Improving MLC Flash Performance with Workload-Aware Differentiated ECC

Qianbin Xia, Weijun Xiao, Dongwei Wang, Liang Xu
Department of Electrical and Computer Engineering
Virginia Commonwealth University
Email: {xiaq2, wxiao, wangd7, xul4}@vcu.edu

***Abstract*—The adoption of small~~er~~ geometries and multi-level cell (MLC) technologies significantly expands the capacity and drops the price of flash memory, which, at the same time, noticeably degrades the performance and reliability of the devices. As incremental-step pulse programming (ISPP) scheme is used to increase the program accuracy for MLC cells, there is a trade-off between the SSD write ~~speed~~ and raw storage reliability. What's more, ECC ~~are~~ widely used in SSDs to provide error-tolerance ability. Therefore, if we could use stronger ECC to increase error correction strength, a low-cost write with coarser step sizes could be applied in the ISPP scheme to promote the write performance. However, stronger ECC scheme may hurt the read performance due to the increased decoding complexity and latency.**

In this paper, we propose a workload-aware differentiated ECC scheme to improve the SSD write performance without sacrificing the read performance. The main idea is to dynamically classify the logical pages into three categories: write-only, read-only, and overlapped part. For write-only logical pages, low-cost write with strong ECC scheme will be applied increase the write ~~speed.~~ For write logical pages in the overlapped part, the low-cost writes with strong ECC will be selectively used based on their relative write and read hotness. While for any read logical pages encoded with a stronger ECC, we will rewrite them with the normal-cost write and ECC scheme if their hotness exceed a pre-defined threshold. The evaluation results show that our workload-aware differentiated ECC scheme could ~~read~~ the write and read response time by 43.4% and 8.4% on average, respectively.

***Index Terms*—ISPP; BCH; LDPC;**

I. Introduction

NAND Flash-based Solid State Disk (SSD) has been widely deployed in various environments because of its high performance, nonvolatile property, and low power consumption. To continuously increase the capacity and reduce the bit cost, manufactures are aggressively scaling down the geometries and storing more bits information per flash cell [13]. However, the adoption of these technologies has inevitably resulted in degraded SSD performance especially the write performance. For example, from the previous SLC SSDs to current 2-bit MLC SSDs, the page write latency has increased from 200 us [15] to 1800 us [37], which has been identified as the major performance bottleneck [4].

A flash memory uses floating gate to store electrons and the amount of electrons will affect the cell's threshold voltage V_{th} [6]. Different V_{th} values could represent different data. These operations performed to inject and remove electrons from the floating gate are called program and erase, respectively [16]. Currently, the incremental-step pulse programming (ISPP) scheme [35] is used to perform the flash write operations. ISPP consists of a series of programming-and-verifying steps. In each step, a programming voltage $V_{program}$ is firstly applied to raise a cell's threshold voltage to $V_{current}$, then during the verification stage, the $V_{current}$ will be compared with the expected threshold voltage V_{expect}. If the $V_{current}$ is larger than the V_{expect}, then the program operation ~~finish~~. Otherwise, the $V_{program}$ will be increased by a step voltage ΔV_{pp} and the programming-and-verifying steps will be repeated. Therefore, increasing the step voltage ΔV_{pp} could reduce the programming steps and time to reach the expected threshold voltage and improve the flash write performance. However, a larger ΔV_{pp} will result in worse raw bit error rate (RBER) due to the wider threshold voltage distribution of each programmed state and less noise margin between adjacent programmed states. To compensate the increased error rate due to a larger ΔV_{pp}, a stronger ECC scheme could be applied to provide stronger error correction capability. Currently, BCH code [7] is being widely used in NAND flash memories [19], [14], [36]. Beside BCH, the low-density parity-check (LDPC) codes [17], [30] have attracted lot of attention in ~~the academical community~~ [43], [21], [44], [31], [38], [45] and are the promising substitutes of the BCH code for the future SSDs. Compared with BCH code, LDPC could provide superior error correction capability. However, the complicated and time-consuming decoding process of LDPC will inevitably worsen the flash read performance and limit its rapid adoption in the real SSD product. It is has been showed in [45] that the LDPC code can increase the read latency by almost 120%, while the effects on the flash write performance is within 2%.

In this work, we propose a workload-aware differentiated ECC design to improve the write performance of 2-bit MLC flash devices. BCH code is used as our baseline ECC scheme, while LDPC as the stronger ECC to accelerate the write performance. First, we dynamically separate the read and write logical pages. For write-only pages, LDPC will be applied to improve flash write performance by increasing the step voltage ΔV_p. For write pages in the overlapped section, LDPC with low-cost write scheme will be selectively applied based on the relative write and read hotness of the logical pages. Since LDPC will dramatically degrade the read performance, a rewrite operation with BCH and normal-cost write scheme will be performed on the logical pages whose read hotness exceed a pre-defined threshold. Our main contributions include:

- We propose a workload-aware differentiated ECC scheme to improve Flash write ~~speed~~ without compromising Flash read performance.
- We design and implement a lightweight read and write separator by modifying the basic multi-bloom filters, which could dynamically separate the read and write logical pages in both the spatial and ~~timing~~ space.
- we present an efficient implementation of our design. Based on the simulations of ~~real world~~ disk traces, we demonstrate that our proposed design could reduce the Flash write latency by up to 55% without sacrificing the read performance.

The rest of th~~is~~ paper is organized as follows. In ~~s~~ection II, we describe the background and related work on flash memory. Section III presents workloads analysis. Section IV depicts the details of our proposed ~~operation~~-aware differentiated ECC scheme. Section V presents the evaluation methodology and the experimental results. Section VI presents concluding remarks. ~~Section VII is the acknowledgement.~~

II. Background and Related Work

A. NAND Flash-Based SSD

In the commercial market, there are three widely used types of flash cells: SLC (single-level cell, storing a single bit per memory cell), MLC (multilevel cell, usually referred to two bits per cell), and TLC (Tri-level cell, storing three bits per cell). Storing multiple bits information in a cell could significantly improve the storage density and capacity with the overhead of sacrificed reliability, lifetime, and performance. In this paper, we only consider the two bits per cell MLC flash memory.

Unlike traditional storage devices, Flash memory can not support in-place update and has the lifetime limitation. In order to make flash memory compatible with the existing file systems designed for in-place update devices, an Flash Translation Layer (FTL) is deployed in SSDs. An typical FTL consists of an address translator, garbage collector, and wear-lever. The address translator translates the logical page number to the physical page number and maintains a mapping table that associates the logical page number with physical page number. There are three main mapping scheme: page-level mapping [12] that can achieve the best performance with the highest memory overhead, block-level mapping [5] that can save huge amount of memory with sacrificed performance, and hybrid mapping [20], [24], [26], [25], [46] that makes a compromise between the page-level mapping and block-level mapping. A garbage collector reclaims the obsolete pages due to out-of-place updates. One most popular garbage collection scheme is the greedy algorithm [8], which always selects the blocks with the fewest number of valid pages as the victim blocks. The objective of a wear-leveler [11] is to evenly distribute the writes among all the flash blocks to improve the overall lifetime of flash memory. For simplicity, we assume that the page-level mapping scheme, greedy garbage collection policy, and none wear-leveler function unit are used in this paper.

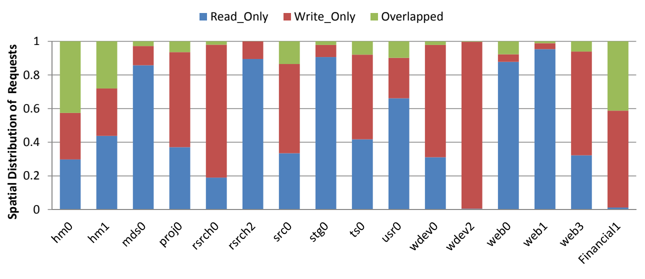


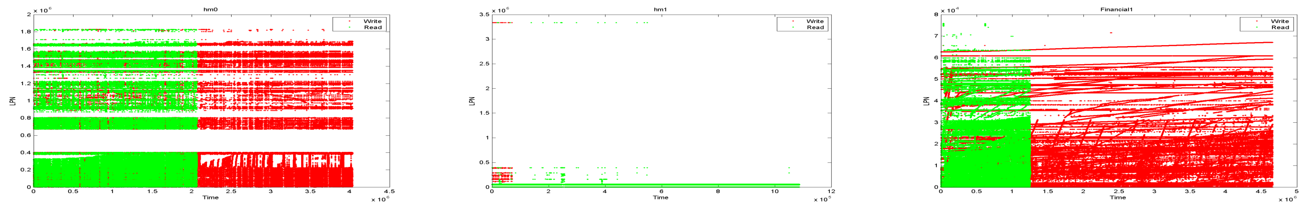
Fig. 1. Spatial distribution of the read and write requests.

B. Related Work

With the adoption of small~~er~~ geometrics and MLC technologies, the degraded performance of flash-based SSDs is becoming a critical issue. Many schemes have been proposed to enhance the performance. Some of the previous research work is based on the trade-off between the RBER and program ~~speed~~. Since the RBER increases with the retention time, Pan et al. [32] and Liu et al. [29] propose to improve the program speed by compromising the retention time requirement. Besides the retention time, RBER also depends on the program/erase cycles and the content hold by the flash cells. It is well known that NAND flash memory gradually wear out with the increasing of program/erase cycles, and which has been exploited in [33] to improve the flash program speed. As the RBER is also depend on the content of the flash cell, Gao et al. [18] propose to apply the fast program scheme to the flash pages with low-content-aware RBER. DiffECC is proposed in [39] to partially use the high cost write scheme during the idle time to reduce the RBER, which then could be handled by weak ECC scheme, to improve the flash read performance. AGCR [27] proposed by Li et al. tries to separate the accesses into three categories: read-only, write-only, and interleaved accesses. For read-only accesses, high-cost writes will be applied to reduce the overhead of the following read requests. While for write-only accesses, low-cost writes will be used to improve the write performance. For the interleaved accesses, the medium-cost writes will be applied to reach a comprise. Among all the above strategies, only AGCR is similar to our scheme. However, AGCR uses a uniform strong ECC scheme, which is designed for the worst cases, for all the flash pages that could have different error rates due to different write scheme applied on the pages. Therefore, AGCR is far from optimal. Besides, to separate the three different kinds of accesses, AGCR need to record the access histories for all logical pages, which will consume too much memory resources, especially with the continuous increasing capacity of SSDs.

III. Workload Analysis

In this section, we present the study on the access characteristics of ~~16~~ representative workloads that are mixed with both read and write requests from the Microsoft Research (MSR) Cambridge [1] and OLTP applications [2]. Figure 1



(a) hm0 (b) hm1 (c) Financial1

Fig. 2. Read and write request distributions in the two-dimensional space (logical address and tim~~ing~~ space).

shows the ratios of write-only, read-only, and overlapped part based on their logical page numbers (here, we assume the flash page size is 8KB). The results clearly indicate that the read and write requests could be well separated based on their logical addresses with the overlapped ratios below 10% or even 5% for most of the workloads. However, hm0, hm1, and Financial1 are three exceptions whose overlapped ratios are around 42%, 28%, and 41%, respectively. Although these three workloads have a very high overlapped ratios, the read and write requests could be separated in the time ~~space~~. Figure 2 shows the read and write request distributions in a two-dimensional space where x-axis and y-axis are the timing space and logical address space, respectively. In Figure 2 the red dots are the write requests, while the green dots are the read requests. From the results, we find that, for workloads with high overlapped ratio in the logical address space, the overlapping of read and write logical pages happens merely within a limited timing period, while in other timing period, they are very well or even totally separated. From the above all results, we make the following two observations ~~in summary:~~

- Read and write requests are well separated in the logical address space for most workloads with overlapped ratio below 10% or even 5%;
- When read and write requests are highly overlapped in the logical address space, they could be easily separated in the tim~~ing~~ space.

These two observations give us the hints to properly design the read and write separation scheme and the whole architecture of our system.

IV. Design and Implementation

In this section, we will first describe our read and write separator. Then the whole architecture and working flow of our proposed design will be presented.

A. *Read and Write Separator*

Multiple bloom filters has been utilized to perform hot data identification for flash-based storage system in [34] that captures both the frequency and recency of the requested data. A multiple bloom filters system consists of V independent bloom filters and K independent hash functions. Each bloom filter (BF) is a M bits array whose initial values are 0. K independent hash functions map the input elements to the corresponding K bit positions of the M bits BFs. To identify the hot data, the logical addresses of the write requests are used as the input for the K hash functions and will be mapped to K corresponding bit positions of the BFs. Each time, one BF will be selected from the V BFs in a round-robin manner, then the values of these K bit positions of the selected BF will be set to 1. To capture both the frequency and recency of the requests, V different weights will be assigned to the V BFs, where the highest and lowest weights are assigned to the latest and oldest updated BFs, respectively. What's more, a decay process is triggered to clear the oldest BF periodically.

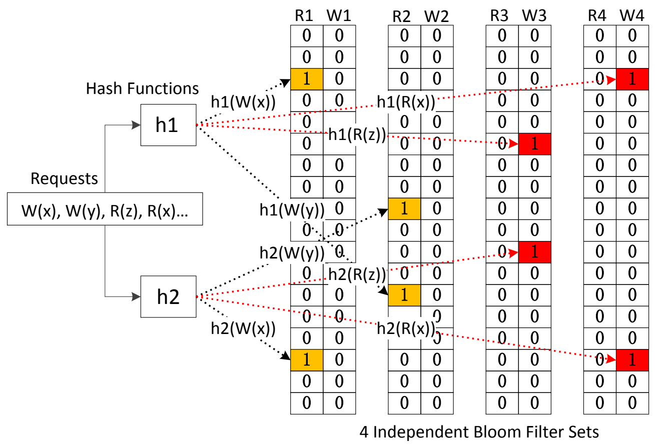


Fig. 3. Read and write separator based on multiple bloom filters.

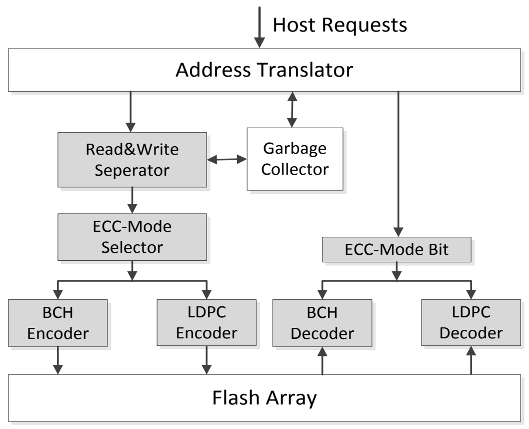


Fig. 4. Architecture of our workload-aware differentiated ECC design.

Unlike the original multiple bloom filters that considers only write requests, our read and write separator replaces all the independent bloom filters with read and write bloom filter pairs to accommodate both read and write requests. The read and write bloom filters within bloom filter pairs will only be updated by read requests and write requests, respectively. Figure 3 presents an example to show the architecture of our read and write separator and how it works. The read and write separator consists of two hash functions and four independent bloom filter pairs. Initially, all the bloom filter pairs are filled with 0. Then, a write request on logical page x comes. Logical page x will be mapped to two bit positions in the bloom filters and the write bloom filter in the first bloom filter pair will be updated. The second request is a write request on logical page y. Similarly, the write bloom filter of the second bloom filter pair will be updated. The following request is a read request on logical page z, the read bloom filter of the third bloom filter pair will be updated. The forth is also a read request and the read bloom filter of the forth bloom filter pair is updated. In this way, the read hotness of a logical address could be calculated based on the read bloom filters, while the write hotness is based on the write bloom filters. By comparing the read hotness with the write hotness, the read requests could be separated with write requests as follows:

- Read only: read_hotness $>$ 0 and write_hotness = 0
- Write only: read_hotness = 0 and write_hotness $>$ 0
- Overlapped: read_hotness $>$ 0 and write_hotness $>$ 0

According to the experiment results in [34], we configure our read and write separator with 2 hash functions, and 4 read and write bloom filter pairs with 2048 bits per bloom filter. The decay period is set as 512 requests. The recency weights are set as 2, 1.5, 1, or 0.5 based on the recency of the bloom filter pairs.

B. Architecture and Working Flow

Figure 4 gives the overview of our workload-aware differentiated ECC design. In addition to the typical components of SSDs like address translator and garbage collector, a read and write separator, ECC model selector, ECC mode bit, and a hybrid ECC system that supports both BCH and LDPC are added in our design. A 1-bit ECC mode tag is attached to each page as metadata to help the system choose the corresponding decoder for a read request.

Whenever a host write request arrives, the write bloom filter will be updated. For write requests from both the host side and background operations like garbage collection, the ECC-Mode selector chooses the proper ECC encoder and write scheme based on the following regulations:

- For a write-only page, a low-cost write scheme with LDPC encoder will be applied;
- For an overlapped page, if the write_hotness - read_hotness $>$ m, the low-cost write scheme with LDPC encoder will be applied. Otherwise, the normal write scheme with BCH encoder will be applied;

While for a host read request, the read bloom filter will be updated. Then the corresponding decoder will be used to decode the data according to the ECC mode bit. Besides, if the read hotness of a logical page exceeds a predefined threshold n and the page is encoded with LDPC, then a rewrite operation with normal-cost write and BCH encoder will be issued to reduce the overhead of the upcoming read requests. The selections of the values for m and n will affect the performance gain of our design. A larger value of m will limit the possibility to applied our low-cost write mode. While, a small value of m may hurt the read performance. For threshold n, a larger value may hurt the read performance and a small value may introduce unnecessary rewrite operations. Through exhaustively exploring the design space, the best overall performance can be achieved with m equals 1 and n equals 4, which will be used as the default configurations in our experiments.

V. Experimental Methodology and Results

A. Experimental Methodology

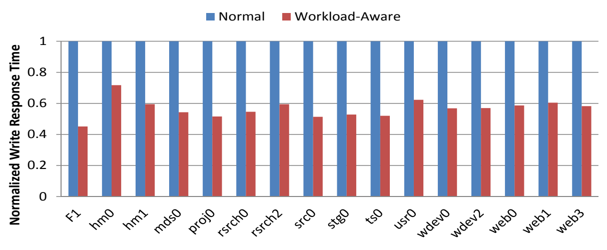
We modified the Disksim with SSD extension [3] to verify our proposed design. We assume only two write schemes are supported by the device, the normal-cost write mode and low-cost write mode that doubles the ΔV_p. Therefore, we set the page normal-cost write latency as 1.3 ms, low-cost write latency as 650 us, block erase latency as 3.8 ms, BCH-encoded page read latency as 75 us, and LDPC-encoded page read latency as 140 us (estimated based on [45]). The sizes of flash page and block are 8 KB and 1 MB, respectively. There are 4 channels with 2 packages per channel. The capacity of the packages are determined by the address space of the specific workloads. Sixteen realistic workloads that has been analyzed in the previous section will be used ~~are the input of~~ our experiments. Details of the characteristics of these workloads are depicted in Table I.

TABLE I
Characteristics of I/O workload traces

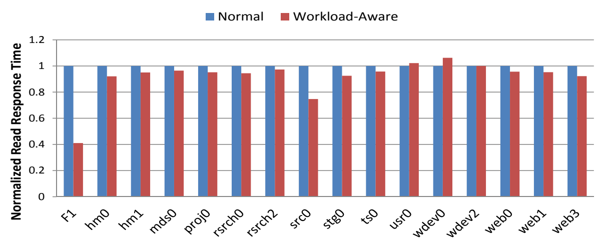
Workload Name	Addr. Space	Request Amount	Read Ratio	Avg. Req. Size (KB)
Financial1	16.67GB	5334857	23.16%	15.15
hm0	13.94GB	3993316	35.5%	7.99
hm1	25.44GB	609311	95.34%	15.16
mds0	33.92GB	1211034	11.89%	9.20
proj0	16.24GB	4224524	12.48%	38.04
rsrch0	16.89GB	1433655	9.32%	8.93
rsrch2	81.65GB	207587	65.69%	4.09
src0	15.63GB	1557814	11.34%	7.21
stg0	10.82GB	2030915	15.19%	11.58
ts0	21.97GB	1801734	17.58%	9.01
usr0	15.9GB	2237889	40.42%	22.67
wdev0	16.96GB	1143261	20.08%	9.08
wdev2	33.92GB	181266	0.1%	8.15
web0	33.91GB	2029945	29.88%	14.99
web1	67.83GB	160891	54.11%	29.07
web3	169.58GB	31380	32.03%	38.14

VI. Experimental Results

Figure 5 shows the normalized read and write response time. With the assistance of our workload-aware differentiated

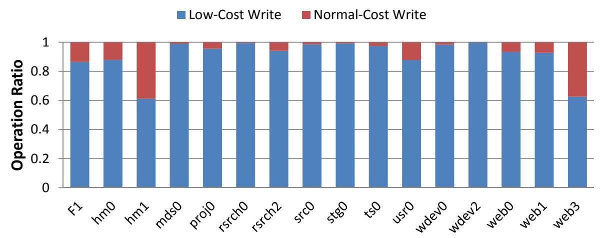


(a) Normalized write response time

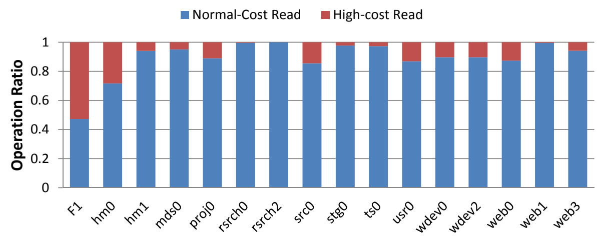


(b) Normalized read response time

Fig. 5. Normalized write and read response time.



(a) Different write ratios



(b) Different read ratios

Fig. 6. Different read and write ratios.

ECC design, the write response time could be reduced by up to 55%. While for the read requests, the read response time for most of the workloads are ~~improved~~ and the average ~~improvement~~ is around 8.4%, which mainly benefits from the reduction of the queueing delay. Two exceptions are the usr0 and wdev0, which degrade the read performance by about 2% and 6%, respectively. What's more, the low-cost write, normal-cost write, normal-cost read, and high-cost read ratios are given out in Figure 6 to help us understand the results. The write performance of our workload-aware design are mainly determined by three factors: the write request ratio, low-cost write ratio, and the queueing delay that depends on the request density. Therefore, workloads with low read and overlapped ratios like mds0, proj0, src0 and so on can significantly reduce the write response time. While, the read performance are determined by the relative effect between extra overhead due to high-cost read and the reduction of the queueing delay. Although Financial1 has a high logical address overlapping and high high-cost read ratio, both the read and write performance are improved significantly, which is from the noticeable reduction of the queueing delay that is about 54.6%.

VII. Conclusion

In this paper, we proposed a workload-aware differentiated ECC scheme. A light-weight read and write separator based on the multiple bloom filters are developed to dynamically distinguish the write-dominant requests from the read-dominant requests. Then a low-cost write scheme with stronger ECC is applied to the write-dominant requests to improve the write performance. Our simulation results demonstrate that our workload-aware differentiated ECC design can improve the write and read performance by 43.4% and 8.4% on average, respectively.

VIII. Acknowledgements

This research is sponsored in part by U.S. National Science Foundation grants CCF-1547804 and CSR-1526190. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] SNIA-Block I/O Traces. http://iotta.snia.org/tracetypes/3.
[2] UMass Trace Repository. http://traces.cs.umass.edu.
[3] Agrawal, N., Prabhakaran, V., Wobber, T., Davis, J. D., Manasse, M. S., and Panigrahy, R. Design tradeoffs for ssd performance. In *USENIX Annual Technical Conference* (2008), pp. 57–70.
[4] Ajwani, D., Malinger, I., Meyer, U., and Toledo, S. *Characterizing the performance of flash memory storage devices and its impact on algorithm design*. Springer, 2008.
[5] Ban, A. Flash file system, Apr. 4 1995. US Patent 5,404,485.
[6] Bez, R., Camerlenghi, E., Modelli, A., and Visconti, A. Introduction to flash memory. *Proceedings of the IEEE 91*, 4 (2003), 489–502.
[7] Bose, R. C., and Ray-Chaudhuri, D. K. On a class of error correcting binary group codes. *Information and control 3*, 1 (1960), 68–79.
[8] Bux, W., and Iliadis, I. Performance of greedy garbage collection in flash-based solid-state drives. *Performance Evaluation 67*, 11 (2010), 1172–1186.
[9] Cai, Y., Luo, Y., Haratsch, E. F., Mai, K., and Mutlu, O. Data retention in mlc nand flash memory: Characterization, optimization, and recovery. In *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on* (2015), IEEE, pp. 551–563.
[10] Cai, Y., Mutlu, O., Haratsch, E. F., and Mai, K. Program interference in mlc nand flash memory: Characterization, modeling, and mitigation. In *Computer Design (ICCD), 2013 IEEE 31st International Conference on* (2013), IEEE, pp. 123–130.

[11] Chang, Y.-H., Hsieh, J.-W., and Kuo, T.-W. Endurance enhancement of flash-memory storage systems: an efficient static wear leveling design. In *Proceedings of the 44th annual Design Automation Conference* (2007), ACM, pp. 212–217.

[12] Chiang, M.-L., Lee, P. C., Chang, R.-C., et al. Using data clustering to improve cleaning performance for flash memory. *SOFTWARE-PRACTICE & EXPERIENCE 29*, 3 (1999), 267–290.

[13] Deal, E. Trends in nand flash memory error correction. *Cyclic Design, June* (2009).

[14] Duann, N. Error correcting techniques for future nand flash memory in ssd applications. *Technology* (2009), 1.

[15] Electronics, S. K9f8g08uxm flash memory datasheet.

[16] Eshghi, K., and Micheloni, R. Ssd architecture and pci express interface. In *Inside Solid State Drives (SSDs)*. Springer, 2013, pp. 19–45.

[17] Gallager, R. G. Low-density parity-check codes. *Information Theory, IRE Transactions on 8*, 1 (1962), 21–28.

[18] Gao, C., Shi, L., Wu, K., Xue, C. J., and Sha, E. H. Exploit asymmetric error rates of cell states to improve the performance of flash memory storage systems. In *Computer Design (ICCD), 2014 32nd IEEE International Conference on* (2014), IEEE, pp. 202–207.

[19] Gregori, S., Cabrini, A., Khouri, O., and Torelli, G. On-chip error correcting techniques for new-generation flash memories. *Proceedings of the IEEE 91*, 4 (2003), 602–616.

[20] Gupta, A., Kim, Y., and Urgaonkar, B. *DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings*, vol. 44. ACM, 2009.

[21] Hu, X. Ldpc codes for flash channel. *Proc. Flash Memory Summit* (2012).

[22] Huang, P., Subedi, P., He, X., He, S., and Zhou, K. Flexecc: partially relaxing ecc of mlc ssd for better cache performance. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)* (2014), pp. 489–500.

[23] Kang, D., Baek, S., Choi, J., Lee, D., Noh, S. H., and Mutlu, O. Amnesic cache management for non-volatile memory. In *Mass Storage Systems and Technologies (MSST), 2015 31st Symposium on* (2015), IEEE, pp. 1–13.

[24] Kang, J.-U., Jo, H., Kim, J.-S., and Lee, J. A superblock-based flash translation layer for nand flash memory. In *Proceedings of the 6th ACM & IEEE International conference on Embedded software* (2006), ACM, pp. 161–170.

[25] Lee, S., Shin, D., Kim, Y.-J., and Kim, J. Last: locality-aware sector translation for nand flash memory-based storage systems. *ACM SIGOPS Operating Systems Review 42*, 6 (2008), 36–42.

[26] Lee, S.-W., Park, D.-J., Chung, T.-S., Lee, D.-H., Park, S., and Song, H.-J. A log buffer-based flash translation layer using fully-associative sector translation. *ACM Transactions on Embedded Computing Systems (TECS) 6*, 3 (2007), 18.

[27] Li, Q., Shi, L., Xue CJason, W. K., Ji, C., Zhuge, Q., and Sha, E. Access characteristic guided read and write cost regulation for performance improvement on flash memory. In *14th USENIX Conference on File and Storage Technologies (FAST16)* (2016), p. 125.

[28] Li, S., and Zhang, T. Improving multi-level nand flash memory storage reliability using concatenated bch-tcm coding. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 18*, 10 (2010), 1412–1420.

[29] Liu, R.-S., Yang, C.-L., and Wu, W. Optimizing nand flash-based ssds via retention relaxation. In *Proceedings of the 10th USENIX conference on File and Storage Technologies (FAST12)* (2012).

[30] MacKay, D. J. Good error-correcting codes based on very sparse matrices. *Information Theory, IEEE Transactions on 45*, 2 (1999), 399–431.

[31] Motwani, R., and Ong, C. Robust decoder architecture for multi-level flash memory storage channels. In *Computing, Networking and Communications (ICNC), 2012 International Conference on* (2012), IEEE, pp. 492–496.

[32] Pan, Y., Dong, G., Wu, Q., and Zhang, T. Quasi-nonvolatile ssd: Trading flash memory nonvolatility to improve storage system performance for enterprise applications. In *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on* (2012), IEEE, pp. 1–10.

[33] Pan, Y., Dong, G., and Zhang, T. Exploiting memory device wear-out dynamics to improve nand flash memory system performance. In *FAST* (2011), vol. 11, pp. 18–18.

[34] Park, D., and Du, D. H. Hot data identification for flash-based storage systems using multiple bloom filters. In *Mass Storage Systems and Technologies (MSST), 2011 IEEE 27th Symposium on* (2011), IEEE, pp. 1–11.

[35] Suh, K.-D., Suh, B.-H., Lim, Y.-H., Kim, J.-K., Choi, Y.-J., Koh, Y.-N., Lee, S.-S., Suk-Chon, S.-C., Choi, B.-S., Yum, J.-S., et al. A 3.3 v 32 mb nand flash memory with incremental step pulse programming scheme. *Solid-State Circuits, IEEE Journal of 30*, 11 (1995), 1149–1156.

[36] Sun, F., Rose, K., and Zhang, T. On the use of strong bch codes for improving multilevel nand flash memory storage capacity. In *IEEE Workshop on Signal Processing Systems (SiPS): Design and Implementation* (2006).

[37] Swanson, S. Flash memory overview.

[38] Tanakamaru, S., Yanagihara, Y., and Takeuchi, K. Over-10×-extended-lifetime 76%-reduced-error solid-state drives (ssds) with error-prediction ldpc architecture and error-recovery scheme. In *2012 IEEE International Solid-State Circuits Conference* (2012).

[39] Wu, G., He, X., Xie, N., and Zhang, T. Diffecc: improving ssd read performance using differentiated error correction coding schemes. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on* (2010), IEEE, pp. 57–66.

[40] Xia, Q., and Xiao, W. Flash-aware high-performance and endurable cache. In *Modeling, Analysis and Simulation of Computer and T-elecommunication Systems (MASCOTS), 2015 IEEE 23rd International Symposium on* (2015), IEEE, pp. 47–50.

[41] Xia, Q., and Xiao, W. High-performance and endurable cache management for flash-based read caching. *IEEE Transactions on Parallel and Distributed Systems PP*, 99 (2016), 1–1.

[42] Xie, N., Dong, G., and Zhang, T. Using lossless data compression in data storage systems: Not for saving space. *Computers, IEEE Transactions on 60*, 3 (2011), 335–345.

[43] Yang, J. Novel ecc architecture enhances storage system reliability. *Proc. Flash Memory Summit* (2012).

[44] Yeo, E. An ldpc-enabled flash controller in 40 nm cmos. *Proc. Flash Memory Summit* (2012).

[45] Zhao, K., Zhao, W., Sun, H., Zhang, X., Zheng, N., and Zhang, T. Ldpc-in-ssd: making advanced error correction codes work effectively in solid state drives. In *Presented as part of the 11th USENIX Conference on File and Storage Technologies (FAST 13)* (2013), pp. 243–256.

[46] Zhou, Y., Wu, F., Huang, P., He, X., Xie, C., and Zhou, J. An efficient page-level ftl to optimize address translation in flash memory. In *Proceedings of the Tenth European Conference on Computer Systems* (2015), ACM, p. 12.