Why and How to Increase SSD Performance Transparency

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

Philipp Gühring
Futureware
pg@futureware.at

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

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

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

Abstract

Even on modern SSDs, I/O scheduling is a first-order performance concern. However, it is unclear how best to optimize I/O patterns for SSDs, because a complex layer of proprietary firmware hides many principal aspects of performance, as well as SSD lifetime. Losing this information leads to research papers drawing incorrect conclusions about prototype systems, as well as real-world systems realizing sub-optimal performance and lifetime. It is our position that a useful performance model of a foundational system component is essential, and the community should support efforts to construct models of SSD performance. We show examples from the literature and our own measurements that illustrate serious limitations of current SSD modeling tools and disk statistics. We observe an opportunity to resolve this problem by reverse engineering SSDs, leveraging recent trends toward component standardization within SSDs. This paper presents a feasibility study and initial results to reverse engineer a commercial SSD's firmware, and discusses limitations and open problems.

CCS Concepts • **Hardware → External storage**; • **Software and its engineering → Operating systems**;

ACM Reference Format:
Aviad Zuck, Philipp Gühring, Tao Zhang, Donald E. Porter, and Dan Tsafrir. 2019. Why and How to Increase SSD Performance Transparency. In *Workshop on Hot Topics in Operating Systems (HotOS '19), May 13–15, 2019, Bertinoro, Italy.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3317550.3321430

HotOS '19, May 13–15, 2019, Bertinoro, Italy

ACM ISBN 978-1-4503-6727-1/19/05…$15.00
https://doi.org/10.1145/3317550.3321430

1 Introduction

System designers need effective performance models for the underlying persistent storage devices; there is a long history of optimizations in file system and database design that try to minimize seeks on tape [1–4] and on mechanical disks [5–7]. On tape or rotating disks, the performance model is largely dictated by the mechanical components of the device, and relatively easy to infer [8, 9]. Although HDDs have some onboard firmware that can optimize requests, research has shown repeatedly that the OS can further improve performance by reordering requests before they are presented to the underlying device, to better match the performance model of the device [10, 11].

A number of papers have also demonstrated performance improvements through better I/O scheduling on newer, flash-based SSDs [12–15], despite the fact that SSDs are much faster and have a narrower gap between sequential and random I/O performance [16, 17]. For backward-compatibility and faster adoption, SSDs present a logical block address (LBA) interface comparable to an HDD, hiding performance-relevant device internals.

Thus, system designers struggle to optimize performance on SSDs without a good performance model. Worse, SSD vendors have strong incentives to hide the performance, as well as the wear-out rate of their devices. Although there are some well-understood, hardware-level performance issues—such as larger I/Os performing better than smaller ones [16, 18] and bank-level parallelism [19, 20]—a complex flash translation layer (FTL) in device firmware can also create significant performance and lifetime artifacts that are hard to optimize for. In particular, this firmware can create artifacts that affect the variance of writes. Such variance can be particularly irritating in a system where a write can go to a different device or location to avoid an FTL-internal delay [21, 22]. Similarly, embedded and safety-critical systems often use flash, and care more about predictability and real-time deadlines than best-case performance; unpredictable, FTL-induced variance leads to otherwise needless pessimism in the analysis and over-provisioning of hardware [23, 24]. From the vendors'

perspective, there is little incentive to disclose FTL details to researchers outside of a strict non-disclosure agreement (NDA), as the FTL is a primary value-add over competitors and typically a trade secret. As a result, a number of papers have reported performance artifacts that they cannot explain without device internals [18, 25–27]. In short, we have a situation where system designers routinely face performance issues they cannot understand without better visibility into FTL behavior.

Consequently, researchers and storage practitioners often resort to various heuristics and conclusions based on observations [16, 27–31] rather than making well-informed decisions. Section 2 illustrates this issue with examples from recent literature. We further discuss the limitations of SSD modeling and show that, even with advanced statistics, it is difficult to accurately predict basic SSD behaviors, such as write amplification.

It is our position that we should not have such a foundational part of modern computer systems be so poorly understood by the community. To be clear, our position is not necessarily against closed-source implementations of device firmware, but that such firmware must provide enough *performance transparency* to enable designers to optimize the system's I/O behavior. For example, recently proposed open-channel SSDs [32, 33] expose the FTL logic to the host, yielding highly predictable I/O performance with perfect scheduling decisions, presenting an upper bound on the improvement potential for SSD transparency. Although understandable, the performance opacity of most SSDs forces system designers to leave performance on the table. We argue that vendors must find a middle ground between protecting the intellectual property of their devices and users' need for performance transparency. If vendors will not provide this information, the community should support efforts to derive it by methods such as measurement and reverse engineering.

This paper further argues that there are new opportunities for reverse engineering SSD behavior, which we discuss in Section 3. The first opportunity lies in the industry-wide standardization of flash package interfacing [34]. We show how, by tracking the electrical signal communication between the SSD's micro-controller and flash packages, we may be able to indirectly infer the policies and mechanisms employed by the device's firmware.

Another, more direct path to understanding SSD inner workings lies in recent work that exploits standardization of component interfaces within the SSD; prior work has looked at this as an avenue for security exploitation [35, 36]. These projects cleverly exploited a combination of firmware binary decompilation and common hardware debugging interfaces to hack SSDs. This paper describes an initial application of this method to a commercial SSD. Our findings uncover basic details of the firmware's translation layer and request handling.

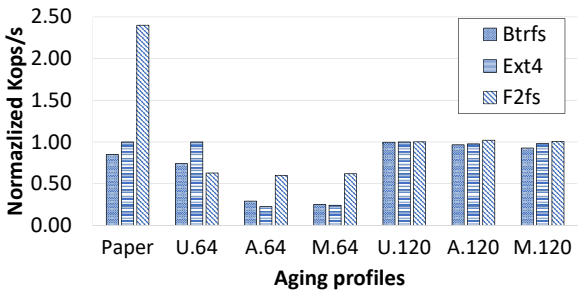


Figure 1. File systems age variably for different SSD models. Reproduced with permission from [25].

2 Motivation

Modern SSDs are a complex embedded hardware platform, including a multi-core micro-controller that processes host I/O requests, and drives multiple, physical flash packages. SSDs employ large RAM caches and utilize hardware accelerators to reduce various computational overheads, such as parity calculation.

SSD FTLs map logical addresses to physical flash pages [37]. Because physical flash pages can only be written once without erasing a larger block, this logical-to-physical mapping is needed to emulate overwriting a logical block by writing that logical block to a new physical location. FTLs are also responsible for many important maintenance tasks, most notably reclaiming space to make way for incoming write requests [22, 38–43]. Furthermore, since flash memories are increasingly constrained in the number of writes they can endure [29, 30, 44, 45], FTLs employ a plethora of methods to extend SSD lifetime [46].

The increasing complexity of SSDs has driven a large body of research on SSD design optimizations to improve performance, power efficiency, and lifetime. Despite these efforts, we know precious little about where real-life devices fit in this design space. Perfect knowledge of the SSD's FTL has the potential for significant improvements in various performance and lifetime aspects, as demonstrated in open-channel SSDs [32, 33]. For instance, by leveraging SSD internals exposed by an open-channel SSD, Wang et al. improve LevelDB performance by more than 4× [47]. Although traditional SSDs and FTLs are just as complex as their open-channel counterparts, their internal scheduling decisions are sub-optimal by design, as the FTL lacks context for likely future accesses or relative priority among concurrent requests.

Not understanding how commercial SSDs work has long frustrated researchers and led to inaccurate, inconsistent, or inconclusive results. For example, the authors of F2FS [48] explored data fragmentation and concluded that "F2FS maintained the performance improvement ratio of two or more over EXT4 across the board" over aged and unaged file systems and devices. However, more recently, Kadekodi et al. [25] explored the performance variance in aging SSDs and concluded that "since FTLs are proprietary, users typically have

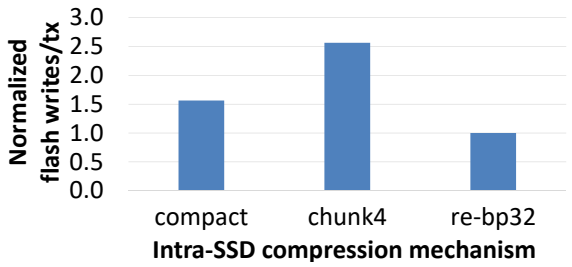


Figure 2. Intra-SSD compression schemes incur a variable number of flash writes per OLTP transaction, normalized to the re-bp32 scheme [49].

no insight into how well (or poorly) an FTL has aged". Specifically, Kadekodi et al. reproduced the simulated file server experiment from the F2FS paper [48] using more devices and aging profiles. Figure 1 (based on their Figure 7 [25]) illustrates their results for a 64GB and 120GB SSD, when the system is unaged (U) and following two different aging processes (A and M). This result indicates that, contrary to the result in the F2FS paper [48], the performance ratio can vary significantly for different SSD models as the system ages. Similarly, Conway et al. [18] conclude that "the gap between sequential and random I/O speeds [in SSDs] is hard to explain conclusively without vendor support" when trying to explain how file system fragmentation affects SSDs. Hao et al. [26] also find that SSDs from some vendors display "much less performance stability".

Another illustration of the importance of understanding FTL internals pertains to intra-SSD compression, a known technique used in commercial SSDs [50, 51] to reduce physical writes [46, 49, 52]. Zuck et al. [49] explored the effectiveness of different intra-SSD compression schemes under OLTP workloads. For example, *chunk4* compresses 16KB worth of data together, while the *compact* method straightforwardly compresses each incoming 4KB request separately and writes it to the log's head. Figure 2 illustrates that, for highly compressible data, different schemes can result in up to 156% more writes to flash than optimal for every OLTP transaction. Consequently, this internal, implementation-specific, FTL feature can significantly affect device lifetime and performance.

2.1 SSD Models are Low Fidelity

Modeling is an important tool in predicting device behaviors and understanding their internals. However, SSDs are difficult to model accurately for several reasons. First, some mechanisms are only used during periods of inactivity, making them virtually "unpredictable background operations" [28]. Some notable examples include idle-time garbage collection [53], page refreshing [54, 55], and self-healing [56, 57] for which the extent of use in commercial products is unknown.

Second, some modeling tools rely on speculation or proprietary knowledge, which may not necessarily apply to different SSD models. For example, the recently proposed MQSim [58]

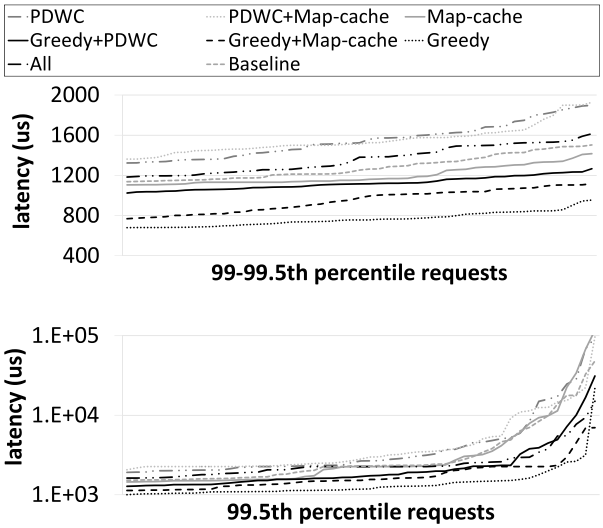
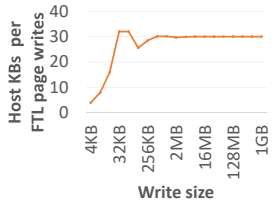


Figure 3. 99th percentile random write latencies for synthetic random write workloads for different FTLs. Requests are ordered in x-axis by latency.

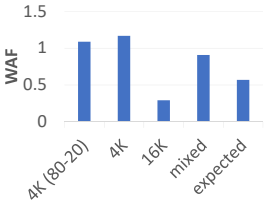
tool verified its accuracy for several commercial SSDs by "enabling all of the advanced GC mechanisms", and using prior knowledge of the on-board DRAM as well as "the specifications of the flash chips used in two of the SSD devices". Unfortunately, knowledge on memory and flash specifications is typically only available under strict NDAs. Furthermore, it is not clear which garbage collection mechanisms are implemented in a given SSD model.

Finally, and crucially, the increasingly proprietary and complex nature of SSDs makes it difficult to correctly understand their internal mechanisms. Consequently, the accuracy of some tools is often rebutted by succeeding works [58, 59]. For example, VSSIM [60] claimed to model an existing SSD with "3% performance offset" but is deemed highly inaccurate by MQSim [58] and "not scalable" by its direct successor FEMU [59], which claims up to 38% performance inaccuracy. Many tools self-validate on only one or no real devices [28, 59–61]. Even tools that were self-validated on several devices can have trouble achieving high fidelity. For example, SSDCheck [14] reports poor tail-latency fidelity, stating that "the lack of publicly available information along with the complex SSD behaviors makes it extremely difficult to construct an accurate performance model for modern SSDs".

To illustrate, we repeated one of MQSim's verification experiments by measuring the response times for synthetic random write workloads with increasing I/O request sizes. We repeated the experiment while varying three basic, write-related design features of the simulated FTL versus the baseline configuration: namely, the garbage collection block selection policy [62] (randomized-greedy algorithm or greedy), write cache designation (data or mapping metadata), and the page allocation scheme [63] (CWDP or PDWC). As a point



(a) For a given single, sequential write request (x-axis) the host-level write size divided by the number of FTL NAND page writes, indicating that a NAND page stores roughly 30KB internally.



(b) Write amplification factor (WAF) for random write workloads when run separately and concurrently according to the reported number of "NAND page writes" metric reported by the SSD.

Figure 4. Crucial MX500 write behavior analysis

of reference, the authors of MQSim consider up to an 18% difference between the model and measured performance to "accurately model the performance of real SSDs" [58]. Our tested configurations represent fundamentally different FTLs and demonstrated performance differences only slightly higher than the 18% threshold, indicating that many high-order differences in FTL behavior are within the margin of error for an SSD simulator.

The main take-away, as Figure 3 illustrates, is that the 99th percentile latency response times are highly variable—up to an order of magnitude difference, depending on the FTL behavior. These results suggest that MQSim may also suffer from poor fidelity in emulating the worst cases—precisely what developers need the model for.

2.2 Black-Box Analysis is Error Prone

It is also difficult to accurately model SSDs solely through observing external performance metrics, such as latency and throughput, or even through advanced profiling information, As an example, we investigate the write amplification on a Crucial MX500 SSD. Unlike most drives [27, 64] the MX500 exposes fine-grained write behavior via S.M.A.R.T. statistics, namely "Host Program Page Count" and "FTL Program Page Count", both measured in "NAND Pages," according to the drive's documentation.

To understand the size of a NAND Page, we first ran a simple, sequential write test of increasing sizes, and divided the host-level write size by the number of NAND page writes reported by the FTL. Figure 4a illustrates that the ratio of host-level writes to NAND writes converges at about 30 KB per NAND page, most likely due to RAIN redundancy [43].

Next we ran three different random write workloads on the SSD in its priming stage using the fio benchmarking tool [65]. To reduce interference, each workload managed its own separate section of the logical address space. The first workload issued 4KB requests distributed uniformly and randomly over the LBA space; the second issued 4KB requests using an 80-20 distribution (80% of writes directed to 20% of the LBA space); and the third issued 16KB uniformly random requests.

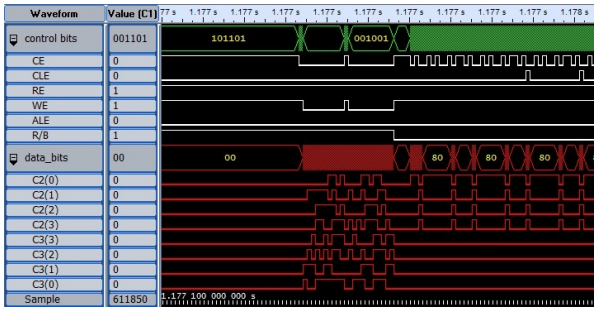


Figure 5. Signal diagram of a flash chip command execution extracted from an OCZ Vertex II SSD.

We ran these workloads separately in 5 minute runs. Finally, we ran all workloads concurrently in a fourth run. We monitored S.M.A.R.T statistics during all runs to measure their resulting write amplification factor (WAF).

Figure 4b illustrates the WAF for each workload. We calculate the expected WAF of the combined fourth run as an average WAF: each sub-workload's WAF is weighted by the number of IOPS the sub-workload issues. This weighting is based on the assumption that FTL metadata write operations are similar for each type of request, regardless of any concurrent operations or the workload type. The results show that, instead of an expected WAF of 0.56, we measure a WAF factor of 0.9 in the mixed run.

Thus, even in this controlled, relatively simple setting, it is unclear how to attribute this increase in WAF to each individual workload. More generally, an attempt to extrapolate expected behavior from black-box measurements was off by nearly a factor of two. Without a better performance model, it is hard to know whether a given set of SSD measurements is representative.

3 Reverse Engineering SSDs

This section discusses two approaches to reverse engineer SSDs. We describe a proof of concept for each proposed method on a commercial SSD, discuss their limitations, and provide initial results on the internals of a real-life SSD.

3.1 Hardware Probes

Flash packages are driven by a controller which issues commands through electrical signals to package pinouts. Over the years, the flash packages, pinouts, and command sets of major flash vendors have consolidated onto similar forms. In 2006 the major flash vendors formed the Open NAND Flash Interface [66] (ONFI) workgroup in order to create a "chip-level standard interface for the attachment of NAND Flash memory to host systems".

This standardization can be utilized for reverse engineering SSDs. By attaching probes to the I/O pinouts of a flash package we can track the high-frequency electrical signals between the micro-controller and the package. Using carefully orchestrated workloads, we can monitor the ensuing

command sequences to the flash packages, and from there, potentially infer firmware polices and mechanisms from how high-level operations map onto and low-level operations.

We demonstrate the applicability of this technique using a single flash package from an OCZ Vertex II 55GB SSD model [67]. We were able to attach probes to all relevant pinouts, safely attach a hardware bridge and extend these probes to a high-end logic analyzer [68]. Next, we verified that the device continues to respond and service commands without malfunctioning, while collecting traces of electrical signals from its ONFI 2.1-compliant flash package pinouts. To illustrate, we instructed the logic analyzer to collect data while we format the SSD with an NTFS file system. Figure 5 shows a sample of the collected trace. At first the signals are flat. Next, we see a short burst of activity, both on the control and data lines, followed by a long burst of data-only transfer in less than 1ms. This burst is most likely correlated to a page write operation which includes an initial command and address input stage, followed by a data transfer stage.

During our investigation, we concluded that there are several technical constraints in applying our proposed technique. Flash packages and SSDs continue to shrink in size. Many modern flash module packages use ball-grid arrays (BGA) whose pinouts are located on the bottom of the package. Electrical delays inserted by the tracing setup may cause the device to malfunction. Packages and pinouts may be coat-protected. These technical obstacles can all be overcome with special equipment and expertise, but may impede the wide-spread use of this method.

Another obstacle is that modern, high-end SSDs can serve hundreds of thousands of requests per second using chips operating at frequencies of hundreds of MHz. Consequently, the probing hardware must be able to handle high-rate tracing and data collection. For example, a suitable logic analyzer costs around $20,000. Finally, and crucially, this technique does not apply to the on-board DRAM cache, which includes data structures and other information that is essential to understanding SSD behavior.

3.2 SSD Hacking

Many embedded systems, including SSD hardware and firmware, have a JTAG-compliant hardware debugging interface [69]. JTAG is an industry standard for testing circuit boards. Using JTAG, SSD developers can inspect and modify on-board memory contents, as well as break and step through code execution.

Post-production JTAG interfaces are often left on the SSD board for possible future data recovery and failure analysis. Several recent works demonstrated how to exploit these debug ports as a security vulnerability [35, 36], for example, by accessing the contents of on-board DRAM and firmware to break device-level encryption.

These security vulnerabilities present new opportunities for systems researchers for understanding SSD internals. We

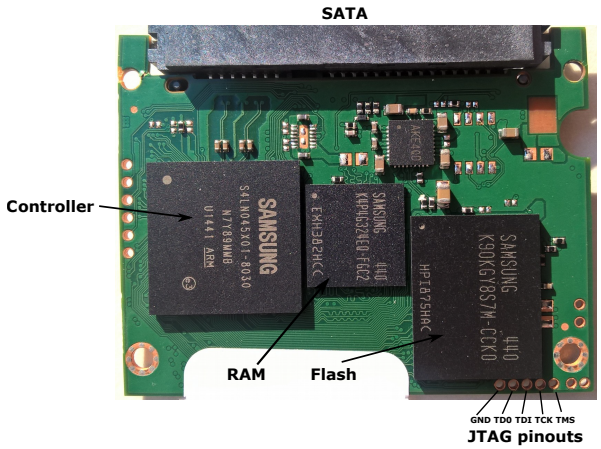


Figure 6. Teardown of the Samsung EVO 840 SSD hardware components and JTAG pinouts.

demonstrate the potential of this approach on a commercial SSD, the Samsung 840 EVO 250GB [70]. The 840 EVO board has a JTAG interface, whose pinout has been previously reverse engineered [71], as illustrated in Figure 6. Given the pinout, one can drive the on-board debug hardware from an external computer, not unlike debugging through GDB on a closed binary.

We drove the SSD from a Novena platform [72] with Linux kernel 4.4 via Linux's pin control subsystem [73], which we attached to the SSD's JTAG pins. Next, we downloaded firmware updates for our drive and used an existing tool to de-obfuscate the retrieved image [74] and used the OpenOCD [75] debugging tool to gain access to the device's memory and various program counters. Finally, through a combination of the code memory map, disassembling and decompiling the firmware binary, and carefully tracing single-sector accesses, we were able to reverse engineer several key features of the device's FTL.

CPU and Flash controller. The controller CPU is an ARM tri-core Cortex-R4. By tracing core activities we observed that one core is dedicated for servicing SATA requests and communication with the host. The remaining cores each manage four out of the device's eight channels by dividing I/O requests according to each 4KB LBA's least significant bit. A dedicated flash controller issues commands to the flash packages, and, commensurate with state of the art literature on SSD power efficiency [76], this controller is turned off during idle times to save power.

Translation Map. The FTL's translation map is composed of eight large arrays dedicated to mapping LBAs to their corresponding physical address. Each core manages four arrays. An additional hashed index is used, presumably to map addresses in the device's pseudo-SLC buffer. With a capacity of 250GB, the device's logical address space is effectively composed of 65M logical addresses. Since each translation entry

only requires $\log_2 65M = 26b$, the translation map theoretically requires only 221MB of on-board memory. However, the mapping table occupies 264MB out of the device's 512MB of on-board memory. We observed some effort to pack the mapping into smaller than 4B logical words, but it did not seem to be as tight as 26b per word; at the time of submission, we did not fully understand the mapping table format. The SSD does not use DRAM to cache data; investigation is ongoing into how the remaining memory is used.

The translation map is persistently stored on flash. The map is divided into chunks, each mapping 117.5MB of the logical address space. A mapping chunk is only loaded on demand, presumably to reduce device boot time.

4 Related Work

A large body of work conjectures SSD internals using black-box analysis and correlative observations. Jung et al. [16] performed systematic tests on various SSDs in order to empirically assert expected SSD behaviors for basic access patterns. Bonetti et al. [77] performed black-box analysis, for forensic analysis and to explore the effectiveness of data erasures. Hao et al. [78] correlate many causal factors affecting the performance of SSDs over time. He et al. [28] used a similar approach to create an "unwritten contract" between file systems and SSDs. Most recently, SSDCheck [14] proposed ways to extract some basic SSD features, such as write buffer size and number of internal volumes, using carefully manipulated access patterns.

There have several recent attempts to reverse engineer storage systems. The Skylight project [79] reverse engineered SMR disk internals by closely inspecting the drive's head movements during operation. Boboila et al. [80] reverse engineered FTLs of several older generation USB sticks by tracing communication from the micro-controller to flash chips. As expected for these simple devices, their findings show trivial garbage collection, mapping schemes, and wear leveling policies.

History and Ethics of Reverse Engineering. Reverse engineering computer software and hardware has a rich history. Many have tried to reverse engineer embedded devices (other than SSDs) of varying complexity to demonstrate security vulnerabilities. Recent examples include hard drives [81], smart light bulbs [82], fitness trackers [83], printers [84], and more [85, 86]. These efforts demonstrate that one can gain useful insights with reasonable effort.

The legality of reverse engineering varies according to each country's laws. Most notably, the US Digital Millennium Copyright Act (DMCA) criminalizes reverse engineering or circumventing software designed to protect copyrighted digital content, such as movies and music. A particularly relevant case law is Lexmark v. Static Control Components, where the court held that reverse engineering printer cartridge authentication firmware did not violate the DMCA [87]. Other precedents deem such efforts legal as long as they "improve interoperability with other components" [90]. Patent protection is generally not applicable to protecting secrecy of a technique, as obtaining a patent requires disclosing the key techniques as part of the patent application, in exchange for a limited monopoly on the technique.

In terms of ethics, a key open question is how to balance the need to protect trade secrets of storage vendors, so that they can continue to profitably build devices, with the need of the research community and practitioners to understand and improve upon the state of the art. We note that having well grounded science, which is largely funded by taxpayers, is a significant public interest. It is not clear whether the degree of secrecy around SSD firmware is strictly needed to maintain commercial viability, and a mutually beneficial "middle path" could involve vendors disclosing more detailed performance models. A related issue is responsible disclosure of vulnerabilities; in this case, we used already public information. In cases where one discovers a new vulnerability, care should be taken in how this information is publicized [91, 92].

5 Conclusions

Researchers and system designers are often left in the dark when it comes to the proprietary mechanisms employed in commercial SSDs. Our results indicate that many current performance modeling techniques can be very inaccurate; without more insight into firmware behavior, it is difficult to know whether a sample of measurements is representative. Until vendors commonly open flash internals, as with open-channel SSDs, the next most promising option is to support reverse engineering efforts. Our preliminary study, as well as technology trends, indicate that exploiting the JTAG interface is the most promising option.

Acknowledgments

We thank the anonymous reviewers for their insightful comments on earlier drafts of the work. This research was supported by a grant from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel, grant # 2017702; the United States National Science Foundation (NSF) grant # CNS-1816263, VMware, and was partially supported by the Technion Hiroshi Fujiwara cyber security research center and the Israel cyber directorate.

References

[1] J. C. Browne, K. M. Chandy, R. M. Brown, T. W. Keller, D. F. Towsley, and C. W. Dissly. Hierarchical techniques for the development of realistic models of complex computer systems. *Proceedings of the IEEE*, 63(6):966–975, June 1975.

[2] D. Pease, A. Amir, L. V. Real, B. Biskeborn, M. Richmond, and A. Abe. The linear tape file system. In *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–8, May 2010.

[3] H. Robinson. A mass storage subsystem using ANSI X3B6 ID-1 recorders. In *Tenth IEEE Symposium on Mass Storage Systems*, volume 00 of *MASS*, pages 43–45, 1990.

[4] Xianbo Zhang, David Du, Jim Hughes, and Ravi Kavuri. HPTFS: A high performance tape file system. In *in Proceedings of 14th NASA Goddard/23rd IEEE conference on Mass Storage System and Technologies*, MSST, 2006.

[5] E. Coffman, L. Klimko, and B. Ryan. Analysis of scanning policies for reducing disk seek times. *SIAM Journal on Computing*, 1(3):269–279, 1972.

[6] Micha Hofri. Disk scheduling: FCFS vs. SSTF revisited. *Communications of ACM*, 23(11):645–653, November 1980.

[7] Margo Seltzer, Peter Chen, and John Ousterhout. Disk scheduling revisited. In *Proceedings of the winter USENIX technical conference*, 1990.

[8] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. *Computer*, 27(3):17–28, March 1994.

[9] The DiskSim simulation environment version 1.0 reference manual. https://cse.umich.edu/techreports/cse/98/CSE-TR-358-98.pdf, 1998.

[10] H. Frank. Analysis and optimization of disk storage devices for time-sharing systems. *J. ACM*, 16(4):602–620, October 1969.

[11] David Boutcher and Abhishek Chandra. Does virtualization make disk scheduling passé? *SIGOPS Oper. Syst. Rev.*, 44(1):20–24, March 2010.

[12] Jiacheng Zhang, Jiwu Shu, and Youyou Lu. ParaFS: A log-structured file system to exploit the internal parallelism of flash devices. In *Proceedings of the 2016 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC, 2016.

[13] B. Mao and S. Wu. Exploiting request characteristics and internal parallelism to improve ssd performance. In *33rd IEEE International Conference on Computer Design*, ICCD, 2015.

[14] J. Kim, P. Park, J. Ahn, J. Kim, J. Kim, and J. Kim. Ssdcheck: Timely and accurate prediction of irregular behaviors in black-box SSDs. In *51st Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO, 2018.

[15] J. Kim, E. Lee, and S. H. Noh. I/o scheduling schemes for better i/o proportionality on flash-based SSDs. In *IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, MASCOTS, 2016.

[16] Myoungsoo Jung and Mahmut Kandemir. Revisiting widely held ssd expectations and rethinking system-level implications. In *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS, 2013.

[17] Sungjoon Koh, Changrim Lee, Miryeong Kwon, and Myoungsoo Jung. Exploring system challenges of ultra-low latency solid state drives. In *10th USENIX Workshop on Hot Topics in Storage and File Systems*, HotStorage, 2018.

[18] Alex Conway, Ainesh Bakshi, Yizheng Jiao, William Jannen, Yang Zhan, Jun Yuan, Michael A. Bender, Rob Johnson, Bradley C. Kuszmaul, Donald E. Porter, and Martin Farach-Colton. File systems fated for senescence? nonsense, says science! In *15th USENIX Conference on File and Storage Technologies*, FAST, 2017.

[19] Yang Hu, Hong Jiang, Dan Feng, Lei Tian, Hao Luo, and Shuping Zhang. Performance impact and interplay of ssd parallelism through advanced commands, allocation strategy and data granularity. In *Proceedings of the International Conference on Supercomputing*, ICS, 2011.

[20] C. Gao, L. Shi, M. Zhao, C. J. Xue, K. Wu, and E. H. . Sha. Exploiting parallelism in I/O scheduling for access conflict minimization in flash-based solid state drives. In *30th Symposium on Mass Storage Systems and Technologies*, MSST, pages 1–11, June 2014.

[21] Feng Chen, David A. Koufaty, and Xiaodong Zhang. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. *SIGMETRICS Performance Evaluation Review*, 37(1):181–192, June 2009.

[22] Peter Desnoyers. Analytic modeling of ssd write performance. In *Proceedings of the 5th Annual International Systems and Storage Conference*, SYSTOR, pages 12:1–12:10, New York, NY, USA, 2012. ACM.

[23] Li-Pin Chang, Tei-Wei Kuo, and Shi-Wu Lo. Real-time garbage collection for flash-memory storage systems of real-time embedded systems. *ACM Trans. Embed. Comput. Syst.*, 3(4):837–863, November 2004.

[24] H. Cho, , and Y. I. Eom. Kast: K-associative sector translation for nand flash memory in real-time systems. In *2009 Design, Automation Test in Europe Conference Exhibition*, pages 507–512, April 2009.

[25] Saurabh Kadekodi, Vaishnavh Nagarajan, and Gregory R. Ganger. Geriatrix: Aging what you see and what you don't see. a file system aging approach for modern storage systems. In *2018 USENIX Annual Technical Conference*, USENIX ATC, 2018.

[26] Mingzhe Hao, Gokul Soundararajan, Deepak Kenchammana-Hosekote, Andrew A. Chien, and Haryadi S. Gunawi. The tail at store: A revelation from millions of hours of disk and SSD deployments. In *14th USENIX Conference on File and Storage Technologies*, FAST, 2016.

[27] Haryadi S. Gunawi, Riza O. Suminto, Russell Sears, Casey Golliher, Swaminathan Sundararaman, Xing Lin, Tim Emami, Weiguang Sheng, Nematollah Bidokhti, Caitie McCaffrey, Gary Grider, Parks M. Fields, Kevin Harms, Robert B. Ross, Andree Jacobson, Robert Ricci, Kirk Webb, Peter Alvaro, H. Birali Runesha, Mingzhe Hao, and Huaicheng Li. Fail-slow at scale: Evidence of hardware performance faults in large production systems. In *16th USENIX Conference on File and Storage Technologies*, FAST, 2018.

[28] Jun He, Sudarsun Kannan, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. The unwritten contract of solid state drives. In *Proceedings of the Twelfth European Conference on Computer Systems*, EuroSys, 2017.

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

[30] Bianca Schroeder, Raghav Lagisetty, and Arif Merchant. Flash reliability in production: The expected and the unexpected. In *14th USENIX Conference on File and Storage Technologies*, FAST. USENIX Association, 2016.

[31] MarÃa F. Borge, Florin Dinu, and Willy Zwaenepoel. Understanding and taming SSD read performance variability: HDFS case study, 2019.

[32] Matias Bjørling, Javier González, and Philippe Bonnet. Lightnvm: The linux open-channel ssd subsystem. In *Proceedings of the 15th Usenix Conference on File and Storage Technologies*, FAST, 2017.

[33] Jian Ouyang, Shiding Lin, Song Jiang, Zhenyu Hou, Yong Wang, and Yuanzheng Wang. Sdf: Software-defined flash for web-scale internet storage systems. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS, 2014.

[34] Open NAND flash interface. http://www.onfi.org.

[35] Lucian Cojocar, Kaveh Razavi, and Herbert Bos. Off-the-shelf embedded devices as platforms for security research. In *Proceedings of the 10th European Workshop on Systems Security*, EuroSec, 2017.

[36] Carlo Meijer and Bernard Van Gastel. Self-encrypting deception: weaknesses in the encryption of solid state drives (SSDs). https://www.ru.nl/publish/pages/909282/draft-paper.pdf, 2018.

[37] A. Birrell, M. Isard, C. Thacker, and T. Wobber. A design for high-performance flash disks. *ACM SIGOPS Operating Systems Review*, 41(2):88–93, 2007.

[38] Xiao-Yu Hu, Evangelos Eleftheriou, Robert Haas, Ilias Iliadis, and Roman Pletka. Write amplification analysis in flash-based solid state drives. In *Proceedings of The Israeli Experimental Systems Conference*, SYSTOR, pages 10:1–10:9, New York, NY, USA, 2009. ACM.

[39] Radu Stoica and Anastasia Ailamaki. Improving flash write performance by using update frequency. *Proceedings of the VLDB Endowment*, 6(9):733–744, July 2013.

[40] Michael Wu and Willy Zwaenepoel. envy: A non-volatile, main memory storage system. *SIGPLAN Not.*, 29(11):86–97, November 1994.

[41] Dongchul Park and D.H.C. Du. Hot data identification for flash-based storage systems using multiple bloom filters. In *IEEE 27th Symposium on Mass Storage Systems and Technologies*, MSST, 2011.

[42] B. Van Houdt. On the necessity of hot and cold data identification to reduce the write amplification in flash-based SSDs. *Performance Evaluation*, 82(0):1 – 14, 2014.

[43] Shiqin Yan, Huaicheng Li, Mingzhe Hao, Michael Hao Tong, Swaminathan Sundararaman, Andrew A. Chien, and Haryadi S. Gunawi. Tinytail flash: Near-perfect elimination of garbage collection tail latencies in NAND ssds. In *15th USENIX Conference on File and Storage Technologies*, FAST, 2017.

[44] Peter Desnoyers. What systems researchers need to know about nand flash. In *Proceedings of the 5th USENIX Conference on Hot Topics in Storage and File Systems*, HotStorage, 2013.

[45] Laura M. Grupp, John D. Davis, and Steven Swanson. The bleak future of NAND flash memory. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies*, FAST. USENIX Association, 2012.

[46] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu. Error characterization, mitigation, and recovery in flash-memory-based solid-state drives. *Proceedings of the IEEE*, 105(9):1666–1704, Sept 2017.

[47] Peng Wang, Guangyu Sun, Song Jiang, Jian Ouyang, Shiding Lin, Chen Zhang, and Jason Cong. An efficient design and implementation of lsm-tree based key-value store on open-channel ssd. In *Proceedings of the Ninth European Conference on Computer Systems*, EuroSys '14, pages 16:1–16:14, New York, NY, USA, 2014. ACM.

[48] Changman Lee, Dongho Sim, Jooyoung Hwang, and Sangyeun Cho. F2fs: A new file system for flash storage. In *13th USENIX Conference on File and Storage Technologies*, FAST, pages 273–286. USENIX Association, February 2015.

[49] Aviad Zuck, Sivan Toledo, Dmitry Sotnikov, and Danny Harnik. Compression and SSDs: Where and how? In *2nd Workshop on Interactions of NVM/Flash with Operating Systems and Workloads*, INFLOW, 2014.

[50] Intel. How data compression affects performance for IntelÂ® solid state drives. https://www.intel.com/content/www/us/en/support/articles/000006354/memory-and-storage.html.

[51] Kingston. Kingston data reduction technology for longer SSD life and greater performance. https://www.kingston.com/en/ssd/enterprise/technical_brief/what_is_durawrite/.

[52] Xuebin Zhang, Jiangpeng Li, Hao Wang, Kai Zhao, and Tong Zhang. Reducing solid-state storage device write stress through opportunistic in-place delta compression. In *Proceedings of the 14th Usenix Conference on File and Storage Technologies*, FAST, 2016.

[53] J. Lee, Y. Kim, G. M. Shipman, S. Oral, and J. Kim. Preemptible i/o scheduling of garbage collection for solid state drives. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(2):247–260, 2013.

[54] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. S. Unsal, and K. Mai. Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime. In *IEEE 30th International Conference on Computer Design*, ICCD, pages 94–101, Sept 2012.

[55] Chengen Yang, Hsing-Min Chen, Trevor Mudge, and Chaitali Chakrabarti. Improving the reliability of MLC NAND flash memories through adaptive data refresh and error control coding. *Journal of Signal Processing Systems*, 76(3):225–234, 2014.

[56] Qi Wu, Guiqiang Dong, and Tong Zhang. 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, 2011.

[57] R. Chen, Y. Wang, D. Liu, Z. Shao, and S. Jiang. Heating dispersal for self-healing NAND flash memory. *IEEE Transactions on Computers*, 66(2):361–367, Feb 2017.

[58] Arash Tavakkol, Juan Gómez-Luna, Mohammad Sadrosadati, Saugata Ghose, and Onur Mutlu. MQSim: A framework for enabling realistic studies of modern multi-queue SSD devices. In *16th USENIX Conference on File and Storage Technologies*, FAST, 2018.

[59] Huaicheng Li, Mingzhe Hao, Michael Hao Tong, Swaminathan Sundararaman, Matias Bjørling, and Haryadi S. Gunawi. The CASE of FEMU: Cheap, accurate, scalable and extensible flash emulator. In *16th USENIX Conference on File and Storage Technologies*, FAST, pages 83–90, 2018.

[60] J. Yoo, Y. Won, J. Hwang, S. Kang, J. Choil, S. Yoon, and J. Cha. Vssim: Virtual machine based SSD simulator. In *IEEE 29th Symposium on Mass Storage Systems and Technologies*, MSST, pages 1–14, May 2013.

[61] M. Jung, J. Zhang, A. Abulila, M. Kwon, N. Shahidi, J. Shalf, N. S. Kim, and M. Kandemir. SimpleSSD: Modeling solid state drives for holistic system simulation. *IEEE Computer Architecture Letters*, 17(1):37–41, Jan 2018.

[62] Benny Van Houdt. A mean field model for a class of garbage collection algorithms in flash-based solid state drives. In *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS, 2013.

[63] Arash Tavakkol, Pooyan Mehrvarzy, Mohammad Arjomand, and Hamid Sarbazi-Azad. Performance evaluation of dynamic page allocation strategies in ssds. *ACM TOPMECS*, 1(2):7:1–7:33, June 2016.

[64] Christian Franke, Philip Williams, and Bruce Allen. smartmontools drive database file). https://github.com/smartmontools/smartmontools/blob/master/smartmontools/drivedb.h, 2018.

[65] fio benchmarking tool. https://github.com/axboe/fio.

[66] New Group Simplifies NAND Flash Integration. http://www.onfi.org/news-events/new-group-simplifies-nand-flash-integration/, May 2006.

[67] Newegg. Ocz vertex 2 2.5" 55gb sata ii mlc internal solid state drive (ssd) oczssd2-2vtxe60g. https://www.newegg.com/Product/Product.aspx?Item=N82E16820227550.

[68] Tektronix. Tla7000 logic analyzer. https://www.tek.com/logic-analyzer/tla7000.

[69] Intel. JTAG 101 IEEE 1149.x and software debug. https://digitallibrary.intel.com/content/dam/ccl/public/jtag-101-ieee-1149x-paper.pdf, 2009.

[70] Samsung. 840 EVO SATA III. https://www.samsung.com/us/support/owners/product/840-evo-sata-iii.

[71] Samsung EVO 840 JTAG pinouts. https://twitter.com/bsmtiam/status/623241828033114112.

[72] Novena open hardware computing platform. https://www.kosagi.com/w/index.php?title=Novena_Main_Page.

[73] Linux Kernel Archives. Pinctrl (pin control) subsystem. https://www.kernel.org/doc/Documentation/pinctrl.txt.

[74] Daming Dominic Chen. Samsung ssd firmware deobfuscation utility. github, https://github.com/ddcc/drive_firmware/tree/master/samsung.

[75] Open on-chip debugger. http://openocd.org.

[76] Seokhei Cho, Changhyun Park, Youjip Won, Sooyong Kang, Jaehyuk Cha, Sungroh Yoon, and Jongmoo Choi. Design tradeoffs of SSDs: From energy consumption's perspective. *Transactions on Storage*, 11(2):8:1–8:24, 2015.

[77] Gabriele Bonetti, Marco Viglione, Alessandro Frossi, Federico Maggi, and Stefano Zanero. A comprehensive black-box methodology for testing the forensic characteristics of solid-state drives. In *Proceedings of the 29th Annual Computer Security Applications Conference*, ACSAC, 2013.

[78] Mingzhe Hao, Gokul Soundararajan, Deepak Kenchammana-Hosekote, Andrew A. Chien, and Haryadi S. Gunawi. The tail at store: A revelation from millions of hours of disk and SSD deployments. In *14th USENIX Conference on File and Storage Technologies*, FAST, 2016.

[79] Abutalib Aghayev and Peter Desnoyers. Skylight—a window on shingled disk operation. In *13th USENIX Conference on File and Storage Technologies*, FAST, 2015.

[80] Simona Boboila and Peter Desnoyers. Write endurance in flash drives: Measurements and analysis. In *Proceedings of the 8th USENIX Conference on File and Storage Technologies*, FAST, 2010.

[81] J Domburg. Hard disk hacking. http://spritesmods.com/?art=hddhack, 2013.

[82] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin OFlynn. Iot goes nuclear: Creating a zigbee chain reaction. In *IEEE Symposium on Security and Privacy*, SP, 2017.

[83] Hossein Fereidooni, Jiska Classen, Tom Spink, Paul Patras, Markus Miettinen, Ahmad-Reza Sadeghi, Matthias Hollick, and Mauro Conti. Breaking fitness records without moving: Reverse engineering and spoofing fitbit. In Marc Dacier, Michael Bailey, Michalis Polychronakis, and Manos Antonakakis, editors, *Research in Attacks, Intrusions, and Defenses*, pages 48–69, Cham, 2017. Springer International Publishing.

[84] Ang Cui, Michael Costello, and Salvatore J. Stolfo. When firmware modifications attack: A case study of embedded exploitation. NDSS, 2013.

[85] Andrei Costin, Jonas Zaddach, Aurélien Francillon, and Davide Balzarotti. A large-scale analysis of the security of embedded firmwares. In *23rd USENIX Security Symposium*, USENIX Security, 2014.

[86] Andrei Costin and Jonas Zaddach. Embedded devices security and firmware reverse engineering. BlackHat USA http://s3.eurecom.fr/docs/bh13us_zaddach.pdf, 2013.

[87] Supreme court rebuffs lexmark in toner cartridge fight. Computer World, https://www.computerworld.com/article/2555558/supreme-court-rebuffs-lexmark-in-toner-cartridge-fight.html, 2005.

[88] Samba: An introduction. Samba.org, https://www.samba.org/samba/docs/SambaIntro.html, 2001.

[89] Dvd copy control ass'n, inc. v. bunner. Wikipedia, https://en.wikipedia.org/wiki/DVD_Copy_Control_Ass%27n,_Inc._v._Bunner.

[90] Software, reverse engineering and the law. lwn.net, https://lwn.net/Articles/134642/, 2005.

[91] Department of Health, Education, and Welfare and National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research. The belmont report. ethical principles and guidelines for the protection of human subjects of research. *The Journal of the American College of Dentists*, 81(3):4â€”13, 2014.

[92] M. Bailey, D. Dittrich, E. Kenneally, and D. Maughan. The menlo report. *IEEE Security Privacy*, 10(2):71–75, March 2012.