Understanding SSD Reliability in Large-Scale Cloud Systems

Erci Xu
Dept. of Computer Science and Engineering
Ohio State University
xu.1556@osu.edu

Mai Zheng
Dept. of Electrical and Computer Engineering
Iowa State University
mai@iastate.edu

Feng Qin
Dept. of Computer Science and Engineering
Ohio State University
qin.34@osu.edu

Jiesheng Wu
Alibaba
Alibaba Inc.

Yikang Xu
Alibaba
Alibaba Inc.

***Abstract*—Modern datacenters increasingly use flash-based solid state drives (SSDs) for high performance and low energy cost. However, SSDs introduce more complex failure modes compared to traditional hard disks. While great efforts have been made to understand the reliability of SSDs itself, it remains unclear how the device-level errors may affect upper layers, or how the services running on top of the storage stack may affect the SSDs.**

In this paper, we take a holistic view to examine the reliability of SSD-based storage systems in Alibabas datacenters, which covers about half-million SSDs under representative cloud services over three years. By vertically analyzing the error events across three layers (i.e., SSDs, OS, and the distributed file system), we discover a number of interesting correlations. For example, SSDs with UltraDMA CRC errors, while seems benign at the device level, are nearly 3 times more likely to lead to OS-level error events. As another example, different cloud services may lead to different usage patterns of SSDs, some of which are detrimental from the devices perspective.

***Index Terms*—SSD, storage system, fault-tolerance**

I. Introduction

Flash-based solid state drives (SSDs) have become an indispensable component in modern datacenters due to its superior performance and low power draw [1]. Various applications, such as database transactions [2], social network [3] and online shopping [4], have been empowered by SSDs-based large-scale storage systems deployed in datacenters around the world. Therefore, the reliability of such storage systems is critically important.

However, maintaining high reliability in SSDs-based storage systems remains challenging. First, at the device level, SSDs can experience multiple types of unique errors [5]–[8], which may not be completely remedied by Error-Correcting Code (ECC) [9], [10] or redundancy provided at the higher levels [11]. Second, at the higher levels, the storage stack can suffer from other events such as incorrect configuration, wrong maneuvering and software bugs [12]–[15]. Moreover, the potential correlations among error events across different levels remains unclear.

Great efforts have been made to understand the reliability of SSDs itself [16]–[19]. For example, Schroeder et al. [18] study the errors of flash chips and SSDs and discover interesting correlations between errors and other factors (e.g., age, wear, lithography). Hao et al. [19] 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, they do not directly reveal how the device-level behavior may affect the system as a whole.

In addition, studies on hard disk drives (HDDs) based storage systems are also abundant [20]–[24]. Apart from understanding HDD errors in the field [20]–[22], researchers have analyzed the failures in the vertical stack of storage systems [23], revealing the correlation between HDD errors and upper-level system failures [24]. However, since SSDs-based systems is different from HDD-based systems (e.g., NAND program errors, trim support throughout the kernel), their studies and findings may not be directly applicable to SSD-based storage stack.

In this paper, we take a holistic view to analyze the error events in large-scale SSD-based storage systems. Our study covers approximately 450,000 enterprise SSDs from three vendors spanning over three year's deployment. We collect error events from three levels: (1) at the device level, SMART [25] logs drive status; (2) at OS level, the kernel and monitoring software logs error events such as missing device or buffer IO errors; (3) at distributed file system (DFS) level, we collect Pangu[1] Master's and Servers' logs, which includes information about a failed IO request in the DFS. With these abundant logs collected, we further analyze their correlations vertically in the context of the entire storage stack. We summarize our preliminary findings as follows:

- **Over 20% of OS-level error events are caused by incorrect manual operations.** This finding suggests that

[1]Pangu is the distributed file system deployed at the datacenters in our study

Model	Capacity	Lith.	Age	Rationale
1-**B**	480GB	20nm	2-3yrs	**B**aseline
1-**C**	800GB	20nm	2-3yrs	**C**apacity
1-**L**	480GB	16nm	1-2yrs	**L**ith.
2-**V**	480GB	20nm	2-3yrs	**V**endor
3-**V**	480GB	20nm	1-2yrs	**V**endor

TABLE I: **SSD Models**. ***Lith.****: Lithography; Each model is named as (Vendor id)-Rationale.*

Services	SSD Models	Setups	Function
Block	1-B,C,L,2-V,3-V	Hy/Mul	Pers/Jour
NoSQL	1-B,L, 2-V,3-V	Hy/Mul	Pers/Jour
Big Data	1-B,C, 2-V	Single	Temp

TABLE II: **Cloud Services.** ***Hy****: Hybrid Setup,* ***Mul****: Multiple Setup,* ***Pers****: Persistent storage,* ***Jour****: Journaling,* ***Temp****: Temporarily storing intermediate data.*

more intelligent administrative automation is highly desired for achieving high reliability. In practice, we deploy a new strategy, called OIOP (One Interface One Purpose), which successfully reduce the failure rates of "Wrong Slot", a dominant type of human errors.

- **Certain cloud services may cause unbalanced usage of SSDs.** In our study, around 5% to 10% SSDs are under unbalanced usage due to improper data placement and IO pattern introduced at the higher levels, which may lead to more error events at the device level and eventually affect the system reliability.
- **SSDs heavily affected by UCRC (UltraDMA CRC) errors are 2.7x more likely to lead to OS-level error events and root cause of error can be faulty interconnection.** UCRC error is often assumed to be benign since it is automatically solved by simple retry at the device level. However, our study shows that such device-level event may be eventually affect the system reliability.

In this study, though we focus on analyzing datasets exclusively from our internal storage systems, we are convinced our analysis and findings should be able to reach a broader audience beyond Alibaba's engineers and administrators. First, our fleet is using commercial and standardized hardwares including SSDs, servers and interconnection components. Many products are also deployed in other data centers serving same or similar functionalities. Hence, administrators from other data centers may also encounter same or similar issues. Second, our software design shares general similarities with other large-scale storage system such as HDFS and Google File System. Software engineers and designers may adapt our findings to avoid certain fault-tolerance vulnerabilities and build stronger systems.

II. System Architecture and Datasets

A. System Architecture

Figure 1 shows the architecture of the SSD-based storage system used in this study. The dotted squares highlight the device, OS, and DFS levels.

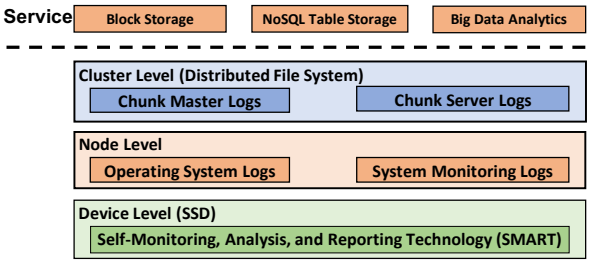

Fig. 1: **Storage System Architecture**. *The each dotted square indicates the scope of a level within the storage stack*

The system includes around 450,000 SSDs spanning three years of deployment. As shown in Table I, the SSDs cover a spectrum of variability in terms of vendors, sizes, lithographies and ages, which enables us to isolate factors contributing to the errors in the storage stack.

Each node can have 1 (Single), 2 (Hybrid) or 12 (Multiple) SSDs. SSDs in Single setup are used for storing temporary data. SSDs in Hybrid setup are used for journaling incoming writes. SSDs in Multiple setup are used for persistent storage. A rack can have 16 to 48 nodes with the same setup. Each service owns an exclusive set of Pangu clusters and each Pangu cluster includes 12 to 18 racks. The system supports three types of cloud services including block storage, NoSQL storage and big data analytics as shown in Table II.

B. Datasets

Table III lists all events we collect from each level in this study and their definitions. Note that the events can be collected with different frequencies in different granularity (last two columns).

C. Workload

In this study, we study three major services in our production systems. NoSQL service is Key-Value storing system. Block Service offers users with virtual Logical Block Address (LBA) where user can treat the LBA as a block device. The BigData Service provides analytic workloads similar to Map Reduce where users' data are stored in an appended-only manner.

D. Error Checking and Correcting

To better understand our results in later sections, we further explain three important device-level errors and their checking logic using Figure 2.

End-to-End (E2E) Checking: E2E checking is a data protection mechanism that offers host-end to device-end data integrity checking. When a host reads a piece of data from the device that fails the E2E parity checking, the SSD will report this event as an E2E error in SMART. Note that host-side software may correct an E2E error.

UDMA CRC Checking: This mechanism verifies the correctness of data transmission between the device and the host in the Ultra DMA (UDMA) transfer process. If the data fails

Level	Event	Definition	Freq.	Gran.
DFS	Read Error	DFS cannot read the requested data on time	Event	Server
	Write Erro	DFS cannot finish writing with replication on time		
OS	Buffer IO Error	A failed read/write from file system to SSD	Event	SSD
	Media Error	Software detected actual data corruption		
	File System Unmountable	Unable to load the file system on a SSD		
	Drive Missing	OS unable to find a plugged SSD		
	Wrong Slot	SSD has been plugged to the Wrong SATA slot		
Device	Host Read	Total amount of LBA read from the SSD	Daily	SSD
	Host Write	Total amount of LBA write from the SSD		
	Program Error	Total # of errors in NAND write operations		
	Raw Bit Error Rate	Total bits corrupted divided by total bits read		
	End-to-End Error	Total # of parity check failures between interfaces		
	Uncorrectable Error	Total # of data corruption beyond ECC's ability		
	UDMA CRC Error	Total # of CRC check failures during Ultra-DMA(UDMA)		

TABLE III: **Events Collected in the Target Storage System.** ***Freq.****: Frequency, event logs can be updated daily ("Daily") or upon new events ("Event");* ***Gran.****: Granularity, the event can be traced back to a server or an SSD.*

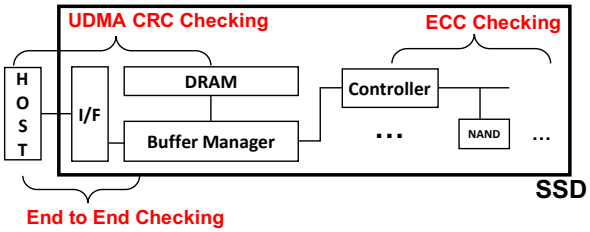

Fig. 2: **SSD Error Checking and Correcting Mechanisms**. The braces mark the coverage of checking/correction logics

the checksum test, the SSD records the event in SMART and trigger a retry. CRC checking only verifies the correctness of data transmission.

ECC Checking/Correcting: At a write operation, the controller encodes an ECC (e.g., BCH [10] or LDPC [9]) before storing data in the NAND chips. Failing to write the data with the ECC can result in a program error, which will be reported by SMART. At a read operation, if the data read does not pass the ECC check, the controller will report this as a raw bit error and start to correct the data. Failing to correct the data would be recorded as an uncorrectable error in SMART.

III. Preliminary Findings & Implications

A. Impact of Human Errors

We summarize the characteristics of the OS-level events in Table IV. For each type, we calculate the percentage of occurrences and the fractions of affected drives for each SSD model. Also, we list the corresponding repairing approaches (2nd to the last column), which are adopted by system administrators one by one from top to bottom until the issue is resolved. Note that the logs only record the working fix. For instance, after observing a Buffer IO Error, the file system volume on the device is first checked and repaired using the local file system checker FSCK (e.g., e2fsck for Ext4). If FSCK fails to repair the file system, the administrator will replace the drive. Based on the working repair method, administrators also derive the most likely root causes of the events (last column).

Among nearly 10,000 OS-level events, we do not observe a dominant type of events or a dominant SSD model which experiences significantly more error events. However, by analyzing the root causes in more details, we have the following finding:

Finding 1: Over 20% of OS-level error events are caused by incorrect manual operations. As shown in Table IV, more than 20% of events are caused by incorrect manual operations. They include 20.0% incidents of plugging drives to the wrong slots and 0.4% software configuration errors that cause file system unmounted. Such error events can be manifested as failure at DFS level. For example, a "Wrong Slot" event on the master node of the DFS may prevent the whole DFS from functioning properly.

Although it may be part of human nature to err, we believe such human mistakes can be reduced by more considerate hardware/software co-design. For example, in the case of plugging drives to the wrong slots, a hardware slot is mapped to a mount point in the file system tree in a strict 1-to-1 fashion, which can easily lead to errors if an unmatched drive is plugged in and used by the system. It is possible to add anther layer of indirection or using an alternative mapping between hardware slots and system software to break the strong dependency and enable more intelligent management of drive replacement. In Alibaba, we deploy a simple but useful strategy, called One Interface One Purpose(OIOP). In OIOP, for each setup, each software functionality, such as *Persistence* or *Journaling* as shown in Table II, is binded to a specific SSD hardware interface including traditional SATA and latest U.2/M.2. For instance, in the hybrid setup, the *Journaling* SSDs are using the M.2 interface while the *Persistence* SSDs use the normal SATA interface. This simple hack successfully reduces the "Wrong Slot" failure rates. Our log shows that, with OIOP, only 3 "Wrong Slot" occurrences in the fleet of 100K SSDs of 6 months deployment where the corresponding average number is 47 previously.

B. Impact of Cloud Services

Host Read and Host Write are the most direct metrics about drive usage. In Table V, we calculate the hourly average value

Event Type	% of Fix in Each Type	Fractions of Affected Drives by Model (in ‰)					Repairing Procedures	Most Likely Root Causes
		1-B	1-C	1-L	2-V	3-V		
Drive Missing	20.1%	3.76	3.84	2.54	1.92	3.94	Resetting Node	Software
	13.9%	2.63	0.66	0.84	2.83	2.90	Replacing Cable	Unstable Connection
	2.3%	0.48	0.40	0.13	0.47	0.21	Replacing SSD	Failed Device
Unmoutable File System	0.4%	0.06	0.08	0.02	0.05	0.21	Resetting	Software Config
	3.0%	0.42	0.44	0.22	1.34	0.90	Drive Repartitioning	File System Corruption
	4.0%	0.80	0.53	0.18	1.51	0.93	Replacing SSD	Failed Device
Buffer I/O Error	13.8%	3.37	1.29	0.91	3.35	1.02	FSCK checking	Data Inconsistency
	1.8%	0.36	0.05	0.15	0.71	0.19	Replacing SSD	Failed Device
Media Error	6.0%	0.42	2.55	0.51	0.88	0.31	Data Checking	Data Corruption
	13.9%	3.00	3.04	2.33	4.67	0.93	Replacing SSD	Failed Device
Wrong Slot	20.0%	4.36	4.10	1.82	5.32	1.35	Replugging SSD	Human Error

TABLE IV: **Summary of OS-Level Events** . *Repairing procedures are tried out one by one from top to bottom in each event category. The logs only record the working fix. After the fix, the root cause would be derived empirically and logged as well.*

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

TABLE V: **Host Read and Host Write Comparison between Services**. ***CV****: Coefficient of Variance, the ratio of standard deviation to mean.*

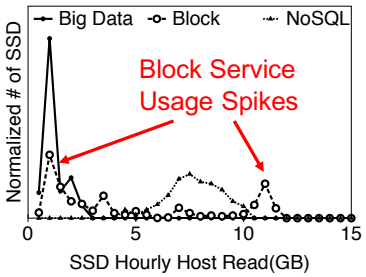

Fig. 3: **Distribution of Hourly Host Read for three services.***Each dot along the line indicates the cumulative amount of devices that fall into a range of 0.5GB/Hour. The arrows mark the two far-apart usage spikes under Block service.*

of Host Read and Host Write under each cloud service (row 2 to row 4). While it is expected that SSDs under different services may have different usage due to the different I/O patterns, we find that under one service, the usage of SSDs may also differ a lot.

Finding 2: Certain cloud services may cause unbalanced usage of SSDs. In Table V, we further measure the variability of the two metrics of the SSDs under the same service using the coefficient of variation (CV), the ratio of standard deviation to mean. Intuitively, a higher CV indicates that the hourly host read and hostly host write vary more across devices. We find that the CVs of the block storage service (35.5% for write, 24.9% for read) are much higher than those of the NoSQL storage service (3.2% for write, 6.2% for read) and the big data analytics service (1.8% for write, 3.7% for read), which implies that the usage of SSDs under the block storage service are much more unbalanced.

Figure 3 further illustrates the distribution of host read under the three services. Each dot in the line equals the cumulative count of SSDs that have hourly host read falls into a range along the X axis, with a step of 0.5GB/hr and starting from 0.5. The majority of SSDs under both NoSQL and Big Data Analytics services have similar values (i.e., one major spike in the corresponding curve). On the other hand, the SSDs under the block storage service shows diverse values (i.e., two spikes far apart) as marked in the figure. The distribution of host write is similar.

Such unbalanced usage of SSDs may lead to a number of consequences. For example, our analysis indicates that, on device level, overly used SSDs have 177.3% higher RBER and 68.4% more program errors which indicate that those SSDs are more likely to suffer from NAND read and write errors. Further, on OS level, we discover that overly used SSDs are 39.4% more likely to encounter “Drive Missing” failure and 261.5% more likely to encounter “Media Error”. Therefore, it is important to monitor the device-level events to identify the unbalanced usage and improve the load balancing.

To understand the root cause of the unbalancing, we look into the design of the block storage service. We find that the unbalanced usage is caused by two factors: the data placement policy and the user I/O pattern. First, to reduce the number of affected users in case a node crash, the Block Storage Service tends to map user’s logical blocks to SSDs on a limited number of nodes. Consequently, each node under the service hosts relatively few users’ data. Second, the I/O patterns of different users vary a lot, which leads to a diverse usage of SSDs under the service. Currently, we are redesigning the data layout of our distributed file system. By using a share-log structure system, users’ data are now more evenly allocated across SSDs. Our test cluster demonstrates that the usage difference would be less than 5% among drives.

C. Impact of “benign” SSD Errors

UCRC (UltraDMA CRC) errors occur when the data transmission between the device and the host goes wrong, which

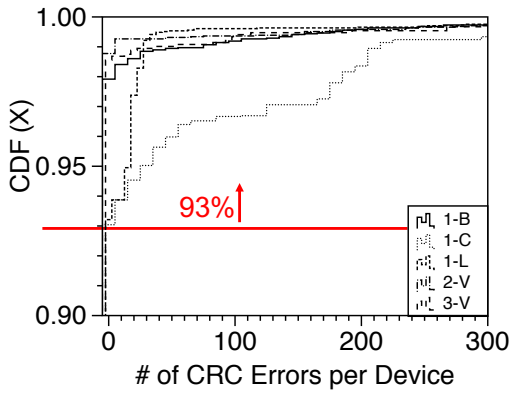

Fig. 4: **Concentration of UCRC Errors among different SSD models.** *UCRC errors are only generated within 7% of the total SSD population.*

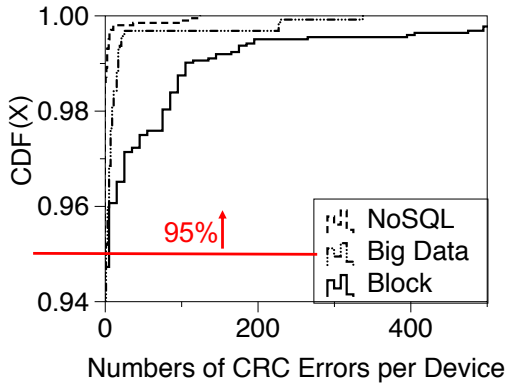

Fig. 5: **Concentration of UCRC Errors among different workloads.** *UCRC errors are only generated within 5% of the total SSD population.*

is often assumed to be transient (e.g., caused by transient environmental factors like unstable voltage) and benign (i.e., automatically solved by a simple retry). However, we find that this may not be true. In our datasets, the devices affected with UCRC errors are more likely in encountering OS level failures. For example, devices that suffered UCRC errors are having about two times higher failure rates in the "Drive Missing" category. To quantitatively measure the impact and analyze the corresponding root cause, we study UCRC errors by understanding its distribution, its correlation with other device level events and its correlation with OS level events.

Finding 3: UCRC error occurrences follow Zipf Law Distribution. To start with, we study how UCRC errors are distributed among different device models and workloads. As shown with the horizontal marker in the Figure 4 and Figure 5, almost all SSD UCRC errors are generated within 10% of total population regardless of its model or workload. To be specific, in the Figure 4, by plotting the Cumulative Distribution of CRC errors per device in each model. For example, for model

	1-B	1-C	1-L	2-V	3-V
Heavy	2.71%	0.87%	0.14%	1.79%	1.85%
Light	1.63%	1.39%	7.19%	0.26%	0.12%

TABLE VI: **Categorization of SSDs affected by UCRC Error** *Each percentage number indicates the fractions of a certain model SSD in the group, either Heavy or Light*

1-C (the lowest curve), 93% of devices do not report any UCRC errors. However, the top 1% of 1-C model can have as many as 200 UCRC errors per device.

Moreover, we observe that UCRC occurrences are not well-balanced among affected drives. Rather, from the visual inspection in the Figure 4 and Figure 5, the pattern of UCRC errors seems following a Zipf Law distribution, where a small group of SSDs is responsible for majority of event occurrences. To quantitatively verify this assumption, we apply the Kolmogorov-Smirnov test (K-S test) [26]. The K-S test is used to measure the distance between an observed CDF (i.e. UCRC distribution in our case) and an empirical CDF (i.e. the Zipf Law distribution). If the result of the K-S test, known as the *p value*, is smaller than a confidence threshold, we can accept the hypothesis that the two CDF follow the same distribution. In our experiments, we set confidence level to be 0.05, a common threshold, and we are able to fit each of the UCRC CDF distribution from five models and three workloads into a Zipf Law CDF with a *p value* less than 0.05. In short, UCRC occurrence follows Zipf Law distribution.

As Zipf Law is a discrete Pareto's Law (also known as Power Law), we are able to further categorize the UCRC affected drives by using the Pareto's Principle (i.e. the 80/20 rule). First, we set up three exclusive groups, *Heavy*, *Light* and *None*. Then, for each model, we rank the devices by its numbers of UCRC occurrences in descending order and place the devices into the *Heavy* group one by one until *Heavy* group accumulates no less than 80% of all UCRC errors. Then for devices which are affected by UCRC error but not in the *Heavy*, we place them in the *Light* and place drives which do not have UCRC errors in the *None*. Table VI demonstrates the results of grouping. In the table, each percentage presents the fractions of a certain model drives under *Heavy* or *Light*. We observe the ratio varies from model to model. For example, in 1-B, the majority of UCRC affected drives belong to the *Heavy* group. While in model 1-L, only 2% (i.e. 0.14% out of 7.33%) of UCRC affected drives is in *Heavy* group.

With the categorization, one question emerges: for SSDs in the *Heavy* group, whether their UCRC errors regularly generates or suddenly spurs. To answer this, we plot the monthly average UCRC error occurrences for devices in *Heavy* as shown in the Figure 6. The upper half of the Figure 6 demonstrates the average newly generated UCRC in the *Heavy* group in each month since device initial deployment. We observe a steady generation of UCRC errors in the *Heavy* group. As UCRC error is generated during data transmission (i.e. host read/write), we also plot the monthly average usage load of devices in the *Heavy* group as shown in the lower

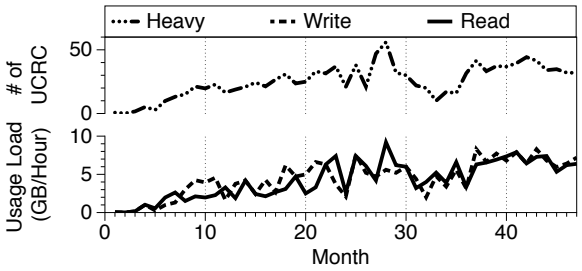

Fig. 6: **UCRC monthly occurrences in *Heavy* Group**. *The upper half shows the steady generation of UCRC errors. The lower half show corresponding average host read and write of these devices*

UCRC Group	Heavy	Light	None	All
Total	53	216	2942	3211
Resetting	2	74	1732	1808
Repl. Cable	41	98	1065	1204
Repl. SSD	10	44	145	199

TABLE VII: **Distribution of UCRC affected SSDs in "Drive Missing" Failure** *Row 2-5 shows the numbers of "Drive Missing" SSDs in different UCRC groups. All is the sum of Heavy,Light and None*

half of Figure 6. From visual inspection, there exist possible correlation between usage load and numbers of UCRC errors. However, it remains unknown whether the UCRC error is the byproduct of other device level errors or independently occurs.

In summary, we find that the UCRC error occurrence follow Zipf Law distribution. Less than 3% of the devices (i.e. the *Heavy* group) contributes more than 80% of all the UCRC errors. Such distribution is consistent among different models and workloads. By definition, UCRC errors indicate the transmission failure between device RAM and host RAM. While transient factors (e.g., voltage spikes) could cause transmission errors, the concentration among a few devices and repeated occurrence along time suggest that many UCRC errors may be caused by non-transient factors (e.g., faulty hardware). Such regular and steady trend suggests that we cannot simply ignore UCRC errors.

Finding 4: UCRC error occurrences are not correlated with other device level errors. To further understand the nature of UCRC errors, we conduct the the correlation study between UCRC error and other device error events. Our goal is to determine whether UCRC error correlates with other device level events or independently occurs. We use Spearman Rank Correlation Coefficient (SRCC) [27] to quantitatively measure the correlation between CRC Error and other events. The value of the SRCC metric ranges from +1 (most positively correlated) to -1 (most negatively correlated). A near-zero SRCC value implies that there is little correlation. ± 0.5 are the threshold of showing moderate correlation.

Figure 7 shows the SRCC between UCRC error and all other six types of events (i.e., Host Read, Host Write, RBER, Program Error, Uncorrectable Error and End-to-End Error) on different SSD models (i.e., 1-B to 3-V). Note that Host Read and Host Write are the usage of SSDs instead of errors. From the figure, we can see that there is positive correlation between UCRC error and Host Read/Host Write on all SSD models, which verify our assumption in Figure 6. Intuitively, as UCRC errors occur during the data transmission between the host and the device, more usage of SSDs naturally lead to more UCRC errors.

Moreover, we also observe correlation in RBER and Program Error as well. However, such correlation might be spurious as all error events can be the byproduct of SSD usage. For example, more Host Write can lead to more Program Errors and consequently more data transmission between device and host which can cause more UCRC Errors. Therefore we can not simply study the correlation without leveling the SSD usage. Hence, we select a subset of SSDs where their usage are about the same (i.e., 75TB for read and 50TB for write within a $\pm 3\%$ variance). This subset covers around 71.2% of the entire population, and enables eliminating the potential superficial correlation among events caused by different usage.

Figure 8 shows the SRCC with the normalized usage. We observe that the no device-level events can be considered moderately correlated with UCRC error. Moreover, for each type of errors, the SRCC values across different models vary from positive to negative, and thus there is no obvious correlation. In summary, we find out that UCRC error is positively correlated with device usage (host read/write) but not correlated with other device level error events.

Finding 5: UCRC is NOT necessarily a *benign* error in the long run. For UCRC error, as plotted in the Figure 9, the *Heavy* and *Light* have higher failure rates than *None* in every category. The biggest difference can be seen in the Drive Missing failure where the likelihood is 2.7x times higher between the *Heavy* and the *None*. The much higher likelihood in failure rate indicates that while UCRC error does not impose an immediate threat, such as a data corruption, it may latently signal the device status is unhealthy.

To further analyze possible root causes of the difference, we first cross reference the repair logs with device level logs to calculate how many SSDs that encountered "Drive Missing" are also affected with UCRC errors. As shown in the first row in the Table VII, there are three exclusive UCRC groups *Heavy*, *Light* and *None* plus the *All* which is the sum of three groups. In each group, we can see the numbers of SSDs that have been repaired by different repairing procedures. For example, in the *Heavy* group, row 3 shows 2 SSDs, which encountered "Drive Missing" failure, have been effectively repaired by using "Resetting".

From the table, we discover that most (i.e. 41 out of 53, 77.4%) SSDs in the *Heavy* group have been successfully repaired by replacing the interconnection cable. However, only 45.4% of SSDs in the *Light* and 37.5% of SSDs in the *None* have been repaired by replacing cable. As UCRC is transmission error, the difference leads to a hypothesis: the

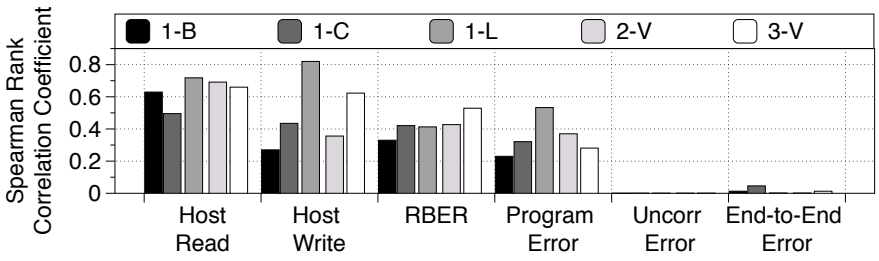

Fig. 7: **Spearman Rank Correlation Coefficient (SRCC) between UCRC Errors and device level events**. *Value ranges from +1 to -1 which indicates strong positive correlation to strong negative correlation. The horizontal markers are threshold of moderate correlation.*

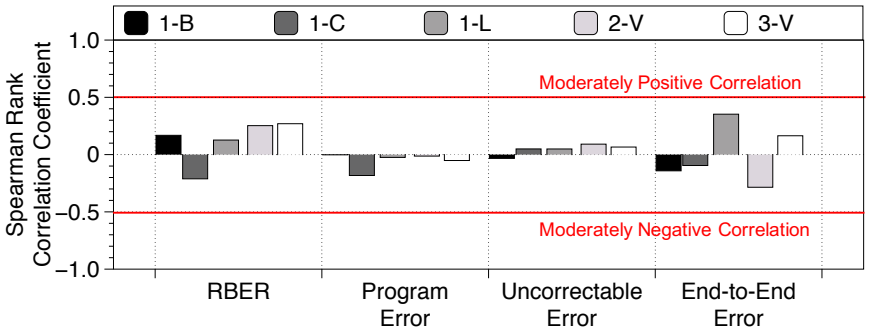

Fig. 8: **Spearman Rank Correlation Coefficient (SRCC) between UCRC Errors and other error events after leveling SSD usage**. *Value ranges from +1 to -1 which indicates strong positive correlation to strong negative correlation. The horizontal markers are threshold of moderate correlation.*

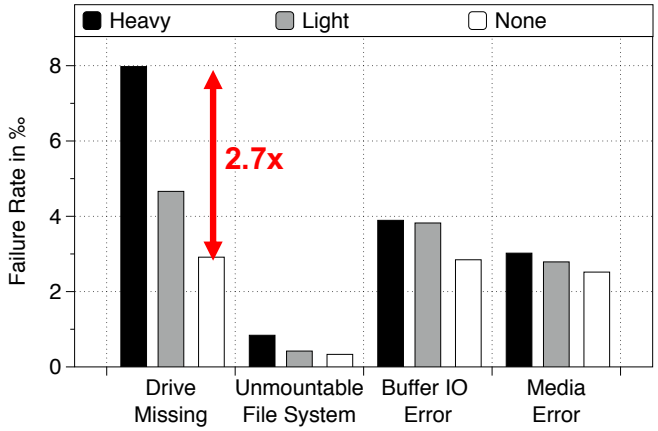

Fig. 9: **Failure Rate of Node Level Error Event in each SSD Group.** ***Heavy***: *most affected SSDs,* ***Light***: *SSDs have at least 1 UCRC error but not in the Heavy group,* ***None***: *SSDs without UCRC errors*

steady generation of UCRC errors can be caused by faulty interconnection between host and device.

To validate this hypothesis, we further study device level logs those repaired SSDs. As shown in Table VIII, we summarize the numbers of SSD that encounter new UCRC occurrences after repairing. For example, in Row 2, 74 and 2 SSDs belong to *Light* and *Heavy* before the Resetting Repairing. After the fix, 69 out of 76 (i.e. 74 + 2) have new UCRC occurrences and 2 out of 76 still have steady generation of UCRC errors.

	Light Before	Heavy Before	New UCRC	Steady UCRC
Resetting	74	2	69	2
Repl. Cable	98	41	7	4
Repl. SSD	44	10	3	0

TABLE VIII: **UCRC Re-occurrence after Repairing** *Column 2-3 shows numbers of SSDs that belong to the Light/Heavy group before the corresponding repairing. Column 4 represents numbers of SSDs generate new UCRC errors after repairing. Column 5 represents numbers of SSDs steadily generate new UCRC errors after repairing.*

By reading the table, we discover that, after replacing the cable (Row 3), most SSDs have not encountered further steady UCRC occurrences. After replacing both the SSD and cable (Row 4), no SSDs encounter steady UCRC errors afterwards. By correlating the effectiveness and the temporal relationship

of error occurrences, we believe that the steady occurrences of UCRC errors can be caused by faulty interconnection. The faulty parts can be both the cable between device and the host or internal interface component inside the SSD.

IV. Lessons and Discussion

In this section, we discuss some lessons learned during our study and hope they will be useful for different parties: system designers and data center administrators.

System Designers. First, when designing a SSD-based storage system, we need to pay great attention to the characteristics of SSDs. For example, while SSDs internally deploy wear leveling technology to combat the wear out effects of NAND chips, the unbalanced data placement policies in the storage system work against the purpose. Hence, the wear feature and higher price tag of SSD can turn a previously acceptable trade-off into a unpleasant deal. As a result, it would be wise to re-examine the relevant mechanisms and policies of storage systems to cater the features of SSDs.

Second, we need an efficient and comprehensive logging system that binds an upper level service request to all the lower level software and hardware components that serves this request. Recent studies work on efficient logging in distributed systems have made a good progress [28], they do not provide an efficient way to identify software/hardware faults once a failure occurs.

Third, while it could be rather difficult to completely eliminate human errors, it is possible for designers to adopt specific architecture hacks and automation strategies to make the system more foolproof.

System Administrators Software-based repairing procedures such as Resetting can efficiently amend the OS level device failure such as "Drive Missing". However, failures can be also caused faulty hardware components such as failing cables. In this case, simply following the *Software-First* routine procedures for repairing may not only waste time but also masking failure instead of fixing them. In short, understanding the root causes of failures is as important as fixing them.

V. Related Work

To the best of our knowledge, we are the first to examine the errors and failures in the vertical stack of SSD-based storage systems. Prior studies mainly focused on either the reliability of SSDs at the *device level* or failures in the *hard drive* based storage stack.

Three prior works focused on the low level errors of SSDs by analyzing the error distribution, the inter-error correlations and the root causes in the field. More specifically, Schroeder et al. [18] examined ten SSD models widely deployed in Google. They carefully studied the prevalence, correlation and implication of device level read and write errors. Meza et al. [16] conducted a similar study on SSDs in Facebook and made several key observations including the failure rate trend of SSDs, prevalence of read errors and correlations between SSD failures and host DRAM usage. Narayanan et al. [17] researched five enterprise SSD models deployed in the Microsoft data centers. They identified failure symptoms of SSDs and then ranked SSD SMART attributes that can be used to predict an imminent device failure. These works provide deep understanding of lower level errors of SSDs in the field. Complementary to these studies, our work focuses on a different perspective, i.e., understanding the errors and failures on the *entire stack* of storage systems. Our findings include distribution and correlation not only in at the device level but also in the upper levels.

Another group of related studies is the reliability analysis of hard drive based storage stack. These works studied storage systems from two perspectives. First, several works target the errors, failure trends and reliability at the disk level. They measured the occurrence of checksum errors [20], analyzed the correlation between SMART attributes and disk failures [21], and discussed the implications behind the Mean Time To Failure (MTTF) of hard drives. Other researchers took a different perspective by studying the correlation between disk errors and storage system failures at the higher level. Jiang et al. [23] observed that, apart from disk errors, protocols bugs and faulty physical interconnection can also incur storage system failures. Bairavasundaram et al. [24] investigated a specific type of disk errors, i.e., the latent sector errors, and studied their trend and the root causes. Additionally, they analyzed the correlation between such errors and the reliability of the storage system design. In summary, these works thoroughly reviewed the reliability issues in the hard drive based storage stack. However, SSD and hard drives are two intrinsically different devices with different corresponding hardware interfaces and supporting software. Their findings may not be directly applicable to SSD-based storage systems.

VI. Conclusion & Future Work

We have analyzed the error events across different layers of large-scale SSD-based storage system. Our findings reveal that the storage system stack can be viewed as an ecosystem. Erroneous behaviors from software/human/hardware can latently or explicitly affect the other levels and reduce the reliability of the system. Our analysis and findings can be helpful for multiple parties including storage system designers, data center administrators and hardware vendors.

In the future, we will further analyze the root causes, error propagation paths and correlations of more error events in the storage stack. It is expected such study will provide valuable insights to help system developers and administrators to rethink the system designs and operational procedures.

References

[1] D. G. Andersen and S. Swanson, "Rethinking Flash in the Data Center," *IEEE Micro*, vol. 30, no. 4, pp. 52–54, 2010.

[2] B. K. Debnath, S. Sengupta, and J. Li, "Flashstore: High throughput persistent key-value store," *PVLDB*, vol. 3, pp. 1414–1425, 2010.

[3] S. Dong, M. Callaghan, L. Galanis, D. Borthakur, T. Savor, and M. Strum, "Optimizing space amplification in rocksdb," in *CIDR*, 2017.

[4] A. Verbitski, A. Gupta, D. Saha, M. Brahmadesam, K. Gupta, R. Mittal, S. Krishnamurthy, S. Maurice, T. Kharatishvili, and X. Bao, "Amazon aurora: Design considerations for high throughput cloud-native relational databases," in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD '17. New York, NY, USA: ACM, 2017, pp. 1041–1052. [Online]. Available: http://doi.acm.org.proxy.lib.ohio-state.edu/10.1145/3035918.3056101

[5] Y. Cai, Y. Luo, E. F. Haratsch, K. Mai, and O. Mutlu, "Data retention in mlc nand flash memory: Characterization, optimization, and recovery," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2015, pp. 551–563.

[6] 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 *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2009, pp. 24–33.

[7] M. Zheng, J. Tucek, F. Qin, and M. Lillibridge, "Understanding the robustness of ssds under power fault," in *Presented as part of the 11th USENIX Conference on File and Storage Technologies (FAST 13)*. San Jose, CA: USENIX, 2013, pp. 271–284. [Online]. Available: https://www.usenix.org/conference/fast13/technical-sessions/presentation/zheng

[8] L. M. Grupp, J. D. Davis, and S. Swanson, "The bleak future of nand flash memory," in *Proceedings of the 10th USENIX Conference on File and Storage Technologies*, ser. FAST'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 2–2. [Online]. Available: http://dl.acm.org.proxy.lib.ohio-state.edu/citation.cfm?id=2208461.2208463

[9] S. Lin and D. J. Costello, *Error Control Coding, Second Edition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2004.

[10] K. Zhao, W. Zhao, H. Sun, X. Zhang, N. Zheng, and T. Zhang, "Ldpc-in-ssd: Making advanced error correction codes work effectively in solid state drives," in *Presented as part of the 11th USENIX Conference on File and Storage Technologies (FAST 13)*. San Jose, CA: USENIX, 2013, pp. 243–256. [Online]. Available: https://www.usenix.org/conference/fast13/technical-sessions/presentation/zhao

[11] A. Ganesan, R. Alagappan, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Redundancy does not imply fault tolerance: Analysis of distributed storage reactions to single errors and corruptions," in *15th USENIX Conference on File and Storage Technologies (FAST 17)*. Santa Clara, CA: USENIX Association, 2017, pp. 149–166. [Online]. Available: https://www.usenix.org/conference/fast17/technical-sessions/presentation/ganesan

[12] H. S. Gunawi, M. Hao, R. O. Suminto, A. Laksono, A. D. Satria, J. Adityatama, and K. J. Eliazar, "Why does the cloud stop computing?: Lessons from hundreds of service outages," in *Proceedings of the Seventh ACM Symposium on Cloud Computing*, ser. SoCC '16. New York, NY, USA: ACM, 2016, pp. 1–16. [Online]. Available: http://doi.acm.org/10.1145/2987550.2987583

[13] P. Huang, W. J. Bolosky, A. Singh, and Y. Zhou, "Confvalley: A systematic configuration validation framework for cloud services," in *Proceedings of the Tenth European Conference on Computer Systems*, ser. EuroSys '15. New York, NY, USA: ACM, 2015, pp. 19:1–19:16. [Online]. Available: http://doi.acm.org/10.1145/2741948.2741963

[14] T. Xu, X. Jin, P. Huang, Y. Zhou, S. Lu, L. Jin, and S. Pasupathy, "Early detection of configuration errors to reduce failure damage," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, 2016, pp. 619–634. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/xu

[15] T. Xu, J. Zhang, P. Huang, J. Zheng, T. Sheng, D. Yuan, Y. Zhou, and S. Pasupathy, "Do not blame users for misconfigurations," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, ser. SOSP '13. New York, NY, USA: ACM, 2013, pp. 244–259. [Online]. Available: http://doi.acm.org/10.1145/2517349.2522727

[16] J. Meza, Q. Wu, S. Kumar, and O. Mutlu, "A Large-Scale Study of Flash Memory Failures in the Field." *SIGMETRICS*, 2015.

[17] I. Narayanan, D. Wang, M. Jeon, B. Sharma, L. Caulfield, A. Sivasubramaniam, B. Cutler, J. Liu, B. M. Khessib, and K. Vaid, "SSD Failures in Datacenters - What? When? and Why?" *SYSTOR*, 2016.

[18] B. Schroeder, R. Lagisetty, and A. Merchant, "Flash Reliability in Production - The Expected and the Unexpected." *FAST*, 2016.

[19] M. Hao, G. Soundararajan, D. R. Kenchammana-Hosekote, A. A. Chien, and H. S. Gunawi, "The Tail at Store - A Revelation from Millions of Hours of Disk and SSD Deployments." *FAST*, 2016.

[20] L. N. Bairavasundaram, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, G. R. Goodson, and B. Schroeder, "An analysis of data corruption in the storage stack," *Trans. Storage*, vol. 4, no. 3, pp. 8:1–8:28, Nov. 2008. [Online]. Available: http://doi.acm.org/10.1145/1416944.1416947

[21] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure Trends in a Large Disk Drive Population." *FAST*, 2007.

[22] B. Schroeder and G. A. Gibson, "Disk Failures in the Real World - What Does an MTTF of 1, 000, 000 Hours Mean to You?" *FAST*, 2007.

[23] W. Jiang, C. Hu, Y. Zhou, and A. Kanevsky, "Are disks the dominant contributor for storage failures?" *ACM Transactions on Storage*, vol. 4, no. 3, pp. 1–25, Nov. 2008.

[24] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler, "An analysis of latent sector errors in disk drives," in *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '07. New York, NY, USA: ACM, 2007, pp. 289–300. [Online]. Available: http://doi.acm.org/10.1145/1254882.1254917

[25] *S.M.A.R.T. attributes*, 2018, https://en.wikipedia.org/wiki/S.M.A.R.T.

[26] M. A. Stephens, "Edf statistics for goodness of fit and some comparisons," *Journal of the American Statistical Association*, vol. 69, no. 347, pp. 730–737, 1974. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1974.10480196

[27] G. Corder and D. Foreman, *Nonparametric Statistics: A Step-by-Step Approach*. Wiley, 2014. [Online]. Available: https://books.google.com/books?id=CIxgAwAAQBAJ

[28] X. Zhao, K. Rodrigues, Y. Luo, M. Stumm, D. Yuan, and Y. Zhou, "Log20: Fully automated optimal placement of log printing statements under specified overhead threshold," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA: ACM, 2017, pp. 565–581. [Online]. Available: http://doi.acm.org/10.1145/3132747.3132778