electronics

MDPI

Article

iLife: Safely Extending Lifetime for Memory-Oriented SSD

Erci Xu [1,*] and Shanshan Li [2]

[1] Engineering Center of Foundational Software, College of Computer Science, National University of Defense Technology, Changsha 410073, China

[2] Department of Computer Science, College of Computer Science, National University of Defense Technology, Changsha 410073, China; shanshanli@nudt.edu.cn

* Correspondence: xuerci@nudt.edu.cn

Received: 26 April 2019; Accepted: 28 May 2019; Published: 30 May 2019

Abstract: Currently, the roles of SSDs have been diversified significantly. Apart for storage purposes, users can also utilize flash drives to cache hot data or buffering incoming writes to achieve high throughput. In this case, the endurance study of Solid State Drives (SSDs) has become an increasingly important topic as users rely on such research to reliably avoid early retirement of their old drives in laptops or machines in data centers. However, current manufacturers adopt Program/Erase (P/E) cycle-based life indicators to estimate the remaining lifetime of an SSD, which is often too conservative due to longtime data retention concerns. In this paper, we begin by analyzing the inaccuracy and other potential issues of existing P/E cycle-based life indicators for SSDs under memory-oriented workloads. To construct an accurate life indicator, we conduct a more extensive and sophisticated experiment to wear out eight SSDs from four different vendors. By monitoring the device status extracted from the standard S.M.A.R.T.attributes in the experiment, we make eleven interesting findings on the relationship between the device lifetime and different types of errors observed. Based on our unique findings, we propose iLife, a pattern-based life indicator for SSDs under memory-oriented workloads. Our evaluation of iLife on a different set of six SSDs shows that iLife is a more accurate life indicator, with an average of 14.2% higher accuracy on lifetime estimation and up to 21.2% safe lifetime extension compared to existing P/E cycle-based indicators.

Keywords: SSD; endurance; reliability

1. Introduction

Thanks to its high performance and low power draw, flash-based Solid State Drives (SSD) have become the staple in today's large-scale storage systems. Moreover, unlike traditional disks, which are dedicated to data persistence, the functionalities of SSD are versatile in modern storage systems. For example, practitioners can use SSDs to buffer incoming writes to improve the performance of the back-end storage clusters [1,2] or adopt them as caches to store intermediate data of analytic workloads, which are too big for memory to digest completely [3]. In such cases, SSDs are treated more like "larger memories" rather than "faster disks".

Nonetheless, the current commodity SSDs, while widely used in many data centers [3–5], are not fully ready for such "memory-oriented" tasks. The root cause is that manufacturers tend to be conservative regarding the lifetime of SSDs as general-purposed SSDs need to balance the trade-off between usage and data retention capability. SSDs are usually equipped with an indicator (i.e., the wear index in S.M.A.R.T.attributes) to inform the users about the remaining lifetime based on a conservative estimation. Recent studies showed that, for cache-like workloads (i.e., short-lived data with intensive I/O), the actual lifetime of SSDs can be twice more than the estimation [6]. As a result, if administrators simply adopt the default indicator for memory-oriented workloads, the SSDs can be retired in a premature manner, which can lead to significant economical costs. Moreover, if the users

continue to use the SSDs by ignoring the indicator alarms, the SSDs can encounter various failures including performance slowdown and even data losses. Therefore, it is crucial to study the endurance and behaviors of SSDs under memory-oriented workloads and, hence, to extend SSDs' lifetime safely.

2. Background

2.1. Wear of Flash Memory

Flash memory is the major component in flash-based SSDs for storing data. In this section, we briefly summarize the underlying mechanism leading to the wear of flash memory, which fundamentally limits the endurance of SSDs.

Figure 1 shows the structure of an NAND flash memory cell [7,8]. To represent digital data in this cell, different numbers of electrons are withheld in the floating gate, which are isolated by the oxide layers. By applying a different voltage on the Control Gate, electrons can be injected into or ejected out of the floating gate, which corresponds to the program and erase operations to the cell, respectively. For each program or erase operation, the electrons go through the oxide layer via the Fowler–Nordheim tunneling effect [9,10].

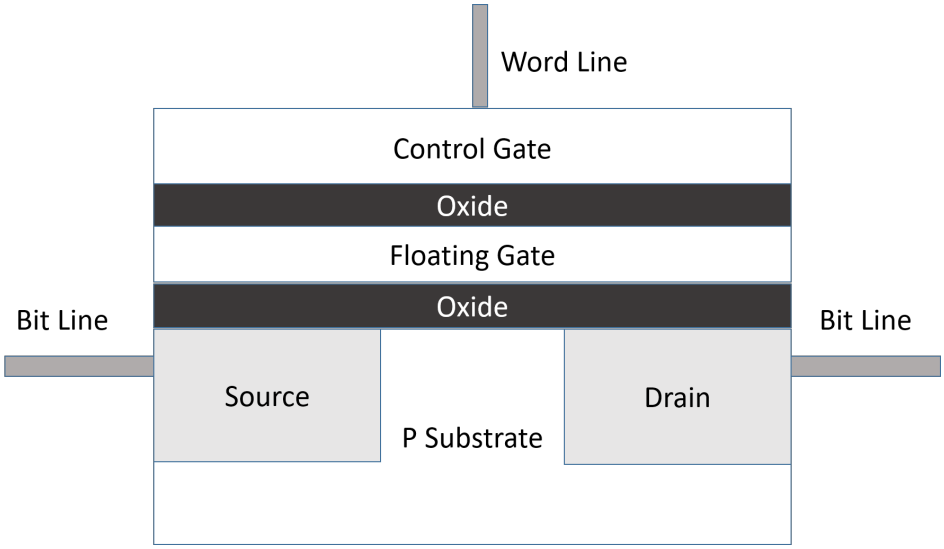

Figure 1. The structure of an NAND flash memory cell.

The tunneling effect gradually breaks down the oxide layer. Consequently, more and more electrons are "stuck" in the oxide layer, which eventually make the cell unable to store information. Therefore, the lifespan of an NAND flash memory cell can be estimated by using the number of Program/Erase cycles (i.e., P/E cycles) [11].

NAND flash memory cells in an SSD device are typically organized into erase blocks and pages, which are large-sized chunks that are physically linked. An erase block is a physically contiguous set of cells (typically 2–4 MB) that can only be erased all together. A page is a physically contiguous set of cells (typically 4–32 KB) that can only be written to as a unit. Because of such coarse granularity of operations, neighboring cells within the same page or block tend to have a similar usage rate, and thus a similar degree of wear.

2.2. Endurance of SSDs

Besides the flash memory, SSDs include sophisticated firmware, called a Flash Translation Layer (FTL), to make the device appear as a simple block device. Internally, FTLs implement a variety of algorithms [12] to manage the hardware and protect against errors. Moreover, there are device-level

features to alleviate the wear of flash memory. We highlight several commonly-used techniques as follows, which directly affect the endurance of SSDs:

Error Correction Code (ECC): ECCs (e.g., BCH [13,14] and LPDC [13] codes) are typically stored in the spare area of blocks and are examined during read operations to detect and correct the potentially corrupted bits within pages. Therefore, the endurance of blocks depends on the strength of the ECC (i.e., the number of correctable bits) used in the device.

Over-provisioning: The actual capacity of flash memory in SSDs is usually bigger than the capacity labeled on the device [15]. This over-provisioned space, which is invisible to the host machine, can serve as the backups for bad blocks, among other purposes.

Wear leveling: Because flash memory cells can only sustain a limited number of P/E cycles, it is desirable to even out the usage of flash cells across the entire SSD device. Therefore, FTLs usually implement wear leveling to balance the writes across SSDs.

In addition, some FTLs may use advanced algorithms to compress data [16] or reduce write amplifications [17], which makes estimating the endurance of SSDs even more challenging.

3. Workload Analysis

In this section, we extensively study the dataset collected from a large-scale storage system where the SSDs are both utilized for caching and persistence purposes. By comparing the impacts from two different workloads, we have made several observations that further guided us to design the testing frameworks.

3.1. Dataset Overview

Our dataset was obtained from a major cloud provider [3]. The dataset consisted of three services, cloud block service, hybrid storage service, and big data analytics. In the cloud block service, a node contained 12 SSDs, which were dedicated to persist users' data. In the hybrid storage service, a node consisted of two SSDs and 36 HDDs where the SSDs buffered all incoming writes and cache frequently accessed data. In big data analytic workloads, a node only has one SSD, which is regarded as a temporary holder for intermediate data during shuffling stages in the map-reduce processes. The SSDs of the hybrid storage service and big data analytics can be assumed as under memory-oriented workloads.

Moreover, the dataset included one month of traces from around 130,000 SSDs where 53.7% of them were used for cloud block service, 12.4% of them were used for hybrid storage service, and the rest were for big data service. The traces included iostats every 15 s (obtained by the querying OS kernel), the daily-collected SMARTattributes and the customized metrics of the service. With three data sources, we were able to observe the differences of the persistence-based workloads (i.e., cloud block storage service) and memory-oriented workloads (i.e., the other two services) in various data characteristics and the impacts on the SSDs (i.e., wear introduced by writing the data).

3.2. Workload Characteristics

Data characteristics: We focus on three important aspects of our dataset, including sizes, I/O patterns, and liveness. By comparing the SSDs of the three use cases from the three perspectives, we can observe the differences posed by the workloads.

First, regarding the sizes, we measured the sizes of data that were written to the SSDs. We quantitatively measured the distribution of data sizes for SSDs in each service as shown in Figure 2. We may observe that the distribution varied greatly among different services. For cloud block services and hybrid storage service, most data fell into ranges of 1 MB to 1 GB, which is in accordance with commonsense data sizes under daily usages. For the big data analytic storage service, the data sizes became much larger as the service usually handles large-scale workloads and the SSDs are used to digest the data that are unable to be digested by the memories.

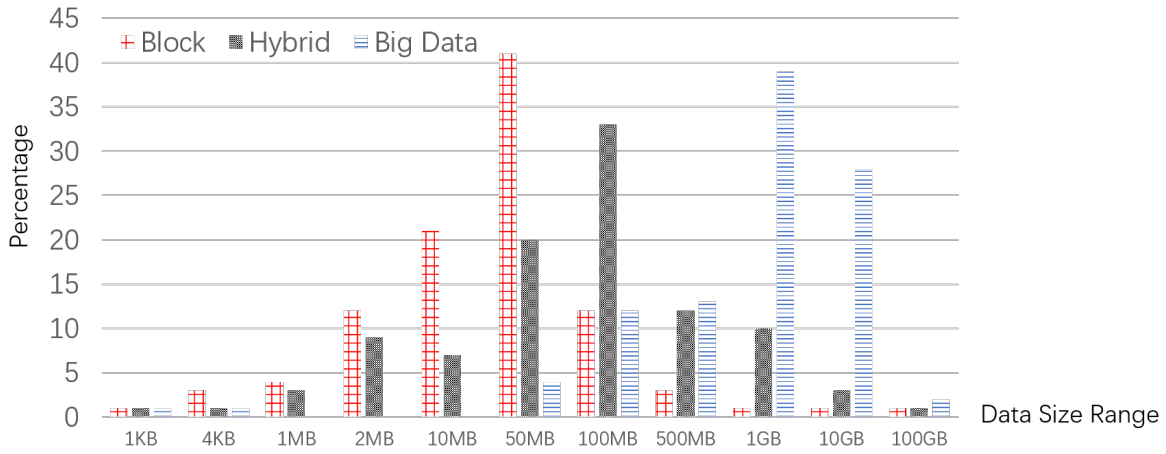

Figure 2. Distribution of data sizes under three workloads. The X axis is the data size range. For example, 4 KB includes data sizes ranging from 1 KB–4 KB. The Y axis is the percentage of each data size range.

Second, for I/O patterns, as shown in the Table 1, we categorized the I/O requests in terms of sizes ranging from 4 KB (including smaller), 128 KB–2 MB, and 2 MB (including ones beyond 2 MB). As 4 KB is the page size and 2 MB is the size of block, the first group can be considered as random writes. and the rest are sequential writes (at the page-level and block-level). As the table shows, for cloud block storage service and hybrid storage services, up to 11.3% of their writes are random. On the contrary, in the big data analytic service, only 0.1% of its workloads are random. The reason behind the difference is that, unlike files accessed in daily usage scenarios, most data in the big data processing service are large files and thus incur much less random writes.

Table 1. Distribution of I/O patterns under three workloads.

Categorization	Range	Block Storage	Hybrid Storage	Big Data
Random	1–4 KB	9.4%	7.2%	1.7%
Sequential	4 KB–2 MB	13.5%	20.3%	0.1%
	>2 MB	77.1%	72.5%	98.2%

Third, we also studied the duration of data in SSDs under each workload. The duration starts as a certain piece of data is written to a certain range of SSD's LBA (Logical Block Address) and ends as the data either have been deleted by the users or flushed by the system. In Figure 3, we plot the average data duration among SSDs under the three workloads, and we may observe that the liveness of data in memory-oriented workloads was much shorter. In hybrid storage workload, 97.8% of the data existed less than 48 hours, and more than half of them (54.1%) spent less than four hours. Regarding the data liveness in the big data analytic workload, the duration was even shorter, as most (i.e., 93.2%) data lasted less than one hour in the SSDs. The reasons for the short data liveness in the memory-oriented workloads are two-fold. First, for SSDs in the hybrid storage service, the two SSDs need to buffer all incoming writes for the back-end 36 HDDs. Usually, the spaces are quickly consumed, and the system thus needs to evict cold data while retaining some hot ones. Second, for big data analytic workloads, the data are temporarily stored and would be quickly removed once the data need to be further processed.

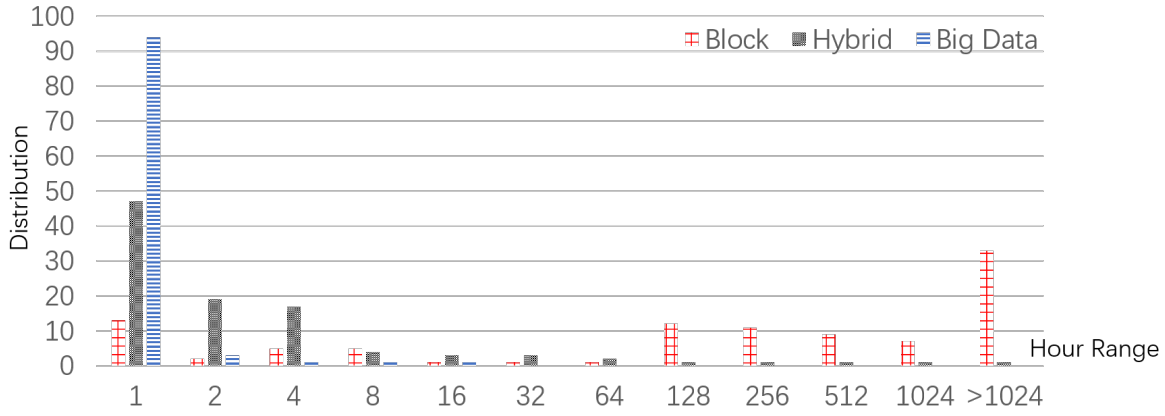

Figure 3. Distribution of workload durations.

Data impacts: We also studied the impacts by examining the SMART attributes' difference before and after running a month of the workloads. While other attributes were also studied, we focused on the wear indicator as users generally refer to this index for estimating the remaining life of the SSDs. As shown in Table 1, we can observe that the wear difference under memory-oriented workloads were much larger than persistence-based workloads. The root cause is that for SSDs under memory-oriented workloads, they need to endure much more writes, which are either the sum of all writes in the backend HDDs or the writes incurred by processing big data workloads.

Comparison with 3D NAND-based SSD: In the study, all our candidates were 2D MLC memory-based flash. Currently, enterprises have become more interested in deploying 3D NAND-based flash SSDs due to their more economic prices and larger storage density. In order to compare the difference between the two generations, we also studied a newly-deployed cluster with around 300 3D flash SSDs. We found that 3D flash drives did not exhibit noticeable differences in workloads including IO patterns, data liveness, and sizes. After standardization, the difference was usually within 5%. We assumed, even with different intrinsic components, that the data allocation scheme currently is not affected by the such difference.

3.3. Conclusion

Based on our analysis of the dataset, we conclude about three major characteristics of memory-oriented workloads:

- **Diverse sizes and I/O patterns**: We observed that the memory-oriented workloads can introduce data with various sizes and both random and sequential writes. This reflects that we may not simply use large-file-based workloads and further repeatedly writing them to estimate the remaining life of SSDs. Rather, a detailed and realistic workload that combines different data sizes and I/O patterns is desired.
- **Short liveness**: Another important feature of the memory-oriented workloads is their short data duration. From our analysis, we found that, unlike persistence-based workloads where data need to be stored for years, the data in memory-oriented workloads usually last for hours, and at most days. The reason behind this phenomenon is that users mainly leverage the SSDs for their high performance instead of their capability of persisting data. Therefore, under such scenarios, we may conclude that the data retention is no longer the priority.
- **Severe impacts**: After comparing the wear index difference between the two workloads, we observed that memory-oriented SSDs were more receiving far more writes and thus led to more severe wear impacts. Therefore, simply referring to the default wear index, the SSDs are likely to be replaced in a premature manner. Thus, an extensive endurance study that enables the users to extend memory-oriented SSDs' lifetime safely is desperately needed.

4. Endurance Testing Framework

Although the experiment in the previous study was effective at exposing that SSDs may function well long after the indicator reached zero, this is insufficient for analyzing the endurance of a device thoroughly. First of all, the workload writes data through a file system [6,18]. However, the file system may also mask some raw device behaviors (e.g., a corrupted journal block in Ext4may never be visible by the end user). Secondly, the dataset only includes large files, and the access pattern is mostly sequential, which is not aligned with our study in the previous section. As a result, the observed behavior of a tested device may not be general. To address these limitations, we designed a comprehensive testing framework as follows.

4.1. Basic Design of the Workload

At the very high level, our workload bypasses the file system and accesses the SSD as a raw block device. It keeps writing and checking a set of carefully-designed records. More specifically, the workload can be described from the following three perspectives, which are summarized in Tables 2–4, respectively.

Table 2. Distribution of writes in terms of size.

Size (KB)	Percentage
0.5	4%
1, 1.5, 2, 2.5, 3,3.5	1%
4	67%
8	10%
16	7%
32, 64	3%

Table 3. Distribution of writes in terms of location. LBA, Logical Block Address.

Location	Percentage
first 5% of LBA space	50%
next 15% of LBA space	30%
the rest	20%

Table 4. Format of a record.

Field	Purpose
checksum	detect corruption
size	record size
range	one of the three LBA ranges in Table 3
LBA	the first LBA of the record
seed	a rand (randomized seed).seed for generating the padding
padding	pad the record to the desired size

Size of write: The dataset in the workload consisted of individual records, the sizes of which ranged from 0.5 KB–64 KB. Each record was written to the device using one single write system call, so different record sizes led to different sizes of write operations. In addition, the percentage of each size among all writes is shown in Table 2. The distribution was based on JESD-219 [19], an industry standard maintained by JEDEC [20] for generating SSD endurance testing workloads. As records that are larger than 8 KB can be assumed as sequential writes, our workload, in this way, represents a realistic mix of sequential and random write operations.

Location of write: Each write is issued to a particular location within the space of LBA. Due to the sophisticated mapping and management of the FTL (Section 2), however, writing to different ranges of LBA may incur different wear for SSDs. Therefore, we intentionally distributed 50% of writes to the first 5% of the LBA space, 30% of writes to the next 15% of the LBA space, and 20% of writes to the rest space. These parameters were also based on JESD-219 [19].

Format of record: Besides following the JESD-219 [19] guideline, we designed a special record format to enable easy detection of fine-grained device failures. As shown in Table 4, each record contained a checksum to detect corruption within the record. Furthermore, the record included self-identification information (e.g., the expected range and LBA) to allow detection of misplacement of the record (e.g., flying writes [21,22]), which may pass the examination of the checksum. The other fields (e.g., size and rand) allowed us to regenerate the exact record for verification uniquely.

With the specifications defined above, the overall work-flow was as follows. Each *iteration of the workload* started with one writing procedure and ended with one checking procedure, which scanned the whole device, unless a device failure was observed (Section 4.3). In the writing procedure, the workload wrote records satisfying the size, location, and format specifications to the target device. After each write operation, we collected the S.M.A.R.T. attributes once. The writing procedure ended when the total amount of data written during the procedure reached the labeled capacity of the target device.

After the writing procedure finished, the checking procedure scanned the whole address space of the device and checked the integrity of every on-drive record. In addition, during the writing procedure, overriding also occurred as 80% data of each workload iteration needs to be written in only 20% of the LBA addresses. When an unchecked record is about to be overwritten, the checking is also triggered to verify that particular record.

In both cases, the checking procedure examines the checksum of the record(s) first. If the checksum exam failed, we regenerated the expected content of the record and performed a bitwise comparison to pinpoint the exact bits of the corruption. In addition, we collected the S.M.A.R.T. attributes after each read operation in the checking procedure for further analysis.

4.2. Optimizations of Workload

Our basic deign of workload strictly followed JESD-219 [19] and enabled fine-grained checking. However, the workload was inefficient for wearing out SSDs. This is mainly because of two reasons. First, as shown in Table 2, the majority of writes was smaller than 16 KB, which led to poor performance during the writing procedure as SSDs favor large sequential writes. Second, the checking procedure may be invoked during the working procedure for examining the unchecked to-be-overwritten record. This interference makes the writing procedure even slower. For example, without any optimizations, D2 would require approximately 320 days to finish.

To accelerate the endurance testing, we optimized the workload in two ways. First, if the S.M.A.R.T. attributes monitored did not show any abnormality, which is typical for devices in their early lifespan, we clustered the small data entries in the form of 16-KB writes during the writing procedure. Once we observed any suspicious changes in S.M.A.R.T. attributes (e.g., a program error occurred), we switched back to the basic distribution of writes. This optimization enabled us to achieve both ends: accelerating the wear of the device at their early stages and being able to analyze the device behaviors with fine granularity when its endurance started becoming questionable.

The second optimization is to minimize the interference of the checking procedure when it is interleaved with the writing procedure. We buffered the records read in the checking procedures without checking them immediately. When the buffer reached a certain threshold (e.g., 2 MB), the checking procedure examined the buffered records all together asynchronously. In this way, the writing procedure can continue with minimal disturbance.

4.3. Termination of Workload

The workload writes the records and checks the integrity of the records iteratively, until one device failure is observed. As mentioned in Section 3, we defined device failures as incorrect behaviors that are perceivable by end users.

Our current prototype was able to detect the following three types of failures and terminate the workload properly:

- **Read Only (RO):** This is one safety mechanism provided by some SSDs to prevent users from further writing when the device becomes unreliable.
- **Device Not Found (DNF)**: The device cannot be recognized by the host operating system.
- **Data Corruption (DC)**: any corruptions of records detected by the checking procedure of the workload.

4.4. Monitoring Device Status

During the workload, we monitored the status of the device based on the S.M.A.R.T. attributes. In addition, we monitored the performance, as well as the failure behavior of the device. We highlight the monitored parameters that were most relevant to the endurance of the device as follows:

- **Program Errors (PEr):** the count of failed program operations in the flash controller since the drive was deployed. A failed program operation may trigger another program operation on a new page, which makes the failed operation transparent to the end user.
- **Erase Errors (EEr):** the count of failed erase operations in the flash controller since the drive was deployed. Similar to the PEr case, a failed erase operation may be transparent to the end user.
- **Correctable Errors (CEr):** the count of internal errors that can be recovered using ECC.
- **Uncorrectable Errors (UCEr):** the count of internal errors that cannot be recovered using ECC. This type of error may lead to user-visible corruption.
- **Cyclic Redundancy Check Errors (CRC):** the count of errors in data transfer via the interface between the host and the device.
- **Temperature:** the case temperature of SSD reported by the SMART attributes.
- **Performance Jitters (Jitter):** the count of unusual performance slowdowns.
- **Device Failures (Fail):** This is the final device failure, which terminates the workload. As discussed in Section 4.3, there are three possible types including Read Only (RO), Device Not Found (DNF), and Data Corruption (DC).

4.5. Target Devices and Testing Environment

We carefully selected a set of representative SSDs from the market as our target devices for testing. As shown in the first column of Table 5, the devices were from four different manufacturers (i.e., A, B, C, D), and there were two devices from each manufacturer (e.g., A1 and A2). The capacity (i.e., the "Size" column) of the devices ranged from 20 GB–120 GB. The Lithography type (i.e., the "Lith." column) included both 25 nm and 20 nm, which represents the relatively mature flash technology.

We used four machines to run the experiments in parallel. Each machine had an Intel Core-i5 CPU, 16 GB memory, and SATA III interfaces for connecting the target devices. The operating system was Ubuntu 14.04 with smartmontools [23] installed. The target devices were not used for any other purpose during the experiment.

The external environment is known to be critical for SSD lifetime [11]. In order to simulate the environment of the current data center setup, we used a 12-slot server node to install all SSDs, and the server was placed at the inlet of the air conditioner with the temperature set at 25 °C.

One concern was that the recent adoption of using 3D NAND-based flash has several intrinsic differences from the products studied and tested in this experiments. Currently, we are focusing on enhancing the indicator accuracy among our deployment, which are still 2D-based MLCSSD products. Yet, we assumed the behaviors of 3D NAND flash may behave quite differently. Therefore, we leave that as part of our future work.

5. Results and Analysis of the Endurance Testing

5.1. Overall Results

Table 5 summarizes the overall results of our endurance testing experiments. All devices were tested from the brand new status to device failure, and all of them indicated the lifetime via the "wear out indicator" attribute of S.M.A.R.T, which we refer to as the *default indicator*. We highlight four major findings as follows:

Finding #1: The actual lifetime of all devices was much longer than that reported by the default indicators. The last two columns of Table 5 compare the lifetime reported by the default indicators and the actual lifetime, both of which are presented using the number of workload iterations. We can see that the actual lifetime well exceeded the indicated lifetime. For example, the indicated lifetime of C2 was 3396 workload iterations, while its actual lifetime was 7013, which is 207% of the indicated one. On average, the actual lifetime was 171% of the indicated lifetime, which implies that the default indicators were generally very conservative and may waste a significant amount of device lifespan.

Finding #2: Some indicators may report misleading values. During the workload, all indicators reported the lifetime as a decreasing positive value, which gradually reached zero. However, after reporting the zero lifetime, the indicators of two devices (i.e., C1, C2) overflowed to a garbage value, which could mislead users about the device lifetime.

Finding #3: When devices eventually failed, they failed in different manners and caused different amounts of data loss. As shown in the "Fail" column, four devices (i.e., A1, A2, B2, D2) ended with Device Not Found (DNF), three devices (i.e., B1, C1, C2) corrupted user data (DC), and one device (D1) became Read Only (RO). Even the devices from the same manufacturer (e.g., B1 and B2) may behave differently (e.g., DC versus DNF). Among all the devices, only D1 was able to fail gracefully without causing any data loss.

Finding #4: The internal status of the devices varied greatly. The fourth–ninth columns (i.e., "PEr" to "Jitter") show the final values of the parameters monitored by our framework, which revealed the internal status of the devices. We can see that these values were not consistent even within devices from the same manufacturer (e.g., B1's PEr was 207, while B2's PEr was 46). Moreover, some devices (e.g., D1 and D2) did not report all parameters. We analyze the trends of these parameters with more details in the next section.

Table 5. Overall results of endurance testing. **Lith**: Lithography; **PEr**: Program Error count; **EEr**: Erase Error count; **CEr**: Correctable Error count; **UCEr**: Uncorrectable Error count; **CRC**: Cyclic Redundancy Check (CRC) error count; **Jitter**: count of performance slowdown; **Fail**: Failure type; **DNF**: Device Not Found; **RO**: Read Only; **DC**: Data Corruption; **Overflow**: whether the lifetime reported by the default indicator overflows (**Y**) or not (**N**) after reaching zero; **Indicated Lifetime**: the lifetime reported by the default indicator in terms of workload iteration count; **Actual Lifetime**: the actual lifetime in terms of workload iteration count and the relative percentage of the indicated lifetime.

SSD	Size (GB)	Lith (nm)	PEr	EEr	CEr	UCEr	CRC	Jitter	Fail	Over-flow?	Indicated Lifetime	Actual Lifetime
A1	32	25	43	14	246,852	0	0	0	DNF	N	3437	5129(149%)
A2	64	25	44	50	423,251	0	1	0	DNF	N	3293	6002 (182%)
B1	64	25	207	399	792,288	0	0	0	DC	N	2842	3165 (111%)
B2	64	25	46	84	520,361	0	0	0	DNF	N	3164	5563 (175%)
C1	64	20	92	13	209,422	9	0	8	DC	Y	3252	6706 (206%)
C2	64	20	47	18	418,754	1	0	2	DC	Y	3396	7013 (207%)
D1	20	20	N/A	N/A	63,407	N/A	15	4	RO	N	2411	3360 (139%)
D2	120	20	N/A	N/A	197,376	N/A	0	3	DNF	N	2519	4914 (195%)

5.2. Analysis of Individual Parameters

To understand the device behavior in more detail, we analyzed the trend of each parameter during the whole lifetime of the devices. We summarize our findings on PEr, EEr, CEr, and jitter (performance jitter) in this section as they are most prevalent among the devices.

Finding #5: Program and erase errors were prevalent, and tended to occur in batches towards the end of device lifetime. Program and erase errors were observed in all devices except for D1 and D2, which did not disclose related attributes. Moreover, we found that the errors tended to occur in batches when the devices were reaching the end of their lifetime. Two examples are shown in Figure 4, which reveals the trend of PEr and EEr during the whole lifetime of A1 and A2. We can see that immediately after the 3188th workload iteration, the PEr of A1 suddenly increased from zero to more than 20. We defined this behavior as *surging*, where a batch of errors occurs in one or more consecutive workload iterations.

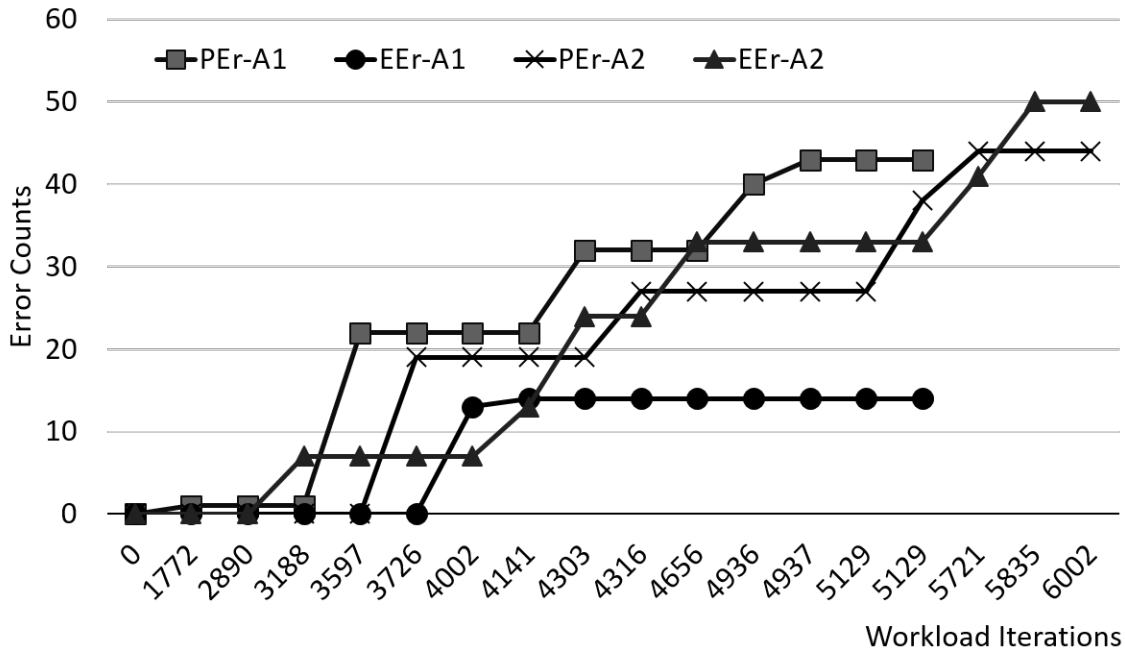

Figure 4. Trend of PEr and EEr during the whole lifetime of A1 and A2. **PEr-A1**: program error count of A1; **EEr-A1**: erase error count of A1; **PEr-A2**: program error count of A2; **EEr-A2**: erase error count of A2; **X axis**: pinpoint workload iteration of error occurrence; intervals are not uniformed.

Several reasons may contribute to this behavior. As mentioned in Section 2, flash cells wear out gradually as more electrons are "stuck" in the oxide layer. Therefore, the program and erase errors are unlikely to happen at the early stage of the device lifetime. Furthermore, for performance reasons, the FTL may apply wear leveling and other algorithms in groups of blocks, which means the blocks within the same group (and the pages within in the same block) tend to have a similar usage rate. As a result, when an erase or program error occurs, a retry on the neighboring block or page is likely to generate a similar error.

Finding #6: Correctable errors appeared most, and they exhibited both similarity and differences among all devices. All drives reported correctable errors, and we have observed a large amount of CErs on every device. Figure 5A–D show the trend of CEr on the eight devices. On the one hand, we can see that the CEr increased slowly in the early stage of all devices. For example, C1 and C2 in Figure 5C had less than 10% of errors in the first 40% of their lifetime. Furthermore, there was *surging* on all devices where the number of errors appeared in batches.

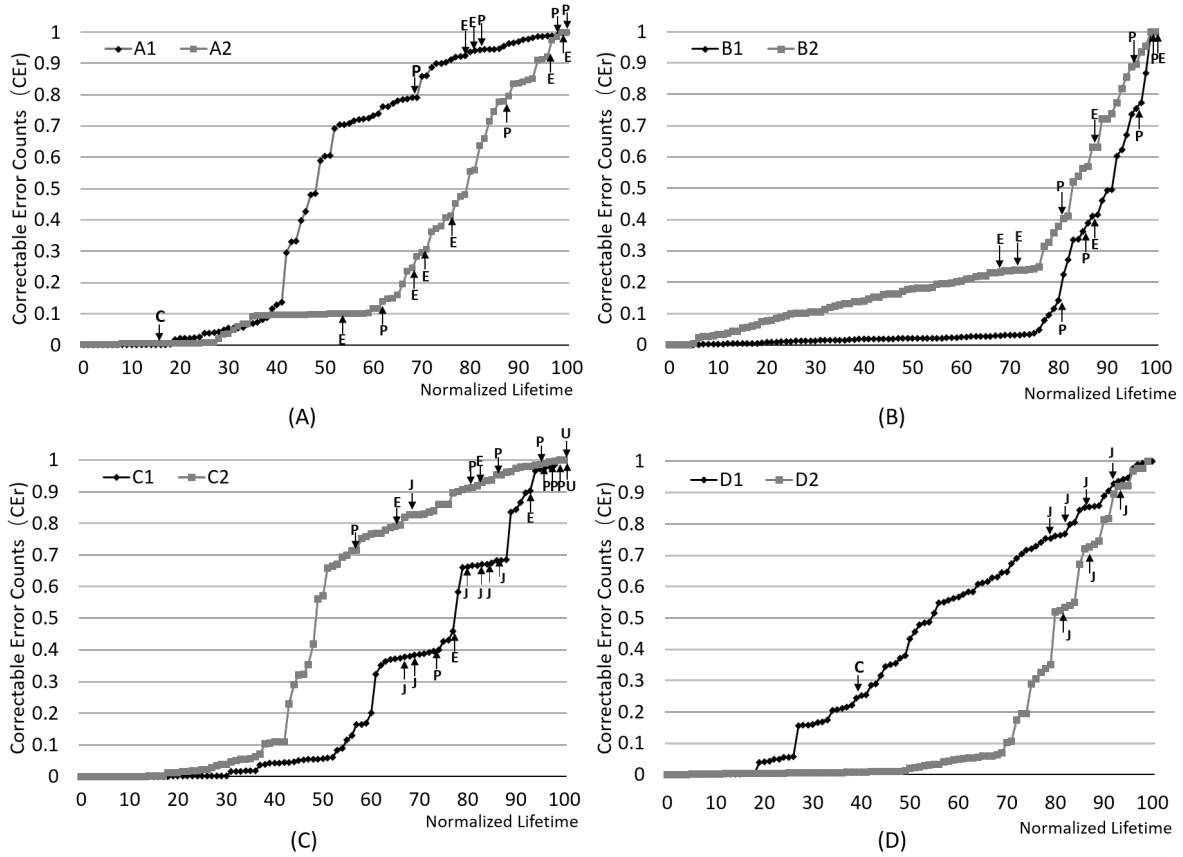

Figure 5. Trend of **CEr** during the whole lifetime of eight drives (e.g., A1 and A2 in (**A**)). P and E mean the surging of Program errors and Erase errors, respectively. U indicates the occurrence of Uncorrectable error. J and C correspondingly indicate the occurrence of performance Jitter and CRC error.

On the other hand, the timing, as well as the amount of errors in the *surging* period may be different even between the devices with the same capacity from the same manufacturer. For example, in Figure 5C, C2's *surging* period started at its 40% lifetime and reached about 0.7 around its 50% lifetime, while C1's first *surging* started after its 50% lifetime and reached less than 0.4 at its 60% lifetime.

Finding #7: CRC errors were not directly related to device failures. Two drives, A2 and D1, experienced two occurrences of CRC errors as shown in the CRC column of Table 5 and illustrated with mark *C* in Figure 5A and Figure 5D respectively. Both were manifested in the early stage of device lifespan, and no other abnormal status was observed after the CRC errors, which implies that this type of error does not directly contribute to device failures. This is because CRC errors are usually caused by the unstable communication channels between the host and the device and may be resolved simply by a retry.

Finding #8: Uncorrectable errors always led to device failures. Experiments on C1 and C2 were terminated by the checking procedure of our framework with bit corruptions observed in our records. Immediately before observing the corruptions, there were uncorrectable errors detected as illustrated with mark *U* in Figure 5C. Therefore, a non-zero uncorrectable error count is a strong indication of an imminent device failure.

Finding #9: Performance may not necessarily decrease when the device reaches the end of its lifetime. As shown in the "Jitter" column of Table 5, we observed different performance slowdown on four devices (i.e., C1, C2, D1, D2) at the end of their lifetime. The jitters can last as long as two workload iterations, and the slowdown for different sizes of writes within each workload iteration was different, which are summarized in Table 6. Furthermore, we mark the jitters that occurred on C1,

C2, D1, and D2 as the arrows in Figure 5C,D. We can see that all jitters occurred after 70% of the device lifetime. One possible reason may be that at the late stage of device lifetime, there are many internal errors, which leads to frequent retry and re-allocation and hurts performance.

On the other hand, however, half of the eight devices (i.e., A1, A2, B1, B2) did not exhibit any performance slowdown throughout the whole lifespan. This suggests that estimating the lifetime based on the performance may not work for some devices.

Table 6. Performance Comparison.

Size (KB)	Normal	Jittered	Percentage
0.5	3.41 MB/s	0.97 MB/s	23%
1 to 3.5	5.52 MB/s	2.11 MB/s	55%
4	10.81 MB/s	7.72 MB/s	17%
8	21.63 MB/s	12.88 MB/s	3%
16	56.24 MB/s	46.24 MB/s	6%
32, 64	114.00 MB/s	45.03 MB/s	8%

5.3. Analysis of Correlation

Besides analyzing each individual parameter, we also analyzed the correlation among parameters to identify more patterns of device behavior.

Finding #10: Jitters tended to occur when correctable errors were relatively stable. As shown in Figure 5C,D, the Jitters (i.e., the arrows) mostly occurred between two surging of correctable errors. This correlation may imply that the internal operations causing performance slowdown may be triggered by the previous surging of correctable errors or the FTLs only schedule those operations when the device status is relatively stable.

5.4. Analysis of the External Environment

Finding #11: Higher temperature can impact the drive lifetime under memory-oriented workloads. Temperature is known to be influential towards SSDs' ability for longtime data retention [24,25]. Through the test, we found out that higher temperature can have certain impacts on the drives' remaining lifetime under workloads of short-lived data. While we tried our best to simulate the field environment of data centers, the SSDs still suffered from temperature variation, which may be because of the transient difference of workloads, air flow, and FTL internal thermal throttling. As shown in Table 7, we demonstrated the distribution of the working temperature of different test devices. We may observe that, for SSDs having higher percentages of *hot* and *extreme* temperatures (namely A2, B1, C1, and D1), they all had shorter lifetimes, as shown in Table 5. This statistically verifies the impacts led by temperature variation. Thus, the external environment (e.g., temperature) should be included when designing a more accurate life indicator.

Table 7. Temperature monitoring of candidate SSDs.

	A1	A2	B1	B2	C1	C2	D1	D2
Average (°C)	44	42	47	41	42	43	45	44
Hot (>50°C)%	13%	17%	22%	13%	18%	14%	14%	11%
Extreme (>60°C)%	5%	10%	13%	4%	5%	5%	7%	5%

6. Design and Evaluation of iLife

Based on the findings of the endurance testing, we design and evaluate iLife, an accurate life indicator for SSDs, in this section.

6.1. Design of iLife

As discussed in Section 5, different errors show different patterns during the lifetime of the device, and the default indicator usually cannot show the lifetime accurately. Therefore, instead of indicating the device lifetime solely based on the "wear indicator" attribute of S.M.A.R.T, iLife combines multiple error patterns together and constructs a parameterized formula to indicate the lifetime.

Table 8 shows the variables used in iLife's formula. The "Importance" column shows the relative importance of each variable for predicating a device failure based on our study. The "Impact Func." means the type of Function applied to the variable in order to weight the variable in the formula.

Table 8. Variables in iLife and their relative importance. The last column shows the type of impact function applied to the variable. A "0" impact function means the variable is not used to determine the lifetime.

Variable	Importance	Impact Function
UCEr	high	Boolean
count of PEr surging	medium	linear
count of EEr surging	medium	linear
count of CEr surging	medium	linear
Wear	medium	linear
temperature	medium	Arrhenius law [26]
CRC	low	0
Jitter	low	0

For example, when the uncorrectable error count (UCEr) became non-zero, we always observed a consequent device failure (Finding #8). Therefore, the UCEr is of high importance for indicating imminent failures. Therefore, we applied a Boolean impact function on UCEr where a non-zero UCEr directly generated a warning of device failure.

As for the program error count (PEr) and erase error count (EEr), their exact values varied greatly across devices (Finding #4) and did not directly reflect the lifetime. However, the surging of these values tended to happen frequently towards the end of the device lifetime (Finding #5). Therefore, we counted the surging of PEr and EEr and applied a linear function to show their impact, which can be defined as $f(x) = kx$, where x is the count of PEr/EEr surging and k is the tunable weight.

Similarly, as for the correctable error count, only the surging matters. However, since CEr may be huge and could appear in the early stage of lifetime on some devices (Finding #6), iLife does not take all the CEr surging into account. Instead, based on the observation that PEr and EEr surging are correlated with CEr surging at the end of lifetime (Finding #10), iLife only counts the CEr surging appearing after a PEr or EEr surging.

The "Wear" variable means the "wear indicator" attribute of S.M.A.R.T, which is currently used by most manufacturers to indicate the remaining lifetime (referred to as the default indicator throughout the paper). Although it is usually inaccurate (Finding #1 and #2), the value still reflects the endurance of the device to a certain degree, and thus, we included it as part of the formula using the linear function.

Regarding temperature variation, we directly utilized the result from previous work [26], which is based on the Arrhenius law [26]. By factoring the temperature, we were able to evaluate quantitatively the impacts led by variation of the temperatures, especially ones under hotter environments (Finding #11).

The CRC error count (CRC) and the count of performance slowdown (jitter) had no direct correlation with the device failures in our experiments (Finding #7 and Finding #9). Therefore, iLife does not include them in the formula. Instead, iLife simply logs the CRC and jitter observed without warning.

Putting them all together, iLife uses the following formula to indicate the lifetime:

$$L = 100 * (\neg UCEr) - \sum_{i=2}^{N} f_i(x_i) \quad (1)$$

where L is the normalized remaining lifetime, which decreases from 100 (most healthy) to zero (imminent device failure). A negative value means imminent device failure, as well. UCEr is a Boolean value that reflects the existence of uncorrectable error. According to the previous analysis, a single occurrence of uncorrectable error will directly reduce L and issues a warning. x_i is one of the variables in Table 8, and f_i is the impact function for variable x_i. Note that although there are only seven variables defined in Table 8 (i.e., $N = 7$), the formula can be easily extended to include more variables if more error patterns are available.

6.2. Evaluation of iLife

We evaluated iLife using a different set of SSDs, which included six new devices from three different manufacturers (i.e., E, F, G), as shown in Table 9. We applied the same framework (Section 4) to test the endurance of the devices. iLife runs as a daemon process, which periodically queries the device status and calculates Formula (1) to indicate the lifetime of the device. The default parameters of iLife (e.g., the weight k in the linear function) were set based on our study in Section 5. In detail, we empirically set the weight for program and erase error surging to be 25. This, in terms, would allow at most four occurrences of program and/or erase error surging. The weight of correctable error was set to be 0.1 to reflect a smaller impact on the remaining life of the disk. In order to measure the accuracy of iLife, we did not stop the experiment after iLife issued a warning. Instead, we continued to wear out the device until it actually failed.

Table 9. Evaluation of iLife. The lifetime is measured in workload iteration count. All percentages are of the actual lifetime (i.e., the actual lifetime is 100%). The last column shows the additional reliable usage time by using iLife, compared with using the default indicator. * Due to the Read Only (RO) protection in G1, both the default indicator and iLife did not report failure before RO.

SSD	Size (GB)	Lith (nm)	Fail Type	Actual Lifetime	Default Indicator		iLife		Extended Lifetime
					Lifetime	Accuracy	Lifetime	Accuracy	
E1	64	20	DC	4815	3479	72.3%	4277	88.9%	16.6%
E2	64	20	DNF	5287	3402	64.3%	4521	85.5%	21.2%
F1	32	25	DNF	5203	3384	65.0%	4078	78.4%	13.4%
F2	64	25	DNF	6917	3295	47.6%	3921	56.7%	9.1%
F3	64	25	DNF	6573	3317	48.9%	3934	53.3%	5.4%
G1 *	16	20	RO	2917	> 2917	N/A	> 2917	N/A	N/A

Accuracy: Table 9 summarizes the evaluation results. On the first five devices (i.e., E1, E2, F1, F2, and F3), both iLife and the default indicator correctly issued warnings before the device failures. Moreover, iLife was more accurate than the default indicator in all five cases. We describe the accuracy as the percentage of actual lifetime. For example, E1's lifetime was 4153 workload iterations based on iLife, which was 88.9% of the actual lifetime, and it was 16.6% more compared with the lifetime reported by the default indicator. On average, iLife extended the usage time by 13.4%.

In addition, iLife can capture the subtle difference between devices of the same model. For example, E1 and E2 were from the same manufacturer and were of the same model. The default indicator showed very similar lifetimes for the two devices (i.e., only differing by 72 workload iterations), while their actual lifetime differed by 472 workload iterations. iLife accurately indicated their lifetime with a difference of 334 workload iterations. This was mainly because iLife used multiple error patterns besides the default wear indicator to model the device behavior more precisely.

One special case was G1. This device suddenly became Read Only (RO) after 2917 workload iterations. Neither iLife nor the default indicator predicted this failure properly. iLife did not model the read only failure because there was only one device showing similar behavior in the testing set (i.e., D1 in Table 5), and few error patterns were observed on that particular device. However, with more devices being studied, iLife could be potentially extended to model such failure accurately. Note that the default indicator also failed in this case. In other words, iLife did not make things worse.
Usage: iLife is a practical and extendable tool. As a lightweight background daemon, iLife only needs to query the device status via the standard S.M.A.R.T. interface. In other words, iLife can be deployed with little intrusion to the target system, and no special hardware support is required.

Moreover, iLife can be extended easily if more device parameters are available for monitoring or more devices are available for modeling. The current prototype of iLife was based on our limited testing set (i.e., the eight devices in Table 5). Despite the limited resources, iLife already showed a much better accuracy compared with existing indicators, which could potentially extend the usage of devices significantly and reliably for users. We believe the manufacturers, who have much more information on device and failure behaviors, can achieve a more accurate life indicator based on the idea of iLife. If devices could disclose more detailed status to users, more variables could be included in the formula of iLife for better modeling.

7. Limitation

While our findings from the extensive study are interesting and iLife has been proven to be effective in small-scale evaluations, there are some limitations and future work for this paper.

Scale: Due to limited lab resources and the lengthy processes, we only tested and analyzed 15 SSDs in total. Given the limited scale, we tried to select a variety of devices, including seven different brands of commercial drives with two lithography levels. Although our findings may only be applicable to similar products, our testing framework can be easily extended to conduct more extensive endurance study on other types of SSDs with different units (e.g., 3D NAND or SLC), different lithography, and even different FTL optimizations.

S.M.A.R.T. attributes: Our experiments and indicators relied on the proper reporting of the S.M.A.R.T. attributes. Although the S.M.A.R.T. attributes are regarded as an industry standard, not all the SSD vendors disclose the details to users, as shown in the drives D1 and D2 in our tests. To facilitate monitoring the health of devices, we suggest manufacturers disclose a more comprehensive set of S.M.A.R.T. attributes, especially for the important ones such as error counts, RBER, and reallocation blocks.

Data retention: In this project, we focused on SSDs that are mainly used in memory-orientated workloads, which thus do not require strong data retention ability. As a part of our future work, we plan to design a generic life indicator by including data retention as part of the study.

Temperature SSDs' reliability and retention ability can be severely impacted under high temperatures. Due to the limits of scale, our current temperature impact design still relied on the results from previous large-scale study and the effects we observed from our small-scale test. In the future, we will further conduct a large-scale temperature experiment to study the impacts of different temperatures towards the SSD's lifetime.

8. Related Work

Our study is related to the following previous works.

SSD failures. Previous studies on SSD devices deployed in data centers revealed the errors and failures occurring in production systems under certain workloads [3–5,27,28]. Some of their findings were different from ours. For example, they reported that the program/erase errors were rare. This is mainly because the SSDs in their studies were at different stages, i.e., approximately one third of the warranty cycles being exercised, from ours. On the contrary, the SSDs in our study were from the

beginning of the life until the device failures. Since our goal was to identify the accurate life indicator of the devices, our findings were more focused on the wear out or close to wear out phase of each SSD.

HDD failures: Relevant research studies have been conducted to estimate the lifetime of spinning disks via machine learning techniques [29–31]. These innovative techniques can reach the accuracy of estimating the remaining lifetime to be as close as 10–15 days. Since their studies were heavily dependent on the features of HDDs, they cannot be directly applied to flash-based SSDs.

Failures of flash memory cells: The reliability issue in NAND cells has been extensively studied since its initial release [25,26,32]. Researchers have been trying to alleviate this issue by both further understanding the internal causes and designing more reliable architectures [33–35]. Our research, however, mainly focused on NAND-based SSDs. Therefore, the internal firmware stacks and various optimizations of the devices made it infeasible to directly apply previous NAND cell research conclusions on our subjects.

9. Conclusions

This paper proposed a testing framework to perform the endurance study on SSDs automatically. With the extensive experiments on 15 different SSDs from six brands and two lithography levels, we gave unique findings on the behaviors when SSDs were close to the failure stage. For example, program and erase errors were clustered and usually occurred in the late stage of a drive's lifespan. Based on these key observations, we proposed iLife, an accurate life indicator of SSDs, by leveraging program, erase, (un)correctable error counts, and other error patterns. Our validation experiments demonstrated that, compared to existing life indicators of SSDs, which are purely dependent on P/E cycles, iLife dramatically improved the estimation accuracy by safely extending the SSD usage for up to 20.5% longer.

Author Contributions: Data curation, investigation, methodology, resources, software and writing—original draft, E.X.; project administration, supervision, writing-review & editing, S.L.

Funding: This research was funded by NSFC Nos. 61872373, 61872375 and 2017ZX01038104-002.

Acknowledgments: The authors in this paper would like to thank the anonymous reviewers for their feedbacks and our editor Coleen Shi for help processing the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kim, J.; Lee, J.; Choi, J.; Lee, D.; Noh, S.H. Enhancing SSD Reliability Through Efficient RAID Support. In Proceedings of the Third ACM SIGOPS Asia-Pacific Conference on Systems, APSys '12, Berkeley, CA, USA, 23–24 July 2012; p. 4.
2. McEwan, A.A.; Komsul, M.Z. Reliability and Performance Enhancements for SSD RAID. *Microprocess. Microsyst.* **2017**, *52*, 461–469. [CrossRef]
3. Xu, E.; Zheng, M.; Qin, F.; Wu, J.; Xu, Y. Understanding SSD Reliability in Large-Scale Cloud Systems. In Proceedings of the 2018 IEEE/ACM 3rd International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems (PDSW-DISCS), Dallas, TX, USA, 12 November 2018; pp. 45–53. [CrossRef]
4. Schroeder, B.; Lagisetty, R.; Merchant, A. Flash Reliability in Production: The Expected and the Unexpected. In Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST 16), Santa Clara, CA, USA, 22–25 February 2016; pp. 67–80.
5. Narayanan, I.; Wang, D.; Jeon, M.; Sharma, B.; Caulfield, L.; Sivasubramaniam, A.; Cutler, B.; Liu, J.; Khessib, B.; Vaid, K. SSD Failures in Datacenters: What? When? And Why? In Proceedings of the 9th ACM International on Systems and Storage Conference, SYSTOR '16, New York, NY, USA, 6–8 June 2016; pp. 7:1–7:11. [CrossRef]
6. SSD Endurance Test by Tech Report. Available online: http://techreport.com/review/27909/ (accessed on 20 August 2013).
7. Pavan, P.; Bez, R.; Olivo, P.; Zanoni, E. Flash memory cells-an overview. *Proc. IEEE* **1997**, *85*, 1248–1271. [CrossRef]

8. Yaakobi, E.; Ma, J.; Grupp, L.; Siegel, P.H.; Swanson, S.; Wolf, J.K. Error characterization and coding schemes for flash memories. In Proceedings of the 2010 IEEE Globecom Workshops, Miami, FL, USA, 6–10 December 2010; pp. 1856–1860. [CrossRef]
9. Wikipedia. Field Electron Emission Wikipedia, The Free Encyclopedia, 2004. Available online: https://en.wikipedia.org/wiki/Field_electron_emission (accessed on 22 July 2004).
10. Bez, R.; Camerlenghi, E.; Modelli, A.; Visconti, A. Introduction to flash memory. *Proc. IEEE* **2003**, *91*, 489–502. [CrossRef]
11. Cai, Y.; Haratsch, E.F.; Mutlu, O.; Mai, K. Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis. In Proceedings of the Conference on Design, Automation and Test in Europe, San Jose, CA, USA, 12–16 March 2012; pp. 521–526.
12. Agrawal, N.; Prabhakaran, V.; Wobber, T.; Davis, J.D.; Manasse, M.; Panigrahy, R. Design Tradeoffs for SSD Performance. In Proceedings of the USENIX 2008 Annual Technical Conference, Berkeley, CA, USA, 22–27 June 2008; pp. 57–70.
13. Sridharan, V.; Liberty, D. A Study of DRAM Failures in the Field. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Los Alamitos, CA, USA, 10–16 November 2012; pp. 76:1–76:11.
14. Schroeder, B.; Pinheiro, E.; Weber, W.D. DRAM Errors in the Wild: A Large-scale Field Study. In Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems, New York, NY, USA, 15–19 June 2009; pp. 193–204. [CrossRef]
15. Huang, P.; Wu, G.; He, X.; Xiao, W. An Aggressive Worn-out Flash Block Management Scheme to Alleviate SSD Performance Degradation. In Proceedings of the Ninth European Conference on Computer Systems, New York, NY, USA, 13–16 April 2014; pp. 22:1–22:14. [CrossRef]
16. Yadgar, G.; Yaakobi, E.; Schuster, A. Write Once, Get 50% Free: Saving SSD Erase Costs Using WOM Codes. In Proceedings of the 13th USENIX Conference on File and Storage Technologies, Berkeley, CA, USA, 16–19 February 2015; pp. 257–271.
17. Lu, Y.; Shu, J.; Zheng, W. Extending the Lifetime of Flash-based Storage through Reducing Write Amplification from File Systems. In Proceedings of the 11th USENIX Conference on File and Storage Technologies (FAST 13), San Jose, CA, USA, 12–15 February 2013; pp. 257–270.
18. SSD Endurance Test. Available online: https://techreport.com/review/27909/the-ssd-endurance-experiment-theyre-all-dead (accessed on 24 April 2019).
19. Solid-State Drive (SSD) Endurance Workloads. Technical Report, JEDEC Solid State Technology Association. Available online: https://www.jedec.org/sites/default/files/Alvin_Cox%20%5BCompatibility%20Mode%5D_0.pdf (accessed on 24 April 2019).
20. Joint Electron Device Engineering Council. Available online: https://www.jedec.org/about-jedec (accessed on 24 April 2019).
21. Zheng, M.; Tucek, J.; Qin, F.; Lillibridge, M. Understanding the Robustness of SSDS Under Power Fault. In Proceedings of the 11th USENIX Conference on File and Storage Technologies, Berkeley, CA, USA, 12–15 February 2013; pp. 271–284.
22. Zheng, M.; Tucek, J.; Qin, F.; Lillibridge, M.; Zhao, B.W.; Yang, E.S. Reliability Analysis of SSDs Under Power Fault. *ACM Trans. Comput. Syst.* **2016**, *34*, 10:1–10:28. [CrossRef]
23. Bruce Allen, Christian Franke, G.G. Smartmontools Ver. 6.5 (Stable), 2016. Available online: https://www.smartmontools.org/ (accessed on 29 April 2019).
24. Solid-State Drive (SSD) Requirements and Endurance Test Method. Technical Report, JEDEC Solid State Technology Association, September, 2010. Available online: https://www.jedec.org/sites/default/files/docs/JESD218.pdf (accessed on 24 April 2019).
25. Aritome, S.; Shirota, R.; Hemink, G.; Endoh, T.; Masuoka, F. Reliability issues of flash memory cells. *Proc. IEEE* **1993**, *81*, 776–788. [CrossRef]
26. Cai, Y.; Luo, Y.; Haratsch, E.F.; Mai, K.; Mutlu, O. Data retention in MLC NAND flash memory: Characterization, optimization, and recovery. In Proceedings of the 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), Burlingame, CA, USA, 7–11 Febuary 2015; pp. 551–563. [CrossRef]
27. Brewer, E.; Ying, L.; Greenfield, L.; Cypher, R.; T'so, T. Disks for Data Centers. Technical Report, Google, 2016. Available online: https://research.google.com/pubs/archive/44830.pdf (accessed on 24 April 2019).

28. Liang, S.; Qiao, Z.; Hochstetler, J.; Huang, S.; Fu, S.; Shi, W.; Tiwari, D.; Chen, H.; Settlemyer, B.; Montoya, D. Reliability Characterization of Solid State Drives in a Scalable Production Datacenter. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, USA, 10–13 December 2018; pp. 3341–3349. [CrossRef]
29. Murray, J.F.; Hughes, G.F.; Kreutz-Delgado, K. Machine Learning Methods for Predicting Failures in Hard Drives: A Multiple-Instance Application. *J. Mach. Learn. Res.* **2005**, *6*, 783–816.
30. Botezatu, M.M.; Giurgiu, I.; Bogojeska, J.; Wiesmann, D. Predicting Disk Replacement Towards Reliable Data Centers. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016. [CrossRef]
31. Hamerly, G.; Elkan, C. Bayesian Approaches to Failure Prediction for Disk Drives. In Proceedings of the Eighteenth International Conference on Machine Learning, Williamstown, MA, USA, 28 June–1 July 2001.
32. Yang, H.; Kim, H.; Park, S.; Kim, J.; Lee, S.; Choi, J.; Hwang, D.; Kim, C.; Park, M.; Lee, K.; et al. Reliability Issues and Models of sub-90nm NAND Flash Memory Cells. In Proceedings of the 8th International Conference on Solid-State and Integrated Circuit Technology Proceedings, Shanghai, China, 23–26 October 2006; pp. 760–762. [CrossRef]
33. Portal, J.M.; Saillet, B.; Nee, D. Flash memory cell diagnosis: High level model. In Proceedings of the IEEE Computational Systems Bioinformatics Conference, Stanford, CA, USA, 17 November 2004. [CrossRef]
34. Mohan, V.; Siddiqua, T.; Gurumurthi, S.; Stan, M.R. How I Learned to Stop Worrying and Love Flash Endurance. In Proceedings of the 2nd USENIX Conference on Hot Topics in Storage and File Systems, Boston, MA, USA, 22–25 June 2010.
35. Lee, J.D.; Choi, J.H.; Park, D.; Kim, K. Data retention characteristics of sub-100 nm NAND flash memory cells. *IEEE Electron Device Lett.* **2003**, *24*, 748–750. [CrossRef]