An Efficient Spare-Line Replacement Scheme to Enhance NVM Security

Jie Xu, Dan Feng, Yu Hua, Fangting Huang, Wen Zhou, Wei Tong and Jingning Liu
Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System (School of Computer Science and Technology, Huazhong University of Science and Technology), Ministry of Education of China
Corresponding author: Dan Feng
{xujie_dsal, dfeng, csyhua, huangfangting, zhouwen, Tongwei, jnliu}@hust.edu.cn

ABSTRACT

Non-volatile memories (NVMs) are vulnerable to serious threat due to the endurance variation. We identify a new type of malicious attack, called Uniform Address Attack (UAA), which performs uniform and sequential writes to each line of the whole memory, and wears out the weaker lines (lines with lower endurance) early. Experimental results show that the lifetime of NVMs under UAA is reduced to 4.1% of the ideal lifetime. To address such attack, we propose a spare-line replacement scheme called Max-WE (Maximize the Weak lines' Endurance). By employing weak-priority and weak-strong-matching strategies for spare-line allocation, Max-WE is able to maximize the number of writes that the weakest lines can endure. Furthermore, Max-WE reduces the storage overhead of the mapping table by 85% through adopting a hybrid spare-line mapping scheme. Experimental results show that Max-WE can improve the lifetime by 9.5X with the spare-line overhead and mapping overhead as 10% and 0.016% of the total space respectively.

1 INTRODUCTION

Emerging non-volatile memory (NVM) technologies, such as Phase Change Memory (PCM) and Resistive RAM (ReRAM), have the disadvantages of limited write endurance [10, 17, 22, 26]. An adversary might write malicious code deliberately to wear out a few memory lines and cause NVMs to fail within a short period of time. Moreover, recent studies [22–25] have revealed that an NVM chip which integrates millions to billions of cells has non-negligible variations in write endurance. Attackers have the potential to accelerate the failure of NVM devices by wearing out the weak cells.

Wear-leveling schemes propose to extend the lifetime by writing uniformly. But the lifetime of NVM under uniform writes is only 4.1% of the ideal lifetime when considering the endurance variation through our experiments. The uniform writes could be an attack, i.e., Uniform Address Attack (UAA) we proposed. Specifically, UAA performs one write operation to each line one by one and repeats such a procedure until many of the memory lines are worn out. The write pattern of UAA does not exhibit the cold/hot characteristic. Existing endurance-variation-aware wear-leveling schemes [5, 21, 25, 27], which try to balance the wear rate by remapping cold/hot data to weak/strong lines, are inefficient to resist the attack of UAA.

To protect NVMs against UAA, we further propose a spare-line replacement scheme called Max-WE (Maximize the Weak lines' Endurance). Max-WE consists of two components: (1) a spare-line allocation component which leverages weak-priority and weak-strong-matching strategies. Weak-priority strategy chooses several weakest regions as the spare regions. Weak-strong-matching strategy uses the strongest spare regions to rescue the weakest regions. Therefore, the number of writes that the weakest lines can endure is balanced and maximized. (2) a hybrid spare-line mapping management component to record the allocation results. By using region-level mapping for the weakest regions and line-level mapping for other regions, the hybrid spare-line mapping scheme can reduce the overhead of the mapping table by 85.0%.

The main contributions of this paper are twofold:

1) **Attack Model**. We propose a new type of malicious attack UAA to NVM device by considering the endurance variation. UAA performs uniform and sequential writes to each line of the whole memory and can invalidate existing wear-out delay techniques. Experimental results show that UAA can wear out an NVM device in only 4.1% of the ideal lifetime.
2) **Endurance Scheme**. To resist UAA, we propose Max-WE. Max-WE maximizes the number of writes that weak lines can endure by employing weak-priority and weak-strong-matching allocation strategies. Weak-priority scheme preferentially chooses several of the weakest regions as spare regions. Weak-strong-matching allocation strategy uses strong regions of spare regions to rescue the weak regions. Max-WE reduces the mapping table overhead by 85.0% by exploiting region-level mapping for the weakest regions. Experimental results show that Max-WE can improve the lifetime by 9.5X.

2 BACKGROUND AND RELATED WORK

In this section, we describe the wear-out problem of NVMs and introduce the existing solutions.

2.1 Endurance Variation of NVM

NVM cells show huge endurance variation due to the variation of advanced process [6, 15, 19]. The endurance distribution parameters can be obtained at the manufacture time. Prior work [24] investigates the endurance variation by dividing the memory space into several equal-size domains and reveals that the programming current of the domains satisfy normal distribution. According to the current-energy formula and the power-law relationship [9, 24, 27],

DAC '19, June 2–6, 2019, Las Vegas, NV, USA

ACM ISBN 978-1-4503-6725-7/19/06…$15.00
https://doi.org/10.1145/3316781.3317767

we obtain the relationship between the endurance and programming current, as shown in Equation 2. $E(I)$ denotes the endurance of the cell when the programming current is I. R is the resistance of the cell. T is the write pulse width. The parameters R and T are constants in Equation 1.

$$E(I) = 10^8 \times (I^2 \times R \times T)^{-6} \tag{1}$$

$$N(I) = \int_{\mu}^{I} (Normpdf(x)dx) \tag{2}$$

We sort the cells by their endurances in descending order. The cell with the programming current I correspond to the Nth number of the descending order. We use $N(I)$ to represent the relationship between N and I. The relationship between $N(I)$ and I is shown in Equation 2, where $Normpdf(x)$ is normal distribution, μ and σ are the expectation and standard deviation of the distribution. By combing Equation 1 and Equation 2, we can get the endurance distribution of the PCM cells.

When considering a $2GB$ PCM divided into 512 domains and with the average current being $0.3mA$ ($\mu = 0.3$) and the standard deviation being 0.033 ($\sigma = 0.033$), the endurance of the strongest domain is $56X$ higher than that of the weakest one. The endurances of PCM and ReRAM are already limited [11, 22, 23] without variation. The endurance distribution makes the lifetime problem more serious because the weakest lines are more likely to fail. NVM cells with low endurance (weak cells) will be worn out early, and NVM device will fail within seconds without protection.

2.2 Related Work

Malicious attackers might wear out a few memory cells and cause the whole PCM device to fail. The existing solutions can be divided into the following categories.

2.2.1 Wear-leveling schemes. The NVM device could fail quickly under wear-leveling techniques without considering the endurance variation (e.g., start-gap [12, 14]). So we mainly discuss endurance-variation-aware wear-leveling schemes. Prior schemes [5, 19, 21, 25, 27] try to balance the wear rate by remapping hot data to strong lines and cold data to weak lines. WAWL [27] associates the chosen probability of each region and the swapping interval with endurance metric of the region. TWL [25] randomly maps writes between two bond blocks (a strong block and a weak block). XML [19] periodically remaps the ReRAM rows that are stressed the most. The endurance-variation-aware wear-leveling schemes are proposed based on the data access locality. If the write patterns do not exhibit the cold/hot property, the remapping schemes will not work. The attack we propose has uniform and sequential write pattern and can invalidate the remapping schemes.

2.2.2 Salvaging techniques. Salvaging techniques [6, 13, 15, 20] correct fault bits with the reserved bits from the fault blocks. They can correct errors in any location, but the error correction rate of each line is limited. For example, ECP [15] can correct six hard failures per line with 11.9% capacity overhead. Hundreds of errors may occur simultaneously in one line, and prior work [13, 15] is incapable to correct so many errors in the attacked line. Free-p [20] and Pay-As-You-Go [13] simply interpret process variation as non-uniform error rate without considering the endurance distribution

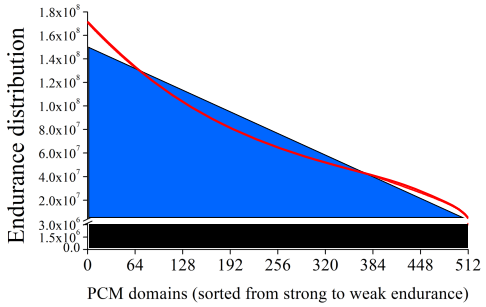

Figure 1: The lifetime analysis of the ideal scenario and under UAA.

of different regions. The endurance differences of spare capacity and working capacity are not distinguished. Our work analyzes the endurance distribution and shows how to select and use the spare capacity based on the endurance variation.

2.2.3 Spare-line replacement schemes. Spare-line replacement techniques automatically remap the failed lines to spare lines. There are two general types of strategies to use spare lines: (1) Physical Capacity Degradation (PCD) for which all the physical lines are initially used and the memory capacity is reduced as failures occur; (2) Physical Sparing (PS) which replaces failed lines with spare lines from the excess capacity. The existing spare-line replacement techniques [4, 8] randomly allocate the spare lines and incur significant mapping table overhead.

3 ATTACK MODEL

3.1 Uniform Address Attack (UAA)

In this section, we present a new type of malicious attack named Uniform Address Attack (UAA). UAA performs one write operation to each line one by one and repeats such a procedure until many of the memory lines are worn out. It should be mentioned that the endurance distribution information can only be obtained at the manufacture time. The attacker is unaware of the endurance distribution. UAA can also work without the endurance information. The lifetime analysis of the ideal scenario and the scenario under UAA is shown in Figure 1. The red color curve in Figure 1 shows the endurance distribution. The ideal lifetime (denoted as L_{Ideal}) can be represented by the area under the endurance distribution curve (highlighted with red color). We approximate the endurance distribution with a tractable linear for which the cells have linearly distributed lifetime between the maximum endurance E_H and the minimum endurance E_L. Therefore, the endurance distribution curve can be approximated to the diagonal which starts from the point of E_H and ends at the point of E_L. The ideal lifetime is approximated to the area under the diagonal. We use N to represent the total number of lines in memory. The ideal lifetime (L_{Ideal}) is shown in Equation 3.

$$L_{Ideal} = N \times (E_H - E_L)/2 + N \times E_L \tag{3}$$

The lifetime under UAA (denoted as L_{UAA}) can be approximated to the area under the horizontal line of E_L, as the black area shows. Hence,

$$L_{UAA} = N \times E_L \tag{4}$$

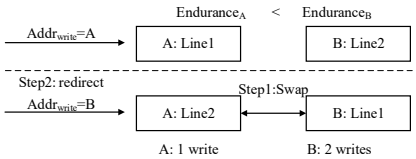

Figure 2: The remapping techniques incur extra writes.

That is, UAA is able to decrease the ideal lifetime by $N \times (E_H - E_L)/2$, which corresponds to the blue triangular area. According to Equations 3 and 4, it's clear that the ratio of the L_{UAA} to L_{Ideal} is

$$\frac{L_{UAA}}{L_{Ideal}} = \frac{2 \times E_L}{E_H + E_L} \tag{5}$$

If E_H is 50 times more than E_L, L_{UAA} will be only 3.9% of the ideal lifetime. When the feature size of manufacture technology decreases, the process variation becomes more serious and the lifetime under UAA decreases significantly.

3.2 Implementation of UAA

Attackers can compromise the operation system by sending malicious codes. We assume that the operating system has been compromised and only the malicious processes are running in the system. The physical main memory is utilized by both the kernel and applications. In a Linux computer with $4GB$ physical main memory, the kernel utilizes only $100MB \sim 200MB$ ($< 5.0\%$) main memory. Malicious application can attack nearly all the physical main memory. To write the physical main memory uniformly, we first use the *malloc* function to allocate the whole main memory. If the user allocates large amount of main memory, operating system will store the unused data into the swap partition temporarily. In the Linux operating system, we can decide when to use the swap partition by setting the value of the parameter "swappiness". To write the main memory as frequently as possible, we set the value of "swappiness" to "0". In this case, the operating system will allocate the swap partition only if the memory usage reaches 100%. Then, we write random data into each address of the allocated space in turn and repeat this write process.

3.3 The Vulnerability of Prior Wear-out Delay Techniques

3.3.1 The Vulnerability of Wear-Leveling Schemes. The idea of endurance-variation-aware wear-leveling schemes is to remap cold (hot) data to weak (strong) lines. Since UAA performs a write to each line for the whole memory every loop, no lines can be identified as hot lines and the remapping scheme will never be triggered. What's more, the remapping operations accelerate the wear-out under UAA, since a remapping operation introduces extra writes to both lines to be remapped. An example to show the vulnerability of remapping scheme is illustrated in Figure 2. The address to be written is A, and the endurance of cells in A is lower than B. The remapping scheme swaps the data stored in A and B, and then the address to be written is redirected to B. The remapping technique aggravates the wear out rather than reduces the writes because it incurs 1 write to A and additional 2 writes to B. Besides, other wear-leveling schemes, which avoid using weak lines when allocating memory [1, 3], cannot work when the operating system has been compromised [16, 18].

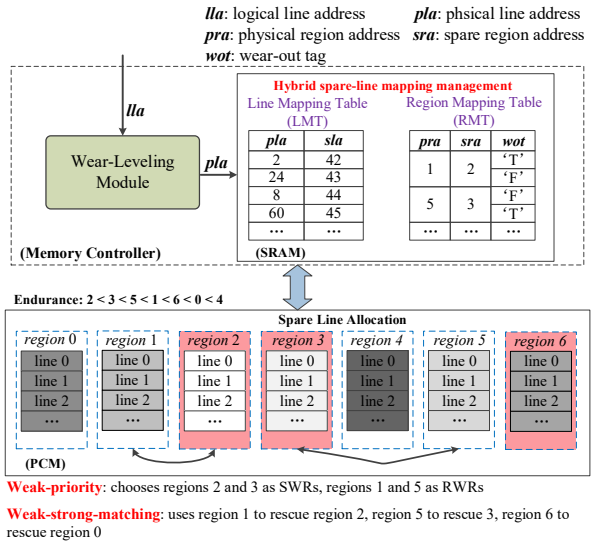

Figure 3: The overview of Max-WE.

3.3.2 The Vulnerability of DRAM Buffer and Write Reduction Techniques. In general, to alleviate the high latency and limited endurance problems of NVM-based main memory, a small-sized off-chip DRAM is used as a last-level buffer. The DRAM buffer is able to cache the hot accessed lines. UAA has uniform write accesses, and therefore the DRAM buffer does not work. Write reduction techniques also suffer from malicious attacks, because an adversary can write specific data to invalidate the techniques. For Flip-N-Write scheme [2], an adversary can always write $0x0000$ and $0x5555$ to the same address in turn. Other encoding methods based on Flip-N-Write [2] or coset coding [7] can also be invalidated by writing specific data patterns. Compression-based techniques are ineffective when writing incompressible data patterns.

4 MAX-WE DETAILS

We describe a novel spare-line replacement scheme called Max-WE to resist malicious attacks and ensure durability.

4.1 The Idea of Max-WE

The real PCM-based main memory may have thousands of regions. We sort the regions by their endurances in advance according to the endurance distribution. Max-WE works based on the endurance distribution. Max-WE consists of a spare-line allocation component which adopts weak-priority and weak-strong-matching strategies, and hybrid spare-line mapping management component to record the allocation results. An example of Max-WE is shown in Figure 3. We assume that the PCM has seven regions. The region IDs arranged in the descending order of the endurance are 2-3-5-1-6-0-4. The sub-schemes of Max-WE work as follows.

Weak-priority scheme preferentially chooses several of the weakest regions as spare regions (denoted as SWRs (Spare Weakest Regions)). SWRs are permanently used to rescue the same number of the weakest regions among the remaining regions (denoted as RWRs (Remaining Weakest Regions)), since RWRs are likely to be worn out early. In this example, Max-WE first chooses the two weakest regions (regions 2 and 3) as SWRs and the two remaining weakest regions (regions 1 and 5) as RWRs.

Weak-strong-matching scheme permanently uses the stronger regions of SWRs to rescue the weaker regions of RWRs. Therefore, the number of writes that each region of RWRs can endure is balanced. Specifically, the regions of SWRs and RWRs are sorted in the descending and the ascending order by the endurance respectively, and are paired in such an order. According to the weak-strong matching principle, the strongest region of RWRs (region 1) is paired with the weakest region of SWRs (region 2), and the weaker region (region 5) is paired with the stronger region (region 3). Furthermore, to address the traditional problem that a few strong lines might be worn out early due to unbalanced wear leveling, a few additional spare regions are used to dynamically rescue the wear-out lines that do not belong to RWRs. The weakest region of the remaining spare region (region 6) is used to rescue all the wear-out lines (region 0) outside the RWRs dynamically.

Max-WE leverages **hybrid spare-line mapping management**, i.e., Region Mapping Table (RMT) and Line Mapping Table (LMT), to track the mapping between wear-out lines and corresponding spare lines. RMT manages the coarse-grained region-level mapping between SWRs and RWRs. To reduce the storage overhead, RMT only records the region id of SWRs and RWRs. Their lines are paired according to the address sequences. Note that since the mapping is generated according to the endurance information once the system is booted and permanent for the whole lifetime, the mapping recorded in RMT will never change. In addition, RMT keeps a wear-out tag per line for the SWRs to indicate whether the line is worn out and replaced by its corresponding spare line. To improve the lifetime of lines outside the RWRs, Max-WE also employs line-level mapping (i.e., LMT) between the wear-out lines (outside the RWRs) and the corresponding spare lines of the additional spare regions. To ensure low address translation latency, RMT and LMT are both stored in SRAM.

4.2 The Implementation of Max-WE

Once a system is booted, the initialized procedure of Max-WE is woke up. Max-WE divides the spare space into two parts: a large portion (> 80%) for rescuing weak regions and a small portion (< 20%) for the lines of other regions. The portions of the regions and lines will be discussed in Section 5.2.2. Max-WE builds the permanent matching relationship for weak regions according to the weak-priority and weak-strong-matching strategies, and records the mapping in RMT. All the wear-out tags in RMT are set to be false initially, which indicates that there are no wear-out lines in the chosen weak regions. The additional spare regions are used to rescue other regions and the LMT is empty.

For a read request, its logical line address is first translated to physical line address by the wear-leveling module, and then translated to the physical line address of its corresponding spare line if the line is worn out and replaced. Max-WE first checks if the physical line address (*pla*) is in the LMT. If *pla* is in LMT, we use the *pla* of the corresponding spare line to access data. If *pla* is not in LMT, Max-WE further checks if its physical region address (*pra*) which is the first several bits of the line address is in the RMT. If *pra* is in RMT, we check the corresponding wear-out tag to see if the line is worn out and replaced; otherwise we use *pla* to access data. If the wear-out tag is true, we replace the *pra* with *sra* and associate *sra*

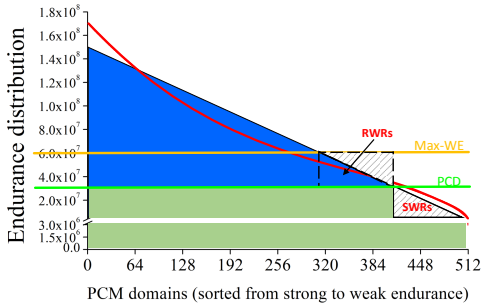

Figure 4: The lifetime analysis of Max-WE under UAA.

with the inter-region address of *pla* to obtain the accessed address; otherwise, the original *pla* is used.

For a write request, the address translation procedure is the same as read request and the data is written to its corresponding spare line if the line is worn out and replaced. When a wear-out failure occurs, the replacement procedure is triggered. If the corresponding region address *ara* of the accessed address is in the *pra* of RMT, the wear-out line is in the RWRs and the line should be replaced by its corresponding line in SWRs. In this case, Max-WE sets its wear-out tag to *true* and builds the mapping relationship between the two lines. Otherwise, Max-WE replaces the line with the available spare line of the additional spare regions. If *ala* is in the LMT, we remove the old entry from LMT before adding a new one. After that, Max-WE allocates the strongest spare line from the additional spare line to replace the wear-out line, and adds the corresponding entry to LMT to record the mapping. If there are no spare lines in the additional spare regions any more, the replacement procedure fails and the whole NVM device is worn out. Note that, once the mapping relationship is built by the RMT or LMT, any further writing to the line will be redirected to its spare line.

4.3 Theoretical Analysis of Max-WE

In this subsection, we analyze the efficiency of Max-WE under UAA, and compared it with other spare line allocation schemes (PCD/PS and the worst case of PS [4]). Based on the analysis of Ferreira et al. [4], the lifetime of PCD and average case of PS for the linear endurance model are very similar. The effect of PCD can be used to approximate the average case of PS. We use PCD/PS to represent the effect of PCD and the average case of PS. We assume that there are S lines to be used as the spare lines. For Max-WE, the weakest S lines are chosen as the spare regions (SWRs), shown in the area marked with the grey shadow in Figure 4. The lifetime of Max-WE under UAA (L_{MaxWE}) can be represented by the product of the endurance of the $(2S+1)^{th}$ weakest line times $N-S$. Hence,

$$L_{Max-WE} = (N-S) \times \{E_L + \frac{2 \times S \times (E_H - E_L)}{N}\} \quad (6)$$

For PCD, the write traffic distributes evenly across the whole memory space (including the spare lines). The lifetime of PCD under UAA (L_{PCD}) can be represented by the area under the green horizontal line and the endurance curve. PCD the and average case of PS have the similar effect (less than 3.0%) [4]. We use PCD to approximate the average case of PS. Thus,

$$L_{PCD/PS} = \frac{S \times (N - S/2) \times (E_H - E_L)}{N} + N \times E_L \quad (7)$$

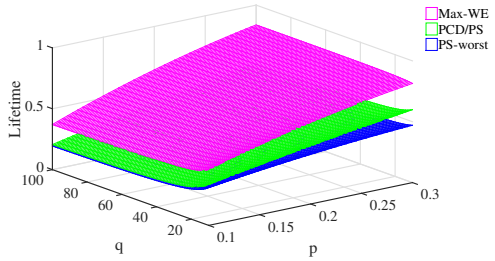

Figure 5: The lifetime comparison of Max-WE, PCD/PS and PS-worst.

For the worst case of PS, S of the strongest $N - 2S$ lines are chosen as the spare lines. Under UAA, the lifetime of the worst case ($L_{PS-worst}$) is determined by the $(S+1)^{th}$ weakest line. Therefore, $L_{PS-worst}$ can be represented by the product of the endurance of the $(S+1)^{th}$ weakest line times $N - S$. Hence,

$$L_{PS-worst} = (N - S) \times \{E_L + \frac{S \times (E_H - E_L)}{N}\} \quad (8)$$

We use p and q to represent the percentage of the spare-line capacity and the degree of process variation (i.e., $p = \frac{S}{N}$, $q = \frac{E_H}{E_L}$). Figure 5 illustrates the lifetime comparison (normalized to the ideal lifetime) of the three schemes ($0.1 \leqslant p \leqslant 0.3$, $10 \leqslant q \leqslant 100$). Max-WE always outperforms both PCD/PS and PS-worst. Assuming that $p = 0.1$ and $q = 50$, Max-WE, PCD/PS and PS-worst can achieve 38.1%, 22.2% and 20.8% of the ideal lifetime.

4.4 Overhead

Line Mapping Table (LMT) and Region Mapping Table (RMT) are used to track the mapping between wear-out lines and corresponding spare lines. We assume that there are N lines and R regions in the memory, S lines are used as spare lines, and q percentage of the spare lines is used as the region-level-mapping SWRs. The storage overhead of LMT and RMT are $(1 - q) \times S \times log_2 N$ bits and $(q \times S \times R \times log_2 R)/N$ bits respectively. Typical mapping table overhead of Max-WE is about $0.16MB$. The detailed calculation of mapping table overhead is shown in Section 5.3.2. The endurance of each region is constant, and it incurs no capacity overhead.

5 PERFORMANCE EVALUATION

5.1 Experiment Setup

We use a $1GB$ NVM bank that consists of 2048 regions and the endurance distribution is obtained according to the model of Zhang et al. [24]. The NVM-based main memory adopts one of the traditional secure wear-leveling schemes (TLSR [16] and PCM-S [18]) or endurance-variation-aware wear-leveling schemes (BWL [21] and WAWL [27]). To prove the efficiency of Max-WE, we use the simulator "NVMsim" to evaluate the lifetime of Max-WE under both UAA and Birthday Paradox Attack (BPA). NVMsim generates the read/write requests according to the attack models, thus avoiding reading memory requests from the workload files. For simplicity, we use normalized lifetime as the metric in our evaluation, which is defined as (the total number of writes before the system fails) / (the sum of the endurance of all memory lines).

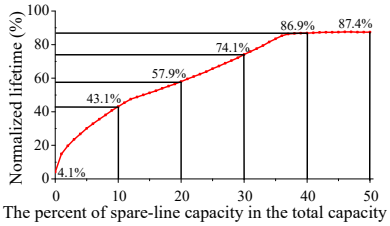

Figure 6: The lifetime of Max-WE with various percentage of spare lines under UAA.

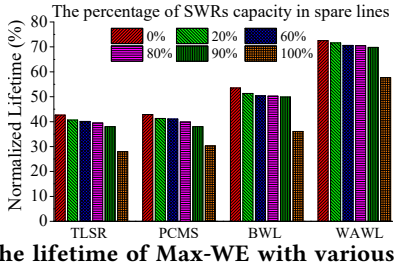

Figure 7: The lifetime of Max-WE with various percentage of SWRs under BPA.

5.2 Parameter Setting

The total capacity is divided into user space and spare lines, and the spare lines are divided into SWRs and additional spare regions. Max-WE needs to determine two important parameters: the percentage of spare-line capacity in the total capacity and the percentage of SWRs capacity in the spare-line capacity.

5.2.1 The Percentage of Spare-Line Capacity. We have demonstrated that the wear-leveling schemes become inefficient due to UAA, and thus the lifetime under UAA is uncorrelated to the types of wear-leveling schemes. In addition, the lifetime under UAA is unrelated to the percentage of SWRs because the spare capacity is constant when the percentage of spare-line capacity is identified. Therefore, we first use UAA to determine the percentage of spare-line capacity. Figure 6 shows the lifetime of NVM with various percentages of spare lines under UAA. The experimental results demonstrate that the lifetime under UAA increases with the capacity of spare lines. However, on the other hand, the available user space is reduced and the mapping table overhead is heavier. As shown in Figure 6, the lifetime under UAA is only 4.1% of the ideal lifetime without any spare lines. The lifetime increases to 14.0% and 43.1% of the ideal lifetime, when the percentage of spare lines increases to 1.0% and 10.0%, respectively. To ensure both security and durability with low overhead, we choose the percentage of spare lines to be 10%.

5.2.2 The Percentage of SWRs Capacity. We use the lifetime under BPA with different wear-leveling schemes to determine the percentage of the SWRs. The lifetime comparison is shown in Figure 7. The experimental results show that when all spare lines are used as the additional spare regions, the system can achieve the highest lifetime, which is 42.7%, 42.8%, 53.5% and 72.5% of the ideal lifetime for TLSR, PCM-S, BWL and WAWL, respectively. This is because the additional spare regions can rescue all the wear-out faults of the whole memory space with line-level mapping, whereas the spare lines of SWRs are permanently used to rescue the corresponding wear-out lines of RWRs with region-level mapping. When 90.0% of

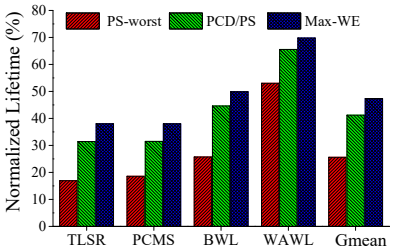

Figure 8: The lifetime comparison of Max-WE, PCD/PS and PS-worst under BPA.

the spare lines are used as SWRs, the lifetime with BWL and WAWL is only reduced by 1.1%. Since line-level mapping consumes much more mapping table storage overhead compared with region-level mapping, we use 90.0% of the spare space as the SWRs.

5.3 The Efficiency of Max-WE

5.3.1 Lifetime Evaluation. The lifetime of PCM will be improved to 43.1%, 30.6% and 28.5% of the ideal lifetime under the protection of Max-WE, PCD/PS and PS-worst when the spare-line capacity is 10.0% of the PCM memory capacity. The improvements of Max-WE, PCD/PS and PS-worst are $9.5X$, $7.4X$ and $6.9X$ compared with the lifetime without protection. Under UAA, Max-WE outperforms PCD/PS and PS-worst with 40.7% and 51.1% lifetime improvement, respectively. Besides, Max-WE can resist other attacks such as BPA. The lifetime comparison under BPA with different wear-leveling schemes and spare-line replacement schemes is shown in Figure 8. Max-WE achieves better lifetime compared with PCD/PS and PS-worst for all the wear-leveling schemes. The geometric means of the lifetime with Max-WE, PCD/PS and PS under BPA are 47.4%, 41.2% and 25.6% of the ideal lifetime, respectively. That is, Max-WE outperforms PCD/PS and PS-worst with 14.8% and 85.0% lifetime improvements, respectively.

5.3.2 Reduction of Mapping Table Overhead. As discussed in Section 4.4, the storage overhead of line-level mapping table and region-level mapping table for Max-WE are $(1-q)\times S\times log_2 N$ bits and $(q\times S\times R\times log_2 R)/N$ bits, respectively. The wear-out tags overhead of Max-WE is $q\times S$ bits. For traditional spare-line replacement schemes (such as PCD) which adopt line-level mapping, the mapping table overhead is $S\times log_2 N$. For a 1GB NVM divided into 2048 regions, assuming that 10.0% of the total capacity is used as spare lines and 90.0% of the spare lines are used as SWRs, the mapping table overhead of Max-WE and line-level mapping are about 0.16MB and 1.1MB, respectively. The mapping table overhead of Max-WE is only 15.0% of the traditional spare-line replacement schemes.

6 CONCLUSION

In this paper, we present UAA, which performs uniform and sequential writes to each line of the memory and can invalidate existing wear-leveling schemes. To resist UAA, we propose Max-WE, which employs weak-priority and weak-strong-matching allocation schemes to maximize the endurance of the weakest lines, and a hybrid mapping scheme to reduce the mapping table overhead. The theoretical analysis and the experimental evaluation demonstrate the feasibility and efficiency of UAA in terms of falling an NVM device and the robustness and lifetime improvements of Max-WE in terms of resisting malicious attacks.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 61821003, Grant 61832007, Grant 61772222, Grant U1705261 and Grant 61772212, in part by the National High Technology Research and Development Program under Grant 2015AA015301, and in part by the Shenzhen Research Funding of Science and Technology under Grant JCYJ20170307172447622. This work was also supported by Key Laboratory of Information Storage System, Ministry of Education, China.

REFERENCES

[1] Chi-Hao Chen et al. 2012. Age-based PCM wear leveling with nearly zero search cost. In *Proceedings of DAC*. ACM.
[2] Sangyeun Cho and Hyunjin Lee. Dec 12-16, 2009. Flip-N-Write: A simple Deterministic Technique to Improve PRAM Write Performance, Energy and Endurance. In *Proceedings of MICRO*.
[3] Gaurav Dhiman et al. 2009. PDRAM: A hybrid PRAM and DRAM main memory system. In *Proceedings of DAC*. IEEE.
[4] Alexandre P Ferreira et al. Mar 4-18, 2011. Impact of Process Variation on Endurance Algorithms for Wear-Prone Memories. In *Proceedings of DATE*.
[5] Yinhe Han et al. 2016. Enhanced Wear-Rate Leveling for PRAM Lifetime Improvement Considering Process Variation. *IEEE Transactions on VLSI Systems* 24, 1 (2016), 92–102.
[6] Engin Ipek et al. Mar 5-11, 2010. Dynamically Replicated Memory: Building Reliable Systems from Nanoscale Resistive Memories. In *Proceedings of ASPLOS*.
[7] Adam N Jacobvitz et al. 2013. Coset coding to extend the lifetime of memory. In *Proceedings of HPCA*.
[8] Lei Jiang et al. Jun 27-30, 2011. LLS: Cooperative Integration of Wear-Leveling and Salvaging for PCM Main Memory. In *Proceedings of DSN*.
[9] Kinarn Kim and Su Jin Ahn. Apr 17-21, 2005. Reliability Investigations for Manufacturable High Density PRAM. In *Proceedings of IRPS*.
[10] Benjamin C Lee et al. Jun 20-24, 2009. Architecting Phase Change Memory as a Scalable DRAM Alternative. In *Proceedings of ISCA*.
[11] Bing Li et al. 2017. Power-Utility-Driven Write Management for MLC PCM. *JETC* 13, 3, Article 50 (April 2017), 22 pages.
[12] Haiyu Mao et al. 2017. Protect non-volatile memory from wear-out attack based on timing difference of row buffer hit/miss. In *Proceedings of DATE*. 1627–1630.
[13] Moinuddin K Qureshi. 2011. Pay-As-You-Go: low-overhead hard-error correction for phase change memories. In *Proceedings of MICRO*. 318–328.
[14] Moinuddin K Qureshi et al. Dec 12-16, 2009. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *Proceedings of MICRO*.
[15] Stuart Schechter et al. Jun 19-23, 2010. Use ECP, not ECC, for Hard Failures in Resistive Memories. In *Proceedings of ISCA*.
[16] Nak Hee Seong et al. Jun 19-23, 2010. Security Refresh: Prevent Malicious Wear-out and Increase Durability for Phase-change Memory with Dynamically Randomized Address Mapping. In *Proceedings of ISCA*.
[17] SeyedMohammad Seyedzadeh et al. 2018. Enabling fine-grain restricted coset coding through word-level compression for pcm. In *Proceedings of HPCA*.
[18] André Seznec. 2009. *Towards phase change memory as a secure main memory*. Ph.D. Dissertation. INRIA Technical Report.
[19] Wen Wen et al. 2018. Wear leveling for crossbar resistive memory. In *Proceedings of DAC*. IEEE, 1–6.
[20] Doe Hyun Yoon et al. Feb 12-16, 2011. FREE-p: Protecting Non-Volatile Memory against both Hard and Soft errors. In *Proceedings of HPCA*.
[21] Joosung Yun et al. 2015. Dynamic Wear Leveling for Phase-change Memories with Endurance Variations. *IEEE Transactions on VLSI Systems* 23, 9 (2015), 1604–1615.
[22] Lunkai Zhang et al. 2016. Mellow writes: Extending lifetime in resistive memories through selective slow write backs. In *Proceedings of ISCA*.
[23] Mingzhe Zhang et al. 2017. Balancing performance and lifetime of MLC PCM by using a region retention monitor. In *Proceedings of HPCA*.
[24] Wangyuan Zhang and Tao Li. Jun 20-24, 2009. Characterizing and Mitigating the Impact of Process Variations on Phase Change based Memory Systems. In *Proceedings of MICRO*.
[25] Xian Zhang and Guangyu Sun. 2017. Toss-up wear leveling: Protecting Phase-Change Memories from inconsistent write patterns. In *Proceedings of DAC*. ACM.
[26] Ping Zhou et al. Jun 20-24, 2009. A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology. In *Proceedings of ISCA*.
[27] Wen Zhou et al. Dec 13-16, 2016. Increasing Lifetime and Security of Phase-Change Memory with Endurance Variation. In *Proceedings of ICPADS*.