PAPER

ARW: Efficient Replacement Policies for Phase Change Memory and NAND Flash*

Xi ZHANG[†a)], ***Member***, **Xinning DUAN**[†b)], **Jincui YANG**[†c)], ***and*** **Jingyuan WANG**[††d)], ***Nonmembers***

SUMMARY The write operations on emerging Non-Volatile Memory (NVM), such as NAND Flash and Phase Change Memory (PCM), usually incur high access latency, and are required to be optimized. In this paper, we propose Asymmetric Read-Write (ARW) policies to minimize the write traffic sent to NVM. ARW policies exploit the asymmetry costs of read and write operations, and make adjustments on the insertion policy and hit-promotion policy of the replacement algorithm. ARW can reduce the write traffic to NVM by preventing dirty data blocks from frequent evictions. We evaluate ARW policies on systems with PCM as main memory and NAND Flash as disk. Simulation results on an 8-core multicore show that ARW adopted on the last-level cache (LLC) can reduce write traffic by more than 15% on average compared to LRU baseline. When used on both LLC and DRAM cache, ARW policies achieve an impressive reduction of 40% in write traffic without system performance degradation. When employed on the on-disk buffer of the Solid State Drive (SSD), ARW demonstrates significant reductions in both write traffic and overall access latency. Moreover, ARW policies are lightweight, easy to implement, and incur negligible storage and runtime overhead.

key words: *replacement policy, NAND flash, phase change memory, non-volatile memory, emerging memory technology*

1. Introduction

Existing memory systems are increasingly constrained by their performance, power and technology scaling limitations [2]. Emerging memory technologies, based on non-volatile materials, are under investigation as a potential evolutionary approach to overcome this limitation and further scale down the technology. NAND Flash, based on poly-crystalline Si floating gate, is the dominant NVM technology driven by the explosive popularity of portable electronic devices. A variety of other memory technologies such as Spin Torque Transfer RAM (STT-RAM), Phase-Change Memory (PCM), and Resistive RAM (RRAM) are promising for high-performance and low-power system [3], [4]. One of the best candidates, PCM, exploits the thermally reversible phase transitions of chalcogenide alloys (e.g., $Ge_2Sb_2Te_5$) to match both the high-density as well as the high-performance specifications. To further improve the performance and scalability of memory systems, we need to explore NVM as feasible alternatives to existing memory technology for future computing systems.

Two promising NVM technologies focused in this paper are *Flash* and *Phase Change Memory* (PCM). NAND Flash can only be used as a disk cache [5] or an alternative for magnetic disks [6] because it is not byte-addressable and is about 200X slower than DRAM. PCM, on the other hand, is only 2x-4x slower than DRAM and can provide up to 4x more density than DRAM, which makes it a promising candidate for main memories [7]. PCM has been backed by key industry manufacturers such as Intel, STMicroelectronics, Samsung and IBM as both off-chip and on-chip memory [8]. Table 1 lists features of several memory technologies: SRAM, DRAM, PCM and NAND Flash based on the data obtained from literature [9]–[11].

There are several challenges to overcome before NAND Flash and PCM become feasible parts of the memory hierarchy. One challenge is the latency and energy costs of write operations. Due to the non-volatile property, it takes more time and energy for NVM to overwrite data. The other challenge is the limited lifetime due to the endurance constraint. For NAND Flash, the number of erase operation allowed to be performed for each block ranges from 10^4 to 10^6 [12], while for PCM, it can tolerate 10^8 – 10^{15} writes per cell [13]. Once the endurance constraint is exceeded, the block or cell will be worn out, making the memory unreliable. Therefore, the write traffic to NVM should be reduced in order to decrease access latency, power and meanwhile increase lifetime.

In this paper, we focus on optimization techniques on replacement policies to decrease write traffic to NAND Flash and PCM. Replacement policies [14]–[17] have been studied extensively to improve cache and buffer efficiency. However, most of previous works don't consider the asymmetry costs of read and write requests in context of NVM. Recently a few cache management techniques [18]–[20] have been proposed to mitigate the write overhead of PCM, but most of them are performed on PCM itself, without optimizing the write traffic sent from LLC. In addition, there are some buffer management policies [12], [21], [22] that can reduce the writebacks to NAND Flash, but they fall short in efficiency and are difficult to implement on chip. To

Manuscript received May 18, 2016.
Manuscript revised August 25, 2016.
Manuscript publicized October 13, 2016.

[†]The authors are with Key Laboratory of Trustworthy Distributed Computing and Service of Ministry of Education, Beijing University of Post and Telecommunications, Beijing, 100876 China.

[††]The author is with School of Computer Science and Engineering, Beihang University, Beijing 100191, China.

*An earlier version of this paper was presented at the 9th International Symposium on Advanced Parallel Processing Technologies [1].

a) E-mail: zhangx@bupt.edu.cn
b) E-mail: mumu131418@bupt.edu.cn
c) E-mail: jincuiyang@sina.com
d) E-mail: jywang@buaa.edu.cn

DOI: 10.1587/transinf.2016EDP7205

Copyright © 2017 The Institute of Electronics, Information and Communication Engineers

Table 1 Comparison of different memory technologies.

Features	SRAM	DRAM	PCM	NAND FLASH
Memory Cell Factor(F^2)	50-200	6	4-10	2-5
Read Time (ns)	<1	30	10-100	10^4
Write /Erase Time (ns)	<1	50	100-300	10^5
Number of Rewrites	10^{16}	10^{16}	$10^8 - 10^{15}$	$10^4 - 10^5$
Dyn. Power	Low	Medium	Medium read High write	Very high read Even high write
Leak. Power	High	Medium	Low	Low
Data Retention	N	N	Y	Y

minimize the write overhead effectively and efficiently, we propose Asymmetric Read-Write (ARW) replacement policies, taking into account the imbalance of read and write costs by setting different priorities of evictions. Specifically, by attaching each data block with a 2-bit or 3-bit tag, ARW can prevent hot dirty data blocks in cache from frequent evictions. When employed in the memory level above NVM, ARW can decrease writebacks sent to the lower-level NVM. However, over-protecting dirty blocks may lead to inefficient use of memory space, i.e., more read misses may occur and overall hit rate degrades. Thus ARW should be delicately designed to prevent performance degradation. Three ARW policies with different trade-offs between performance and cost are proposed.

This paper makes the following contributions.

- The base ARW policies, which consider the asymmetric costs of read and write operations by setting different evicting priorities, are proposed to minimize the write traffic.
- To prevent dead lines from wasting memory capacity, improved ARW policies are proposed by distinguishing access frequencies, which can further reduce write traffic.
- ARW policies are evaluated on two PCM-based systems: with and without DRAM cache. Simulation results show that write traffic can be reduced by 47.4% and 16.0% respectively against LRU baseline. Moreover, ARW policies incur negligible runtime and storage overhead.
- Trace-driven simulation on NAND Flash-based system shows that when adopted by on-disk buffer, ARW can significantly decrease write traffic as well as overall memory latency than pervious methods. ARW is lightweight, simple to implement, and easy to maintain.

Note that our previous study [1] proposed Read-Write Aware replacement policies for PCM, which are only the subset of ARW policies in this paper. A new policy, namely, 2 bit ARW-I, is proposed in this paper, which can outperform the policies proposed in [1] with less storage overhead. In addition, we also extend our methods for NAND Flash in this paper. The design details are described in Sect. 3.

2. NVM Characteristics and Its Memory Hierarchy

In this section, we first review the technology background of PCM and NAND Flash, and then introduce the NVM-based memory hierarchy.

2.1 Technology Background

Different NVM technologies have their particular storage mechanisms.

1) PCM Properties

Tradition DRAM requires consistent refresh operations to sustain data. Unlike DRAM, PCM is a type of non-volatile memory that can retain its data even when the power is turned off. The PCM cell consists of a standard NMOS access transistor and phase change material. There are two kinds of PCM write operations, SET and RESET, and these two write operations are controlled by different electronic currents. Since the SET status and RESET status have a large variance on their equivalent resistance, the data stored in the cell can be retrieved by sensing the device resistance by applying very low power.

In general, PCM is 2X-4X slower than DRAM, and the write latency is much larger than read (about an order of magnitude). In addition, PCM can provide up to 4X more density than DRAM. The reason is that PCM cell can be in different degrees of partial crystallization and therefore is able to store two or more bits in each cell. Finally, PCM cells can endure a maximum of $10^8 - 10^{15}$ writes, which poses a reliability problem. To make PCM a feasible candidate for main memory, the write overhead should be minimized, which will be addressed in our work.

2) NAND Flash Properties

The physical mechanism of the flash memory is to store bits in the floating gate and control the gate threshold voltage. The memory cell modifies its threshold voltage by adding electrons to or subtracting electrons from the isolated floating gate. As its capacity increases and price decreases, flash-memory-based solid state disk (SSD) has been used as a cache for magnetic disk, or even replace it. Different from magnetic disk, NAND Flash memory is a kind of write-once and bulk erase material. It is organized in terms of blocks, and each block consists of a fixed number of pages.

There are three operations for NAND Flash, that is, *read*, *write* and *erase* operations, which provide very different overheads. Specifically, the latency of a write operation is about 10X higher than that of a read operation. The erase operation further costs 7.5X more clocks than write opera-

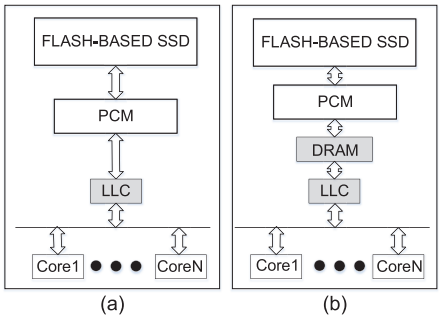


Fig. 1 (a) System with PCM, (b) systemg with hybrid memory.

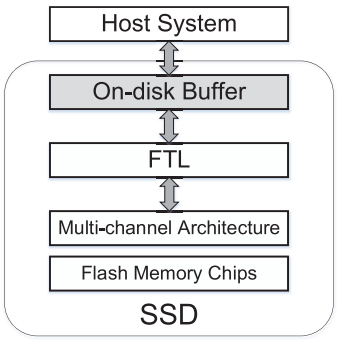


Fig. 2 NAND flash-based SSD with on-disk buffer.

tion, due to the erase-before-write requirement of the material. That is, NAND Flash cannot directly update the data by overwriting it until an erase operation is performed. In addition, the read and write operations are performed on a page granularity, while the erase operation can only be done in a coarser granularity called "block", which involves a number of pages. Consequently, even a small update to a single page requires erasing all the pages within the block.

2.2 NVM-Based Memory Hierarchy

We introduce NVM-based memory hierarchy where PCM acts as main memory and NAND Flash works as disk.

1) Organizations with PCM as Main Memory

Figure 1 shows two systems in which PCM acts as main memory. In Fig. 1 (a), PCM completely replaces DRAM to increase capacity. However, the higher access latency of PCM may lead to performance degradation. To mitigate the latency gap, a DRAM buffer can be introduced as the cache of PCM (shown in Fig. 1 (b)). In such a hybrid main memory, system can get benefits from both sides, the large capacity of PCM and the low access latency of DRAM. It has been shown that a relative small DRAM buffer (3% size of the PCM storage) can filter most of the accesses to PCM [10]. This organization is similar to that proposed in [10], but is different in access granularity and data placing sequence. In the previous hybrid system, PCM acts as same as DRAM in a traditional system, and is therefore managed by Operating System. In contrast, the DRAM cache in Fig. 1 (b) is not visible to OS and organized by DRAM controller. Note that in our system DRAM is off-chip, however, on-chip embedded DRAM also applies as that in IBM POWER7 [23]. Therefore DRAM is organized at cache block granularity as same as SRAM cache on chip. The reason why we don't use page granularity is that when an eviction occurs in DRAM, only a dirty line is written back to PCM instead of a dirty page, making the write traffic minimized. This mechanism requires Dirty Bit attached to each DRAM cache line. In addition, Valid Bit and LRU Bits are also required per cache line. In order to reduce access latency, the tags of DRAM cache can be made of SRAM.

We assume write-back SRAM cache and write-back DRAM cache (rather than write-through cache) to further reduce the write traffic. Therefore PCM is written only when dirty data lines are replaced from the upper-level memory, whose replacement policy determines the amount of write traffic to PCM. The default replacement policy of SRAM cache and DRAM cache is LRU, which doesn't consider the imbalance of replacement cost between read and write requests. ARW policies, instead, distinguish the dirty lines and clean lines, and are adopted in the upper-level memory above PCM. In organization shown in Fig. 1 (a), the upper-level memory is the on-chip LLC. In Fig. 1 (b), ARW can be used in both LLC and the DRAM buffer.

Please note that, when *Asymmetric Read-Write* policies are applied for PCM-based main memory, *Read* here refers to load and store requests as they don't overwrite data lines, while *Write* refers to writeback requests which produce dirty data lines.

2) NAND Flash-based Disk Organization

Figure 2 shows the modern SSD with an on-disk buffer between the host interface and a flash translation layer (FTL). FTL is employed to hide the idiographic characteristic of flash memory and to mimic themselves as block devices. FTL provides an address mapping between the logical addresses used by the host and the physical addresses used in flash memory. Besides, FTL internally issues extra read, write, or erase operations to efficiently manage the storage space, and the number of these operations depends both on the data access pattern from the upper layer and the algorithm adopted by the address mapping [24]. The buffer stores data from the host first and then writes the data to the NAND Flash memory afterwards. The replacement policy employed by the buffer is responsible for scheduling write requests, making it a vital role in minimizing the write traffic.

3. Asymmetric Read-Write Replacement Policy

In this section, we first describe the overview of ARW, and then illustrate the base ARW and its improved version in details. Note that since ARW is a general technique and can be applied for both PCM and NAND Flash, for simplicity, we take PCM-based memory as an example to illustrate ARW. The difference in implementation issues between applying

ARW for NAND Flash and for PCM will be addressed in Sect. 3.4.

3.1 ARW Overview

ARW policies are proposed aiming at reducing evictions of dirty lines to PCM, which can be achieved by retaining dirty lines long enough to satisfy subsequent writebacks. The adjustments on the replacement algorithms can be designed based on the recently proposed Re-reference Interval Prediction (RRIP) policy [16].

RRIP is able to prevent blocks with a distant re-reference interval from polluting the cache. An M-bit register per cache block is used to store its Re-reference Prediction Value (RRPV), which predicts the re-reference interval is *near-immediate*, *long* or in the *distant* future. Blocks with *distant* RRPV will be evicted first at capacity conflict, then will be the blocks with *long* RRPV. Finally *near-immediate* RRPV blocks will be evicted. By always inserting new blocks with a *long* RRPV, new blocks can be evicted earlier compared to LRU. Thus, RRIP prevents cache blocks with a *distant* re-reference interval from evicting blocks having a *near-immediate* re-reference interval, making it beneficial for workloads with stream access and mixed access patterns.

Inspired by RRIP, we proposed ARW, which distinguishes read and write requests by setting different RRPVs. Therefore dirty lines can be prevented from frequent evictions. We assume an M-bit register per cache line to store RRPV. Similar to RRIP, RRPV 0 means the data line is predicted to be re-referenced in *near-immediate* term and has the lowest priority to be evicted, while RRPV $(2^M - 1)$ means the data line is predicted to be re-referenced in *distant* future and has the highest priority to be evicted. RRPV between 0 and $(2^M - 1)$ means the data line has medium re-reference interval. The lager RRPV is, the higher priority the data line will be evicted. Based on this intuition, through setting a small RRPV, dirty data lines can be retained longer in cache to avoid frequent evictions. Therefore, write operations to lower level PCM can be decreased. With various M and RRPV settings, ARW can have different trade-offs between performance and cost. Three ARW policies are implemented and evaluated in this paper, a base ARW policy with 3-bit RRPV register called 3bit ARW-B, and two improved versions: 3bit ARW-I policy, and a more cost-effective 2bit ARW-I policy (with 2bit RRPV register).

3.2 Base ARW Policy

In 3bit ARW-B, a 3-bit RRPV register is attached to each cache line. Therefore, there are eight RRPVs (i.e.,from 0 to 7) and the replacement priorities fall into eight categories. To better illustrate 3bit ARW-B, the replacement algorithm is decomposed into three parts: the insertion policy, the promotion policy and the victim selection policy.

The Insertion Policy. On a load or store miss, RRPV of the new block is set to 6. On a writeback miss, RRPV of the new block is set to 0.

The Promotion Policy. On hit, no matter load hit, store hit or writeback hit, RRPV of the hit line is set to 0.

The Victim Selection Policy. Search for the first data line from left to right whose RRPV is 7. Once found, make it the victim to be replaced. If not exist, all the RRPVs in this set are incremented until some RRPV reaches 7.

Behaviors of 3-bit ARW-B are shown in Fig. 3 (a).

3.3 Improved ARW Policy

In the base ARW policy, all dirty lines are protected to decrease writebacks. For example, newly installed dirty line is given the lowest priority to be evicted. Once there are subsequent writebacks to the dirty line in a short term, evictions can be avoided. However, if it is a single-use dirty line (i.e. a dead line without subsequent accesses), protecting it may decrease hit rate. According to the promotion policy, the dead line won't be replaced until its RRPV is incremented to $(2^M - 1)$. Furthermore, this RRPV incrementing process is much slower than that under LRU policy. The reason is that compared to LRU, newly inserted data lines on a read miss is prone to be evicted under ARW, and thus it takes longer to evict the already inserted dead lines.To sum up, over-protecting the dead dirty lines in base ARW leads to memory under-utilization problem.

To address this limitation, new replacement algorithms are required not only to distinguish clean and dirty blocks, but also to consider the access frequency. Based on 3bit ARW-B, an improved policy called 3bit ARW-I is proposed, which evicts the dead dirty lines as early as possible. The insertion policy in 3bit ARW-I is different from that in 3bit ARW-B. Specifically, newly inserted dirty lines will be given a medium RRPV value instead of 0. Such behavior can prevent single-use dirty lines from polluting the cache, and meanwhile gives them a certain amount of time to receive subsequent accesses. Once re-referenced, its RRPV is promoted to 0. Consequently, only dirty lines with multiple uses are protected.

However, always giving new dirty lines medium RRPV will shrink the capacity of dirty lines. Because under the new insertion policy in 3bit ARW-I, write operations have similar behaviors as read operations, and dirty lines are not given higher priority enough for protection than clean lines. Accordingly, more dirty line evictions may be triggered. To solve this problem, load/store hit-promotion policy is also modified. Once load/store hit, RRPV of the hit line is set to some medium value (e.g., 3) instead of 0, which is different from writeback hit-promotion policy that sets RRPV to 0. In summary, 3bit ARW-I is described as follows:

The insertion policy. On a load or store miss, RRPV of the new block is set to 6. On a writeback miss, RRPV of the new block is set to 5.

The promotion policy. On read (load/store) hit, RRPV of the hit line is set to 3; on write hit, RRPV of the hit line is set to 0.

The victim selection policy. The same as 3bit ARW-B

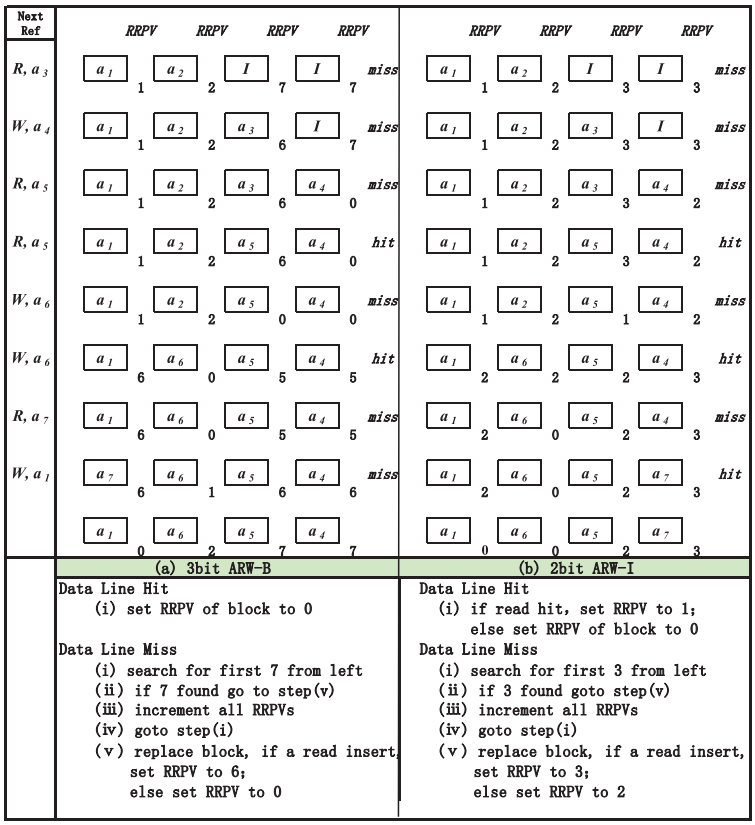


Fig. 3 Examples of ARW policies.

policy.

As illustrated above, the more bits in the RRPV register, the more replacement priorities in the policy, and it is more difficult for dead lines with RRPV 0 to get evicted. Consequently, for access patterns with infrequent re-references, reduce the number of bits in RRPV register can improve performance. Thus, 2bit ARW-I is proposed, with 2 bits in each RRPV register and totally 4 RRPVs. 2bit ARW-I is described as follows:

The insertion policy. On a load or store miss, RRPV of the new block is set to 3. On a writeback miss, RRPV of the new block is set to 2.

The promotion policy. On read (load/store) hit, RRPV of the hit line is set to 1; on write hit, RRPV of the hit line is set to 0.

The victim selection policy. Search for the first data line from left to right whose RRPV is 3. Once found, make it the victim. If not exist, all the RRPVs in this set are incremented until some RRPV reaches 3.

Behaviors of 2-bit ARW-I are shown in Fig. 3 (b), which depicts the benefits of improved ARW with an example. The reference sequence {Read a_3, Write a_4, Read a_5, Read a_5, Write a_6, Write a_6, Read a_7, Write a_1} is injected to caches adopted with 3bit ARW-B and 2bit ARW-I respectively, and a_4 is a single-use dirty line. When a_7 read miss occurs, 2bit ARW-I evicts a_4 while 3bit ARW-B retains a_4. It shows that 2bit ARW-I can evict the dead line earlier than 3bit ARW-B. Please note that 3bit ARW-B and 3bit ARW-I policies are actually RWA policies in [1]. In other words, RWA and RWA-I proposed in [1] are denoted as 3bit ARW-B and 3bit ARW-I in this paper. The 2bit ARW-I policy is a new policy proposed in this study.

3.4 Applying ARW for NAND Flash

We have already described how ARW policies work and how to apply them for PCM. But when applying ARW for NAND Flash, there are several differences to address. First of all is the different memory levels to be applied. ARW is employed by LLC and DRAM cache above PCM, while for NAND Flash, ARW is used by on-disk buffer. The second difference is the access granularity. The data blocks are replaced in terms of a cache line in LLC for PCM, compared to the frame granularity managed in buffer for NAND Flash, where frame consists of one or more pages. The third difference lies in the implementation methods. ARW policies are implemented as hardware logic integrated into cache controller for PCM. For NAND Flash, however, they are usually implemented as a software component which acts as a part of the operation system or database management system.

4. Evaluation

In this section, we evaluate ARW policies for both PCM-

Table 2 Baseline system parameters for evaluation on PCM.

Component	Parameter
Processing Core	8-core/Sparc V9 ISA/in-order/
L1 Cache	I-cache and D-cache each 16KB, 2-way, 64B linesize, writeback policy, LRU, 2 cycles
L2 cache	4M, 8 banks, 8-way, 64B linesize, writeback policy, LRU, 15 cycles
DRAM cache	256MB, 8 banks, 8-way, 64B linesize writeback policy, 200 cycles
PCM memory	8GB, 1024 cycles read latency, 4096 cycles write latency, 4 KB page size
Flash-based SSD	100K cyckes, 100% hit rate
Network Configuration	4x2 Mesh, 1-cycle hop latency

Table 3 PCM workloads.

Workload	Problem Size	Workload	Problem Size
FFT	256K points	x264	simsmall
RADIX	1048576 keys, radix=1024	VIPS	simsmall
OCEAN	514 x 514 ocean	SWAPTIONS	simsmall
BARNES	16K particles	RAYTRACE	simsmall
CHOLESKY	tk15.o, Postpast partition size:32	CANNEAL	simsmall
STREAMCLUSTER	simsmall	FACESIM	simsmall

based and NAND Flash-based systems, with execution-driven simulation method and trace-driven simulation method respectively. We evaluated 3bit ARW-B, 3bit ARW-I and 2bit ARW-I with three previous methods, that is, LRU, RRIP and CCFLRU [12]. CCFLRU is a competitive baseline originally designed to reduce writebacks to SSD. With CCFLRU, data blocks are classified into four categories: clean cold, clean hot, dirty cold and dirty hot, by maintaining two LRU lists. Though it's not a feasible technique for LLC due to its extreme complexity, we still simulate it on-chip to make a competitive comparison with ARW policies. All policies are evaluated in terms of the number of write operations and runtime (or overall latency).

4.1 Experiment Setup for Evaluation under PCM

Experiments under PCM are conducted with full-system simulation based on Simics [25] and the GEMS toolset [26], and the parameters of baseline configurations are given in Table 2. Note that we have two baseline systems as shown in Fig. 1. Except DRAM, all parameters of the two baseline systems are the same. Specifically, an eight-core CMP system with simple in-order core modal is adopted. L1 cache is private, while L2 cache, DRAM cache and PCM memory are all shared. L2 cache is inclusive, and DRAM cache is non-inclusive. LRU policy is employed in both L2 and DRAM cache, while NRU [14] page replacement policy is used on PCM memory. Detailed memory controllers are also simulated. Since PCM is an emerging memory technology, the projection of its feature tends to be varied. However, the parameters are consistent with other researchers' assumptions.

We selected twelve representative scientific workloads with relatively large working set as shown in Table 3, five from SPLASH-2 [27] and seven from PARSEC [28]. Large working set may result in a large amount of evictions and accordingly more write operations to PCM. All simulation results are normalized to the LRU baseline.

Table 4 NAND flash simulation parameters.

Operation	Access time	Access granularity
Read	20 μs /page	Page(2KB)
Write	200 μs/page	Page(2KB)
Erase	1.5 ms/block	Block (128KB = 64 pages)

Table 5 Synthetic traces for NAND flash accesses.

Type	Total references	Different pages accessed	Read / Write ratio	Locality
T9182	300,000	10,000	90% /10%	80% / 20%
T5582	300,000	10,000	50% / 50%	80% / 20%
T1982	300,000	10,000	10% / 90%	80% / 20%
T9155	300,000	10,000	90% / 10%	50% / 50%
T5555	300,000	10,000	50% / 50%	50% / 50%
T1955	300,000	10,000	10% / 90%	50% / 50%

Table 6 The OLTP Trace.

Attribute	Value
Total Buffer Request	555132
Read / Write Ratio	77% / 23%
Total Different Pages Accessed	49680

4.2 Experiment Setup for Evaluation under NAND Flash

Experiments under NAND Flash-based system are conducted with a trace-driven simulator called Flash-DBSim [29]. We simulate a 128MB NAND Flash device with 64 pages per block and 2KB per page. The parameters of the I/O operations are given in Table 4, which is the default configuration of Flash-DBSim. The erasure limitation of a block is 100,000 cycles. For simplicity, we assume that the size of a frame in the buffer is equal to that of a page in NAND Flash, and this assumption will not have impacts on performance comparison.

We use both real and synthetic traces for evaluation. Table 5 gives the six synthetic traces, where the read / write ratio of "90% / 10%" denotes the read and write operations in the traces are 90 and 10 percents respectively, and the locality of "80% / 20%" refers to that 80% of the total references are performed on 20% of the pages. Table 6 introduces the real-world OLTP trace collected in a bank system [30].

4.3 Results under PCM

4.3.1 Results on the Organization without DRAM Buffer

We first show the results under the organization without DRAM cache (as shown in Fig. 1 (a)). Figure 4 presents the amount of write traffic to PCM normalized to LRU baseline. 2bit ARW-I and CCFLRU outperform other baselines, and achieve reductions by 16.0% and 16.3% respectively.

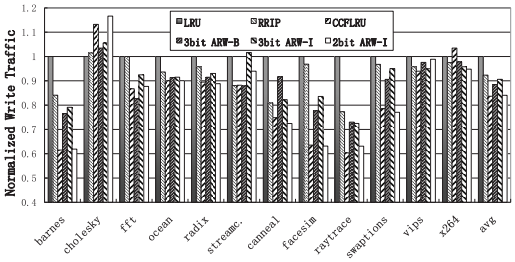


Fig. 4 Normalized write traffic to PCM without DRAM buffer.

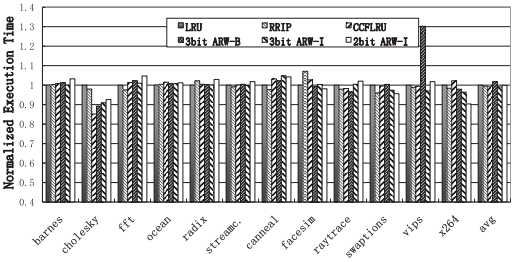


Fig. 5 Normalized execution time without DRAM buffer.

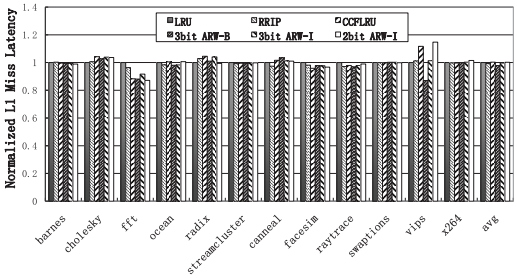


Fig. 6 Normalized L1 miss latency without DRAM buffer.

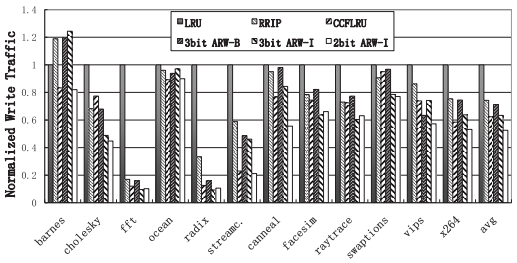


Fig. 7 Normalized write traffic to PCM with DRAM buffer.

Among the twelve workloads, 2bit ARW-I gets comparative performance with respect to CCFLRU in five workloads, gets more reduction in three workloads. Note that 2bit ARW-I achieves the comparable performance as CCFLRU with much reduced complexity. 3bit ARW-B performs a little better than 3bit ARW-I, and they reduce write traffic by 11.5% and 9.4% respectively. Since 2bit ARW-I performs better than both 3bit ARW-I and 3bit ARW-B, it implies that early eviction of dead lines can benefit most of the cases. RRIP decreases write traffic by 7.7% since it can evict single-use data lines earlier than LRU, and therefore improves hit rate and reduce the number of evictions to some extent. However, RRIP is a general replacement policy, with performance not as good as other policies specially designed for optimizing write traffic.

Figure 5 shows the execution time normalized to LRU baseline. On average, only 3bit ARW-B slightly increases runtime by 1.7%, while all other policies decrease runtime within 1%. In can be concluded that these policies are efficient and have negligible impact on system runtime. Figure 6 shows the L1 miss latency normalized to LRU baseline. 3bit ARW-B decreases L1 miss latency by 2.2%, while all other policies achieve equivalent results as LRU. The possible reason might be the decrease in write miss penalty is compensated by the increase in read miss penalty, and thus the L1 miss latency remains stable.

4.3.2 Results on the Organization with DRAM Buffer

The simulation results under the organization comprised of hybrid main memory are presented here, where ARW policies are adopted by both LLC and DRAM cache. Figure 7 shows the normalized write traffic to PCM. Compared to LRU baseline, RRIP, CCFLRU, 3bit ARW-B, 3bit ARW-I and 2bit ARW-I reduce write traffic by 25.7%, 37.6%, 28.8%, 36.7% and 47.4% respectively. 2bit ARW-I achieves the best performance, and CCFLRU takes the second place. Compared to 3bit ARW-B, 3bit ARW-I prevents single-use dirty lines from over-protecting, and thus gets better performance. 2bit ARW-I has fewer replacement priorities than 3bit ARW-I, and therefore is able to evict dead lines earlier. In eight out of twelve workloads, 2bit ARW-I outperforms 3bit ARW-I.

Recall Fig. 4, under the organization without DRAM buffer, 3bit ARW-B performs better than 3bit ARW-I, different from what we observe under the organization with DRAM buffer. The reason is that with DRAM buffer, the requests are filtered by two cache levels, which leads to increase in the proportion of single-use lines in DRAM cache. Such pattern is suitable to 3bit ARW-I policies. Based on the similar reason, 2bit ARW-I also gets benefits and performs better than CCFLRU by 9.8%.

There is an anomalous case that 3bit ARW-I and 3bit ARW-B increase write traffic against LRU in workload barnes. Recalling the results shown in Fig. 4, both 3bit ARW-I and 3bit ARW-B reduce write traffic when they are only adopted in L2 cache. It can be inferred that the adoption of 3bit ARW-I and 3bit ARW-B on DRAM cache results in increase in write traffic. The possible reason is that for barnes, most dirty lines in DRAM cache has only a few rereferences. To improve performance, the dead lines should be evicted as soon as possible. However, there are eight priorities in 3bit ARW-I and 3bit ARW-B, which take more time to evict deadlines than LRU and 2bit ARW-I, and thus leads to negative impact on performance.

Figure 8 shows the normalized execution time compared to LRU baseline. In general, the execution time remains stable. On average, RRIP, 3bit ARW-B and 3bit

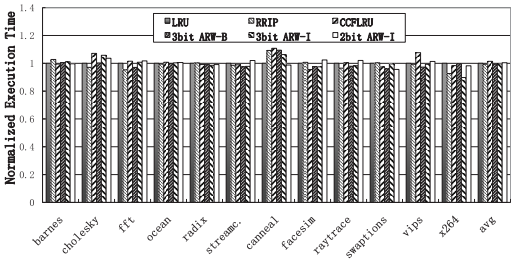


Fig. 8 Normalized execution time with DRAM buffer.

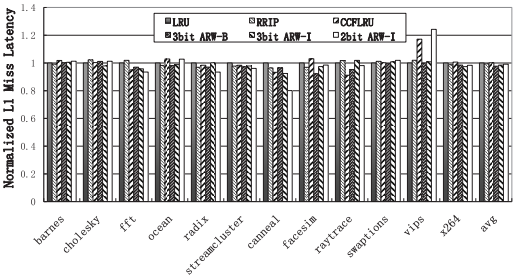


Fig. 9 Normalized L1 miss latency with DRAM buffer.

ARW-I can reduce execution time by 0.7%, 0.4% and 0.6%. Meanwhile, CCFLRU and 2bit ARW-I increase execution time by 1.4% and 0.4%. Figure 9 shows the normalized L1 miss latency against LRU baseline. On average, CCFLRU has almost the same results as LRU, while RRIP, 3bit ARW-B, 3bit ARW-I, 2bit ARW-I can reduce L1 miss latency by 0.6%, 2.4%, 1.5% and 0.9% respectively.

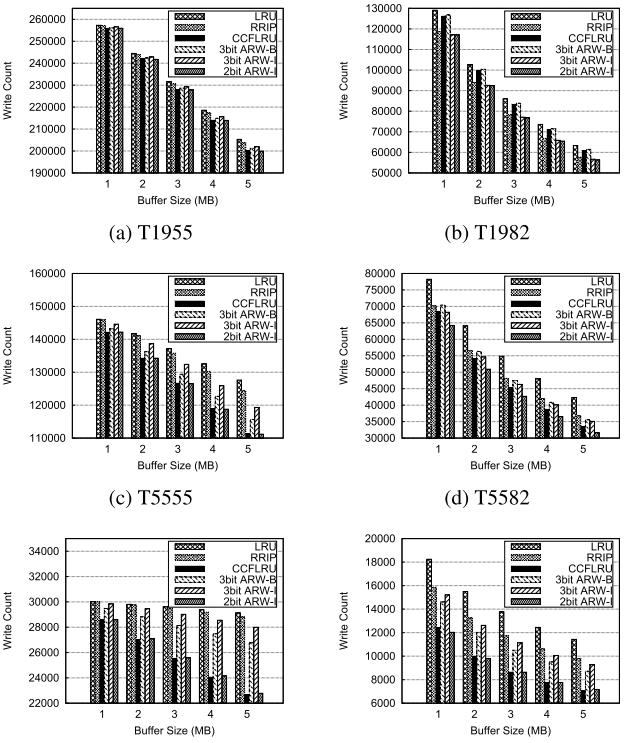
4.3.3 Overhead

The hardware overhead of our proposal consists of two main components, the registers for RRPV in cache tag-store entries, and the update logic. All the cache sets share one set of logic to update RRPV. Table 7 lists the storage overhead in tag-store entries for different policies. Since ARW policies are adopted on LLC and DRAM cache, the associativity is generally larger than eight. Under such assumption, ARW policies incur less storage overhead than LRU and CCFLRU, and have good scalability for high-associativity cache. When taking update logic into consideration, ARW policies are simpler than LRU and CCFLRU. CCFLRU is difficult to implement because it has to distinguish four categories of data lines: cold-clean, cold-dirty, hot-clean, hot-dirty. In addition, it has to traverse the whole set several times to choose the victim, and takes longer time and consumes more power. Thus, it is not a feasible cache-level replacement policy designed for frequent accesses. ARW policies, on the contrary, is simple to implement and easy to be integrated into cache controllers. Moreover, the design changes for ARW policies are not on the critical path and thus do not affect the cache access time.

Table 7 Storage overhead

Replacement Policy	Hardware Overhead*
LRU	$n \cdot \log_2 n$
CCFLRU	$n \cdot \log_2 n + n$
3bit ARW-B	$3n$
3bit ARW-I	$3n$
2bit ARW-I	$2n$

* Assuming an n-way set associativity cache, HW overhead is measured in number of bits required per cache set.



(a) T1955 (b) T1982

(c) T5555 (d) T5582

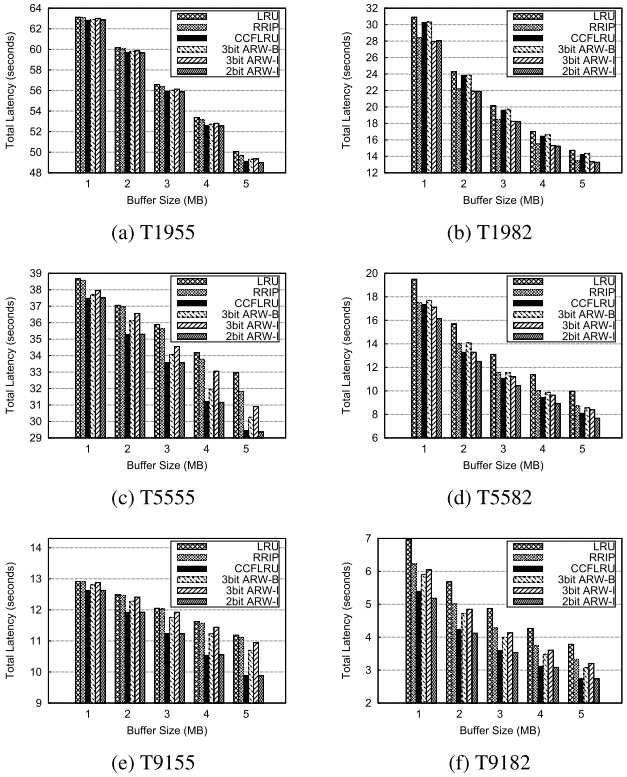
(e) T9155 (f) T9182

Fig. 10 The number of write operations for synthetic traces under various buffer sizes.

4.4 Results under NAND Flash

4.4.1 Results on Synthetic Traces under Various Buffer Sizes

Figure 10 shows the number of write operations with the synthetic traces under various buffer sizes. It can be observed that all ARW policies can reduce write traffic compared to LRU baseline. 2bit ARW-I provides better performance than all other policies, and CCFLRU takes the second place. The trends of results shown here are consistent with those under PCM-based organization and the explanations of the results are also similar to what we have discussed before. It can also be observed that the performance gap between 2bit ARW-I and CCFLRU gets larger for workloads with good locality (e.g., T1982, T5582 and T9182) than those with poor locality, which means that compared to CCFLRU, 2bit ARW-I can better exploit the property of lo-



(a) T1955 (b) T1982

(c) T5555 (d) T5582

(e) T9155 (f) T9182

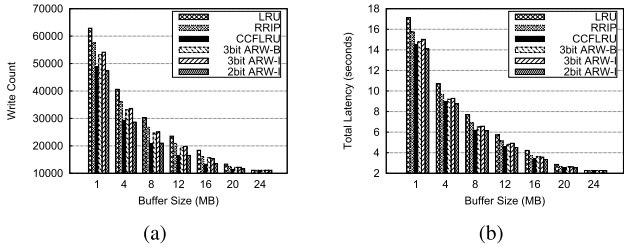
Fig. 11 Total memory access latency for synthetic traces under various buffer sizes.

cality to further reduce write traffic. In addition, according to results of T1982, T5555 and T9182, we can observe that the more percents of write operations in the trace, the more reductions can be achieved, which confirms the effectiveness of our proposal under a large amount of write traffic.

Figure 11 shows the overall access latency for the synthetic traces under various buffer sizes. The trends of the latency results are quite similar to those of write traffic results shown in Fig. 10. It can be observed that 2bit ARW-I outperforms all other policies. The reduction in latency benefits from two sources, one is the reduction in the number of write operations, and the other is the simplicity of 2bit ARW-I policy, leading to reduced access latency. Note that we only need to maintain negligible 2 bits or 3 bits per each block under ARW policies, while CCFLRU, on the contrary, has to manage two LRU lists and four categories by using much more bits to tag each block. Thus CCFLRU is not only difficult to implement and maintain but also takes more time for operating than 2bit ARW-I.

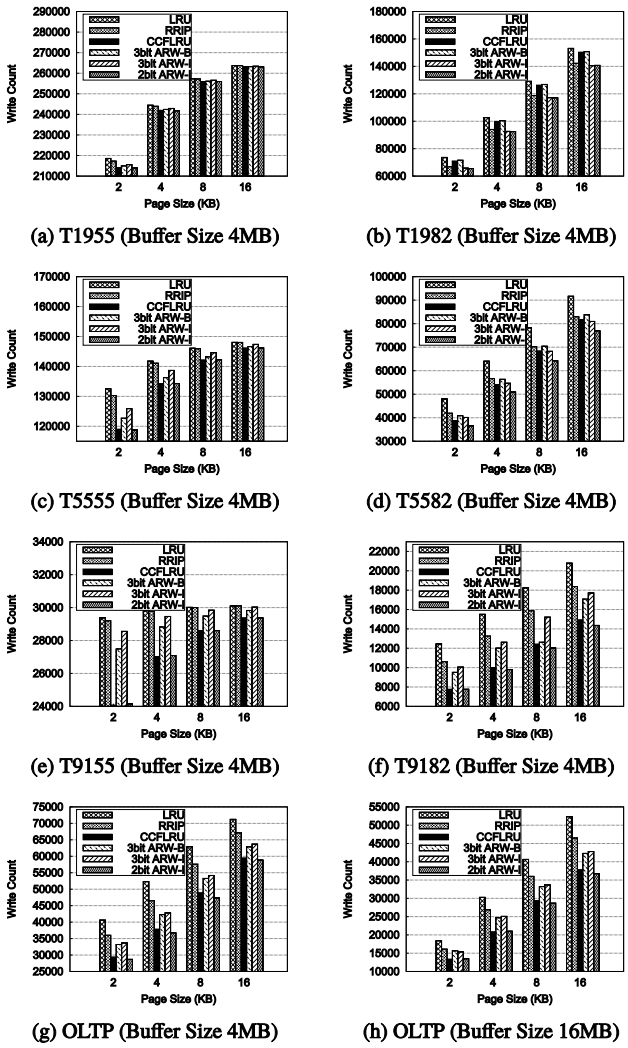
4.4.2 Results on Real OLTP Trace under Various Buffer Sizes

Figure 12 shows the number of write operations and the overall access latency for the real-world OLTP traces under various buffer sizes. Since real-world workloads generally display good locality, the trends of results shown here are similar to those under synthetic workloads with good locality. As the results shown before, 2bit ARW-I provides the best performance. Compared to LRU baseline, RRIP, CCFLRU, 3bit ARW-B and 3bit ARW-I, 2bit ARW-I can reduce the number of write operations by 8.4%, 22.2%, 15.4%, 3.9% and 24.8% respectively, and can reduce overall latency by 9.1%, 15.2%, 13.6%, 12.4% and 17.9% respectively.



(a) (b)

Fig. 12 The number of write operations (a) and the total memory access latency (b) for OLTP traces under various buffer sizes.



(a) T1955 (Buffer Size 4MB) (b) T1982 (Buffer Size 4MB)

(c) T5555 (Buffer Size 4MB) (d) T5582 (Buffer Size 4MB)

(e) T9155 (Buffer Size 4MB) (f) T9182 (Buffer Size 4MB)

(g) OLTP (Buffer Size 4MB) (h) OLTP (Buffer Size 16MB)

Fig. 13 Total memory access latency under various page sizes.

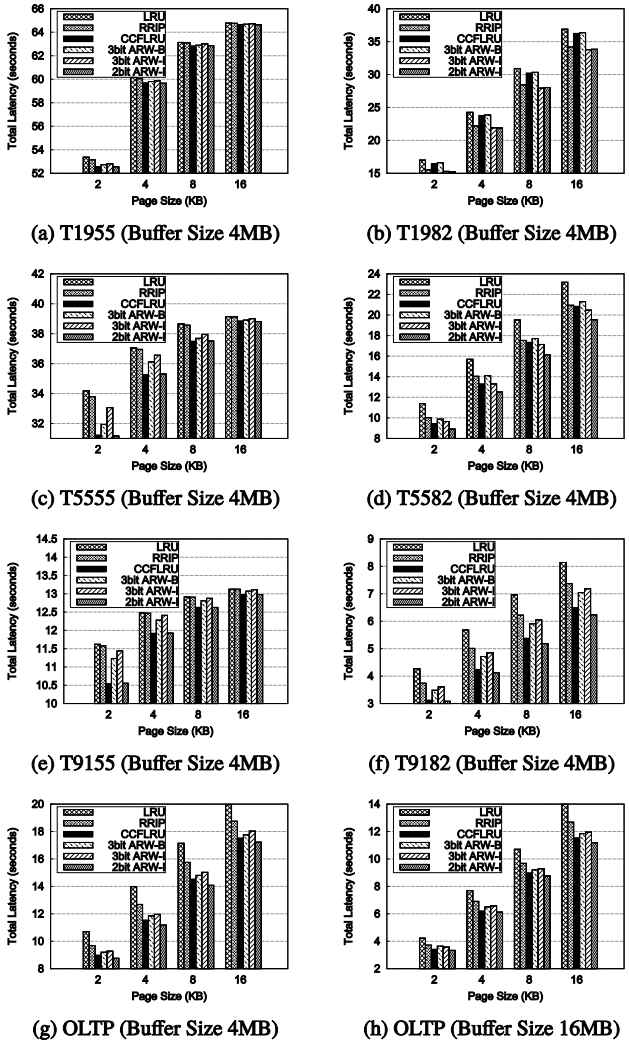


Fig. 14 Total memory access latency under various page sizes.

4.4.3 Results under Various Page Sizes

Figure 13 shows the number of write operations with the synthetic traces and real-world OLTP traces under various page sizes. It can be observed that 2bit ARW-I generally achieves the best performance. It also shows that under a certain buffer size (e.g., 4MB or 16MB in our setting), when the page size grows larger, the number of write counts increases. The possible reason is that smaller pages result in less amount of internal fragmentation, making the buffer get better utilized, which hence can reduce the number of page faults and writebacks. In addition, a smaller page size allows each page to match program locality more accurately [31]. However, decreasing the page size increases the number of pages and hence increases the overhead to operate the pages (e.g., page table size). Please note that the decision of best page size is not within the scope of this paper. Our results show that in most cases, 2bit ARW-I outperforms all other policies under the page sizes ranging from 2KB to 16KB, confirming the effectiveness of our proposal.

Figure 14 shows the overall access latency under various page sizes, indicating that 2bit ARW-I outperforms all other policies. The trends of the latency results are quite similar to those of write traffic results shown in Fig. 13. In addition, it also shows that as the page size grows larger, the access latency increases. The reason is that since increasing page sizes results in more write traffic (as shown in Fig. 13), larger access latency is hence observed due to the high latency of the write operations for NAND Flash.

5. Related Work

A variety of memory organizations together with optimization techniques have been proposed to reduce the performance overhead incurred by writing PCM. Hybrid architectures [10], [18], [32] work together with buffers and intelligent scheduling to bridge the latency gap. Qureshi et al. [10] adopted a smaller DRAM acting as a cache for a large PCM main memory. Qureshi et al. [19] proposed adaptive Write Cancelation policies combined with Write Canceling policies to improve read performance. Wu [33] et al. proposed read-write aware Hybrid Cache Architecture, where a single level cache can be partitioned into read region (SRAM) and write region (PCM or MRAM). Qureshi et al. [10] proposed lazy write organization, which doesn't write back unmodified page in DRAM. Flip-N-Write [34] examines new word with original word to reduce the number of bits to write. Most of them are not general techniques, requiring either specific organization or even instruction extension. Our technique is orthogonal to most of these works and can be combined to achieve better performance.

Though there are some techniques proposed to decrease write overhead on PCM via replacement policies, most of them are employed on PCM or DRAM with the page or even coarser grained block as the replacement unit. Thus, we only describe the fine-grained policies as ARW which are adopted in LLC. Write-back-aware cache partitioning (WCP) and write-queue balancing (WQP) replacement policy are designed aiming to partition cache to different cores to reduce writebacks [35]. A write-back-aware dynamic cache management technique (WADE) [4] is proposed to predict frequently written back blocks and dynamically adjust the cache size [36]. Cache replacement algorithms are proposed in [37], which combines two or more replacement policies, incurring significant hardware overhead. Compared to these methods, ARW is more lightweight and can partition the cache implicitly without any parameter setting or organization modification.

Flash-aware replacement policies have been proposed to reduce the number of writes to SSD. CFLRU [38] maintains a clean-first region at one end of LRU list, where clean pages are always selected as victims over dirty pages. LRU-WSR [39] is based on LRU and Second Chance, which evicts clean and cold-dirty pages and keeping hot-dirty pages in buffer as long as possible. REF [6] has a victim window similar to CFLRU. However, it doesn't distinguish clean and dirty states of pages. Cold-Clean-First

LRU (CCFLRU) [12] policy enhanced CFLRU and LRU-WSR methods by differentiating clean pages into cold and hot ones, and evicting cold clean pages first and delaying the eviction of hot clean pages. AD-LRU [21] sets the lowest limit for the size of the cold list. FOR [22] combines inter-operation distance and operation recency to determine the hotness of an operation, which are used to calculate the weight of each page to evict. These NAND Flash-based policies are designed in page or coarser granularity, and are too complex to be employed by on-chip cache. In contrast, ARW policies are lightweight schemes and easier to be integrated into existing cache structure or off-chip buffer at any data granularity with negligible overhead.

6. Conclusions

In this article, we propose a set of memory replacement policies called ARW in the context of NVM-based memory hierarchy. ARW policies improve system performance and extend the lifetime of PCM and NAND Flash by reducing the writeback requests. The basic version of ARW keeps dirty blocks in the cache (or buffer), while the improved version can further improve performance by evicting the single-use dirty blocks. When ARW policies are employed by LLC and DRAM cache, full-system simulation results on an 8-core CMP show that ARW can significantly reduce the write traffic to PCM without degrading the system performance. When ARW policies are adopted by on-disk buffer in SSD, the simulation results shows similar trends, and both the number of write operations and the overall latency can be reduced impressively. Moreover, ARW policies incur negligible hardware overhead or runtime overhead. Therefore, ARW policies are feasible and can be easily integrated into existing hardware or SSD management system.

Acknowledgments

We appreciate Prof. Gerhard Weikum, the Research Director at the Max-Planck Institute for Informatics (MPII) in Saarbruecken, Germany, for providing the real OLTP trace. This work is supported by the National Natural Science Foundation of China (No. 61300014, No. 61572059), the State Key Program of National Natural Science of China (Grant No. 71531001), the Basic Science Research, the State's Key Project of Research and Development Plan (2016YFC1000307), and Industry-University-Research Cooperation Project of Guangdong Province (no. 2016B090921001).

References

[1] X. Zhang, Q. Hu, D. Wang, C. Li, and H. Wang, "A read-write aware replacement policy for phase change memory," International Workshop on Advanced Parallel Processing Technologies, pp.31–45, Springer, 2011.

[2] R. Bez and A. Pirovano, "Non-volatile memory technologies: emerging concepts and new materials," Materials Science in Semiconductor Processing, vol.7, no.4, pp.349–355, 2004.

[3] X. Pan and R. Teodorescu, "Using STT-RAM to enable energy-efficient near-threshold chip multiprocessors," Proceedings of the 23rd international conference on Parallel architectures and compilation, pp.485–486, ACM, 2014.

[4] Z. Wang, S. Shan, T. Cao, J. Gu, Y. Xu, S. Mu, Y. Xie, and D.A. Jiménez, "WADE: Writeback-aware dynamic cache management for nvm-based main memory system," ACM Transactions on Architecture and Code Optimization (TACO), vol.10, no.4, p.51, 2013.

[5] T. Kgil, D. Roberts, and T. Mudge, "Improving nand flash based disk caches," Computer Architecture, 2008. ISCA'08. 35th International Symposium on, pp.327–338, IEEE, 2008.

[6] D. Seo and D. Shin, "Recently-evicted-first buffer replacement policy for flash storage devices," IEEE Trans. Consum. Electron., vol.54, no.3, pp.1228–1235, 2008.

[7] S. Raoux, G.W. Burr, M.J. Breitwisch, C.T. Rettner, Y.-C. Chen, R.M. Shelby, M. Salinga, D. Krebs, S.-H. Chen, H.-L. Lung and C.H. Lam, "Phase-change random access memory: A scalable technology," IBM Journal of Research and Development, vol.52, no.4.5, pp.465–479, 2008.

[8] Intel, "Intel, stmicroelectronics deliver industry's first phase change memory prototypes," 2008.

[9] X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony, and Y. Xie, "Hybrid cache architecture with disparate memory technologies," ACM SIGARCH computer architecture news, pp.34–45, ACM, 2009.

[10] M.K. Qureshi, V. Srinivasan, and J.A. Rivers, "Scalable high performance main memory system using phase-change memory technology," ACM SIGARCH Computer Architecture News, vol.37, no.3, pp.24–33, 2009.

[11] Numonyx, "The basics of phase change memory technology," http://www.signallake.com/innovation/PhaseChangeMemory.pdf. Accessed: 2016-08-22.

[12] Z. Li, P. Jin, X. Su, K. Cui, and L. Yue, "CCF-LRU: a new buffer replacement algorithm for flash memory," IEEE Trans. Consum. Electron., vol.55, no.3, pp.1351–1359, 2009.

[13] D.-H. Kang, J.-H. Lee, J. Kong, D. Ha, J. Yu, C.Y. Um, J.H. Park, F. Yeung, J.H. Kim, W.I. Park, Y.J. Jeon, M.K. Lee, Y.J. Song, J.H. Oh, G.T. Jeong and H.S. Jeong, "Two-bit cell operation in diode-switch phase change memory cells with 90nm technology," 2008 Symposium on VLSI Technology, pp.98–99, IEEE, 2008.

[14] T. UltraSPARC, "Supplement to the UltraSPARC architecture 2007," 2006.

[15] M.K. Qureshi, A. Jaleel, Y.N. Patt, S.C. Steely, and J. Emer, "Adaptive insertion policies for high performance caching," ACM SIGARCH Computer Architecture News, pp.381–391, ACM, 2007.

[16] A. Jaleel, K.B. Theobald, S.C. Steely Jr, and J. Emer, "High performance cache replacement using re-reference interval prediction (RRIP)," ACM SIGARCH Computer Architecture News, pp.60–71, ACM, 2010.

[17] X. Zhang, L. Chongmin, L. Zhenyu, W. Haixia, W. Dongsheng, and T. Ikenaga, "A novel cache replacement policy via dynamic adaptive insertion and re-reference prediction," IEICE Trans. Electron., vol.94, no.4, pp.468–476, 2011.

[18] T.J. Ham, B.K. Chelepalli, N. Xue, and B.C. Lee, "Disintegrated control for energy-efficient and heterogeneous memory systems," High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on, pp.424–435, IEEE, 2013.

[19] M.K. Qureshi, M.M. Franceschini, and L.A. Lastras-Montano, "Improving read performance of phase change memories via write cancellation and write pausing," HPCA-16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture, pp.1–11, IEEE, 2010.

[20] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," ACM SIGARCH computer architecture news, pp.14–23, ACM, 2009.

[21] P. Jin, Y. Ou, T. Härder, and Z. Li, "AD-LRU: An efficient buffer replacement algorithm for flash-based databases," Data & Knowledge Engineering, vol.72, pp.83–102, 2012.

[22] Y. Lv, B. Cui, B. He, and X. Chen, "Operation-aware buffer management in flash-based systems," Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pp.13–24, ACM, 2011.
[23] R. Kalla, B. Sinharoy, W.J. Starke, and M. Floyd, "POWER7: IBM's next-generation server processor," IEEE micro, vol.30, no.2, pp.7–15, 2010.
[24] R. Wang, Z. Chen, N. Xiao, M. Zhang, and W. Dong, "CAWR: Buffer replacement with channel-aware write reordering mechanism for ssds," ETRI Journal, vol.37, no.1, pp.147–156, 2015.
[25] "Simics full system simulator," http://www.windriver.com/products/simics/. Accessed: 2016-08-22.
[26] "Wisconsin multifacet gems simulator," http://www.cs.wisc.edu/gems/. Accessed: 2016-08-22.
[27] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," ACM SIGARCH Computer Architecture News, pp.24–36, ACM, 1995.
[28] C. Bienia, S. Kumar, J.P. Singh, and K. Li, "The PARSEC benchmark suite: characterization and architectural implications," Proceedings of the 17th international conference on Parallel architectures and compilation techniques, pp.72–81, ACM, 2008.
[29] X. Su, P. Jin, X. Xiang, K. Cui, and L. Yue, "Flash-DBSim: a simulation tool for evaluating flash-based database algorithms," Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on, pp.185–189, IEEE, 2009.
[30] M.G. Baker, J.H. Hartman, M.D. Kupfer, K.W. Shirriff, and J.K. Ousterhout, "Measurements of a distributed file system," ACM SIGOPS Operating Systems Review, pp.198–212, ACM, 1991.
[31] P.B. Galvin, G. Gagne, and A. Silberschatz, Operating system concepts, John Wiley & Sons, 2013.
[32] G. Dhiman, R. Ayoub, and T. Rosing, "PDRAM: a hybrid pram and dram main memory system," Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE, pp.664–669, IEEE, 2009.
[33] X. Wu, J. Li, L. Zhang, E. Speight, and Y. Xie, "Power and performance of read-write aware hybrid caches with non-volatile memories," 2009 Design, Automation & Test in Europe Conference & Exhibition, pp.737–742, IEEE, 2009.
[34] S. Cho and H. Lee, "Flip-N-Write: a simple deterministic technique to improve pram write performance, energy and endurance," 2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp.347–357, IEEE, 2009.
[35] M. Zhou, Y. Du, B. Childers, R. Melhem, and D. Mossé, "Write-back-aware partitioning and replacement for last-level caches in phase change main memory systems," ACM Transactions on Architecture and Code Optimization (TACO), vol.8, no.4, pp.1–21, 2012.
[36] S.M. Khan, Z. Wang, and D.A. Jimenez, "Decoupled dynamic cache segmentation," IEEE International Symposium on High-Performance Comp Architecture, pp.1–12, IEEE, 2012.
[37] R. Rodríguez-Rodríguez, F. Castro, D. Chaver, L. Pinuel, and F. Tirado, "Reducing writes in phase-change memory environments by using efficient cache replacement policies," Proceedings of the Conference on Design, Automation and Test in Europe, pp.93–96, EDA Consortium, 2013.
[38] S.-y. Park, D. Jung, J.-u. Kang, J.-s. Kim, and J. Lee, "CFLRU: a replacement algorithm for flash memory," Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems, pp.234–241, ACM, 2006.
[39] H. Jung, H. Shim, S. Park, S. Kang, and J. Cha, "LRU-WSR: integration of LRU and writes sequence reordering for flash memory," IEEE Trans. Consum. Electron., vol.54, no.3, pp.1215–1223, 2008.



Xi Zhang received his B.S. degree from Harbin Institute of Technology, China in 2006, and his Ph.D. degree from Tsinghua University in 2012. He is now an assistant professor in School of Computer Science at Beijing University of Posts and Telecommunications (BUPT). He is also in Key Laboratory of Trustworthy Distributed Computing and Service of Ministry of Education. He has worked in the areas of multicore architecture, non-volatile memory (NVM) and data mining.



Xinning Duan received her B.S. degree from BUPT, China in 2016. She is now a master student in BUPT. She has worked in the areas of computer architecture and data mining.



Jincui Yang received her B.S. degree from XiDian University, China in 1991, and her Ph.D. degree from BUPT in 2013. She is now an assistant professor in School of Software Engineering at BUPT. She has worked in the areas of software engineering, trustworthy computing and services and The Internet of Things.



Jingyuan Wang received his B.S. degree from Beijing Institute of Technology, China in 2006, and his Ph.D. degree from Tsinghua University in 2011. He is now an associate professor in School of Computer Science at Beihang University. He has worked in the areas of computer architecture, network protocol and urban computing.