Flash Drive Lifespan *is* a Problem

Tao Zhang
The University of North Carolina at Chapel Hill
zhtao@cs.unc.edu

Aviad Zuck
Technion – Israel Institute of Technology
aviadzuc@cs.technion.ac.il

Donald E. Porter
The University of North Carolina at Chapel Hill
porter@cs.unc.edu

Dan Tsafrir
Technion – Israel Institute of Technology
dan@cs.technion.ac.il

ABSTRACT

When flash was introduced, wear-out was a known problem. Over time, a number of techniques have been developed to estimate the expected number of program/erase cycles under typical usage patterns, and sufficiently over-provision the cells such that the device meets its expected lifespan, even if individual cells fail. This paper started as a simple experiment: measuring whether the lifespan of flash devices in smartphones and other mobile devices, match the estimates. To our surprise, we find that, *in a matter of days*, simple, unprivileged applications can render the drive of several smartphones (and thus, the phone) inoperable. This result is concerning, as it means that installing malicious or poorly-written software could destroy the device itself. We experimentally demonstrate the problem, discuss reasons why it occurs, and consider potential solutions.

CCS CONCEPTS

• **Security and privacy → Mobile platform security**; • **Hardware → External storage**; • **Software and its engineering → Operating systems**;

ACM Reference format:
Tao Zhang, Aviad Zuck, Donald E. Porter, and Dan Tsafrir. 2017. Flash Drive Lifespan *is* a Problem. In *Proceedings of HotOS '17, Whistler, BC, Canada, May 08-10, 2017*, 8 pages.
https://doi.org/10.1145/3102980.3102988

1 INTRODUCTION

Flash technology is becoming increasingly popular for non-volatile data storage, especially on mobile devices. Flash has many advantages, such as fast random access, shock-resistance, high density, and decreasing costs. An important limitation is that flash cells have a limited number of program/erase cycles (i.e., writes). Flash cells encode logical bits by charging cells with a given voltage level; this charging mechanism eventually wears out individual cells. Vendors and applications apply many methods to increase the lifetime of flash packages [2, 15, 19, 23, 28, 30, 36, 47, 50, 51, 80]. Flash drive lifetime can be roughly estimated using "back-of-the-envelope" calculations: take the expected number of writes for the advertised LBA space over a 3 year period, divide by the expected P/E cycles per cell, and that will give you the number of physical cells to over-provision [10, 14, 17, 35, 52, 55, 62, 69, 70, 72].

This paper empirically demonstrates that complacently relying on the "back-of-the-envelope" is a problem for mobile flash. We explain the I/O performance characteristics of mobile storage devices. We then demonstrate how to issue workloads that can render such devices inoperable within a matter of days. Finally, we present a trivial, unprivileged app that is able to quickly cause the smartphone's internal storage to enter a state where the storage device is not considered reliable anymore, and finally gets the device into an unbootable state (i.e., "bricks" the device). Our method works by using less than 3% of the system's storage capacity, is applicable to a range of modern mobile storage devices, and is not hampered by various optimizations such as improved mobile storage interfaces [41] and hybrid memory hierarchies [21, 51]. In fact, the technology trends in future generations of flash devices, such as encoding more bits in fewer cells with more, fine-grained charging cycles (MLC and TLC flash), will exacerbate this problem.

This issue raises a practical security concern for mobile users: mobile ecosystems casually create a false sense of trust in consumers who are unwary of downloading third-party applications [34, 38, 56], rooting their devices, and trusting applications from marketplaces that may perform only a lax review process [58, 74, 82]. Moreover, storage in mobile devices is not user-serviceable; in terms of repair cost, destroying the flash is tantamount to destroying the device. We focus here on smartphones, but we believe the same issues apply to any small, flash-based devices on which third-party software can be loaded, potentially including critical infrastructure or internet-connected medical devices.

From a research perspective, this raises the question: how do we design systems for managing permanently-consumable resources? Issues such as disk scheduling or space allocation are treated as performance optimizations; in reality, a "free" app can become rather expensive for a user, yet current systems are not equipped to help the user reason about these issues. It is unclear whether file system design can mitigate this problem; designs that reduce write amplification may help, although part of the problem may be in the device firmware.

HotOS '17, May 08-10, 2017, Whistler, BC, Canada

ACM ISBN 978-1-4503-5068-6/17/05…$15.00
https://doi.org/10.1145/3102980.3102988

2 THE PERCEPTION OF FLASH LIFETIME

This section explains why flash has limited lifetime, details methods to extend the lifetime, and how lifetime is perceived by users and vendors.

2.1 Background

Flash packages are divided into blocks, typically 256–2048 KB in size. Blocks are further divided into pages, typically 4–16 KB in size. Pages are stored on physical wordlines, which are serially-connected cells. When data is written, the cells are electrically charged to be above or below a predefined voltage threshold. The logical value of a cell (e.g., a 0 or 1) is read by comparing its measured voltage to the threshold voltages.

At the granularity of a cell, charge can only be added, pushing logical values in one direction; to rewrite a page, one must lower the voltage of some cells. Lowering voltage requires an erase operation, which is implemented at the granularity of a block. Crucially, the program/erasure process degrades the cell, and worsens over time [17, 59]. Small electric charges tend to accumulate in cells, which eventually cause logical bit errors. The result is that, after a number of Program/Erase (P/E) cycles, flash blocks produce too many bit errors to be transparently corrected with parity checks.

The problem of flash endurance is exacerbated by the way vendors increase densities of flash memories. Modern flash packages differentiate between increasingly smaller levels of electrical charges. This method increases the likelihood that small, accumulated charges will cause bit errors. Earlier generations of flash chips, which store only 1 bit per cell, achieved figures of up to 100K P/E cycles. More modern chips which typically store two logical bits per cell, can only endure 3–10K P/E cycles [32, 65], and numbers as low as 1K P/E cycles have been reported when storing three bits per cell.

2.2 Extending Flash Lifetime

Many ways have been suggested to increase the lifetime of flash-based storage devices. Primarily, research in this area focused on two directions. Wear-leveling policies were suggested to ensure an equal distribution of P/E cycles over flash blocks [2, 13, 30]. Another significant body of work is dedicated to Error Correction Coding (ECC) schemes [15, 28], which give a measure of tolerance to bit errors as the device ages.

Other methods for extending SSD lifetime include adjusting the reference voltage levels up over time [19], hybrids of high and low-endurance flash memories [21, 51], periodic refreshing of data [20, 80], identifying pages with low endurance [44], additional over-provisioning [36], data reduction [23, 76], and improving host-side I/O applications and access patterns [43, 47, 50].

Over a long period, flash can heal as trapped charge dissipates. Recent research has proposed to accelerate the process by applying heat to worn out cells, accelerating the process of freeing trapped electrons [22, 24, 25, 77]. However, this technology is not yet widely used.

2.3 SSD Lifetime Estimation in the Wild

Lifetime estimates of flash packages allow us to speculate on the lifetime of SSDs. For example, we consider a typical consumer-grade SSD which can endure 3K P/E cycles [65]. Even if we conservatively assume that various optimizations in hardware, firmware, and software balance out ill-behaved user workloads, it is fair to assume that the SSD can endure at least as many rewrites as its underlying storage media, i.e., 3K rewrites of the drive's entire data. Therefore, the drive can be completely rewritten three times a day over for three years.

Several tests corroborate this calculation. Several datacenter providers independently concluded that various SSDs last for years [57, 65], a high figure under relatively strenuous usage patterns. Others found that even consumer-level SSDs can write petabytes of data before failing [10, 69]. Such findings lead many consumers to believe that SSDs last for extremely long periods of time [14, 52, 69] since "modern SSDs easily write far more data than most consumers will ever need" [69]. This view is also prevalent in the research community [17, 55]. Some researchers even posit that manufacturers' endurance estimates are "extremely conservative" [35].

Drive warranties are a concrete and conservative measure for SSD usage and lifetime expectations. Existing warranties from major vendors demonstrate that they too expect modern flash-based drives to last for years under typical usage patterns. Samsung SSDs ship with a limited 3–10 year warranty [62], contingent on the volume of writes issued to the device. Intel [39] and Toshiba [70, 72] provide similar guarantees for various SSD offerings. SanDisk also offers a 1–10 year warranty for many of its products [64], and a lifetime guarantee for a variety of low-end storage devices (i.e., SD cards).

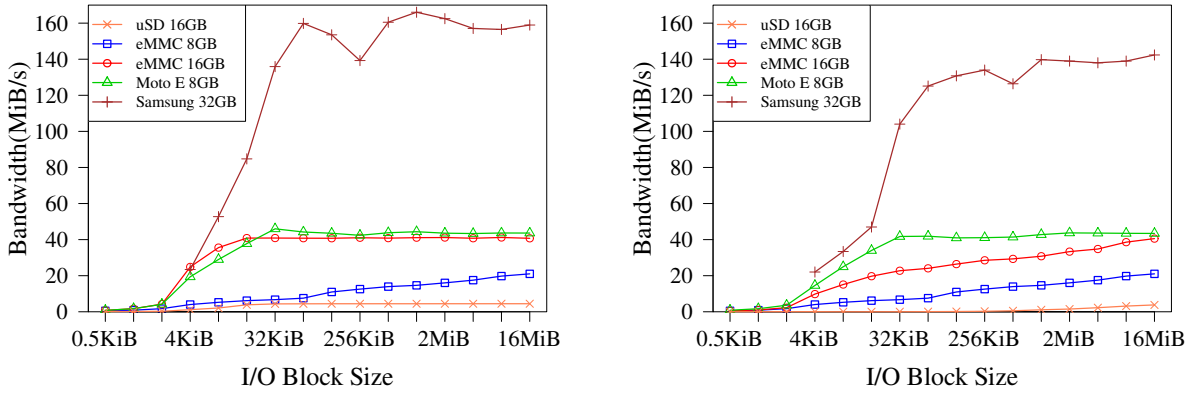
Such guarantees from multiple leading vendors, coupled with the fact that commercial SSD products last for years and allow data on the device to be entirely rewritten several thousands of times, reflect a common notion that endurance is effectively a non-issue.

3 WHAT ABOUT MOBILE STORAGE?

This section explains the reasons why we focus this study on mobile devices, and why we suspected there might be a problem with flash lifespan on mobile devices.

Increasing Popularity of Smartphones. Users increasingly switch from personal computers to mobile devices. Despite the falling prices of hardware, personal computer sales are declining [31, 45, 75]; among other reasons, because of "the inexorable growth of smartphones and other mobile devices" [45] as "consumers in this segment have high dependency on smartphones" [31].

False Sense of Safety. Users have a relatively high degree of trust in mobile ecosystems for several reasons. First, the rules that govern the ecosystem of mobile devices are stricter than the ones for personal computers. Mobile operating systems employ tighter security models [7–9] to prevent applications from accessing sensitive system information, and the data of other applications. For example, the default settings of most desktop operating systems do not prevent a weather application from accessing music files in the media player's library. Second, mobile device users typically download applications from large marketplaces such as Apple's App Store for iOS and Google Play for the Android OS. Although it is not strictly safe [58, 74, 82], marketplaces employ a review process to identify



(a) Sequential Write (b) Random Write

Figure 1: Write performance of external and smartphone eMMC cards.

some security problems before an application is made available to end users [16, 67].

Mobile ecosystems still have security problems [34, 38, 56], and users engage in risky behavior. Over a third of users install third-party applications, such as apps from unknown providers. 16% of users jailbreak or root their devices. A similar portion of users install apps requesting unnecessary permissions [29, 38]. Various apps cause serious harm, whether in the form of malware [74, 82] or poorly written code, as recently demonstrated in a bug in the Spotify app that redundantly issued large volumes of I/O to the underlying storage [26].

The result is a false sense of security, as users are more inclined to install harmful apps, including apps that may destroy the underlying storage in their mobile device.

Mobile Storage: Unknown Unknowns. Full-fledged SSDs are high-capacity, multi-chip devices, with high-end controllers and large onboard RAM. The specifications, performance characteristics, and lifetime estimates of SSDs are usually made public by vendors, as SSDs can easily be purchased and installed independently by consumers. Mobile storage devices are different in many senses: they are low-cost, have much smaller capacity, and typically contain only a few flash chips, which are managed using a simple controller. Unlike SSDs, mobile storage devices are usually soldered down to the mobile platform at the manufacturing process and accessed through dedicated interfaces [40, 41]. Vendors generally do not publicly detail the specifications, performance characteristics, lifetime guarantees, and warranties, for these devices. In summary, vendors don't make any claims about longevity, and have a much tighter hardware budget, which in turn limits the applicability of many common lifespan extension techniques.

4 THE PROBLEM

In this section we measure the performance characteristics of several mobile flash devices, and then show that all of these devices can be worn out relatively quickly, despite differences in the underlying architecture and hardware.

4.1 Evaluation Setup

All of our experiments use embedded flash storage devices like eMMC, uSD or UFS, which are currently the prevailing mobile storage solutions on the market. The first class of measured devices includes two external eMMC chips, Toshiba THGBMBG6D1KBAIL 8GB eMMC[71], and SanDisk 16GB iNAND 7030 [63], referred to as "eMMC 8GB" and "eMMC 16GB", respectively. All experiments on external eMMC chips were performed using the ODROID C2 platform [33]. We also experiment with Kingston SDC4/16GB [49] (labeled "uSD 16GB"), a conventional MicroSD card that is commonly used as external, additional storage in smartphones.

The second class of devices is smartphones. We conduct most of our experiments on a Moto E 2nd Gen smartphone [5], a mid-range smartphone with 8GB internal eMMC storage. We also use a Samsung S6 smartphone [6], a high-end smartphone with 32GB of internal storage. The Samsung smartphone uses a UFS [41] storage device, which is a recent successor to eMMC. We refer to these phones as "Moto E 8GB" and "Samsung S6 32GB", respectively. We also examine two budget smartphones, referred to as "BLU 512MB"[4] and "BLU 4GB"[3], to see how cheaper hardware affects lifespan.

The operating systems involved in the experiments include Linux and Android systems. Specifically, the experiments on external eMMC chips and MicroSD cards are conducted on Linux system (Ubuntu 16.04 LTS with kernel version 3.14.79) with Ext4[54] file systems. Smartphone experiments are conducted using their stock Android systems, with versions including 5.1 (Moto E 8GB), 6.0.1 (Samsung S6 32GB), and 4.4 (BLU 512MB, BLU 4GB). Most of these smartphones use the default Ext4 file system, except the Moto E 8GB which uses F2FS[50]. To study the influences introduced by file systems, we conducted the same experiments on two Moto E 8GB phones, one with F2FS and the other with Ext4. In the rest of this paper, Ext4 will be the default file system unless specified otherwise.

4.2 Performance Characteristics

We first explore the I/O characteristics of eMMC chips, to understand their behavior and find the most problematic I/O access patterns for these devices. Figure 1 shows the write performance micro-benchmark results for write I/O patterns (sequential/random) with different synchronous request sizes. For brevity, we omit read results, which were similar to the write results.

We draw the following conclusions from these results:

- eMMC chips outperform the MicroSD card in all I/O patterns, including random I/O.
- eMMC chips perform similarly for random and sequential access patterns.
- eMMC write I/O throughput generally scales linearly until it plateaus.

These results do not match the common assumption that eMMC chips are essentially MicroSD cards with soldered-down packages [27, 48]. If this were the case, the I/O performance of eMMC devices would have been much worse due to increased garbage collection overhead and reduced parallelism [48, 73]. We conclude that the I/O performance of modern eMMC devices hinges on request size. Larger requests utilize more internal hardware units in parallel [2, 79] (e.g., chips) and increase I/O performance until full internal parallelism is reached.

4.3 External eMMC Wear-out

Based on the micro-benchmark results, we hypothesize that eMMC chips can serve a large volume of intense write I/O activity within a relatively short span of time. Such write activity can quickly consume the P/E cycle quota of the underlying flash cells.

To test this hypothesis we use the eMMC lifetime estimation indicator. This indicator partitions the estimated lifespan of the chip (as monitored by the firmware) into 11 levels starting from 1 to 11. When the indicator has value n, it means $(n-1)*10\% \sim n*10\%$ of this chip's lifetime was consumed. Indicator value of 11 means the chip has exceeded its maximum estimated lifetime, may introduce uncorrectable errors in stored data, and should be considered unreliable[40].

We repeatedly rewrote small, randomly-selected regions of four 100MB files on each external card, and measured the wear-out indicator, shown in Figure 2. The results show that the required I/O volume is mostly constant throughout the lifetime of the devices. Specifically, it takes a maximum of 992GiB to increment the wear-out level by 10% in the 8GB eMMC chip. From the OS's perspective, this is roughly three times lower than the "back-of-the-envelope" three thousand or more complete rewrites. Moreover, at a maximum throughput of 20 MiB/s, one could write this volume of data in 140 hours (6 days). For the 16GB eMMC chip, 23 TiB of writes are required to reach end-of-life after 164 hours (7 days) at 40 MiB/s.

Advanced Factors Affecting Wear-out. We further explore additional factors that may affect the wear-out time of eMMC chips. The first factor is internal write overheads [30, 36]. The performance of flash storage systems is highly affected by write amplification—the amount of garbage collection overhead required to make way for new writes. Write amplification can be increased by reducing the amount of free space in the system.

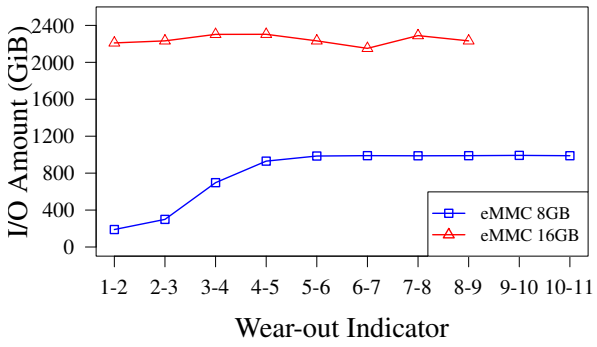


Figure 2: Amount of I/O needed to increment the wear-out indicator on two external eMMC chips.

The second factor we explored is storage heterogeneity. Some flash-based storage devices combine different types of flash memories. The faster, more expensive memory has a higher lifetime, and is used sparingly for storing hot data and caching purposes [37]. For such hardware architectures, eMMC supports two different wear-out indicators, one for each memory type (labeled "Type A" and "Type B"). Note that these different memory types are managed by FTL, and presented to the OS as a single device. The distinction is only visible at the level of these differentiated wear indicators.

To explore these factors we tested the time to increment the wear-out counters by varying (1) I/O access patterns (small random/large sequential); and (2) Space utilization (0%-90% of static data). We performed our tests on the eMMC 16GB model, which employs a hybrid storage design. Table 1 shows the wear-out counter value for each memory type over the course of our tests.

The results indicate that memory "Type B" is steadily wearing out under various I/O patterns. The "Type A" memory requires almost 6x more writes to increment its wear-out indicator from 1–2 (i.e., increment 1–6 for "Type B"). These results led us to conclude that "Type A" is the smaller, more performant type of memory.

We also note that "Type A" memory wears out much faster under high utilization setups, while "Type B" memories wear-out much slower. We further tested this result by modifying the experiment's access patterns so that data rewrites are aimed at the large utilized space instead of the free space ("Type B" 8-9, "Type A" 3-7 in Table 1). In this setup the wear-out rate is further increased. We believe these results indicate that the firmware of this specific chip model dynamically combines "Type A" and "Type B" memories into a single storage pool when the device is highly utilized and fragmented. The resulting additional free space helps reduce write amplification.

4.4 Smartphone Wear-out Experiments

Finally, we perform wear-out experiments on smartphones. Mobile applications, by default, do not have direct access to the underlying storage device. Therefore, we used a simple application (963 LoC, mostly for UI and various Android hooks) that continuously rewrites 100MB files in the application's private storage area, which is allocated to the application by default. Notably, our application required no special permissions.

Type A Flash Cell					Type B Flash Cell				
Indic.	I/O Vol. (GiB)	Increment Time (Hour)	I/O Pattern	Space Util.	Indic.	I/O Vol. (GiB)	Increment Time (Hour)	I/O Pattern	Space Util.
1-2	11935.94	-	-	-	1-2	2210.16	28.23	4 KiB rand	0%
					2-3	2232.42	28.66	4 KiB rand	0%
					3-4	2302.73	15.77	128 KiB seq	0%
					4-5	2303.52	15.66	128 KiB seq	0%
					5-6	2232.81	28.66	4 KiB rand	0%
2-3	3903.52	-	-	-	6-7	2151.17	29.52	4 KiB rand	90%+
					7-8	2289.45	27.41	4 KiB rand	50%+
3-4	439.06	20.44	4 KiB rand rewrite	90%+	8-9	2232.42	103.93	4 KiB rand rewrite	90%+
4-5	439.45	20.42	4 KiB rand rewrite	90%+					
5-6	438.67	20.39	4 KiB rand rewrite	90%+					
6-7	439.06	20.47	4 KiB rand rewrite	90%+					
...	...	...	...	...	...	...	...	...	...

Table 1: eMMC 16GB wear-out indicators over time (higher is worse). Lower rows indicate later measurements.

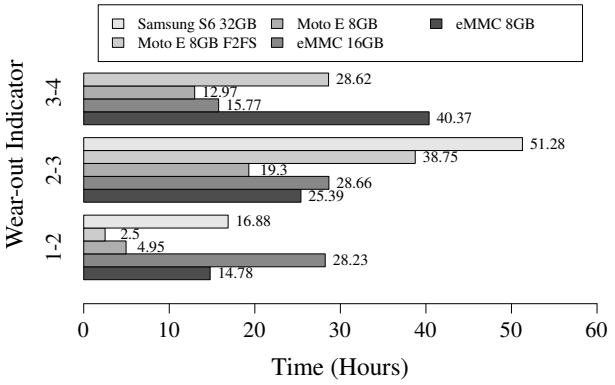


Figure 3: Time to increment wear-out indicators on two smartphone models and two external eMMC chips.

Figure 3 illustrates the experiment time required to increment the wear-out counter for the two tested phone models, based on the volume of writes required and the maximum write throughput of the phone's storage device. For comparison we also include the relevant results for the two external eMMC chips. Timing results vary, even for the same device model, most likely due to firmware-specific behaviors and optimizations. Even so, we indicate that the storage device in all phone models can be worn out in a matter of days to a few weeks. In the two budget smartphones "BLU 512MB" and "BLU 4GB" the eMMC chip did not provide reliable wear-out indications. However, both phones were bricked within two weeks.

Figure 4 shows the I/O volume needed to increment the wear-out indicator on two Moto E 8GB phones, using the F2FS and Ext4 file systems respectively. The results on the phone which uses the Ext4 file system exhibit similar wear-out characteristics to the eMMC 8GB chip in Figure 2. With F2FS, wearing out the phone's storage requires about half of the I/O volume, because the additional mapping mechanism in F2FS doubles the amount of I/O reaching the storage device under 4KiB synchronous writes. On the other hand, the wear-out workload has lower throughput when using F2FS. Therefore, the time required to increment the wear-out indicator when using F2FS is longer than Ext4, as shown in Figure 3. This comparison demonstrates that using F2FS, a flash-friendly file system, does not mitigate the wear-out problem, except inasmuch as it inadvertently rate limits *all* I/O to the device.

Detection. We experiment with the difficulty of hiding a malicious, I/O intensive application on Android. We observe two likely indicators of a problematic app that would manifest before the device is bricked, and find that both are easily evaded. First, Android monitors energy consumption, but only when on battery. Thus, we can evade detection via power monitoring by only running I/O intensive work when the phone is charging; the app can tell when the phone is charging. Second, Android shows apps currently running in background or as services (cf., `ps` on Unix). We observe that the refresh time for this monitor is around one second, and the app can detect when the screen is lit. By suspending malicious I/O when the screen is on, one can effectively evade this process monitor. In summary, most phones spend a significant fraction of the day charging with the screen disabled; even a stealthy version of this experiment could brick a phone within some reasonable factor of the time in these experiments. We note that continuously running malicious applications may cause the phone to abnormally heat up, which may raise the suspicion of users, though such extent of heating may be attributed to heat generated by the charging process. We leave the exploration of this, and other possible detection methods for the suggested malicious app as future research.

4.5 Discussion on solutions

Although a systematic solution remains an open problem, we consider several practical solutions to mitigate the problem of wearing out mobile flash storage.

First, the system may choose to expose and monitor the wear-out indicator to applications and users, similarly to the S.M.A.R.T. system on disks. Although this solution would not help pinpoint the application which is harming the device, it can at least provide an indication to users that the device's lifespan may be in jeopardy.

To help recognize potential malicious applications, the system can collect app-specific I/O statistics, much like the cellular data usage. Users can then locate applications which are issuing an unexpected amount of I/O.

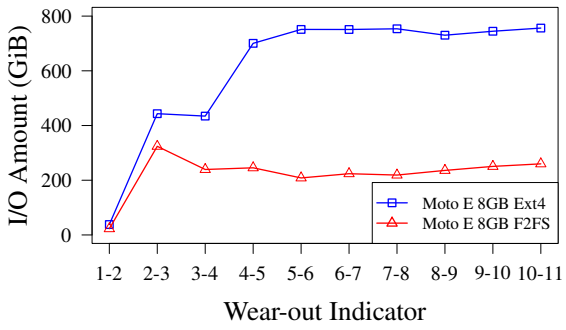


Figure 4: Amount of I/O needed to increment the wear-out indicator on two Moto E 8GB smartphones with different file systems.

The system may also try to limit application I/O to a rate that ensures an acceptable device lifespan. However, this may harm benign applications that rely on bursts of I/O requests (e.g., file transfer), and negatively affect user experience. A more refined approach would distinguish between benign and malicious I/O patterns, to selectively rate limit only harmful applications without affecting the performance of normal applications. We plan to investigate the efficacy of this approach in future work, although such a solution should be driven by a model of expected mobile application I/O behavior.

5 RELATED WORK

Access control of various platform components is a prominent issue in mobile devices research [11, 12, 18, 42, 60, 78]. However, these efforts are usually geared towards enforcing permissions and preserving privacy rather than the integrity of physical resources. Aware[61] provides a security framework to manage app accesses to I/O devices. But it is mainly used to prevent remote access trojans from accessing sensitive I/O devices (e.g., camera), and is not suitable to avoid high-volume I/O workloads.

The consumption of power, another depletable resource in mobile devices, has also gained much attraction in recent years. Tools to monitor [68, 81], save [1, 46, 66, 68, 83] and rate limit [53] power consumption were suggested both by researchers and as commercial tools. Power, unlike flash lifetime, is also a renewable resource. Even so, we believe that some of these approaches may be applicable for the problem presented in this work.

6 CONCLUSIONS

The life of a mobile flash device is shorter than users think, and this lifespan is easily squandered by a buggy or malicious app. It is unclear whether simple approaches, such as rate limiting, will be effective without harming usability of desirable applications. Perhaps designing storage systems that reduce write amplification will help, but enough about mobile storage is hidden that it is hard to tell what is simply within the hardware design, and what could be improved in software or firmware. OS designs historically have not had abstractions for budgeting a non-renewable resource, but our measurements indicate this is a timely research topic.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments on earlier drafts of the work. This research was supported by Grant 2014621 from the United States-Israel Binational Science Foundation (BSF), by Grant CNS-1526707 from the United States National Science Foundation (NSF), and VMware. This work was done in part while Porter and Zhang were at Stony Brook University.

REFERENCES

[1] 2EASY. 2017. Easy Battery Saver. http://www.2easydroid.com. (2017).
[2] Nitin Agrawal, Vijayan Prabhakaran, Ted Wobber, John D. Davis, Mark Manasse, and Rina Panigrahy. 2008. Design Tradeoffs for SSD Performance. In *USENIX Annual Technical Conference (ATC)*. USENIX Association, 57–70. http://dl.acm.org/citation.cfm?id=1404014.1404019
[3] Amazon.com. 2017. BLU Advance 4.0L. https://www.amazon.com/BLU-Advance-4-0L-Unlocked-Smartphone/dp/B00YCTIL88. (2017).
[4] Amazon.com. 2017. BLU Dash D171a. https://www.amazon.com/BLU-Dash-D171a-Unlocked-White/dp/B00F9AK2D6. (2017).
[5] Amazon.com. 2017. Motorola Moto E LTE. https://www.amazon.com/Motorola-Moto-LTE-Contract-Cellular/dp/B00XQVDW6Y. (2017).
[6] Amazon.com. 2017. Samsung Galaxy S6. https://www.amazon.com/Samsung-Galaxy-S6-Factory-Unlocked/dp/B0143NXM68. (2017).
[7] Android.com. 2016. Application security. https://source.android.com/security/overview/app-security.html#how-users-understand-third-party-applications. (2016).
[8] Android.com. 2017. Android Practices for Security and Privacy. https://developer.android.com/training/articles/security-tips.html#StoringData. (2017).
[9] Apple.com. 2017. iOS App Sandbox Design Guide. https://developer.apple.com/library/content/documentation/Security/Conceptual/AppSandboxDesignGuide/AboutAppSandbox/AboutAppSandbox.html. (2017).
[10] ArsTechnica. 2014. Consumer-grade SSDs actually last a hell of a long time. https://arstechnica.com/gadgets/2014/06/consumer-grade-ssds-actually-last-a-hell-of-a-long-time/. (2014).
[11] Michael Backes, Sven Bugiel, Sebastian Gerling, and Philipp von Styp-Rekowsky. 2014. Android Security Framework: Extensible Multi-layered Access Control on Android. In *Proceedings of the 30th Annual Computer Security Applications Conference (ACSAC)*. ACM, 46–55. https://doi.org/10.1145/2664243.2664265
[12] Michael Backes, Sebastian Gerling, Christian Hammer, Matteo Maffei, and Philipp von Styp-Rekowsky. 2013. *AppGuard – Enforcing User Requirements on Android Apps*. Springer, 543–548. https://doi.org/10.1007/978-3-642-36742-7_39
[13] Avraham Ben-Aroya and Sivan Toledo. 2006. Competitive Analysis of Flash-memory Algorithms. In *Proceedings of the 14th Conference on Annual European Symposium - Volume 14 (ESA)*. 100–111.
[14] betanews. 2014. Modern SSDs can last a lifetime. betanews, https://betanews.com/2014/12/05/modern-ssds-can-last-a-lifetime/. (2014).
[15] Richard E. Blahut. 2003. *Algebraic Codes for Data Transmission:*. Cambridge University Press. https://doi.org/10.1017/CBO9780511800467
[16] Android Developers Blog. 2015. Creating Better User Experiences on Google Play. Android developrs blog, https://android-developers.googleblog.com/2015/03/creating-better-user-experiences-on.html. (17 March 2015).
[17] Simona Boboila and Peter Desnoyers. 2010. Write Endurance in Flash Drives: Measurements and Analysis. In *Proceedings of the 8th USENIX Conference on File and Storage Technologies (FAST)*.
[18] Sven Bugiel, Lucas Davi, Ra Dmitrienko, and Thomas Fischer. 2012. Towards taming privilege-escalation attacks on Android. In *Proceedings of the 19th Annual Network & Distributed System Security Symposium*.
[19] Y. Cai, O. Mutlu, E. F. Haratsch, and K. Mai. 2013. Program interference in MLC NAND flash memory: Characterization, modeling, and mitigation. In *IEEE 31st International Conference on Computer Design (ICCD)*. 123–130. https://doi.org/10.1109/ICCD.2013.6657034
[20] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. S. Unsal, and K. Mai. 2012. Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime. In *IEEE 30th International Conference on Computer Design (ICCD)*. 94–101. https://doi.org/10.1109/ICCD.2012.6378623
[21] Li-Pin Chang. 2008. Hybrid solid-state disks: Combining heterogeneous NAND flash in large SSDs. In *Asia and South Pacific Design Automation Conference*. 428–433. https://doi.org/10.1109/ASPDAC.2008.4483988
[22] Yu-Ming Chang, Yuan-Hao Chang, Jian-Jia Chen, Tei-Wei Kuo, Hsiang-Pang Li, and Hang-Ting Lue. 2014. On Trading Wear-leveling with Heal-leveling. In

Proceedings of the 51st Annual Design Automation Conference (DAC '14). ACM, New York, NY, USA, Article 83, 6 pages. https://doi.org/10.1145/2593069.2593172

[23] Feng Chen, Tian Luo, and Xiaodong Zhang. 2011. CAFTL: A Content-aware Flash Translation Layer Enhancing the Lifespan of Flash Memory Based Solid State Drives. In *Proceedings of the 9th USENIX Conference on File and Stroage Technologies (FAST)*.

[24] R. Chen, Y. Wang, D. Liu, Z. Shao, and S. Jiang. 2017. Heating Dispersal for Self-Healing NAND Flash Memory. *IEEE Trans. Comput.* 66, 2 (Feb 2017), 361–367. https://doi.org/10.1109/TC.2016.2595572

[25] Y. T. Chiu. 2012. Forever Flash. *IEEE Spectrum* 49, 12 (December 2012), 11–12. https://doi.org/10.1109/MSPEC.2012.6361744

[26] Dan Goodin. 2016. Spotify is writing massive amounts of junk data to storage drives. http://arstechnica.com/information-technology/2016/11/for-five-months-spotify-has-badly-abused-users-storage-drives/. (2016).

[27] Peter Desnoyers. 2013. What Systems Researchers Need to Know about NAND Flash. In *Presented as part of the 5th USENIX Workshop on Hot Topics in Storage and File Systems*. https://www.usenix.org/conference/hotstorage13/workshop-program/presentation/desnoyers

[28] G. Dong, N. Xie, and T. Zhang. 2011. On the Use of Soft-Decision Error-Correction Codes in NAND Flash Memory. *IEEE Transactions on Circuits and Systems I: Regular Papers* 58, 2 (Feb 2011), 429–439. https://doi.org/10.1109/TCSI.2010.2071990

[29] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. 2012. Android Permissions: User Attention, Comprehension, and Behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security (SOUPS)*. ACM.

[30] Eran Gal and Sivan Toledo. 2005. Algorithms and Data Structures for Flash Memories. *Computing Surveys* 37, 2 (June 2005), 138–163. https://doi.org/10.1145/1089733.1089735

[31] Gartner. 2017. Gartner Says 2016 Marked Fifth Consecutive Year of Worldwide PC Shipment Decline. Gartner, http://www.gartner.com/newsroom/id/3568420. (January 2017).

[32] Laura M. Grupp, John D. Davis, and Steven Swanson. 2012. The Bleak Future of NAND Flash Memory. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST)*. USENIX Association. http://dl.acm.org/citation.cfm?id=2208461.2208463

[33] Hardkernel. 2017. ODROID | Hardkernel. http://www.hardkernel.com/main/products/prdt_info.php?g_code=G145457216438. (2017).

[34] Mark A. Harris, Steven Furnell, and Karen Patten. 2014. Comparing the Mobile Device Security Behavior of College Students and Information Technology Professionals. *Journal of Information Privacy and Security* 10, 4 (2014), 186–202. https://doi.org/10.1080/15536548.2014.974429 arXiv:http://dx.doi.org/10.1080/15536548.2014.974429

[35] Damien Hogan, Tom Arbuckle, and Conor Ryan. 2013. Estimating MLC NAND Flash Endurance: A Genetic Programming Based Symbolic Regression Application. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO)*. ACM, 1285–1292. https://doi.org/10.1145/2463372.2463537

[36] Xiao-Yu Hu, Evangelos Eleftheriou, Robert Haas, Ilias Iliadis, and Roman Pletka. 2009. Write Amplification Analysis in Flash-based Solid State Drives. In *Proceedings of The Israeli Experimental Systems Conference (SYSTOR)*. ACM, Article 10, 9 pages. https://doi.org/10.1145/1534530.1534544

[37] Soojun Im and Dongkun Shin. 2009. Storage Architecture and Software Support for SLC/MLC Combined Flash Memory. In *Proceedings of the 2009 ACM Symposium on Applied Computing (SAC '09)*. ACM, 1664–1669. https://doi.org/10.1145/1529282.1529655

[38] James Imgraben, Alewyn Engelbrecht, and Kim-Kwang Raymond Choo. 2014. Always connected, but are smart mobile users getting more security savvy? A survey of smart mobile device users. *Behaviour & Information Technology* 33, 12 (2014), 1347–1360. https://doi.org/10.1080/0144929X.2014.934286 arXiv:http://dx.doi.org/10.1080/0144929X.2014.934286

[39] Intel. 2017. Limited Warranties for Intel® Solid State Drives. Intel, http://www.intel.com/content/www/us/en/support/solid-state-drives/000005861.html. (2017).

[40] JEDEC. 2015. JEDEC e.MMC standard v5.1. https://www.jedec.org/standards-documents/technology-focus-areas/flash-memory-ssds-ufs-emmc/e-mmc. (Feb 2015).

[41] JEDEC. 2016. JEDEC Universal Flash Storage (UFS) standard v2.1. http://www.jedec.org/standards-documents/focus/flash/universal-flash-storage-ufs. (May 2016).

[42] Jinseong Jeon, Kristopher K. Micinski, Jeffrey A. Vaughan, Ari Fogel, Nikhilesh Reddy, Jeffrey S. Foster, and Todd Millstein. 2012. Dr. Android and Mr. Hide: Fine-grained Permissions in Android Applications. In *Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM)*.

[43] Sooman Jeong, Kisung Lee, Seongjin Lee, Seoungbum Son, and Youjip Won. 2013. I/O Stack Optimization for Smartphones. In *Presented as part of the USENIX Annual Technical Conference (USENIX ATC)*. USENIX, 309–320. https://www.usenix.org/conference/atc13/technical-sessions/presentation/jeong

[44] Xavier Jimenez, David Novo, and Paolo Ienne. 2014. Wear Unleveling: Improving NAND Flash Lifetime by Balancing Page Endurance. In *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST)*. USENIX, 47–59. https://www.usenix.org/conference/fast14/technical-sessions/presentation/jimenez

[45] Wall Street Journal. 2016. PC Sales Drop to Historic Lows. Wall Street Journal, http://www.wsj.com/articles/pc-sales-drop-to-historic-lows-1452634605. (2016).

[46] JuiceDefender. 2016. JuiceDefender battery saver. http://www.juicedefender.com. (2016).

[47] Hyojun Kim, Nitin Agrawal, and Cristian Ungureanu. 2012. Revisiting storage for smartphones. *ACM Transactions on Storage* 8, 4 (November 2012), 1–25.

[48] J. M. Kim and J. S. Kim. 2012. Advil: A Pain Reliever for the Storage Performance of Mobile Devices. In *IEEE 15th International Conference on Computational Science and Engineering (CSE)*. 429–436. https://doi.org/10.1109/ICCSE.2012.66

[49] Kingston. 2017. Class 4 microSDHC Card - 4GB–32GB | Kingston. https://www.kingston.com/us/flash/microsd_cards/sdc4. (2017).

[50] Changman Lee, Dongho Sim, Jooyoung Hwang, and Sangyeun Cho. 2015. F2FS: A New File System for Flash Storage. In *13th USENIX Conference on File and Storage Technologies (FAST)*. USENIX Association, 273–286. https://www.usenix.org/conference/fast15/technical-sessions/presentation/lee

[51] Sungjin Lee, Keonsoo Ha, Kangwon Zhang, Jihong Kim, and Junghwan Kim. 2009. FlexFS: A Flexible Flash File System for MLC NAND Flash Memory. In *Proceedings of the Conference on USENIX Annual Technical Conference (USENIX)*. 1. http://dl.acm.org/citation.cfm?id=1855807.1855816

[52] LifeHacker. 2015. How Long Will My Hard Drives Really Last? LifeHacker, http://lifehacker.com/how-long-will-my-hard-drives-really-last-1700405627. (2015).

[53] Marcelo Martins, Justin Cappos, and Rodrigo Fonseca. 2015. Selectively Taming Background Android Apps to Improve Battery Lifetime. In *USENIX Annual Technical Conference (USENIX ATC)*. USENIX Association, 563–575. https://www.usenix.org/conference/atc15/technical-session/presentation/martins

[54] Avantika Mathur, Mingming Cao, Suparna Bhattacharya, Andreas Dilger, Alex Tomas, and Laurent Vivier. 2007. The New Ext4 Filesystem: Current Status and Future Plans.

[55] Vidyabhushan Mohan, Taniya Siddiqua, Sudhanva Gurumurthi, and Mircea R. Stan. 2010. 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 (HotStorage)*. USENIX Association. http://dl.acm.org/citation.cfm?id=1863122.1863125

[56] Alexios Mylonas, Anastasia Kastania, and Dimitris Gritzalis. 2013. Delegate the smartphone user? Security awareness in smartphone platforms. *Computers & Security* 34 (2013), 47 – 66. https://doi.org/10.1016/j.cose.2012.11.004

[57] Iyswarya Narayanan, Di Wang, Myeongjae Jeon, Bikash Sharma, Laura Caulfield, Anand Sivasubramaniam, Ben Cutler, Jie Liu, Badriddine Khessib, and Kushagra Vaid. 2016. SSD Failures in Datacenters: What, When and Why? *SIGMETRICS Performance Evaluation* 44, 1 (June 2016).

[58] Jon Oberheide and Charlie Miller. 2012. Dissecting the android bouncer. *SummerCon* (2012).

[59] T. R. Oldham, M. Friendlich, M. A. Carts, C. M. Seidleck, and K. A. LaBel. 2009. Effect of Radiation Exposure on the Endurance of Commercial nand Flash Memory. *IEEE Transactions on Nuclear Science* 56, 6 (Dec 2009), 3280–3284. https://doi.org/10.1109/TNS.2009.2034463

[60] Rahul Pandita, Xusheng Xiao, Wei Yang, William Enck, and Tao Xie. 2013. WHYPER: Towards Automating Risk Assessment of Mobile Applications. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security)*. USENIX, 527–542. https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/pandita

[61] Giuseppe Petracca, Ahmad Atamli, Yuqiong Sun, Jens Grossklags, and Trent Jaeger. 2016. Aware: Controlling App Access to I/O Devices on Mobile Platforms. (apr 2016). arXiv:1604.02171 http://arxiv.org/abs/1604.02171

[62] Samsung. 2017. SAMSUNG SSD Limited Warranty For All Samsung SSDs. Samsung, http://www.samsung.com/semiconductor/minisite/ssd/downloads/warranty/SAMSUNG_SSD_Limited_Warranty_English_US.pdf. (2017).

[63] Sandisk. 2017. Sandisk iNand Embedded Flash Drives. https://www.sandisk.com/oem-design/mobile/inand. (2017).

[64] SanDisk. 2017. SanDisk Product Warranty. SanDisk, https://www.sandisk.com/about/legal/warranty/warranty-table. (2017).

[65] Bianca Schroeder, Raghav Lagisetty, and Arif Merchant. 2016. Flash Reliability in Production: The Expected and the Unexpected. In *14th USENIX Conference on File and Storage Technologies (FAST)*. USENIX Association, 67–80. http://usenix.org/conference/fast16/technical-sessions/presentation/schroeder

[66] Wook Song, Nosub Sung, Byung-Gon Chun, and Jihong Kim. 2014. Reducing Energy Consumption of Smartphones Using User-perceived Response Time Analysis. In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications (HotMobile)*. ACM, Article 20, 6 pages. https://doi.org/10.1145/2565585.2565595

[67] App Store. 2017. Review guidelines. App Store, https://developer.apple.com/app-store/review/. (2017).
[68] Play Store. 2017. Battery Doctor. https://play.google.com/store/apps/details?id=com.ijinshan.kbatterydoctor_en&hl=en. (2017).
[69] The TechReport. 2015. The SSD Endurance Experiment: They're all dead. TechReport, http://techreport.com/review/27909/the-ssd-endurance-experiment-theyre-all-dead. (March 2015).
[70] Toshiba. 2016. Product manual, SG5 client SSD series. Toshiba, https://toshiba.semicon-storage.com/content/dam/toshiba-ss/asia-pacific/docs/product/storage/product-manual/SSD-SG5-Series-Brochure-Revision1.0.pdf. (2016).
[71] Toshiba. 2017. e·MMC™| TOSHIBA Storage & Electronic Devices Solutions Company. https://toshiba.semicon-storage.com/us/product/memory/nand-flash/mlc-nand/emmc.html. (2017).
[72] Toshiba. 2017. Storage Products Warranty. Toshiba, http://toshiba.semicon-storage.com/us/product/storage-products.html. (2017).
[73] Harvey Tuch, Cyprien Laplace, Kenneth C. Barr, and Bi Wu. 2012. Block Storage Virtualization with Commodity Secure Digital Cards. In *Proceedings of the 8th ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments (VEE)*. ACM, 191–202. https://doi.org/10.1145/2151024.2151050
[74] Tielei Wang, Kangjie Lu, Long Lu, Simon Chung, and Wenke Lee. 2013. Jekyll on iOS: When Benign Apps Become Evil. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security)*. USENIX, 559–572. https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/wang_tielei
[75] Wired. 2015. In Less Than Two Years, a Smartphone Could Be Your Only Computer. Wired, https://www.wired.com/2015/02/smartphone-only-computer/. (2015).
[76] Guanying Wu and Xubin He. 2012. Delta-FTL: Improving SSD Lifetime via Exploiting Content Locality. In *Proceedings of the 7th ACM European Conference on Computer Systems (EuroSys)*. ACM, 253–266. https://doi.org/10.1145/2168836.2168862
[77] Qi Wu, Guiqiang Dong, and Tong Zhang. 2011. Exploiting Heat-accelerated Flash Memory Wear-out Recovery to Enable Self-healing SSDs. In *Proceedings of the 3rd USENIX Conference on Hot Topics in Storage and File Systems (HotStorage'11)*. USENIX Association, Berkeley, CA, USA, 4–4. http://dl.acm.org/citation.cfm?id=2002218.2002222
[78] Zhi Xu and Sencun Zhu. 2015. SemaDroid: A Privacy-Aware Sensor Management Framework for Smartphones. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy (CODASPY)*. ACM, 61–72. https://doi.org/10.1145/2699026.2699114
[79] S. y. Park, E. Seo, J. Y. Shin, S. Maeng, and J. Lee. 2010. Exploiting Internal Parallelism of Flash-based SSDs. *IEEE Computer Architecture Letters* 9, 1 (Jan 2010), 9–12. https://doi.org/10.1109/L-CA.2010.3
[80] Chengen Yang, Hsing-Min Chen, TrevorN. Mudge, and Chaitali Chakrabarti. 2014. Improving the Reliability of MLC NAND Flash Memories Through Adaptive Data Refresh and Error Control Coding. *Journal of Signal Processing Systems* 76, 3 (2014), 225–234. https://doi.org/10.1007/s11265-014-0880-5
[81] Chanmin Yoon, Dongwon Kim, Wonwoo Jung, Chulkoo Kang, and Hojung Cha. 2012. AppScope: Application Energy Metering Framework for Android Smartphone Using Kernel Activity Monitoring. In *Presented as part of the USENIX Annual Technical Conference (USENIX ATC)*. USENIX, 387–400. https://www.usenix.org/conference/atc12/technical-sessions/presentation/yoon
[82] Yajin Zhou and Xuxian Jiang. 2012. Dissecting android malware: Characterization and evolution. In *IEEE Symposium on Security and Privacy*. IEEE, 95–109.
[83] Meng Zhu and Kai Shen. 2016. Energy Discounted Computing on Multi-core Smartphones. In *USENIX Annual Technical Conference (USENIX ATC)*. USENIX Association, 129–141. https://www.usenix.org/conference/atc16/technical-sessions/presentation/zhu