Wear Rate Leveling: Lifetime Enhancement of PRAM with Endurance Variation

Jianbo Dong, Lei Zhang, Yinhe Han, Ying Wang, and Xiaowei Li
Key Laboratory of Computer System and Architecture
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
{dongjianbo, zlei, yinhes, wangying2009, lxw}@ict.ac.cn

ABSTRACT

The limited write endurance of phase change random access memory (PRAM) is one of the major obstacles for PRAM-based main memory. Wear leveling techniques were proposed to extend its lifetime by balancing writes traffic. Another important concern that need to be considered is endurance variation in PRAM chips. When different PRAM cells have distinct endurance, balanced writes will result in lifetime degradation due to the weakest cells. Instead of balancing writes traffic, in this paper we propose *wear rate leveling* (WRL), a variant of wear leveling, to balance *wear rates* (i.e., writes traffic/edudrance) of cells across the PRAM chip. After investigating writing behavior of applications and endurance variation, we propose an architecture-level WRL mechanism. Moreover, there is an important tradeoff between endurance improvement and swapping data volume. To co-optimize endurance and swapping, a novel algorithm, *Max Hyper-weight Rematching*, is proposed to maximize PRAM lifetime and minimize performance degradation. Experimental results show 19x endurance improvement to prior Wear Leveling.

Categories and Subject Descriptors

B.3.m [Memory Structure]: Miscellaneous

General Terms

Algorithms

Keywords

PRAM endurance, wear leveling, wear rate leveling

1. INTRODUCTION

Phase change random access memory (PRAM) is considered as one of the most promising candidates for the next generation main memory design, which presents attractive scalability and power efficiency. Extensive research work had been performed on this emerging technology [7] [9] [10] [13]. However, one of the major obstacles for PRAM to replace DRAM as the main memory is its unsatisfactory endurance. Various solutions have been proposed to tackle this problem, which can be generally categorized into two types: *writes reduction* and *wear leveling* [8] [15] [3] [6] [11] [16]. The former tries to reduce the times of writes to PRAM, while the latter tries to balance writes traffic across memory space to avoid excessive wearing of certain locations.

Another important concern that need to be considered is parameter variation. Since PRAM cells require different minimum RESET and SET currents due to process variations, employing uniform programming current across the chip will result in unnecessary over-programming of partial cells, thus endurance degradation [5]. Therefore, the PRAM design as in [4] divides a chip into domains, which are supplied with independent programming currents. Such kind of variation-aware PRAM chips demonstrate endurance variation across the device.

When PRAM cells are identified as strong and weak according to their endurance characteristics, however, prior wear leveling mechanism can not work well any more, since balanced write traffic to PRAM will wear out weak parts much earlier than strong ones. This inspires us to investigate new wear leveling solutions, considering both endurance variations of PRAM and writing behavior of applications. Though the major source of lifetime degradation is RESET current, we assume the possibilities of writing "0" and "1" are statistically equal. Thus we propose wear rate leveling mechanism to balance wear rate, i.e., *write traffic/endurance*, through physical address remapping and data swapping.

The two major components of wear rate leveling are runtime writing traffic prediction and an effective remapping algorithm. To understand writing behavior of applications, massive memory reference traces of real system were analyzed. We show that applications' writes traffic are predictable, if sufficient writes were tracked in advance. And, we assume the endurance of different domains in the PRAM can be obtained via manufacturing testing procedures. With both applications' writing behavior and PRAM's endurance distribution information in hand, an intuitive solution is to migrate the hot/cold data to the strong/weak domains.

However, the wear rate leveling algorithm design faces an important tradeoff between endurance enhancement and swapping data volume. Though the "hot/cold-to-strong/weak" method can achieve optimal PRAM lifetime, however, it will incur great amount of memory data swapping and cause performance overhead for applications. The co-optimization of PRAM lifetime and memory data migration overhead can

This work was supported in part by National Basic Research Program of China(973) under grant No. 2011CB302503,in part by NSFC under grant No. (60806014, 61076037, 60921002, 60906018).

DAC 2011, June 5-10, 2011, San Diego, California, USA.

be formulated as a multi-objectives 0-1 programming problem. However, the complexity and huge problem size (due to large amount of on-chip domains) motive us to find a satisfactory solution at runtime. The major contribution of proposed algorithm, *Max Hyper-weight Rematching*, is to translate the above problem to a max weight perfect matching problem by introducing a new compound objective, i.e., *hyper-weight*. Wear rate leveling can efficiently achieve optimal PRAM lifetime and minimal data migration.

The rest of the paper is organized as follows. We present the related work in Section II, and investigate the writing behavior and endurance variation in Section III. Section IV and V show our proposed wear rate leveling mechanism and algorithm. Thereafter, experimental results are shown in Section VI. And Section VII concludes this paper.

2. RELATED WORK

Writes reduction techniques include Lazy Write, Line Level Write-back, and Page Level Bypass [15], Partial Writes [8], Redundant Bit-write Removal [6], and so on. Lazy Write, Line Level Write-back, and Page Level Bypass take advantages of DRAM buffer to reduce write traffic to PRAM. Partial Writes reduces write accesses to memory by tracking cache modification and writing only dirty lines. Redundant Bit-write Removal was proposed to reduce write frequency to PRAM only if stored value is different with writing value.

Wear leveling techniques include Start-Gap [3], Row shifting & Segment swapping [6], Fine Grained Wear Leveling [15], PDRAM [11], and so forth. As the distribution of write traffic to PRAM is usually non-uniform, Start-Gap and Row shifting & Segment swapping strive to balance writes across the entire range of memory, and Fine Grained Wear Leveling tries to balance writes across a page. While PDRAM proposed a hybrid memory and tries to swap hot pages from PRAM to DRAM by OS-level page migration. Though these techniques effectively extended PRAM lifetime, the variation effects on PRAM was not taken into concern.

In [4], the authors thoroughly analyzed the sources of variation and their impact in PRAM. The variations of each source are modeled individually, including systematic and random. Putting these factors into a one-dimensional heat conduction model, the programming current variations were generated. The endurance variations were then estimated using the power law relationship between programming pulse energy and endurance of PRAM cells [5].

Having obtained endurance variation, adaptive write reduction techniques, including adaptive data comparison write and adaptive memory compression, were proposed in [4] to extend PRAM lifetime. The write reduction techniques were enabled in weak regions, while disabled in strong regions. Thus, the lifetime of PRAM chip was extended, with less write reduction induced performance overhead.

[3] and [4] are the most related studies to this paper. Our work differs from the [3] by taking endurance variation into concern. And we propose to extend PRAM lifetime through wear rate leveling, instead of writes reduction in [4].

3. UNDERSTANDING WRITING BEHAVIOR AND ENDURANCE VARIATION

As it is known, applications' memory accesses present temporal and space locality. It is expected that the former accessed data will be re-accessed later. Thus, the memory space can be classified into hot and cold domains, and the current "temperature" indicates the future "temperature".

In this section, we use the *Hybrid Hardware/Software Memory Trace Tool* (HMTT) [1] to monitor the memory accesses of real system. The kernel component of HMTT is the Memory Trace Board (FPGA) plugged into a DIMM slot, through which we can track the memory trace of full system, including OS, libraries and applications. Thus the information of memory traces we obtained, including memory physical address, write/read and time stamp, are complete and undistorted writing behavior of real system, which is more accurate than the results of simulation and binary instrument. The detailed system configuration is presented in table 1.

Table 1: System configuration

Platform	X86/Linux
Processor	Intel E4500, dual-core, 2.2GHz
L1 I/D-cache	32KB, 8way associate
L2 cache	2048KB, 8way associate
Memory	2GB, DDR2, 333MHz

Table 2: Benchmarks Tracking Periods

perlbench	112.9s	17.5%	bzip2	105.5s	10.4%
gcc	101.0s	11.1%	leslie3d	63.2s	4.1%
gobmk	418.6s	49.1%	dealII	220.1s	31.9%
soplex	64.0s	6.0%	calculix	1242.4s	60.1%
GemsFDTD	57.6s	3.2%	libquantum	54.5s	2.6%
tonto	251.8s	22.8%	lbm	58.8s	10.0%
omnetpp	70.5s	11.8%	sphinx3	69.0s	5.2%
average	206.4s	17.6%			

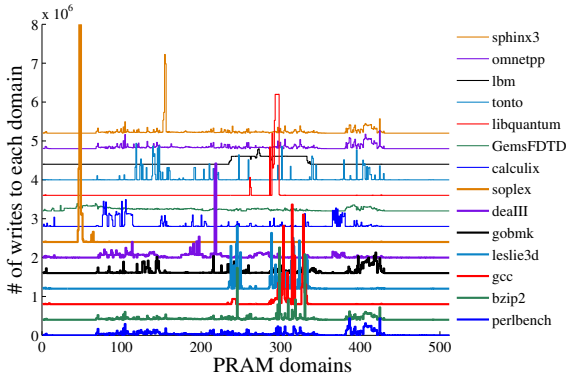

Figure 1: The forepart writes distribution.

Running 14 SPEC2006 benchmarks, we record 3×10^9 memory accesses for each application because of storage limitation and massive dumped traces (16GB each). The tracking periods and percentage of entire applications are presented in table 2. Because the number of writes in different applications ranges from 2×10^8 to 1.2×10^9, we choose the aligned 2×10^8 for evaluation. Stepping the number of forepart writes from 1×10^7 to 5×10^7, we present that 2×10^7 is suitable for writes distribution prediction (evaluation in Section VI). For each application, the first 2×10^7 writes are

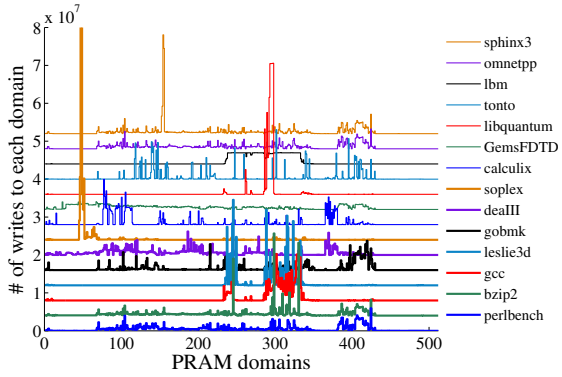

Figure 2: The total writes distribution.

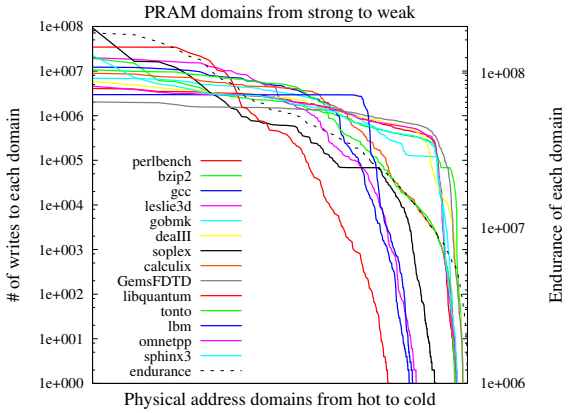

Figure 3: Similarity of PRAM endurance and write pattern.

shown in figure 1, and the total 2×10^8 writes are shown in figure 2. We present write counts in vertical axis, and separate the curves of applications from each other for clarity. The horizontal axis indicates memory physical address from low to high, which is divided into 512 domains. As shown, for all the application, the corresponding curves in figure 1 and figure 2 show certain similarity because of applications's temporal locality. That is, we can use the distribution of few writes to predict the following.

Having obtained the write traffic distribution, we need variation model to analyze the relationship between write traffic non-uniformity and endurance variation of PRAM. In [4], the authors thoroughly analyzed the effects of virous parameter variations on PRAM programming current. The systematic variation was simulated using quad-tree model [14], and the random variation was simulated with the same standard deviation as the systematic, i.e., $\sigma_{rand} = \sigma_{sys} = \sigma/\sqrt{2}$. Putting these factors into an one-dimensional heat conduction model, the minimum RESET programming current presented 40% variation, i.e., a normal distribution with μ=0.3mA and σ/μ=11%. Taking advantages of the variation model, they generate 2GB PRAM chips comprised of 512 domains (4MB) with separate programming currents. Thus, the corresponding endurance can be estimated using the power law relationship between programming pulse energy and endurance of PRAM cells [5]. The analysis and evaluations of endurance variation are based on such a PRAM chip, in this paper.

The curves in figure 2 show that writes non-uniformity results in hot and cold domain in physical address space. Ranking them from hot to cold, the curves in figure 2 are re-plotted in figure 3 in logarithmic scale, as the solid curves indicated. For comparison, the endurance distribution of PRAM is also plotted in figure 3 after ranking, as the dashed line indicated. Apparently, systematic variation results in strong and weak domains within PRAM chips, ranging from 2×10^6 to 3×10^8. Thus, balanced writes will result in endurance degradation because the weakest cell retires earlier. We observe that the solid lines present similar trend to the dashed curve, which provides the opportunity of wear rate leveling via physical address remapping.

4. WEAR RATE LEVELING FRAMEWORK

Though there exists intra-domain writes non-uniformity, it not the focus of this paper. That is because the PRAM cells in the same domain were provided with a uniform programming current, and present no endurance variation. Thus it can be tackled by wear leveling techniques, e.g. Start-Gap [3] and Fine Grained Wear Leveling [15].

We introduce a *Remapping Table* in memory controller to translate physical address to real address. The table is indexed by the physical domains address (PA domains), i.e., the higher bits of physical address. The corresponding entries store the real domains addresses (RA domains), which are used for PRAM access. When data in PRAM are swapped, we need only to update the corresponding entries in Remapping Table. Since Remapping Table updating is transparent to TLB and caches, there is no incoherence problem that OS-level page migration faces. Moreover, Remapping Table accessing latency is negligible in two aspects. First it needs no searching and comparison. Second, memory requests usually wait in scheduler for long time, which is enough for address translation.

To provide endurance and writes traffic distribution to *Max Hyper-weight Rematching* algorithm, we need a *Write Counts Table* in memory controller and a *Endurance Records Table* in PRAM chip. The Write Counts Table is also indexed by domain address. It increases the corresponding entry on receiving a write request, and right-shifts all the entries on overflowing. We assume the Endurance Records Table is initialized according to the programming currents of domains in manufacturing phases.

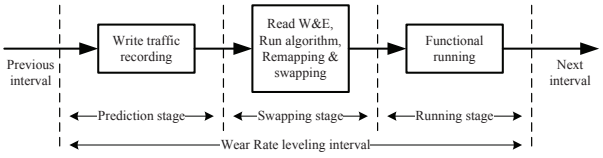

Figure 4: Working mechanism overview.

The overview of wear leveling in this paper is shown in figure 4. It is triggered periodically according to the write counts into memory, which is recorded in a Memory Writes Register. The predefined write counts threshold divides execution flow into wear leveling intervals. Based on the op-

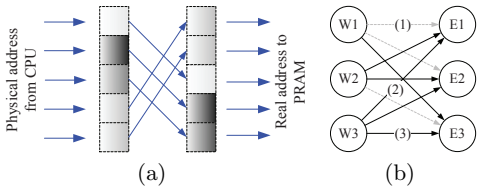

Figure 5: (a) Physical/Real address remapping; (b) hyper-weights assignment.

erations differences, a wear leveling interval can be further divided into 3 stages, i.e., prediction, swapping and running stage. We records write counts of each domain in the prediction stage. This stage finishes after certain number of writes, which is also predefined. A remapping interrupt is then issued. The swapping stage starts when OS receives the remapping interrupt. It then reads the writes and endurance of domains from Write Counts Table and Endurance Records Table, and calls the *Max Hyper-weight Rematching* algorithm to update the Remapping Table. At the same time, it sends the request to swap the data in memory according to the remapping results. Thus, the writing behavior in the following running stage is optimized. When the writes number equals the predefined threshold, another wear leveling interval starts and Write Counts Table entries are cleared for recounting.

Apparently, data swapping is the major source of performance overhead. We will explore the tradeoff of endurance and swapping and co-optimize them in next section.

5. WEAR RATE LEVELING ALGORITHM

5.1 Baseline HC-to-SW Algorithm

As mentioned before, wear rate leveling tries to balance the wear rates of PRAM domains. Suppose there are n RA Domains and n PA Domains. Arrays W and E indicate the write counts to PA Domains and endurance of RA Domains. We have, $W = \{W_1, W_2, ..., W_n\}$ and $E = \{E_1, E_2, ..., E_n\}$, where $< W_i, E_i >$ represents a mapping from PA Domain to RA Domain. So W_i/E_i indicates the wear rate of domain i. To balance the wear rates across memory, we should minimize the maximum wear rate of all the domains. The wear rate leveling problem can be expressed as:

$$\begin{aligned} &object: \quad min\ max\{\tfrac{W_{i1}}{E_{j1}}, \tfrac{W_{i2}}{E_{j2}}, ..., \tfrac{W_{in}}{E_{jn}}\}; \\ &s.t.: \\ &\qquad i1 \neq i2 \neq ... \neq in; \\ &\qquad j1 \neq j2 \neq ... \neq jn; \\ &\qquad 1 \leq i1, i2, ..., in \leq n; \\ &\qquad 1 \leq j1, j2, ..., jn \leq n. \end{aligned} \tag{1}$$

According to the intuitive "hot/cold-to-strong/weak" remapping strategy, we compose the *HC-to-SW* algorithm as follows.

1) Sort the arrays W and E in descent order; we have the sorted arrays, W' and E'.

2) Remap domain W'_i to E'_i, where $1 \leq i \leq n$.

The solution can be proved to be optimal in terms of endurance by contradiction. Suppose W_i/E_i equals the optimal wear rate. The only way to decrease the value of W_i/E_i is swapping W_i with lower value, i.e., W_{i+1}, W_{i+2}, ... , or W_n. Suppose W_i is swapped with W_j, where $i < j \leq n$, we have $W_i > W_j$, and $E_i > E_j$. The resulted wear rate of physical domains i and j are W_j/E_i and W_i/E_j respectively. Though the wear rate of domain i is decreased, the wear rate of domain j is much enlarged. Since $W_i/E_j > W_i/E_i$, the optimal wear rate is increased to W_i/E_j. We can conclude that every swapping with W_i will cause increment in the optimal wear rate. That is, the original solution is optimal.

5.2 Max Hyper-weight Rematching Algorithm

Though the *HC-to-SW* algorithm achieves optimal endurance easily, it migrates almost all the domains in PRAM as shown in next section. The swapping data volume should be minimized, since it is crucial to performance overhead. To co-optimize endurance and swapping, the wear leveling problem is much complicated.

We introduce two $n \times n$ matrixes, O and N, to indicate the old and new mapping. O_{ij} and N_{ij} is 1 or 0, indicating the ith PA domains is mapped to the jth RA domain or not.

$$\begin{aligned} &objects: \quad min\ max\{\tfrac{W_{i1}}{E_{j1}}, \tfrac{W_{i1}}{E_{j1}}, ..., \tfrac{W_{in}}{E_{jn}}\}; \\ &\qquad\qquad max \sum O_{ij} \times N_{ij}. \\ &s.t.: \\ &\qquad \textstyle\sum_i O_{ij} = \sum_i N_{ij} = 1; \\ &\qquad \textstyle\sum_j O_{ij} = \sum_j N_{ij} = 1; \\ &\qquad O_{ij} \in \{0,1\};\ N_{ij} \in \{0,1\}. \end{aligned} \tag{2}$$

It seems that the multi-objectives 0-1 programming problem have a complexity of $O(n!)$. However, we show that the problem can be solved more efficiently, without accuracy loss. Apparently, every mapping from PA Domains to RA Domains is a perfect matching in the bipartite graph G composed of W and E, as shown in figure 5(b). We assign a *Hyper-weight* to edge $< W_i, E_j >$, according to the wear rate W_i/E_j and whether the edge is in original mapping.

To prevent endurance loss, we run the *HC-to-SW* algorithm to obtain the optimal wear rate. For the edges whose wear rate is over optimal wear rate, we set the corresponding hyper-weight to 0. Taking swapping into consideration, hyper-weights assignment should show preference of original edges, and present the number of original edges at the same time. Thus, we assign weights to all the edges in this way: *hyper-weight=n+1*, if in the original mapping; *hyper-weight=n*, if not.

Thus we classified the all edges in bipartite into 3 categories with corresponding hyper-weights, as shown in figure 5(b). To achieve maximum endurance and minimum swapping, we needs only to maximize the sum of hyper-weights of all edges. We then translate the problem into the *Maximum Weight Perfect Matching Problem* in bipartite graph, which can be solved using *Hungarian Algorithm* [2].

For each matching, if the sum of weights s is over n^2, the result indicates a mapping with $s - n^2$ original edges. While if the sum of weights s is below n^2, it means at least one over-wear edge is included in the result, which causes optimal wear rate increment. However, this could never happen, because the result of *HC-to-SW* algorithm is one of feasible matching with optimal wear rate in G.

The *Max Hyper-weight Rematching Algorithm* can be expressed formally as algorithm 1. Note that, by adjusting the threshold of wear rate, the endurance constraint can be loosed to investigate the tradeoff between endurance and swapping. It presents the complexity of $O(n^3)$, which is determined by the complexity of *Hungarian Algorithm*.

6. EVALUATIONS

Algorithm 1: Max Hyper-weight Rematching

Input: Writes and endurance distribution W and E for n domains
Output: Mapping with minimum wear rate and minimum data swapping

Compose bipartite G, with W and E;
Calculate wear rate of edges, W_i/E_j;
Run HC-to-SW to obtain the optimal wear rate;
Set weights for edges:
  (1) *hyper-weight=0*, if wear rate is over the optimal;
  (2) *hyper-weight=n+1*, if in original mapping;
  (3) *hyper-weight=n*, if not;
Run Hungarian algorithm to solve the Maximum Weight Perfect Matching Problem in G;

6.1 Endurance Enhancement

We present the effects of prediction stage length on wear rate in figure 6. The horizontal axis indicates the prediction stages length, in which n means $n \times 10^7$ writes. The vertical axis shows the endurance improvement of wear rate leveling, when compared with *Base* and *WL*, indicating the cases without and with wear leveling respectively. It presents that 1×10^7 is immature for prediction. While mature predictions show similar endurance improvement. Since shorter prediction period will benefit more write accesses, we use 2×10^7 writes for prediction in this paper.

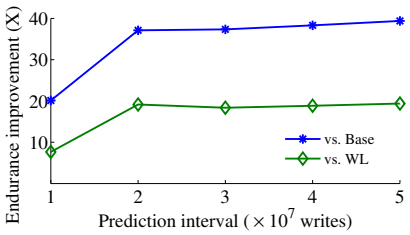

Figure 6: Prediction stage effects on endurance.

Figure 7 shows the wear rates of different benchmarks. The value of wear rates are amplified to plot them positively in logarithm scale vertical axis. Note that, the *WL* here means uniform writes distribution in PRAM chip, which is the ideal result of wear leveling. On average, *WRL* shows about 37x and 19x wear rate reduction when compared with *Base* and *WL*, as the last group bars show.

When compared with *Base*, effectiveness of wear rate leveling is highly related to the intrinsic writes distribution. For the applications whose writes is intrinsically balanced, e.g., *GemsFDTD* shown in figure 1 and 2, the proposed algorithm achieves less endurance improvement. While for the applications that show concentrated write traffic, e.g., *soplex*, the endurance improvement is much higher.

When compared with *WL*, effectiveness of wear rate leveling is concerned with prediction accuracy of write distribution. For example, *gcc* and *deaIII* present lower endurance improvement, though write traffics are not balanced. Thus, more accurate prediction techniques will further improve the effects of our proposed wear rate leveling mechanism.

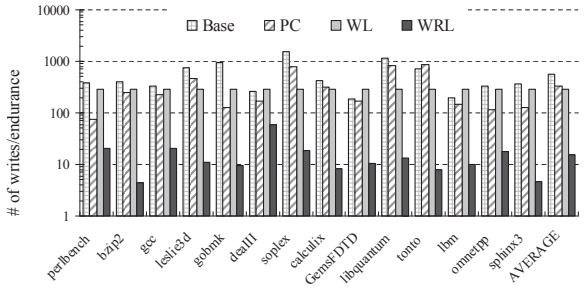

Figure 7: Wear rate reduction.

OS-level page classification [4] tries to migrate hot/cold pages to strong/weak domains. However, the threshold for hot/cold pages identification was not presented. We set the threshold as 20% for evaluation. The resulted endurance is shown as *PC*. The endurance of *PC* is suboptimal, since the 60% un-migrated pages limited maximum wear rate. Note that, increasing the threshold will result in higher endurance and more swapping. And *HC-to-SW* can be regarded as an extreme case of page classification.

6.2 Tradeoff between Endurance and Swapping

Though data swapping is inevitable for wear leveling, we can resort to smart algorithm to reduce swapping data volume. The swapping data volume comparison between *HC-to-SW* and *Max Hyper-weight Rematching* algorithm is shown in figure 8. It clearly shows that *HC-to-SW* algorithm migrates almost all the domains, because of the strict constraint of ranking. While *Max Hyper-weight Rematching* reduces the swapping data volume significantly, without endurance loss. The swapping reduction is 68% on average, ranging from 37% to 98%.

Prior research [6] presented that reading and writing a row (1KB) took at most 36.28ns and 120.27ns respectively. With the throughput 1KB/156.55ns (about 6.5GB/s), we present the performance overheads of *Max Hyper-weight Rematching* and *HC-to-SW* in figure 9. Obviously, *Max Hyper-weight Rematching* is superior to *HC-to-SW* for all the applications, and presents no more than 2.4% overhead in any case. For the most memory-intensive applications, *lbm*, *HC-to-SW* results in 9.2% performance degradation, while *Max Hyper-weight Rematching* results in only 2% of overhead.

Moreover, performance overhead can be further mitigated by relaxing the endurance constraint. Figure 10 presents the tradeoff between data swapping and endurance. The horizontal axis shows the relaxed endurance constraints, in which n means the upper bound of wear rate is n times of the optimal. And the vertical axis shows the number of swapped domains. As endurance constraint relaxes, the average swapping data volume, as the wide blue line shown, decreases sharply at beginning, while slightly later. On average, swapping volume is decreased to 19.5% and 14% when endurance constraint is relaxed to 2x and 3x respectively.

6.3 Discusses

The hardware overheads of wear rate leveling include a Remapping Table and a Write Counts Table in memory controller, a Endurance Recording Table and a spare domains as swapping buffer in PRAM chip, and a Writes Record-

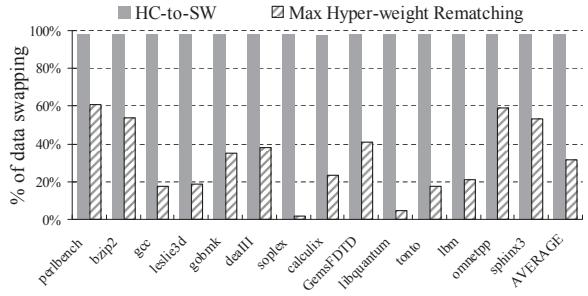

Figure 8: Comparison of data swapping.

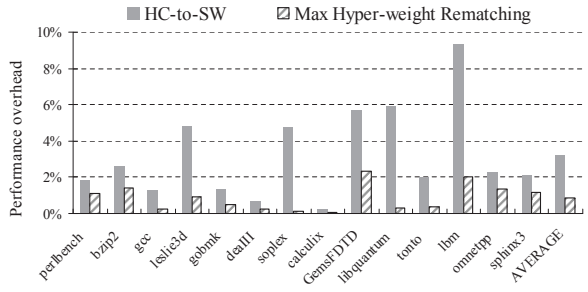

Figure 9: Comparison of performance overhead.

ing Register. With 23bits Write Counts Table entry and 9bits Remampping Table entry, the two 512-entries tables in memory controller introduce only 2KB space overhead. With 64bit entries, Endurance Recording Table introduce 4KB extra memory space. Thus, the overhead of spare domain (4MB) and the Endurance Recording Table in PRAM chip is acceptable in large memory chips.

Though *Max Hyper-weight Rematching* brings one extra write to swapped domains every wear rate leveling interval, the rarely triggered operation presents no endurance concerns on the spare domain and no visible endurance overhead on PRAM chip.

7. CONCLUSION

In this paper, we propose wear rate leveling mechanism to enhance PRAM lifetime, by balancing wear rate across the entire memory. Tracking SPEC2006 benchmarks' memory accesses, we show that an application's write distribution to PRAM can be predicated by tracking its forepart writing behavior. Thus the prediction-based wear rate leveling mechanism was proposed. *Max Hyper-weight Rematching* algorithm not only results in 19x endurance improvement when compared with prior variation-unaware wear leveling, but also minimizes data swapping volume.

8. ACKNOWLEDGMENTS

We would like to thank Prof. Dongbo Bu for his suggestions in algorithm design, Prof. Ninghui Sun for his insightful discussion to our work, Dr. Guihai Yan for his long time help. Correspondence of this work should be addressed to Yinhe Han. We also thank the anonymous reviewers for constructive comments.

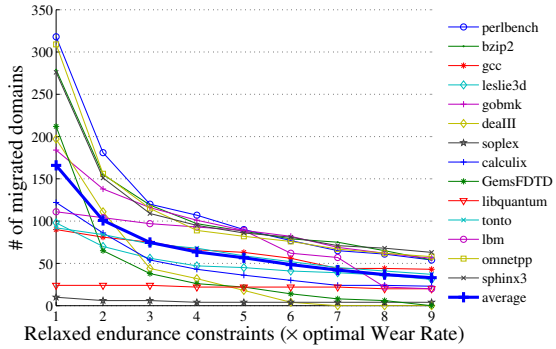

Figure 10: Tradeoff between endurance and data swapping.

9. REFERENCES

[1] Y. Bao, M. Chen, Y. Ruan, L. Liu, J. Fan, Q. Yuan, B. Song, J. Xu, "HMTT: A Platform Independent Full-System Memory Trace Monitoring System", *SIGMETRICS*, pages 229-240, 2008.

[2] H. W. Kuhn, "The Hungarian Method for the assignment problem", *Naval Research Logistics Quarterly*, 2:83-97, 1955.

[3] M. K. Qureshi, J. Karidis, M. Franceschini, "Enhancing lifetime and Security of PCM-Based Main Memory with Start-Gap Wear Leveling", *MICRO*, pages 14-23, 2009.

[4] W. Zhang and T. Li, "Characterizing and Mitigating the Impact of Process Variations on Phase Change based Memory Systems", *MICRO*, pages 2-13, 2009.

[5] K. Kim and S. J. Ahn, "Reliability Investigation for Manufacturable High Density PRAM", *IRPS*, pages 157-62, 2005.

[6] P. Zhou, B. Zhao, J. Yang and Y. Zhang, "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology", p*ISCA*, ages 14-23, 2009.

[7] S. Kang, et al., "A 0.1-μm 1.8V 256-Mb Phase-Change Random Access Memory (PRAM) with 66-MHz Synchronous Burst-Read Operation", *JSSC*, 42(1):210-218, 2007.

[8] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting Phase Change Memory as a Scalable DRAM Alternative", *ISCA*, pages 2-13, 2009.

[9] K. Lee, et al., "A 90nm 1.8V 512Mb Diode-Switch PRAM with 266MB/s Read Throughput", *JSSC*, 43(1):150-162, 2008.

[10] X. Wu, J. Li, L, Zhang, E. Speight, R. Rajamony, Y Xie, "Hybrid Cache Architecture with Disparate Memory Technologies", *ISCA*, pages 20-24, 2009.

[11] G. Dhiman, R. Ayoub, and T. Rosing, "PDRAM: A Hybrid PRAM and DRAM Main Memory System", *DAC*, pages 664-469, 2009.

[12] E. Ipek, J. Condit, E. B. Nightingale, D. Burger, and T. Moscibroda, "Dynamically Replicated Memory: Building Reliable Systems from Nanoscale Resistive Memories", *ASPLOS*, pages 3-14, 2010.

[13] P. Mangalagiri, A. Yanamandra, Y. Xie, N. Vijaykrishnan, M. J. Irwin, K. Sarpatwari, and O. A. Karim, "A Low-Power Phase Change Memory Based Hybrid Cache Architecture", *GLSVLSI*, pages 395-398, 2008.

[14] A. Agarwal, D. Blaauw, V. Zolotov, S. Sundareswaran, M. Zhao, K. Gala, and R. Panda, "Path-based statistical timing analysis considering inter and intradie correlations", *TAU*, pages 16-21, 2002.

[15] M. K. Qureshi, V. Srinivasan, J. A. Rivers, "Scalable High Performance Main Memory System Using Phase-Change Memory Technology", *ISCA*, pages 24-33, 2009.

[16] W. Zhang and T. Li, "Exploring Phase Change Memory and 3D Die- Stacking for Power/Thermal Friendly, Fast and Durable Memory Architectures", *PACT*, pages 101-112, 2009.