A Large-Scale Study of Flash Memory Failures in the Field

Justin Meza
Carnegie Mellon University
meza@cmu.edu

Qiang Wu
Facebook, Inc.
qwu@fb.com

Sanjeev Kumar
Facebook, Inc.
skumar@fb.com

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu

ABSTRACT

Servers use flash memory based solid state drives (SSDs) as a high-performance alternative to hard disk drives to store persistent data. Unfortunately, recent increases in flash density have also brought about decreases in chip-level reliability. In a data center environment, flash-based SSD failures can lead to downtime and, in the worst case, data loss. As a result, it is important to understand flash memory reliability characteristics over flash lifetime in a realistic production data center environment running modern applications and system software.

This paper presents the first large-scale study of flash-based SSD reliability in the field. We analyze data collected across a majority of flash-based solid state drives at Facebook data centers over nearly four years and many millions of operational hours in order to understand failure properties and trends of flash-based SSDs. Our study considers a variety of SSD characteristics, including: the amount of data written to and read from flash chips; how data is mapped within the SSD address space; the amount of data copied, erased, and discarded by the flash controller; and flash board temperature and bus power.

Based on our field analysis of how flash memory errors manifest when running modern workloads on modern SSDs, this paper is the first to make several major observations: (1) *SSD* failure rates do *not* increase monotonically with *flash chip* wear; instead they go through several distinct periods corresponding to how failures emerge and are subsequently detected, (2) the effects of read disturbance errors are *not* prevalent in the field, (3) sparse logical data layout across an SSD's physical address space (e.g., non-contiguous data), as measured by the amount of metadata required to track logical address translations stored in an SSD-internal DRAM buffer, can greatly affect SSD failure rate, (4) higher temperatures lead to higher failure rates, but techniques that throttle SSD operation appear to greatly *reduce* the negative reliability impact of higher temperatures, and (5) data written by the operating system to flash-based SSDs does *not* always accurately indicate the amount of wear induced on flash cells due to optimizations in the SSD controller and buffering employed in the system software. We hope that the findings of this first large-scale flash memory reliability study can inspire others to develop other publicly-available analyses and novel flash reliability solutions.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s). Copyright is held by the author/owner(s).

SIGMETRICS'15, June 15–19, 2015, Portland, OR, USA.
ACM 978-1-4503-3486-0/15/06.
http://dx.doi.org/10.1145/2745844.2745848.

Categories and Subject Descriptors

B.3.4. [**Hardware**]: Memory Structures—*Reliability, Testing, and Fault-Tolerance*

Keywords

flash memory; reliability; warehouse-scale data centers

1. INTRODUCTION

Servers use flash memory for persistent storage due to the low access latency of flash chips compared to hard disk drives. Historically, flash capacity has lagged behind hard disk drive capacity, limiting the use of flash memory. In the past decade, however, advances in NAND flash memory technology have increased flash capacity by more than 1000×. This rapid increase in flash capacity has brought both an increase in flash memory use and a decrease in flash memory reliability. For example, the number of times that a cell can be reliably programmed and erased before wearing out and failing dropped from 10,000 times for 50 nm cells to only 2,000 times for 20 nm cells [28]. This trend is expected to continue for newer generations of flash memory. Therefore, if we want to improve the operational lifetime and reliability of flash memory-based devices, we must first fully understand their failure characteristics.

In the past, a large body of prior work examined the failure characteristics of flash cells in controlled environments using small numbers e.g., tens) of raw flash chips (e.g., [36, 23, 21, 27, 22, 25, 16, 33, 14, 5, 18, 4, 24, 40, 41, 26, 31, 30, 37, 6, 11, 10, 7, 13, 9, 8, 12, 20]). This work quantified a variety of flash cell failure modes and formed the basis of the community's understanding of flash cell reliability. Yet prior work was limited in its analysis because these studies: (1) were conducted on small numbers of raw flash chips accessed in adversarial manners over short amounts of time, (2) did not examine failures when using real applications running on modern servers and instead used synthetic access patterns, and (3) did not account for the storage software stack that real applications need to go through to access flash memories. Such conditions assumed in these prior studies are substantially different from those experienced by flash-based SSDs in large-scale installations in the field. In such large-scale systems: (1) real applications access flash-based SSDs in different ways over a time span of *years*, (2) applications access SSDs via the storage software stack, which employs various amounts of buffering and hence affects the access pattern seen by the flash chips, (3) flash-based SSDs employ aggressive techniques to reduce device wear and to correct errors, (4) factors in platform design, including how many SSDs are present in a node, can affect the access patterns to SSDs, (5) there can be significant variation in reliability due to the existence of a very large number of SSDs and flash chips. All of these real-world conditions present in large-scale

systems likely influence the observed reliability characteristics and trends of flash-based SSDs.

Our goal is to understand the nature of flash-based solid state drive failures in the field. To this end, we provide the *first comprehensive study of flash memory reliability trends in a large-scale production data center environment.* Our study is based on data collected from a majority of flash-based SSDs in Facebook's server fleet, with operational lifetimes extending over nearly four years and comprising many millions of device-days of usage. We analyze SSDs of different capacities and data transfer technologies, with a focus on understanding how various *internal factors* (i.e., those related to how the device operates) and *external factors* (i.e., those related to the environment the SSD operates in) affect flash-based SSD reliability.

Our main contribution is a rigorous characterization of the reliability trends of flash-based SSDs in the field. We observe several reliability trends for flash-based SSDs that have not been discussed before in prior works:

1. Flash-based SSDs do *not* fail at a monotonically increasing rate with wear. They instead go through *several* distinct reliability periods corresponding to how failures emerge and are subsequently detected. Unlike the monotonically-increasing failure trends for *individual flash chips* [6, 12, 20], across a large number of *flash-based SSDs* we observe *early detection*, *early failure*, *usable life*, and *wearout* periods, whose failure rates can vary by up to 81.7%.
2. Read disturbance errors (i.e., errors caused in neighboring pages due to a read [11, 8]) are *not* prevalent in the field. SSDs that have read the most data do not show a statistically significant increase in failure rates.
3. Sparse logical data layout across an SSD's physical address space (e.g., non-contiguous data), as measured by the amount of SSD-internal DRAM buffer usage for flash translation layer metadata, greatly affects device failure rate. In addition, dense logical data layout (e.g., contiguous data) with adversarial access patterns (e.g., small, sparse writes) also negatively affects SSD reliability.
4. Higher temperatures lead to higher failure rates, but techniques used in modern SSDs that throttle SSD operation (and, consequently, the amount of data written to flash chips) appear to greatly reduce the reliability impact of higher temperatures by reducing access rates to the raw flash chips.
5. The amount of data written by the operating system to *an SSD* is not the same as the amount of data that is eventually written to flash *cells*. This is due to system-level buffering and wear reduction techniques employed in the storage software stack and the SSDs. It is important that system-level flash reliability studies account for these effects.

2. BACKGROUND AND METHODOLOGY

2.1 Server Flash Device Architecture

We examine a set of flash-based SSDs similar to those employed commonly in server hardware and available from companies such as Fusion-io, Hitachi, Intel, OCZ, Seagate, and Virident. These SSDs connect to the server using the PCI Express (PCIe) bus. The SSD controller coordinates the transfer of data to and from the host and performs various functions designed to improve the performance and reliability of flash memory. Figure 1 provides an illustration of the architecture of a server SSD.

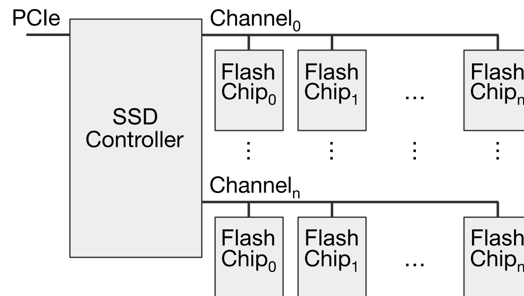

Figure 1: Server SSD architecture.

In order to achieve high bandwidth and low access latency – despite potentially long chip read and write latencies – these SSDs typically employ many channels that can be operated in parallel, with a number of chips connected to each channel. By mapping data across the channels, multiple channels can provide data in parallel, helping to ensure that the SSD can service data at close to the bandwidth of the PCIe connection.

In addition to orchestrating flash chip access, the SSD controller logic also plays a role in managing the reliability of the flash memory chips. Since flash cells wear out after a certain number of writes, SSD controllers must perform *wear leveling* in order to distribute cell wear more evenly across the cells in each flash chip. To do this, the SSD controller determines when and where pages of flash data should be erased or copied and maintains a mapping of logical addresses to physical locations, known as the flash translation layer (FTL) [15]. FTL data is stored in a DRAM buffer and is managed by the SSD controller. When blocks of flash cells are deemed unreliable for further use, the SSD controller *discards* them to avoid the risk of encountering an uncorrectable error on them.

2.2 Flash Errors and Their Handling

As prior works have shown in controlled environments, the repeated wear caused by reading and writing data from/to flash cells degrades their reliability over time, leading to relatively high error rates [36, 23, 21, 27, 22, 25, 16, 33, 14, 5, 18, 4, 24, 40, 41, 26, 31, 30, 37, 6, 10, 7, 9, 8, 12, 20]. To protect against such errors, additional data is stored in each channel to maintain various error correcting codes (ECC) computed over data in in the channel. These codes must be sufficiently strong to protect against the errors that may occur in flash chips over time.

The SSD controller is responsible for using the ECC information to detect errors in the flash chips. Such flash controllers typically use a progressive approach to error correction: *small errors* (on the order of bits per KB of data) are quickly corrected using logic in the SSD controller while *large errors* (on the order of >10's of bits per KB) are corrected using more complex controller logic or with assistance from a driver running on the host. Thus, large errors that may be uncorrectable from an SSD's perspective, may be correctable from the system's perspective. For example, during the course of a read operation, if the SSD controller is unable to correct the errors in a particular chunk of data, the data and the ECC information is sent to the host machine where a driver uses the host machine's CPU to perform more complex error cor-

Platform	SSDs	PCIe	Per SSD				
			Capacity	Age (years)	Data written	Data read	UBER
A	1	v1, ×4	720 GB	2.4 ± 1.0	27.2 TB	23.8 TB	5.2×10^{-10}
B	2				48.5 TB	45.1 TB	2.6×10^{-9}
C	1	v2, ×4	1.2 TB	1.6 ± 0.9	37.8 TB	43.4 TB	1.5×10^{-10}
D	2				18.9 TB	30.6 TB	5.7×10^{-11}
E	1		3.2 TB	0.5 ± 0.5	23.9 TB	51.1 TB	5.1×10^{-11}
F	2				14.8 TB	18.2 TB	1.8×10^{-10}

Table 1: The platforms examined in our study. PCIe technology is denoted by vX, ×Y where X = version and Y = number of lanes. Data was collected over the entire age of the SSDs. Data written and data read are to/from the physical storage over an SSD's lifetime. UBER = uncorrectable bit error rate (Section 3.2).

rection and forwards the result to the host's OS. Errors that are not correctable by the driver result in data loss.[1]

Our SSDs allow us to collect information only on large errors that are uncorrectable by the SSD but correctable by the host. For this reason, our results are in terms of such SSD-uncorrectable but host-correctable errors, and when we refer to *uncorrectable errors* we mean these type of errors. We refer to the occurrence of such uncorrectable errors in an SSD as an *SSD failure*.

2.3 The Systems

We examined the majority of flash-based SSDs in Facebook's server fleet, which have operational lifetimes extending over nearly four years and comprising many millions of SSD-days of usage. Data was collected over the lifetime of the SSDs. We found it useful to separate the SSDs based on the type of *platform* an SSD was used in. We define a platform as a combination of the device capacity used in the system, the PCIe technology used, and the number of SSDs in the system. Table 1 characterizes the platforms we examine in our study.

We examined a range of high-capacity multi-level cell (MLC) flash-based SSDs with capacities of 720 GB, 1.2 TB, and 3.2 TB. The technologies we examined spanned two generations of PCIe, versions 1 and 2. Some of the systems in the fleet employed one SSD while others employed two. Platforms A and B contain SSDs with around two or more years of operation and represent 16.6% of the SSDs examined; Platforms C and D contain SSDs with around one to two years of operation and represent 50.6% of the SSDs examined; and Platforms E and F contain SSDs with around half a year of operation and represent around 22.8% of the SSDs examined.

2.4 Measurement Methodology

The flash devices in our fleet contain registers that keep track of statistics related to SSD operation (e.g., number of bytes read, number of bytes written, number of uncorrectable errors). These registers are similar to, but distinct from, the standardized SMART data stored by some SSDs to monitor their reliability characteristics [3]. The values in these registers can be queried by the host machine. We use a collector script to retrieve the raw values from the SSD and parse them into a form that can be curated in a Hive [38] table. This process is done in real time every hour.

The scale of the systems we analyzed and the amount of data being collected posed challenges for analysis. To process the many millions of SSD-days of information, we used a cluster of machines to perform a parallelized aggregation of the data stored in Hive using MapReduce jobs in order to get a set of lifetime statistics for each of the SSDs we analyzed. We then processed this summary data in R [2] to collect our results.

2.5 Analytical Methodology

Our infrastructure allows us to examine a *snapshot* of SSD data at a point in time (i.e., our infrastructure *does not* store *timeseries* information for the many SSDs in the fleet). This limits our analysis to the behavior of the fleet of SSDs at a point in time (for our study, we focus on a snapshot of SSD data taken during November 2014). Fortunately, the number of SSDs we examine and the diversity of their characteristics allow us to examine how reliability trends change with various characteristics. When we analyze an SSD characteristic (e.g., the amount of data written to an SSD), we group SSDs into buckets based on their value for that characteristic in the snapshot and plot the failure rate for SSDs in each bucket.

For example, if an SSD s is placed in a bucket for N TB of data written, it is *not* also placed in the bucket for $(N-k)$ TB of data written (even though at some point in its life it did have only $(N-k)$ TB of data written). When performing bucketing, we round the value of an SSD's characteristic to the nearest bucket and we eliminate buckets that contain less than 0.1% of the SSDs analyzed to have a statistically significant sample of SSDs for our measurements. In order to express the confidence of our observations across buckets that contain different numbers of servers, we show the 95th percentile confidence interval for all of our data (using a binomial distribution when considering failure rates). We measure SSD failure rate for each bucket in terms of the fraction of SSDs that have had an uncorrectable error compared to the total number of SSDs in that bucket.

3. BASELINE STATISTICS

We will first focus on the overall error rate and error distribution among the SSDs in the platforms we analyze and then examine correlations between different failure events.

3.1 Bit Error Rate

The bit error rate (BER) of an SSD is the rate at which errors occur relative to the amount of information that is transmitted from/to the SSD. BER can be used to gauge the reliability of data transmission across a medium. We measure the uncorrectable bit error rate (UBER) from the SSD as:

$$UBER = \frac{Uncorrectable\ errors}{Bits\ accessed}$$

For flash-based SSDs, UBER is an important reliability metric that is related to the SSD lifetime. SSDs with high UBERs are expected to have more failed cells and encounter more (severe) errors that may potentially go undetected and corrupt

[1] We do not examine the rate of data loss in this work.

data than SSDs with low UBERs. Recent work by Grupp et al. examined the BER of *raw* MLC flash chips (without performing error correction in the flash controller) in a controlled environment [20]. They found the raw BER to vary from 1×10^{-1} for the least reliable flash chips down to 1×10^{-8} for the most reliable, with most chips having a BER in the 1×10^{-6} to 1×10^{-8} range. Their study did *not* analyze the effects of the use of chips *in SSDs under real workloads and system software.*

Table 1 shows the UBER of the platforms that we examine. We find that for flash-based SSDs used in servers, the UBER ranges from 2.6×10^{-9} to 5.1×10^{-11}. While we expect that the UBER of the SSDs that we measure (which correct small errors, perform wear leveling, and are not at the end of their rated life but still being used in a production environment) should be less than the raw chip BER in Grupp et al.'s study (which did not correct any errors in the flash controller, exercised flash chips until the end of their rated life, and accessed flash chips in an adversarial manner), we find that in some cases the BERs were within around one order of magnitude of each other. For example, the UBER of Platform B in our study, 2.6×10^{-9}, comes close to the lower end of the raw chip BER range reported in prior work, 1×10^{-8}.

Thus, we observe that in flash-based SSDs that employ error correction for small errors and wear leveling, the UBER ranges from around 1/10 to 1/1000× the raw BER of similar flash chips examined in prior work [20]. This is likely due to the fact that our flash-based SSDs correct small errors, perform wear leveling, and are not at the end of their rated life. As a result, the error rate we see is smaller than the error rate the previous study observed.

As shown by the SSD UBER, the effects of uncorrectable errors are noticeable across the SSDs that we examine. We next turn to understanding how errors are distributed among a population of SSDs and how failures occur within SSDs.

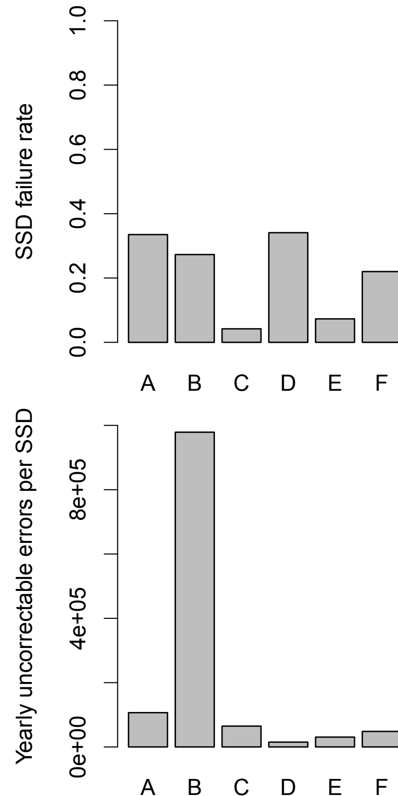

Figure 2: The failure rate (top) and average yearly uncorrectable errors per SSD (bottom), among SSDs within each platform.

3.2 SSD Failure Rate and Error Count

Figure 2 (top) shows the SSD incidence failure rate within each platform – the fraction of SSDs in each platform that have reported at least one uncorrectable error. We find that SSD failures are relatively common events with between 4.2% and 34.1% of the SSDs in the platforms we examine reporting uncorrectable errors. Interestingly, the failure rate is much lower for Platforms C and E despite their comparable amounts of data written and read (cf. Table 1). This suggests that there are differences in the failure process among platforms. We analyze which factors play a role in determining the occurrence of uncorrectable errors in Sections 4 and 5.

Figure 2 (bottom) shows the average yearly uncorrectable error rate among SSDs within the different platforms – the sum of errors that occurred on all servers within a platform over 12 months ending in November 2014 divided by the number of servers in the platform. The yearly rates of uncorrectable errors on the SSDs we examined range from 15,128 for Platform D to 978,806 for Platform B. The older Platforms A and B have a higher error rate than the younger Platforms C through F, suggesting that the incidence of uncorrectable errors increases as SSDs are utilized more. We will examine this relationship further in Section 4.

Platform B has a much higher average yearly rate of uncorrectable errors (978,806) compared to the other platforms (the second highest amount, for Platform A, is 106,678). We find that this is due to a small number of SSDs having a much higher number of errors in that platform: Figure 3 quantifies the distribution of errors among SSDs in each platform. The x axis is the normalized SSD number within the platform, with SSDs sorted based on the number of errors they have had over their lifetime. The y axis plots the number of errors for a given SSD in log scale. For every platform, we observe that the top 10% of SSDs with the most errors have over 80% of all uncorrectable errors seen for that platform. For Platforms B, C, E, and F, the distribution is even more skewed, with 10% of SSDs with errors making up over 95% of all observed uncorrectable errors. We also find that the distribution of number of errors among SSDs in a platform is similar to that of a Weibull distribution with a shape parameter of 0.3 and scale parameter of 5×10^3. The solid black line on Figure 3 plots this distribution.

An explanation for the relatively large differences in errors per machine could be that error events are correlated. Examining the data shows that this is indeed the case: during a recent two weeks, 99.8% of the SSDs that had an error during the first week also had an error during the second week. We therefore conclude that an SSD that has had an error in the past is highly likely to continue to have errors in the future. With this in mind, we next turn to understanding the correlation between errors occurring on both devices in the two-SSD systems we examine.

3.3 Correlations Between Different SSDs

Given that some of the platforms we examine have two flash-based SSDs, we are interested in understanding if the likeli-

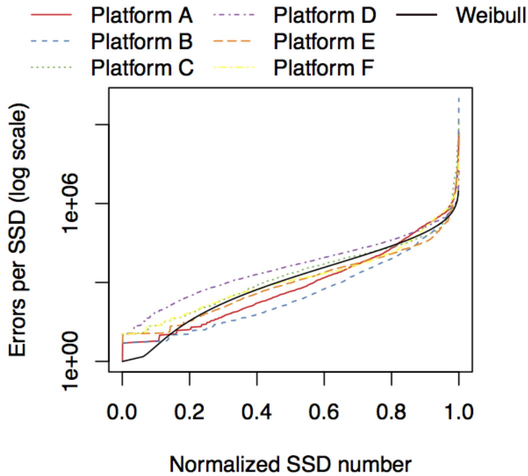

Figure 3: The distribution of uncorrectable error count across SSDs. The total number of errors per SSD is highly skewed, with a small fraction of SSDs accounting for a majority of the errors. The solid dark line plots a Weibull distribution that resembles the error trends.

hood of one SSD failing is affected by the other SSD failing. To examine this, we computed the conditional probability of both SSDs failing given that one SSD failed. We compute the conditional probability by dividing the number of systems in which both SSDs failed over their lifetime by the number of systems in which at least one SSD failed over its lifetime.

We find that the conditional probability of both SSDs failing given one SSD has failed is 42.2% for Platform B, 59.9% for Platform D, and 39.8% for Platform F. For comparison, if there were no correlation between SSD failures in the same machine and failures were uniformly distributed among SSDs, we would expect the conditional probability of both devices failing given that one SSD already failed to be similar to the uncorrectable error rate. For example, if one SSD in a server in Platform B fails, and that failure has no influence on the other SSD in the same server failing, the probability of the other SSD failing should be the same as the first SSD failing, 27.3%.

Instead, we find that one SSD failing in a machine *does* increase the probability of the other SSD failing (by 14.9% for Platform B, 25.8% for Platform D, and 17.8% for Platform F). This suggests that operational conditions related to the platform that SSDs are in contribute to the SSD failure trends we observe.

These baseline statistics we have examined raise questions about SSD failure in the field. For example, *how do workload-dependent characteristics (such as the amount of data written or read) affect failure rates? What role does the SSD controller play in SSD failure?* and *How do external factors (such as temperature) affect SSD failure trends?* We seek to answer these questions by examining how a variety of *internal* and *external* characteristics affect SSD failure trends, in Sections 4 and 5.

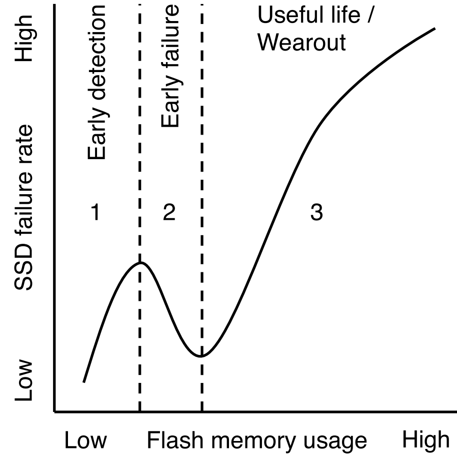

Figure 4: SSD lifecycle failure pattern. SSDs fail at different rates during several distinct periods throughout their lifetime (measured by usage)

4. THE ROLE OF INTERNAL FACTORS

We examine next how factors related to the *internals of the flash chips and the SSD controller* affect uncorrectable errors over an SSD's lifetime. We examine the effects of writing data to flash cells, reading data from flash cells, copying and erasing data, discarding unusable blocks, and the SSD-internal DRAM buffer usage.

4.1 Data Written to Flash Cells

Due to the physical properties that govern them, flash cels have been shown to become less reliable the more times their contents are *programmed and erased*, i.e., *P/E cycles* the flash cell has endured (we refer the reader to [6], which provides a good summary of flash cell operation and characteristics). Several recent works have quantified the effects of P/E cycles on various error mechanisms in small sets of recent flash chips (e.g., [6, 7, 8, 9, 10, 11, 12, 13, 20]). The effects of P/E cycles on flash reliability are important to understand because they can be exacerbated in higher capacity MLC chips commonly used in flash-based SSDs.

In order to examine the effect of P/E cycles on flash reliability, we consider the amount of data *written directly to flash cells* over each SSD's lifetime. Our framework allows us to measure this value, which more accurately portrays flash memory usage over its lifetime compared to software-level writes. Note that software-level writes (i.e., write requests sent by software to the storage device) do *not* directly get exposed to the flash cells due to layers of buffering in the storage software stack (we will examine their effects in Section 5.3).

Prior reliability studies on hard disk drives in the field (e.g., [35]) observed a trend with respect to writes, called the *bathtub curve*. The bathtub curve is so named because of its shape with respect to device failure rate over time: devices initially experience a high failure rate during an early failure period, followed by a low rate of failure during the useful life period, and ultimately a high failure rate once again during their wearout period. In the SSDs we examine, we notice an additional period *before* the early failure period that we call the *early detection period*. During the early detection period, which occurs when SSDs are young (measured by data written

directly to flash cells), cells are read, programmed, and erased, and unreliable cells are identified by the SSD controller, resulting in an initially high failure rate among devices. Figure 4 pictorially illustrates the lifecycle failure pattern we observe, which is quantified by Figure 5.

Figure 5 plots how the failure rate of SSDs varies with the amount of data written to the flash cells. We have grouped the platforms based on the individual capacity of their SSDs. Notice that across most platforms, the failure rate is low when little data is written to flash cells. It then increases at first (corresponding to the early detection period, region 1 in the figures), and decreases next (corresponding to the early failure period, region 2 in the figures). Finally, the error rate generally increases for the remainder of the SSD's lifetime (corresponding to the useful life and wearout periods, region 3 in the figures). An obvious outlier for this trend is Platform C – in Section 5, we observe that some *external* characteristics of this platform lead to its atypical lifecycle failure pattern.

Note that different platforms are in different stages in their lifecycle depending on the amount of data written to flash cells. For example, SSDs in Platforms D and F, which have the least amount of data written to flash cells on average, are mainly in the early detection or early failure periods. On the other hand, SSDs in older Platforms A and B, which have more data written to flash cells, span all stages of the lifecycle failure pattern (depicted in Figure 4). For SSDs in Platform A, we observe up to an 81.7% difference between the failure rates of SSDs in the early detection period and SSDs in the wearout period of the lifecycle.

As explained before and as depicted in Figure 4, the lifecycle failure rates we observe with the amount of data written to flash cells does *not* follow the conventional bathtub curve. In particular, the new *early detection period* we observe across the large number of devices leads us to investigate why this "early detection period" behavior exists. Recall that the early detection period refers to failure rate increasing early in lifetime (i.e., when a small amount of data is written to the SSD). After the early detection period, failure rate starts decreasing.

We hypothesize that this non-monotonic error rate behavior during and after the early detection period can be accounted for by a *two-pool* model of flash blocks: one pool of blocks, called the *weaker pool*, consists of cells whose error rate increases much faster than the other pool of blocks, called the *stronger pool*. The weaker pool quickly generates uncorrectable errors (leading to increasing failure rates observed in the early detection period as these blocks keep failing). The cells comprising this pool ultimately fail and are taken out of use early by the SSD controller. As the blocks in the weaker pool are exhausted, the overall error rate starts decreasing (as we observe *after* the end of what we call the *early detection period*) and it continues to decrease until the more durable blocks in the stronger pool start to wear out due to typical use.

We notice a general increase in the duration of the lifecycle periods (in terms of data written) for SSDs with larger capacities. For example, while the early detection period ends after around 3 TB of data written for 720 GB and 1.2TB SSDs (in Platforms A, B, C, and D), it ends after around 10 TB of data written for 3.2 TB SSDs (in Platforms E and F). Similarly, the early failure period ends after around 15 TB of data written for 720 GB SSDs (in Platforms A and B), 28 TB for 1.2 TB SSDs (in Platforms C and D), and 75 TB for 3.2 TB SSDs (in Platforms E and F). This is likely due to the more flexibility a higher-capacity SSD has in reducing wear across a larger number of flash cells.

4.2 Data Read from Flash Cells

Similar to *data written*, our framework allows us to measure the amount of *data directly read from flash cells* over each SSD's lifetime. Prior works in controlled environments have shown that errors can be induced due to read-based access patterns [5, 32, 6, 8]. We are interested in understanding how prevalent this effect is across the SSDs we examine.

Figure 6 plots how failure rate of SSDs varies with the amount of data read from flash cells. For most platforms (i.e., A, B, C, and F), the failure rate trends we see in Figure 6 are similar to those we observed in Figure 5. We find this similarity when the *average* amount of data written to flash cells by SSDs in a platform is more than the *average* amount of data read from the flash cells. Platforms A, B, C, and F show this behavior.[2]

[2]Though Platform B has a set of SSDs with a large amount of data read, the *average* amount of data read across all SSDs in this platform is less than the *average* amount of data written.

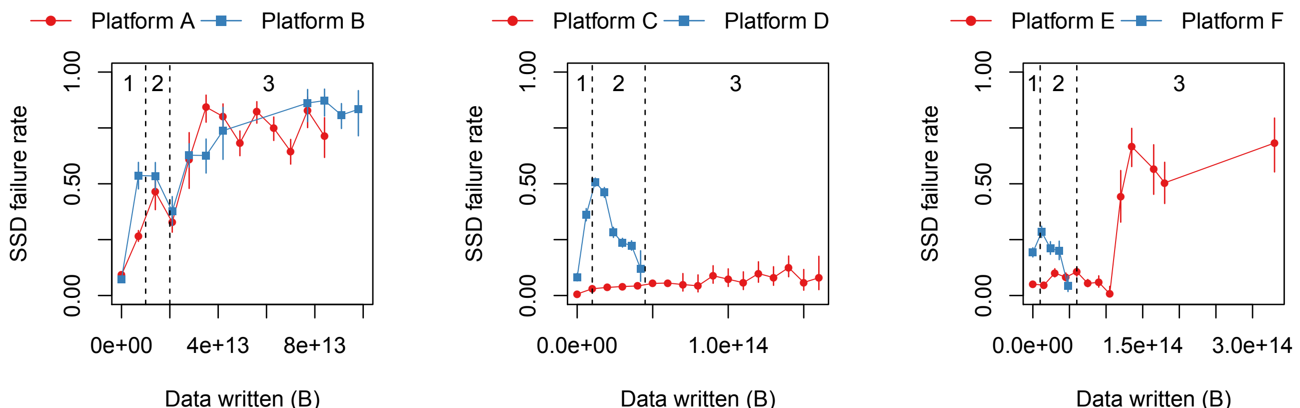

Figure 5: SSD failure rate vs. the amount of data written to flash cells. SSDs go through several distinct phases throughout their life: increasing failure rates during early detection of less reliable cells (1), decreasing failure rates during early cell failure and subsequent removal (2), and eventually increasing error rates during cell wearout (3).

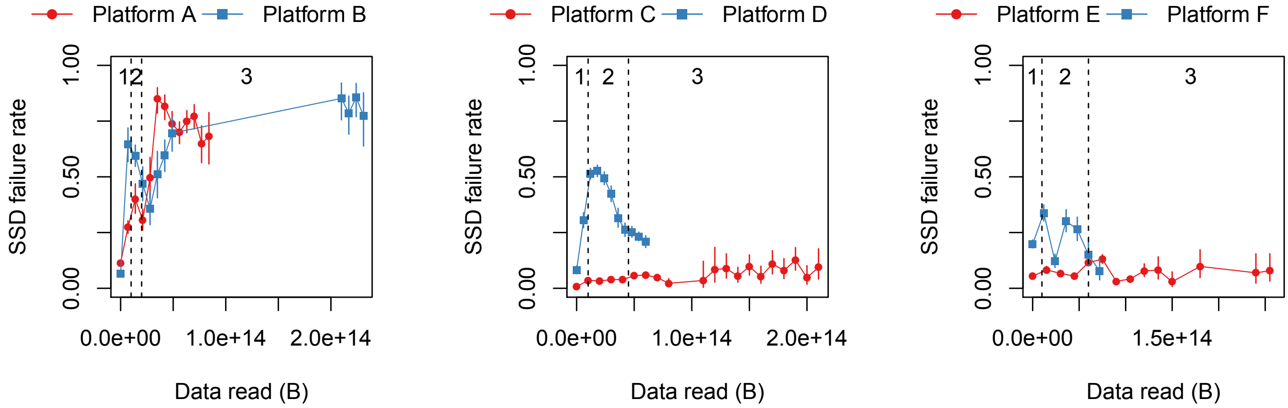

Figure 6: SSD failure rate vs. the amount of data read from flash cells. SSDs in Platform E, where over twice as much data is read from flash cells as written to flash cells, do not show failures dependent on the amount of data read from flash cells.

In Platform D, where more data is read from flash cells than written to flash cells (30.6 TB versus 18.9 TB, on average), we notice error trends associated with the early detection and early failure periods. Unfortunately, even though devices in Platform D may be prime candidates for observing the occurrence of read disturbance errors (because more data is read from flash cells than written to flash cells), the effects of early detection and early failure appear to dominate the types of errors observed on this platform.

In Platform E, however, over twice as much data is read from flash cells as written to flash cells (51.1 TB versus 23.9 TB on average). In addition, unlike Platform D, devices in Platform E display a diverse amount of utilization. Under these conditions, we *do not* observe a statistically significant difference in the failure rate between SSDs that have read the most amount of data versus those that have read the least amount of data. This suggests that the effect of read-induced errors in the SSDs we examine is not predominant compared to other effects such as write-induced errors. This corroborates prior flash cell analysis that showed that most errors occur due to retention effects and not read disturbance [32, 11, 12].

4.3 Block Erases

Before new data can be written to a page in flash memory, the entire block needs to be first erased.[3] Each erase wears out the block as shown in previous works [32, 11, 12]. Our infrastructure tracks the average number of blocks erased when an SSD controller performs *garbage collection*. Garbage collection occurs in the background and erases blocks to ensure that there is enough available space to write new data. The basic steps involved in garbage collection are: (1) selecting blocks to erase (e.g., blocks with many unused pages), (2) copying pages from the selected block to a free area, and (3) erasing the selected block [29]. While we do not have statistics on how frequently garbage collection is performed, we examine the *number of erases per garbage collection period* metric as an indicator of the average amount of erasing that occurs within each SSD.

We examined the relationship between the number of erases performed versus the failure rate (not plotted). Across the SSDs we examine, we find that the failure trends in terms of erases are correlated with the trends in terms of data written (shown in Figure 5). This behavior reflects the typical operation of garbage collection in SSD controllers: As more data is written, wear must be leveled across the flash cells within an SSD, requiring more blocks to be erased and more pages to be copied to different locations (an effect we analyze next).

4.4 Page Copies

As data is written to flash-based SSDs, pages are copied to free areas during garbage collection in order to free up blocks with unused data to be erased and to more evenly level wear across the flash chips. Our infrastructure measures the number of pages copied across an SSD's lifetime.

We examined the relationship between the number of pages copied versus SSD failure rate (not plotted). Since the page copying process is used to free up space to write data and also to balance the wear due to writes, its operation is largely dictated by the amount of data written to an SSD. Accordingly, we observe similar SSD failure rate trends with the number of copied pages (not plotted) as we observed with the amount of data written to flash cells (shown in Figure 5).

4.5 Discarded Blocks

Blocks are discarded by the SSD controller when they are deemed unreliable for further use. Discarding blocks affects the usable lifetime of a flash-based SSD by reducing the amount of over-provisioned capacity of the SSD. At the same time, discarding blocks has the potential to reduce the amount of errors generated by an SSD, by preventing unreliable cells from being accessed. Understanding the reliability effects of discarding blocks is important, as this process is the main defense that SSD controllers have against failed cells and is also a major impediment for SSD lifetime.

Figure 7 plots the SSD failure rate versus number of discarded blocks over the SSD's lifetime. For SSDs in most platforms, we observe an initially decreasing trend in SSD failure rate with respect to the number of blocks discarded followed by an increasing and then finally decreasing failure rate trend.

We attribute the initially high SSD failure rate when few blocks have been discarded to the two-pool model of flash cell failure discussed in Section 4.1. In this case, the weaker pool of flash blocks (corresponding to weak blocks that fail

[3]Flash pages (around 8 KB in size) compose each flash block. A flash block is around 128 × 8 KB flash pages.

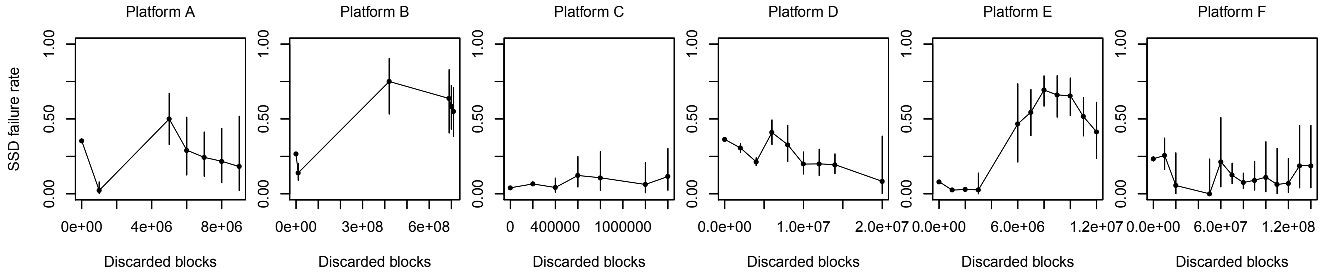

Figure 7: SSD failure rate vs. the number of discarded blocks. We find that (1) failure rate is relatively high among SSDs that have discarded few blocks (far left of plots), (2) some SSDs seem to effectively mask failures by discarding blocks (initial decrease in failure rates), and (3) discarding a large amount of blocks is indicative of higher failure rates (toward the right of plots).

early) cause errors before they can be discarded by the SSD controller, causing an initially high SSD failure rate. On the other hand, the stronger pool of flash blocks (which fail due to gradual wearout over time) contribute to the subsequent decrease in SSD failure rate. Some SSDs that have discarded a large number of blocks have high error rates (e.g., Platforms A, B, and E), indicating that discarding a large number of blocks is a good indicator of the likelihood of SSD failure.

We further examined the devices with the largest number of discarded blocks and found that the number of blocks that are discarded is *not* correlated with the amount of data written to or read from flash cells (examined in Sections 4.1 and 4.2). In other words, we observe SSDs of both low *and* high usage across their lifetime (measured by flash cell reads and writes) that have discarded large amounts of blocks and have high failure rates. This suggests that the relationship between discarded blocks and failure rate is to some extent intrinsic to the SSD. Thus, some SSDs, despite discarding many blocks, continue to encounter uncorrectable errors at a much higher rate than their peers.

4.6 DRAM Buffer Usage

Flash-based SSDs use internal DRAM to provide buffer space for SSD controller metadata or for data to be written to the flash chips. The SSDs we examine use the internal DRAM buffer space only to store metadata related to the flash translation layer mapping for logical addresses to physical addresses. This allows the SSD controller to locate data on an SSD quickly, reducing the performance impact of address translation.

The SSDs we examine use the DRAM buffer *less* when data is *densely allocated* (e.g., contiguous data) and use the DRAM buffer *more* when data is *sparsely allocated* (e.g., non-contiguous data). As an illustrative example, using an SSD to read from a *large* file would lead to a contiguous allocation of data on the SSD and a *small* DRAM buffer usage (due to a small number of mappings that need to be tracked in the FTL). On the other hand, using an SSD to read from and write to many *small* files would lead to many non-contiguous allocations of data on the SSD and a *large* DRAM buffer usage (due to a large number of mappings that need to be tracked in the FTL).

Examining DRAM buffer usage can therefore provide an indication of how system data allocation behavior affects flash reliability. To capture the DRAM buffer usage of SSDs, we examine the average amount of DRAM buffer space used over two recent weeks of SSD operation, sampled at one hour intervals.

Figure 8 plots the failure rate for SSDs that use different amounts of DRAM buffer space on average. For some platforms (A, B, and D), we observe that as more DRAM buffer space is used, SSD failure rate increases. Such a trend indicates that some systems that allocate data more sparsely have higher failure rates. Such behavior is potentially due to the fact that sparse data allocation can correspond to access patterns that write small amounts of non-contiguous data, causing the SSD controller to more frequently erase and copy data compared to writing contiguous data.

While Platform C does not appear to be sensitive to DRAM buffer usage, Platforms E and F demonstrate a trend of higher error rates at lower DRAM buffer usage. We attribute this behavior to a separate pattern of write behavior that we hypothesize these SSDs are subjected to. In these platforms, we believe that the SSDs allocate large, contiguous regions of data (resulting in low DRAM buffer usage) but write to them sparsely but intensely (leading to cell wearout). This could occur, for example, when frequently updating small fields in a large, contiguously-allocated data structure.

Interestingly, we observe similar failure behavior with DRAM buffer usage at the *application level* (Figure 9). We examine six of the largest distributions of applications on Platform B: Graph Search is a distributed graph search service; Batch Processor executes long-running asynchronous jobs; Key–Value Store stores persistent mappings of keys to values; Load Balancer is a programmable traffic routing framework; Distributed Key–Value Store is like a Key–Value Store with strong reliability, availability, and consistency guarantees; and Flash Cache is a cache for data with a large working set and low access frequency [19]. Applications across SSDs have increasing failure rates with increasing DRAM buffer usage (sparse data mappings) and in some cases have increases in failure rates at lower DRAM buffer usage (dense data mappings, e.g., Batch Processor and Flash Cache).

We conclude that small, sparse writes negatively affect SSD failure rates the most for sparse data mappings (e.g., non-contiguous data). But failures are also noticeable for dense data mappings (e.g., contiguous data). Given this behavior, we believe that there is the potential for developing more effective techniques to handle the adversarial access pattern of small, sparse writes.

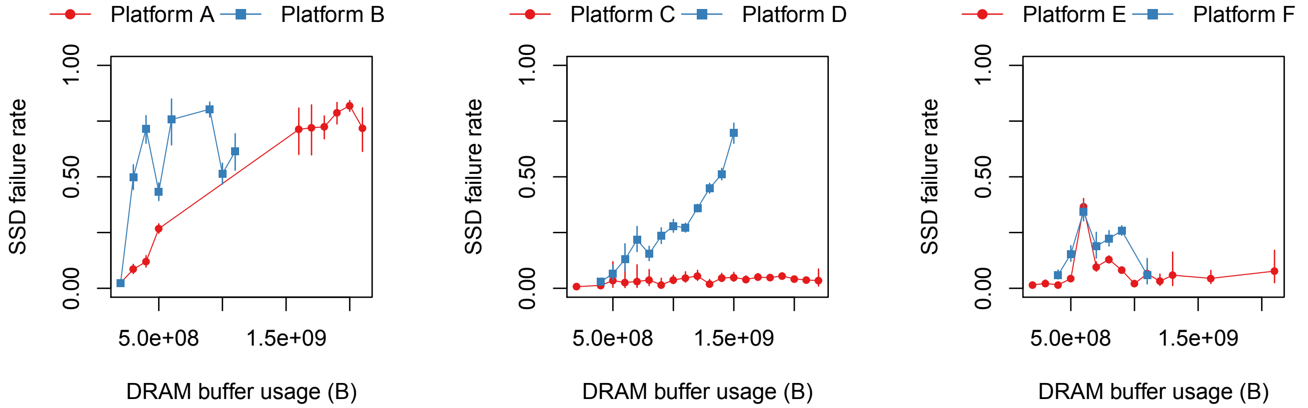

Figure 8: SSD failure rate vs. DRAM buffer usage. Sparse data mappings (e.g., non-contiguous data, indicated by high DRAM buffer usage to store flash translation layer metadata) negatively affect SSD reliability the most (Platforms A, B, and D). Additionally, some dense data mappings (e.g., contiguous data in Platforms E and F) also negatively affect SSD reliability, likely due to the effect of small, sparse writes.

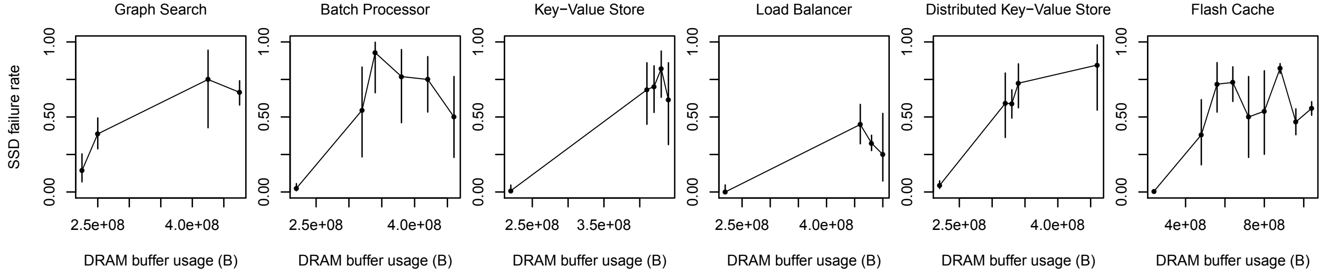

Figure 9: SSD failure rate vs. DRAM buffer usage across six applications that run on Platform B. We observe similar DRAM buffer effects to Figure 8, even among SSDs running the same application.

5. THE ROLE OF EXTERNAL FACTORS

We next examine how factors external to the SSD influence the errors observed over an SSD's lifetime. We examine the effects of temperature, PCIe bus power, and system-level writes reported by the OS.

5.1 Temperature

It is commonly assumed that higher temperature negatively affects the operation of flash-based SSDs. In flash *cells*, higher temperatures have been shown to cause cells to age more quickly due to the temperature-activated Arrhenius effect [39]. Temperature-dependent effects are especially important to understand for flash-based SSDs in order to make adequate data center provisioning and cooling decisions. To examine the effects of temperature, we used temperature measurements from temperature sensors embedded on the SSD cards, which provide a more accurate portrayal of the temperature of flash cells than temperature sensors at the server or rack level.

Figure 10 plots the failure rate for SSDs that have various average operating temperatures. We find that at an operating temperature range of 30 to 40°C, SSDs across server platforms see similar failure rates or slight increases in failure rates as temperature increases.

Outside of this range (at temperatures of 40°C and higher), we find that SSDs fall into one of three categories with respect to their reliability trends vs. temperature: (1) temperature-sensitive with increasing failure rate (Platforms A and B), (2) less temperature-sensitive (Platforms C and E), and (3) temperature-sensitive with decreasing failure rate (Platforms D and F). There are two factors that may affect the trends we observe with respect to SSD temperature.

One potential factor when analyzing the effects of temperature is the operation of the SSD controller in response to changes in temperature. The SSD controllers in some of the SSDs we examine attempt to ensure that SSDs do not exceed certain temperature thresholds (starting around 80°C). Similar to techniques employed in processors to reduce the amount of processor activity in order to keep the processor within a certain temperature range, our SSDs attempt to change their behavior (e.g., reduce the frequency of SSD access or, in the extreme case, shut down the SSD) in order not to exceed temperature thresholds.

A second potential factor is the thermal characteristics of the machines in each platform. The existence of two SSDs in a machine (in Platforms B, D, and F) compared to one SSD in a machine may (1) increase the thermal capacity of the machine (causing its SSDs to reach higher temperatures more quickly and increase the work required to cool the SSDs) and (2) potentially reduce airflow to the components, prolonging the effects of high temperatures when they occur.

One hypothesis is that temperature-sensitive SSDs with increasing error rates, such as Platforms A and B, may not employ as aggressive temperature reduction techniques as other platforms. While we cannot directly measure the actions the SSD controllers take in response to temperature events, we examined an event that can be correlated with temperature

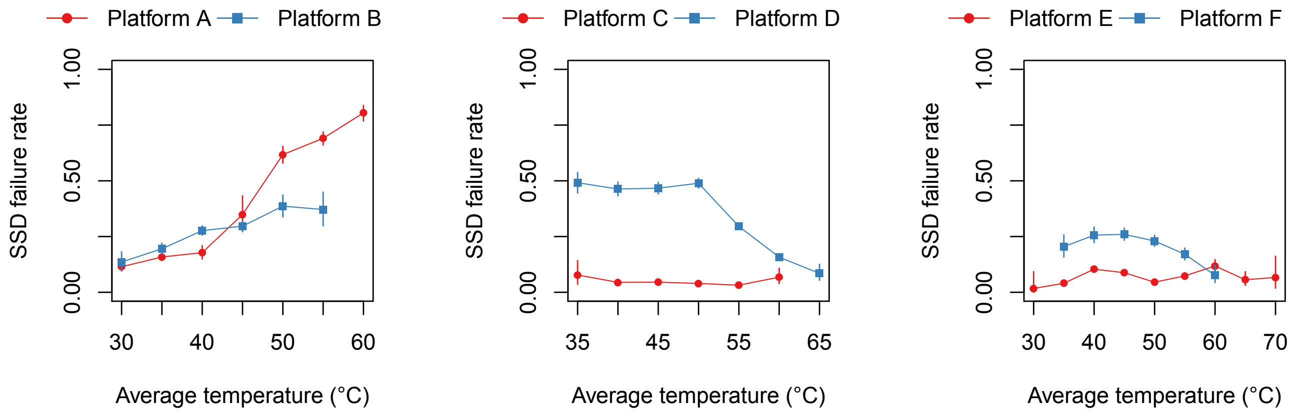

Figure 10: SSD failure rate vs. temperature. Operational temperatures of 30 to 40°C generally show increasing failure rates. Failure trends at and above 40°C follow three distinct failure rate patterns: increasing, not sensitive, and decreasing.

reduction: whether or not an SSD has throttled its operation in order to reduce its power consumption. Performing a large number of writes to an SSD consumes power and increases the temperature of the SSD. Figure 11 plots, for each temperature, the fraction of machines that have ever been throttled. Examining this figure confirms that Platforms A and B, where no machines or few machines have been throttled, exhibit behavior that is typical for SSDs without much preventative action against temperature increases. In these platforms, as temperature increases, failure rate of SSDs increases.

In contrast to Platforms A and B, Platforms C and E, which are less temperature-sensitive, throttle their SSDs more aggressively across a range of temperatures. From Figure 10 we can see that throttled SSDs have lower failure rates (in Platforms C and E) compared to SSDs that are throttled less or not throttled at all (Platforms A and B). We attribute the relatively low SSD failure rate for Platforms C and E we have observed in our measurements to the very aggressive throttling that occurs for SSDs in these two platforms. Such throttling could potentially reduce performance, though we are not able to examine this effect.

SSDs in Platforms D and F employ a relatively low amount of throttling (Figure 11), but exhibit the counter-intuitive trend of decreasing failure rate with increased temperature. Recall from Section 4.1 that these SSDs are predominantly in their early detection and early failure periods and so the failure rates for most SSDs in these platforms are relatively high compared to their peers in Platforms C and E. It is likely that a combination of power throttling and some other form of temperature-dependent throttling employed by the SSD controller that we are not able to measure is responsible for reducing the failure rate among the SSDs in Platforms D and F as temperature increases.

5.2 Bus Power Consumption

According to the PCIe standard, the nominal bus power consumption for the PCIe ×4 SSDs that we analyze is 10 W (regardless of PCIe version 1 or 2). Our infrastructure allows us to measure the average amount of power consumed on the PCIe bus by SSDs. As power consumption in servers can lead to increased electricity use for operation and cooling in data centers, we are interested in understanding the role that SSD power draw plays with respect to errors.

Figure 12 plots the failure rate for SSDs that operate at different average amounts of bus power consumption. Recall that Platforms A and B use PCIe v1 and that Platforms C through F use PCIe v2. We make three observations about the operation of these SSDs. First, PCIe v2 SSDs (Platforms C through F) support twice the bandwidth of PCIe v1 SSDs (Platforms A and B) by operating at twice the frequency, leading to around twice the amount of power consumed between the two sets of SSDs: SSDs supporting PCIe v1 operate in the range of 4 to 7.5 W and SSDs supporting PCIe v2 operate in the range of 8 to 14.5 W. Second, we find that the bus power that SSDs consume can vary over a range of around 2× between the SSDs that consume the least bus power and

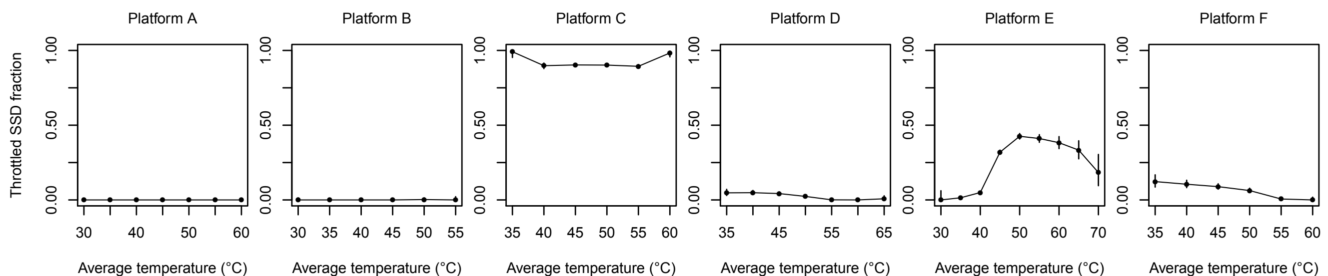

Figure 11: Fraction of SSDs ever throttled vs. SSD temperature. While SSDs in some platforms are never throttled (A and B), others are throttled more aggressively (C and E).

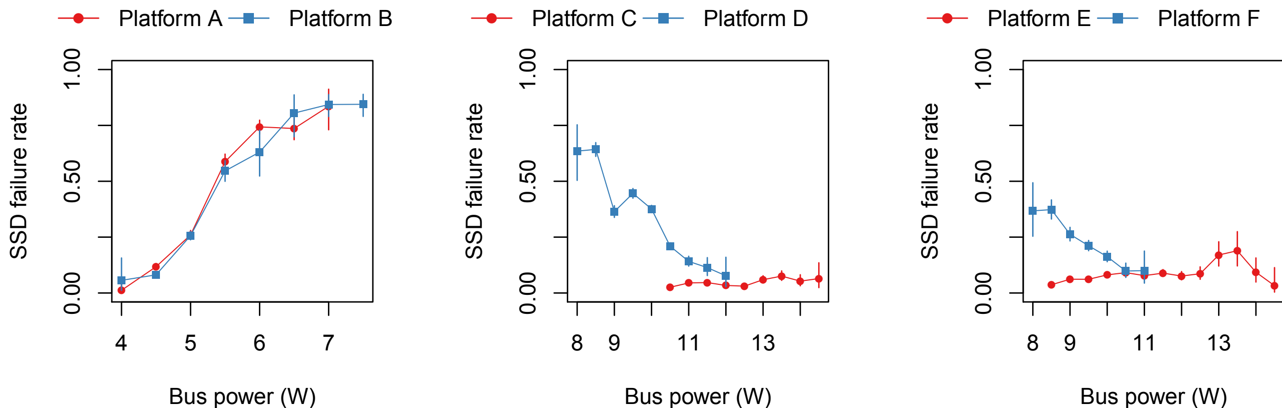

Figure 12: SSD failure rate vs. bus power consumption. PCIe v2 SSDs (Platforms C through F) consume more bus power than PCIe v1 SSDs (Platforms A and B) due to their higher frequency. Bus power consumption range (1) spans up to around 2× for SSDs within a platform (e.g., Platform B) and (2) is typically different from the nominal PCIe bus power of 10 W.

those that consume the most within a platform. Third, we find that PCIe v2 SSDs in systems that use two SSDs (Platforms D and F) tend to consume lower bus power compared to PCIe v2 SSDs in systems that use one SSD (Platforms C and E). This may be due to SSD access being distributed between the two SSDs. Note, however, that the *total* bus power consumption of a two-SSD system is larger than a single-SSD system.

With regard to failure rates, we observe that the trends for bus power consumption are correlated with those of average temperature. We believe that the higher bus power consumption is due to more data being transmitted and more data being transmitted requires more power to read or write from the SSD, which in turn increases the SSD temperature. One interesting result of this correlation is that techniques that aim to reduce SSD temperature in the absence of precise temperature sensors may be able to use bus power consumption as a proxy.

5.3 Data Written by the System Software

While we have shown that the amount of data that is written to flash cells has a direct effect on flash-based SSD lifetime, it is less clear what effect *system-level writes* reported by the OS (i.e., writes initiated by the system software on behalf of user programs or the file system) have on SSD error characteristics. Note that not all data written by system software may get written to flash chips due to system-level buffering. However, the reduction of writes performed by the system software is often used as a figure of merit in techniques that evaluate flash reliability in lieu of modeling the wear reduction techniques present in SSDs. Therefore, we are interested in examining whether or not different amounts of writes performed by the system software lead to different error characteristics in SSDs.

To this end, we measured the number of sectors (512 B in size) modified by the OS on behalf of the applications running on our machines over their lifetime, which we refer to as *data written by the system software*. Figure 13 plots the failure rate for SSDs where the OS has written different numbers of sectors over their lifetime.

In general, we observe that the total amount of data written by the system software is correlated with increased failure rates (following similar trends as the SSD lifecycle from Figure 4). There is one important caveat: some systems where a large amount of data has been written by the system software, have lower failure rates. This can be seen, for example, in Platforms A, B, and F: while there is a general trend toward higher SSD failure rates with larger amounts of data written by the system software for other platforms, the SSDs in Platforms A, B, and F can have lower failure rates with larger amounts of data written by the system software. We believe that this is due to the fact that systems that write more data may be able to benefit more from system-level buffering (e.g., the OS page cache) along with techniques employed by the SSD controller to reduce the amount of wear on flash cells (e.g., data encoding schemes that help reduce the number of program or erase operations).

To confirm this trend, Figure 14 plots the amount of data *actually written to the flash chips* compared to the amount of data *the system software reported as having written*. SSDs in Platforms B and F clearly show that writing more data at the system software-level does *not* always imply physically writing more data to flash chips. In fact, writing more data at the system software level may actually offer the system software more opportunities to *coalesce* data in system-level buffers before writing it to the flash cells, resulting in smaller numbers of actual writes to the flash chips. This observation suggests that work concerned with improving flash reliability *must* consider the effects of buffering across the system when evaluating different techniques.

6. LIMITATIONS AND POTENTIAL CONFOUNDING FACTORS

Workload changes due to SSD failure. In order to make systems more resilient to SSD failures, data replication is employed at the software level in some workloads. Such replication stores multiple copies of data (e.g., three copies) across different servers. In the event that data on one SSD becomes unavailable (e.g., due to an SSD failure), a copy of data can still be readily accessible.

In Section 3.3, we examined the correlation between SSD failures *within* the same server. Note that, with replication, an SSD failure will not shift the workload to the other SSD in the same server, as replicated copies are spread out among

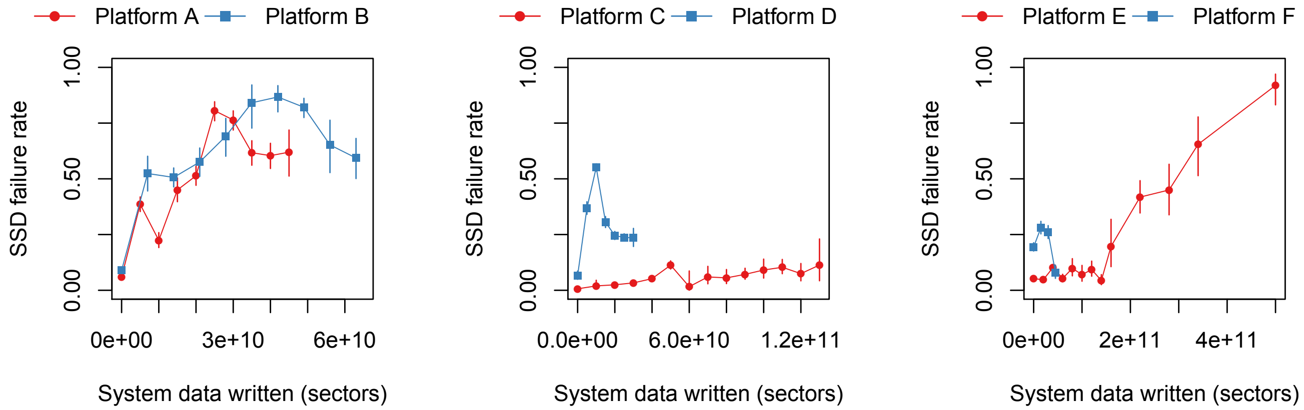

Figure 13: SSD failure rate vs. system data written by the system software. Interestingly, the amount of data written by the system software is not always an accurate indication of the amount of wear induced in an SSD, as seen in Platforms A and B, where more data written by the system software can correspond to lower failure rates.

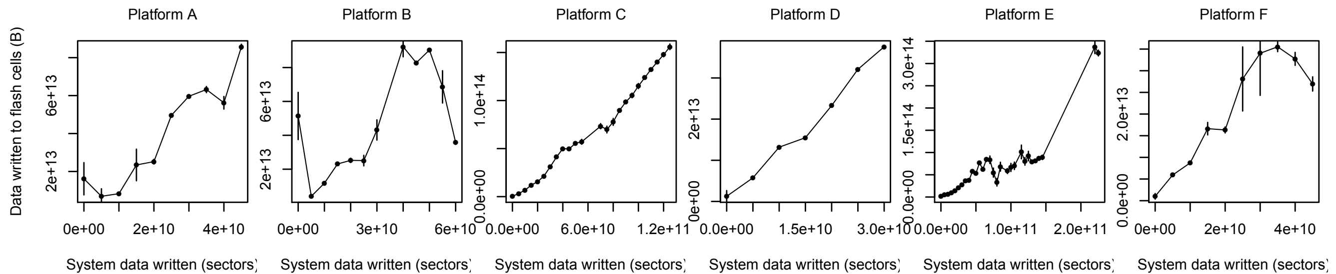

Figure 14: Actual data written to flash *cells* vs. data the OS reported as having been written. Due to buffering present in the system software (e.g., the page cache), the amount of data written by the system software does not always correspond to the amount of data written to the SSD (e.g., in Platforms A, B, and F).

different servers. Replication will, however, increase the utilization of other SSDs that contain copies of data. As such, correlation between SSD failures in *different* servers may exist, though we do not examine such effects in this work.

Performance effects. We also did not examine how the performance of a system was affected by the rate of uncorrectable errors on its SSDs and whether this made machines with SSDs reporting errors less effective at executing their workload.

SSD and flash chip access patterns. Our data collection infrastructure did not allow us to collect detailed access traces to the SSDs and flash chips we examined. This posed challenges for understanding the underlying causes of error behavior. While we examined the aggregate behavior of SSDs (e.g., amount of data written/read to/from flash chips in Sections 4.1 and 4.2) in order to gain insight into potential underlying access pattern trends, we believe that further SSD and flash chip access pattern analysis is needed to fully understand the underlying causes of the reliability trends we have identified.

Error correction scheme. The SSDs we examined used one viable strategy for error correction. Data uncorrectable by the SSD is forwarded to the host for repair. We note that such an error correction scheme may *not* be employed on other SSDs (and may not even be possible for some SSDs, such as those that use the NVMe interface [1]). While our results are still representative of the types of uncorrectable errors that such SSDs face, we believe that it will be beneficial for future studies to examine the effects of different error correction schemes in SSDs.

SSD data loss and repair. Recall from Section 2.2 that when an error is not correctable by an SSD, it is forwarded for repair to the host that the SSD is connected to. When the host cannot repair an error, data loss ensues. We did not examine the rate of data loss in the systems we evaluated. Nor did we examine the rate of SSD replacement. However, we believe that such an analysis (potentially combined with various error correction schemes, as mentioned above) would be very useful to perform in future work.

As we also mentioned in Section 2.2, the amount of errors the SSDs we examined can tolerate without the assistance of the host is on the order of bits per KB of data (similar to data reported in [17]). However, we do not have visibility into the error correction schemes employed on the host for larger amounts of errors (>10's of bits per KB of data).

7. RELATED WORK

To our knowledge, this paper provides the first comprehensive study of flash-based SSD reliability trends in the field. Prior flash memory reliability studies focused on flash *chip* reliability and evaluated only *a small number* of flash chips in *controlled* environments.

The closest related work, by Grupp et al. [20], examined the cost, performance, capacity, and reliability trends of flash memory in the context of a prototypical server flash storage device similar in architecture to SSDs deployed in data centers (Figure 1). Based on their study, the authors project several challenges for the adoption of flash in a server context. One drawback of the reliability results presented in [20] is that the experiments were performed in a controlled environment, under synthetic workloads, while modeling only the latency – but not the function – of the SSD controller on a small number (45) of flash chips.

A recent work by Ouyang et al. [34] performed a study of programmable SSD controllers at a large-scale web services company, Baidu. While this work examined flash-based SSDs in the field, it did not analyze the reliability characteristics of SSDs and instead focused on bandwidth, capacity, and cost.

Several studies examined flash chips in controlled environments in order to understand their error characteristics. Our study is orthogonal to these works. Early analysis of flash circuitry identified flash reliability trends attributed to program disturbance due to tunneling effects and hot-electron injection [36, 23, 21, 27, 22], quantum-level noise effects [25, 16], erratic tunneling [33, 14], data-retention and read disturbance [5, 18, 4], and detrapping-induced retention effects [24, 40, 41, 26, 31, 30, 37]. A series of works by Cai et al. [6, 11, 10, 7, 9, 13, 8, 12] identified patterns of errors for more recent MLC flash *chips* and proposed techniques to help mitigate their effects. These studies can help identify (and mitigate) the trends in flash error patterns at the raw chip level. However, it is unclear from these studies to what extent *flash-based SSDs* in the field running real workloads and real system software over a long time may exhibit such error patterns (and how to ensure the reliability of such SSDs). For example, compared to prior chip-level studies, we find that *SSD* failure rates do *not* increase monotonically with *flash chip* wear; that read disturbance errors are *not* prevalent in the field; that the data layout across an SSD's physical address space has an effect on SSD failure rate; that SSD controller throttling techniques can help mitigate temperature-induced errors; and that optimizations in the SSD controllers and buffering employed in the system software reduces the amount of data written to flash chips. These differences from prior studies can be attributed to our holistic study of flash errors when running on real SSD controllers, under real system software, and with real data center workloads.

8. SUMMARY AND CONCLUSIONS

We performed an extensive analysis of the effects of various factors on flash-based SSD reliability across a majority of the SSDs employed at Facebook, running production data center workloads. We analyze a variety of internal and external characteristics of SSDs and examine how these characteristics affect the trends for uncorrectable errors. To conclude, we briefly summarize the key observations from our study and discuss their implications for SSD and system design.

Observation 1: *We observe that SSDs go through several distinct failure periods – early detection, early failure, usable life, and wearout – during their lifecycle, corresponding to the amount of data written to flash chips.*

Due to pools of flash blocks with different reliability characteristics, failure rate in a population does *not* monotonically increase with respect to amount of data written to flash chips. This is unlike the failure rate trends seen in raw flash chips. We suggest that techniques should be designed to help reduce or tolerate errors *throughout SSD lifecycle*. For example, additional error correction *at the beginning of an SSD's life* could help reduce the failure rates we see during the *early detection* period.

Observation 2: *We find that the effect of read disturbance errors is not a predominant source of errors in the SSDs we examine.*

While prior work has shown that such errors can occur under certain access patterns in controlled environments [5, 32, 6, 8], we do *not* observe this effect across the SSDs we examine. This corroborates prior work which showed that the effect of retention errors in flash cells dominate error rate compared to read disturbance [32, 6]. It may be beneficial to perform a more detailed study of the effect of these types of errors in flash-based SSDs used in servers.

Observation 3: *Sparse data layout across an SSD's physical address space (e.g., non-contiguously allocated data) leads to high SSD failure rates; dense data layout (e.g., contiguous data) can also negatively impact reliability under certain conditions, likely due to adversarial access patterns.*

Further research into flash write coalescing policies with information from the system level may help improve SSD reliability. For example, information about write access patterns from the operating system could potentially inform SSD controllers of non-contiguous data that is accessed very frequently, which may be one type of access pattern that adversely affects SSD reliability and is a candidate for storing in a separate write buffer.

Observation 4: *Higher temperatures lead to increased failure rates, but do so most noticeably for SSDs that do not employ throttling techniques.*

In general, we find techniques like *throttling*, which may be employed to reduce SSD temperature, to be effective at reducing the failure rate of SSDs. We also find that SSD temperature is correlated with the power used to transmit data across the PCIe bus, which can potentially be used as a proxy for temperature in the absence of SSD temperature sensors.

Observation 5: *The amount of data reported to be written by the system software can overstate the amount of data actually written to flash chips, due to system-level buffering and wear reduction techniques.*

Techniques that simply reduce the *rate* of software-level writes may not reduce the failure rate of SSDs. Studies seeking to model the effects of reducing software-level writes on flash reliability should also consider how other aspects of SSD operation, such as system-level buffering and SSD controller wear leveling, affect the *actual* amount of data written to SSDs.

Conclusions. We hope that our new observations, with real workloads and real systems from the field, can aid in (1) understanding the effects of different factors, including system software, applications, and SSD controllers on flash memory reliability, (2) the design of more reliable flash architectures and systems, and (3) improving the evaluation methodologies for future flash memory reliability studies.

9. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments. This work is supported in part by Samsung and the Intel Science and Technology Center for Cloud Computing, as well as NSF grants 0953246, 1065112, 1212962, and 1320531.

10. REFERENCES

[1] NVM Express Specification. http://www.nvmexpress.org/specifications/.

[2] The R Project for Statistical Computing. http://www.r-project.org/.

[3] American National Standards Institute. AT Attachment 8 – ATA/ATAPI Command Set. http://www.t13.org/documents/uploadeddocuments/docs2008/d1699r6a-ata8-acs.pdf, 2008.

[4] H. Belgal, N. Righos, I. Kalastirsky, et al. A New Reliability Model for Post-Cycling Charge Retention of Flash Memories. IRPS, 2002.

[5] A. Brand, K. Wu, S. Pan, et al. Novel Read Disturb Failure Mechanism Induced By Flash Cycling. IRPS, 1993.

[6] Y. Cai, E. F. Haratsch, O. Mutlu, et al. Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis. In *DATE*, 2012.

[7] Y. Cai, E. F. Haratsch, O. Mutlu, et al. Threshold Voltage Distribution in MLC NAND Flash Memory: Characterization, Analysis, and Modeling. In *DATE*, 2013.

[8] Y. Cai, Y. Luo, S. Ghose, et al. Read Disturb Errors in MLC NAND Flash Memory: Characterization and Mitigation. In *DSN*, 2015.

[9] Y. Cai, Y. Luo, E. F. Haratsch, et al. Data Retention in MLC NAND Flash Memory: Characterization, Optimization and Recovery. In *HPCA*, 2015.

[10] Y. Cai, O. Mutlu, E. F. Haratsch, et al. Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation. In *ICCD*, 2013.

[11] Y. Cai, G. Yalcin, O. Mutlu, et al. Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Lifetime. In *ICCD*, 2012.

[12] Y. Cai, G. Yalcin, O. Mutlu, et al. Error Analysis and Retention-Aware Error Management for NAND Flash Memory. *ITJ*, 2013.

[13] Y. Cai, G. Yalcin, O. Mutlu, et al. Neighbor-Cell Assisted Error Correction for MLC NAND Flash Memories. In *SIGMETRICS*, 2014.

[14] A. Chimenton and P. Olivo. Erratic Erase in Flash Memories – Part I: Basic Experimental and Statistical Characterization. *IEEE Trans. Elect. Dev.*, 50(4), 2003.

[15] T.-S. Chung, D.-J. Park, S. Park, et al. A survey of flash translation layer. *J. Sys. Arch.*, 55, 2009.

[16] C. Compagnoni, A. Spinelli, R. Gusmeroli, et al. First Evidence for Injection Statistics Accuracy Limitations in NAND Flash Constant-Current Fowler-Nordheim Programming. IEDM Tech Dig., 2007.

[17] J. Cooke. The Inconvenient Truths of NAND Flash Memory. In *Flash Memory Summit*, 2007.

[18] R. Degraeve, F. Schuler, B. Kaczer, et al. Analytical Percolation Model for Predicting Anomalous Charge Loss in Flash Memories. *IEEE Trans. Elect. Dev.*, 51(9), 2004.

[19] A. Gartrell, M. Srinivasan, B. Alger, et al. McDipper: A Key-Value Cache for Flash Storage. https://www.facebook.com/notes/10151347090423920, 2013.

[20] L. M. Grupp, J. D. Davis, and S. Swanson. The Bleak Future of NAND Flash Memory. In *FAST*, 2012.

[21] S. Hur, J. Lee, M. Park, et al. Effective Program Inhibition Beyond 90nm NAND Flash Memories. NVSM, 2004.

[22] S. Joo, H. Yang, K. Noh, et al. Abnormal Disturbance Mechanism of Sub-100 nm NAND Flash Memory. *Japanese J. Applied Physics*, 45(8A), 2006.

[23] T. Jung, Y. Choi, K. Suh, et al. A 3.3V 128Mb Multi-Level NAND Flash Memory for Mass Storage Applications. ISSCC, 1996.

[24] M. Kato, N. Miyamoto, H. Kume, et al. Read-Disturb Degradation Mechanism Due to Electron Trapping in the Tunnel Oxide for Low-Voltage Flash Memories. IEDM, 1994.

[25] H. Kurata, K. Otsuga, A. Kotabe, et al. The Impact of Random Telegraph Signals on the Scaling of Multilevel Flash Memories. VLSI, 2006.

[26] J. Lee, J. Choi, D. Park, et al. Degradation of Tunnel Oxide by FN Current Stress and Its Effects on Data Retention Characteristics of 90-nm NAND Flash Memory. IRPS, 2003.

[27] J. Lee, C. Lee, M. Lee, et al. A New Program Disturbance Phenomenon in NAND Flash Memory by Source/Drain Hot-Electrons Generated by GIDL Current. NVSM, 2006.

[28] A. Maislos. A New Era in Embedded Flash Memory, 2011. Presentation at Flash Memory Summit.

[29] Micron. Garbage Collection in Single-Level Cell NAND Flash Memory. Technical report, 2011.

[30] N. Mielke, H. Belgal, A. Fazio, et al. Recovery Effects in the Distributed Cycling of Flash Memories. IRPS, 2006.

[31] N. Mielke, H. Belgal, I. Kalastirsky, et al. Flash EEPROM Threshold Instabilities due to Charge Trapping During Program/Erase Cycling. *IEEE Trans. Dev. and Mat. Reliability*, 2(3), 2004.

[32] N. Mielke, T. Marquart, N. Wu, et al. Bit Error Rate in NAND Flash Memories. In *IRPS*, 2008.

[33] T. Ong, A. Fazio, N. Mielke, et al. Erratic Erase in ETOX™ Flash Memory Array. VLSI, 1993.

[34] J. Ouyang, S. Lin, S. Jiang, et al. SDF: Software-Defined Flash for Web-Scale Internet Storage Systems. ASPLOS, 2014.

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

[36] K. Suh, B. Suh, Y. Lim, et al. A 3.3 V 32 Mb NAND Flash Memory with Incremental Step Pulse Programming Scheme. *IEEE JSSC*, 30(11), 1995.

[37] K. Takeuchi, S. Satoh, T. Tanaka, et al. A Negative Vth Cell Architecture for Highly Scalable, Excellently Noise-Immune, and Highly Reliable NAND Flash Memories. *IEEE JSSC*, 34(5), 1995.

[38] A. Thusoo, J. Sen Sarma, N. Jain, et al. Hive – A Petabyte Scale Data Warehouse Using Hadoop. In *ICDE*, 2010.

[39] M. Xu, C. Tan, and L. MingFu. Extended Arrhenius Law of Time-to-Breakdown of Ultrathin Gate Oxides. *Applied Physics Letters*, 82(15), 2003.

[40] R. Yamada, Y. Mori, Y. Okuyama, et al. Analysis of Detrap Current Due to Oxide Traps to Improve Flash Memory Retention. IRPS, 2000.

[41] R. Yamada, T. Sekiguchi, Y. Okuyama, et al. A Novel Analysis Method of Threshold Voltage Shift Due to Detrap in a Multi-Level Flash Memory. VLSI, 2001.