Weight-based Wear-leveling considering Performance in Phase Change Memory

Taehee You
School of Electrical and Electronic Engineering
Yonsei University
Seoul, Republic of Korea
xoqhd1212@yonsei.ac.kr

Hyeokjun Seo
School of Electrical and Electronic Engineering
Yonsei University
Seoul, Republic of Korea
jjsky7@yonsei.ac.kr

Eui-Young Chung
School of Electrical and Electronic Engineering
Yonsei University
Seoul, Republic of Korea
eychung@yonsei.ac.kr

***Abstract*—Phase Change Memory (PCM) is one of the most promising memories which can replace traditional memories. However, PCM requires wear-leveling scheme due to a limited write counts. Although many works study for wear-leveling, there is a lack of consideration for system performance degradation caused by wear-leveling. Therefore, we propose weight-based wear-leveling that minimizes the impact on system performance by considering write count, write locality, and wear-leveling overhead. When PCM is used as storage, the proposed method reduces the write standard deviation by an average of 47.4% and a maximum of 74.8% compared to the baseline. Also, when PCM is used as main memory, the standard deviation decreased by an average of 65% and a maximum of 86.9%, and the read latency was improved by 21.8%.**

Keywords-component; Phase Change Memory, wear-leveling, write count, write locality, system performance

I. INTRODUCTION

Phase Change Memory (PCM) is one of the emerging memories, which offers nonvolatile, high durability, and byte addressability. Because of these characteristics, PCM is studied for use in various memory layers (e.g. main memory, storage, cache). Unfortunately, PCM can be reliably written only a limited write count which is 10^8 to 10^9 [1][2]. Therefore, wear-leveling scheme should be essentially applied to PCM.

Wear-leveling is a method of spreading write that is concentrated on a specific line repeatably to distribute write evenly. In wear-leveling, three factors should be considered important. First, the write count of line should be considered. Because repetitive writing to a specific line reduces the lifetime of PCM, the method is needed to distribute the write evenly. To resolve the problem, [3][4] swap the lines between high and low write count, and [5-7] remap the address every predefined interval.

Second, the write locality should be considered. Locality means that the accessed address or around the address is likely to referenced in the near future. However, frequently accessed addresses change according to applications, even the line with a high write count may no longer be written. Therefore, both write count and locality should be considered. In [3][4], data of frequently accessed address is defined as hot data, and it is written to the line with a low write count.

Lastly, system performance degradation due to additional read/write caused by wear-leveling should be considered [called swap overhead]. In PCM, wear-leveling progresses line swap to balance the write count, which generates read/write. If a host request is issued during line swap, the host request is delayed. Because PCM has different read and write latency, the additional read/write has a greater impact on performance degradation.

Most of the previous works suggest wear-leveling related to the first and second factor. In [4], they consider swap overhead but does not consider when host request and swap occur simultaneously. We propose a weight-base wear-leveling (WWL) scheme considering the write count, write locality and swap overhead to increase the life time of PCM. WWL is processed 2-levels that are inter-region consisting of several lines and intra-region. We use [5] with simple and low overhead in intra-region and applies WWL between inter-regions.

The contribution of this paper is as follows.

- We propose wear-leveling considering write locality as well as write count. This can be used to prevent unnecessary swap by determining whether write is continuously issued on a line with a high write count. This enables efficient wear-leveling and minimizes unnecessary swap.
- We propose wear-leveling to minimize swap overhead. The inter-region swap wear-leveling additionally generates a lot of read/write. For this, we propose a method to swap line-by-line without affecting system performance when inter-regions swaps occur.

The remainder of this paper is organized as follows. In Section II, we present the related work and motivation. In Section III, we introduce proposed wear-leveling. In Section IV, we present our experimental results and our conclusions in Section V.

II. RELATED WORK AND MOTIVATION

The related work considering the writ count, write locality, and swap overhead are as follows.

In [7], they propose 2-levels wear-leveling to counter malicious attacks. It prevents the writing of a specific line by

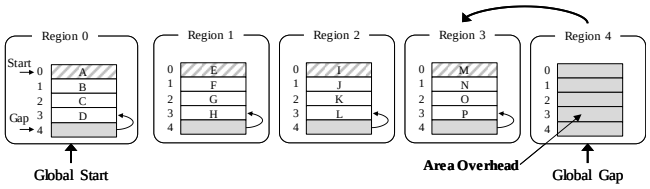

Figure 1. Region based Start-gap wear-leveling [5]

randomly changing the address of the target line every predetermined interval. However, wear-leveling occurs in the case of even write, which causes additional read/write.

In [4], they propose wear-leveling considering system performance. Periodically, they calculate the weight and proceed to the region swap if they exceed the predefined threshold. However, there is no wear-leveling in the region, and there is insufficient consideration of the swap overhead.

In [3], they have the advantage of a small overhead for managing wear-leveling based on 2-levels. However, as with [4], there is a lack of consideration for the swap overhead during region swap.

In [5], they are simple and low overhead. As shown in Fig. 1, it includes start and gap pointer, and the position of the gap is shifted at the predefined interval. The start pointer is incremented if the gap pointer is located on the 0th line. As a result, the address is constantly changed at the predefined interval, so it is possible to write evenly. For this reason, [5] has been combined with various wear-leveling schemes in [6][10][11]. However, when using 2-levels as shown in Fig. 1, additional storage space is needed for the region size. Also, depending on the region size, the movement of the start and the gap pointer takes a long time, and the write count imbalance occurs. In this paper, we use start-gap wear-leveling. To compensate for the disadvantages of the start-gap, it is applied only to wear-leveling within the intra-region.

We propose weight-based wear-leveling considering the write count, write locality, and swap overhead. Depending on the application, the space frequently accessed is constantly changing over time [4]. That is, in case of wear-leveling considering only write count, unnecessary swap is generated. In addition, 2-levels wear-leveling can have a significant effect on system performance during region swap, so a region swap method is needed to minimize it. We reduce the region swap and propose a way to minimize the impact on the system performance when a region swap occurs.

III. PROPOSED METHOD

A. Overview of Weight-based Wear-leveling (WWL)

The proposed weight-based wear-leveling (WWL) compensates for the problem of [5] and minimizes performance degradation. Fig. 2 shows the overview of the proposed wear-leveling. The intra-region wear-leveling uses the start-gap and the inter-region wear-leveling uses the proposed WWL. WWL calculates the weight (WT) of each region at a predefined swap interval. If the difference between WT_{max} and WT_{min} is greater than the predefined

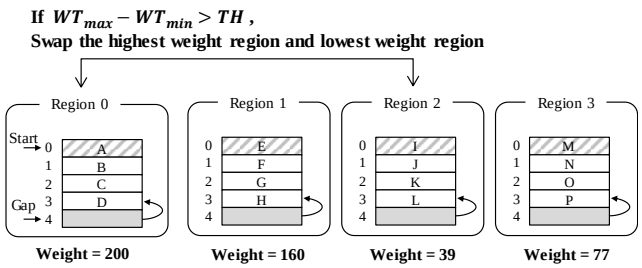

Figure 2. Overview of weight-based wear-leveling

threshold (TH), swap the two regions. WT includes the write count and write locality, two important factors to be considered in the wear-leveling mentioned above.

The formula for calculating the WT of each region is as follows.

$$WT_i = \alpha \times RC_i + (1-\alpha) \times IC_i, \ (0 < \alpha < 1) \quad (1)$$

$$RC_i = WC_i - \min(\forall WC) \quad (2)$$

In (1), WT_i means the weight value of the i-th region and is determined by relative region write count (RC) and intermediate region write count (IC). The α is a coefficient factor obtained by profiling in Section IV.

RC is a factor corresponding to the write count and can be obtained from (2). RC indicates how many write counts are in the region than the region with the lowest write count. In other words, RC reflects the degree of imbalance in the number of writes in each region.

IC is a factor reflecting write locality, which means the write count of the region during the swap interval. Since the memory access pattern is different depending on the application, the write access frequency may be lowered even though the current write count is high. We can prevent unnecessary wear-leveling through IC.

B. Weight-based wear-leveling strategy

WWL applies different wear-leveling scheme to intra-region and inter-region. The intra-region wear-leveling is the same as [5], and the inter-region wear-leveling operates as in Fig. 3.

In Fig. 3, the inter-region swap check that a region swap is needed every swap interval. The inter-region swap decision is determined by (1). If the difference between WT_{max} and WT_{min} is less than TH, the swap is not performed due to the swap overhead. If the inter-region swap that requires a lot of line swaps is determined, it identifies whether the region swap is possible. It can be done via WL_en and allows swap if the memory controller does not have host read requests. The read requests have more impact on system performance than write request because the read request is completed when the data of memory is transferred to the host, and the write request is completed when the request is transferred from the host to the memory. Therefore, we grant the inter-region swap if there is not a read request

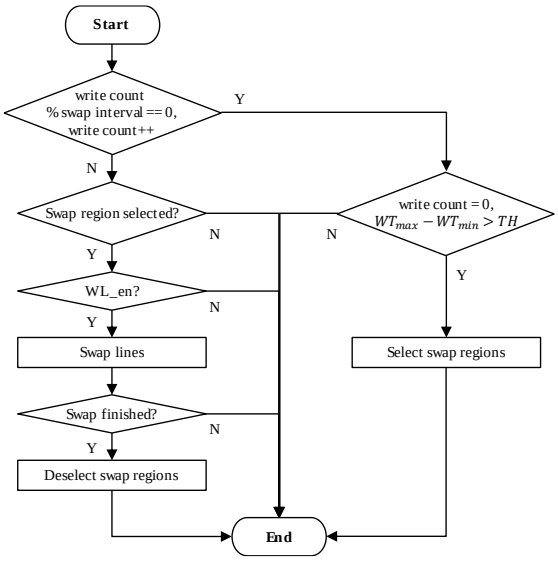

Figure 3. The operation flow chart of weight-based wear-leveling

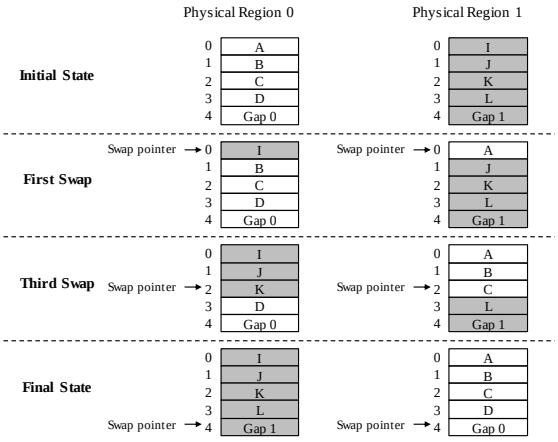

Figure 4. Example of step by step region swap

of the memory controller. Finally, the inter-region swap is terminated after all lines of region have been swapped.

If the all the data between the two selected regions are swapped at once, the performance will decrease because no other requests can be processed until the swap is completed. Therefore, the swap should be performed step by step, and an example of the swap operation applied is shown in Fig. 4. For this, WWL contains a swap pointer register, which stores the progress of the swap. First, when two regions for swap are selected, initialize the swap pointer to zero. Then, if WL_en is 1, we swap the lines pointed to by the swap pointer and increment the swap pointer by one. Finally, we repeat the above steps to completely swap data between the two regions.

If a host request is issued before swap is completely terminated, we can use the swap pointer to get the physical address of the host request. For example, in Fig. 4, if a host read request is issued to line 1 of region 0 at the third swap, we read line 1 of the relative region (e.g. region 1) because line 1 is located before the swap pointer. We can minimize

TABLE I. SIMULATION CONFIGURATION

Attribute		Value
Environment		Synopsys's Platform Architecture (SystemC)
PCM congifuration		4 channel 1 way (1 GB / channel)
Wear-leveling configuration	Intra-region	start-gap swap interval = 100
	Inter-region	α = 0.3 swap interval = (# of line / region) * 100 number of regions = 128

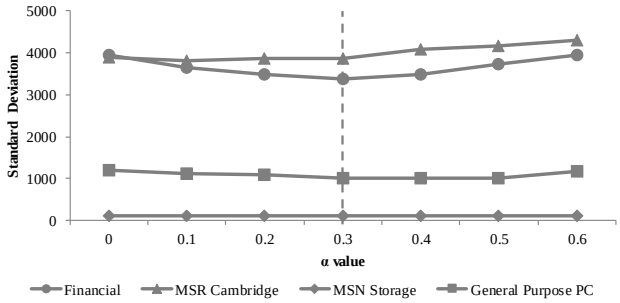

Figure 5. Profiling to define α

the system performance degradation through region swap by step by step.

IV. EXPERIMENT

We evaluated the proposed wear-leveling scheme by implementing a simulator based on Synopsys's Platform Architect using PCM as storage and main memory. The configurations of simulator are summarized in Table 1. We measured evaluation metric such as standard deviation of write count of PCM and read latency. The timing parameters of PCM followed by [12]. We use the financial, MSR Cambridge, MSN storage, and general-purpose PC as storage benchmark, with read/write ratios of (1:5.5), (1:2.3), (2.1:1), and (3.4:1), respectively. Also, we used benchmarks with a lot of memory access as the main memory, and each benchmarks are described in [13]. Our experiments compare the region-based start-gap wear-leveling (RBSG) [5].

Fig. 5 shows the standard deviation of the write count of PCM as a result of experiment to set α. The α is the coefficient of (1) to obtain the weight of each region. The MSN Storage is not affected by the α because it distributes the address evenly without the wear-leveling. Therefore, we set α except MSN Storage. Since the standard deviation is low in most benchmarks when α is 0.3, we set α to 0.3.

Fig. 6 shows the standard deviation of write count according to the number of regions. As the number of regions increases, the wear leveling effect also increases. However, the increase in the number of regions increases the hardware overhead for storing information for each region. Since the standard deviation reduction rate decreases after 128 regions, we set the number of regions to 128.

Fig. 7 is an experiment to analyze the wear leveling performance difference between RBSG and WWL.

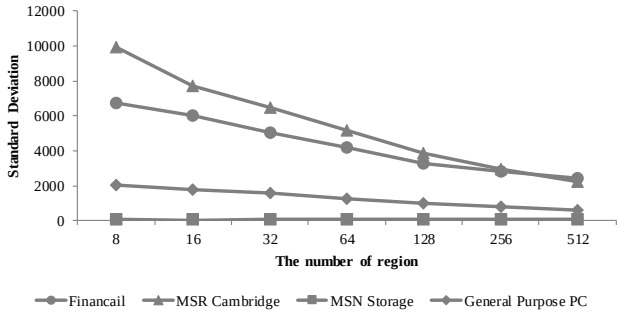

Figure 6. Profiling to define the number of region

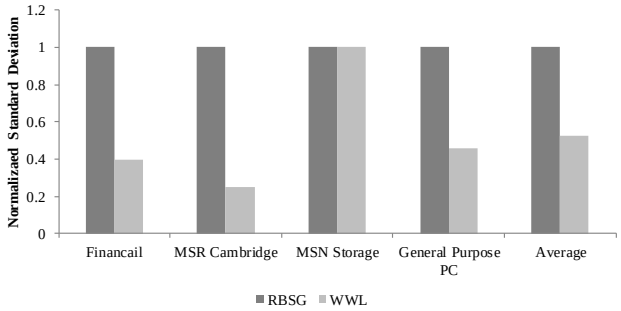

Figure 7. Normalizaed write standard deviation in storage

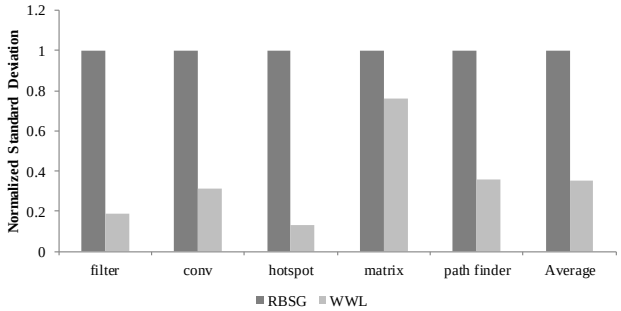

Figure 8. Normalizaed write standard deviation in main memory

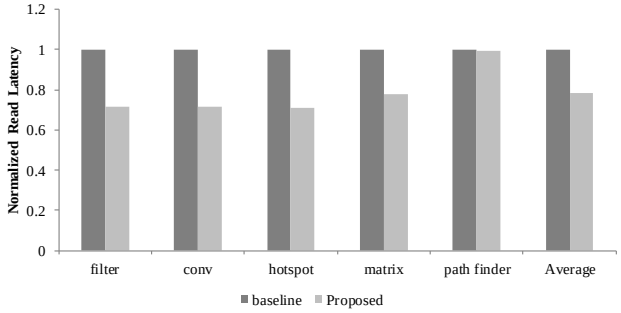

Figure 9. Normalizaed read latency in main memory

The MSN Storage has similar standard deviations in both methods because it contains even distribution of write as mentioned above. In other benchmarks, however, RBSG shows a high standard deviation. It is because RBSG requires a lot of write until inter-region swap occurs. WWL reduced the standard deviation of write by up to 74.8% and average by 47.4% compared to RBSG.

Fig. 8 shows the standard deviation of write when using PCM as main memory. We can see that the standard deviation of RBSG and WWL is much different when we use PCM as storage. This is because more write requests have occurred in a particular region than the storage benchmark. As a result, WWL improved by a maximum of 86.9% and an average 65% standard deviation from the RBSG.

Fig. 9 shows the read latency of wear-leveling considering performance when using PCM as main memory. RBSG was excluded from comparison because of the large standard deviation from the WWL. Also, because of the large time interval between requests in the case of storage, the difference in read latency is not clearly visible and is excluded. On the other hand, in main memory, there are many differences in read latency because the time interval between requests is dense. In the case of the path finder, there was less improvement in read latency due to fewer read requests, but we could improve the average 21.7% read latency through wear-leveling considering performance.

V. CONCLUSION

We have proposed a weight-base wear-leveling method considering performance. This includes write count, write locality, and performance factor that should be considered important in wear-leveling. We reduced the write standard deviation of the average 47.4% and 65% compared with RBSG when using pcm in storage and main memory and reduced the read latency by 21.7% when using pcm as main memory. As a future work, the proposed method could be extended to improve wear-leveling performance by optimizing the size of line and region. Also, we will apply wear-leveling schemes in the new memories (e.g. STT-MRAM, ReRAM) considering each memory characteristic, such as size of line or region and read/write latency.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (2016R1A2B4011799), and in part by Multi-Ministry Collaborative R&D Program(R&D program for complex cognitive technology) through the National Research Foundation of Korea(NRF) funded by MSIT, MOTIE, KNPA(2018M3E3A1057248), and in part by the EDA tool was supported by the IC Design Education Center(IDEC), Korea.

REFERENCES

[1] XUE, Chun Jason, et al. Emerging non-volatile memories: opportunities and challenges. In: Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2011 Proceedings of the 9th International Conference on. IEEE, 2011. p. 325-334.

[2] BURR, Geoffrey W., et al. Phase change memory technology. Journal of Vacuum Science & Technology B, Nanotechnology and Microelectronics: Materials, Processing, Measurement, and Phenomena, 2010, 28.2: 223-262.

[3] YUN, Joosung; LEE, Sunggu; YOO, Sungjoo. Bloom filter-based dynamic wear leveling for phase-change RAM. In: Proceedings of the

Conference on Design, Automation and Test in Europe. EDA Consortium, 2012. p. 1513-1518.

[4] DONG, Jianbo, et al. Wear rate leveling: Lifetime enhancement of PRAM with endurance variation. In: Proceedings of the 48th Design Automation Conference. ACM, 2011. p. 972-977.Bloom filter based dynamic wear leveling.

[5] QURESHI, Moinuddin K., et al. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture. ACM, 2009. p. 14-23.Security refresh 2010.

[6] JIANG, Lei, et al. LLS: Cooperative integration of wear-leveling and salvaging for PCM main memory. 2011.

[7] SEONG, Nak Hee; WOO, Dong Hyuk; LEE, Hsien-Hsin S. Security refresh: prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping. In: ACM SIGARCH computer architecture news. ACM, 2010. p. 383-394.

[8] LIU, Duo, et al. Curling-PCM: Application-specific wear leveling for phase change memory based embedded systems. In: Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific. IEEE, 2013. p. 279-284.

[9] CHANG, Hung-Sheng, et al. Marching-based wear-leveling for PCM-based storage systems. ACM Transactions on Design Automation of Electronic Systems (TODAES), 2015, 20.2: 25.

[10] ZHANG, Jiangwei, et al. RETROFIT: Fault-aware Wear Leveling. IEEE Computer Architecture Letters, 2018.

[11] JIANG, Lei, et al. Improving write operations in MLC phase change memory. In: High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on. IEEE, 2012. p. 1-10.

[12] Y. Choi, I. Song, M. H. Park, H. Chung, S. Chang, B. Cho, et al., “A 20nm 1.8 V 8Gb PRAM with 40MB/s program bandwidth,” In SolidState Circuits Conference Digest of Technical Papers (ISSCC), 2012, pp.46-48, February 2012.

[13] PARK, Sang-Hoon, et al. Wear-leveling scheduler for phase-change RAM main memory for mobile consumer electronics. In: Consumer Electronics (ISCE 2014), The 18th IEEE International Symposium on. IEEE, 2014. p. 1-3.