



Access Characteristic Guided Read and Write Cost Regulation for Performance Improvement on Flash Memory

Qiao Li and Liang Shi, Chongqing University; Chun Jason Xue, City University of Hong Kong; Kaijie Wu, Chongqing University; Cheng Ji, City University of Hong Kong; Qingfeng Zhuge and Edwin H.-M. Sha, Chongqing University

<https://www.usenix.org/conference/fast16/technical-sessions/presentation/li-qiao>

**This paper is included in the Proceedings of the
14th USENIX Conference on
File and Storage Technologies (FAST '16).**

February 22–25, 2016 • Santa Clara, CA, USA

ISBN 978-1-931971-28-7

**Open access to the Proceedings of the
14th USENIX Conference on
File and Storage Technologies
is sponsored by USENIX**

Access Characteristic Guided Read and Write Cost Regulation for Performance Improvement on Flash Memory

Qiao Li[§], Liang Shi^{§*}, Chun Jason Xue[‡]

Kaijie Wu[§], Cheng Ji[‡], Qingfeng Zhuge[§], and Edwin H.-M. Sha[§]

[§]College of Computer Science, Chongqing University

[‡]Department of Computer Science, City University of Hong Kong

*Corresponding author: shi.liang.hk@gmail.com

Abstract

The relatively high cost of write operations has become the performance bottleneck of flash memory. Write cost refers to the time needed to program a flash page using incremental-step pulse programming (ISPP), while read cost refers to the time needed to sense and transfer a page from the storage. If a flash page is written with a higher cost by using a finer step size during the ISPP process, it can be read with a relatively low cost due to the time saved in sensing and transferring, and vice versa.

We introduce AGCR, an access characteristic guided cost regulation scheme that exploits this tradeoff to improve flash performance. Based on workload characteristics, logical pages receiving more reads will be written using a finer step size so that their read cost is reduced. Similarly, logical pages receiving more writes will be written using a coarser step size so that their write cost is reduced. Our evaluation shows that AGCR incurs negligible overhead, while improving performance by 15% on average, compared to previous approaches.

1 Introduction

NAND flash memory sees an increasing deployment in embedded systems, personal computers, mobile devices and servers over the last decades due to its advantages, such as light weight, high performance, and small form factors [1][2]. Although flash-based storage outperforms most magnetic-based storage, the costs of write and read operations are still the performance bottleneck of current systems. The write operation of flash memory is performed by the incremental-step pulse programming (ISPP) [3] scheme. ISPP is designed to iteratively increase the program voltage and use a small verification voltage to reliably program flash cells to their specified voltages. In each iteration, the program voltage is increased by a predefined program step size. Generally, write cost is much higher than read cost, usually by 10 to 20 times, and has been identified as a major performance

bottleneck [4][5]. At the same time, read cost has also increased significantly with the increasing bit density and technology scaling [6][7][8].

The following tradeoffs of read and write costs have been identified [9][10][11][12][13]. First, read cost highly depends on the maximum raw bit error rate (RBER) of data pages and the deployed error correcting code (ECC). For a specific ECC, the lower the maximum RBER, the lower the read cost [6][7][14]. Second, write cost can be reduced by increasing the program step sizes in the ISPP process, at the cost of increasing the maximum RBER of the programmed pages. This in turn reduces the maximum retention time [9][10] and increases the read cost of the programmed page [13].

Previous approaches differ in the strategies used to select the data pages that will be written with different write costs. Pan et al. [9] and Liu et al. [10] proposed to apply low-cost writes (i.e., using coarser step sizes) to the data pages through retention time relaxation. Their approach was motivated by the significantly short lifetime of most data in several workloads compared to the predefined retention time. On the other hand, a high-cost write reduces the cost of the following reads to the same page [6][14]. Based on this characteristic, Li et al. [13] proposed to apply low-cost writes when there are several queued requests, and high-cost writes otherwise. Wu et al. [12] proposed to apply high-cost writes when, according to their estimation, the next access operation will not be delayed due to the increased cost. However, none of these works exploit the access characteristics of workloads.

This work is the first to exploit the access characteristics of workloads for cost regulation. We define three page access characteristics: If almost all the accesses to a data page are read (write) requests, the page is characterized as *read-only* (*write-only*). If the accesses to a data page are interleaved with reads and writes, the page is characterized as *interleaved-access*. Our approach is based on the observation that most accesses

from the host are performed on either read-only or write-only pages. We exploit this observation to regulate access costs: for accesses identified as read-only, low-cost reads are preferred; for accesses identified as write-only, low-cost writes are preferred. The proposed approach chooses low-density parity code (LDPC) as the ECC, which is the best candidate for the advanced flash memory storage systems [6][14]. For the pages that are written with low cost, the required retention time can still be maintained as long as the resulting error rates do not exceed the error correction capability of the deployed LDPC code. Thus, read and write costs can be regulated to significantly improve overall performance. Our main contributions are as follows:

- We propose AGCR, a comprehensive approach to regulate the cost of writes and reads;
- We present a preliminary study to show the potential performance improvement of AGCR;
- We present an efficient implementation of AGCR with negligible overhead. Experimental results on a trace driven simulator [15][16] with 12 workloads [17] show that AGCR achieves significant performance improvement.

The rest of the paper is organized as follows. Section 2 presents the motivation and problem statement. Section 3 presents our proposed approach. Experiments and analysis are presented in Section 4. Section 5 concludes this work.

2 Motivation and Preliminary Study

2.1 Read and Write in Flash Memory

Write operations are the performance bottleneck of flash memory, not only because they require more time than read operations, but also because they block waiting operations. Moreover, the cost of write operations implicitly determines the cost of subsequent reads. Flash technology uses incremental-step pulse programming (ISPP) to program pages [3]. ISPP is designed to reliably program flash cells to their specific voltage levels using an iterative program-verify algorithm. The iterative algorithm has two stages in each iteration step: it first programs a flash cell with an incremental program voltage, and then verifies the voltage of the cell. If the voltage is lower than the predefined threshold, the process continues. In each iteration, the program voltage increases by a step size of ΔV_{pp} . The step size ΔV_{pp} determines the write cost – a coarser step size indicates a smaller number of steps (hence a lower write cost), but results in higher raw bit error rate (RBER) in the programmed page, and vice versa.

Low-density parity code (LDPC) is deployed as the default ECC for most advanced flash memory technologies [14]. For a high RBER, LDPC requires fine-grained

memory-cell sensing, which is realized by comparing a series of N reference voltages. The error correction capability of LDPC increases with N , but a larger N results in a higher read cost.

Thus, there is a strong relationship between ISPP and LDPC on the read and write cost of flash memory. With a large step size in the ISPP process, the write cost is reduced but the RBER and read cost increase, and vice versa. The step size also affects the retention time of flash pages. A longer retention time is expected for pages written with finer step sizes [9][10]. In this work, we focus on read and write cost interaction by tuning the step size, where the tuning is constrained by the minimum required retention time [18].

2.2 Read and Write Cost Regulator

Several strategies have been recently proposed for exploiting the reliability characteristics of flash memory to regulate read and write costs. Pan et al. [9] and Liu et al. [10] proposed to reduce write costs by relaxing the retention time requirement of the programmed pages. Wu et al. [12] proposed to apply a high-cost write to reduce the cost of read requests performed on the same page. The high-cost write is performed on the premise that upcoming requests will not be delayed by the increased cost. Li et al. [13] proposed to apply low-cost writes when there are queued requests to reduce the queueing delay, and apply high-cost writes otherwise, allowing low-cost reads of the page. However, none of these works exploit the access characteristics for cost regulation. Several types of workload access characteristics have been studied in previous works, including hot and cold accesses [19], inter-reference gaps [20], and temporal and spacial locality [21][22][23]. These access characteristics have been employed to guide the design of buffer caches [20][21][22] and flash translation layers (FTL) [19][23]. However, these characteristics imply the access frequency, which cannot be exploited for read and write cost regulation.

In this work, we applied the cost regulator proposed in Li et al. [13], which has been validated by simulation in previous studies [14][12][24]. The regulator consists of a write cost regulator and a read cost regulator. The write cost is inversely proportional to the program step size of ISPP [3][9][10]. Thus, the write cost regulator is defined as:

$$WC(\Delta V_{pp}) = \gamma \times \frac{1}{\Delta V_{pp}}$$

where $RBER(\Delta V_{pp}) < CBER_{LDPC}(N)$; (1)

$WC(\Delta V_{pp})$ denotes the write cost when the program step size is ΔV_{pp} , and γ is a variable. A coarser step size results in a lower write cost, but a higher RBER. In addition, the reduction of the write cost is limited by the

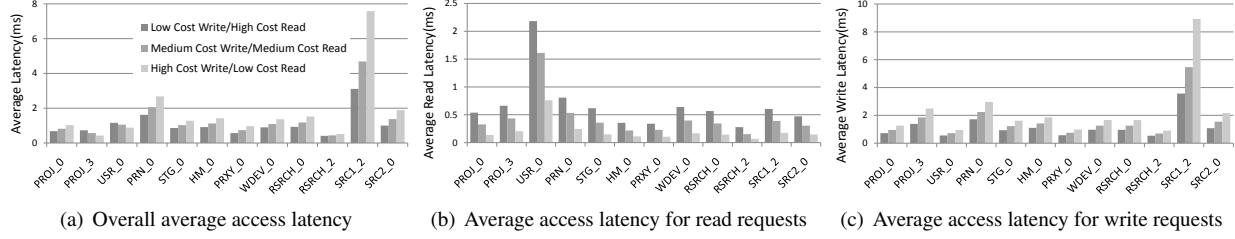


Figure 1: IO latency comparison for read and write requests with different access costs.

condition that the resulting RBER should be within the error correction capability of the deployed LDPC code, $CBER_{LDPC}(N)$, with N reference voltages.

The read cost is composed of the time to sense the page, which is proportional to N , and the time to transfer the data from the page to the controller, which is proportional to the size of the transferred information [6][7][14]. Thus, the read cost regulator is defined as:

$$RC(N) = \alpha \times N + \beta \times \lceil \log(N+1) \rceil$$

where $RBER(\Delta V_{pp}) < CBER_{LDPC}(N)$; (2)

where $RC(N)$ denotes the read cost with N reference voltages in the deployed LDPC code, and α and β are two variables. With a larger N , longer on-chip memory sensing time and longer flash-to-controller data transfer time are required, which lead to increased read cost.

Based on the two parts of the regulator, a low-cost write leaves the page with a higher RBER, thereby requiring an LDPC with more reference voltages, which further results in higher-cost reads. On the other hand, a high-cost write leaves the page with a lower RBER, thereby requiring an LDPC with less reference voltages, which further results in lower-cost reads. Note that the regulator ensures that the retention time requirement is always satisfied when writing data using different costs.

2.3 Preliminary Study

In the preliminary study, we evaluate three combinations of read and write costs: 1) All the writes are performed with a high cost, followed by low-cost reads (HCW/LCR); 2) All the writes are performed with a medium cost, followed by medium-cost reads (MCW/MCR); 3) All the writes are performed with a low cost, followed by high-cost reads (LCW/HCR). The detailed settings for the regulator can be found in Section 4.1.

Figure 1 presents the comparison of access latency for 12 representative traces of the enterprise servers from Microsoft Research (MSR) Cambridge [17]. Compared to the default MCW/MCR, HCW/LCR improves read performance by 54%, and LCW/HCR improves write performance by 26%. The significant performance improvement comes from the corresponding reduction in access costs. Comparing LCW/HCR and HCW/LCR, the

differences in their read and write latencies are 114% and 61%, respectively. The performance gap indicates that the read and write cost regulator should be applied carefully. While LCW/HCR is able to improve the overall performance, it introduces the worst read performance, as shown Figure 1(b). However, read operations are always in the critical path, which motivates our approach.

3 AGCR: Access Characteristic Guided Cost Regulation

In this section, we propose AGCR, an access characteristic guided cost regulation scheme. We first present the study on the access characteristics of several workloads. Then, based on the observations from this study, we propose a comprehensive approach that includes a high-accuracy access characteristic identification method and a guided cost regulation scheme. Finally, we present the implementation and the overhead analysis.

3.1 Access Characteristics of Workloads

In order to guide the read and write cost regulation, the access characteristics of workloads are very important. For example, if the accesses to a data page (logical page at the host system) are dominated by read requests, accessing the data page with low-cost reads can significantly improve performance.

Figure 2 presents the statistical results for the workloads analyzed in our preliminary study. We collected access characteristics for all data pages at the host system and distinguished between three types of data accesses:

1. Read-only: If almost all the accesses (>95%) to a data page are read requests, we characterize this page as read-only. This is typical when accessing media files or other read-only files;
2. Write-only: If almost all the accesses (>95%) to a data page are write requests, we characterize this page as write-only. This is typical in periodical data flushes from the memory system to storage for consistency maintenance;
3. Interleaved-access: If the accesses to a data page are interleaved with reads and writes, we characterize this page as interleaved-access.

Figure 2 shows the request distributions for these logical data page access characteristics. Figure 2(a) shows the

Read cost regulation. A low-cost read can only be issued if the page was written with a high-cost write. Thus, a high-cost write will be *inserted* to the list of reads to re-write the page before upcoming reads, if the page was not written with a high cost before. As shown in Figure 3, there are three cases of read cost regulation. Upon arrival of a read request, the page is characterized, and its cost is regulated as follows:

- For a *read-only* page with a low-cost read, nothing should be done;
- For an *interleaved-access* page, the read cost is not regulated;
- For a *read-only* page with a high-cost read, a high-cost re-write operation is inserted to the re-write queue and performed during idle time to reduce the cost of upcoming reads (denoted with **H** in Figure 3(c)).

3.4 Implementation and Overheads

Figure 4 shows the implementation of AGCR in the flash memory controller. We add three new components: Access Characteristic Identification, Cost Regulator and Re-Write Queue. In addition, each mapping entry in the FTL is extended with two fields, as shown in Figure 5. The first is the access history, and the second is a 1-bit low-cost write tag. When the low-cost write tag is set for a read-only page, this indicates that the page must be re-written. When an I/O request is issued, the page is characterized, and its cost is regulated according to the rules in Section 3.3. If a re-write operation is needed for read cost regulation, the logical address of the data page is added to the re-write queue. During idle time, re-write operations are triggered to program the data in the queue with a high cost, which guarantees low-cost reads on these read-only pages. The re-write overhead is evaluated in the experiments. This implementation

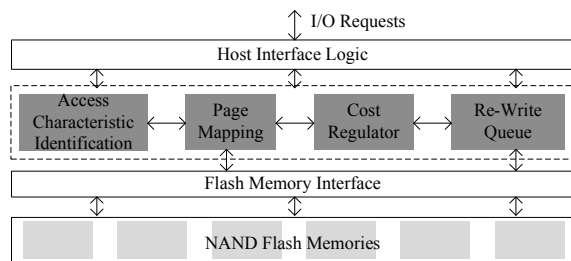


Figure 4: The implementation of the proposed approach.

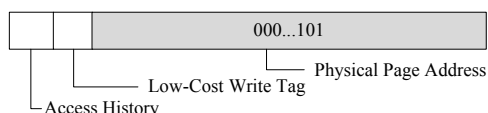


Figure 5: An example on the FTL mapping entry.

incurs three types of overheads: storage, hardware and firmware overhead. The storage overhead includes, for

each page, a number of bits for access history and one bit for the low-cost write tag. Assuming access history is set to one bit, which records only the most recent request to the page, the storage overhead is 4MB for a 64GB flash memory with 4KB pages. The hardware overhead includes the voltage thresholds needed to support the three sets of reads and writes with different costs. This overhead is negligible according to previous studies [10][25]. The firmware overhead includes the processes involved in access characterization, cost regulation and re-write queuing. The overhead of these simple processes is negligible. AGCR does not introduce reliability issues thanks to the constraints in the read and write cost regulators. In addition, the energy consumption of the additional components is negligible.

4 Evaluation

4.1 Experimental Setup

Table 1: The access cost configurations.

Read Cost (μs)	Low	Medium	High
	70	170	310
Write Cost (μs)	High	Medium	Low
	800	600	450
Erase Cost (μs)	3000	3000	3000

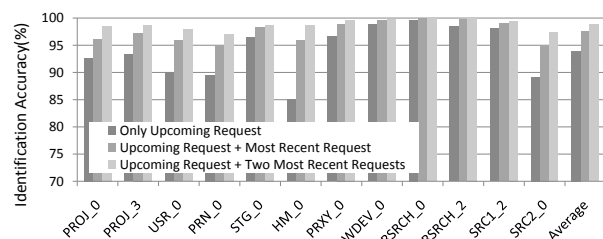


Figure 6: Window size impact on identification accuracy.

We use SSDsim [15] and workloads from MSR [17] to evaluate our cost-regulation approach. From our studies, we found that all workloads in MSR have similar access characteristics. We select the 12 representative traces from Figure 2 for our evaluation. The simulated storage system is configured with two bits per cell MLC flash memory. It has eight channels, with eight chips per channel and four planes per chip. Each chip has 2048 blocks and each block has 64 4KB pages. Default page mapping based FTL, garbage collection, and wear leveling are implemented in the simulator, representing

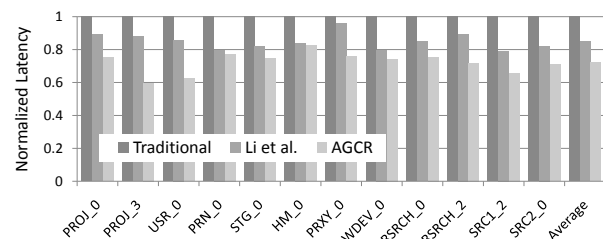


Figure 7: Overall performance comparison.

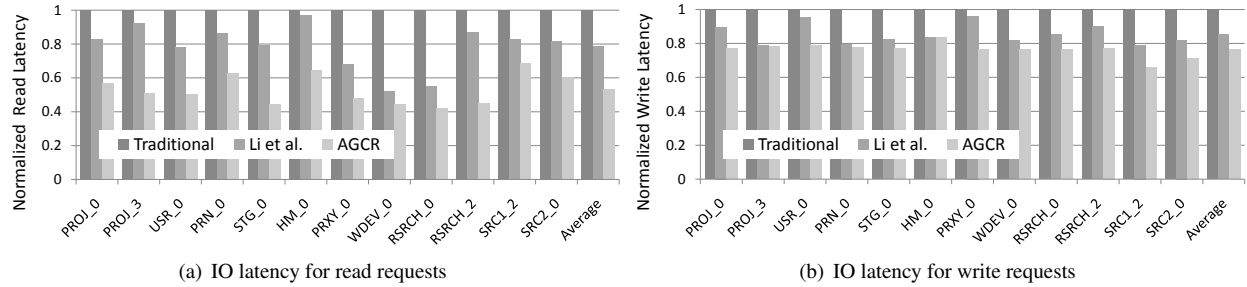


Figure 8: Normalized access latency compared with traditional case and Li et al.[13].

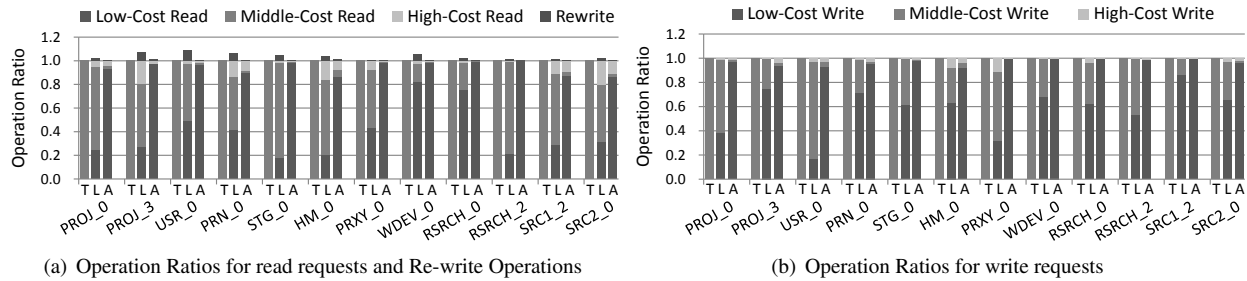


Figure 9: Distributions of different cost operations: T-Traditional, L-Li et al., and A-AGCR.

state-of-the-art storage systems [16]. The internal read and write operations for these mechanisms are processed with the regulated access costs. For fair comparison, we use the parameters for the cost regulator from Li et al. [13]. Table 1 shows the access costs. The cost of the erase operation does not depend on the write cost.

One of the most important parameters is the size of history window in the identification method. It affects the storage overhead as well as the identification accuracy. Figure 6 shows the identification accuracy for three window sizes. The results show that a larger window results in a higher accuracy. However, the increase in accuracy decreases as the size increases. In the following experiments, we consider only the most recent request and the upcoming request for identification to trade-off the storage overhead and identification accuracy.

4.2 Experimental Results

In this section, read, write and overall performance of AGCR are evaluated and compared with the traditional case and the state-of-the-art work from Li et al. [13]. In the traditional case, all reads and writes are performed with medium-cost, while Li et al. used low-cost reads or writes to reduce queueing delay. Cost regulation highly depends on the access history in AGCR. If there is no history for the accessed page, a high-cost write is used for write requests and a low-cost read is used for read requests. Figures 7 and 8 show the normalized access latency compared to the traditional, non-regulated costs (MCW/MCR in Figure 1) and to Li et al. Figure 7 shows that, compared to Li et al., AGCR achieves the best overall performance improvement, 15%, on average. In addition, as shown in Figures 8(a) and 8(b), AGCR achieves

up to 48.3% and 20.4% latency improvement for reads and writes, respectively, over Li et al.

To understand the performance variations, Figure 9 shows the distributions of operations of different costs, including fast, medium, and slow reads and writes, as well as re-writes. Comparing to Li et al.'s work, AGCR issues considerably more low-cost reads and writes, thus achieving great improvement. In addition, the percentage of re-write operations is negligible, no more than 1% of all accesses issued by the host, thanks to the small portion of interleaved-access page requests.

5 Conclusions and Future Work

In this paper, we introduced AGCR, a cost regulation approach for flash memory to improve access performance. The proposed approach is motivated by observations derived from widely studied workloads. Based on these observations, we propose an accurate access characterization method, and regulate access costs to improve performance. Our simulation results show that AGCR is effective, reducing I/O latency by as much as 48.3% and 20.4% for read and write requests respectively, compared to the state-of-the-art approach.

Acknowledgments

We thank our shepherd Gala Yadgar and the anonymous reviewers for their comments and suggestions. This work is supported by the Fundamental Research Funds for the Central Universities (CDJZR185502), NSFC 61402059, National 863 Programs 2015AA015304 and 2013AA013202, NSFC 61472052, and Chongqing Research Program cstc2014yykfb40007.

References

- [1] C. Lee, D. Sim, J. Hwang, and S. Cho, “F2FS: A new file system for flash storage,” in *Proceedings of the USENIX Conference on File and Storage Technologies*, 2015, pp. 273–286.
- [2] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. S. Manasse, and R. Panigrahy, “Design tradeoffs for SSD performance,” in *Proceedings of USENIX Annual Technical Conference*, 2008, pp. 226–229.
- [3] K.-D. Suh, B.-H. Suh, Y.-H. Lim, J.-K. Kim, Y.-J. Choi, Y.-N. Koh, S.-S. Lee, S.-C. Kwon, B.-S. Choi, J.-S. Yum, J.-H. Choi, J.-R. Kim, and H.-K. Lim, “A 3.3 V 32 Mb NAND flash memory with incremental step pulse programming scheme,” *IEEE Journal of Solid-State Circuits*, vol. 30, no. 11, pp. 1149–1156, 1995.
- [4] D. Ajwani, I. Malinger, U. Meyer, and S. Toledo, “Characterizing the performance of flash memory storage devices and its impact on algorithm design,” in *Proceedings of the International Conference on Experimental Algorithms*, 2008, pp. 208–219.
- [5] G. Yadgar, E. Yaakobi, and A. Schuster, “Write once, get 50% free: Saving SSD erase costs using WOM codes,” in *Proceedings of the USENIX Conference on File and Storage Technologies*, 2015, pp. 257–271.
- [6] G. Dong, N. Xie, and T. Zhang, “Enabling NAND flash memory use soft-decision error correction codes at minimal read latency overhead,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 9, pp. 2412–2421, 2013.
- [7] K.-C. Ho, P.-C. Fang, H.-P. Li, C.-Y. Wang, and H.-C. Chang, “A 45nm 6b/cell charge-trapping flash memory using LDPC-based ECC and drift-immune soft-sensing engine,” in *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, 2013, pp. 222–223.
- [8] X. Jimenez, D. Novo, and P. Ienne, “Phoenix: reviving MLC blocks as SLC to extend NAND flash devices lifetime,” in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2013, pp. 226–229.
- [9] Y. Pan, G. Dong, Q. Wu, and T. Zhang, “Quasi-nonvolatile SSD: Trading flash memory nonvolatility to improve storage system performance for enterprise applications,” in *Proceedings of the High Performance Computer Architecture*, 2012, pp. 1–10.
- [10] R.-S. Liu, C.-L. Yang, and W. Wu, “Optimizing NAND flash-based SSDs via retention relaxation,” in *Proceedings of the USENIX conference on File and Storage Technologies*, 2012, pp. 1–11.
- [11] Y. Luo, Y. Cai, S. Ghose, J. Choi, and O. Mutlu, “WARM: Improving NAND flash memory lifetime with write-hotness aware retention management,” in *Proceedings of Mass Storage Systems and*, 2015, pp. 1–14.
- [12] G. Wu, X. He, N. Xie, and T. Zhang, “Exploiting workload dynamics to improve ssd read latency via differentiated error correction codes,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 18, no. 4, pp. 55:1–55:22, 2013.
- [13] Q. Li, L. Shi, C. Gao, K. Wu, C. J. Xue, Q. Zhuge, and E. H.-M. Sha, “Maximizing IO performance via conflict reduction for flash memory storage systems,” in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, 2015, pp. 904–907.
- [14] K. Zhao, W. Zhao, H. Sun, T. Zhang, X. Zhang, and N. Zheng, “LDPC-in-SSD: Making advanced error correction codes work effectively in solid state drives,” in *Proceedings of the USENIX Conference on File and Storage Technologies*, 2013, pp. 244–256.
- [15] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and C. Ren, “Exploring and exploiting the multilevel parallelism inside SSDs for improved performance and endurance,” *IEEE Transactions on Computers*, vol. 62, no. 6, pp. 1141–1155, 2013.
- [16] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and S. Zhang, “Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity,” in *Proceedings of the international conference on Supercomputing*, 2011, pp. 96–107.
- [17] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron, “Migrating server storage to SSDs: analysis of tradeoffs,” in *Proceedings of ACM European conference on Computer systems*, 2009, pp. 145–158.
- [18] H. Park, J. Kim, J. Choi, D. Lee, and S. H. Noh, “Incremental redundancy to reduce data retention errors in flash-based SSDs,” in *Proceedings of Mass Storage Systems and Technologies*, 2015, pp. 1–13.

- [19] D. Park and D. H. Du, “Hot data identification for flash-based storage systems using multiple bloom filters,” in *IEEE Symposium on Mass Storage Systems and Technologies*, 2011, pp. 1–11.
- [20] G. Wu, B. Eckart, and X. He, “BPAC: An adaptive write buffer management scheme for flash-based Solid State Drives,” in *IEEE Symposium on Mass Storage Systems and Technologies*, 2010, pp. 1–6.
- [21] L. Shi, J. Li, C. J. Xue, C. Yang, and X. Zhou, “ExLRU: a unified write buffer cache management for flash memory,” in *Proceedings of ACM international conference on Embedded software*, 2011, pp. 339–348.
- [22] H. Kim and S. Ahn, “BPLRU: A Buffer Management Scheme for Improving Random Writes in Flash Storage,” in *Proceedings of the USENIX Conference on File and Storage Technologies*, 2008, pp. 1–14.
- [23] S. Lee, D. Shin, Y.-J. Kim, and J. Kim, “LAST: locality-aware sector translation for NAND flash memory-based storage systems,” *ACM SIGOPS Operating Systems Review*, vol. 42, no. 6, pp. 36–42, 2008.
- [24] K. Zhao, K. S. Venkataraman, X. Zhang, J. Li, N. Zheng, and T. Zhang, “Over-clocked SSD: Safely running beyond flash memory chip I/O clock specs,” in *Proceedings of High Performance Computer Architecture*, 2014, pp. 536–545.
- [25] Y. Pan, G. Dong, and T. Zhang, “Exploiting memory device wear-out dynamics to improve NAND flash memory system performance,” in *Proceedings of the USENIX conference on File and Storage Technologies*, 2011, pp. 1–14.