Unified Wear-leveling Technique for NVM-based Buffer of SSD

Young Min Park, Joonhyeok Yeom, Dohyun Kim, and Eui-Young Chung, *Member, IEEE*

***Abstract*—Since the gradual emergence of DRAM cost issues from the perspective of solid-state drive (SSD) system design, studies to replace DRAM buffer with a non-volatile memory (NVM) have been conducted. However, the inferior durability of NVM requires additional wear-leveling algorithms for the buffer. Related works on the wear-leveling of the SSD internal buffer are significant only for specific applications or restricted partitions separated by user data and metadata. This paper proposes a novel wear-leveling technique that covers the entire buffer without being constrained by specific applications. We introduce metadata characteristic-aware allocation (MCAA), which utilizes an appropriate hot data detection scheme depending on the type of update pattern on metadata. We also propose unified wear-leveling (UWL) that hierarchically integrates the wear-leveling techniques of each partition, including MCAA. The proposed method improves the maximum write count by 39.3% and the standard deviation of the wear-out by 25.9% on average compared to the conventional technique.**

***Index Terms*—SSD, non-volatile memory, endurance, buffer wear-leveling, flash translation layer, metadata.**

I. Introduction

IN the traditional solid-state drive (SSD) architecture, DRAM is utilized as an internal buffer to address the low latency and throughput of NAND flash memory (NFM). The internal buffer serves as a cache memory for user data and flash translation layer (FTL)-related metadata and is divided into partitions for each data type. Therefore, the user data partition has a data structure in page units, and the metadata partition is composed of data structures of general data types such as integer and bool. As the capacity of the SSD expands, the size of this internal DRAM has also increased proportionally, which has resulted in increased manufacturing costs [1].

As an approach to reduce cost, studies have been conducted to replace the DRAM with cost-efficient non-volatile memory (NVM), such as phase-change RAM (PRAM) or magnetoresistive RAM (MRAM) [2]–[10]. NVM is a suitable device for replacing DRAM with read and program latency similar to DRAM and non-volatility that does not require the refresh operation. However, the limited durability of NVM compared to the semi-permanent durability of DRAM can cause lifespan issues when NVM is used as a buffer. Therefore, wear-leveling of the buffer area is essential to maximize the lifespan of the buffer [11].

In an NVM-based buffer of an SSD, the endurance of each buffer partition occupied by user data and metadata is consumed differently. User data are updated based on the page size, and metadata are updated using the cache line size. Therefore, each buffer partition should be handled using appropriate techniques. Consequently, previous studies [12]–[14] investigated wear-leveling techniques by focusing on one of the partitions. In particular, Curling [12], which is a technique for the metadata partition, improves the lifetime of the buffer via swap-based management of grouped hot data. Conversely, multi-bloom filter-based wear-leveling (MBF) [14], a technique used for the user data partition, has achieved noticeable performance improvements by evicting cold data and replacing its location with hot data.

However, these studies have two limitations. First, contrary to the scheme for user data, the technique for the metadata partition is host application-specific because the hot data are detected via profiling during design time. Second, because it manages only transactions in a specific partition, managing durability consumption by transactions occurring in other partitions is insufficient.

In this study, we propose a novel wear-leveling technique through step-by-step wear-leveling from the partition to the entire buffer to overcome these issues. The main contribution of our proposal is twofold. First, we propose a metadata characteristic-aware allocation (MCAA) to improve the local wear-leveling of the metadata partition. Metadata is classified into two types: metadata with fixed hot spots and metadata with random hot spots due to the operation characteristics of the SSD. The MCAA leverages suitable hot data detection for two types of metadata to address the challenges of traditional metadata partition techniques. Second, we propose unified wear-leveling (UWL), which equalizes the wear-out of the entire buffer based on local wear-leveling.

This architecture achieves local wear-leveling within the user data partition and metadata partition using MBF and MCAA-based swap management, respectively. Subsequently, the durability of the entire buffer is equalized as the physical location of each partition is gradually shuffled through the extended swap operation.

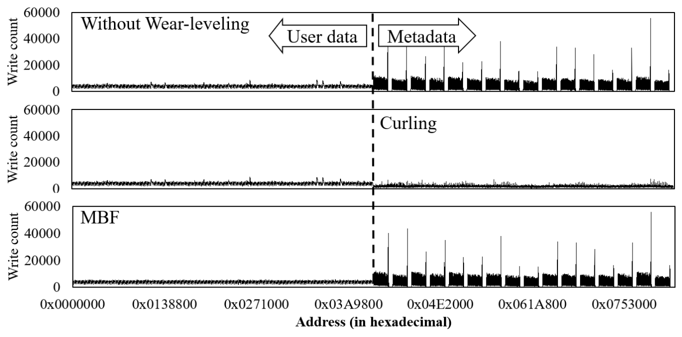
II. Related Work And Motivation

A. Related works

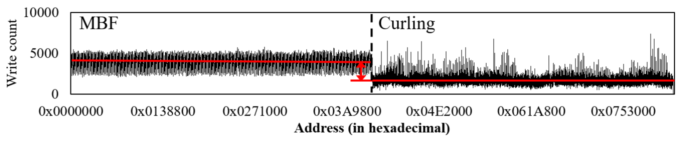
Curling [12], for the metadata partition, focuses on extreme updates concentrated in a specific narrow metadata partition.

This work was supported by Institute of Information communications Technology Planning Evaluation(IITP) grant funded by the Korea government(MSIT) (No.2022-0-00050, Development of PIM Computing Architecture based on Data-Flow). The EDA tool was supported by the IC Design Education Center(IDEC), Korea. (Corresponding author: Eui-Young Chung.)

The authors are with the School of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, South Korea (e-mail: ympark0225@yonsei.ac.kr; joonhyeok_yeom@yonsei.ac.kr; middk@yonsei.ac.kr; eychung@yonsei.ac.kr)



(a) Effect of conventional techniques for partitions



(b) Wear-out deviation between partitions on simple integration of techniques

Fig. 1. Visualization of write counts on the buffer with conventional techniques in the Financial benchmark.

To distribute intensive updates, they suggest to manage the metadata in which intensive updates appear by grouping them into hot regions. The hot spots of metadata in this management are analyzed based on profiling host application, and are statically allocated to the hot regions with the help of the compiler. Then, wear-leveling is performed sequentially swapping the clustered hot regions with cold regions.

Another work for the user data partition, MBF [14], presented a wear-leveling technique that focused on managing user data transactions of page size and neglected the metadata transactions of a relatively small size. This technique distinguishes the hot and cold page numbers in the user data entries using an MBF. Then, by evicting the data in the existing cold page entries to NFM and replacing it with the hot data of user data to be updated, the endurance in the user data partition is balanced.

B. Motivation

Fig.1(a) shows the excellent wear-leveling effect of each technique in partitions as well as their limitations. For MBF, the peak update in the metadata area cannot be controlled because the metadata area traffic is ignored. In contrast, the write traffic in the user data partition is evenly distributed. Thus, limited to the user data partition, we can exploit it as a technique for the user data partition without additional modification. Curling also improves the write distribution of metadata; however, it does not cover the intensive traffic of the user data partition. A simple integration of wear-leveling techniques for each partition can serve as an approach to address the technological limitations.

Fig.1(b) shows the experimental results of the simple integration of the techniques of each partition. Here, wear-leveling of each partition is achieved by adopting a technique suitable for the transaction characteristics of the partition. However, the experimental results also show that the user data partition accumulates 2.5 times more programs on average

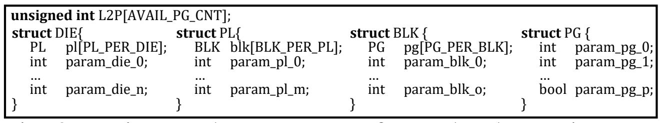

```
unsigned int L2P[AVAIL_PG_CNT];
struct DIE{                 struct PL{                  struct BLK {                struct PG {
    PL    pl[PL_PER_DIE];       BLK   blk[BLK_PER_PL];      PG    pg[PG_PER_BLK];       int   param_pg_0;
    int   param_die_0;          int   param_pl_0;           int   param_blk_0;          int   param_pg_1;
    ...                         ...                         ...                         ...
    int   param_die_n;          int   param_pl_m;           int   param_blk_o;          bool  param_pg_p;
}                           }                           }                           }
```

Fig. 2. Basic metadata structure of page-level mapping FTL.

than the metadata partition because of the different total sizes of transactions. The management method for each partition inevitably causes these differences in endurance consumption between partitions. Consequently, the user data partition is the first to wear out and reach the end of the buffer lifetime. If the deviation between partitions can be mitigated, lifetime of the entire buffer can be further improved.

We also note the need to address the limitation of the application-specific wear-leveling technique for the metadata partition from the FTL operation and metadata structure. Fig.2 shows an example of a typical metadata structure. These configurations are generic and independent of the language of the firmware; only the hierarchical constructs depend on the address translation type of the FTL. We use a page-level FTL for the description in this brief because most SSD prioritizes performance and adopt page-level mapping [15].

Logical to physical (L2P) mapping information is stored in an array with the size of AVAIL_PG_CNT, indicating the page count available to NFM. For metadata for the physical information of NFM (MPIN), the information is stored in a hierarchical data structure that resembles the architecture of the NFM. For instance, the data structure for a block, BLK, consists of PG_PER_BLK(representing the number of pages per block) instances of a page and parameters associated with the block. Parameters such as param_blk_0 in the BLK can be used to record writable pages for the essential operation of the FTL or block erase counts for garbage collection. While updating these MPINs, the sequential program order and wear-leveling mechanism of NFM, which uniformly uses blocks in all the planes, results in architecture-dependent write patterns and fixed hot spots on the MPIN. Therefore, metadata are divided into two types with different hot spot occurrence characteristics: the L2P table and MPIN.

In this brief, we reinforce the technique for the metadata partition based on the above-mentioned motivation. We also introduce unified wear-leveling to balance the endurance of the entire buffer by integrating the partition techniques.

III. Proposed Method

We introduce two key developments in our proposed method. First, we propose MCAA that differentiates hot data allocation according to metadata type. Second, we propose UWL, which hierarchically integrates MCAA-based swap management and MBF, which are techniques for partitions.

A. Metadata Characteristic-aware Allocation

The core of swap-based metadata partition management is detecting and clustering hot data into hot regions, which are intensively managed. Fig.3 shows the metadata allocation strategy in the conventional curling (CC) and the proposed MCAA. In the CC, metadata are statically allocated into cold regions and hot regions based on the profiling of the

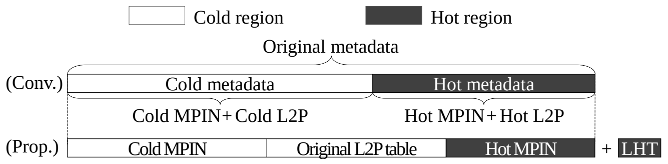


Fig. 3. Metadata allocation strategy in the conventional and the proposed MCAA.

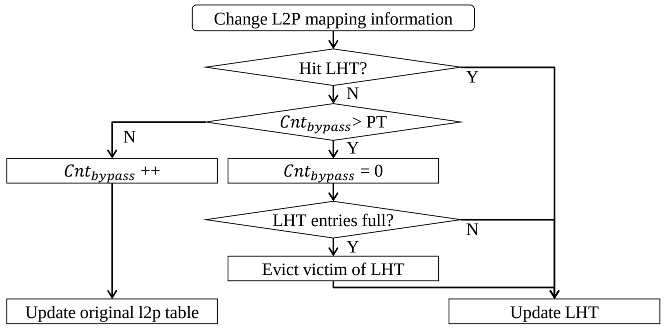


Fig. 4. Bypass mechanism of the LHT.

application. Therefore, it is impossible to respond to various applications at runtime owing to the characteristics of the L2P table. To solve this issue, we use different allocation strategies to keep hot data in hot regions according to the two types of metadata mentioned in Section II-B, MPIN and L2P table.

1) MPIN: Owing to sequential program order and MPIN structure, we can infer that MPIN included in higher layers such as die, plane, and block require continuous updates and will become fixed hot spots. In contrast, most of the MPINs related to the page are cold data. Therefore, hot and cold MPIN can be distinguished by profiling the FTL and MPIN structures before runtime.

2) L2P table: In contrast to MPIN, the L2P table represents an application-dependent update pattern because it is indexed by a logical address of a host request. Thus, hot spots in the L2P table are unpredictable and should be handled at runtime. To concentrate the traffic of updating L2P information in hot regions, we allocate the L2P history table (LHT) for each plane of NFMs to hot regions separately from the original L2P table and utilize this by modifying the FTL. A single entry in the LHT contains a logical address, physical address, and two variables for the least recently used (LRU) operation. During runtime, the modified FTL preferentially updates L2P information to the LHT. The FTL operates the LHT as an LRU victim policy and evicts victim entries to the original mapping table. The LHT then maintains the host request logical address where continuous L2P updates due to temporal locality.

However, misses in the LHT cause additional write amplification owing to updating the original L2P table of the victim entry. To minimize the miss penalty, we implemented a bypass mechanism. Fig.4 shows the L2P information update flow, including the bypass algorithm. The LHT updates information only when bypass count (Cnt_{bypass}) reaches the predetermined threshold(PT). If the condition is not satisfied, Cnt_{bypass} is incremented and the original table updates the L2P information.

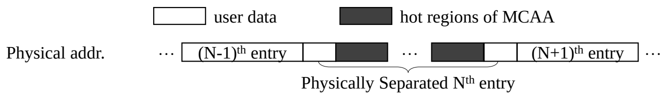


Fig. 5. Example of incomplete transaction of user data.

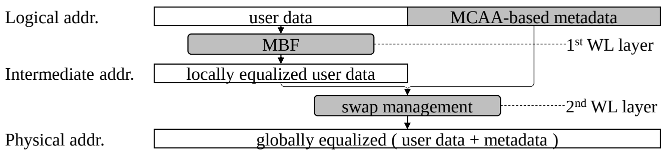


Fig. 6. Unified wear-leveling with MCAA.

B. Unified wear-leveling with MCAA

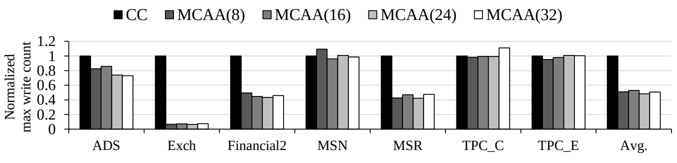
The swap-based management of MCAA-based metadata and MBF on user data can accomplish local wear-leveling in each partition. As mentioned in Section II-B, differences in the wear-out between partitions cannot be resolved with simple integration of techniques for partitions. The physical location of partitions should be shuffled through separate management to balance the endurance of the entire buffer. However, separate management causes overhead owing to additional swap operations. To minimize the overhead due to separate management, we propose a UWL that hierarchically integrates wear-leveling techniques for partitions. In UWL, we extend the operation of the swap-based management of MCAA to the entire buffer and leverage it. Extending the coverage of MBF to the entire buffer is impossible because the mechanism of MBF is the replacement of data and not the swap.

One problem needs to be solved to expand the range of the operation of the swap-based management of the MCAA. Fig.5 shows a problem case of an arbitrary user data entry separated by the hot regions of the MCAA. In this case, we implement a buffer request divider to supplement user data transactions. The buffer request divider preferentially converts the logical information of user data requests to physical information. Then, the physical address of the user data request is compared with the start and end address of hot regions to determine whether the requested user data entry is physically separated. If hot regions of the MCAA separate user data entry, the buffer request divider splits the original request into sub-requests to maintain the integrity of the data IO.

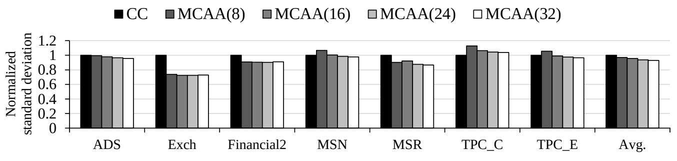
Fig.6 shows an overview of the UWL consisting of two wear-leveling layers. The write distribution is performed as follows. First, user data are distributed uniformly at the intermediate address through the MBF of the first wear-leveling layer. Subsequently, hot metadata distinguished through MCAA are distributed to the entire buffer through the periodic swap management of the second wear-leveling layer. Simultaneously, the user data partition and metadata partition are gradually shuffled to equalize globally by the swap operation of the second wear-leveling layer. As this swap operation progresses, the logical to the physical address mapping of the entire buffer continuously shifts. Within swap management, the logical to physical mapping information of the buffer is maintained using simple arithmetic-based address translation similar to CC.

TABLE I. Hardware specifications.

NFM	NFM capacity		4GB MLC		
	NFM channel/way/plane		4/4/1		
	Page size(KB)		16384		
	Latency	tR(us)	45		
		tPROG(us)	700		
		tBERS(ms)	3.5		
PRAM	PRAM capacity(MB)		8	12	20
	Latency	tR(ns)	45		
		tPROG(ns)	600		



(a) Maximum write count of the buffer.



(b) Standard deviation of the write counts of the buffer.

Fig. 7. Wear-leveling performance of MCAA according to number of entries in the LHT.

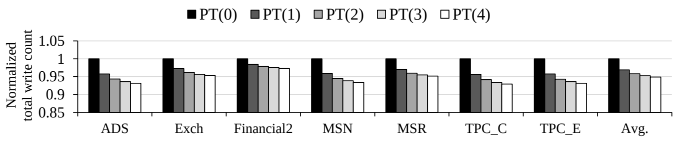
IV. Experiment

A. Experimental Setup

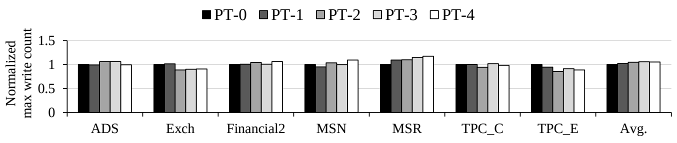
We implemented a trace-driven SSD simulator using the Synopsys Platform Architect to evaluate the MCAA and UWL. The implemented SSD operates a page-level FTL which performs greedy garbage collection. We applied PRAM to the buffer, and Table I shows hardware specifications for NFM and PRAM, respectively. In particular, to observe the global wear-leveling effect of the UWL, experiments were conducted in various environments with buffer sizes of 8, 12, and 20MB. The workload used in the evaluation was acquired from [16] and [17], and their characteristics are summarized in [18].

B. Experimental Results

1) Effect of MCAA: We evaluated MCAA compared with CC in the metadata partition. Figs.7(a) and 7(b) demonstrate the wear-leveling performance of MCAA according to the number of entries in the LHT. In the figure legend, MCAA(N) indicates that the number of LHT entries for each plane in MCAA is N. As shown in Fig.7(a), MCAA dramatically reduces the maximum write count by 51.0% on average over CC. There is a fluctuation of wear-leveling performance according to the LHT size change due to the characteristic of reflecting only the outlier, but MCAA shows improvement in specific benchmarks such as ADS, Exchange, Financial2, and MSR, in which updates of specific logical addresses are excessive as summarized in [18]. Because the peak count is eliminated by effectively handling the random hot spot through the LHT in the metadata partition. The visualized write count graphs related to the experiment are provided in [18]. For the standard deviation, there is an improvement of 3.1% compared to CC because of the removal of the peak write count as shown in Fig.7(b). Especially, It is observed that as the number of LHT entries increases, the concentration of LHT updates in specific entries decreases, resulting in a reduction in the standard deviation. This observation demonstrates the effectiveness of increasing the LHT size in MCAA for achieving improved wear-leveling performance.



(a) Total write counts on buffer.



(b) Maximum write count on buffer

Fig. 8. Overhead improvement and performance degradation according to PT change.

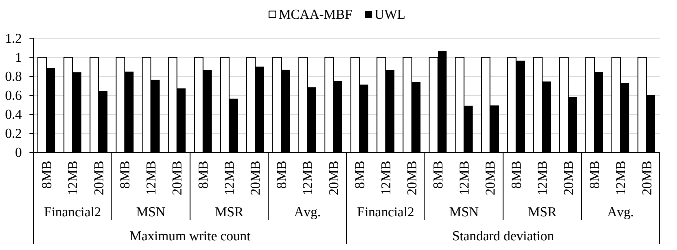
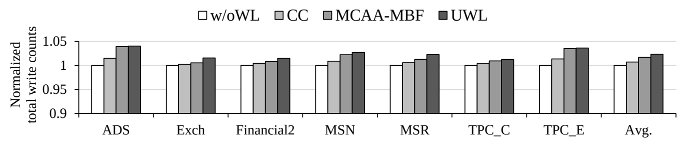


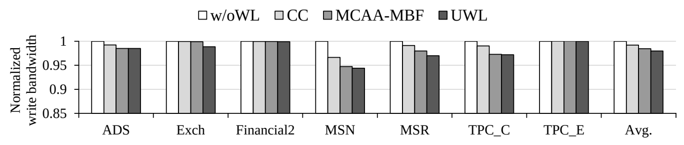
Fig. 9. Wear-leveling performance of UWL normalized to MCAA-MBF.

Figs.8(a) and 8(b) show the write amplification reduction effect and performance degradation, respectively, when we apply the bypass mechanism to the MCAA with 16 entries of the LHT. In the figure legend, PT(N) represents the model in which the value of the predetermined threshold is N in the bypass mechanism. PT(0) is the same as the general model for operating the LHT without the bypass mechanism. As the value of PT increases, Fig.8(a) shows that the total write counts decrease by 3.1% to 5.1% on average compared to PT(0). However, even if the value of PT increases, the maximum write count, an indicator of wear-leveling performance, has only a 6.0% performance degradation on average compared to PT(0), even in a case such as PT(4) as shown in Fig.8(b). These two results indicate that the bypass mechanism effectively eliminates the write amplification caused by unnecessarily registering logical addresses with weak temporal locality in the LHT. Hence, MCAA with bypass mechanism can expect a peak write count reduction of approximately 50% on average compared to CC and can contribute to preventing the shortening of the lifespan of the metadata partition.

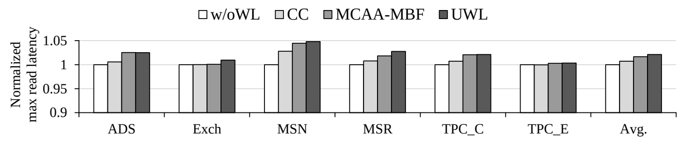
2) Wear-leveling Performance of UWL: In the experiment, we evaluated the UWL by varying the buffer size with benchmarks such as Financial2, MSN, and MSR; these benchmarks have sufficient write requests for observing the global wear-leveling effect. Fig.9 shows the results of the wear-leveling performance of UWL in comparison with that of MCAA-MBF. In the 20MB environment, UWL showed a noticeable improvement of 39.3% in terms of the maximum write count and 25.9% in terms of the standard deviation over MCAA-



(a) Total write counts on buffer.



(b) Write bandwidth of SSD.



(c) Maximum read latency.

Fig. 10. Overhead of wear-leveling.

MBF. On the other hand, in the 8MB environment, the MSN experiment showed an increase of 6.6% in standard deviation. This is because the MSN benchmark has a low locality, causing the user data partition to already be evenly utilized, and aligned with the average of the entire buffer at this environment. With the swap operation of UWL, the increase in noise led to the observed result of increased standard deviation. The detailed write count graphs related Fig.9 are provided in [18]. These results indicate that the UWL effectively addressed the limitation of the MCAA-MBF structure, in which the variation in the endurance consumption between partitions increases with the increase in the buffer size. This mitigation effect of UWL becomes essential in real world scenarios where workloads consist of combinations of benchmarks with varying characteristics despite the slight degradation observed in extreme case.

3) Overhead Analysis: The MCAA-based swap management and buffer request divider have negligible area overhead if they are implemented in the hardware. This is because they require fewer than two steps of arithmetic operations based on fewer than ten simple parameters, similar to CC. For an additional buffer area due to the use of the LHT, 16 bytes for one entry are consumed, and because each plane has 16 entries, 16 planes make a total of 4096 bytes. This is also negligible overhead with less than 0.05% of the total size of the buffer at 20MB. In addition, the swap-based wear-leveling technique inevitably caused an overhead, which increased the total write counts of the buffer. Fig.10(a) demonstrates the total number of writes that increased due to the write amplification of MCAA-MBF and UWL in an environment where the MCAA has a PT value of 2 and the number of entries is 16. Even with workloads such as extreme ADS with 4.0% overhead, the average overhead of the write amplification increases by 2.3% over an environment without wear-leveling is sufficient. Figs.10(b) and 10(c) depict decrease in the write bandwidth and increase in the maximum read latency, which are indicators of system performance. An average system performance degradation of 2.2% was observed in the write bandwidth, and the read latency was reduced by 2.1% on average. This performance overhead is tolerable for extending the lifespan of the internal buffer of an SSD.

V. Conclusion

We propose a novel wear-leveling scheme for the NVM-based buffer in an SSD. MCAA reduces the peak writes counts on the metadata partition by 51.2% and maintains the standard deviation as compared to the conventional method. Unlike CC, the MCAA does not require profiling for each application change. In addition, the UWL with MCAA reduces the maximum write count and standard deviation by 39.3% and 25.9%, respectively, as compared to the simple integration of techniques for each buffer partition. The UWL has a negligible overhead of 2% and 0.05% under, respectively, for the performance and buffer area compared to the existing scheme. Our proposed technique will be more beneficial in a practical environment as it maintains a flexible performance that is suitable for various applications.

References

[1] S. Park, Y. Kim, B. Urgaonkar, J. Lee, and E. Seo, "A comprehensive study of energy efficiency and performance of flash-based ssd," *Journal of Systems Architecture*, vol. 57, no. 4, pp. 354–365, 2011.

[2] S. Kang, S. Park, H. Jung, H. Shim, and J. Cha, "Performance trade-offs in using nvram write buffer for flash memory-based storage devices," *IEEE Transactions on Computers*, vol. 58, no. 6, pp. 744–758, 2009.

[3] Y. Liu, C. Zhou, and X. Cheng, "Hybrid ssd with pcm," in *2011 11th Annual Non-Volatile Memory Technology Symposium Proceeding*, 2011, pp. 1–5.

[4] N. Lu, I.-S. Choi, S.-H. Ko, and S.-D. Kim, "An effective hierarchical pram-slc-mlc hybrid solid state disk," in *2012 IEEE/ACIS 11th International Conference on Computer and Information Science*, 2012, pp. 113–118.

[5] Q. Wei, C. Chen, and J. Yang, "Cbm: A cooperative buffer management for ssd," in *2014 30th Symposium on Mass Storage Systems and Technologies (MSST)*, 2014, pp. 1–12.

[6] J.-Y. Kim, S.-H. Park, H. Seo, T. You, and E.-Y. Chung, "A read-while-write-based out-of-order scheduling for high performance nand flash-based storage devices," in *The 18th IEEE International Symposium on Consumer Electronics (ISCE 2014)*, 2014, pp. 1–2.

[7] D. Kim and S. Kang, "Dual region write buffering: Making large-scale nonvolatile buffer using small capacitor in ssd," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, ser. SAC '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 2039–2046.

[8] S. J. Han, D. H. Kang, and Y. I. Eom, "Hybrid write buffer algorithm for improving performance and endurance of nand flash storages," in *2016 IEEE International Conference on Consumer Electronics (ICCE)*, 2016, pp. 83–84.

[9] S. Ikegawa, F. B. Mancoff, J. Janesky, and S. Aggarwal, "Magnetoresistive random access memory: Present and future," *IEEE Transactions on Electron Devices*, vol. 67, no. 4, pp. 1407–1419, 2020.

[10] S. Ikegawa, F. B. Mancoff, and S. Aggarwal, "Commercialization of mram – historical and future perspective," in *2021 IEEE International Interconnect Technology Conference (IITC)*, 2021, pp. 1–3.

[11] H. Farbeh and N. Rohbani, "Pcm-oriented cache management strategies for solid-state disks," in *2018 Real-Time and Embedded Systems and Technologies (RTEST)*, 2018, pp. 16–23.

[12] D. Liu, T. Wang, Y. Wang, Z. Shao, Q. Zhuge, and E. H.-M. Sha, "Application-specific wear leveling for extending lifetime of phase change memory in embedded systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 10, pp. 1450–1462, 2014.

[13] S. J. Kwon, "Non-volatile translation layer for pcm+nand in wearable devices," *IEEE Transactions on Consumer Electronics*, vol. 63, no. 4, pp. 483–489, 2017.

[14] Zhao *et al.*, "Architectural exploration to address the reliability challenges for reram-based buffer in ssd," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 1, pp. 226–238, 2019.

[15] H. Chen, C. Li, Y. Pan, M. Lyu, Y. Li, and Y. Xu, "Hcftl: A locality-aware page-level flash translation layer," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019, pp. 590–593.
[16] "U mass trace repositor," http://traces.cs.umass.edu.
[17] "Storage networking industry association," http://iotta.snia.org.
[18] (2023) *Supplementary Data*. [Online]. Available: https://dtl.yonsei.ac.kr/docs/UWLsuppl.pdf