An Efficient PCM-based Main Memory System via Exploiting Fine-grained Dirtiness of Cachelines

Jie Xu, Dan Feng, Yu Hua, Wei Tong, Jingning Liu, Chunyan Li, and Zheng Li
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
Email:{xujie_dsal, dfeng, csyhua, Tongwei, jnliu, lichunyan, lizheng}@hust.edu.cn

***Abstract*—Phase Change Memory (PCM) has the potential to replace traditional DRAM memory due to its better scalability and non-volatility. However, PCM also suffers from high write latency and energy consumption. To mitigate the write overhead of PCM-based main memory, we propose a Fine-grained Dirtiness Aware (FDA) last-level cache (LLC) victimization scheme. The key idea of FDA is to preferentially evict cachelines with fewer dirty words when victimizing dirty cachelines. The modified word is defined to be dirty. FDA exploits two key observations. First, the write service time of a cacheline is proportional to the number of dirty words. Second, a cacheline with fewer dirty words has the same or lower reference frequency compared with other dirty cachelines. Therefore, evicting cachelines with fewer dirty words can reduce the write service time of cachelines, and will not increase the miss rate. To reduce the write service time of cachelines, FDA evicts the cacheline with the fewest dirty words when victimizing dirty cachelines. We also present FDARP to decrease the miss rate by further synergizing the number of dirty words with Re-reference Prediction Value. Experimental results show that FDA (FDARP) can improve the IPC performance by 8.3% (14.8%), decrease the write service time of cachelines by 37.0% (36.3%) and reduce write energy consumption of PCM by 27.0% (32.5%) under the mixed benchmarks.**

I. Introduction

With the rapid development of big data and multi-core technology, the demand for a large-size, low power as well as fast and reliable main memory system is more important than ever. Traditional Dynamic Random Access Memory (DRAM) technology has come into its limit of the feature size, and emerging Phase Change Memory (PCM) technology is promising to replace the DRAM technology due to its zero standby power, low read latency and high density [1][2].

However, the use of PCM also presents challenges. The write latency and energy of a PCM chip are several times more than DRAM [3][4]. In addition, since the size of concurrent bit-write is restricted inside a single PCM chip, several sequential write processes are required to write the entire cacheline into PCM-based main memory [5][6][7]. Supposing that the write latency of a PCM chip is T_{write} and the size of concurrent bit-write to PCM-based main memory is 64 bits [8][9], the write service time of a cacheline (64 bytes) is $64bytes/64bits = 8 \times T_{write}$. The write performance of PCM-based main memory degrades seriously due to the long write service time.

We find that not all the eight sequential writes are necessary because a large number of words in dirty cachelines are clean (not modified). The write service time of a cacheline is proportional to the number of dirty words when eliminating these clean word writes. Moreover, a cacheline with fewer dirty words has the same or lower reference frequency (the number of times a cacheline is referenced in cache) than other dirty cachelines in most benchmarks. Therefore, preferentially evicting cachelines with fewer dirty words can reduce the write service time, and will not increase the miss rate. This motivates us to propose more efficient LLC victimization schemes by considering the number of dirty words of cachelines. Some works [12][13][14][15][16][17] proposed to reserve dirty cachelines in the last-level cache (LLC), and evict more clean cachelines. They did not consider the different write cost between dirty cachelines, i.e., the write service time is related to the number of dirty words. Different from the prior works, this work reduces the write service time of cachelines by considering fine-grained dirtiness of cachelines in the LLC victimization schemes, and the new schemes will hardly increase the miss rate simultaneously. Our schemes are designed based on DRRIP [18]. If the cacheline to be evicted in DRRIP [18] is dirty, we re-find a dirty cacheline with the fewest dirty words in the same set and evict this new cacheline. The number of dirty words (NDW) is further combined with Re-reference Prediction Value ($RRPV$) [18] to decrease the miss rate.

We have the following contributions in this paper:

- We observe that a cacheline with fewer dirty words has a shorter write service time, and the same or lower reference frequency than other dirty cachelines in most benchmarks. The insights indicate that a cacheline with fewer dirty words should be evicted with higher priority.
- To reduce the write service time of cachelines, we propose FDA, which evicts the cacheline with the fewest dirty words if the cacheline to be evicted in DRRIP [18] is dirty. We also present FDARP to synergize the number of dirty words with $RRPV$ to decrease the miss rate.
- Experimental results show that FDA (FDARP) can improve the IPC performance by 8.3% (14.8%), decrease the write service time of cachelines by 37.0% (36.3%) and reduce write energy consumption of PCM by 27.0% (32.5%) than DRRIP [18] under the mixed benchmarks.

The rest of this paper is structured as follows. Sections

II and III introduce the background and motivation. Section IV presents the design and implementation. Sections V, VI and VII describe the experimental setup, related work and conclusion.

II. Background

A. Cache replacement policy

The most efficient caching algorithm would be to always discard the cacheline that will not be referenced for the longest time in the future [19]. Different cache replacement policies proposed in the literature try to find and victimize such a cacheline through prediction. The following two techniques are related to our work.

1) Least Frequently Used: The idea of Least Frequently Used (LFU) [20] replacement policy is that cachelines which are frequently accessed will be re-referenced (or reused) again. LFU uses a counter to measure the reference frequency of a cacheline. If the cacheline is hit, LFU increments its counter with one. If the counter is saturated, all the cachelines in the set will have their counters halved. The cacheline with the lowest reference frequency is selected as victim and will be evicted to the next level of memory.

2) Re-reference Interval Prediction: Re-Reference Interval Prediction (RRIP) [18] replacement policy uses an M-bit saturating counter ($RRPV$ or Re-reference Predict Value) per cacheline to predict the reuse distance. RRIP consists of three different cache replacement policies, i.e., SRRIP, BRRIP and DRRIP. The insertion, promotion and victimization policies of SRRIP are as follows:

- Insertion: A newly inserted cacheline is given a $RRPV$ of $2^M - 2$.
- Promotion: There are two different options, i.e., Hit Priority (HP) and Frequency Priority (FP). HP sets $RRPV$ to '0', and FP decreases $RRPV$ by '1'.
- Victimization: SRRIP selects the victim cacheline by finding the first cacheline with $RRPV = 2^M - 1$. If such cacheline does not exist, SRRIP increases the $RRPV$ of all the cachelines in the set and repeats the search.

Different from SRRIP, BRRIP inserts the majority of cachelines with $RRPV = 2^M - 1$ and infrequently inserts new cachelines with $RRPV = 2^M - 2$. DRRIP dynamically chooses between SRRIP and BRRIP by using Set Dueling [21].

B. Write unit of PCM

The maximum instantaneous power of a PCM chip is often limited due to noise minimization, and hence the size of concurrent bit-write (the write unit size) is restricted to a predefined constant [8][5][6][22]. Typical values of the write unit size are 2, 4, 8 and 16.

In this work, we suppose that the write unit size is 16 bits per chip [8][9], and a memory bank consists of four chips, as shown in Fig. 1. 64-bit data can be written into a memory bank in parallel, and the 64-byte cacheline needs 8 sequential write processes to finish writing [8][9]. The long write service time results in significant performance degradation of PCM-based main memory system.

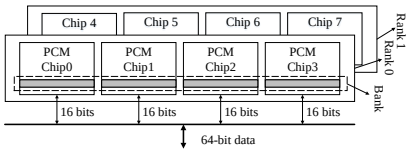

Fig. 1. PCM-based main memory architecture

Recent LLC replacement policies [18][20][21] mainly consider the cache misses. However, the long write service time of cachelines should also be taken into account in the LLC replacement policy due to the poor write performance of PCM.

III. Motivation

A cacheline with fewer dirty words has shorter write service time. Therefore, it should be evicted with high priority. On the other hand, the cacheline with fewer dirty words has the same or lower reference frequency than other dirty cachelines. Preferentially evicting it will hardly increase the miss rate. This motivates us to consider the number of dirty words in the LLC victimization policy.

A. Writing the dirty words matters

A cacheline requires eight sequential write processes to finish writing the entire data into PCM-based main memory. However, most of the write processes can be avoided because a large number of words are clean in dirty cachelines. The proportion of cachelines with different numbers of dirty words are shown in Fig. 2. Writing the dirty words only can reduce the write service time by 56% across the 16 different benchmarks from SPEC CPU2006 [23] on average. The zero word modified cachelines in Fig. 2 are different from clean cachelines. Zero word modified cacheline has been written at least once and is marked dirty in the LLC, while clean cacheline has never been written and is marked clean.

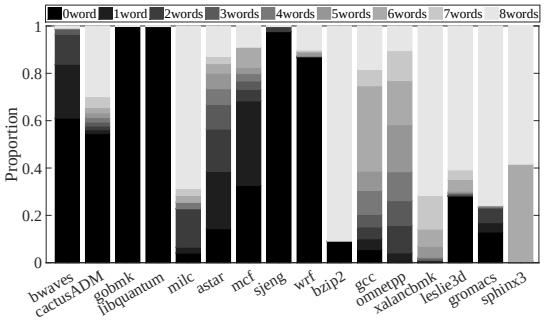

Fig. 2. Proportion of cachelines with different numbers of dirty words.

If the clean word writes are eliminated, the write service time of a cacheline will be proportional to the number of dirty words, as shown in Equation 1. In Equation 1, $T_{service}$, NDW, and T_{write} represent the write service time, the number of dirty words and write latency of a PCM chip respectively. Cachelines with fewer dirty words have shorter write service time.

$$T_{service} = NDW \times T_{write} \quad (1)$$

B. The number of dirty words varies

The number of dirty words in a cacheline varies in the benchmarks, as illustrated in Fig. 2. For example, in the bwaves benchmark, about 60% of the cachelines have zero modified word, and about 20% have one modified word. In the milc benchmark, about 70% have eight modified words and 20% have two modified words. The variation of the number of dirty words in the single benchmarks suggests that considering the number of dirty words could be effective under the single benchmarks. Besides, the number of dirty words varies significantly between benchmarks such as gobmk and bzip2. In gobmk, nearly all the cachelines have zero modified word, while most (90%) cachelines have eight modified words in bzip2. Even if a benchmark has a single dominant number of dirty words, running these benchmarks together on different cores will result in the shared LLC experiencing a mix of the number of dirty words. Therefore, considering the number of dirty words can be more important for reducing the write service time in a multi-core system running mixed benchmarks.

C. Average reference frequency of cachelines with different numbers of dirty words

Preferentially evicting cachelines with fewer dirty words can reduce the write service time. However, it may increase the miss rate if the reuse is not concerned. The write unit of CPU is word. The cacheline with more dirty words has more words referenced by write operations. Reference frequency can be used to predict the reuse [20], and we explore the relationship between the number of dirty words and the reference frequency. We have analyzed the average reference frequency of cachelines with different numbers of dirty words in sixteen memory-intensive benchmarks. The average reference frequency of cachelines with i dirty words is the ratio between the total reference frequency of cachelines with i dirty words and the total number of cachelines with i dirty words. Fig. 3 illustrates the results of nine representative benchmarks. We find that the benchmarks are divided into three categories by the relationship between the number of dirty words and average reference frequency.

- Type1 (half of the benchmarks): The average reference frequency is almost the same for different numbers of dirty words. The benchmarks include bwaves, cactusADM, gobmk, libquantum, omnetpp, xalancbmk, gromacs and sphinx3.
- Type2 (a quarter of the benchmarks): The average reference frequency is proportional to the number of dirty words. The benchmarks include astar, wrf, bzip2 and gcc.
- Type3 (a quarter of the benchmarks): There is no obvious relationship between the average reference frequency and the number of dirty words. The benchmarks include milc, mcf, sjeng, and leslie3d.

D. Observation Summary and Insights

We have the following observations.

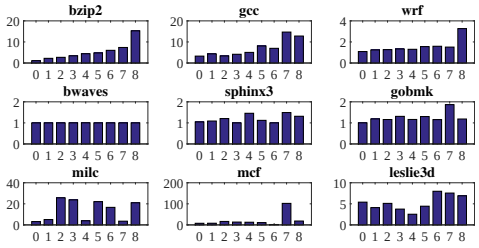

Fig. 3. Average reference frequency (Y-axis) of cachelines with different numbers of dirty words (X-axis) in nine representative benchmarks.

- Observation 1: The write service time of a cacheline is proportional to the number of dirty words.
- Observation 2: The number of dirty words varies in the single benchmarks and is more various in mixed benchmarks.
- Observation 3: A cacheline with fewer dirty words has the same or lower reference frequency than other dirty cachelines in most (Type1 and Type2) benchmarks.

For all the benchmarks, preferentially evicting cachelines with the fewest dirty words can reduce the write service time. For benchmarks in Type1, the miss rate will not increase. For benchmarks in Type2, the miss rate will decrease. For benchmarks in Type3, the miss rate may increase, since fewer dirty words does not mean lower reference frequency. Therefore, preferentially evicting cachelines with the fewest dirty words can improve the performance of PCM-based main memory system due to the decrease of the write service time (in all the sixteen benchmarks), and will not decrease the miss rate (in the benchmarks of Type1 and Type2). To decrease the miss rate, the number of dirty words can synergize with $RRPV$ [18] to reach a compromise between the write service time and miss rate.

IV. Design and Implementation

A. Design

Our schemes are designed based on DRRIP because of its simplicity and state-of-the-art performance. Two different cache victimization schemes, i.e., FDA and FDARP, are proposed. In our new cache replacement policies, the insertion and promotion policies are the same as DRRIP, while the victimization policies of DRRIP are replaced by FDA and FDARP.

1) FDA: A cacheline with fewer dirty words has shorter service time and the same or lower reference frequency in most benchmarks. To mitigate the write overhead of PCM as much as possible, the evicted cacheline is the one with the fewest dirty words among all the dirty cachelines in the set. Fig. 4 illustrates the idea of FDA. Assuming that the LLC is 16-way set-associative. If the cacheline to be evicted in DRRIP is clean, it will be silently dropped. If the cacheline to be evicted in DRRIP is dirty, a new dirty cacheline with the fewest dirty words (shortest write service time) in the same set is found. This new dirty cacheline will be evicted. FDA works only if the cacheline to be evicted in DRRIP is dirty, and the new victim is chosen among all the dirty cachelines rather than clean cachelines. In this work, a 3-bit $RRPV$ is

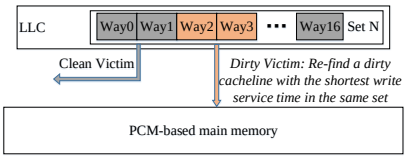

Fig. 4. Considering the different write service time among dirty cachelines when victimizing dirty cachelines.

used and the promotion policy uses HP (Hit Priority). FDA has a two-stage process. At the first stage, a cacheline with $RRPV = 7$ $(2^3 - 1)$ is found. The first stage is the same as the victimization policy of DRRIP. If the cacheline found in the first stage is dirty, the second stage starts. A cacheline with the minimum NDW is found among all the dirty cachelines in the second stage, and this cacheline is the new victim. The insertion, promotion and victimization policies of the new cache replacement policy are as follows:

- Insertion and Promotion: The same as DRRIP.
- Victimization (FDA): A cacheline with $RRPV = 7$ $(2^3 - 1)$ is found in the first stage. If the selected replacement candidate is clean, it's evicted. If the selected replacement candidate is dirty, a new replacement candidate with the minimum number of dirty words among all the dirty cachelines is selected in the second stage.

2) FDARP: Since cachelines with fewer dirty words may have higher reference frequency (in the benchmarks of Type3), FDA may increase the miss rate and degrade the IPC performance. FDARP synergizes the write service time with the reuse distance. A cacheline with a larger $RRPV$ [18] is predicted to be used further in the future, while a smaller NDW means shorter write service time. FDARP uses the value of λ ($\lambda = RRPV + 8 - NDW$) to combine the NDW with $RRPV$, and aims to reach a compromise between the write service time and reuse distance. FDARP is also a two-stage process. At the first stage, a cacheline with $RRPV = 7$ $(2^3 - 1)$ is found. If the cacheline found in the first stage is dirty, the second stage starts. A cacheline with the maximum of λ is found among all the dirty cachelines in the second stage, and this cacheline is the new victim. The insertion, promotion and victimization policies of the new cache replacement policy are as follows:

- Insertion and Promotion: The same as DRRIP.
- Victimization (FDARP): A cacheline with $RRPV = 7$ $(2^3 - 1)$ is found in the first stage. If the selected replacement candidate is clean, it's evicted. If the selected replacement candidate is dirty, a new replacement candidate with the maximum of λ among all the dirty cachelines is selected in the second stage.

B. Implementation and overhead

In this work, we assume the system has three levels of cache hierarchy, i.e., L1, L2 and LLC. When a new dirty cacheline in L2 cache is evicted and written back to the LLC, the old cacheline is read from the LLC. The read operation does not incur extra latency overhead because it can be overlapped with tag matching of the write operation. Then, the old cacheline is compared with the new cacheline at the word granularity. If a word is marked clean before and the current write modifies its value, the corresponding tag bit is set. A word has 64 bits while the cacheline size is 64 bytes, and therefore each cacheline has an 8-bit tag. The capacity overhead is 1.6%. Another 64-bit comparator is required to compare the old word with the new word, and a calculator is needed to count the number of dirty words according to the 8-bit tag. We use the Synopsys Design Compiler to synthesize the comparator and calculator. The logic incurs 250 gates, $1.2pJ$ write energy, and $1ns$ write latency overhead. The latency overhead is added only if the LLC misses occur, and the overhead is negligible compared with the LLC miss latency. Our victimization schemes need another search to find the cacheline with the minimum NDW or the maximum λ. This search can be implemented by replicating the same Find First One (FFO) logic as used in DRRIP [18]. Besides, this search is needed only if the cacheline to be evicted in DRRIP is dirty. Our schemes integrate into the existing DRRIP with minor modifications to the hardware of victimization policy.

V. Experimental Setup

We evaluate four different replacement policies using a cycle-accurate system simulator Gem5 [24]. We modify Gem5 simulator to support different cache replacement policies. The main memory model is based on a cycle-level main memory simulator NVMain [25]. In this work, we modify NVMain to eliminate the writes to clean words and redundant bits.

TABLE I
System configurations

Cores	4-Core, 3.2GHz, out-of-order
L1 I/D cache (per core)	32KB, LRU, 2-way, 2-cycle latency
L2 Cache (per core)	1MB, LRU, 8-way, 20-cycle latency
L3 Cache (shared)	16MB, 16-way, 50-cycle latency
Memory Controller	FRFCFS-WQF, 32-entry R/W queues
Memory Organization	4GB, 8B write unit size, Read 50ns Reset 100ns, 4.8pJ, Set 150ns, 2.8pJ

TABLE II
Mixed benchmarks

mix1	cactusADM, milc, 2×astar
mix2	2×mcf, 2×bzip
mix3	gcc, omnetpp, bzip2, sphinx3
mix4	gobmk, milc, gcc, gromacs
mix5	leslie3d, gromacs, wrf, bzip2
mix6	astar, mcf, gromacs, sphinx3
mix7	sjeng, wrf, bzip2, xalancbmk
mix8	libquantum, omnetpp, gcc, leslie3d
mix9	sjeng, omnetpp, gobmk, gcc

The configuration of the target system is given in Table I, and the system is based on a four-core processor. Sixteen single benchmarks (four copies) and nine mixed benchmarks shown in Table II are used in our experiment. All these benchmarks are selected from SPEC CPU 2006 [23]. We evaluate the following four different schemes in four-core system respectively:

- DRRIP: the cache replacement policy is selected from SRRIP and BRRIP.
- PM-VH-SD [14][15]: the insertion, promotion and victimization policies of DRRIP is modified to give priority

to clean cachelines. Set Dueling [21] is used to choose the policy from SRRIP or BRRIP which exhibits fewer writebacks. Clean cachelines are promoted using FP promotion, and dirty cachelines are promoted using HP promotion. Clean cachelines are victimized first, and dirty cachelines are victimized if no clean cacheline is found.

- FDA: victimize the cacheline with the fewest dirty words among all the dirty cachelines if the cacheline to be evicted is dirty.
- FDARP: victimize the cacheline with the maximum of λ ($RRPV + 8 - NDW$) among all the dirty cachelines if the cacheline to be evicted is dirty.

A. Experimental results

The proposed designs are evaluated in terms of write service time, miss rate, performance and write energy. All the experimental results are normalized to DRRIP [18].

1) Write service time of cachelines: The write service time of cachelines is shown in Fig. 5. For the single benchmarks (mixed benchmarks), the write service time of the cachelines decreases 4.6% (0.2%), 11.0% (37.0%) and 6.7% (36.3%) in PM-VH-SD, FDA and FDARP. The reduction of the write service time is the highest in FDA because it evicts the dirty cacheline with the fewest dirty words. The decrease of the write service time in FDARP is less than FDA because it combines the number of dirty words with $RRPV$ to reach a compromise. As motivated in section III-B, the number of dirty words is more various when running mixed benchmarks, and it's easy to find a victim cacheline with fewer dirty words when running mixed benchmarks. Therefore, the write service time reduces more in mixed benchmarks than in the single benchmarks. The write service time hardly decreases in PM-VH-SD, because fine-grained dirtiness is not considered in PM-VH-SD.

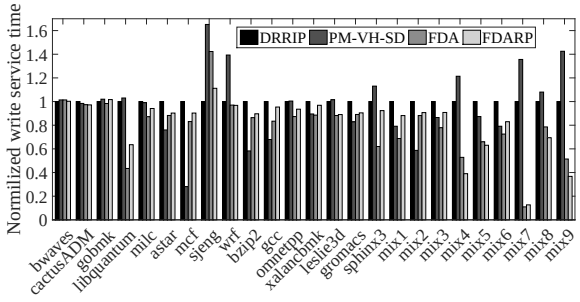

Fig. 5. Write service time of different replacement policies.

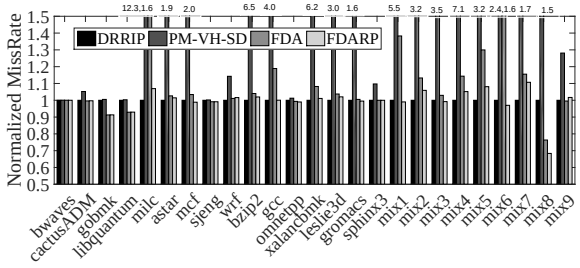

Fig. 6. Miss Rate of different replacement policies.

2) Miss rate: The miss rates of different schemes are shown in Fig. 6. For the single benchmarks (mixed benchmarks), the miss rates decrease by -186% (-226%), -5.5% (-16.8%) and 0.3% (0.5%) in PM-VH-SD, FDA and FDARP. The miss rate of PM-VH-SD increases a lot because it evicts clean cachelines without considering the reuse. The miss rate of FDARP decreases because it considers both the miss rate and the write service time.

3) IPC Performance: The IPC performance of different schemes is presented in Fig. 7. For the single benchmarks (mixed benchmarks), the IPC improvement of PM-VH-SD, FDA and FDARP is -28% (-43.4%), 0.0% (8.3%) and 3.0% (14.8%). The IPC performance is related to both the write service time and miss rate. For the libquantum benchmark, FDA and FDARP have the same miss rate, and FDA has shorter write service than FDARP. Therefore, FDA gets better IPC performance than FDARP for libquantum. For the mixed benchmarks, the IPC performance of FDARP outperforms DRRIP in all the mixed benchmarks, especially in mix4 (29.5%) and mix7 (27.9%). When running mixed benchmarks, cachelines with different numbers of dirty words are more evenly distributed. Therefore, it's easy to find a cacheline with the shortest write service time in mixed benchmarks, and the IPC of mixed benchmarks increases more than the single benchmarks. FDARP can decrease both the miss rate and the write service time, and therefore it outperforms other three schemes. The IPC of PM-VH-SD decreases seriously because clean cachelines are evicted before dirty cachelines, and it sacrifices the read performance.

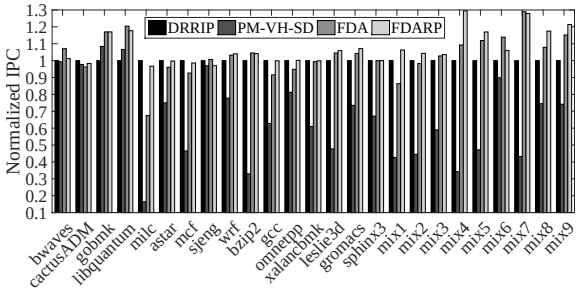

Fig. 7. Performance of different replacement policies.

4) Write energy: The reduction of write energy is illustrated in Fig. 8. For the single benchmarks (mixed benchmarks), the reduction of write energy is -0.1% (27.0%) and 3.3% (32.5%) in FDA and FDARP. The write energy increases several times in PM-VH-SD due to its high miss rate. FDARP and FDA can decrease the number of dirty words to write, and the write energy is related to the number of dirty words. Therefore, the write energy reduces in FDA and FDARP.

VI. Related Work

The related work can be divided into two categories: reducing the write service time of cachelines and reducing writes to PCM by modifying the LLC replacement policy.

A. Reducing the write service time of cachelines

Some techniques were proposed to reduce the service time of a cacheline and improve the throughput of main memory.

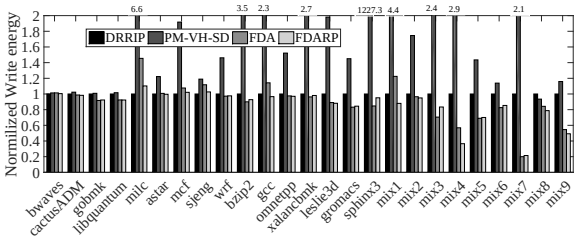

Fig. 8. Write energy of different replacement policies.

These techniques tried to maximize the power utilization for improving the write parallelism. Flip-N-Write [11] flipped the new data to ensure that the number of bits to write was fewer than half of the data width. Two-stage-write [8] divided a write into an accelerated speed write-0 stage and a parallel write-1 stage. Bit Mapping [7] distributed modified bits in a balanced way to get almost identical write service time in different cell groups. Min-WU [10] used simplified Frequent Pattern Compression to compress the data before writing. MaxPB [9] packaged the data units into the fewest number of write units under the power constraints. These techniques were implemented in the main memory controller. Unlike these techniques, our schemes reduce the write service time of cachelines by new LLC replacement policy.

B. Reducing writes to PCM by modifying the LLC replacement policy

Some schemes were designed based on Least Recently Used (LRU). ARI [16] partitioned a set into low-hit partition and high-hit partition, and gave priority to evict the clean cachelines in the low-hit partition. WCP [13] partitioned a shared LLC among multiple applications to reduce the writebacks and write misses. WADE [12] mitigated the write overhead by keeping highly reused dirty cache cachelines in the LLC. DRRIP [18] is more efficient than LRU, and some schemes were designed based on DRRIP. RWA [17] prevented dirty cachelines from frequent evictions by giving read and write operations hit/miss different $RRPVs$. Rodrguez-Rodrguez R et al. [14][15] proposed changing the DRRIP insertion, promotion and victimization policies by differing the read and write operations. None of those works considered the different write cost among dirty cachelines. Different from their approaches, our schemes try to reduce the write service time of cachelines and the miss rate is considered simultaneously.

VII. Conclusion

We introduce two novel cache victimization schemes to mitigate the write overhead of PCM-based main memory system. Our new schemes exploit two key observations. First, the write service time of a cacheline is proportional to the number of dirty words. Second, a cacheline with fewer dirty words has the same or lower reference frequency than other dirty cachelines. We propose FDA to reduce the write service time as much as possible. Moreover, the number of dirty words is synergized with $RRPV$ [18] in FDARP to decrease the miss rate. Experimental results show that our schemes fit will with mixed benchmarks, and FDARP gets better performance. FDARP can improve the IPC performance by 3.0% (14.8%), decrease the write service time of cachelines by 6.7% (36.3%) and reduce write energy consumption of PCM by 3.3% (32.5%) under the single benchmarks (mixed benchmarks) on average.

References

[1] B. C. Lee *et al.*, "Architecting phase change memory as a scalable dram alternative," in *Proceedings of ISCA*, 2009, pp. 2–13.

[2] M. Jalili and H. Sarbazi-Azad, "Tolerating more hard errors in mlc pcms using compression," in *Proceedings of ICCD*, 2016, pp. 304–311.

[3] P. Zhou *et al.*, "A durable and energy efficient main memory using phase change memory technology," in *Proceedings of ISCA*, 2009.

[4] X. Zhang *et al.*, "Tristate-set: Proactive set for improved performance of mlc phase change memories," in *Proceedings of ICCD*, 2015, pp. 659–665.

[5] S. Hanzawa *et al.*, "A 512kb embedded phase change memory with 416kb/s write throughput at 100a cell write current," in *Proceedings of ISSCC*, 2007, pp. 474–616.

[6] S. Kang *et al.*, "A 0.1-um 1.8-v 256-mb phase-change random access memory (pram) with 66-mhz synchronous burst-read operation," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 42, no. 1, pp. 210–218, 2007.

[7] Y. Du *et al.*, "Bit mapping for balanced pcm cell programming," in *Proceedings of ISCA*, 2013, pp. 428–439.

[8] J. Yue and Y. Zhu, "Accelerating write by exploiting pcm asymmetries," in *Proceedings of HPCA*, 2013, pp. 282–293.

[9] Z. Li *et al.*, "Maxpb: Accelerating pcm write by maximizing the power budget utilization," *ACM Trans. Archit. Code Optim. (TACO)*, vol. 13, December 2016.

[10] ——, "Exploiting more parallelism from write operations on pcm," in *Proceedings of DATE*, 2016, pp. 768–773.

[11] S. Cho and H. Lee, "Flip-n-write: A simple deterministic technique to improve pram write performance, energy and endurance," in *Proceedings of MICRO*, 2009, pp. 347–357.

[12] Z. Wang *et al.*, "Wade: Writeback-aware dynamic cache management for nvm-based main memory system," *ACM Trans. Archit. Code Optim. (TACO)*, vol. 10, December 2013.

[13] M. Zhou *et al.*, "Writeback-aware partitioning and replacement for last-level caches in phase change main memory systems," *ACM Trans. Archit. Code Optim. (TACO)*, vol. 8, 2012.

[14] R. Rodrguez-Rodrguez *et al.*, "Write-aware replacement policies for pcm-based systems," *The Computer Journal*, vol. 58, no. 9, p. 2000, 2015.

[15] R. Rodríguez-Rodríguez *et al.*, "Reducing writes in phase-change memory environments by using efficient cache replacement policies," in *Proceedings of DATE*, 2013, pp. 93–96.

[16] V. V. Fedorov *et al.*, "Ari: Adaptive llc-memory traffic management," *ACM Trans. Archit. Code Optim. (TACO)*, vol. 10, December 2013.

[17] X. Zhang *et al.*, *A Read-Write Aware Replacement Policy for Phase Change Memory*, Berlin, Heidelberg, 2011.

[18] A. Jaleel *et al.*, "High performance cache replacement using re-reference interval prediction (rrip)," in *Proceedings of ISCA*, 2010, pp. 60–71.

[19] L. A. Belady, "A study of replacement algorithms for a virtual-storage computer," *IBM Systems Journal*, vol. 5, pp. 78–101, 1966.

[20] D. Lee *et al.*, "Lrfu: A spectrum of policies that subsumes the least recently used and least frequently used policies," *IEEE Trans. Comput.*, vol. 50, Dec. 2001.

[21] M. K. Qureshi *et al.*, "Adaptive insertion policies for high performance caching," in *Proceedings of ISCA*, 2007, pp. 381–391.

[22] P. M. Palangappa and K. Mohanram, "Compex: Compression-expansion coding for energy, latency, and lifetime improvements in mlc/tlc nvm," in *Proceedings of HPCA*, 2016, pp. 90–101.

[23] J. L. Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, Sep. 2006.

[24] N. Binkert *et al.*, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, Aug.

[25] M. Poremba *et al.*, "Nvmain 2.0: A user-friendly memory simulator to model (non-)volatile memory systems," *IEEE Computer Architecture Letters*, vol. 14, no. 2, 2015.