2016 IEEE 22nd International Conference on Parallel and Distributed Systems

Increasing Lifetime and Security of Phase-Change Memory with Endurance Variation

Wen Zhou, Dan Feng, Yu Hua, Jingning Liu*✉, Fangting Huang, Pengfei Zuo
Wuhan National Lab for Optoelectronics, School of Computer Science and Technology
Huazhong University of Science and Technology, China
**Corresponding Author: Jingning Liu*
{*zhouwen, dfeng, csyhua, huangfangting, pfzuo*}*@hust.edu.cn*

***Abstract*—Phase Change Memory (PCM) has emerged as a promising candidate for building the future main memory systems. However, the limited write endurance is one of the major obstacles for PCM to be practically applied. Traditional wear-leveling techniques try to uniformly balance the write traffics under both general applications and malicious attacks to enhance the PCM lifetime. However, these techniques fail to consider the endurance variation in PCM chips, and result in severe lifespan degradation since uniform write distribution leads to the weakest cell to be worn out much earlier.**

In this paper, we propose a weight-based algebraic wear-leveling (WAWL) scheme to balance wear rates (i.e., write traffics/endurance) in a secure manner according to the endurance distribution. In WAWL, the entire memory space is divided into multiple regions. When the number of the writes to a region reaches a threshold (i.e., swapping interval), the region is swapped with a randomly chosen region. The basic idea behind WAWL is that the swapping interval and the chosen probability of each region are variable and associated with the endurance metric of the region. By deploying suitable swapping interval and chosen probability, WAWL achieves uniform wear-rate distribution across the entire memory in an undetectable way. In addition, to reduce space consumption and alleviate performance degradation during region swapping, we propose a fine-grained swapping scheme which migrates the lines one-by-one between the candidate regions. Experimental evaluation driven by the various attacks demonstrates that WAWL significantly increases the PCM lifespan and improves security with slight performance degradation and affordable hardware overhead.

***Keywords*-PCM; wear leveling; endurance variation; lifetime; security.**

I. Introduction

With the increasing number of cores in many-core processors, the applications with large numbers of threads demand a large capacity of the memory system. However, with the increasing size of memory, conventional DRAM technology suffers from large power leakage and poor scalability problems. To this looming crisis, emerging non-volatile memory (NVM) technologies [2][11], such as Spin-Transfer Torque RAM (STT-RAM), Phase-Change Memory (PCM) and Resistive Random-Access Memory (RRAM) have been proposed to alleviate these problems. Among them, PCM is considered as the most promising candidate to replace or supplement DRAM in the main memory [13].

Compared with DRAM, PCM has many prominent advantages, such as non-volatility, superior scalability and lower standby power. Unfortunately, PCM suffers from limited write endurance problem. That means, a memory line is worn out when the accumulated writes exceeds its endurance, and the whole PCM system will fail when there are too many worn-out lines. The poor endurance restricts its system lifetime, exposes security vulnerability, and hinders its practical use as main memory. On one hand, the write traffics of general applications are non-uniform distribution, making the lifetime to be 20X lower than the ideal lifetime [12]. On the other hand, malicious attackers may wear out a few lines and make the whole PCM system fail in a short period [14]. To address the lifetime and security issues, wear-leveling schemes [12][14][16] are proposed to balance the write traffics across the entire memory space to prolong PCM lifetime.

Moreover, the recent studies [3][17] exhibit the endurance variation feature of PCM that the endurance of each cell varies widely. Endurance is severely affected by process variation which derives from two factors: a) the advanced manufacturing process (e.g., sub-20 nm) will unavoidably introduce the variability of physical dimension parameters (e.g., the thickness of the phase change material layer, the height of the heater and the contact size), which impacts the programming current and cell endurance [20]; b) the diverse programming modes of PCM chips (e.g., a cell adopts SLC mode with writing '00' and '11' or MLC mode with writing '01' and '10'), make the cell endurance varies widely [3]. The lifetime problem under endurance variation is more severe, since the weak cells are more easily to be worn out.

Unfortunately, traditional wear-leveling schemes [12][14][21] are not suitable for PCM systems with endurance variation. Since they try to make the number of writes to every line uniform, the weakest cell will fail much earlier, resulting in severe lifespan degradation. A recent study [6] takes the endurance variation into account, and balance the wear rate (i.e., write traffic/endurance) of each block rather than just write traffic. However,

1521-9097/16 $31.00 © 2016 IEEE
DOI 10.1109/ICPADS.2016.114

according to our in-depth analysis (as shown in Section 2) and experimental results, the Wear-Rate Leveling (WRL) scheme [6] fails to resist malicious attacks (e.g., Address Inference Attack (AIA) [18]) due to the existence of a compromised OS and inherent vulnerability of deterministic mapping pattern.

In this paper, we propose a weight-based algebraic wear-leveling (WAWL) scheme to distribute the wear rate in a secure manner according to the endurance distribution. Similar with existing schemes [12][14], WAWL divides the entire memory space into many equal-sized regions. If a region is written certain number of times, it is swapped with another region. The number of writes to trigger region swapping is defined as swapping interval. To ensure security, the region to be swapped is randomly chosen in WAWL. Moreover, considering the endurance variation, WAWL varies the probability of each region to be chosen according to its endurance. The strong regions (i.e., the regions with high endurance) are given a higher probability and more easily to be selected, so that more write operations are shifted to the strong regions and thus the PCM systems achieve uniform wear-rate leveling. Furthermore, WAWL varies the swapping interval of each region according to its endurance to accelerate the swapping of weak regions, so that the writes to the weak regions (i.e., the regions with low endurance) can be quickly shifted out and thus protects PCM systems against malicious attacks. The main contributions of this paper can be summarized as follows:

First, to improve lifetime and security of PCM systems with endurance variation, we propose a weight-based algebraic wear-leveling (WAWL) scheme that carries out a secure region swapping in the hardware layer. The basic idea behind WAWL is that the swapping interval and the chosen probability of each region are variable and associated with the endurance metric of the region. By deploying suitable algebraic configurations, WAWL achieves uniform wear-rate distribution across the entire memory in a random manner.

Second, to reduce space consumption and alleviate performance degradation during region swapping, we propose a fine-grained swapping scheme which adopts an associated algebraic function between two to-be-exchanged regions. Hence, the lines between the two regions can be swapped one by one.

Third, we develop an in-house simulator to evaluate the lifetime of PCM systems with WAWL scheme. Experimental evaluation driven by the Birthday Paradox Attack (BPA) and Address Inference Attack (AIA) demonstrates that, compared with existing algorithms, WAWL significantly increases the PCM lifespan by 7.3X with negligible performance degradation and affordable hardware overheads.

The rest of the paper is organized as follows. Section 2 introduces the background and motivation. The overall design is described in Section 3. Section 4 presents the evaluation results and analysis. Section 5 reviews the related work. We conclude the paper with remarks on future work in Section 6.

II. Background and Motivation

In this section, we present the background on endurance variation and wear-leveling algorithms on PCM to motivate our WAWL design.

A. Endurance Variation Model

The recent surveys [9][20] reveal that the cell endurance of PCM chips is affected by the process variation, such as industrial manufacturing process and diverse programming modes. The process variation leads to the different programming current requirement for each cell. Employing uniform programming current across the chip results in unnecessary over-programming of partial cells, thus incurring severe endurance degradation. Therefore, Zhang et al. [20] propose a PCM model that partitions the entire memory into many domains and uses a fine-grained current budget scheme. The new PCM supplies independent programming current for each domain to mitigate over-programming and thus extends chip lifetime. Further studies show that the minimum current requirement of each domain exhibits more random distribution [8].

The process variation makes the endurance among the domains vary widely. According to the current-energy formula and the power-law relationship [9], the relationship between cell endurance and programming current can be obtained by the equation of $E = 10^8 \times (I^2 \times R \times T)^{-6}$, where E denotes the endurance, I represents the writing current, R indicates the dynamic resistance of the memory cell and T is the applied pulse width. Hence, the endurance (E) is proportional to I^{-12}. According to a large number of experimental statistics [20], the programming current of each domain satisfies the normal distribution with the average current being 0.3mA and the standard deviation being 0.033, as shown in Figure 1(a).

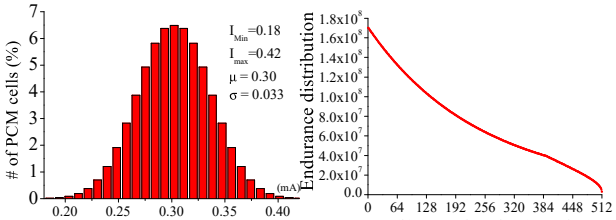


Figure 1: The distribution of minimum required programming current and cell endurance for a PCM chip.

In this paper, we refer to a unified 2GB PCM model that consists of 512 domains, and supplies with independent programming current for each domain. Thus the memory lines within a domain exhibit uniform endurance, while the memory lines among different domains shows huge

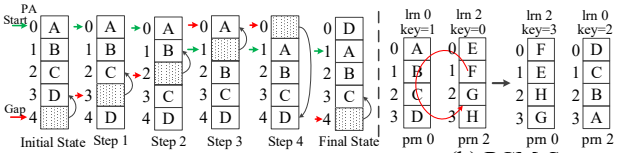
endurance variation [20]. As shown in Figure 1 (b), as the current increases from 0.3mA to 0.42mA, the endurance of each domain varies from 3.0×10^6 to 1.7×10^8. That is, the endurance of the strongest domain is 56 times higher than that of the weakest one.

B. Wear-leveling technique

PCM faces serious lifetime problem because of its limited endurance. For example, a PCM system fails in several seconds under malicious attacks or 3 months under general applications with uneven write distribution [12]. To be practical and commercially viable, PCM systems should provide at least 3-5 years of lifetime under the worst-case scenarios. To address the lifetime problem, wear-leveling schemes [12][14][16][7] have been designed to distribute writes across the entire memory evenly.

Among existing algorithms, the algebraic wear leveling (AWL) is the most efficient approach to improve the PCM lifetime under both malicious attacks and general applications. The representative algorithms include Start-Gap [12], Security Refresh [14] and PCM-S [16]. The two former algorithms remap a given logical address to a new physical address by algebraic mapping functions. Figure 2(a) shows the overview of Start-Gap. After each φ writes, Start-Gap exchanges the contents between the Gap line (i.e., a reserved blank line) and its current neighboring line circularly, i.e., line D (Step 1), line C (Step 2), line B (Step 3), and line A (Step 4). After a complete rotation of the Gap line, all lines in the memory have been moved (i.e., circularly shifted) exactly one position (Final State). Different with Start-Gap, Security Refresh uses dynamically generated random keys and XOR operations to change address mapping in a more unpredictable way to reduce the security vulnerability of Start-Gap. Further, to avoid malicious attacks and ensure sufficient data exchanges, the entire memory space is divided into multiple small-sized regions, and each region performs independent Start-Gap or Security Refresh scheme.

Different with Start-Gap and Security Refresh, PCM-S [16] uses a mapping table to keep track of the mapping relationship between the logical regions and the physical regions, and leverages an algebraic function to obtain the physical location within the physical region for a given logical address. The Figure 2(b) shows an example of the PCM-S algorithm. On certain number of writes to a region, PCM-S swaps all the lines of *prn*0 and *prn*2 in one operation, where *prn*0 and *prn*2 represent two physical regions. In the meantime, the locations of the memory lines within *prn*0 and *prn*2 are exchanged. For example, the line A, B, C, D in *prn*0 are moved to the line D, C, B, A in *prn*2. For address translation, the physical region address can be obtained by looking up the mapping table. In the meantime, the physical address offset (*pao*) can be calculated by the algebraic computation of $pao = lao \oplus key$, where *lao* represents a logical address offset and *key* indicates algebraic parameter.



(a) Start-Gap (b) PCM-S

Figure 2: The respective algebraic wear-leveling schemes.

However, the process variation reveals new challenges, and the future wear-leveling algorithms need to consider endurance variation dimension. The conventional AWL algorithms work well on the PCM systems with unified endurance distribution under both malicious attacks and general applications, which however fail to provide long lifetime on the PCM systems with endurance variation. Because simply balancing the number of writes across the memory space results in wearing out the weak cells much earlier. Thus, the conventional algorithms cannot be applied on the new scenario directly.

Considering the process variation, a recent study [6] propose a Wear-Rate Leveling (WRL) algorithm with software/hardware co-design. This scheme keeps track of the write counts of each domain in hardware layer and accurately exchanges the domains by Max Hyper-weight Rematching algorithm [10] in software layer. WRL minimizes the swapping overhead and maximizes the PCM lifetime. However, the temporal overhead of WRL is extremely high, i.e., $O(n!)$ for sorting and $O(n^3)$ for computation. More seriously, it fails to resist malicious attacks, which is very important for the practical application of PCM. Previous studies [7] have demonstrated that the mapping between logical address and physical address must be randomized to ensure security. Unfortunately, WRL algorithm is deterministic in nature, in which the mapping between logical address and physical address can be guessed easily. Therefore, the algorithm is vulnerable to the address inference attack (AIA) that tracks and keeps writing to a certain domains repeatedly [18]. The attacks are extremely simple to implement and can reduce system lifetime to a few hours under deterministic wear-leveling algorithms.

In summary, existing wear-leveling algorithms cannot address the security vulnerability of PCM systems with endurance variation. The above analysis and observations on existing wear-leveling algorithms motivate us to propose a secure wear-leveling algorithm considering endurance variation, i.e., WAWL. WAWL significantly enhances the PCM lifetime by effectively addressing the uneven wear-rate distribution on the entire space with very low hardware overhead and performance degradation.

III. Design and Implementation of WAWL

In previous sections, we have presented that existing wear-leveling schemes cannot ensure security or achieve high

lifetime for PCM systems with endurance variation. The efficiency of wear-leveling schemes is critical for the application of PCM as part of the main memory. In this section, we propose a novel secure wear-leveling scheme named weight-based algebraic wear-leveling (WAWL) scheme. By exploring suitable swapping interval and chosen probability distribution among the regions, WAWL achieves uniform wear-rate distribution across the entire memory in a secure manner. In what follows, we illustrate our procedure of WAWL in detail.

A. Weight-based algebraic swapping

Similar with conventional AWL schemes [14][16][18], WAWL divides the whole memory space into multiple equal-sized regions and swaps a region with a random chosen one if it is written certain number of times. The number of regions is one of the most critical parameters. On one hand, to achieve wear leveling within the regions, the number of regions should be large enough to ensure sufficient data exchange [12]. On the other hand, increasing the number of regions incurs more spatial overhead for storing address mapping table. In addition, to guarantee all the lines within a region have the same endurance, the region size should be smaller than the size of a domain. How many regions is appropriate for WAWL is empirically studied in Section 4.

Considering the endurance variation among the regions, WAWL varies the chosen probability and the swapping interval of each region associated with its endurance metric to achieve uniform wear-rate distribution.

1) Chosen probability: To prevent the malicious attackers to detect the new physical location of a logical region, the region is swapped with a randomly selected region when being remapped. Considering the endurance variation among the regions, PCM should deploy each region with diverse chosen probability which is associated with its endurance metric. Hence, the strong regions are more easily to be selected, and PCM shifts more writes to the strong regions to balance wear rates among the regions. Since the chosen probability has a great impact on PCM lifetime, WAWL explores a suitable function for the chosen probability according to endurance to achieve wear-rate leveling. To facilitate our presentation of the algorithm, we first define the necessary terms and notations. Assume that the entire memory contains n regions and E_i denotes the intrinsic endurance for physical region prn (i).

Then we show how to choose the to-be-swapped region for a remapped region. We associate a weight (e.g., $\gamma_i = {E_i}^{\alpha}$, where α is an important variable and called *weight factor*, i = 1,2,3,...) with each region, and a region is randomly chosen with the probability of $P_i = \gamma_i / \sum_1^n \gamma_j$. That is, the lower endurance the region has, the less the probability with which it will be chosen is. For the concrete implementation, we first fetch the endurance configuration from PCM cells, and then calculate the accumulated weight of each region (e.g., $w_i = \sum_1^i \gamma_j$, i = 1,2,..,n, and j = 1,2,..,i), and finally store it on the weight-counter registers in memory controller at the boot time. After that, we generate a random number K ($K \in [0, w_n]$), and leverage binary search algorithm to obtain the corresponding index of a physical region. If K belongs to the corresponding space of a region, the region is selected to be exchanged. For example, $w_i \leq K < w_{i+1}$, hence, R_i is picked out.

Finally, WAWL swaps the memory lines within R_i with the candidate region and shifts the location within the regions simultaneously. When the region swapping completes, the mapping table is updated.

2) Swapping interval: When the accumulated number of writes to a region reaches M (i.e., swapping interval), the remapping for this region is triggered. In general, M is set to be proportional to the number of memory lines within a region (K), e.g., $M = K \times \varphi$, where φ is the average swapping interval of each line. As mentioned in previous literature [12], the algebraic wear-leveling scheme must satisfy the *security vulnerability condition*, i.e., $K \times \varphi \ll Endurance$. Otherwise, some memory lines, especially the lines in weak regions, will be worn out in a few round under simple pinpoint attack. To resist the malicious attacks, the weak regions should employ low swapping interval to achieve uniform wear leveling within the regions. Hence, WAWL deploys each region with different swapping interval.

Since the average swapping interval (φ) is restricted by the write endurance (E) and always not continuous in existing AWL algorithms [14][18], we use a ladder-like distribution to express the relation between average swapping interval and write endurance. For example, equation (2) shows the swapping interval of $\varphi(2^{24}, 2^{25}, 2^{26}, 2^{27})$. The example shows that the higher the endurance is, the longer the swapping interval of the region should be. The swapping interval with ladder-like distribution is able to achieve long lifetime and low write overhead simultaneously. In next section, we will discuss the suitable relationship between swapping interval and region endurance to obtain better lifetime.

$$\varphi = \begin{cases} 16 & E < 2^{24} \\ 32 & 2^{24} \leq E < 2^{25} \\ 64 & 2^{25} \leq E < 2^{26} \\ 128 & 2^{26} \leq E < 2^{27} \\ 256 & E \geq 2^{27} \end{cases} \quad (1)$$

3) Address translation: The wear-leveling process makes the mapping between logical regions and its physical locations change dynamically. Hence, an address mapping table is needed to keep track of the logical region addresses and its physical locations. Furthermore, WAWL leverages an algebraic remapping to guarantee uniform write distribution within each region, e.g., WAWL uses a dynamically generated key and XOR operation to change address mapping. In

each round, a region uses different key.

The translation of logical address to physical address for WAWL is obtained by looking up table and computation. For a given logical memory address (*lma*), its physical address is obtained by the following steps:

First, WAWL computes the logical region number (*lrn*) and the logical address offset (*lao*), i.e., $lrn = lma/K$ and $lao = lma\%K$, where K represents the number of lines within a region. Second, WAWL obtains the physical region number (*prn*) by inquiring the address mapping table. In the meantime, the physical address offset (*pao*) is calculated by the formula of $lao \oplus key$. Finally, WAWL obtains the physical memory address (*pma*) by combining the *prn* and *pao* according to the formula: $pma = prn \times K + pao$.

B. Fine-grained swapping scheme

In general, a region consists of many physical lines (e.g., tens of thousands of memory lines), which serves to reduce the spatial overhead of the mapping table. A large number of lines will be exchanged during a region swapping, which requires plenty of space to buffer the exchanged data, e.g., a 2GB bank with 512 regions requires 8MB space. More seriously, the coarse-grained region swapping can severely interfere with normal memory requests because the exchange of lines shares the same resources required by normal memory requests. To reduce space consumption and alleviate performance degradation during region swapping, we propose a fine-grained swapping scheme which migrates the lines one-by-one between the candidate regions.

The key idea behind the swapping scheme is to construct an algebraic mapping function which satisfies two conditions: 1) each line in a region will be remapped to a physical line which is now occupied by a line called its pair. 2) its pair line will be remapped to its old location. Thus, remapping is to swap the data of a line and its pair. Specifically, assume that $P0$ and $P1$ represent the physical locations of $L0$ and $L1$ lines. After region swapping completes, the $L0$ and $L1$ lines are remapped to $P1$ and $P0$, respectively. If the swapping satisfies the algebraic remapping, the lines within the regions can be exchanged one by one. There are many ways to achieve the algebraic function, and Formula (2) shows one of the examples. The detailed proof is shown as follows.

Proof. Given that A_m is a logical memory line in region A, it will be migrated to the physical location which is occupied by B_n in region B in current round. The old physical offset address of A_m is $A_m \oplus key_p$(A) and the new offset address is $A_m \oplus key_c$(A). The old physical offset address of B_n is $B_n \oplus key_p$(B). Since A_m is remapped to B_n, the current physical offset address of A_m equals to the previous physical offset address of B_n, i.e., $B_n \oplus key_p$(B) = $A_m \oplus key_c$(A). By combining with formula (2), the new *pao* of B_n is $B_n \oplus key_c$(B) = $A_m \oplus key_c$(A) $\oplus key_p$(B) $\oplus key_c$(B) = $A_m \oplus key_p$(A). Thus, B_n will be shifted to the previous location of A_m. The A_m and B_n can be exchanged in pair with a fine-grained manner. Only very small space is required to buffer the temporary data.

$$key_c(A) = key_p(A) \oplus key_p(B) \oplus key_c(B) \tag{2}$$

During region swapping, a current swapping pointer (*csp*) is required to record the number of lines that has been swapped. After the migration, the *csp* value will be increased. Data migration will be actively triggered during idle time or forcibly executed when the bank is in busy state.

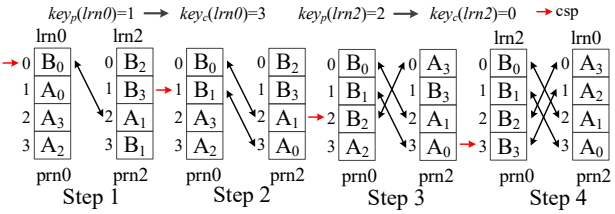


Figure 3: An example of fine-grained swapping scheme.

Figure 3 shows an example of swapping a pair of regions by using fine-grained swapping scheme. In the initial state, the logical regions have remapped to the corresponding physical positions, e.g., $lrn0$ and $lrn2$ are remapped to $prn0$ and $prn2$, respectively. The offsets of the memory lines within the $lrn0$ and $lrn2$ are located to the physical positions with a simple algebraic mapping, e.g., $pao = lao \oplus key$. Assume that the original data of $lao0$ within $lrn0$ is A_0, the current physical offset position of $lao0$ is remapped to $pao3$ based on the calculation of $0 \oplus 3 = 3$. When $lrn0$ and $lrn2$ are beginning to exchange in a new round, new keys $(3,0)$ are generated for $prn0$ and $prn2$ respectively by a hardware random number generator, and *csp* is initialized to $pao0$ in $prn0$. In Step1, the data in $pao0$ in $prn0$ are shifted to $pao2$ in $prn2$. The data in the positions of *csp* and *csp* $\oplus\ key_p(prn0) \oplus key_c(prn0)$ will be swapped in pairs. After each line swapping operation, the content of *csp* is increased to point a new location. Next, the data in B_1 and A_0 (Step2), B_2 and A_3 (Step3), B_3 and A_2 (Step4) are swapped. After *csp* reaches the last line in $prn0$, all the swapping operations are completed. Upon the next swapping, a new key is generated for the to-be-swapped regions.

During the region swapping, the memory controller obstructs the occurrence of other regions' swapping. If there are many urgent requests, the controller will suspend the swapping operations and respond the impending requests. The regions' write count will continuously increase but not exceed the swapping interval (M). When the exchange of region completes, the mapping table is updated and write-count registers are set to zero.

IV. Performance Evaluation

In this section, we present the experimental evaluation for WAWL in terms of lifetime, performance impact and hardware overhead.

A. Evaluation Methodology

In our experiments, we use an in-house simulator to measure the lifetime in a time-efficient way and an open-source Gem5 simulator [1] to evaluate the performance impact of the wear-leveling schemes. The lifetime measure module is able to accurately count the number of writes to each memory line under various configurations. We emulate a 2GB PCM bank with 512 4MB domains, and the endurance distribution of each domain are demonstrated in Section 2. For the ideal situation that all the lines of the PCM are worn out, the lifetime is 11.8 years based on the calculation of $8M \times \frac{1}{512}\sum_{1}^{512} E_i \times (150+450)ns$, where 8M represents the total number of memory lines in the bank, 150ns and 450ns are the latency of read and write operations, respectively. Considering that DCW (data-comparison write) technologies are widely used in PCM to eliminate un-modified writes, we take the sum of read and write latencies as the actual write latency, since DCW technologies requires an extra read before writing. In addition, we use 32MB spare space to tolerate 128K worn-out memory lines to prevent early failures.

To evaluate the effect of wear-leveling algorithms on the security of PCM-based main memory, we use the pinpoint attack for a certain logical memory line to emulate birthday paradox attacks (BPA) [15], which is a standard way of assessing robustness [14][18]. To evaluate the impact of PCM system under general applications on performance, we use the benchmarks from SPEC CPU2006 suite [5], which have been widely used in existing lifetime analysis experiments [12][21].

B. Parameter Training

The key of WAWL scheme is to deploy the suitable configuration parameters (e.g., chosen probability and swapping interval) on the practical systems. In what follows, we obtain the optimal parameters by sensitivity study in a simulation manner.

Figure 4 shows the normalized lifetime (normalized to the ideal lifetime) as a function of the weight factor. As shown in the figure (a) - (d), the weight factor and swapping interval show remarkable impact on PCM lifetime under various region configurations and swapping interval. As the swapping interval decreases or the number of regions increases, the normalized lifetime improves significantly. In Figure 4(a), the best lifetime of PCM systems achieves 59% and 89% when the swapping interval being 256 and 16, respectively. With the number of regions increases, the normalized lifetime grows synchronously. For example, when the number of regions increases to 4096, the normalized lifetime of PCM systems reaches 84% and 93% with the swapping interval being 256 and 16, respectively.

To attain long lifetime, we must use suitable weight factor. When the PCM systems use optimal weight factor, the PCM systems are able to achieve the best lifetime. It is noted that the optimal weight factor is not permanent and always varies under different region configurations. In practical systems, we choose and deploy the optimal weight factor according to the configurations of regions and swapping interval. For example, the appropriate weight factor should be 1.4 under the configuration of 512 regions and the swapping interval being 256, while the value of interval being 1.1 is more suitable with the configuration of 4096 regions.

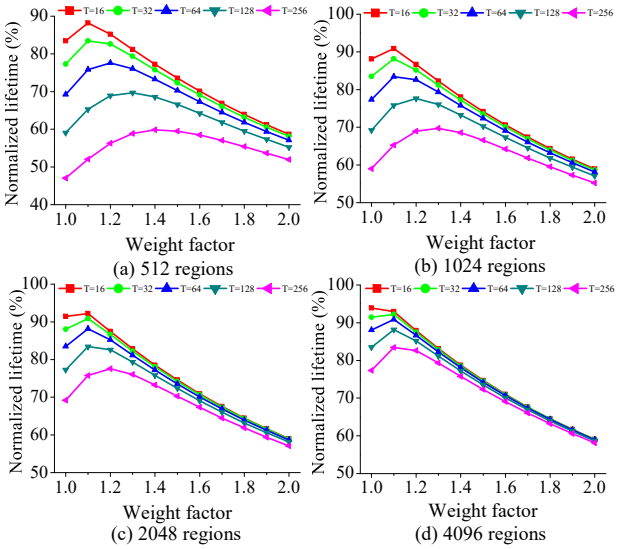


Figure 4: Normalized lifetime of PCM systems as a function of the weight factor.

Figure 5 shows the normalized lifetime as a function of the swapping interval. Fixed-φ (λ) represents a fixed swapping interval of each region being λ, and the Alter-φ (λ) denotes the variable swapping interval of each region, and the maximum swapping interval is λ. The Alter-φ_1 and Alter-φ_2 indicate the $\varphi(2^{23}, 2^{24}, 2^{25}, 2^{26})$ and $\varphi(2^{24}, 2^{25}, 2^{26}, 2^{27})$ respectively. As shown in the figure, when the swapping interval is lesser than 64, the PCM lifetime with variable swapping interval shows slightly improvement compared with the baseline system which uses the fixed swapping interval. However, as the swapping interval exceeds 128, the lifetime of PCM systems with alterable interval is unstable. For example, when the swapping interval being 256, the PCM lifetime with Fixed-φ is higher than that with Alter-φ_2, but lower than that with Alter-φ_1. Because PCM with variable swapping interval excessively wear the memory lines with middle/high endurance. Hence, when the swapping interval exceeds 128, tuning the swapping interval is adventurous, and we do not deploy it on the practical systems.

C. Lifetime Comparison

In order to evaluate the reliability of PCM-based main memory for handling malicious codes, we evaluate the

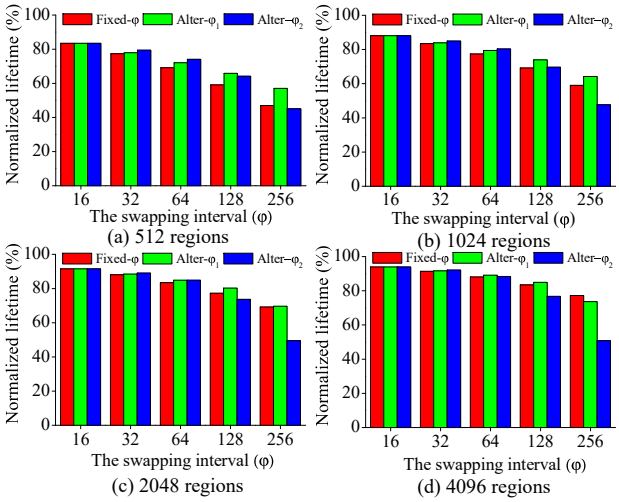


Figure 5: Normalized lifetime of PCM system with the varying of swapping interval.

lifetime of WRL [6], TLSR [14], PCM-S [16] and WAWL for their worst case. For the WRL algorithm, we use Address Inference Attack (AIA) [18] program that compromise the operating system, and thereafter infer the logical blocks that will be subsequently mapped to the same physical block based on the knowledge of the wear-leveling algorithm. For the TLSR, PCM-S and WAWL algorithms that adopt random swapping scheme, the BPA programs are more efficient [18]. For a fair comparison, the swapping interval of various algorithms are all fixed at 128.

Figure 6 shows a comparison among WRL, TLSR, PCM-S and WAWL algorithms in terms of PCM lifetime. As shown in the figure, WRL exhibits severe shortened lifetime regardless of the number of regions varying, e.g., 2.28 months (1.5% of the ideal lifetime). Because AIA is able to wear out a region quickly. As the number of regions increases, the trends of the TLSR, PCM-S and WAWL algorithms vary. For TLSR and PCM-S algorithms, the best lifetime of PCM systems only lasts for 1.43 years (12.1% of the ideal lifetime) when the number of regions is 4096. TLSR and PCM-S result in severe lifespan degradation since uniform write distribution leads to the memory lines in the weakest region to be worn out much earlier. Compared with existing algorithms, WAWL exhibits longer lifetime as the number of regions increases due to evenly disperse wear rates across the entire memory. When the number of regions increases to 4096, WAWL improves the PCM lifetime to 10.04 months (85% of the ideal lifetime), which is 7.3X higher than existing algorithms. With the swapping period being 128, the write overhead of WAWL is only 1.56% (2/128). Overall, the lifetime of PCM system with WAWL scheme is superior to that with WRL and PCM-S under the same configuration.

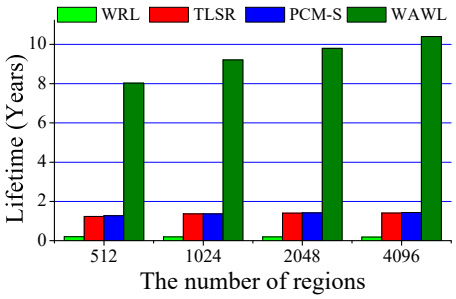


Figure 6: The lifetime of a PCM system with various wear-leveling algorithms under malicious attacks.

D. Performance Impact

We used the Gem5 simulator [1] to evaluate the performance impact of WAWL. In our experiment platform, the system consists of an 8-core processor (3.2 GHz), private 32KB L1 cache, shared 256 KB L2 cache, and an 8 MB L3 DRAM cache. The read and write latencies of PCM are 150ns and 450ns, respectively. We use a queue length of 32 and the FR-FCFS scheduling scheme in memory controller. The latency of address translation requires 3 cycles. We run the 27 SPEC2006 applications and compare the IPC measure with the Baseline (without any wear-leveling scheme). The swapping period of WAWL algorithm is set to 64/128/256. The average IPC measure is decreased by 1.7%, 0.9% and 0.5%, respectively. Some applications, such as the *bzip*2 and *gcc*, show no IPC degradation at all. Because the memory access in these applications are relatively sparse and the swapping requests can be serviced during the idle periods. Compared with PCM-S which adopts coarse-grained swapping scheme, the WAWL is able to quickly serve the normal requests during region swapping. Therefore, the results demonstrate that the performance impact of WAWL is arguably negligible.

E. Hardware overhead

In this subsection, we estimate the hardware overhead of the WAWL algorithm. Given that a PCM system consists of 2^n regions with each containing 2^m lines, and the swapping period is 2^k. The space overhead of address-mapping table will be $2^n \times (m+n)$ bits. The write-counter registers occupy $2^n \times (m+k)$ bits. Moreover, the weight-counter registers take up $2^n \times x$ bits, where x denotes the digits of each weight counter.

For a 2GB PCM system with 512 regions, WAWL uses a 128 swapping interval and 20bits for each weight counter, the spatial overhead of the address-mapping table, write-counter and weight-counter registers are 1.43KB, 1.31KB and 1.25KB, respectively. Thus, the total overhead is 4.0KB. Moreover, as the number of regions increases to 4096, the overall overhead is only 30.5KB. The registers and SRAM resources in memory controller are sufficient for storing these table and counters. Therefore, the WAWL scheme has

affordable hardware overhead.

V. Related Work

Wear-leveling techniques attempt to uniformly distribute write operations on physical devices. The conventional table-based wear-leveling (TBWL) algorithms include Fine-Grained Wear Leveling (FGWL)[13], line swapping [4], row shifting and segment swapping [21]. To achieve long lifetime, the granularity of the mapping unit should be sufficiently small, which incurs huge space overhead. Besides, most of these algorithms adopt a deterministic exchanging policy, leading to severe security vulnerability for malicious attacks. To address these problem, the algebraic wear-leveling (AWL) algorithms, such as randomized region based Start-Gap (RBSG) [12], multi-level Security Refresh (TLSR) [14], PCM-S [15] and MWSR [18] are proposed. The conventional AWL algorithms overcome the space overhead problem of TBWL algorithms, but the lifetimes are shortened for PCM systems with endurance variation because the uniform write distribution incurs the weak cells wear out early.

To address the endurance variation issue, Yun et al. [19] propose a fine-grained wear-leveling method that uses bloom filters to identify hot data and exploits the endurance variation of PCM cells to avoid mapping hot data to weak cells. However, this scheme incurs high storage overhead (e.g., 0.3%). To overcome these problems, our WAWL leverages weight-based algebraic function to remap the logical address to its physical position. By deploying WAWL scheme in PCM systems, the weak cells are rarely to be chosen and the writes to the weak cells are quickly shifted to the strong cells in the memory. Thus the scheme significantly improves PCM lifetime and security.

VI. Conclusion

The uneven endurance distribution feature of PCM exposes new challenge for wear-leveling schemes. In this paper, we propose weight-based algebraic wear-leveling (WAWL) scheme to balance the wear rates in a secure manner according to the endurance distribution. To ensure security, WAWL randomly selects the region to be swapped. Moreover, to achieve better lifetime, for each region, the probability to be chosen and the swapping interval are varied according to its endurance. Experimental evaluations driven by various malicious attacks demonstrate that WAWL significantly improves the PCM lifespan and security with negligible performance degradation and affordable hardware overhead.

Acknowledgment

This work was supported by the National High-tech R & D Program of China (863 Program) No. 2015AA016701, No. 2015AA015301, No. 2013AA013203; NSFC No. 61303046, No. 61402189, No. 61173043; State Key Laboratory of Computer Architecture under Grant CARCH201505. This work was also supported by Key Laboratory of Information Storage System, Ministry of Education, China.

References

[1] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, et al. The gem5 simulator. *ISCA*, 2011.

[2] E. Chen, D. Apalkov, Z. Diao, A. Driskill-Smith, et al. Advances and future prospects of spin-transfer torque random access memory. *IEEE Transactions on Magnetics*, 2010.

[3] X. Dong and Y. Xie. AdaMS: Adaptive MLC/SLC phase-change memory design for file storage. In *ASP-DAC*, 2011.

[4] E. Gal and S. Toledo. Algorithms and data structures for flash memories. *ACM Computing Surveys*, 2005.

[5] D. Gove. CPU2006 working set size. *SIGARCH Computer Architecture News*, 2007.

[6] Y. Han, J. Dong, K. Weng, Y. Wang, and X. Li. Enhanced Wear-Rate Leveling for PRAM Lifetime Improvement Considering Process Variation. *VLSI*, 2016.

[7] F. Huang, D. Feng, W. Xia, W. Zhou, Y. Zhang, and M. Fu. Security RBSG: protecting Phase Change Memory with security-level adjustable dynamic mapping. In *IPDPS*, 2016.

[8] L. Jiang, Y. Du, Y. Zhang, B. R. Childers, and J. Yang. LLS: Cooperative integration of wear-leveling and salvaging for PCM main memory. In *DSN*, 2011.

[9] K. Kim and S. J. Ahn. Reliability investigations for manufacturable high density PRAM. In *International Reliability Physics Symposium*, 2005.

[10] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955.

[11] E. Kültürsay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu. Evaluating STT-RAM as An Energy-Efficient Main Memory Alternative. In *ISPASS*, 2013.

[12] M. K. Qureshi, J. Karidis, M. Franceschini, and V. Srinivasan. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *MICRO*, 2009.

[13] M. K. Qureshi, V. Srinivasan, and J. A. Rivers. Scalable high performance main memory system using phase-change memory technology. *ISCA*, 2009.

[14] N. H. Seong, D. H. Woo, and H.-H. S. Lee. Security Refresh: Prevent Malicious Wear-Out and Increase Durability for Phase-Change Memory with Dynamically Randomized Address Mapping. In *ISCA*, 2010.

[15] A. Seznec. A phase change memory as a secure main memory. *Computer Architecture Letters*, 2010.

[16] A. Seznec. Towards Phase Change Memory as a Secure Main Memory. In *WEST*, 2010.

[17] C. J. Xue, Y. Zhang, Y. Chen, G. Sun, J. J. Yang, and H. Li. Emerging non-volatile memories: opportunities and challenges. In *International Conference on Hardware/Software Codesign and System Synthesis*, 2011.

[18] H. Yu and Y. Du. Increasing Endurance and Security of Phase-Change Memory with Multi-Way Wear-Leveling. *IEEE Transactions on Computers*, 2014.

[19] J. Yun, S. Lee, and S. Yoo. Dynamic wear leveling for phase-change memories with endurance variations. *VLSI*, 2015.

[20] W. Zhang and T. Li. Characterizing and mitigating the impact of process variations on phase change based memory systems. In *MICRO*, 2009.

[21] P. Zhou, B. Zhao, J. Yang, and Y. Zhang. A durable and energy efficient main memory using phase change memory technology. In *ISCA*, 2009.