSS and issues the I/O requests to the disk subsystem (i.e., HDD equipped with SSD-based I/O cache configuration).⁶ SWM tracks the issued I/O requests and detects the parameters such as issue time, completion time, request size, request type, and the checksum of written data on disk. In addition, SWM keeps the

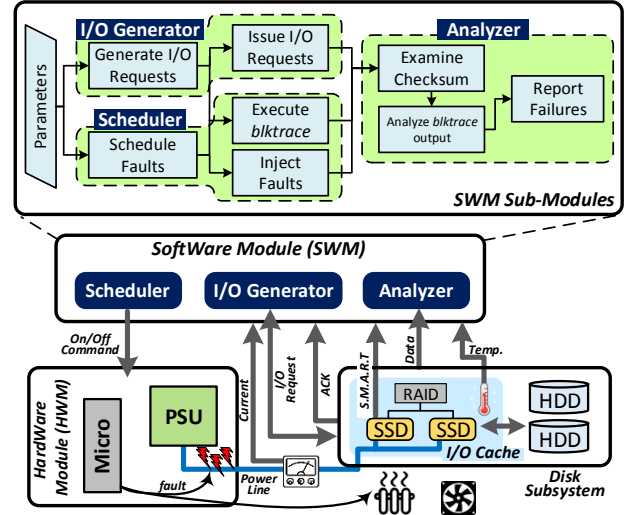


Fig. 3: Overview of the proposed test platform.

information of the I/O requests in a database in two conditions: 1) before issuing the request to the disk subsystem and 2) after issuing the I/O to the disk subsystem.⁷ SWM is designed to detect different types of failures (as discussed in Section 3) based on the information that keeps in the database and the header of the I/O requests. SWM 1) schedules fault injection intervals, 2) manages HWM to inject physical faults, and 3) receives the information about SSDs power consumption, temperature, and current.

As shown in Fig. 3, three main sub-modules in SWM namely a) *Scheduler*, b) *I/O Generator*, and c) *Analyzer* perform 1) scheduling faults, 2) submitting I/O requests to the disk subsystem, and 3) detecting failures, respectively. In the following, we describe how each part of SWM works in more details.

4.1.1 Scheduler

Scheduler is responsible for determining the fault injection intervals. It chooses random time instances that fault injection will be occurred. *Scheduler* communicates with HWM and sends *On/Off Commands* to HWM. It receives information such as 1) temperature, 2) current, and 3) power consumption of the SSDs from HWM. HWM waits to receive the commands from SWM and turns the SSDs *on* or *off* based on *Scheduler*. In addition, the *Scheduler* sets the temperature of the under test SSDs where HWM increases or decreases the temperature of the SSDs. Based on *Scheduler's* command, HWM turns on or off the heater or the fan to regulate the temperature of the test platform.

4.1.2 I/O Generator

I/O Generator creates different types of workloads with different parameters such as 1) request size, 2) request access pattern, 3) request type, 4) *Working Set Size (WSS)*, and 5) sequence of the accesses. Next, it commits the I/O requests of the workloads to the disk subsystem. The generated requests are named as *data packets* that are issued to the SSDs by the *I/O Generator*. The structure of *data packets* is shown in Fig. 4. It can be seen that a *data packet* consists of two parts: 1) header and 2) data. The data which can include *sub-requests* is produced randomly and the header includes the key information about data such as: 1) size, 2) destination address, 3) issue time (i.e., when the request enters into the disk queue), 4) completion time (i.e., when the ACK of

6. SSDs can be configured in different RAID levels, however, RAID-1 (mirrored) is the common RAID configuration for SSD-based I/O caches [1, 3].

7. Our proposed reliability test platform also is able to execute real storage workloads, in which the information about requests including 1) size, 2) address, 3) type, and 4) issue time are extracted from workload trace.

the request is received in the application layer), 5) checksum of data before issuing the request, and 6) checksum of written data to the disk subsystem (Table 1 provides the detailed information about the content of data packet). The information in the header of *data packet* is used in detecting different types of failures (in future we elaborate the failure detection mechanism). Each *sub-request* included in the data part of *data packet* consists of 1) *ID*, 2) *size*, 3) *address*, 4) *checksum*, and 5) *data* (as depicted in Fig. 4). *Data packets* are stored in a database to be used in failure detection phase.

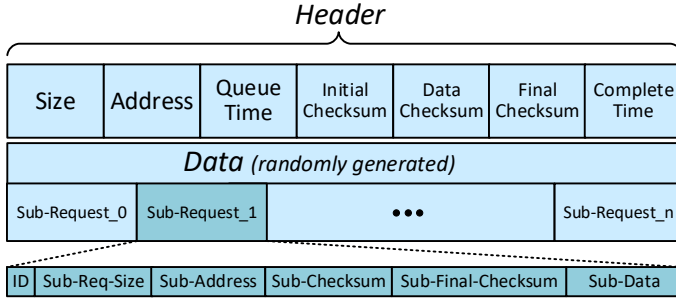


Fig. 4: Structure of *data packets*.

TABLE 1: Description of fields in data packet.

Field	Description
Size	The total size of I/O request.
Address	Destination address of the I/O request.
Queue_Time	When the I/O request enters in disk queue.
Initial_Checksum	The checksum of destination address before issuing the request.
Data_Checksum	The checksum of I/O request (including Data part).
Final_Checksum	The checksum of destination address after receiving ACK
Complete_Time	When the application receives ACK from disk.
Sub_Request_i	The i-th part of the request.
ID	The sub-request number.
Sub_Req_Size	The size of sub-request.
Sub_Address	The destination address of sub-request.
Sub_Checksum	The checksum of sub-request (only sub-data).
Sub_Final_Checksum	The checksum of sub-request after completion.
Sub_Data	The data part of sub-request.

4.1.3 Analyzer

Analyzer keeps the track of I/O requests and verifies the correctness of written data in the disk subsystem. Based on collected I/O traces, *Analyzer* compares the checksum of “completed” requests with the stored checksum of corresponding *data packet* in the database. In case of any inconsistency, the *Analyzer* reports a *data failure*. *Analyzer* employs *blktrace* as a block layer I/O tracing tool to keep the track of the requests. *Blktrace* is available in Linux kernel (version 2.6 and upper) and provides the information of I/O requests in user level without any performance overhead. *Analyzer* detects the other types of failures such as device failure and I/O error beside data failures based on the collected information by HWM. In Section 4.2, we elaborate how we trace the I/O requests and detect different types of failures.

4.2 Failure Detection

In this section, we show how our proposed failure detection algorithm of the test platform detects different types of failure. To do so, we have employed an I/O tracing mechanism which completely traces the I/O requests during running workloads. Such mechanism works online and keeps the state of the I/O requests in different levels. In order to track the I/O requests, we employ the Linux comprehensive I/O tracing tools, namely *blktrace* and *blkparse* providing required details of the request without any performance overhead. In the post-process level, we

have employed a modified version of *btt* tool to extract additional information such as standard format of *timing information* of the I/O requests. Such modification helps us in detecting *complete* and *incomplete* I/O requests (we call a request as *complete* when we receive the ACK of the request). To do so, we have modified the operation of “-per-io-dump” option in *btt*. This option extracts the trace of an individual I/O request where such modification creates the trace of large size request that are divided into “sub-requests” with the detailed timing information.

Failure detection process works based on the collected information of the I/O requests in the header of the *data packets*. Note that we have two versions of *data packet*: 1) the generated one by the I/O Generator kept in database and 2) the written one in the disk subsystem. We start failure detection process in two cases: 1) when we receive a request time out response and 2) when we receive the ACK of a request (i.e., when the *complete* flag of the request is set. A request is set to *complete* when all its sub-requests are set as *complete*).

Algorithm 1 shows how we detect different types of data failures in the platform.⁸ First, in line 2 we check the timeout response of the request. In case of timeout, we report *I/O Error* in line 3. Then in line 5, we check the checksum of written data (by comparing *dataChecksum* and *finalChecksum*) to detect if data failure is occurred or not. In case of equality, we mark the data as *correct* and report “no failure”. Otherwise, in case of any inconsistency between *dataChecksum* and *finalChecksum*, in line 6, we compare the *finalChecksum* with *initialChecksum* of corresponding address (which was extracted before write operation and kept in the database) to validate the write operation. In case of equality, in line 7, we check whether multiple writes are submitted to the disk subsystem. To do so, we check if the sequence of the accesses is *Write After Write* (WAW) or not. If so, the failure is due to *Unserializability* of two parallel write accesses (line 8). Otherwise in line 11 we report *False Write Acknowledge* (FWA) which is due to the fact that data is not written in the SSD flash cells while we receive acknowledgment in application level. Then in line 16 we check the correctness of sub-requests and keep the number of failed sub-requests in variable *numOfFailedSubReqs*. In line 20, if we find failed sub-requests, we report *Shorn Write*. In line 23 in case of failure of all sub-requests, entire data is corrupted and we report *Full Data Corruption* (FDC). We report *Flying write* in line 26 in case of inconsistency of sub-addresses and the address of I/O request. To do so, we scan entire addresses of disk subsystem which takes long time.

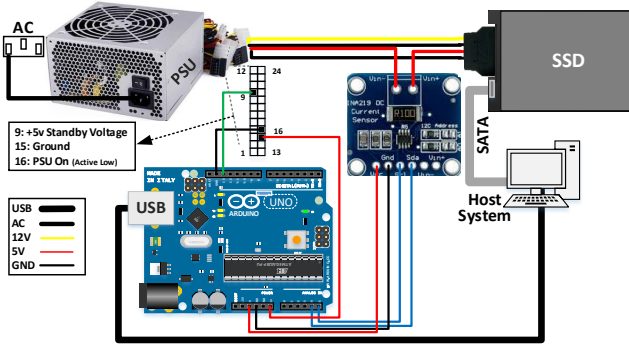
4.3 HardWare Module (HWM)

The structure of HWM in our proposed reliability test platform is shown in Fig. 5. HWM is responsible for injecting real and physical faults such as power outage and high temperature failures. HWM receives the fault injection commands from SWM through a USB connection. Fig. 5a shows the schematic of HWM that injects physical power outage faults to the SSDs. It can be seen that HWM is placed in the path of power lines of the SSDs to perform real fault injections. Fig. 5b shows the schematic of HWM which is responsible for high temperature fault injection. It can be seen that HWM manages the *heater* and the *fan* to control the temperature of the SSDs as assigned by SWM (*Scheduler*). The physical view of the proposed reliability test platform is provided in Fig. 5c and Fig. 5d.

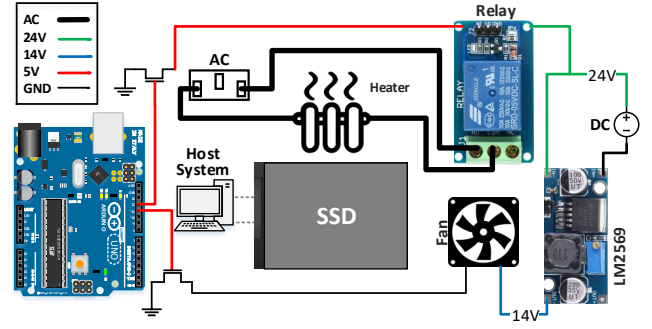
HWM includes the following parts:

- 1) Atmega32 micro-controller embedded in an Arduino UNO board. This micro-controller receives and decodes the commands from SWM. It turns the power of SSDs *on* or *off* based

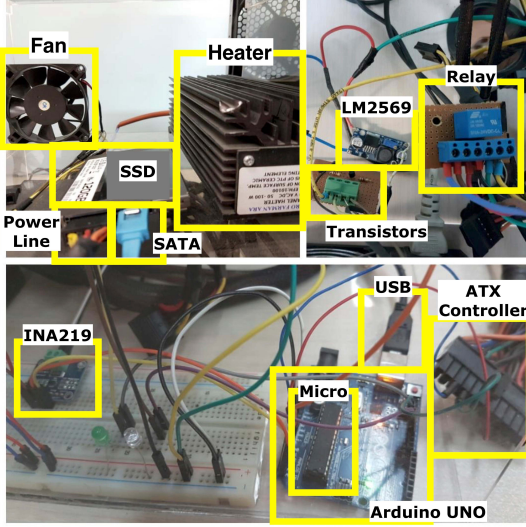
⁸ Detection of dead device failure (mentioned in Section 3.3) is mainly performed in the hardware layer and needs further tests in separated boards and is not performed within SWM.



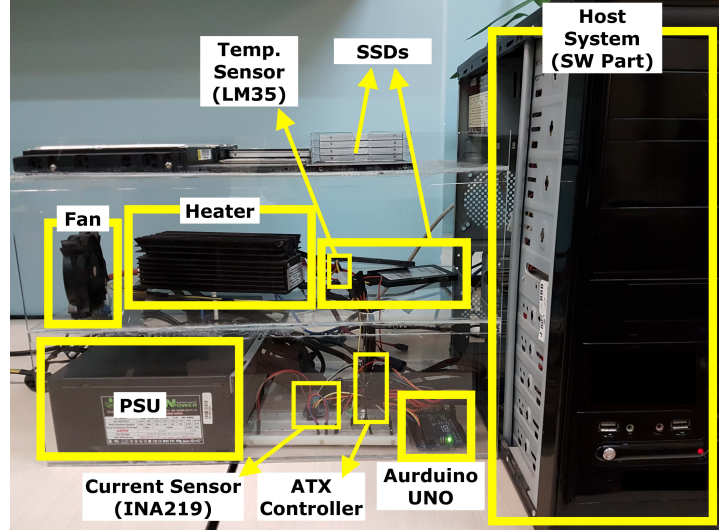
(a) Schematic of HWM for injecting power outage faults.



(b) Schematic of HWM for injecting high temperature faults.



(c) Proposed reliability test platform (close view).



(d) Proposed reliability test platform (overall).

Fig. 5: Proposed reliability test platform (HWM).

on the received command in determined time instances. In addition, the micro-controller is programmed to decode the commands related to temperature, fan, and measuring modules.

- 2) INA219 module is used for measuring the current and voltage of SSDs. This module measures and sends the current of the SSDs to SWM and makes the SWM able to detect the status and operation of the SSDs.
- 3) SRD24v relay which controls the power of the heater module in the platform. This relay is controlled by the micro-controller and turns the heater *on* or *off* in determined time instances which is assigned by the micro-controller based on the temperature of the SSDs.
- 4) LM35 sensor measures the temperature of the platform and the SSDs. The information about temperature of the SSDs are passed to SWM. SWM also receives the temperature of the SSDs from S.M.A.R.T report provided by the SSD manufactures.
- 5) The heater is responsible for injecting high temperature faults to the SSDs. The employed heater is able to increase the temperature up to 100 degrees Celsius in 5 minutes (in such condition the temperature of the SSDs increases up to 70 degrees Celsius).
- 6) The fan is used to cool the platform and decrease the temperature of the SSDs as determined by micro-controller during high temperature fault injection.

HWM communicates with SWM (*Host System*) through a USB serial connection. The embedded micro-controller in *Ar-*

duino UNO receives commands from SWM. The micro-controller switches the power of SSD to ON or OFF state by controlling the pin 16 of the ATX controller of the PSU which drives the under test SSDs power. Pin 16 of the ATX controller is active low and cuts off the output power of the PSU by applying a high voltage (+5V). To inject the high temperature faults, micro-controller manages the heater and fan to control the temperature of the SSDs as SWM decides.

The proposed reliability test platform injects the real physical fault. To inject real power outage fault, we model the real discharge delay of large size capacitors that are employed in the PSU. By conducting several experiments, we observed the impact of discharge delay of such capacitors on the input voltage of SSDs. As depicted in Fig. 6, we observed the output voltage of the PSU in different cases: 1) when the PSU drives no SSD (Fig. 6a) and 2) when the PSU drives two SSDs (Fig. 6b). The results of the experiments reveal that the full discharge (5v to 0) delay of the PSU takes about 1,900ms when it drives two SSDs. The SSDs become unavailable through the application layer (SWM in *Host System*) when the input voltage drops to 4.5v during 5ms after power fault injection.

5 EVALUATIONS AND OBSERVATIONS

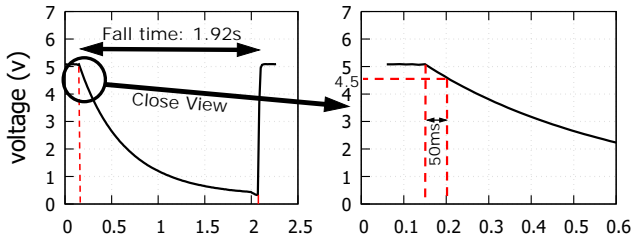
In this section, we evaluate the reliability of I/O cache including SSDs in RAID-1 configuration and show the ratio of different types of failures (described in Section 3) under power outage and high temperature. To do so, we have conducted experiments

Algorithm 1: Data failure detection algorithm

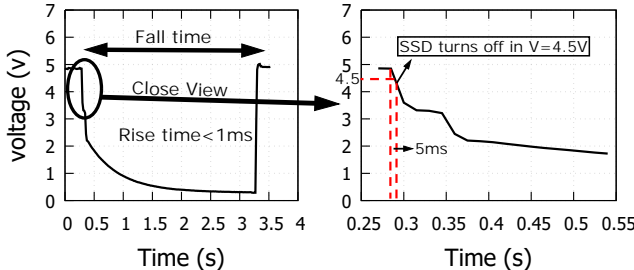
```

Input: initialChecksum, dataChecksum, finalChecksum, subChecksum,
numOfSubReqs, dstAddress
/* timeout is the default parameter defined in linux
kernel for I/O requests. */
1 Function failureDetection is
/* First we check if the timeout is occurred or not. In
case of timeout, IO Error is reported. */
2 if responseTime > timeout then
3   Report IO Error.
4 end
/* We check the correctness of checksum: */
5 if finalChecksum != dataChecksum then
/* In case of inequality, data failure is
detected. Then we find the type of data
failure. */
/* We check if data is written in SSD or not? */
6 if initialChecksum == finalChecksum then
/* In this case, we receive the ACK in
application level while the data is not
written in the SSD. */
/* If we have multiple writes from different
threads: */
7   if WAW then
8     Report Unserializability.
9   end
10  else
11    Report FWA.
12  end
13 end
/* We check the correctness of subrequests.
numOfFailedSubReqs shows the number of failed
subrequests. */
14 numOfFailedSubReqs = 0
15 for i = 0 to numOfSubReqs do
16   if subFinalChecksum != subDataChecksum then
17     numOfFailedSubReqs ++
18   end
19 end
/* If we find any failed subrequest */
20 if numOfFailedSubReqs > 0 then
21   Report Short Write.
22 end
/* If all sub-requests are corrupted we have FDC. */
23 else if numOfFailedSubReqs == numOfSubReqs then
24   Report FDC.
25 end
26 if numOfFailedSubReqs == 0 and subAddress does not fit in
dstAddress then
27   Report Flying Write.
28 end
29 end

```



(a) When the PSU does not drive any device.



(b) When the PSU drives two SSDs in RAID-1 configuration.

Fig. 6: The output voltage of the PSU.

using a realistic fault injection platform on more than ten enterprise SSDs from different vendors where the detailed technical information of the SSDs is provided in Table 2. Note that the SSDs from type C support “power loss data protection” and hence experience small number of failures compared to other SSD types.

The detailed information about *Host System* used as SWM in the test platform is reported in Table 3. To implement I/O cache, we use an open-source cache module, EnhanceIO, as a kernel module where the under test SSDs in RAID-1 configuration are used in the cache layer of HDD. The I/O cache is configured with the *Least Recently Used* (LRU) replacement policy and the block size is equal to 4KB. The total cache size is equal to 100 GB (20% of disk space) to create promotion and eviction operations on the cache. Note that in case of larger cache size, the cache has enough space to serve all accesses with minimum (near zero) miss ratio and eviction/promotion operation. In the following experiments, the write policy is set to *Write Back* (WB) while in Section 5.9, we configure the cache policy in three types of WB, *Write Through* (WT), and *Read Only* (RO) to compare the impact of cache write policy on the reliability. We implement RAID-1 configuration using *mdadm* as a software-based RAID management tool for Linux systems.

We perform experiments and examine the impact of workload dependent parameters on the failure rate of the SSD-based I/O cache (in RAID-1 configuration) in presence of power outage and high operating temperature. We also study the impact of 1) workload WSS, 2) type of requests (read/write), 3) request size, 4) requested *Input Output Operation per Second* (IOPS), 5) access pattern (random/sequential), and 6) sequence of the accesses (i.e., *Read After Read* (RAR), *Read After Write* (RAW), *Write After Read* (WAR), and *Write After Write* (WAR)). In each experiment, we commit at least 24,000 accesses and impose 600 faults into the SSDs. However, to investigate the impact of workload characteristics, we perform multiple experiments that increases the number of accesses and injected faults.⁹ In the following, we elaborate the experiments and report the impact of above-mentioned parameters on the failure rate and the ratio of different types of failures in presence of power outage and high temperature.

5.1 Impact of Workload Working Set Size (WSS)

In this section, we evaluate the reliability of I/O cache to examine the impact of workloads WSS on the ratio of different types of failures. To do so, we perform ten experiments by using the workloads with different WSSs. We change the workloads WSS from 1GB to 350GB and measure the failure ratio under power outage. In these experiments, the accesses are distributed in uniform random pattern and the requests size varies between 4KB and 1MB. We issue more than 96,000 writes to the disk subsystem where the SSDs experience more than 2,400 power failures during the experiments.

The results of the experiments are shown in Fig. 7 (data failure per power fault is shown in the right side axis). We make two major observations: 1) the failure rate in the I/O cache increases by 44% when we increase the WSS from 1GB to 350GB under power failures that represents fewer failures in the workloads with smaller WSS. This is due to the fact that the time intervals between updating cache blocks in workloads with smaller WSS is less than the workloads with larger WSS. In this case, the written data resides in the cache for a short time where the probability of power outage in a such short time is low. 2) We do not experience unserializable writes and flying writes failures during these experiments.

We conclude that the workload with large WSS experiences more data failure rate compared to the workloads with smaller

9. We mainly compare the number of failures per power fault that reveals no dependency between the number of fault injection and average number of data failures.

TABLE 2: Specification of employed SSDs in the experiments (DWPd: Driver Write Per Day, UBER: Unrecoverable Bit Error Rate, MTBF: Mean Time Between Failures, LDPC: Low Density Parity Check [10]).

SSD Type	Size	\$/GB	Bit per Cell	Release Year	Read/Write IOPS (4KB)	Sequential Read/Write (MB/s)	DWPd	Driver Life Time (TB)	UBER	MTBF (Million Hours)	Other Features
A	120 GB	0.5	TLC	2015	60/70 K	530/410	0.68	90	-	1.5	LDPC, SLC caching
B	120 GB	0.61	MLC	NA	11.5/52 K	420/120	2.75	354	-	1.0	-
C	2 TB	0.69	MLC	2015	95/28 K	510/485	3.6	12, 320	10^{-17}	2.0	Power loss data protection
D	2 TB	0.4	TLC	2016	93/24 K	540/520	0.9	3, 200	10^{-17}	3.0	-

TABLE 3: HW and SW specification of *Host System* used in reliability test platform.

Hardware	
Motherboard	Z97-A from ASUSTeK corp.
CPU	Intel(R) Core(TM) i5
RAM	8GB DDR3 from Hynix Semiconductor
HDD (OS)	7.2K RPM, 500GB from SEAGATE corp.
HDD (Disk Subsystem)	IntelliPower, 500GB from Western Digital
Under test SSDs	According to Table 2
Software	
OS	Ubuntu 17.04
Kernel	4.10.0-19-generic

WSS. In this case, the failures from shorn write type become dominant.

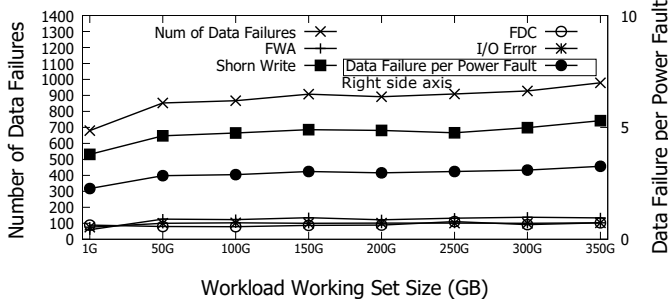


Fig. 7: Impact of workload working set size on different types of failure.

5.2 Impact of Request Type

In this section, we conduct experiments to study the impact of request types of the workloads on the ratio of different types of failures in the I/O cache. To this end, we have performed five experiments where we change the percentage of read operations in the workloads from 0 to 100% and measure the failure ratio under power outage. In these experiments, the WSS of the running workloads is set to 380GB and the accesses are distributed in uniform random pattern. The requests size varies between 4KB and 1MB. We issue more than 150,000 writes to the disk subsystem where the SSDs experience more than 3,000 power failures during the experiments.

The results of the experiments are shown in Fig. 8 (data failure per power fault is shown in the right side axis). We make four major observations: 1) the failure rate decreases by increasing the number of read operations in the workloads. 2) We observe failures only from FDC type in the 100% read workload due to the failure of write operations during promoting missed data blocks into the I/O cache. 3) We observe no unserializable write and flying write types of failure. The reason behind zero unserializable write failures is the workload type. We submit I/O requests with uniform random access pattern that the likelihood of submitting two consecutive write accesses in an identical address is low, and hence we do not observe any unserializable write failures. 4) The shorn write failures are the dominant type in these experiments.

We conclude that in the workloads with low number of read operations, the I/O cache experiences more data failures. In addition, in the workloads with large number of read operations,

sudden power outage corrupts the data blocks of I/O cache (due to the corruption of write operations during promoting data blocks).

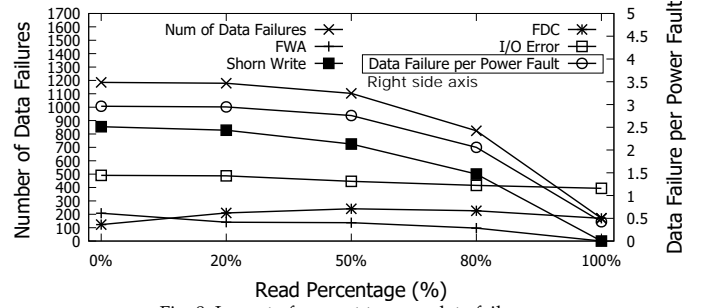


Fig. 8: Impact of request type on data failures.

5.3 Impact of Request Size

In this section, we evaluate the impact of request size of the workloads on the ratio of different types of failures. To this end, we have performed five experiments where the request size of the workloads is fixed to 4KB, 16KB, 64KB, 256KB, and 1MB in each experiment, and then we measure the failure ratio under power outage. In these experiments, accesses are distributed in uniform random pattern and the requests size varies between 4KB and 1MB. We issue more than 225,000 writes to the disk subsystem where the SSDs experience more than 4,500 power failures during the experiments.

Fig. 9 shows the results of these experiments (data failure per power fault is shown in the right side axis). We make three major observations: 1) the failure rate decreases by increasing the request size of the workloads. 2) In the workloads with smaller request size we do not observe shorn write while we experience large number of FWA failures (we will elaborate this observation in the description of Fig. 10). 3) We observe no failure from flying write and unserializable write types.

Fig. 10 shows the percentage of FWA, shorn write, and FDC failures based on the workloads request size. We observe that in the workloads with average request size less than 480KB, the FWA failure is dominant while in the workloads with larger request size, the I/O cache experiences mainly shorn writes. This is because to the fact that in the workloads with small writes, large number of write pending request is buffered in the SSD volatile elements within internal data path such as host FIFO buffer and DRAM buffer [26, 28, 29, 38, 39] and hence, in this case, power outage causes more FWA failures compared to the other type of failures. On the other hand, in case of submitting large size requests (larger than 500KB), the SSDs in the cache layer experience higher number of shorn writes than FWA. This observation also verifies the results of experiments presented in Fig. 8. This is because shorn writes mainly occur due to an interrupt (here power outage) during write operation. For large size requests that the request is partitioned into multiple sub-requests, in case of power outage, the committed sub-requests before power outage will be completed while the operations during power outage are failed leading to shorn write failure. In contrast, small size requests finish and receive acknowledgment in a short time and the likelihood of power outage during small

size write operation is much less than large size accesses. Thus, the number of shorn write failure in the workloads with large size requests is higher than the workloads with small size write accesses. In addition, in the workloads with larger request sizes, we observe lower failures compared to workloads with smaller request size. Finally, we observe increased failures from FDC type in the large size request.

We conclude that the workloads with small random write accesses experience large number of data failures under power outage compared to the workloads with large request sizes.

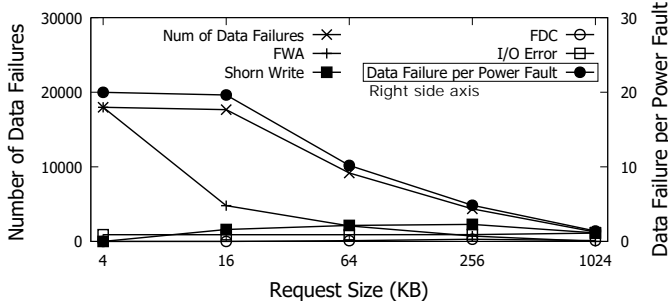


Fig. 9: Impact of request size on data failure.

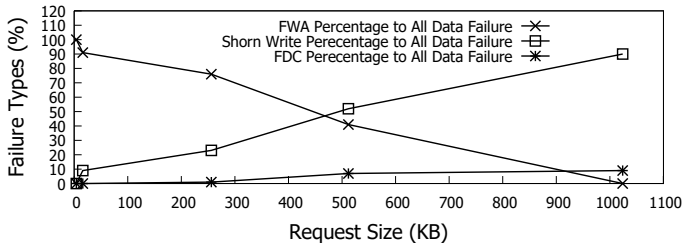


Fig. 10: Impact of request size on data failure.

5.4 Impact of Requests Access Pattern (Random/Sequential)

Here, we study the impact of workload access pattern on the ratio of different types of failures in the I/O cache configuration. To this end, we perform experiments with three different workloads including random and sequential access patterns (including write accesses with average request size equal to 512KB) and one real storage benchmark, Cello99 (14-Jan), mainly including 8KB partially sequential read and write accesses. Then we measure the failure ratio under power outage. We commit more than 26,000 write requests to the disk subsystem and inject more than 200 sudden power failures to the SSDs.

Fig. 11 shows the results of the experiments (data failure per power fault is shown in the right side axis). We make four major observations: 1) in the workloads with sequential access pattern, we experience only 2% more data failure compared to the workloads with random accesses. 2) In all workloads, we observe larger number of FWA failures than shorn write failures. 3) We observe no unserializable write and flying write types of failure. 4) We observe higher range of failures (especially FWA) for Cello99 workload, which is mainly due to smaller requests size in this workload (8KB) compared to random and sequential workloads (512KB). This observation also verifies the results of experiments presented in Fig. 9 and Fig. 10.

We conclude that the access pattern of the workloads (sequential or random) has no significant impact on the failure rate of the I/O cache. In contrast, we observe a considerable impact of requests size on the failure ratio.

5.5 Impact of IOPS

In this section, we evaluate the impact of requested *Input/Output Per Second* (IOPS) (i.e., the number of operations that are submitted and responded by the SSD in one second) issued by the

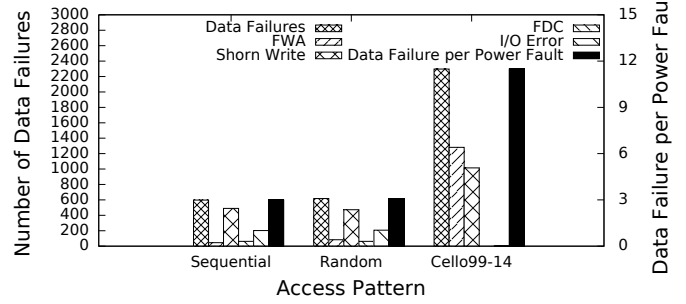


Fig. 11: Impact of access pattern on data failure (examining both synthetic and real storage workload)

workload on the ratio of different types of failures in the I/O cache (in RAID-1 configuration). To do so, we have performed five experiments where the requested IOPS of the workloads is fixed to 1.2K, 2.4K, 6K, 12K, and 20K in each experiment, then we measure the failure ratio under power outage. The WSS of the experiments is set to 380GB and the requests size varies between 4KB and 1MB which are distributed in a uniform random pattern. In these experiments, we commit more than 120,000 writes and inject more than 3,000 power failures to the SSDs.

Fig. 12 shows the results of these experiments (data failure per power fault is shown in the right side axis). We make two main observations: 1) both responded IOPS from RAID configuration and failure rate are saturated when we increase the requested IOPS to 6K or greater. 2) The shorn write failure is dominant type of failure.

We conclude that in the workloads with high I/O load when the responded IOPS saturates the failure ratio saturates, respectively.

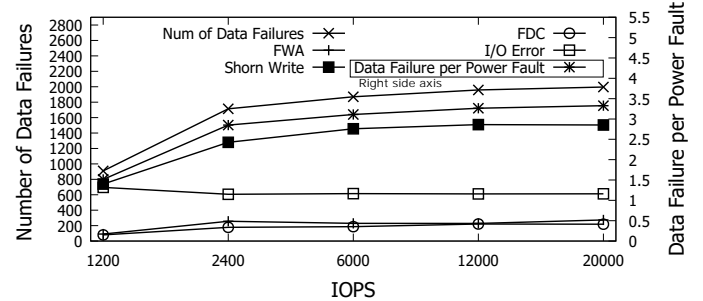


Fig. 12: Impact of requested IOPS submitted to the SSD on number of data failures.

5.6 Impact of Sequence of the Accesses

In this section, we conduct experiments to evaluate the impact of the workloads sequence of accesses on the reliability of I/O cache. To this end, we have performed experiments under power failures with different workloads where each workload mainly includes 1) RAW, 2) WAR, 3) RAR, and 4) WAW accesses. The requests size is between 4KB and 1MB which the accesses are distributed in a uniform random pattern. In these experiments, 144,000 write requests are committed to the subsystem where the SSDs experience 3,600 sudden power outage.

Fig. 13 shows the results of these experiments (data failure per power fault is shown in the right side axis). We make four major observations: 1) unserializable write failure occurs frequently in the workloads with WAW accesses. 2) We observe large number of shorn write failures in the workloads with WAR and WAW accesses. 3) The workloads with RAR accesses experience both I/O errors and data failures from FDC type under power outage. 4) FWA failure occurs in the workloads with RAW, WAR, and WAW accesses where the minimum and maximum number of such failure is observed in the workloads with WAR and WAW accesses, respectively.

We conclude that the unserializable write failures mainly occur in the workloads with large number of WAW accesses under

power outage. In the workloads with RAR accesses, although there are no write requests from the application level, but we experience data failures due to the corruption of write operations during promoting data to the I/O cache. The shorn write failure occurs in all types of workloads while in the workloads with WAR and WAW accesses such failure becomes dominant.

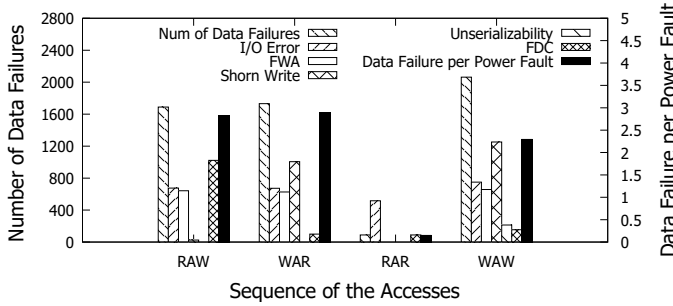


Fig. 13: Impact of sequence of the accesses on data failure.

5.7 Impact of Disks Order in RAID-1 Configuration

In this section, we evaluate the impact of using SSDs from different vendors in different orders in RAID-1, used in I/O cache configuration, on the ratio of failures. In a mirrored (i.e., RAID-1) configuration with two disks, the first disk is called “primary” and the second is called “secondary” [15–21]. To provide *data consistency* in such subsystem, the controller periodically compares the written data in both disks. In case of any inconsistency between primary and secondary disks, the primary one is determined to store the valid data, and hence, the secondary disk will be updated with the primary disk’s data. Furthermore, in RAID-1 configuration, write operations are committed to both primary and secondary disks, while read operations are *only* supplied by the primary one. A read operation is provided by the secondary disk when the primary one is busy due to supplying previous requests [15–21]. Considering two above-mentioned RAID-1 properties, using disks with different levels of reliability (i.e., from various vendors) as primary disk in mirrored configuration can highly affect the failure ratio. To do so, we perform experiments under power failures with different RAID-1 configurations. In these experiments, we employ SSDs from various vendors providing different count of written logical blocks (i.e., different aging level) and different levels of reliability as either primary or secondary disks.

In the experiments, we commit more than 48,000 writes to the subsystem while the SSDs experience more than 1,200 power failures. Fig. 14 shows the results of these experiments (number of data failures per power fault is shown in the right side axis). We observe that when we use a low reliable SSD (e.g., Type-A and Type-B) as the primary disk, the failure ratio increases by 52% (this is due to the fact that RAID-1 cannot mask total failures in the SSDs). In contrast, using SSDs from Type-D or Type-C as primary disk provides higher range of reliability by masking data failures occurred in the secondary disk that provides lower level of reliability. We conclude that the ratio of data failure is significantly affected by the order of disks in I/O caches in RAID-1 configuration where employing a low reliable SSD as the primary disk can increase the data failure about 52%.

5.8 Impact of High Temperature Faults

In this section, we evaluate the reliability of I/O cache under high temperature failures. To this end, we perform experiments where the SSDs experience high temperature without any power failure. In the experiments, we commit more than 65,000 random request to the SSDs. We increase the temperature of the SSDs up to 61 degrees Celsius (measured by SSDs S.M.A.R.T) and 64 degrees Celsius (measured by temperature sensor in the test platform).

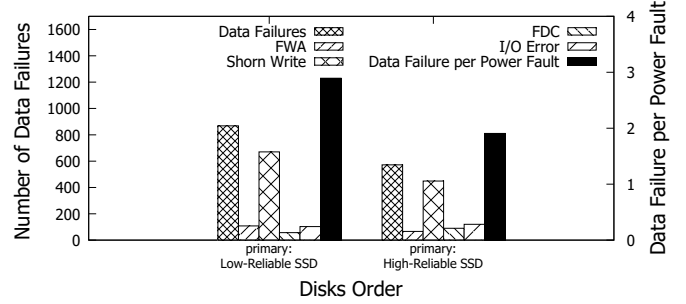


Fig. 14: Impact of disks order in RAID-1 Configuration.

The temperature of the SSDs in the experiments does not exceed the reported value in the datasheet.

Fig. 15 shows the measured temperature of the SSDs during experiments. In this figure, we show the temperature of the SSDs in two cases: 1) during running the workload without any additional temperature faults and 2) during running the workload with injecting high temperature faults. It can be seen that, when SSDs supply the requests of the workload (described previously) the average temperature is about 31 degrees Celsius.

In the experiments, we observe no data failure in the SSDs by increasing the temperature of the SSDs up to 64 degrees Celsius. This is due to the fact that although high operating temperature increases the retention speed and failure rate, but existing enterprise SSDs reduce the number of accesses to the underlying SSDs (using throttling technique) resulting in reduced failure rate due to high temperature [26, 39].

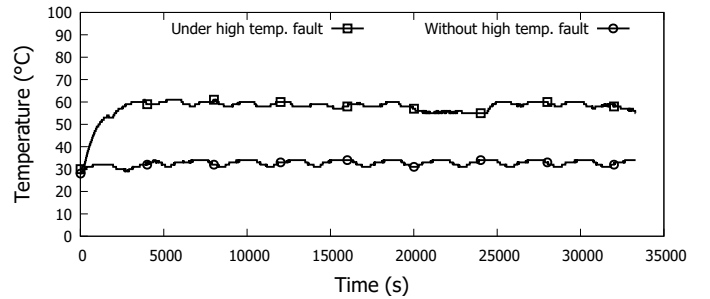


Fig. 15: Disks temperature during experiments.

5.9 Impact of I/O Cache Write Policy

In this section, we evaluate the reliability of I/O cache with different write policies under power outage. To this end, we conducted four experiments in which the cache write policy is set to 1) *Write Back* (WB), 2) *Write Through* (WT), and 3) *Read Only* (RO). We examine the RO cache with two types of workloads including 100% write (i.e., 0% read) and 50% read. The WSS of the workloads in all experiments is equal to 380GB. We commit more than 96,000 requests to the disk subsystem where the size of accesses is between 4KB and 1MB with a uniform random access pattern. In each experiment, the SSDs experience more than 2,400 sudden power failures.

Fig. 16 shows the results of the experiments which reveals four major observations: 1) failure rate in the WB cache is more than other cache policies by 20% (up to 32X). This is due to the fact that WB cache buffers all requests (read and write) where the write requests only reside in the cache until they become evicted. In a WB cache, if a data failure occurs on a data within the cache, such failed data will be evicted to the disk subsystem. Such condition affects the written data and also further read accesses which are responded from both cache and disk subsystem (i.e., hit or miss accesses). 2) Although WT cache keeps two copies of data in both cache and disk, but data failure occurs in such configuration which cannot be recovered by the WT cache. In a WT cache, write requests first are supplied by disk subsystem

and ACK is sent to the application, then the data is written to the cache for supplying future requests. In this case, if the written data in the cache fails, further read requests supplied from the cache will fail. 3) WT cache experiences smaller failure rate compared to WB cache. The reason is that in the WT cache, there is no eviction from cache to the disk subsystem and hence, no corrupted data will be evicted to the disk. On the other hand, since WB cache keeps dirty blocks and evicts them to the disk subsystem, failed data in the cache is propagated to the disk subsystem. 4) Although RO cache does not supply write requests, data failure occurs in the RO cache. In the experiment which we run the 0% read workload on a RO cache, no read and write requests are supplied by cache (i.e., there is no accesses to the SSDs in the cache since all write requests are directed to the disk subsystem where there is no further read accesses to them). While in the second experiment with 50% read workload, the RO cache directs writes to the disk subsystem while the read misses are buffered in the cache (i.e., are written in the cache). Sudden power outage during promoting the data to the cache may disrupt the write operation which leads to data failure. In this case, further read accesses which hit in the cache will fail.

We conclude that the I/O cache in WB, WT, and RO policies experiences data failure where the failure rate in the WB cache is more than others (by 20%). Furthermore, if power failure occurs during promoting data to the cache, further read accesses will fail.

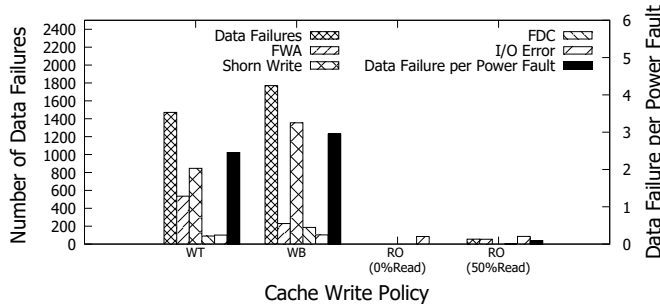
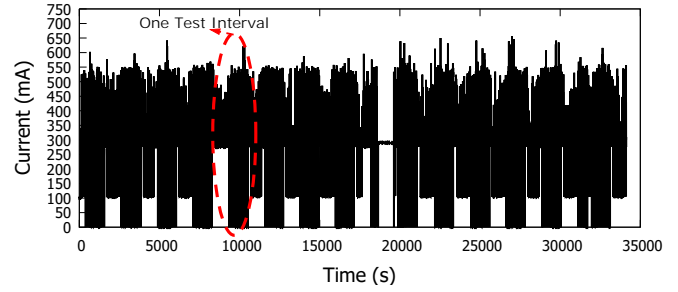


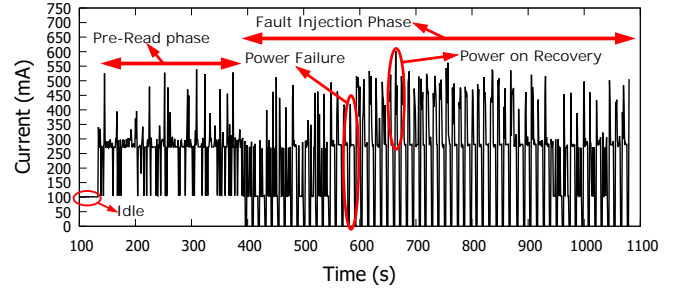
Fig. 16: Impact of cache write policy on the failure rate.

5.10 Measuring the Current of SSDs

In this section, the current of the SSDs is measured during 10 hours of experiments using INA219 module. In this experiment, we commit more than 60,000 write requests to the disk subsystem with the SSD-based I/O cache. Fig. 17 shows the current of the SSDs in two cases: 1) during 10 hours (as depicted in Fig. 17a) and 2) only the first hour (as depicted in Fig. 17b). Using this report, we find the state of the disk and check the operations performing on the disk (idle, on, off, read operation, write operation, power on recovery, and fault injection instances). Then we decide when to inject the fault or when to start the failure detection algorithm. As shown in Fig. 17b, in the first 2-minutes, SSDs are idle and the current is less than 100mA. Then from $t = 120s$ to $t = 400s$, we have a reading phase (before each I/O request) where we read all written data in the addresses that we will rewrite to calculate the checksum of previously written data into the SSDs. At $t = 400s$, when the reading phase finishes, we start the fault injection interval. In each power outage, the current of the SSDs is equal to zero. After each power outage, we observe a power on recovery interval (as an internal operation of the SSDs), in which we wait for the disk to become available through the application layer.



(a) Disks current during 10 hours of experiments (including 17 test intervals).



(b) Disks current during one test interval (indicating read, write, and power outage phases).

Fig. 17: Disks current during experiments.

5.11 Failure Characterization of I/O Caches in Enterprise Storage Systems

In this section, we gather the results of previous experiments to indicate the frequency and workload dependency¹⁰ of different types of failures (discussed in Fig. 3) as depicted in Fig. 18 and Table 4. The main points of Fig. 18 and Table 4 are as follows:

- 1) As shown in Fig. 7, we observe numerous types of SSD failures but *none* of them are affected by varying workload WSS. In contrast, FWA, FDC, and shorn write failures are highly affected by request type.
- 2) In the workloads with different rates of read accesses, the ratio of FWA varies more than 100%. We observe similar behavior for FDC and shorn writes where the range of these failures varies respectively by 100% and more than 100% under different ratios of read accesses (Fig. 8).
- 3) FWA and shorn writes are highly related to request size where they vary more than 100% under different request sizes (Fig. 9 and Fig. 10), while request size has *no* impact on other types of failures.
- 4) FWA is partially affected by workload access pattern (less than 5%). Similarly, we observe only a range of 5% and 20% variation for FDC and shorn write failures under different access patterns.
- 5) FWA and FDC are highly related to requested IOPS of the running workload (by 100%), while shorn write only varies by 75% under different requested IOPS. We also observe other types of failures while they are not dependent to the requested IOPS of the workload.
- 6) FWA, shorn write, and unserializable write failures are highly related to the sequence of the accesses. We observe significant (more than 100%) variation in the ratio of unserializable writes under different sequences of accesses, where such failure only occurs in case of WAW accesses. Similarly, FDC is also highly related to sequence of accesses.
- 7) We do *not* observe any relation between I/O error and dead device failures with workload characteristics.

¹⁰ Workload dependency represents how different types of SSD failures are affected by different workload characteristics.

- 8) In our experiments, we do *not* observe flying write failure on the SSDs. However, we consider a small percentage of workload dependency for this type of failure compared to I/O error and dead device. This is because, flying writes mainly occur due to wrong destination address which seems to be related to workload characteristics.

According to the above-mentioned observations, we sort the dependency of different failures to the workload characteristics (Fig. 18), where FWA and shorn writes are determined as the most workload dependent failures. Unserializable writes and FDC are respectively less workload dependent failures compared to two previous types. We determine the flying writes, I/O error, and dead device as the failures with *zero* workload dependency.

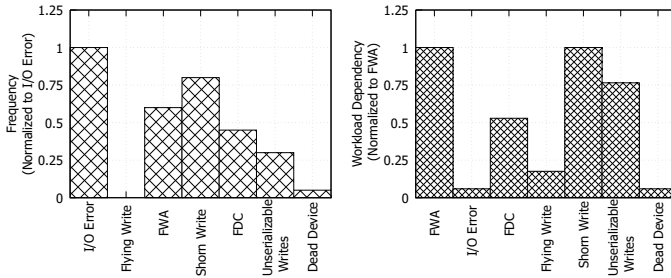


Fig. 18: Frequency and importance of different types of failures.

TABLE 4: Impact of workload characteristics on different types of SSD failures observed in our experiments.

Section	FWA	I/O Error	FDC	Shorn Write	Flying Write	Unserial Write	Dead Device
Sec. 5.1	-	-	-	-	-	-	-
Sec. 5.2	> 100%	-	100%	> 100%	-	-	-
Sec. 5.3	> 100%	-	-	> 100%	-	-	-
Sec. 5.4	< 5%	-	< 5%	< 20%	-	-	-
Sec. 5.5	100%	-	< 100%	> 75%	-	-	-
Sec. 5.6	> 100%	-	> 100%	> 100%	-	>> 100%	-
Workload Dep. (1: Low, 4: High)	4	1	2	4	1	3	1

6 CONCLUSION

In this paper, we evaluated the reliability of SSD-based I/O cache architectures in enterprise storage systems. To do so, we developed a Hardware-Software based reliability test platform for the SSDs that injects the physical failures such as power outage and high temperature faults that may occur commonly in large-size datacenters. The proposed test platform measures current, temperature, and power consumption of the SSDs and detects various types of failure that may occur in the SSDs and RAID configuration. We recognized different types of failures namely: *False Write Acknowledge* (FWA), *unserializable writes*, *Full Data Corruption* (FDC), *shorn writes*, *flying writes*, *I/O error*, and *dead device* and measured the failure ratio in different conditions. We evaluated the impact of workload dependent parameters such as workload *Working Set Size* (WSS), request size, request type, access pattern, requested I/O load, and sequence of the accesses on the reliability of SSD-based I/O caches in RAID-1 configuration. We conducted extensive experiments with various enterprise SSDs from top ten enterprise vendors and observed that the failure ratio in SSD-based I/O cache architecture under power outage is highly related to the I/O parameters of the workload such as requests size and WSS of the requests while other parameters such as access patterns have no impact on the failure rate. We observed no data failure in the SSDs upon high temperature faults. Furthermore, we observed that despite the high reliability of RAID-1 configuration, it fails in case of data inconsistency between disks which frequently happens upon power outage in data centers.

ACKNOWLEDGEMENT

This work has been partially supported by *Iran National Science Foundation* (INSF) under grant number 9606071 and by HPDS Corp.

REFERENCES

- [1] Dell EMC Corp., "EMC UNITY: FAST TECHNOLOGY OVERVIEW," White paper, Accessed: Jan. 2019.
- [2] NetApp, "SSD Cache Feature," <https://library.netapp.com>, Accessed: Jan. 2019.
- [3] HP, "HPE Smart Array SR SmartCache," <https://h20195.www2.hp.com>, Accessed: Jan. 2019.
- [4] S. Ahmadian, O. Mutlu, and H. Asadi, "ECI-Cache: A High-Endurance and Cost-Efficient I/O Caching Scheme for Virtualized Platforms," *Proceedings of the ACM on Measurement and Analysis of Computing Systems (POMACS)*, vol. 2, no. 1, p. 9, 2018.
- [5] R. Salkhordeh, S. Ebrahimi, and H. Asadi, "ReCA: An Efficient Reconfigurable Cache Architecture for Storage Systems with Online Workload Characterization," *IEEE Transactions on Parallel & Distributed Systems (TPDS)*, no. 7, pp. 1605–1620, 2018.
- [6] S. Ahmadian, R. Salkhordeh, and H. Asadi, "LBICA: A Load Balancer for I/O Cache Architectures," in *to appear in Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2019.
- [7] R. Micheloni, A. Marelli, and K. Eshghi, *Inside Solid State Drives (SSDs)*. Springer Science & Business Media, 2012.
- [8] R. Salkhordeh, H. Asadi, and S. Ebrahimi, "Operating System Level Data Tiering Using Online Workload Characterization," *The Journal of Supercomputing*, vol. 71, no. 4, pp. 1534–1562, 2015.
- [9] Y. Li, B. Shen, Y. Pan, Y. Xu, Z. Li, and J. C. Lui, "Workload-Aware Elastic Striping With Hot Data Identification for SSD RAID Arrays," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 36, no. 5, pp. 815–828, 2017.
- [10] L. Zuolo, C. Zambelli, A. Marelli, R. Micheloni, and P. Olivo, "LDPC Soft Decoding with Improved Performance in 1X-2X MLC and TLC NAND Flash-Based Solid State Drives," *IEEE Transactions on Emerging Topics in Computing (TETC)*, pp. 1–1, 2018.
- [11] Leventhal, Adam, "Flash Storage Memory," *Communications of the ACM*, 2008.
- [12] H.-W. Tseng, L. Grupp, and S. Swanson, "Understanding the Impact of Power Loss on Flash Memory," in *Design Automation Conference (DAC)*. ACM, 2011, pp. 35–40.
- [13] M. Zheng, J. Tucek, F. Qin, and M. Lillibridge, "Understanding the Robustness of SSDs under Power Fault," in *File and Storage Technologies (FAST)*, 2013, pp. 271–284.
- [14] S. Ahmadian, F. Taheri, M. Lotfi, M. Karimi, and H. Asadi, "Investigating Power Outage Effects on Reliability of Solid-State Drives," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 207–212.
- [15] Chen, Peter M and Lee, Edward K and Gibson, Garth A and Katz, Randy H and Patterson, David A, "RAID: High-Performance, Reliable Secondary Storage," *ACM Computing Surveys (CSUR)*, 1994.
- [16] A. Thomasian and C. Liu, "Performance Comparison of Mirrored Disk Scheduling Methods with a Shared Non-Volatile Cache," *Distributed and Parallel Databases*, vol. 18, no. 3, pp. 253–281, 2005.
- [17] A. Thomasian, "Multilevel RAID Disk Arrays," in *Proc. of the 23rd IEEE/14th NASA Goddard Conf. on Mass Storage Systems and Technologies*. Citeseer, 2006.
- [18] A. Thomasian and M. Blaum, "Higher Reliability Redundant Disk Arrays: Organization, Operation, and Coding," *ACM Transactions on Storage (TOS)*, vol. 5, no. 3, p. 7, 2009.
- [19] A. Thomasian, "Mirrored Disk Rouing and Scheduling," *Cluster Computing*, vol. 9, no. 4, pp. 475–484, 2006.
- [20] A. Thomasian and Y. Tang, "Performance, Reliability, and Performability of a Hybrid RAID Array and a Comparison with Traditional RAID1 Arrays," *Cluster Computing*, vol. 15, no. 3, pp. 239–253, 2012.

- [21] C. Weddle, M. Oldham, J. Qian, A.-I. A. Wang, P. Reiher, and G. Kuenning, "PARAID: A Gear-Shifting Power-Aware RAID," *ACM Transactions on Storage (TOS)*, vol. 3, no. 3, p. 13, 2007.
- [22] Y. Cai, S. Ghose, Y. Luo, K. Mai, O. Mutlu, and E. F. Haratsch, "Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques," in *High Performance Computer Architecture (HPCA)*, IEEE, 2017, pp. 49–60.
- [23] Dell, "Dell PowerEdge RAID Controller Guide: Consistency Checks," <http://www.dell.com/support/manuals>, Accessed: Jan. 2019.
- [24] A. Grossi, C. Zambelli, P. Olivo, P. Pellati, M. Ramponi, C. Wenger, J. Alvarez-Herault, and K. Mackay, "An Automated Test Equipment for Characterization of Emerging MRAM and RRAM Arrays," *IEEE Transactions on Emerging Topics in Computing (TETC)*, vol. 6, no. 2, pp. 269–277, 2018.
- [25] P. Pouyan, E. Amat, and A. Rubio, "Memristive Crossbar Memory Lifetime Evaluation and Reconfiguration Strategies," *IEEE Transactions on Emerging Topics in Computing (TETC)*, vol. 6, no. 2, pp. 207–218, 2018.
- [26] J. Meza, Q. Wu, S. Kumar, and O. Mutlu, "A Large-Scale Study of Flash Memory Failures in the Field," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 1. ACM, 2015, pp. 177–190.
- [27] Y. Cai, Y. Luo, E. F. Haratsch, K. Mai, and O. Mutlu, "Data Retention in MLC NAND Flash Memory: Characterization," *High Performance Computer Architecture (HPCA)*, pp. 551–563, 2015.
- [28] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid-State Drives," *Proceedings of the IEEE*, vol. 105, pp. 1666–1704, 2017.
- [29] —, "Errors in Flash-Memory-Based Solid-State Drives: Analysis, Mitigation, and Recovery," *arXiv preprint arXiv:1711.11427*, 2017.
- [30] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, 2012, pp. 521–526.
- [31] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Crista, O. S. Unsal, and K. Mai, "Error Analysis and Retention-Aware Error Management for NAND Flash Memory," *Intel Technology Journal*, vol. 17, no. 1, 2013.
- [32] S. Boboila and P. Desnoyers, "Write Endurance in Flash Drives: Measurements and Analysis," in *File and Storage Technologies (FAST)*, 2010, pp. 115–128.
- [33] I. Narayanan, D. Wang, M. Jeon, B. Sharma, L. Caulfield, A. Sivasubramaniam, B. Cutler, J. Liu, B. Khessib, and K. Vaid, "SSD Failures in Datacenters: What? When? and Why?" in *Proceedings of the 9th ACM International on Systems and Storage Conference (SYSTOR)*. ACM, 2016, p. 7.
- [34] 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 *Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2009, pp. 24–33.
- [35] B. Schroeder, R. Lagisetty, and A. Merchant, "Flash Reliability in Production: The Expected and the Unexpected," in *File and Storage Technologies (FAST)*, 2016.
- [36] Y.-M. Chang, Y.-H. Chang, T.-W. Kuo, Y.-C. Li, and H.-P. Li, "Disturbance Relaxation for 3D Flash Memory," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1467–1483, 2016.
- [37] S.-K. Kim, J. Choi, D. Lee, S. H. Noh, and S. L. Min, "Virtual Framework for Testing the Reliability of System Software on Embedded Systems," in *Proceedings of the 2007 ACM symposium on Applied computing (SAC)*. ACM, 2007, pp. 1192–1196.
- [38] D. Rollins, Micron Technology Inc., "A Comparison of Client and Enterprise SSD Data Path Protection," <https://www.micron.com>, Accessed: Jan. 2019.
- [39] Y. Luo, "Architectural Techniques for Improving NAND Flash Memory Reliability," Ph.D. dissertation, Seagate Technology, 2018.
- [40] Intel Corp., "AHCI Specification for Serial ATA," [https://www.intel.com/content/www/us/en/io/serial-](https://www.intel.com/content/www/us/en/io/serial-ata/ahci.html)

[ata/ahci.html](https://www.intel.com/content/www/us/en/io/serial-ata/ahci.html), Accessed: Jan. 2019.



Saba Ahmadian received the B.Sc. and M.Sc. degrees in computer engineering from *Sharif University of Technology (SUT)*, Tehran, Iran, in 2013 and 2015, respectively. From 2011 to 2012, she was a member of *Energy Aware Systems (EASY)* Lab, SUT, where she researched on power reduction techniques on embedded CPUs. From 2012 to 2015, she was a member of *Embedded Systems Research (ESR)* Lab, SUT, where she researched on low power and reliability-aware techniques on Automata-based embedded systems. Currently, she is a Ph.D. candidate at *Data Storage, Networks, and Processing (DSN)* Lab at SUT under supervision of Dr. Hossein Asadi. Her research interests include storage systems design, virtualization platforms, fault tolerant design, and low power systems design.



Farhd Taheri received the B.S. degree in computer engineering from *Shahid Bahonar University*, Kerman, Iran, in 2016 and M.Sc. degree at DSN Lab in SUT under supervision of Dr. Hossein Asadi. Currently, he is a Ph.D. student at *Smart and Secure Systems (3S)* Laboratory at SUT under supervision of Dr. S. Bayat-Sarmadi. His research interests include Hardware security, Side Channel Attack, Solid-State Drives and fault tolerant design.



Hossein Asadi (M'08, SM'14) received the B.Sc. and M.Sc. degrees in computer engineering from the SUT, Tehran, Iran, in 2000 and 2002, respectively, and the Ph.D. degree in electrical and computer engineering from *Northeastern University*, Boston, MA, USA, in 2007.

He was with EMC Corporation, Hopkinton, MA, USA, as a Research Scientist and Senior Hardware Engineer, from 2006 to 2009. From 2002 to 2003, he was a member of the Dependable Systems Laboratory, SUT, where he researched hardware verification techniques. From 2001 to 2002, he was a member of the Sharif Rescue Robots Group. He has been with the Department of Computer Engineering, SUT, since 2009, where he is currently a tenured Associate Professor. He is the Founder and Director of the DSN Laboratory, Director of Sharif *High-Performance Computing (HPC)* Center, the Director of Sharif *Information and Communications Technology Center (ICTC)*, and the President of Sharif ICT Innovation Center. He spent three months in the summer 2015 as a Visiting Professor at the School of Computer and Communication Sciences at the *Ecole Polytechnique Fédérale de Lausanne (EPFL)*. He is also the co-founder of HPDS corp., designing and fabricating midrange and high-end data storage systems. He has authored and co-authored more than eighty technical papers in reputed journals and conference proceedings. His current research interests include data storage systems and networks, solid-state drives, operating system support for I/O and memory management, and reconfigurable and dependable computing.

Dr. Asadi was a recipient of the Technical Award for the Best Robot Design from the International RoboCup Rescue Competition, organized by AAAI and RoboCup, a recipient of Best Paper Award at the 15th CSI International Symposium on *Computer Architecture & Digital Systems (CADS)*, the Distinguished Lecturer Award from SUT in 2010, the Distinguished Researcher Award and the Distinguished Research Institute Award from SUT in 2016, and the Distinguished Technology Award from SUT in 2017. He is also recipient of Extraordinary Ability in Science visa from US Citizenship and Immigration Services in 2008. He has also served as the publication chair of several national and international conferences including CNDS2013, AISP2013, and CSSE2013 during the past four years. Most recently, he has served as a Guest Editor of *IEEE Transactions on Computers*, an Associate Editor of *Microelectronics Reliability*, a Program Co-Chair of CADS2015, and the Program Chair of CSI National Computer Conference (CSICC2017).