Reducing Solid-State Drive Read Latency by Optimizing Read-Retry
Extended Abstract

Jisung Park[1] Myungsuk Kim[2,3] Myoungjun Chun[2] Lois Orosa[1] Jihong Kim[2] Onur Mutlu[1]
[1]*ETH Zürich* [2]*Seoul National University* [3]*Kyungpook National University*

1. Motivation

This work tackles the performance degradation of modern NAND flash-based SSDs due to a large number of *read-retry* operations essential to ensuring the reliability of stored data. While 3D NAND technology and multi-level cell (MLC) techniques enable continuous increase of storage density, they also negatively affect the reliability of modern NAND flash chips. NAND flash memory stores data as the *threshold voltage (V_{TH})* of each flash cell, which depends on the amount of charge in the cell. New cell designs and organizations in 3D NAND flash memory cause a flash cell to more easily leak its charge [3, 4, 20, 21]. In addition, MLC technology significantly reduces the margin between different V_{TH} levels to store multiple bits in a single cell. Consequently, the V_{TH} level of a 3D NAND flash cell with advanced MLC techniques (e.g., triple-level cell (TLC) [16] or quad-level cell (QLC) [15, 17]) can quickly shift beyond the read-reference voltage V_{REF} after programming, which results in an error when reading the cell.

To provide reliability guarantees for stored data, a modern SSD commonly adopts two main approaches. First, a modern SSD employs a strong *error-correcting code (ECC)* that can detect and correct several tens of raw bit errors (e.g., 72 bits per 1-KiB codeword [24]). Second, when ECC fails to correct all bit errors, the SSD controller performs a *read-retry operation* [6] that reads the erroneous page again with *slightly-adjusted* V_{REF} values. Since bit errors occur due to shift of the V_{TH} levels of flash cells beyond the V_{REF} values, sensing the cells with appropriately-shifted V_{REF} values can greatly reduce the number of raw bit errors [2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 14, 19, 20, 21, 25].

Even though read-retry is essential to ensuring the reliability of modern NAND flash memory, it comes at the cost of significant performance degradation. A read-retry operation *repeats* a retry step that reads the target page while adjusting V_{REF}, until it finds a V_{REF} value that allows the page's raw bit-error rate (RBER) to be lower than the ECC correction capability. Recent work [25] shows that a modern SSD with long retention ages (i.e., how long data is stored) and high program/erase (P/E) cycles (i.e., how many program/erase operations are performed) suffers from a large number of read-retry operations, which in turn increases the read latency linearly with the number of retry steps. Our experimental characterization using 160 real 3D TLC NAND flash chips, in this work, shows that a read frequently incurs *multiple* retry steps even under modest operating conditions (e.g., on average 4.5 retry steps under a 3-month data retention age at *zero* P/E cycles, i.e., at the beginning of SSD lifetime).

Considering that 1) read-retry operations would occur even more frequently in newer NAND flash memory, and 2) many key applications in modern computing systems (e.g., key-value stores and graph analytics) require high read performance on storage devices, it is important to minimize the performance overhead of read-retry operations.

2. Limitations of the State of the Art

To mitigate the performance overhead of read-retry operations, prior works [9, 10, 19, 21, 25] propose to keep track of pre-optimized V_{REF} values for each page to use them for future read requests. For example, Shim et al. [25] propose to read a page using V_{REF} values that have been recently used for a read-retry operation on other pages exhibiting similar error characteristics with the page to read. By starting a read (and retry) operation with the V_{REF} values close to the optimal read-reference voltage (V_{OPT}) values, their proposal significantly reduces the number of retry steps in modern NAND flash-based SSD.

Although prior techniques are effective at reducing the number of retry steps on an erroneous page, read-retry is a fundamental problem *hard to completely avoid* in modern SSDs. For example, the state-of-the-art technique described above can reduce about 70% of retry steps, but *every read* incurs at least three retry steps in an aged SSD [25]. This is because, in modern NAND flash memory, the V_{TH} levels of flash cells change quickly and significantly over time, which makes it extremely difficult to identify the exact V_{REF} values that can avoid read-retry before reading the target page.

3. Key Insights

We identify new opportunities to reduce the read-retry latency by exploiting two advanced features in modern SSDs: 1) the `CACHE READ` *command* [18, 22, 23] and 2) *strong ECC engine*. First, we find that it is possible to reduce the total execution time of a read-retry operation using the `CACHE READ` command that allows a NAND flash chip to perform consecutive reads in a pipelined manner. Since each retry step is effectively the same as a regular page read, the `CACHE READ` also enables concurrent execution of consecutive retry steps.

Second, we find that a large ECC-capability margin exists in the final retry step. This may sound contradictory as a read-retry occurs only when the page's RBER exceeds the ECC capability, i.e., when there is no ECC-capability margin. However, when a read-retry operation succeeds, the page is eventually read *without* any uncorrectable error, which means that there always exists *a positive* ECC-capability margin in the final retry step. We hypothesize that the ECC-capability margin is large due to two reasons. First, a modern SSD uses a *strong* ECC that can correct several tens of raw bit errors in a codeword. Second, in the final retry step, the page can be read by using *near-optimal* V_{REF} values that drastically decrease the page's RBER. If we can leverage the large ECC-

capability margin to reduce the *page-sensing latency* `tR`, we can optimize the latency of *every* retry step. Doing so can allow not only the final retry step to quickly read the page without uncorrectable errors but also the earlier retry steps (which would fail anyway with the default `tR`) to be finished more quickly. To validate our hypothesis, we characterize 1) the ECC-capability margin in each retry step and 2) the impact of reducing `tR` on the page's RBER, using 160 real 3D TLC NAND flash chips. The results show that we can safely reduce `tR` of each retry step by 25% even under the worst operating conditions prescribed by manufacturers (e.g., a 1-year data retention age [13] at 1.5K P/E cycles [24]).

The optimization opportunities that we identify enable new techniques that reduce *the latency of each retry step without increasing the number of retry steps*. Such techniques can effectively complement existing techniques [9, 10, 19, 21, 25] that aim to reduce the *number* of retry steps on an erroneous page.

4. Main Artifacts

We develop two new read-retry mechanisms that effectively reduce the read-retry latency. First, we propose Pipelined Read Retry (PR^2) that performs consecutive retry steps in a pipelined manner using the `CACHE READ` command. Unlike the regular read-retry mechanism that starts a retry step *after* finishing the previous step, PR^2 performs page sensing of a retry step during data transfer of the previous step, which removes data transfer and ECC decoding from the critical path of a read-retry operation, reducing the latency of a retry step by 28.5%.

Second, we introduce Adaptive Read Retry (AR^2) that performs each retry step with reduced page-sensing latency (`tR`), leading to a further 25% latency reduction even under the worst operating conditions. Since reducing `tR` inevitably increases the read page's RBER, an excessive `tR` reduction can potentially cause the final retry step to fail to read the page without uncorrectable errors. This, in turn, introduces one or more additional retry steps, which could increase the overall read latency. To avoid increasing the number of retry steps, AR^2 uses the best `tR` value for a certain operating condition that we find via extensive and rigorous characterization of 160 real 3D NAND flash chips.

Our two techniques require only small modifications to the SSD controller or firmware but no change to underlying NAND flash chips. This makes our techniques easy to integrate into an SSD along with existing read-retry mitigation techniques that aim to reduce the number of retry steps.

We evaluate our techniques using MQSim [1, 26], an open-source multi-queue SSD simulator. We extend MQSim to simulate more realistic read-retry characteristics in modern SSDs based on our real-device characterization results. We also evaluate the performance improvement of our techniques when combined with a state-of-the-art technique [25]. We use twelve real-world workloads with different I/O characteristics while varying the data retention age and P/E-cycle count.

5. Key Results and Contributions

Our main evaluation results show that PR^2 and AR^2, when combined, significantly improve the SSD response time, by up to 50.8% (35.2% on average) over a high-end SSD. Compared to a state-of-the-art baseline [25], our proposal further reduces SSD response time by up to 31.5% (17% on average) in read-dominant workloads.

This paper makes the following key contributions:

- To our knowledge, this work is the first to identify new opportunities to reduce the latency of each retry step by exploiting advanced architectural features in modern SSDs.
- Through extensive and rigorous characterization of 160 real 3D TLC NAND flash chips, we make three new observations on modern NAND flash memory. First, a read-retry operation with multiple retry steps frequently occurs even under modest operating conditions. Second, when a read-retry occurs, there is a large ECC-capability margin in the final retry step even under the worst operating conditions. Third, there is substantial margin in read-timing parameters, which enables safe reduction of the read-retry latency.
- Based on our findings and characterization results, we propose two new techniques, PR^2 and AR^2, which effectively reduce the latency of each retry step, thereby reducing overall read latency and thus improving application performance. Our techniques require only very small changes to the SSD controller or firmware. By reducing the latency of each retry step while keeping the same number of retry steps during a flash read, our proposal effectively complements existing techniques [9, 10, 19, 21, 25] that aim to reduce the number of retry steps, as we empirically demonstrate in the paper.

Why ASPLOS? Our work emphasizes the synergy between two fundamental aspects of storage systems: 1) firmware (i.e., system software) and 2) architecture. Read-retry is an essential mechanism in SSD firmware to ensure the reliability of storage systems, but it can significantly degrade SSD I/O performance that is critical to data-intensive applications. Through extensive real-device characterizations, we introduce new opportunities to significantly reduce read-retry latency by exploiting advanced architectural features widely adopted in modern SSDs. Therefore, this work emphasizes the importance and effectiveness of optimizations based on comprehensive understanding of the storage firmware, architecture, and device characteristics.

Citation for Most Influential Paper Award. This paper proposes new techniques to optimize the read-retry mechanism, which is essential to ensuring the reliability of modern NAND flash-based SSDs at the expense of significant latency overhead. This work is the first to demonstrate that the large reliability margin in modern SSDs can be used to improve the read latency, which has impacted many real SSD designs and inspired many creative follow-on works to achieve high I/O performance by better exploiting the performance-reliability trade-off.

References

[1] MQSim GitHub repository. https://github.com/CMU-SAFARI/MQSim.

[2] Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu. Error characterization, mitigation, and recovery in flash-memory-based solid-state drives. *Proc. IEEE*, 2017.

[3] Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu. Errors in flash-memory-based solid-state drives: Analysis, mitigation, and recovery. *arXiv*, 2017.

[4] Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu. Reliability issues in flash-memory-based solid-state drives: Experimental analysis, mitigation, recovery. In *Inside Solid State Drives*. Springer, 2018.

[5] Yu Cai, Saugata Ghose, Yixin Luo, Ken Mai, Onur Mutlu, and Erich F. Haratsch. Vulnerabilities in MLC NAND flash memory programming: Experimental analysis, exploits, and mitigation techniques. In *HPCA*, 2017.

[6] Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai. Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis. In *DATE*, 2012.

[7] Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai. Threshold voltage distribution in MLC NAND flash memory: Characterization, analysis, and modeling. In *DATE*, 2013.

[8] Yu Cai, Yixin Luo, Saugata Ghose, and Onur Mutlu. Read disturb errors in MLC NAND flash memory: Characterization, mitigation, and recovery. In *DSN*, 2015.

[9] Yu Cai, Yixin Luo, Erich F. Haratsch, Ken Mai, and Onur Mutlu. Data retention in MLC NAND flash memory: Characterization, optimization, and recovery. In *HPCA*, 2015.

[10] Yu Cai, Onur Mutlu, Erich F. Haratsch, and Ken Mai. Program interference in MLC NAND flash memory: Characterization, modeling, and mitigation. In *ICCD*, 2013.

[11] Yu Cai, Gulay Yalcin, Onur Mutlu, Erich F. Haratsch, Adrian Crista, Osman S. Unsal, and Ken Mai. Error analysis and retention-aware management for NAND flash memory. *Intel Tech. J.*, 2013.

[12] Yu Cai, Gulay Yalcin, Onur Mutlu, F. Erich Haratsch, Osman Unsal, Adrian Cristal, and Ken Mai. Neighbor-cell assisted error correction for MLC NAND flash memories. In *SIGMETRICS*, 2014.

[13] Alvin Cox. JEDEC SSD endurance workloads. In *FMS*, 2011.

[14] Aya Fukami, Saugata Ghose, Yixin Luo, Yu Cai, and Onur Mutlu. Improving the reliability of chip-off forensic analysis of NAND flash memory devices. In *DFRWS EU*, 2014.

[15] Hwang Huh, Wanik Cho, Jinhaeng Lee, Yujong Noh, Yongsoon Park, Sunghwa Ok, Jongwoo Kim, Kayoung Cho, Hyunchul Lee, Geonu Kim, Kangwoo Park, Kwanho Kim, Heejoo Lee, Sooyeol Chai, Chankeun Kwon, Hanna Cho, Chanhui Jeong, Yujin Yang, Jayoon Goo, Jangwon Park, Juhyeong Lee, Heonki Kim, Kangwook Jo, Cheoljoong Park, Hyeonsu Nam, Hyunseok Song, Sangkyu Lee, Woopyo Jeong, Kun-Ok Ahn, and Tae-Sung Jung. A 1Tb 4b/cell 96-stacked-WL 3D NAND flash memory with 30MB/s program throughput using peripheral circuit under memory cell array technique. In *ISSCC*, 2020.

[16] Dongku Kang, Woopyo Jeong, Chulbum Kim, Doo-Hyun Kim, Yong Sung Cho, Kyung-Tae Kang, Jinho Ryu, Kyung-Min Kang, Sungyeon Lee, Wandong Kim, Hanjun Lee, Jaedoeg Yu, Nayoung Choi, Dong-Su Jang, Jeong-Don Ihm, Doogon Kim, Young-Sun Min, Moo-Sung Kim, An-Soo Park, Jae-Ick Son, In-Mo Kim, Pansuk Kwak, Bong-Kil Jung, Doo-Sub Lee, Hyunggon Kim, Hyang-Ja Yang, Dae-Seok Byeon, Ki-Tae Park, Kye-Hyun Kyung, and Jeong-Hyuk Choi. 256Gb 3b/cell V-NAND flash memory with 48 stacked WL layers. In *ISSCC*, 2016.

[17] Doo-Hyun Kim, Hyunggon Kim, Sungwon Yun, Youngsun Song, Jisu Kim, Sung-Min Joe, Kyung-Hwa Kang, Joonsuc Jang, Hyun-Jun Yoon, Kanabin Lee, Minseok Kim, Joonsoo Kwon, Jonghoo Jo, Sehwan Park, Jiyoon Park, Jisoo Cho, Sohyun Park, Garam Kim, Jinbae Bang, Heejin Kim, Jongeun Park, Deokwoo Lee, Seonyong Lee, Hwajun Jang, Han-Jun Lee, Donghyun Shin, Jungmin Park, Jungkwan Kim, Jongmin Kim, Kichang Jang, Il Han Park, Seung Hyun Moon, Myung-Hoon Choi, Pansuk Kwak, Joo-Yong Park, Youngdon Choi, Sang-Lok Kim, Seungjae Lee, Dongku Kang, Jeong-Don Lim, Dae-Seok Byeon, Kiwhan Song, Junghwan Choi, Sang Joon Hwang, and Jaeheon Jeong. A 1Tb 4b/cell NAND flash memory with tPROG=2ms, tR=110μs and 1.2Gb/s high-speed IO rate. In *ISSCC*, 2020.

[18] Nancy Leong, Sachit Chandra, and Hounien Chen. Random cache read using a double memory, 2008. US Patent 7,423,915.

[19] Yixin Luo, Saugata Ghose, Yu Cai, Erich F. Haratsch, and Onur Mutlu. Enabling accurate and practical online flash channel modeling for modern MLC NAND flash memory. *IEEE JSAC*, 2016.

[20] Yixin Luo, Saugata Ghose, Yu Cai, Erich F. Haratsch, and Onur Mutlu. Heatwatch: Improving 3D NAND flash memory device reliability by exploiting self-recovery and temperature awareness. In *HPCA*, 2018.

[21] Yixin Luo, Saugata Ghose, Yu Cai, Erich F. Haratsch, and Onur Mutlu. Improving 3D NAND flash memory lifetime by tolerating early retention loss and process variation. In *SIGMETRICS*, 2018.

[22] Macronix. Technical note: Improving NAND throughput with two-plane and cache operations, 2013. https://www.macronix.com/Lists/ApplicationNote/Attachments/1907/AN0268V1_Improving%20NAND%20Throughput%20with%20Two-Plane%20and%20Cache%20Operations.pdf.

[23] Micron. Technical note: NAND flash performance increase using the Micron PAGE READ CACHE MODE command, 2004. https://www.micron.com/-/media/client/global/Documents/Products/Technical%20Note/NAND%20Flash/tn2901.pdf.

[24] Micron. Product flyer: Micron 3D NAND flash memory, 2016. https://www.micron.com/-/media/client/global/documents/products/product-flyer/3d_nand_flyer.pdf?la=en.

[25] Youngseop Shim, Myungsuk Kim, Myoungjun Chun, Jisung Park, Yoona Kim, and Jihong Kim. Exploiting process similarity of 3D flash memory for high performance SSDs. In *MICRO*, 2019.

[26] 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 *FAST*, 2018.