Lessons and Actions: What We Learned from 10K SSD-Related Storage System Failures

Erci Xu, *Ohio State University;* Mai Zheng, *Iowa State University;* Feng Qin, *Ohio State University;* Yikang Xu and Jiesheng Wu, *Alibaba Group*

https://www.usenix.org/conference/atc19/presentation/xu

This paper is included in the Proceedings of the 2019 USENIX Annual Technical Conference.

July 10–12, 2019 • Renton, WA, USA

ISBN 978-1-939133-03-8

Lessons and Actions: What We Learned from 10K SSD-Related Storage System Failures

Erci Xu
Ohio State University

Mai Zheng
Iowa State University

Feng Qin
Ohio State University

Yikang Xu
Alibaba Group

Jiesheng Wu
Alibaba Group

Abstract

Modern datacenters increasingly use flash-based solid state drives (SSDs) for high performance and low energy cost. However, SSD introduces more complex failure modes compared to traditional hard disk. While great efforts have been made to understand the reliability of SSD itself, it remains unclear what types of system level failures are related to SSD, what are the root causes, and how the rest of the system interacts with SSD and contributes to failures. Answering these questions can help practitioners build and maintain highly reliable SSD-based storage systems.

In this paper, we study the reliability of SSD-based storage systems deployed in Alibaba Cloud, which cover near half a million SSDs and span over three years of usage under representative cloud services. We take a holistic view to analyze both device errors and system failures to better understand the potential casual relations. Particularly, we focus on failures that are **R**eported **A**s "**S**SD-**R**elated" (**RASR**) by system status monitoring daemons. Through log analysis, field studies, and validation experiments, we identify the characteristics of RASR failures in terms of their distribution, symptoms, and correlations. Moreover, we derive a number of major lessons and a set of effective methods to address the issues observed. We believe that our study and experience would be beneficial to the community and could facilitate building highly-reliable SSD-based storage systems.

1 Introduction

Flash-based solid state drives (SSDs) have become an indispensable component of modern datacenters due to its superior performance and low power draw [6]. Various applications, including databases [14], social network [15], and online shopping [41], have been supported by large-scale SSD-based storage systems. Therefore, the reliability of such systems is of critical importance.

However, it is challenging to maintain the high reliability of SSD-based storage systems. First, unlike hard disk drives (HDDs), SSDs may experience unique flash errors (e.g. program errors [20, 38]) which are sensitive to the environment (e.g., temperature [30]). Therefore, our decades of collective wisdom on HDDs is not fully applicable. Second, issues in the traditional HDD-based storage stack (e.g., faulty interconnection and human mistakes [28]) may continue to haunt SSD-based storage systems. In addition, due to the complexity of storage systems, the potential correlations among various events across different levels/components are not well-understood, rendering extreme difficulty in pinpointing the root causes of system failures or comping up with effective fixes.

To address the challenges, substantial efforts have been made to understand the reliability of SSD itself [24, 33, 38, 44]. For example, Schroeder et al. [38] study flash errors and discover correlations between flash errors and other device attributes (e.g., age, wear, lithography). Zheng et al. [44] analyze the behavior of SSDs under power faults. Narayanan et al. [33] analyze a diverse set of device factors (e.g., design and provisioning) and their correlations with failed SSDs. Hao et al. [24] study the performance instability involving millions of drive hours, especially the device latency in RAID groups. While these studies provide valuable insights on the characteristics of SSDs, it remains unclear how SSDs interact with the rest of the system and contribute to system failures.

Besides the work on SSDs, studies on HDD-based storage systems are also abundant [7, 8, 28, 35, 37]. Apart from understanding HDD errors in the field [7, 35, 37], researchers analyze the correlations between HDD errors and system failures [8, 28]. However, since SSD-based systems are significantly different from HDD-based systems (e.g., the TRIM command support throughout the OS kernel [2]), it is unlikely that these studies and findings are directly applicable to SSD-based storage systems.

In this paper, we look into the storage systems deployed in 7 datacenters of Alibaba Cloud, which includes around 450,000 SSDs over 3 years' deployment. Similar to other large-scale deployed systems [16, 17, 29, 42], our target systems are equipped with system monitoring daemons deployed on each node of the clusters. The daemon monitors abnormal behaviors by constantly checking the BIOS messages at boot time, the kernel syslog at runtime, and the functionality and availability of the cloud services. Upon an abnormal event, the daemon will report a failure ticket with the timestamp, the component involved, and a snippet of corresponding logs.

Among all failure tickets, we focus on failures that are **R**eported **A**s "**S**SD-**R**elated" (**RASR**) in this paper. Also, we collect the corresponding repair logs of the failures as well as the SMART [4] logs of the SSDs involved. By holistically analyzing the three datasets (i.e., failure tickets, repair logs, and SMART logs) in the context of the storage systems design and deployment, we identify a number of interesting characteristics of RASR failures in terms of distributions, symptoms, and correlations. Moreover, we perform field studies and validation experiments to understand in depth the factors affecting RASR failures, and to derive a number of major lessons as well as realistic remedies for hardware architects, software engineers, and system administrators. More specifically, our contributions include the following:

(1) Characteristics of RASR Failures. We collect about over 150K failure tickets in total from the target systems. Among these failure tickets, we find that 5.6% are RASR failures (i.e., about 10K instances), which manifested in five symptoms: *Node Unbootable*, *File System Unmountable*, *Drive Unfound*, *Buffer IO Error*, and *Media Error*. By correlating the RASR failures with the repair logs, we find that a significant number (34.4%) of RASR failures are *not* caused by the SSD device. For example, plugging SSDs into wrong drive slots, a typical *human mistake*, accounts for 20.1% of RASR failures. Moreover, for RASR failures caused by SSDs, we find that both the location of devices (i.e., in different datacenters) and the type of cloud services may affect SSD failure rates.

(2) Lessons and Actions for Hardware Architects. We find that the suboptimal intra-node SSD stacking and intra-rack node placement can lead to *passive heating* (i.e., heating on *idle* SSDs by neighboring active SSDs), which may in turn cause a large number of device errors and high failure rates. Moreover, by experimenting on a dedicated cluster with continuous temperature monitoring, we are able to verify that the poor rack architecture can increase the temperature of *idle* SSDs by up to 28 C°, resulting in 57% more device errors after 128 hours of passive heating.

To reduce the impact of passive heating, we formulate a new strategy for intra-rack node placement. Furthermore, we propose a proactive approach to alleviate the passive heating by routinely scanning the entire device to trigger the FTL internal read refresh [11]. Different from the traditional data scrubbing [5, 31], the scanning is lightweight enough to be scheduled more frequently to reduce the effect of passive heating. Our results show that performing a scanning every 4 hours can offset most negative impact of passive heating. Although not observed in our experiments, the scanning may potentially lead to more read disturbs [10], affecting the device negatively. Therefore, we believe it would be ideal for the vendors to implement the proactive scanning at the FTL.

(3) Lessons and Actions for Software Engineers. We find that both the data allocation scheme in the service software stack and the I/O pattern of cloud services play important roles in affecting SSD reliability and leading to RASR failures. For example, the Block service, empowered by a direct mapping based data allocation scheme, can cause severe imbalance of SSD usage when running on top of an HDFS-like distributed file system (DFS): 15-20% SSDs are overly used, which causes up to 77.3% more device errors and up to 18.7% higher device failure rate. Inspired by the log-structured file system [36], we optimize the data allocation scheme on the target systems by adding a shared appending log, and thus mitigate the imbalance issue.

(4) Lessons and Actions for System Administrators. By co-analyzing device-level and system-level logs, we discover a strong correlation between one type of RASR failures (i.e., those caused by faulty interconnection) and one type of device errors (i.e., Ultra-DMA CRC or UCRC). Based on this observation, we design an indicator for the faulty interconnection issue based on the accumulation of UCRC errors, which significantly improves the repair procedure of relevant failures. In addition, we find that SSDs on the target systems serve three different purposes (i.e., system drives, storage, and buffering), but they all use the same SATA interface. This causes much confusion for system administrators who need to replace drives. To reduce the chance of plugging SSDs into wrong drive slots (cause of 20.1% RASR failures), we adapt the systems to use different SSD interfaces for different purposes (e.g., U.2/M.2 for system drives and SATA for storage). This optimization effectively eliminates the confusion and reduce the corresponding failures.

To the best of our knowledge, our work is the first effort on understanding the characteristics of RASR failures as well as the causal relation between SSD errors and the system design and usage, in large-scale production systems. Based on this study, we have significantly improved the reliability of practical systems through a number of simple yet effective mechanisms (e.g., proactive data scanning, UCRC-based indicator and specializing interfaces). We believe that our study and lessons would be beneficial to the community, and could facilitate building highly-reliable SSD-based storage systems.

The rest of the paper is organized as follows: §2 introduces our methodology; §3 analyzes the characteristics of RASR failures; §4 - §6 discusses our lessons and actions for hardware architects, software engineers, and system administrators, respectively; §7 discusses related work, and §8 concludes the paper.

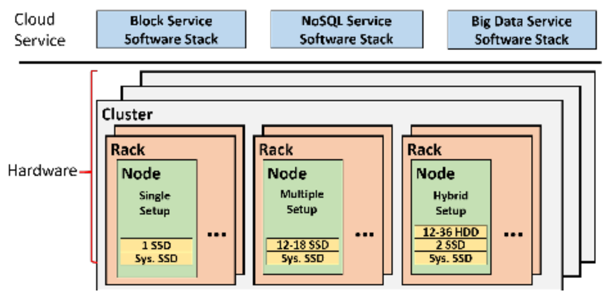

Figure 1: Architecture of target systems.

Model	Capacity	Lithography	Age	Vendor
M1	480 GB	20 nm	2-3 yrs	A
M2	800 GB	20 nm	2-3 yrs	A
M3	480 GB	16 nm	1-2 yrs	A
M4	480 GB	20 nm	2-3 yrs	B
M5	480 GB	20 nm	1-2 yrs	C

Table 1: Characteristics of SSDs in the target systems.

2 Methodology

2.1 System Architecture

We study SSD-based large-scale storage systems deployed in 7 datacenters. The architecture of the target systems is shown in Figure 1.

At the device level, the systems include around 450,000 SSDs spanning three years of deployment. As shown in Table 1, these SSDs cover a spectrum of variability in terms of capacity, lithography, age, and vendors. Note that all five models in our dataset are using SATA interfaces and based on MLC NAND cells.

Each node in the systems employs one of three different setups of SSDs: (1) *Single*: a node contains one SSD for storing temporary data; (2) *Multiple*: a node contains 12 to 18 SSDs for persistent storage; (3) *Hybrid*: a node contains 2 SSDs and 12 to 36 HDDs where the SSDs are used for buffering incoming writes. In addition, each node has one SSD serving as the system drive.

A rack consists of 16 to 48 nodes, and a DFS cluster spans 12 to 18 racks. On top of the DFS, the system supports three types of cloud services, including Block service, NoSQL service, and Big Data service. As shown in Table 2, the cloud services may run on different setups where the SSDs are used for different purposes.

2.2 Raw Datasets

The target systems include sophisticated monitoring mechanisms, similar to other large-scale systems [16, 17, 29, 42]. The monitoring daemons are deployed on each node of the clusters, and they log various events either periodically or upon the occurrence of an event.

At the system level, the daemons monitor BIOS messages,

Service	SSD Model	Setup	Usage
Block	all models	Hy/Mul	Pers/Buf
NoSQL	M1, M3, M4, M5	Hy/Mul	Pers/Buf
Big Data	M1, M2, M4	Single	Temp

Table 2: Cloud services and SSD usages. ***Hy****: Hybrid;* ***Mul****: Multiple;* ***Pers****: Persistent storage;* ***Buf****: Buffering writes;* ***Temp****: Temporarily storing intermediate data.*

kernel syslogs, and the service-level verification of data integrity. Upon an abnormal event, the daemon reports a failure ticket with the timestamp, the related hardware component, and a log snippet describing the failure. Each failure ticket is tagged based on the component involved. For example, if an SSD appears to be missing from the system, the failure ticket is tagged as ”SSD-related”. If there is no clear hardware component recorded in the logs, the ticket would be tagged as Unknown.

At the device level, the daemons record SMART attributes [4] on a daily basis, which cover a wide set of device behaviors (e.g., total LBA written, uncorrectable errors).

Besides the failure tickets and SMART logs, we collect the repair logs of all RASR failures, which are generated by onsite engineers after fixing the failures. For each failure event, the corresponding repair log records the failure symptom, the diagnosis procedure, and the successful fix.

2.3 Study Approaches

After collecting the failure tickets, the SMART logs, as well as the repair logs, we apply the following approaches to derive insights:

- **Log analysis:** We calculate the distributions of failure events along multiple dimensions (e.g., hardware types, manifestation symptoms). Moreover, since the number and the variety of events in the logs is large, we leverage classic statistical algorithms (e.g., Spearman Rank Correlation Coefficient [12]) to analyze the characteristics of individual events as well as the potential correlations among different events.
- **Field studies:** Besides the log analysis, we visit the datacenters in person to investigate the potential variance of target systems in terms of cluster architectures, which turns out to be critical for discovering the passive heating phenomenon (§4). Also, we discuss with on-site engineers to empirically verify our hypothesis on RASR failures.
- **Validation.** We build a dedicated cluster to validate our hypothesis. Moreover, to address the issues exposed in our study, we design a set of remedy methods, and validate the effectiveness and practicability on production systems.

2.4 Limitations

Failure Reporting. Our study relies on the failure tickets reported by distributed daemons that automatically monitor the health condition of system components from hardware to software. The daemons may fail to record (e.g. network

RASR Failure Symptom	Meaning	Distribution
Node Unbootable	Unable to boot the OS on a node	2.6%
File System Unmoutable	Unable to mount a local file system	7.4%
Drive Unfound	A device cannot be found by the system software	53.7%
Buffer IO Error	Unable to write data from memory buffer to the device	17.3%
Media Error	Unable to read correct data from the device	19.0%

Table 3: Distribution of RASR failures based on manifestation symptoms. *This table shows five different symptoms of RASR failures and the corresponding percentage.*

Hardware Type		Distribution
CPU		0.7%
Memory		8.5%
Network		34.0%
Motherboard		5.4%
Storage	HDD	22.1%
	SSD	5.6%
Unknown		23.7%

Table 4: Distribution of failure tickets based on hardware types. *This table shows the distribution of failure tickets that are tagged as related to major hardware components.*

RASR Failure Symptom	Affected Rate (‰)				
	M1	M2	M3	M4	M5
Node Unbootable	0.24	0.42	0.15	0.13	0.07
FS Unmountable	1.28	1.05	0.42	2.90	2.04
Drive Unfound	11.19	8.58	5.31	11.51	4.38
Buffer IO Error	3.73	1.34	1.36	4.06	1.21
Media Error	3.42	5.24	2.81	5.73	1.33

Table 5: Distribution of RASR failures among five SSD models. *This table shows the affected rate of each SSD model (M1-M5), which is the number of SSDs involved in one type of RASR failures divided by the total number of SSDs with the same model.*

failure during log collection) or inaccurately tag the events (e.g. a node crash tagged as "Unknown" due to insufficient logs). However, to the best of our knowledge, the way the tickets are reported is the common practice widely used in major large-scale production systems and previous studies on large-scale deployed systems also rely on similar mechanisms for collecting datasets [16, 17, 29, 42].

Software Stack Design. Our software stack includes OS, DFS and service components. Apart from using a major distribution of Linux, our DFS and service software are not open-source. Nonetheless, they share generic similarities with popular large-scale storage systems such as HDFS [39] and Google File System [19], and similar high-level services are provided by other companies such as EBS [1] and DataStore [3].

Hardware Products. Like previous works [32, 33], the target systems use off-the-shelf hardware products such as SSDs and interconnects. Many products are also widely deployed in the datacenters of other organizations. Therefore, users from other organizations may encounter the same or similar hardware-related issues, and we hope they can benefit from our experiences.

3 Characteristics of RASR Failures

3.1 Overview of Failure Tickets

We collect all failure tickets reported as related to hardware components, over 150K tickets in total. Table 4 shows the distribution of the failure events based on the types of hardware components involved, including CPU, Memory, Network, Motherboard, HDD/SSD, and Unknown. The Unknown type refers to the failures where a relevant component is not specified in the daemon-reported ticket. The second column shows the percentage of failure events for each type of hardware. According to our daemon setup, no failure event is tagged with more than one type.

As shown in Table 4, storage components (i.e., HDD and SSD combined) contribute to 27.7% (i.e., 22.1% + 5.6%) of all hardware-related failure events. RASR failures alone account for 5.6%. Compared with other hardware components (e.g., Network which accounts for 34.0%), RASR failures are much fewer in our dataset. This is consistent with the findings from previous studies that SSD is a relatively reliable component among all hardware components deployed in datacenters [6, 33].

Nonetheless, since the total number of failure events is large (i.e., over 150K), even a relatively small percentage (i.e. 5.6%) of failures cannot be ignored. Therefore, we perform an in-depth analysis on RASR failures in this study and present detailed results in the following sections.

3.2 Symptoms of RASR Failures

After analyzing all RASR failure logs, we find that RASR failures can manifest in multiple ways. As shown in Table 3, there are five different types of manifestation symptoms, including *Node Unbootable, File System Unmountable, Drive Unfound, Buffer IO Error,* and *Media Error*. The meaning of each symptom is described in the second column of the table. Also, the distribution of each type of symptoms is listed in the last column of Table 3.

Among the five symptoms, the Drive Unfound type, which means the device cannot be found by the system software, is the dominant one (i.e., accounts for 53.7%). Based on

Node Unbootable	File System Unmountable	Drive Unfound	Buffer IO Error	Media Error

1.Slot Check(53.8%)
2.Repl. SSD(46.2%)
	1.Mnt. Opt. Check(5.4%)
	2.FSCK(40.5%)
	3.Repl. SSD(54.1%)
		1.Rebooting(22.2%)
		2.Slot Check(34.8%)
		3.Repl. Cable(25.9%)
		4.Repl. SSD(16.1%)
			1.FSCK(79.8%)
			2.Repl.SSD(20.2%)
				1.Data Check(30.2%)
				2.Repl. SSD(69.8%)

Table 6: Repairing procedures of RASR Failures and their successful rates grouped by symptom. *The first row shows five manifestation symptoms of RASR failures. The 2nd row lists repairing procedures for each symptom. The repairing follows an order as indicated by the number before each fix approach. The rate in the parentheses after each fix indicates within that symptom group the percentage of failures fixed by that approach. Repl.: replacing; Mnt. Opt.: Mount Options.*

RASR	Affected Rate (‰)		
Failure Symptom	Block	NoSQL	BigData
Node Unbootable	0.27	0.12	0.35
FS Unmountable	1.43	1.05	1.42
Drive Unfound	13.25	10.58	9.31
Buffer IO Error	5.73	2.34	5.36
Media Error	8.42	3.24	3.77

Table 7: Distribution of RASR failures among cloud services. *This table shows the affected rate of each cloud service (i.e. Block service, NoSQL service and Big Data service), which is the number of M1 SSDs involved in one type of RASR failures divided by the total number of M1 SSDs within the same cloud service.*

RASR	Affected Rate (‰)				
Failure Symptom	DC1	DC2	DC3	DC4	DC5
Node Unbootable	0.35	0.31	0.21	0.27	0.23
FS Unmountable	1.08	1.25	1.42	1.90	1.04
Drive Unfound	10.33	12.72	13.31	13.96	14.10
Buffer IO Error	2.95	2.14	1.98	1.86	2.12
Media Error	2.06	3.04	2.85	**7.73**	3.75

Table 8: Distribution of RASR failures among datacenters. *This table shows the affected rate of each M1 SSD under the Block service from 5 datacenters (DC1-DC5), which is the number of M1 SSDs involved in one type of RASR failures divided by the total number of M1 SSDs under the Block service within the same datacenter.*

our discussion with on-site engineers, a Drive Unfound event may be masked by the system software (e.g., automatic redirection of I/O requests and re-replication of data), and may not necessarily lead to data loss. However, the event can still cause additional latency on the I/O requests involved, and usually requires engineers to diagnose the issue on site. Similarly, other types of RASR failures may also affect system performance and consume manual efforts. Therefore, it is important to understand the root causes of RASR failures and improve the failure handling. We discuss the analysis on fix procedures in §3.3.

After observing the distribution of RASR failure symptoms, we further study the correlation between RASR failure symptoms and other important factors, including SSD models, service workloads, and datacenter locations.

As mentioned in Table 1, there are five different SSD models in our target systems. To further understand the potential impact of SSD models on RASR failures, we calculate the failure affected rate for each model, which is the number of SSDs involved in one type of RASR failures divided by the total number of SSDs with the same model. As summarized in Table 5, the five RASR failure symptoms have been observed on all five SSD models (M1-M5). The affected rate ranges from 0.07‰ (i.e., M5 SSDs with the Node Unbootable symptom) to 11.51‰ (i.e., M4 SSDs with the Drive Unfound symptom). We do not observe statistically significant difference among SSD models in terms of the affected rate of RASR failures, which suggests that RASR failures may not be directly related to the characteristics of SSD models.

To study the correlation between RASR failures and service workloads running on the target systems, we use M1 SSDs, a popular model accounting for 35% of the drive population. Table 7 shows the affected rates of M1 SSDs under three cloud services. We observe that the Block service (2nd column) has the highest affected rates in four out of five types of RASR failures (except Node Unbootable). This finding motivates us to further investigate the cloud services with their designs, drive usage and device level errors in §5.

In addition, we study whether the location (i.e. datacenters) plays a role in RASR failures. We evaluate the affected rates of M1 SSDs under the Block service (i.e. the main service accounting for for 57% of SSD deployment) in different datacenters (DCs). Table 8 summarizes the results. Note that M1 SSDs of the Block service are only used in five datacenters, i.e., from DC1 to DC5. From the table, we observe that while no DC dominates all failure types, DC4 has substantially more Media Errors (i.e. last row), indicating more data corruptions. To better understand the potential root causes, we study the uniqueness of DC4 in terms of hardware architectures, especially the SSD placement in §4.

3.3 Fixes of RASR Failures

To understand the potential root causes of RASR failures, we further analyze the corresponding repair logs. For each failure, administrators apply a symptom-based repairing procedure, i.e., trying a pre-defined sequence of fix candidates one by one based on the failure symptom until the failure disappears. Each repair log records the repairing process and the successful fix of a failure event. Table 6 summarizes the pre-defined sequence of fix candidates for each RASR failure symptom. Also, for each fix candidate, we calculate its successful rate in the group of failures with the same symptom (shown in the parentheses).

For instance, after observing a Drive Unfound failure (3rd column of Table 6), administrators will first attempt to remotely reboot the node to check whether the failure is transient ("Rebooting"). If not, administrators will manually check whether the device is plugged into the correct slot ("Slot Check"). If the slot is correct, administrators will then try replacing the cable ("Repl. Cable"), followed by replacing SSD ("Repl. SSD") as a final resort until the failure is resolved. Note that all RASR failures are eventually fixed by replacing SSDs if previous attempts do not work.

One observation on Table 6 initially puzzling us is that the first fix attempt is not always the most effective one within each group of failures. For example, "Mnt. Opt. Check" (Mount Options Check) works only for around 5% of File System Unmountable failures (2nd column). Similarly, "Data Check" cures 30.2% of Media Error events (last column). After discussing with administrators, we realize that the symptom-based repairing procedure overall is simple yet effective. Specifically, the order of the fix candidates for each failure symptom is first based on their costs, followed by their effectiveness. As a result, the set of software-based fixes (i.e., checking mount options, rebooting, FSCK, and data check) are always preferred over the set of manual or hardware-based ones (i.e., slot check, replacing cable, and replacing SSD). The order within either set of fix candidates is based on their effectiveness to solve the failure symptoms in administrators' past experiences. The sequence of fix candidates for repairing Drive Unfound (3rd column) clearly demonstrates the ordering consideration.

Although existing fix procedure is effective to certain degree, it is a black-box approach (trail-and-error) since the administrators do not know the root causes before applying the fix candidates. This motivates us to conduct in-depth study on the potential root causes of RASR failures for helping system administrators with better fix strategy. As will be discussed in §6, we identify an accurate indicator for one type of failures (§6.1) and propose a method for avoiding another type of failures (§6.2).

3.4 SMART Logs under RASR Failures

The device level SMART [4] log is an important dataset for analyzing SSD behaviors and failures in the field [32, 33, 38]. Similar to previous studies [32, 33, 38], we analyze a subset of SMART attributes (as shown in Table 9) on our target systems in depth and observe a number of characteristics which are consistent with the prior work (e.g., the prevalence of uncorrectable errors and the high raw bit error rate on failed drives). Due to space limit, we do not discuss the observations or the distribution of SMART attributes that are similar to prior work. Instead, we correlate the SMART logs with RASR failures in later sections and analyze the impact of different factors (e.g., hardware architecture and software design) on drive behaviors.

Device Level Event	Definition
Host Read	Total amount of host LBA read from SSD
Host Write	Total amount of host LBA write to SSD
Program Error	Total # of errors in NAND programming operation
Raw Bit Error Rate (RBER)	Total bit corrupted divided by total bits accessed
End-to-End Error (E2E)	Total # parity check failures between drive and host
Uncorrectable Error	Total # of data corruption beyond ECC's ability
UDMA CRC Error (UCRC)	Total # of CRC check failures during Ultra-DMA

Table 9: Device level events collected in our study.. *Device level events are collected via SMART [4]. All events are recorded in a cumulative manner.*

4 Lessons & Actions for Hardware Architects

During our characteristics study of RASR failures (§3), we observe that SSDs deployed in one particular datacenter (DC4) experience much more Media Error under the Block Storage service (Table 8). Moreover, these SSDs have higher Raw Bit Error Rate (RBER) and Uncorrectable Bit Error Rate (UBER) based on the SMART logs.

To understand why the Block service in DC4 is so unique, we perform field studies at DC4 and other datacenters. We find that there are two potential factors. First, in DC4, about 27.1% Block service nodes are equipped with 18 SSDs, while in other datacenters less than 5.3% Block service nodes have 18 SSDs (most nodes have 12 SSDs). Second, in DC4, nodes for different services are often co-located in the same rack, while in other datacenters a rack is exclusively used for a single service. In this paper, we refer to the two factors as *intra-node SSD stacking* and *intra-rack node placement*, respectively, both of which affect the SSD placement in the systems. Since NAND flash memory is known to be less reliable under higher temperature [9] due to the Arrhenius Law[34], we suspect that the SSD placement may affect the airflow in nodes and racks, which may in turn affect the operating temperature of neighboring SSDs, and then lead to abnormal behaviors. We refer to this hypothesis as *passive heating*.

Note that passive heating is different from heating mechanisms used in prior work for analyzing NAND flash or

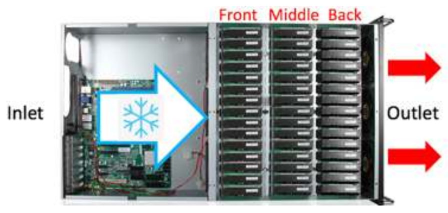

Figure 2: Intra-node SSD stacking and the airflow. *This figure shows the stacking of SSDs within a node, which include three groups:front, middle and back. The arrows indicate the direction of the airflow for cooling. Note that due to confidentiality, we cannot show the real photo of the node deployed in our target systems; however, the actual node is very similar to this example.*

SSD under high temperatures. Specifically, in the studies on NAND retention errors, NAND chips are heated to high temperature without power supply (e.g., heating in the oven) [9]. Differently, SSDs in our study are always powered on, where the FTL may proactively reduce NAND errors. In previous studies on the impact of SSD temperatures, they mostly focus on active heating, i.e., heating the SSDs by *heavily accessing* them. In such case, high temperature may trigger the throttling mechanism in FTL to reduce errors [32]. On the contrary, the passing heating we observe may affect *idle* SSDs, which cannot be remedied by throttling (because there is no heavy on-the-fly flash operations to throttle). Therefore, we believe it is necessary to investigate the passive heating further.

4.1 Identify and Verify Passive Heating

With the help of on-site engineers, we identify three potential scenarios where SSDs may suffer from excessive passive heating:

- **Hot Airflow.** Figure 2 shows an example of stacking of multiple SSDs within a node and the airflow for cooling. In this design, idle SSDs at the outlet of the airflow may be heated up when the front SSDs are being accessed heavily.
- **Hot Neighbors.** If an idle node is close to another node running intensive workloads, SSDs in the idle node may be heated up by the hot neighboring node.
- **Hot Air Recirculation.** When a node is removed out of the rack, the empty node slot may serve as a channel for tunneling hot airflow and passing heat to nearby nodes (one empty node slot away).

To verify and measure the passive heating, we build an experimental cluster with continuous monitoring of SSD temperatures and controlled workloads. The cluster includes 8 nodes in a dedicated rack, and each node has 18 SSDs. We perform the following experiments to analyze the passive heating in each of the aforementioned scenarios.

For Hot Airflow, we first record the initial temperature of the SSDs near the outlet of the airflow when a node is just powered on. Then, we run intensive workloads to access the 6 front SSDs (i.e. SSDs close to the inlet of the airflow), but leave the remaining 12 SSDs idle. We compare the temperatures of the idle drives before and after running the workloads.

For Hot Neighbors, we run intensive workloads on some nodes, and keep monitoring the temperatures of the SSDs on the neighboring idle nodes. We try three configurations where the hot neighbor(s) is atop, below, or are at both sides of the idle node.

For Hot Air Recirculation, we remove a node from the rack and examine whether the temperature of the SSDs of an idle node can be affected by a hot neighbor that is one node slot away.

Our experiments show that for an idle SSD initially at 25 C°, it can be heated up by 23 C°, 9 C°, and 17 C° (i.e., reaching 48 C°, 34 C°, and 42 C°) via Hot Airflow, Hot Neighbors, and Hot Air Recirculation, respectively. Moreover, when combining the three effects, an idle SSD can be heated up by 28 C° (i.e., reaching 53 C°) on our cluster.

4.2 Effects of Passive Heating on SSDs

After verifying that the suboptimal SSD placement may generate undesirable passive heating on SSDs, we look into the impact of passive heating on SSDs' behavior. In this set of experiments, we compare the raw bit errors of SSDs at three levels of temperatures under passive heating: 35C°, 45C°, and 55C°. Note that we use a fixed temperature interval (i.e., 10C°) to make the correlation between errors and passive heating more clear.

Specifically, in each experiment, we heat up idle SSDs (initially 25C°) through passive heating until they reach one of the three levels of higher temperatures, i.e., 35C°, 45C°, or 55C°. At each level, we maintain the same temperature for a range of time durations, i.e., from 1 to 128 hours, by carefully adjusting the workloads on neighboring nodes based on the feedback of the measured SSD temperature. After the stable passive heating period finishes, we scan the whole device and measure the Raw Bit Errors[1] newly generated during the heating period.

Figure 3 summarizes the results. We find that all three levels of passive heating (i.e., 35C°, 45C°, and 55C°) may lead to more Raw Bit Errors compared with normal case (i.e., 25C°). Additionally, a higher level of passive heating (e.g., 55C°) for a longer period of time (e.g., 64 hours) can generate more Raw Bit Errors, and the increasing trend is nonlinear. Moreover, after 128 hours of heating, we observe that idle SSDs suffer from 57% more Raw Bit Errors.

Note that our observation in this set of experiments (i.e., higher temperature leads to more retention errors) aligns well with previous studies and industry standards[27, 32]. However, it contradicts to a recent study on 3D NAND flash chips [30]. This is likely because the structure and characteristics of 3D NAND are different from those used in our systems

[1]We do not use the Raw Bit Errors Rate (RBER) attribute directly because it is a cumulative value over the entire lifespan of a device.

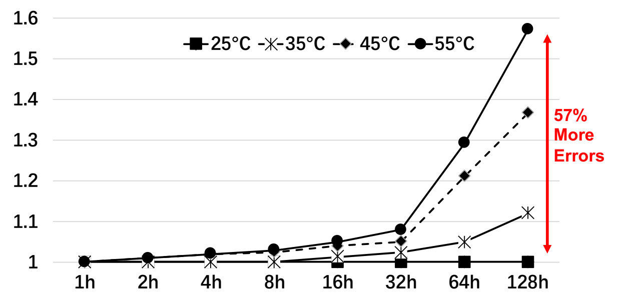

Figure 3: Raw Bit Errors generation under passive heating through time

(e.g., charging trap versus floating gate).

Although in our small scale experiments we do not observe uncorrectable errors or RASR failures, we believe that the results (e.g., increasing Raw Bit Errors) indicate that SSDs may become less reliable due to passive heating, and the phenomenon deserves more attention.

4.3 Offset the Impact of Passive Heating

Since the idle SSDs that are suffered from passive heating do not serve any I/O (before measuring the Raw Bit Errors), the increased Raw Bit Errors are most likely due to a retention issue. Classic techniques like data scrubbing can effectively mitigate retention issue by scanning and checking data integrity. However, it is unrealistic to apply such techniques frequently due to the prohibitive performance overhead.

On the other hand, we realize that the FTL in SSDs usually has a mechanism called Read Refresh [11] to correct bit errors and reallocate data during reading. So we propose to apply a lightweight regular software-based scanning to trigger read refresh (without computing checksums) to offset the negative impact of passing heating on idle SSDs. To verify our proposed method, we experiment on different intervals of scanning (e.g., 1 to 128 hours) and measure the reduction of Raw Bit Errors. Our experimental results are very promising: a routine scanning of every 4 hours can effectively control Raw Bit Errors without incurring too much overhead in our target systems. For example, after we perform a 4-hour routine scanning on the idle SSD during its 128 hours of passive heating under 55 C°, we only observe 1% more Raw Bit Errors, which is in stark contrast to 57% more Raw Bit Errors without scanning. Further increasing the frequency of scanning do not reduce the errors much. Therefore, the 4-hour-scanning routine achieves a good balance between the effectiveness and overhead in our systems.

While triggering read refresh by routine scanning is helpful for offsetting the impacts from passive heating, there are other potential issues with its direct deployment on production systems. First, the routine scanning requires fine-grained temperature monitoring to detect passive heating. Currently, the SSD temperature on our target systems is obtained by querying the SMART logs. Similar to other cloud companies [32, 33], the SMART logs are pulled on a daily basis in our production systems, which is insufficient for monitoring passive heating. Increasing the query rate requires changes to the distributed monitoring daemons and may affect the quality of service. While some hardware-based temperature querying methods (e.g., IoT sensors [18]) are relatively lightweight, integrating them into production systems may require significant efforts.

Second, the scanning might introduce more read disturb errors [10]. Although the scanning does not necessarily read the entire disk (i.e. only the stored data) or blindly get executed every 4 hours (i.e. only when SSD is in passive heating for more than four hours), the SSD may still suffer from increasing device errors due to read disturbance. This may further deteriorate as the lithography becomes smaller. Therefore, while effective, it is difficult to directly apply the routine scanning used in our experiments to production systems.

Alternatively, it is possible to implement our proposed technique of detecting and remedying passive heating in FTL with vendors' support. First, many SSDs today support heat throttling in the FTL, which implies that the temperature is already closely monitored by the device. Second, the FTL has the best knowledge of which parts of the data have higher error rates, and thus can react accordingly by proactively read refreshing the corresponding data. Therefore, the FTL-based solution may be more effective. We hope our study can raise the awareness of passive heating and facilitate addressing the issue.

5 Lessons & Actions for Software Engineers

As shown in Table 7, the SSDs under the Block service suffer more RASR failures than the devices under the other two services. This finding motivates us to further investigate the behavior differences of the SSDs among the three services, as well as the potential causes and fixes.

5.1 Usage Imbalance in Block Service

We start with the SSD usage, the most fundamental statistics of device behaviors. The three cloud services (i.e. Block, NoSQL, and Big Data Analytics) supported by our target systems are intrinsically different in terms of data placement policies and I/O patterns, which may lead to different usage patterns of SSDs. To understand the basic usage, we compare two device-level events: *Host Read* and *Host Write*, which measure the amount of data read from or written to the device by the host.

More specifically, we measure the hourly average value of host read/write (i.e., total sizes of host read/write divided by total power-on hours) on all SSDs under each cloud service. Moreover, we calculate the variability of the two metrics among SSDs under the same service using the coefficient of variation (CV), which is the ratio of standard deviation to

		Host Read	Host Write
Avg. Value /Hour	Block	7.69 GB	6.56 GB
	NoSQL	6.10 GB	5.28 GB
	BigData	1.57 GB	1.22 GB
CV	Block	35.5%	24.9%
	NoSQL	3.2%	6.2%
	BigData	1.8%	3.7%

Table 10: Comparison of SSD usages under three services in terms of host read and host write. *CV: Coefficient of Variance, the ratio of standard deviation to mean.*

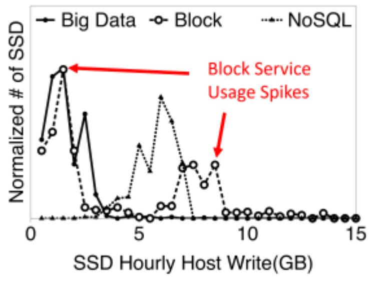

Figure 4: Distribution of SSDs under three services. *This figure shows the distribution of SSDs in terms of hourly host write under three services. The arrows mark the bimodal usage under the Block service.*

mean. Intuitively, a higher CV indicates that the hourly host read/write varies more across SSDs.

Table 10 summarizes the results. We can see that the hourly average value of host read and host write of the Block service are 7.68 GB and 6.56 GB, respectively, which are similar to those of the NoSQL service. However, the Block service has much higher variances for the two metrics (i.e., 35.5% and 24.9%), which implies that the usage of SSDs under this service is much more unbalanced.

Figure 4 further illustrates the distribution of SSDs in terms of hourly host write under the three services by using a histogram with 0.5 GB buckets along the x-axis. Each dot on the line (e.g., solid line for Big Data) represents the cumulative count of SSDs in the corresponding usage bucket. We can see from the figure that the majority of SSDs under NoSQL and Big Data Analytics services have similar usages (i.e., one major spike on the corresponding curve). In contrast, the SSDs under Block Storage service shows bimodal usages (i.e., two spikes far apart) as marked in the figure. Further analysis shows that the overly used drives (i.e. the right spike) account for around 17% of all SSDs in the Block Storage service and have 227.1% more write usage. The distribution of SSDs in terms of hourly host read exhibits similar pattern.

With such an unbalanced usage pattern, the overly-used set of SSDs may be worn out quickly. As a result, compared with averagely-used SSDs (i.e., balanced usage), overly-used

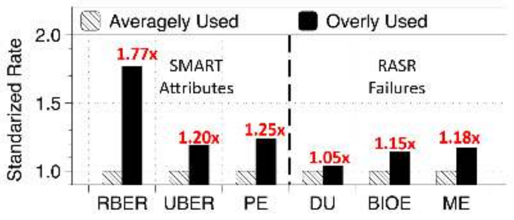

Figure 5: Comparison of overly used SSDs and averagely used SSDs. *This figure shows overly used SSDs exhibit more device level errors and RASR failures compared with averagely used SSDs. RBER: raw bit error rate; UBER: uncorrectable bit error rate; PE: program error count; DU: Drive Unfound; BIOE: Buffer IO Error; ME: Media Error.*

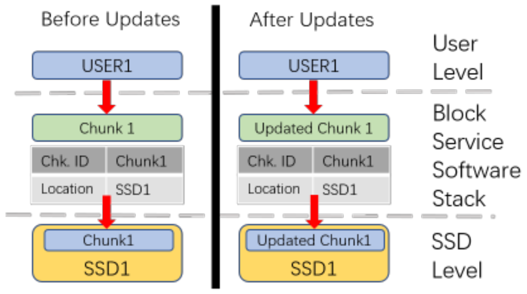

Figure 6: The data path of an update operation (original).

SSDs may exhibit more device-level errors and potentially lead to more RASR failures. To verify this hypothesis, we quantitatively measure such difference based on our dataset. We use the classic 80/20 rule to group the SSDs. The SSDs with top 20% usage within the Block Service are labeled as overly-used and the rest are labeled as averagely-used. As shown in Figure 5, overly-used SSDs have noticeably higher numbers of device errors including RBER (1.77X), UBER (1.20X) and PE (1.25X). Moreover, they tend to incur more RASR failures including Drive Unfound (1.05X), Buffer IO Error (1.15X), and Media Error (1.18X). This result suggests that load balancing is indeed important.

5.2 Root Causes of Usage Imbalance

After looking into the design of software stacks of the three cloud services, we identify two major factors for the unbalanced usage of SSDs in the Block service: the update policy and the user I/O patterns.

Figure 6 shows the simplified update policy in the Block service (for clarity, irrelevant details such as sharding and replication are omitted). The Block service offers users the storage capacity at the granularity of chunks. The left part of the figure shows that USER1 subscribes one chunk ("Chunk1") from the service. The software stack of the Block service maintains a mapping table from the chunk to a fixed SSD (i.e., storing "Chunk1" on"SSD1").

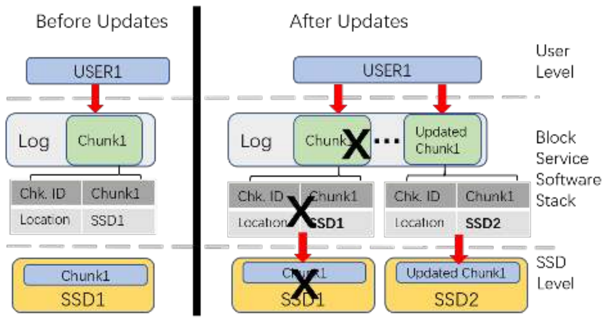

Figure 7: The data path of an update operation (optimized).

Fix	Percentage	Root Cause
Rebooting	11.9%	Transient
Mount Options Check	0.4%	Human Mistake
FSCK	16.5%	Undetermined
Data Check	6.0%	Undetermined
Slot Check	20.1%	Human Mistake
Replacing Cable	13.9%	Faulty Cable
Replacing SSD	31.2%	Failed Device

Table 11: Working fixes of RASR failures. *The first column shows working fixes of RASR failures. The 2nd column lists the percentage of RASR failures repaired by each fix. The 3rd column lists the corresponding root cause derived from the working fix.*

Upon an update operation, shown in the right part of Figure 6, the software stack queries the mapping table and writes the updated chunk to the *same* SSD (i.e., ”Updated Chunk1” on ”SSD1”). In other words, in the Block service performs *in-place* updates, i.e., updates are always flushed to the initially-allocated SSDs.

In addition, we find that the Block service receives a diverse set of I/O requests from different users. Some users generate many update operations while others do not. This diversity and the in-place update policy lead to the unbalanced usage of SSDs under the Block service.

Unlike Block Service, the other two cloud services do not cause severe usage imbalance because they have a different update policy or I/O pattern. Particularly, the NoSQL service merges small updates together and always generates a new chunk for the updated data, which can be mapped to a different SSD. In the Big Data service, reading and adding new data are consistently much more frequent than updating existing ones. Therefore, SSDs under both services have a relatively balanced usage.

5.3 Mitigating Usage Imbalance

To address the usage imbalance issue, we optimize the software stack of the Block service by adding a shared append-only log, similar to LFS [36].

Figure 7 shows the update operation after the optimization. Similar to Figure 6, USER1 subscribes a chunk from the Block Service. Unlike the original design, the chunk is now maintained in a log which appends the latest update to its end. Upon receiving an update, the software stack invalidates the previous chunk (marked with an “X”), appends the update to the log, and changes the mapping table to map the updated chunk to a new SSD (“SSD2”). The outdated chunk will be invalidated for garbage collection. This log-based design mitigates the usage variance among SSDs, as each updated chunk will be allocated to a different SSD based on the wear among available drives. Note that, in certain cases, the updated chunk may still mapped to the original SSD if the original one happens to be the most appropriate candidate.

After applying the optimized design on a subset of our target systems for 7 months, we observe that the coefficient of variance (CV) of host read/write under the Block service reduces significantly (i.e., from 24.9% to 5.2% among all SSDs under the same service). The log-structured design also provides other benefits for our target systems (e.g., better support for snapshots), which are beyond the scope of this paper.

6 Lessons & Actions for System Admins

To help system administrators with better fix strategy, we need better understanding of the root causes of RASR failures. While the repair log of a RASR failure does not explicitly state the root cause, we can infer the potential root cause based on the successful fix in the log. For example, if a failure can only be fixed by replacing the SSD, it is likely that the root cause is a failed SSD. Table 11 shows all the seven fixes deployed in the repairing procedure of RASR failures, the percentage of RASR failures being successfully repaired by each fix, and the potential root causes.

We observe that not all RASR failures are caused by failed SSDs. There are two main non-SSD causes for RASR failures: (1) *human mistakes* contribute 20.5% of RASR failures, including plugging the device to the wrong slot (“Slot Check”) and incorrect configuration (“Mount Options Check”); and (2) *faulty interconnections* fixed by replacing cable, which is outside of SSD, account for 13.9% of RASR failures. Note that, although “Replacing SSD” accounts for the most (31.2%) of RASR failures, we still leave it as the last resort in the fix strategy due to the high cost of the devices and labors. Hence, we are interested in whether faulty interconnections and human mistakes can be quickly diagnosed or largely avoided.

6.1 Faulty Interconnection

Faulty interconnection is a well-known issue in large-scale storage systems [28]. To fix the RASR failures (Drive Unfound) caused by faulty interconnection, replacing the cable between SSD and host is an effective method. However, our symptom-based repairing procedure (shown in Table 6) lists replacing cable as the third step to try out if a Drive Unfound

Fix	Heavy Group	Light Group
Rebooting	4.4%	25.0%
Slot Check	7.6%	35.7%
Repl. Cable	70.6%	24.2%
Repl. SSD	17.4%	15.1%

Table 12: Success rates of fixes in two SSD groups. *This table shows the success rate of each fix for the "Drive Unfound" failures in two SSD groups classified by the indicator.*

failure occurs. This incentivizes us to quest for good indicators of faulty interconnection. If successful, administrators can directly replace cable instead of trying the first two failed attempts – significantly improving the repairing procedures of Drive Unfound (a major source of RASR failures).

6.1.1 Identifying Potential Indicators

To find a suitable indicator for faulty interconnection, we study the correlation between five representative device errors (i.e., Ultra-DMA CRC (UCRC), RBER, Uncorrectable Errors, Program Errors, and End-to-End Errors) and faulty interconnection by using Spearman Rank Correlation Coefficient [12]. The result shows that only the UCRC has strong correlation with faulty interconnection. This generally indicates that the more UCRC errors an SSD has, the more likely a Drive Unfound failure is caused by faulty interconnection. Therefore, we select the number of UCRC errors for designing the indicator.

6.1.2 Refining the Indicator

Since UCRC errors may also occasionally caused by transient factors (e.g., voltage spike), it is necessary to set a proper threshold for indicating faulty interconnection. To this end, we apply a set of classic statistics methods (e.g., Kolmogorov-Smirnov Test [13]) to analyze the UCRC errors and derive the optimal threshold. We find that the distribution of UCRC errors follows the 80/20 rule (i.e., Pareto Law [13]). So if the accumulation of UCRC errors on an SSD is in the top 20% among all drives, we assign the SSD to the "Heavy" group. Our analysis shows that 17 is the best threshold for our systems. In other words, when an SSD has 17 or more UCRC errors, it is a strong indication of a faulty interconnection in the target systems. Note that the threshold can be re-calculated and updated periodically for the change of systems and environment (e.g., aging of SSDs, workload changes). We leave the sensitivity study of the threshold as future work.

6.1.3 Verifying the Indicator

We further use our existing dataset to verify the effectiveness of the UCRC-based indicator. Specifically, we first divide all SSDs into two groups based on the threshold: "Heavy" (above or equal to the threshold) and "Light" (below the threshold). Then, we calculate the successful rate of each fix candidate for the "Drive Unfound" failures in each group.

Table 12 demonstrates that our indicator for faulty interconnection would be very effective for improving the repairing procedure of Drive Unfound failures. Most (i.e., 70.6%) SSDs in the "Heavy" group have been successfully repaired by replacing cable. On the contrary, only 12.0% of SSDs in the group are fixed by the first two candidates (i.e., node rebooting and slot check). This result suggests that, it can significantly improve the successful rate of the first attempt if we directly replace cables for the drives that are severely affected with UCRC errors. As for the "Light" group of SSDs, whose root causes are not identified as faulty interconnection by our indicator, the successful rate of replacing cable is similar to the first two fix candidates. This shows that the existing repair procedure is good for "Light" group of SSDs.

6.1.4 Benefits of Using Indicator

We have applied the UCRC-based indicator to our target systems. With the new repairing procedure for Drive Unfound failures, if the number of UCRC errors in SSD is higher than the threshold, on-site engineers will start with replacing cable first.

One might think that rebooting is simple and it should be the first attempt no matter what the root cause is. However, the side effect of rebooting can be notable and cascading in large-scale production systems. For example, a node may hang at BIOS during reboot if the system drive is inaccessible due to a faulty cable, which may further trigger large data transfer in a 3-replica system. Therefore, when the root cause is likely to be a faulty cable (i.e., the heavy group), we use cable replacement first.

Based on the feedback of the on-site engineers on the new 89 cases of Drive Unfound (not included in our dataset), the indicator helps them reduce the repairing time by 21.1%, because of the saving on time that would have been spent on the first two unsuccessful attempts (i.e., node rebooting and slot check).

6.2 Human Mistakes & Solution

As shown in Table 11, human mistake is another major source of RASR failures. Particularly, plugging the device to the wrong slot (i.e., fixed by "Slot Check") accounts for 20.1% RASR failures. Although it may be part of human nature to err, we believe such mistakes should be avoided.

To address the issue, we design an approach called One Interface One Purpose (OIOP) for our latest and future deployment, where SSDs serve for different purposes use different hardware interfaces. Table 13 lists the interface for each type of SSD functionality in our target systems. We use U.2/M.2 interface for system drives as our motherboard usually has 1 or 2 such sockets. The NVMe interface is used for temporary storage and buffering as these SSDs require high bandwidth and low latency. The SATA interface is used for persistent storage for compatibility concerns (i.e., re-using current SSDs on new racks). With such an OIOP design, the

SSD Functionality	SSD Interface
System drive	U.2/M.2
Temporary storage	NVMe
Buffering writes	NVMe
Persistent storage	SATA

Table 13: Mapping between SSD functionality and interface.

SSDs and slots for different purposes are easily differentiable by system administrators. Note that these interfaces are unlikely to be transitional as each interface has its unique market/purpose (e.g., U.2/M.2 for embedded, NVMe for high-performance).

Although simple, the OIOP design has effectively reduced the RASR failures caused by human mistakes in practice. In the 6-month deployment of an OIOP storage system with about 100K SSDs, we only observe 3 RASR failures caused by plugging a device to a wrong slot. In stark contrast, we observe an average of 47 such cases on comparable size of current storage systems without OIOP.

Besides OIOP, another possible solution is to use a status light to differentiate the functionalities of drives. Status light has been used for indicating drive status in RAID systems [40], and it can be applied to motherboards without multiple interfaces. However, adding status lights requires support from hardware manufacturers.

7 Related Works

Our work is mainly related to the following three lines of research studies: (1) reliability of SSDs and SSD-based storage systems, (2) reliability of HDD-based storage system, and (3) large-scale failure studies.

Great efforts have been made on analyzing the reliability of SSDs and SSD-based storage systems [32, 33, 38, 43, 44, 45]. For example, Schroeder et al. [38] conduct a large-scale field study covering millions of drive days and analyze a wide range of device characteristics and errors (especially RBER and UBER) as well as their correlation. Meza et al. [32] study flash memory failures in the field as well as their correlation with other factors (e.g., data written from OS). Narayanan et al. [33] analyze the correlation between failed SSDs and other factors (e.g., hardware utilization). Our work is different in a number of ways. First, we focus on RASR failures, which have not been studied before. Second, our study covers system-level failure symptoms, repair procedures, root causes, as well as the casual relations among events. Third, we design and validate a set of simple yet effective solutions. Therefore, we believe our work is complementary to the existing efforts.

Research efforts on HDD-based storage stack are also abundant [7, 8, 28, 35]. For example, Jiang et al. [28] study the logs from around 40K storage systems and discover several findings including the significant contribution of physical components and protocol stacks in failures, the “bursty” failure pattern, and the benefit of using redundant interconnection. Bairavasundaram et al. [7] analyze over 1.5 million hard drives and find out the severity differences of data corruption among enterprise and nearline disks, the spatial and temporal locality of checksum mismatches, and the correlation of data corruption across different disks. Based on the same dataset, Bairavasundaram et al. [8] also analyze factors that contribute to the latent sector errors along with the trends and further explore possible remedies towards building a more robust storage subsystem. However, due to the difference in both hardware design and software support, their findings may not be directly applicable to SSD-based storage systems.

In addition, our work is closely related to two groups of studies on large-scale failures. The first group focuses on using failure reports (e.g., news and descriptive records) to understand failures in modern storage systems [21, 22, 23, 43]. For example, Gunawi et al. [23] collect around 100 hardware fail-slow reports across multiple large-scale deployments from several institutions and study the behaviors, root causes and lessons for dianosis of fail-slow failures. In the second group of studies, researchers have made efforts on diagnosing and detecting failures from software perspective. For instance, Huang et al. [25, 26] analyze the production systems deployed at Microsoft and discover a key feature of the gray failure, differential observability, which leads them to build a fast detection tool. Regarding the first group, our work is different as we use multiple log sources (e.g. SMART logs, kernel logs) to conduct quantitative analysis and further derive the causal relationships of the failures. Compared with the second group, our work targets the various aspects of system reliability maintenance, including not only the software but also the hardware and administration.

8 Conclusions

We study the characteristics of RASR failures in large-scale storage systems in this paper. Our study reveals the distribution, symptoms, and causes of RASR failures. Moreover, we derive several lessons on system reliability, including the passive heating phenomenon, the usage imbalance, human mistakes, etc. In addition, we design and validate a set of simple yet effective methods to address the issues observed. We believe our findings and solutions would be beneficial to the community, and could facilitate building highly-reliable SSD-based storage systems.

Acknowledgments

We thank the anonymous reviewers and Mahesh Balakrishnan (our shepherd) for their insightful feedback. We also thank Yong Wang, Qingda Lu, Cheng He in Alibaba for the invaluable discussion.

References

[1] Amazon elastic block store, 2018. https://aws.amazon.com/ebs/.

[2] fstrim(8) - linux man page, 2018. https://linux.die.net/man/8/fstrim.

[3] Google cloud datastore, 2018. https://cloud.google.com/datastore/.

[4] Self-monitoring, analysis and reporting technology (s.m.a.r.t.) attributes, 2018. https://en.wikipedia.org/wiki/S.M.A.R.T.

[5] AMVROSIADIS, G., BROWN, A. D., AND GOEL, A. Opportunistic storage maintenance. In *Proceedings of the 25th Symposium on Operating Systems Principles (SOSP)* (2015).

[6] ANDERSEN, D. G., AND SWANSON, S. Rethinking Flash in the Data Center. *IEEE Micro 30*, 4 (2010), 52–54.

[7] BAIRAVASUNDARAM, L. N., ARPACI-DUSSEAU, A. C., ARPACI-DUSSEAU, R. H., GOODSON, G. R., AND SCHROEDER, B. An analysis of data corruption in the storage stack. *ACM Transactions on Storage (TOS) 4*, 3 (2008), 8:1–8:28.

[8] BAIRAVASUNDARAM, L. N., GOODSON, G. R., PASUPATHY, S., AND SCHINDLER, J. An analysis of latent sector errors in disk drives. In *Proceedings of the 2017 ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)* (2007).

[9] CAI, Y., HARATSCH, E. F., MUTLU, O., AND MAI, K. Error patterns in mlc nand flash memory: Measurement, characterization, and analysis. In *Proceedings of the 2012 Design, Automation Test in Europe Conference Exhibition (DATE)* (2012).

[10] CAI, Y., LUO, Y., GHOSE, S., AND MUTLU, O. Read disturb errors in mlc nand flash memory: Characterization, mitigation, and recovery. In *Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (2015).

[11] CHO, S. Nand reliability improvement with controller assisted algorithms in ssd. In *Proceedings of the 2013 Flash Memory Summit* (2013).

[12] CORDER, G., AND FOREMAN, D. *Nonparametric Statistics: A Step-by-Step Approach*. Wiley, 2014.

[13] DANIEL, W. *Applied nonparametric statistics*. PWS-Kent Publ., 1990.

[14] DEBNATH, B., SENGUPTA, S., AND LI, J. Flashstore: High throughput persistent key-value store. In *Proceedings of the 36th International Conference on Very Large Data Bases (VLDB)* (2010).

[15] DONG, S., CALLAGHAN, M., GALANIS, L., BORTHAKUR, D., SAVOR, T., AND STRUM, M. Optimizing space amplification in rocksdb. In *Proceedings of the 8th biennial Conference on Innovative Data Systems Research (CIDR)* (2017).

[16] FORD, D., LABELLE, F., POPOVICI, F. I., STOKELY, M., TRUONG, V.-A., BARROSO, L., GRIMES, C., AND QUINLAN, S. Availability in globally distributed storage systems. In *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (2010).

[17] GARRAGHAN, P., TOWNEND, P., AND XU, J. An empirical failure-analysis of a large-scale cloud computing environment. In *Proceedings of the 15th IEEE International Symposium on High-Assurance Systems Engineering* (2014).

[18] GARULLI, N., BONI, A., CASELLI, M., MAGNANINI, A., AND TONELLI, M. A low power temperature sensor for iot applications in cmos 65nm technology. In *Proceedings of the 7th IEEE International Conference on Consumer Electronics - Berlin (ICCE-Berlin)* (2017).

[19] GHEMAWAT, S., GOBIOFF, H., AND LEUNG, S.-T. The google file system. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP)* (2003).

[20] GRUPP, L. M., CAULFIELD, A. M., COBURN, J., SWANSON, S., YAAKOBI, E., SIEGEL, P. H., AND WOLF, J. K. Characterizing flash memory: Anomalies, observations, and applications. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (2009).

[21] GUNAWI, H. S., HAO, M., LEESATAPORNWONGSA, T., PATANA-ANAKE, T., DO, T., ADITYATAMA, J., ELIAZAR, K. J., LAKSONO, A., LUKMAN, J. F., MARTIN, V., AND SATRIA, A. D. What bugs live in the cloud? a study of 3000+ issues in cloud systems. In *Proceedings of the 5th ACM Symposium on Cloud Computing (SoCC)* (2014).

[22] GUNAWI, H. S., HAO, M., SUMINTO, R. O., LAKSONO, A., SATRIA, A. D., ADITYATAMA, J., AND ELIAZAR, K. J. Why does the cloud stop computing?: Lessons from hundreds of service outages. In *Proceedings of the 7th ACM Symposium on Cloud Computing (SoCC)* (2016).

[23] Gunawi, H. S., Suminto, R. O., Sears, R., Golliher, C., Sundararaman, S., Lin, X., Emami, T., Sheng, W., Bidokhti, N., McCaffrey, C., Grider, G., Fields, P. M., Harms, K., Ross, R. B., Jacobson, A., Ricci, R., Webb, K., Alvaro, P., Runesha, H. B., Hao, M., and Li, H. Fail-slow at scale: Evidence of hardware performance faults in large production systems. In *Proceedings of the 16th USENIX Conference on File and Storage Technologies (FAST)* (2018).

[24] Hao, M., Soundararajan, G., Kenchammana-Hosekote, D. R., Chien, A. A., and Gunawi, H. S. The Tail at Store - A Revelation from Millions of Hours of Disk and SSD Deployments. In *Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST)* (2016).

[25] Huang, P., Guo, C., Lorch, J. R., Zhou, L., and Dang, Y. Capturing and enhancing in situ system observability for failure detection. In *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (2018).

[26] Huang, R., Guo, C., Zhou, L., Lorch, J., Dang, Y., Chintalapati, M., and Yao, R. Gray failure: The achilles' heel of cloud-scale systems. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems (HotOS)* (2017).

[27] JEDEC. *Solid-State Drive (SSD) Requirements and Endurance Test Method.*, Sept. 2010.

[28] Jiang, W., Hu, C., Zhou, Y., and Kanevsky, A. Are disks the dominant contributor for storage failures? *ACM Transactions on Storage 4*, 3 (2008), 1–25.

[29] Liang, Y., Zhang, Y., Sivasubramaniam, A., Jette, M., and Sahoo, R. Bluegene/l failure analysis and prediction models. In *Proceedings of the 36th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (2006).

[30] Luo, Y., Ghose, S., Cai, Y., Haratsch, E. F., and Mutlu, O. Heatwatch: Improving 3d nand flash memory device reliability by exploiting self-recovery and temperature awareness. In *Proceedings of the 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)* (2018).

[31] Mahdisoltani, F., Stefanovici, I., and Schroeder, B. Proactive error prediction to improve storage system reliability. In *Proceedings of the 2017 USENIX Annual Technical Conference (ATC)* (2017).

[32] Meza, J., Wu, Q., Kumar, S., and Mutlu, O. A Large-Scale Study of Flash Memory Failures in the Field. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)* (2015).

[33] Narayanan, I., Wang, D., Jeon, M., Sharma, B., Caulfield, L., Sivasubramaniam, A., Cutler, B., Liu, J., Khessib, B. M., and Vaid, K. SSD Failures in Datacenters - What? When? and Why? In *Proceedings of the 9th ACM International on Systems and Storage Conference (SYSTOR)* (2016).

[34] Papandreou, N., Parnell, T., Pozidis, H., Mittelholzer, T., Eleftheriou, E., Camp, C., Griffin, T., Tressler, G., and Walls, A. Using adaptive read voltage thresholds to enhance the reliability of mlc nand flash memory systems. In *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI (GLSVLSI)* (2014).

[35] Pinheiro, E., Weber, W.-D., and Barroso, L. A. Failure Trends in a Large Disk Drive Population. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST)* (2007).

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

[37] Schroeder, B., and Gibson, G. A. Disk Failures in the Real World - What Does an MTTF of 1, 000, 000 Hours Mean to You? In *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST)* (2007).

[38] Schroeder, B., Lagisetty, R., and Merchant, A. Flash Reliability Production - The Expected and the Unexpected. In *Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST)* (2016).

[39] Shvachko, K., Kuang, H., Radia, S., and Chansler, R. The hadoop distributed file system. In *Proceedings of the IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)* (2010).

[40] Sun Microsystems. *Sun StorEdge 3000 Family Installation, Operation, and Service Manual.*

[41] Verbitski, A., Gupta, A., Saha, D., Brahmadesam, M., Gupta, K., Mittal, R., Krishnamurthy, S., Maurice, S., Kharatishvili, T., and Bao, X. Amazon aurora: Design considerations for high throughput cloud-native relational databases. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD)* (2017).

[42] VISHWANATH, K., AND NAGAPPAN, N. Characterizing cloud computing hardware reliability. In *Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC)* (2010).

[43] XU, E., ZHENG, M., QIN, F., WU, J., AND XU, Y. Understanding SSD reliability in large-scale cloud systems. In *Proceedings of the 3rd IEEE/ACM International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems (PDSW-DISCS)* (2018).

[44] ZHENG, M., TUCEK, J., QIN, F., AND LILLIBRIDGE, M. Understanding the robustness of ssds under power fault. In *Proceedings of the 11th USENIX Conference on File and Storage Technologies (FAST)* (2013).

[45] ZHENG, M., TUCEK, J., QIN, F., LILLIBRIDGE, M., ZHAO, B. W., AND YANG, E. S. Reliability analysis of ssds under power fault. *ACM Transactions on Computer Systems (TOCS) 34*, 4 (2017), 10:1–10:28.