Generating Realistic Wear Distributions for SSDs

Ziyang Jiao
Syracuse University
zjiao04@syr.edu

Bryan S. Kim
Syracuse University
bkim01@syr.edu

ABSTRACT

We present FF-SSD, a machine learning-based SSD aging framework that generates representative future wear-out states. FF-SSD is accurate (up to 99% similarity), efficient (accelerates simulation time by 2×), and modular (can be integrated with existing simulators and emulators).

CCS CONCEPTS

• **Computing methodologies** → **Simulation tools**; *Supervised learning by regression*; • **Information systems** → *Flash memory*.

KEYWORDS

SSD, wear-out, simulation, machine learning

ACM Reference Format:
Ziyang Jiao and Bryan S. Kim. 2022. Generating Realistic Wear Distributions for SSDs. In *14th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage '22), June 27–28, 2022, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3538643.3539757

1 INTRODUCTION

Understanding the aged behavior of SSDs (solid-state drives) is important because the errors in SSDs increase over time as flash memory wears out [3, 16]. Errors not only corrupt the data the SSD stores (silent data corruption) [2, 6], but also induce fail-slow symptoms where performance degrades as the SSD attempts to correct and prevent these errors [8, 11].

However, no existing SSD development frameworks (such as Amber [7], FEMU [14], and MQSim [22]) consider aging in their design. Aging through pre-conditioning is prohibitively expensive as it takes years' worth of simulation time to reach that aged state. Alternatively, the initial erase count can be pre-set to a higher non-zero value, but this will be an unrealistic wear state because modern SSDs cannot effectively even the wear with its wear leveler [15].

In this work, we propose *Fast-Forwardable SSD* (FF-SSD), a machine learning-based SSD aging framework that generates representative future wear-out states. Within a typical SSD model, many components perform repetitive work. For example, the garbage collector may keep selecting several hot blocks as the victim over a period of time. Similarly, the trigger condition of the wear leveler is computed regularly during the lifetime of the SSD. Thus, the behavior within an SSD can be learned from past executions to predict the future SSD-internal state.

However, the challenges of using a machine learning approach for making online, fine-grained inferences on SSD internal states are two-fold. First, the inference must be accurate. Modern SSDs are complex embedded systems, managing all of their internal resources with background operations such as garbage collection, wear leveling, error handling, and data scrubbing. These internal complexities need to be learned to make the inference highly accurate. Second, the inference must be fast and efficient relative to the simulation time; otherwise, it either brings negligible benefits or even prolongs the overall process. Deep learning models like convolutional neural network or recurrent neural network would introduce more complexities and may result in a slow training and inference performance. To address these, FF-SSD incrementally builds a lightweight regression model for each block to capture the changes in SSD-internal states and predicts their trajectory using the information from past executions. This model would approximate the future wear state of an SSD device if the same workload were to be repeated, resulting in a faster simulation time.

We present the design, implementation, and usage scenarios of the FF-SSD framework[1]. We build FF-SSD with the following quality attributes: (1) *accuracy* by generating realistic distributions that match up to 99% of the full simulation results, (2) *efficiency* by accelerating the simulation by 2× to reach a desired aged state, and (3) *modularity* by building the prediction module that can be integrated across multiple platforms. We evaluate our design using real-world workloads [10, 12, 24] across multiple platforms [5, 7, 14] to demonstrate the usefulness of FF-SSD for SSD aging.

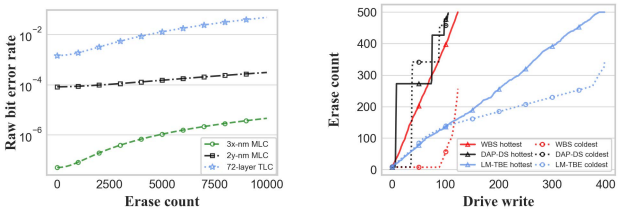
HotStorage '22, June 27–28, 2022, Virtual Event, USA

ACM ISBN 978-1-4503-9399-7/22/06…$15.00
https://doi.org/10.1145/3538643.3539757

[1] FF-SSD is available at https://github.com/ZiyangJiao/FF-SSD.

2 MOTIVATION AND RELATED WORKS

We begin by asking a basic question: does the wear state really matter in SSDs? We then describe the irregularity of the SSD's wear, and briefly discuss related works.



(a) The relationship between RBER and erase count. **(b) The irregularity of block erase count.**

Figure 1: Figure 1a shows the error rate as the erase count increases for three flash memory chips. Figure 1b shows the erase behavior under three real-work workloads [10] with a 256GB SSD. The end of the line indicates the failure of that device.

Fail-slow symptoms. Previous works [8, 11] have shown the fail-slow symptoms manifest in SSDs as they correct and prevent errors whose rate naturally increases due to wear. Figure 1a shows the relationship between raw bit error rate (RBER) and erase count for three flash memory chips derived from RBER models [11]. As an SSD ages and wears out, the error rate increases, which in turn, degrades the performance [11].

Irregularity of block erase count. An SSD consists of hundreds of thousands of flash blocks and they have different erase count trajectories during the lifetime of the device. Figure 1b shows the changes in erase count for the hottest and coldest block under three real-work workloads [10] (WBS, DAP-DS, and LM-TBE), for a 256GiB SSD. As shown in the figure, blocks behave distinctly across three workloads, and the erase counts change at different rates through SSD's lifetime, demonstrating the irregularity of erasure behavior within the SSD. Even though a wear leveler is used [4], the SSD cannot implement perfect wear leveling, similar to the observation from a field study on millions of SSDs [15].

File system aging. Although there are existing research and tools for file system aging [1, 9, 20], these cannot be directly applied to SSD aging. File system aging tools generate a fragmented state of logical block layouts, but SSD aging needs to model the physical aging of blocks as the reliability properties of a young and an old block are different. Preconditioning an SSD is more akin to file system aging by populating and invalidating the address space [21], and cannot sufficiently age the device to an end-of-life state.

Machine learning for simulation. We find two prior works that use machine learning to accelerate simulations: DEVS (Discrete EVent Specification) [19] and CML (Continuous Machine Learning) [18]. At a high-level, DEVS considers multiple model candidates and selects the best one for prediction. This model, however, is not always updated before each prediction stage. On the other hand, CML continuously incorporates the latest data to update its model. However, both are not designed for SSD aging and ignore the complexity of modern SSDs. CML focuses on performance estimation rather than generating internal states, and DEVS is for discrete-event modeling and simulation that assumes that in between events (i.e., external I/O requests), the state of the system does not change [23].

3 SYSTEM DESIGN

We first describe the overall process of learning the wear-out behavior of the SSD to predict the future state, and discuss further optimizations to improve the efficiency of our tool.

3.1 Overall Architecture

Our generative process consists of five phases: (1-2) simulation and observation, (3) training, (4) prediction, and (5) projection, as shown in Figure 2.

Simulation and observation. `FF-SSD` starts from actual simulation during a workload sample to observe the SSD's internal activities, and collect information that will later be used for the prediction. To infer the future erase count of each block, the following features will be recorded at the end of every observation period: (1) the block identifiers, (2) the current erase count, (3) the observation time, and (4) the write amplification factor during this observation period. We filter out features that may seemingly look important but in reality, are not for the prediction. These excluded features relate to the workload characteristics observed from the host side such as workload hotness, access footprint, and I/O size, as their effect on erase count heavily depends on the implementation of the flash translation layer (FTL). The selected features are logged throughout the simulation.

Training. Once enough workload has been sampled, we use the gathered data to build a lightweight regression model for each block and predict the future state. Specifically, the system will extract and normalize the time-series data from the observation and train the model for each block. The main challenge here is to decide how much history should be used for the training process. Since changes in wear for each block are dynamic during the SSD's lifetime, most recent activities are more relevant to the future states. We thus filter out stale information and use only the latest history. These models are updated before each prediction, so they can learn the most recent trend within the SSD.

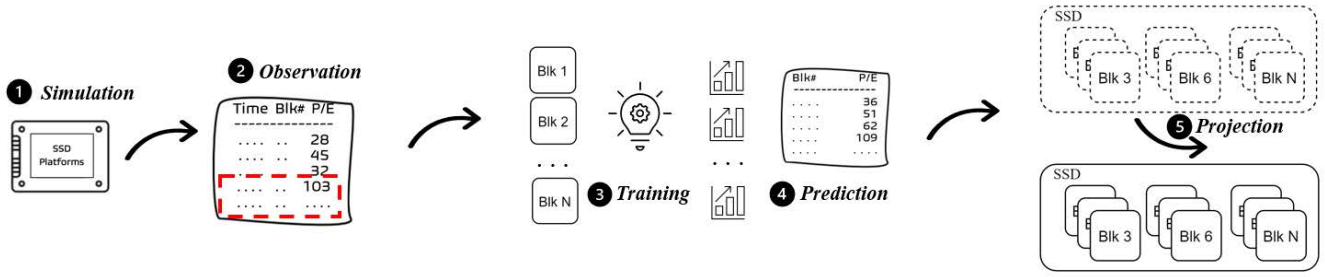


Figure 2: FF-SSD overview. FF-SSD starts from actual (1) simulation to (2) observe the SSD's internal activities, then (3) train lightweight regression models to (4) predict the trajectory for the block's wear out, and (5) project SSD states based on the prediction results.

Prediction. The training phase generates the weights in the models that will be temporarily kept in FF-SSD. For each block, the model is then activated to infer its future erase count based on the *acceleration factor* (*AF*), dictated by the user. In this work, we define *AF* as the ratio of full workloads to the simulated workloads. Thus, there is an inherent tradeoff between prediction accuracy and acceleration efficiency. An aggressive acceleration saves more time to reach an aged state, but at the cost of decreased accuracy compared to moderate acceleration.

Projection. Finally, these generated future states are fed back into the simulation, where it will continue to sample and run workloads. The relevant components are also updated within the platform based on the new state. These stages will repeat until the SSD reaches its desired age.

3.2 Enhancing Efficiency

Since we approach by incrementally building multiple lightweight regression models for all blocks, the prediction overhead is proportional to the number of blocks. Thus, reducing the number of blocks to model and infer would improve the efficiency. We next present an analytic approach to further improve inference efficiency based on distribution modeling.

We assume that the wear distribution of blocks adheres to an underlying measurable distribution $\rho(\cdot)$, such as normal distribution. We then divide all blocks into several groups based on wear conditions (e.g., quantiles) and build one model for each group. Then we estimate the future wear for each block according to the prediction result of these groups and the density function that models the overall underlying distribution.

In this work, we observed that the discrete wear distribution can be almost perfectly matched by a skew-normal distribution in most cases, with skewness α, location μ, and scale parameter σ. Figure 3 shows the histogram of erase counts under WBS workloads, overlaid with the approximating skew-normal distribution with $\alpha = 0.75, \mu = 310, \sigma = 15.1$.

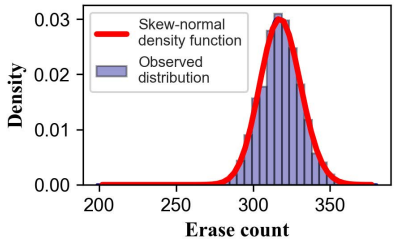


Figure 3: Skew norm fit to the measured distribution. The red line indicates the approximating skew-normal distribution, matching with the observed distribution.

Since no statistical method can affirm whether two distributions are the same, we run Kolmogorov–Smirnov goodness-of-fit test to check if the observed distribution fits our statistical model. We fail to reject the null hypothesis that the wear distribution matches the skew-normal distribution on 10^5 samples with $p > 0.1$.

4 EVALUATION

In this section, we first evaluate the effectiveness of FF-SSD for SSD aging, then explore the optimal point along the tradeoff between accuracy and efficiency. Table 1 outlines the system configurations for our evaluation (FTLSim [5], Amber [7], and FEMU [14]).

For the workload, YCSB-A is from running YCSB [24], VDI is from a virtual desktop infrastructure [13], and the remaining workloads are from Microsoft production servers and Microsoft enterprise servers [10]. The traces are modified to fit the logical capacity of the SSD, and all the requests are aligned to 4KiB boundaries. We use a fully simulated result ($AF = 1$) as the baseline and compare our work against two prior works, DEVS [19] and CML [18].

4.1 Effectiveness of FF-SSD

4.1.1 FTLSim. We first examine the effectiveness of FF-SSD on FTLSim. We configure a 256GiB SSD and set the acceleration factor (*AF*) to be 1.5. We set the endurance limit to 500 for each block and run until the number of bad blocks

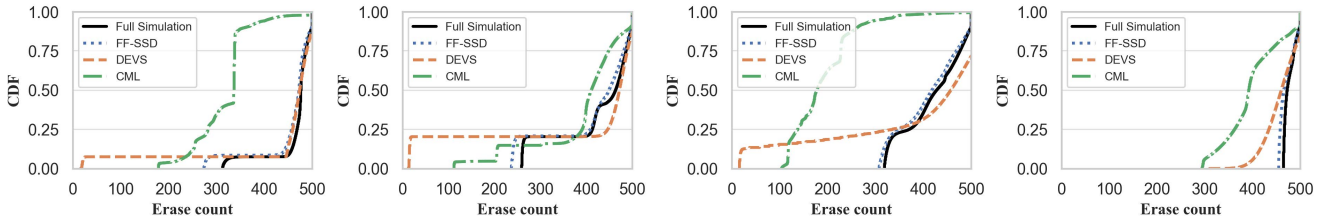


Figure 4: SSD aging until failure on FTLSim. FF-SSD achieves the highest accuracy (91% - 97%) compared to DEVS (60% - 93%) and CML (48% - 84%). The accuracy is computed using the mean difference in erase counts across all blocks relative to their real values from the full simulation.

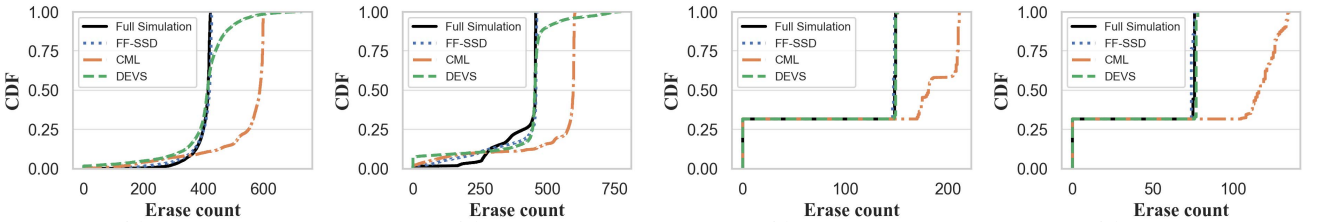


Figure 5: SSD aging with 600 iterations of the workloads on Amber. The accuracy achieved by FF-SSD (88% - 99%) outperforms DEVS (83% - 99%) by a small margin but by a large margin for CML (40% - 60%).

Table 1: Platform specific configurations.

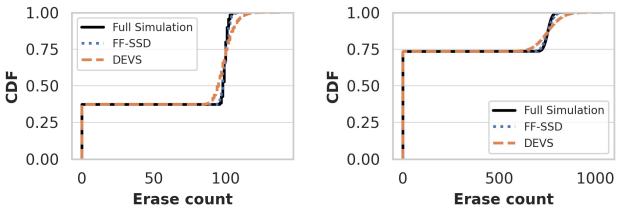
FTLSim			
Page per block	256	Physical capacity	284GiB
Page size	4KiB	Logical capacity	256GiB
Endurance limit	500	Over-provisioning	0.11
Wear leveling	PWL [4]	Garbage collection	Greedy
Amber			
Channels	8	Page size	4KiB
Packages per channel	4	Physical capacity	284GiB
Die per package	2	Logical capacity	256GiB
Plane per die	2	Over-provisioning	0.11
Block per plane	1136	Garbage collection	Greedy
Pages per block	512	Wear leveling	Var-based
FEMU			
Channels	8	Page size	4KiB
Luns per channel	8	Physical capacity	16GiB
Planes per lun	1	Logical capacity	15GiB
Blocks per plane	256	Over-provisioning	0.07
Pages per block	256	Garbage collection	Greedy

exceeds its over-provisioning. In this work, the accuracy is computed using the mean difference in erase counts across all blocks relative to their real values from the full simulation. Figure 4 shows our experiment results: with $AF = 1.5$, FF-SSD continuously learns the behavior within the SSD using the information from the past two iterations of the workload, and then predicts the wear state after one additional iteration. FF-SSD generates the final states of SSD, and achieves the highest accuracy compared to DEVS and CML.

For FF-SSD, we observe that the average accuracy is 94% across all workloads, and as high as 97% for VDI. On the other hand, the overall accuracy for DEVS and CML ranges from 60%-93% and 48%-83%, respectively. The lowest accuracy for FF-SSD is 91% under YCSB workloads, while DEVS and CML only achieve 60% and 48%, underperforming our design by a large margin. Essentially, we accelerate the simulation by 50% with this workload sampling configuration and the predicted distribution closely follows the fully simulated result.

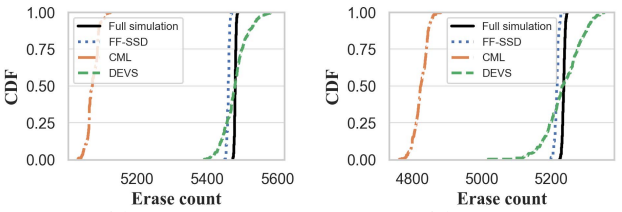
4.1.2 Amber. We next study the performance of FF-SSD on Amber and apply a more aggressive acceleration factor. Specifically, we generate the wear distributions by applying 600 iterations of the workloads with $AF = 2$ and compare them with the baseline. However, the endurance limit is set to be sufficiently large at 100,000, the default value of Amber.

Figure 5 shows the experiment results on Amber. We observe that FF-SSD outperforms DEVS and CML for all the workloads we evaluated. The accuracy ranges from 88% (for WBS) to 99% (for DAP-PS), with a mean of 93%. On the other hand, the accuracy achieved by CML only ranges from 40% (for RAD-AS) to 60% (for WBS), with a mean accuracy of 49%. DEVS presents different behavior on Amber compared to its results on FTLSim. DEVS delivers a similar accuracy



(a) DTRS (b) WBS

Figure 6: Performance comparison of FF-SSD and DEVS on FTLSim without WL. DEVS presents a similar performance to FF-SSD.



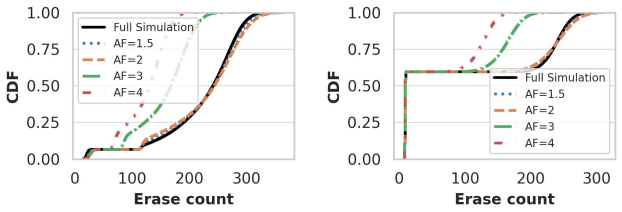
(a) TPCC (b) TPCE

Figure 7: SSD aging with 50 iterations of the workloads on FEMU. FF-SSD delivers a higher accuracy (93% for TPCC and 91% for TPCE) than DEVS (79% for TPCC and 60% for TPCE) and CML (18% for TPCC and 11% for TPCE).

with FF-SSD on Amber, ranging from 83% (for DTRS) to 99% (for DAP-PS), with a mean accuracy of 91%.

To further investigate why DEVS performs well on Amber while not on FTLSim, we find that apart from the implementation details of FTLSim and Amber, these two platforms deployed different wear leveling policies. PWL [4] is used in FTLSim, which adopts an adaptive threshold-driven approach for selecting victim blocks for erases; on the other hand, Amber applies a predefined threshold for the *uneven factor* to determine the trigger condition. We run another experiment to study how the wear leveling policy affects the inference accuracy. Figure 6 shows the result on FTLSim after applying 600 iterations of DTRS and WBS workloads without wear leveling. Without wear leveling in FTLSim, DEVS performs similarly to FF-SSD, indicating that the performance of DEVS is sensitive to the FTL algorithm.

4.1.3 FEMU. The previous experiments show the performance of FF-SSD on SSD simulators. To further improve usability, we turn to study the effectiveness of FF-SSD on FEMU, a state-of-the-art SSD emulator. We use FEMU to emulate a 16GiB SSD and generate the wear distribution after 50 iterations of two Microsoft enterprise server workloads, ME-TPCC and ME-TPCE [10]. The acceleration factor is set to be 2, and all I/O requests are issued by btreplay to the underlying SSD.



(a) LM-TBE (b) RAD-BEFS

Figure 8: The tradeoff between aging accuracy and efficiency. FF-SSD generates an accurate estimation when at least half of the workloads are observed ($AF \leq 2$) and occur more errors beyond that point.

Figure 7 shows the experiment results on FEMU. FF-SSD achieves the highest accuracy (93% for TPCC and 91% for TPCE) compared to DEVS (79% for TPCC and 60% for TPCE) and CML (18% for TPCC and 11% for TPCE). Moreover, the wear states generated by FF-SSD align with the overall shape of our baselines, while DEVS and CML present apparent discrepancies. Comparing this to the previous results, we find that CML underperforms other methods when applied to SSD aging and DEVS only works well under a particular configuration. On the other hand, FF-SSD generates a more realistic distribution on all the platforms while accelerating the emulation by 2×.

4.2 Accuracy and Efficiency Tradeoff

In this section, we quantitatively analyze how different acceleration factors affect the overall accuracy and then provide a conservative AF to balance this tradeoff.

We apply four different acceleration factors (1.5, 2, 3, and 4) to FF-SSD and generate the wear states of running 100 iterations of LM-TBE and MSN-BEFS workloads on FTLSim. The performance comparison is shown in Figure 8. Overall, the accuracy is similar for $AF = 1.5$ and $AF = 2$ and drops distinctively when AF is greater than 2. Specially, for LM-TBE, the accuracy ranges from 56% ($AF = 4$) - 91% ($AF = 1.5$). The accuracy decreases by 4% from $AF = 1.5$ to $AF = 2$, while 19% from $AF = 2$ to $AF = 3$. Similarly, for RAD-BEFS, the accuracy is 98% for $AF = 1.5$ and 97% for $AF = 2$, and decreases by 13% for $AF = 3$ and 20% for $AF = 4$. Given the experiment results above, we conclude that FF-SSD generates an accurate distribution when at least half of the workloads are observed ($AF \leq 2$) and may occur more errors beyond that point.

5 CONCLUSION AND FUTURE WORK

We present *Fast-Forwardable SSD*, to the best of our knowledge, the first ML-based SSD aging framework that generates representative future wear-out states. We examine the effectiveness and usefulness of FF-SSD across state-of-the-art SSD development platforms. Our evaluations show that

FF-SSD generates the desired age states of SSDs with high accuracy under real-world workloads. This work suggests many promising directions, and our immediate plan includes improving the accuracy through adaptive acceleration and predicting the wear states on real SSDs.

Improving accuracy through adaptive acceleration. We plan to implement *outlier detection* [17] on WAF to further improve accuracy and achieve adaptive acceleration control. WAF is defined by the increased writes caused by the background processes, measured as the ratio of total internal to external writes. A stable WAF over a long observation is an indicator that SSD is in a steady-state (i.e., the activeness of SSD background operations keeps at the same level). In this case, FF-SSD submits the estimation with high confidence and a more aggressive AF should be applied to maximize aging efficiency. On the other hand, if the current WAF is an outlier from the past observations, indicative of the changes in host-side workloads or configurations, FF-SSD should adopt a moderate AF or even revoke the estimation to avoid high inference error.

Predicting on the wear states real SSDs. Our work would have greater applicability if we extend our validation to real SSDs where their internal details and states such as FTL logic and erase counts are not accessible. For these cases, we plan to use the information available through the SSDs' SMART (Self-Monitoring, Analysis, and Reporting Technology) interface. For example, the changes in *percentage lifetime used* can be used to estimate how much host data the SSD can sustain under the current workload before its failure.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments and suggestions that help us to improve the quality of this paper. This research was supported, in part, by the National Science Foundation award CNS-2008453.

REFERENCES

[1] Nitin Agrawal, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. 2009. Generating Realistic Impressions for File-System Benchmarking. In *USENIX Conference on File and Storage Technologies (FAST)*. 125–138. http://www.usenix.org/events/fast09/tech/full_papers/agrawal/agrawal.pdf

[2] Lakshmi N. Bairavasundaram, Garth R. Goodson, Bianca Schroeder, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. 2008. An Analysis of Data Corruption in the Storage Stack. In *USENIX Conference on File and Storage Technologies (FAST)*.

[3] Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu. 2017. Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives. *Proc. IEEE* 105, 9 (2017), 1666–1704. https://doi.org/10.1109/JPROC.2017.2713127

[4] Fu-Hsin Chen, Ming-Chang Yang, Yuan-Hao Chang, and Tei-Wei Kuo. 2015. PWL: a progressive wear leveling to minimize data migration overheads for NAND flash devices. In *Design, Automation & Test in Europe Conference & Exhibition, (DATE)*.

[5] Peter Desnoyers. 2012. Analytic modeling of SSD write performance. In *International Systems and Storage Conference (SYSTOR)*.

[6] Aishwarya Ganesan, Ramnatthan Alagappan, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. 2017. Redundancy Does Not Imply Fault Tolerance: Analysis of Distributed Storage Reactions to Single Errors and Corruptions. In *USENIX Conference on File and Storage Technologies (FAST)*.

[7] Donghyun Gouk, Miryeong Kwon, Jie Zhang, Sungjoon Koh, Wonil Choi, Nam Sung Kim, Mahmut Kandemir, and Myoungsoo Jung. 2018. Amber*: Enabling Precise Full-System Simulation with Detailed Modeling of All SSD Resources. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 469–481. https://doi.org/10.1109/MICRO.2018.00045

[8] Haryadi S. Gunawi, Riza O. Suminto, Russell Sears, Casey Golliher, Swaminathan Sundararaman, Xing Lin, Tim Emami, Weiguang Sheng, Nematollah Bidokhti, Caitie McCaffrey, Deepthi Srinivasan, Biswaranjan Panda, Andrew Baptist, 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. 2018. Fail-Slow at Scale: Evidence of Hardware Performance Faults in Large Production Systems. *ACM Trans. Storage* 14, 3, Article 23 (Oct. 2018), 26 pages. https://doi.org/10.1145/3242086

[9] Saurabh Kadekodi, Vaishnavh Nagarajan, and Gregory R. Ganger. 2018. Geriatrix: Aging what you see and what you don't see. A file system aging approach for modern storage systems. In *USENIX Annual Technical Conference (ATC)*. 691–704. https://www.usenix.org/conference/atc18/presentation/kadekodi

[10] Swaroop Kavalanekar, Bruce L. Worthington, Qi Zhang, and Vishal Sharda. 2008. Characterization of storage workload traces from production Windows Servers. In *International Symposium on Workload Characterization (IISWC)*.

[11] Bryan S. Kim, Jongmoo Choi, and Sang Lyul Min. 2019. Design Tradeoffs for SSD Reliability. In *Proceedings of the 17th USENIX Conference on File and Storage Technologies (FAST'19)*. USENIX Association, 281–294.

[12] Chunghan Lee, Tatsuo Kumano, Tatsuma Matsuki, Hiroshi Endo, Naoto Fukumoto, and Mariko Sugawara. 2017. Understanding storage traffic characteristics on enterprise virtual desktop infrastructure. In *ACM International Systems and Storage Conference (SYSTOR)*.

[13] Chunghan Lee, Tatsuo Kumano, Tatsuma Matsuki, Hiroshi Endo, Naoto Fukumoto, and Mariko Sugawara. 2017. Understanding storage traffic characteristics on enterprise virtual desktop infrastructure. In *ACM International Systems and Storage Conference (SYSTOR)*.

[14] Huaicheng Li, Mingzhe Hao, Michael Hao Tong, Swaminatahan Sundararaman, Matias Bjørling, and Haryadi S. Gunawi. 2018. The Case of FEMU: Cheap, Accurate, Scalable and Extensible Flash Emulator. In *Proceedings of the 16th USENIX Conference on File and Storage Technologies (FAST'18)*. USENIX Association, 83–90.

[15] Stathis Maneas, Kaveh Mahdaviani, Tim Emami, and Bianca Schroeder. 2022. Operational Characteristics of SSDs in Enterprise Storage Systems: A Large-Scale Field Study. In *USENIX Conference on File and Storage Technologies (FAST)*. 165–180. https://www.usenix.org/conference/fast22/presentation/maneas

[16] Neal R. Mielke, Robert E. Frickey, Ivan Kalastirsky, Minyan Quan, Dmitry Ustinov, and Venkatesh J. Vasudevan. 2017. Reliability of Solid-State Drives Based on NAND Flash Memory. *Proc. IEEE* 105, 9 (Sep. 2017), 1725–1750. https://doi.org/10.1109/JPROC.2017.2725738

[17] A. O'Hagan and Tom Leonard. 1976. Bayes estimation subject to uncertainty about parameter constraints. *Biometrika* 63, 1 (04 1976), 201–203. https://doi.org/10.1093/biomet/63.1.201

[18] Daniel Christopher Powell and Björn Franke. 2009. Using Continuous Statistical Machine Learning to Enable High-Speed Performance

Prediction in Hybrid Instruction-/Cycle-Accurate Instruction Set Simulators. In *Proceedings of the 7th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '09)*. Association for Computing Machinery, 315–324. https://doi.org/10.1145/1629435.1629478

[19] Hesham Saadawi, Gabriel Wainer, and German Pliego. 2016. DEVS execution acceleration with machine learning. In *2016 Symposium on Theory of Modeling and Simulation (TMS-DEVS)*. 1–6. https://doi.org/10.23919/TMS.2016.7918816

[20] Keith A. Smith and Margo I. Seltzer. 1997. File System Aging - Increasing the Relevance of File System Benchmarks. In *ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*. 203–213. https://doi.org/10.1145/258612.258689

[21] SNIA. 2013. Solid State Storage (SSS) Performance Test Specification (PTS) Enterprise Version 1.1. https://www.snia.org/sites/default/files/technical-work/pts/release/SNIA-SSS-PTS-Enterprise-v1.1.pdf.

[22] Arash Tavakkol, Juan Gómez-Luna, Mohammad Sadrosadati, Saugata Ghose, and Onur Mutlu. 2018. MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices. In *Proceedings of the 16th USENIX Conference on File and Storage Technologies (FAST'18)*. USENIX Association, 49–65.

[23] Yentl Van Tendeloo and Hans Vangheluwe. 2018. Discrete Event System Specification Modeling and simulation. In *2018 Winter Simulation Conference, WSC 2018, Gothenburg, Sweden, December 9-12, 2018*.

[24] Gala Yadgar, Moshe Gabel, Shehbaz Jaffer, and Bianca Schroeder. 2021. SSD-based Workload Characteristics and Their Performance Implications. *ACM Trans. Storage* 17, 1 (2021), 8:1–8:26.