A Novel Page Replacement Algorithm for the Hybrid Memory Architecture Involving PCM and DRAM

Kaimeng Chen[1], Peiquan Jin[1,2], and Lihua Yue[1,2]

[1] School of Computer Science and Technology,
University of Science and Technology of China, Hefei, China
[2] Key Laboratory of Electromagnetic Space Information, Chinese Academy of Sciences,
Hefei, China
jpq@ustc.edu.cn

Abstract. Recently, the development of phase change memory (PCM) motivates new hybrid memory architectures that consist of PCM and DRAM. An important issue in such hybrid memory architectures is how to manage the pages resisting in heterogeneous memories. For example, when a requested page is missing in the hybrid memory and the memory has no free spaces, what pages in which type of memory (PCM or DRAM) should be replaced? This problem is much different from traditional buffer replacement management, where they do not consider the special properties of different types of memories. In particular, differing from DRAM, PCM is non-volatile but it has lower access speeds than DRAM. Further, PCM has a limited write endurance which implies that it cannot be written endlessly. Therefore, we have to design a new page replacement algorithm that can not only maintain a high hit ratio as traditional algorithms do but also can avoid frequent writes to PCM. In this paper, aiming to provide a new solution to the page replacement problem in PCM/DRAM-based hybrid memories, we propose a new algorithm called MHR-LRU (Maintain-hit-ratio LRU). The objective of our algorithm is to reduce PCM writes while maintaining a high hit ratio. Specially, it keeps recently updated pages in DRAM and performs page migrations between PCM and DRAM. The migrations take into account both page access patterns and the influences of page faults. We conduct trace-driven experiments and compared our proposal with some existing algorithms including LRU, LRU-WPAM, and CLOCK-DWF. The results show that our proposal is able to efficiently reduce PCM writes without degrading the hit ratio. Thus, our study offers a better solution for the page replacement issue in PCM/DRAM-based hybrid memory systems than previous approaches.

Keywords: Page replacement, Phase change memory, Hybrid memory.

1 Introduction

Phase change memory (PCM) is one of the most promising non-volatile memories. PCM is byte-addressable and a type of random-access memories. Compared with DRAM, PCM has the advantages of durability, scalability, and low energy

C.-H. Hsu et al. (Eds.): NPC 2014, LNCS 8707, pp. 108–119, 2014.

consumption. Thus, many researchers have proposed to incorporate PCM into the memory hierarchy of computer systems [1-3]. However, two problems of PCM make it difficult to totally replace DRAM in current computer systems. First, the write latency of PCM is about 6 to10 times slower than that of DRAM. Second, PCM has a worn-out problem because each PCM cell has limited write endurance. Thus, PCM is not suitable for update-intensive applications. As a summary, Table 1 shows a comparison between DRAM and PCM.

Table 1. Comparison between PCM and DRAM

Attributes	DRAM	PCM
Durability	Volatile	Non-volatile
Read Latency	50 ns	50 ns
Write latency	20 – 50 ns	350 – 1000 ns
Read Energy	~ 0.1 nJ/b	~ 0.1 nJ/b
Write Energy	~ 0.1 nJ/b	~ 0.5 nJ/b
Idle Power	~ 1.3 W/GB	~ 0.05 W
Density	Low	High (~ 4X DRAM)
Endurance	∞	10^8 for write

Therefore, a more practical way to utilize PCM in memory architectures is to use PCM and DRAM and thus to construct a hybrid memory architecture [4, 5]. Generally, there are two architectures to integrate PCM in DRAM-based main memory, namely *DRAM cache* architecture and *hybrid memory architecture*, as shown in Fig. 1. The *DRAM cache architecture* uses PCM as main memory and uses DRAM as the cache of PCM [4]. The DRAM cache is hidden to the operation system, like the L1 and L2 caches for CPU. The *hybrid memory architecture* puts PCM and DRAM at the same level in main memory [5]. The hybrid memory is regarded as the union of DRAM and PCM, and both of their storage capacities are used as main memory. In this situation, all the pages in DRAM and PCM are managed by the operation system.

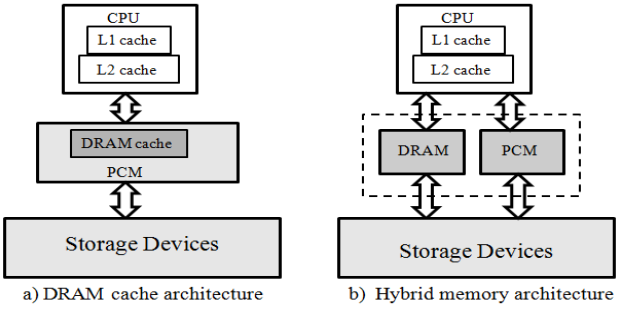


Fig. 1. Memory architectures consists of PCM and DRAM

This paper focuses on the *hybrid memory architecture*. In particular, we concentrate on the page replacement problem for the hybrid memory architecture. Traditional page replacement algorithms are designed for DRAM-only main memory

architecture. They are not suitable for the hybrid memory architecture because PCM and DRAM have different characteristics and the page replacement algorithms have to be aware of these differences. So far, a few page replacement algorithms for the hybrid memory architecture are proposed; some of them focus on PCM and DRAM [6, 7] while others are on flash memory and HDD [10, 11]. The similar idea of the existing algorithms is to keep write-intensive pages in DRAM and to let read-intensive pages in PCM so that DRAM can absorb most writes. For this purpose, specific page migration schemes are introduced in previous algorithms. The basic process of a page migration is as follows: when a page request comes to the memory, the page migration algorithm determines whether the requested page needs to be moved according to its access pattern. The read-intensive page in DRAM is moved to PCM and the write-intensive page in PCM is moved to DRAM.

However, there are two problems in the existing algorithms for hybrid memory systems. First, these algorithms always place pages read from disk in PCM and move read-intensive pages in DRAM to PCM. Because page placement and migration also incur writes to PCM, always caching read-intensive pages in PCM may lead to additional writes to PCM especially under read-intensive workloads. Second, moving a page between PCM and DRAM has to consider the problem that the memory is full. To accommodate the moved page in the target medium, the previous algorithms have to choose a victim page to release its space [6, 7]. This has an impact on the hit ratio of memory request. Thus, compared with conventional algorithms, the existing algorithms for hybrid memory usually introduce more page faults when processing memory requests.

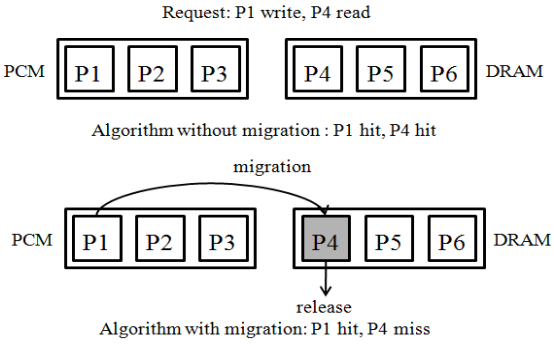


Fig. 2. A page fault occurs because of page migration

Fig. 2 shows an example that how page migration can affect the hit ratio. Initially, both PCM and DRAM are full. For the algorithm with page migration for hybrid memory, when a page P1 in PCM is migrated to DRAM, P4 in DRAM is selected as a victim to make space for P1. Then, the read request to P4 misses. For the conventional algorithm without page migration, both requests to P1 and to P4 hit in the memory. Because the access latency of hard disk is much slower than the write access latency of PCM, decreasing hit ratio to reduce write access count on PCM may lower the overall performance of hybrid memory.

In this paper, we present an efficient page replacement algorithm called MHR-LRU (Maintain-hit-ratio LRU) for PCM/DRAM-based hybrid memory. The

algorithm aims to maintain a high hit ratio and to reduce PCM writes for the hybrid memory. Differing from previous algorithms, our algorithm does not move pages between PCM and DRAM when page requests arrive. Instead, we perform page migrations when page faults occur. Thus, the page migrations in our algorithm need not to release extra pages. This is helpful to maintain a high hit ratio, because releasing extra pages will lower the hit ratio. Besides, MHR-LRU places write-intensive pages in DRAM to reduce PCM writes. Under read-intensive workloads, DRAM is efficiently used to absorb most read requests and to limit the number of writes to PCM triggered by page placement and page migration.

We perform trace-driven experiments in a hybrid memory simulation environment to evaluate the performance of MHR-LRU. We use different types of workloads and conduct comparisons with other algorithms including LRU, LRU-WPAM [6], and CLOCK-DWF [7]. The results show that our algorithm is able to maintain a high hit ratio for different workloads and outperform the other three competitors considering PCM writes.

The remainder of this paper is organized as follows. In Section 2, we sketch the related work. In Section 3, we present the MHR-LRU page replacement algorithm for the hybrid memory architecture. Section 4 describes the details about the experiments and the performance evaluation results. Finally, Section 5 concludes the paper.

2 Related Work

Conventional page replacement algorithms have been designed for DRAM-based main memory with uniform access latency and unlimited write endurance. Hit ratio is the key metric to evaluate the performance.

LRU (Least Recently Used) is a conventional page replacement algorithm that has been widely used. LRU aligns all pages in memory in order of their most recent reference times. When a page fault occurs and the buffer pool is full, the least recently used page in memory is selected as a victim. LRU has also been widely used for buffer management over new types of storage media such as flash memory [12, 13].

Page replacement algorithms for hybrid main memory as shown in Fig. 1(b) should consider not only the hit ratio but also the number of PCM writes because PCM has the long write latency and limited endurance.

LRU-WPAM (LRU-With-Prediction-And-Migration) is an LRU-based page replacement algorithm for hybrid main memory [6]. The algorithm aligns all pages in hybrid main memory as a LRU queue, and use four monitoring queues: a DRAM read queue, a DRAM write queue, a PCM read queue and a PCM write queue. Each page is retained into both LRU list and one of the four queues according to its access pattern and located memory type. To measure the access pattern of a page in hybrid memory, the algorithm provides a weight value for each page. Each time a page hits in the memory, the page's weight is calculated again according to the type of this access request, if its weight value is above the threshold, the page will be migrated. If the memory need to choose a victim to release for receiving the migrated page, DRAM choose the least recently used page in DRAM read queue and PCM choose the least recently used page in PCM write queue.

CLOCK-DWF is a CLOCK-based page replacement algorithm for hybrid main memory [7]. When a page fault occurs, if the request is read, the page is put on PCM; otherwise the page is put on DRAM. When a page on PCM hits by write request, the page is migrated to DRAM. To get a free page frame while the memory is full, PCM use conventional CLOCK algorithm to select a victim page to release [8], but DRAM migrates a low write frequency page to PCM.

Both LRU-WPAM and CLOCK-DWF release pages in page migration, this would causes hit ratio degradation. Both of the two algorithms force read-bound pages to place on PCM, this may incurs higher PCM write count than conventional algorithms. For these problem, our study present a new method to reduce PCM write count without sacrificing hit ratio by merging page migration into page replacement process and just limiting write-bound pages to the DRAM. The details will be given in Section 3.

3 The MHR-LRU Algorithm

In this section, we present the details of the MHR-LRU algorithm for hybrid main memory. MHR-LRU aims to reduce the writes to PCM without degrading hit ratio so as to improve the overall performance of hybrid main memory. In order to accomplish this goal, we design the scheme that performs the page migration when page replacement occurs to make write-intensive pages on PCM.

3.1 The Main Idea

The main idea of MHR-LRU algorithm is described as follows:

(1) The algorithm use LRU list to manage pages together in hybrid main memory. All pages in hybrid memory are aligned in order of their most recent reference time. When a page fault occurs, the page in the LRU position will be selected as victim no matter where it locates.

(2) The algorithm uses a special data structure called DWL (DRAM Write-aware LRU list) to manage pages in DRAM. DWL aligns all pages in DRAM in order of their most recent write reference time.

(3) When a page fault occurs and the victim has been selected, MHR-LRU check the page access type and the victim's location, if the page's access request is write and the victim is located on PCM, MHR-LRU perform page migration: the victim on PCM is released and the page in the LRU position of DWL is migrated to PCM, then the requested page is put on DRAM. By doing so, MHR-LRU can get the following benefits: First, the page migration does not cause extra page release, so it does not affect the hit ratio; Second, since putting a page from disk on PCM and migrating a page in DRAM to PCM incur the same amount of write on PCM, compared with putting the requested page on PCM then performing write operation on it, migrating an in-DRAM page to PCM and putting the requested page on DRAM for write operation can immediately reduce the amount of write on PCM; Third, according to principle of temporal locality, page with the most recently write reference has a higher possibility to be write again than the page in the LRU position of DWL, this page migration can reduce the number of future write operation on PCM.

3.2 The Detailed Algorithm

Fig. 3 shows the detailed algorithm of MHR-LRU. If a requested page is found in DRAM, the page is also maintained in both the LRU list and DWL. Hence, we check the type of this page request. If it is a read request, we move the page to the MRU position of the LRU list. If it is a write request, we move the page to the MRU position of the LRU list and the MRU position of DWL (Line 1 ~ 7). If the requested page is in PCM, the page is maintained in LRU list, we move the page to the MRU position of the LRU list (Line 8 ~ 10).

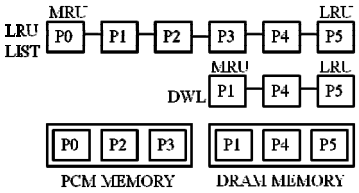
Algorithm. *MHR-LRU*(request *q*)

Input: a page request *q*
Output: a reference to the requested page
Preliminary: (1) *L* is the LRU list of the memory, DWL is the DRAM write-aware list.
(2) *p* is the requested page.

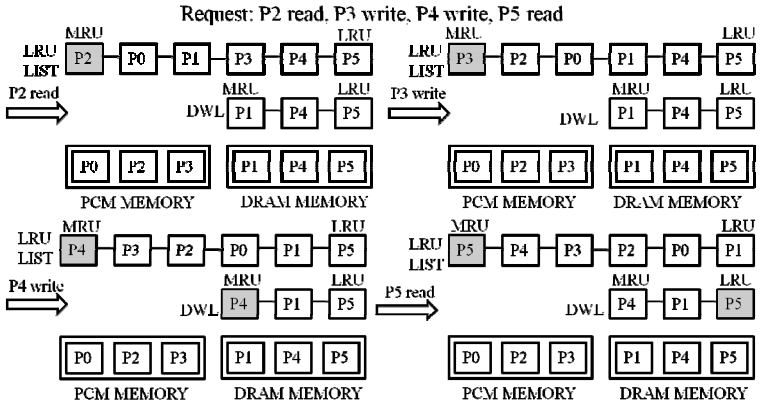
```
1:    if p is in DRAM then
2:        if q is a read request then
3:            move p to MRU position of L
4:        else
5:            move p to MRU position of L;
6:            move p to MRU position of DWL;
7:        return a reference to p;
8:    else if p is in PCM then
9:        move p to MRU position of L;
10:       return a reference to p;
      /*page fault occurs*/
11:   else
12:       if there is a free frame in hybrid main memory then
13:           put p into the free frame;
14:       else
15:           get victim from LRU position of L;
16:           if victim is in PCM and q is a write request then
17:               get page m from LRU position of DWL;
18:               release victim and migrate m to PCM;
19:               put p into the free frame of DRAM;
20:           else
21:               release victim and put p into the free frame;
22:       insert p into MRU position of L;
23:       if p is in DRAM then
24:           if q is a write request then
25:               insert p into MRU position of DWL;
26:           else
27:               insert p into LRU position of DWL;
28:   return a reference to p;
```

Fig. 3. The detailed algorithm of MHR-LRU

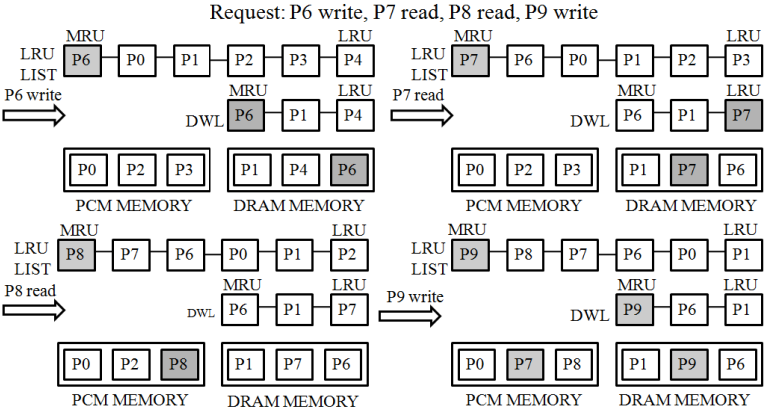
If a page is missing, we have to find a free frame to cache the requested page. If the memory has free spaces, we put the requested page into a randomly-selected free frame (Line 12 ~ 13). If the memory is full, we select the page in the LRU position of the LRU list as victim, if the victim is in PCM and the page's request is a write request, the victim is released. We do not put the requested page in PCM but move the page in the LRU position of DWL to PCM, and put the requested page into



(a) Buffer initial situation for MHR-LRU algorithm



(b) The situation that pages in PCM and DRAM hit by read request and write request



c) The situation that page replacement and page migration occur.

Fig. 4. An example of the MHR-LRU algorithm

DRAM. Otherwise, we just evict the victim for the requested page (Line 14 ~ 21). After putting the requested page into the hybrid memory, we insert the requested page to the MRU position of the LRU list. If the requested page is in DRAM, we insert it to the DWL according to its request type. If the page's request is a write request, it is inserted to the MRU position of DWL, else it is inserted to the LRU position of DWL (Line 22 ~ 28).

Fig. 4 gives an example of MHR-LRU. Fig. 4 (a) shows the initial state of the hybrid memory. The buffer contains P0, P1, P2, P3, P4, P5. P0, P2, P3 are in PCM, and P1, P4, and P5 are in DRAM. All pages are in LRU list and P1, P4, P5 are in DWL. Fig. 4 (b) shows the situation of page hits, and Fig. 4 (c) shows the situation of page faults.

Fig. 4 indicates that, when page faults occur, MHR-LRU selects and releases the least recently used page, which is similar to the LRU algorithm. This ensures that our algorithm can have the similar hit ratio as LRU. However, our algorithm does not release pages when page hits occurs, which is different from LRU-WPAM and CLOCK-DWF.

4 Performance Evaluation

In this section, we compare our algorithm with LRU, LRU-WPAM, and CLOCK-DWF. LRU is the reprehensive of traditional page replacement algorithms. LRU-WPAM and CLOCK-DWF are the recently proposed algorithms for the hybrid memory architecture.

4.1 Experimental Setup and Datasets

We use simulation experiments to evaluate our algorithm. We design the simulator for the hybrid memory architecture. In the experiments, the DRAM-to-PCM ratio is set to 1:4 because PCM density is expected to be four times higher than that of DRAM. Based on the simulator, the compared page replacement algorithms are implemented and trace-driven experiments are performed for performance evaluation. The parameters used in LRU-WPAM and CLOCK-DWF are the same as those in the original papers [6, 7].

We perform our simulation experiments with six types of synthetic traces. These traces are generated by DiskSim [9]. The characteristics of these traces are given in Table 2. The locality in Table 2, for example 80% / 20%, means that eighty percent of total references are focused on twenty percent of the pages.

4.2 Hit Ratios

Hit ratio is a key metric for the performance of page replacement algorithms. First, we compare the hit ratio of our algorithm with other three ones by varying the size of memory. We use the number of page faults to measure hit ratio. The results are shown in Fig. 5.

As Fig. 5 shows, when measures under the workloads with low localities (T9155, T5555, T1955), the hit ratios of LRU-WPAM and CLOCK-DWF are almost the same

Table 2. Six types of synthetic traces

Type	Total Reference	Different Pages Accessed	Read/Write Ratio	Locality
T9182	300,000	10,000	90% / 10%	80% / 20%
T9155	300,000	10,000	90% / 10%	50% / 50%
T5582	300,000	10,000	50% / 50%	80% / 20%
T5555	300,000	10,000	50% / 50%	50% / 50%
T1982	300,000	10,000	10% / 90%	80% / 20%
T1955	300,000	10,000	10% / 90%	50% / 50%

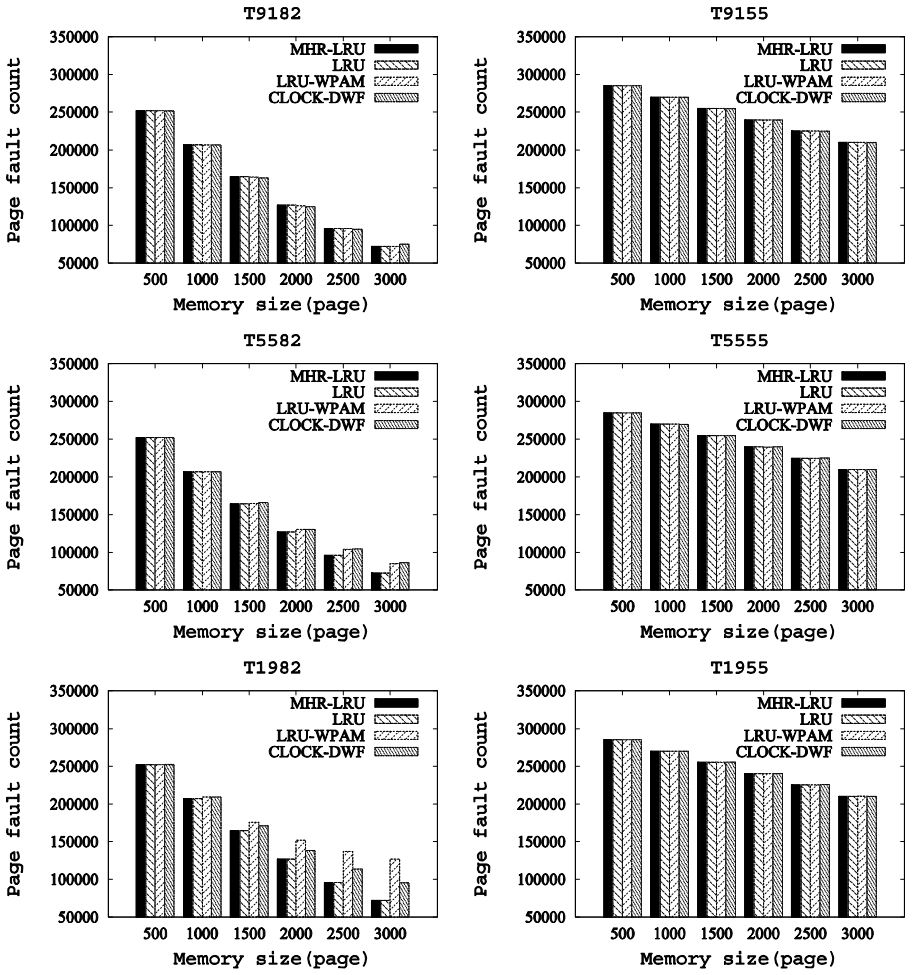
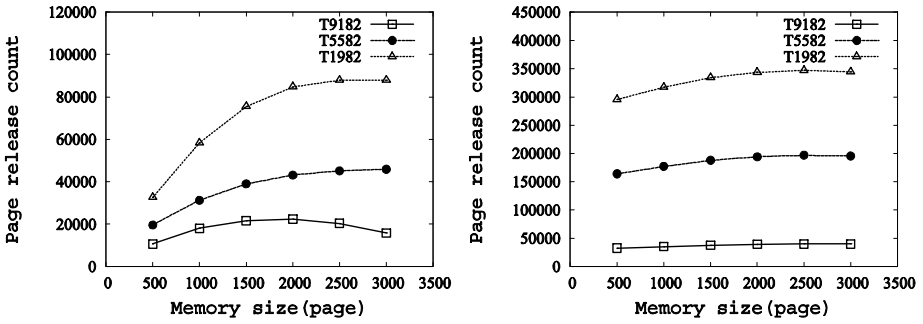


Fig. 5. Number of page faults under the six synthetic traces

as the hit ratio of LRU and MHR-LRU. Because all algorithms are based on temporal locality, a low locality access pattern leads to very similar hit ratios of four algorithms. When using the high-locality workloads (T9182, T5582, T1982), with the increase of the ratio of write operations in workloads, LRU-WPAM and CLOCK-DWF show higher number of page faults than LRU and MHR-LRU do.

LRU-WPAM and CLOCK-DWF usually release pages when they perform page migration. As shown in Fig. 2, this page release can introduce page fault. To show the relationship between page fault and page release, during the experiment, we collect page release count of LRU-WPAM and CLOCK-DWF under high-locality workloads that LRU-WPAM and CLOCK-DWF show higher number of page fault. When page migration occurs and one page in the target medium has been released, the page release count increases. The result is shown in Fig. 6. As Fig. 6 shows, for LRU-WPAM and CLOCK-DWF, workloads with high ratio of write operations cause more pages to be released because of page migrations. This consequently leads to a higher number of page faults.



(a) LRU-WPAM (b) CLOCK-DWF

Fig. 6. Number of page releases triggered by page migrations

4.3 Writes to PCM

The writes count to PCM is related to the overall write performance of the hybrid memory and the lifetime of PCM. In this section, we measure the number of write operations on PCM incurred by MHR-LRU in comparison with LRU, LRU-WPAM, and CLOCK-DWF.

Fig. 7 shows the number of PCM writes for LRU, LRU-WPAM, CLOCK-DWF, and MHR-LRU. MHR-LRU obtains less PCM writes than LRU-WPAM and CLOCK-DWF do in most cases. Compared with LRU, MHR-LRU reduces 17.45% of PCM writes on average, and reduces up to 34.1% of PCM writes. The write count reduction of LRU-WPAM is 1.01% on average, and the best result for write reduction is 8.69%. CLOCK-DWF is able to reduce averagely 9.82% more writes than LRU. Specially, MHR-LRU can still reduce 6.5% of PCM writes averagely in the worst case (under the workload T9155).

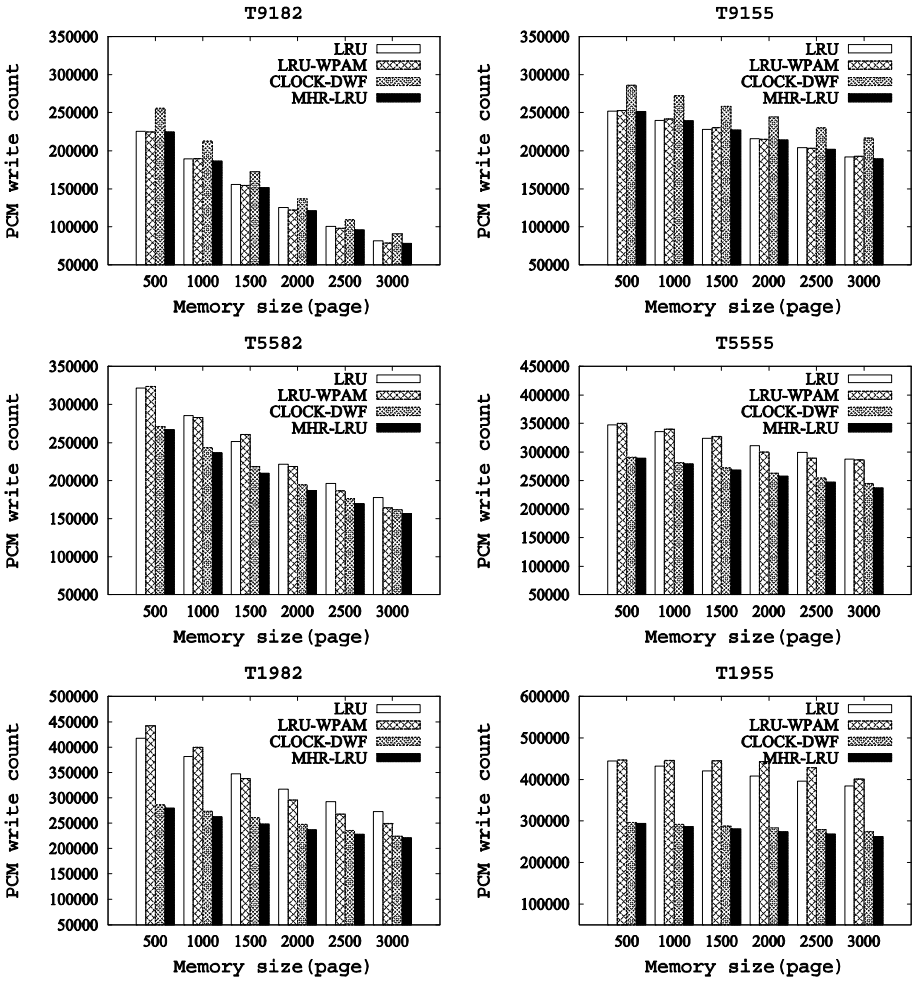


Fig. 7. PCM writes under the six synthetic traces

5 Conclusions

PCM has emerged as one of the most promising memories to be used in main memory hierarchy. A lot of studies propose to construct hybrid memory architectures involving PCM and DRAM to utilize the advantages of both media. In this paper, based on such hybrid memory architecture, we propose a new page replacement algorithm called MHR-LRU to handle the problems incurred by the hybrid memory architecture. MHR-LRU is able to maintain a high hit ratio and is able to reduce PCM writes effectively. We conduct trace-driven experiments in a simulation environment using six types of synthetic traces, and compare our algorithm with three competitors including LRU, LRU-WPAM, and CLOCK-DWF in terms of different metrics. The results show that our algorithm outperforms all the other algorithms.

Acknowledgement. This paper is supported by the National Science Foundation of China (No. 61073039, 61379037, and 61272317) and the OATF project funded by University of Science and Technology of China.

References

1. Burr, G.W., Kurdi, B.N., Campbell Scott, J., et al.: Overview of candidate device technologies for storage-class memory. IBM Journal of Research and Development 52(4-5), 449–464 (2008)
2. Freitas, R.F., Wilcke, W.W.: Storage-class memory: The next storage system technology. IBM Journal of Research and Development 52(4-5), 439–448 (2008)
3. Benjamin, C.: Lee, Engin Ipek, Onur Mutlu, et al., Architecting phase change memory as a scalable dram alternative. In: Proc. of ISCA, pp. 2–13 (2009)
4. Qureshi, M.K., Srinivasan, V., Rivers, J.A.: Scalable high performance main memory system using phase-change memory technology. In: Proc. of ISCA, pp. 24–33 (2009)
5. Dhiman, G., Ayoub, R.Z., Rosing, T.: PDRAM: a hybrid PRAM and DRAM main memory system. In: Proc. of DAC, pp. 664–669 (2009)
6. Seok, H., Park, Y., Park, K.W., et al.: Efficient page caching algorithm with prediction and migration for a hybrid main memory. ACM SIGAPP Applied Computing Review 11(4), 38–48 (2011)
7. Lee, S., Bahn, H., Noh, S.H.: Characterizing Memory Write References for Efficient Management of Hybrid PCM and DRAM Memory. In: Proc. of MASCOTS, pp. 168–175 (2011)
8. Corbato, F.J.: A Paging Experiment with the Multics System. In: Honor of P. M. Morse, pp. 217–228. MIT Press (1969)
9. Bucy, J.S., Schindler, J., Schlosser, S.W., et al.: The disksim simulation environment version 4.0 reference manual (cmu-pdl-08-101). Parallel Data Laboratory 26 (2008)
10. Yang, P., Jin, P., Yue, L.: Hybrid Storage with Disk Based Write Cache. In: Xu, J., Yu, G., Zhou, S., Unland, R. (eds.) DASFAA Workshops 2011. LNCS, vol. 6637, pp. 264–275. Springer, Heidelberg (2011)
11. Yang, P., Jin, P., Wan, S., Yue, L.: HB-Storage: Optimizing SSDs with a HDD Write Buffer. In: Gao, Y., Shim, K., Ding, Z., Jin, P., Ren, Z., Xiao, Y., Liu, A., Qiao, S. (eds.) WAIM 2013 Workshops 2013. LNCS, vol. 7901, pp. 28–39. Springer, Heidelberg (2013)
12. Jin, P., Ou, Y., Haerder, T., Li, Z.: ADLRU: An Efficient Buffer Replacement Algorithm for Flash-based Databases. Data and Knowledge Engineering (DKE) 72, 83–102 (2012)
13. Li, Z., Jin, P., Su, X., Cui, K., Yue, L.: CCF-LRU: A New Buffer Replacement Algorithm for Flash Memory. IEEE Trans. on Consumer Electronics 55(3), 1351–1359 (2009)