International Journal of Control and Automation
Vol.9, No.1 (2016), pp.445-458
http://dx.doi.org/10.14257/ijca.2016.9.1.38

EfficientMemory Management for NVM-Based Hybrid Memory Systems

Zhangling Wu[1,2], Peiquan Jin[#1,2], Chengcheng Yang[1,2] and Lihua Yue[1,2]

[1]*School of Computer Science and Technology, University of Science and Technology of China, Hefei, 230027, China*
[2]*Key Laboratory of Electromagnetic Space Information, Chinese Academy of Sciences, Hefei 230027, China*
jpq@ustc.edu.cn

Abstract

Non-volatile memory (NVM)such as phase change memory (PCM) has emerged as an attractive type of next-generation memories. NVMcan be incorporated into memory hierarchy to solve the problems of current DRAM instorage density and energy consumption.NVM is byte addressable and its latency is close to main memory, it exhibits higher density and lower idle power consumption than DRAM. However, NVM haslimited write endurance and asymmetriclatency and energy consumption for read and write operations. Therefore, hybrid memory systems involving both NVM and DRAM have been proposed for better utilizing NVM and DRAM. In this paper, we present a novel hybrid memory management schemecalled W-HCLOCK (Write-aware Hybrid CLOCK) that aims to keep a high hit ratio and minimize writes to NVM. Particularly, we introduce the concept of Recent-Rewrite-Distance (RRD), and combine it with recency of write operations to estimate the write-hotness of pages. Then, weplacewrite-intensive pages in DRAM to reduce NVM writes. In addition, we keepadditional information about the writes to evicted pages to avoid unnecessary migrations between DRAM and PCM. We conduct simulation experiments on four synthetic traces and one real OLTP trace. The results suggest the efficiency of our proposal.

***Keywords**: Memory management, NVM, Hybrid memory system*

1. Introduction

Currently, the I/O performance between DRAM and secondary storages has become akey bottleneck ofmassive data storage [1]. Increasing memory capacity may alleviate this problem, but it brings other problems such as scalability [2] and energy consumption [3].Fortunately, A class of new memory technologies called non-volatile memory (NVM), such as phase change memory, Magnetic RAM, Ferroelectric RAM, flash memory, are being developed [4], which introduce opportunities for building efficient hybrid memory systems involving DRAM and NVM.

NVM provides low latency reads and writes on the sameorder of magnitude as DRAM. On the other side, NVMhas largestorage capacity like flash memory. The key properties of NVM are non-volatile, byte-addressability, low idle energy consumption, read/write asymmetry and limited write endurance.With these properties, NVM brings new challenge in the memory hierarchy; one of the attempts is presenting both DRAM and NVM at the same main memory level [5], which takes the advantage of the low latencies of DRAM and the high capacity of NVM.This paper is based on this architecture, in which NVM and DRAM are managed together under a single physical address space.

Traditional main memory management policies are not suitable for new memory architecture due to the long write latency and limited lifespan of NVM. Several researches to handle the read/write asymmetry and endurance problem of NVM have been proposed.

Copyright ⓒ 2016 SERSC

Most of them ameliorate traditional CLOCK or LRU policy, utilizing DRAM to absorb write requests.LRU-based algorithms, such as LRU-WPAM [6], MHR-LRU [7] and APP-LRU [8], try to put read-tendency pages or cold pages to NVM and write-tendency pages or hot pages to DRAM, but LRU-WPAM may evict a page from memory when migrating page between two memory medium, which increases the probability of page fault, and the latter two algorithms cannot migrate write-tendency page from NVM to DRAM in time because migration only performs when page fault occurs. More importantly, relatively complicated LRU-based policies cannot be used in virtual memory management because of frequently list operations and hardware support. Therefore, the CLOCK-based policies, approximate of LRU, become practical choices in virtual memory replacement. CLOCK-DWF [9] and D-CLOCK [10] have been designed to length the lifetime of NVM in hybrid memory system, which denote to make write-hot pages (pages which are frequently written recently) be stored in DRAM and write-cold pages (pages that are rarely written) be stored in NVM.However, both of them design migration strategies at a cost of hit ratio reduction.

In this paper, we present a CLOCK-based hybrid virtual memory management policy called write-aware hybrid CLOCK (W-HCLOCK).W-HCLOCKworks in a similar fashion as CLOCK and tries to put write-hot pages in DRAM and write-cold pages in NVM to reduce the total write count to NVM, the main contributions are as follows:

1) W-HCLOCKremains original data structure and page replacement policy of CLOCK [11], hence can guarantee the same hit ratio as CLOCK.
2) Inspired by the concept of Inter-Reference-Recency (IRR) in the buffer cache replacement algorithm, LIRS [12], W-HCLOCK introduces a new design ofRecent Rewrite Distance (RRD) to decide the write hotness of pages.W-HCLOCK maintains a write clock structure to record the history write information, and we can judge the write hotness (whether a page is write-hot or write-cold) of pages based on the information in the write clock. Meanwhile, when a page is evicted from memory, we do not discard the corresponding write information immediately, but reserve to predict its write hotness (hot or cold) in case of it is accessed in the near future. By doing so, unnecessary page migration can be avoid.W-HCLOCKdoes not count the write count of each page, so the size overhead used to store the write information can be greatly minimized.
3) Page migrations are triggered when writing a page which is not in the right device.W-HCLOCK always migrate write-cold pages to NVM. Particularly, W-HCLOCK givesDRAM pages,which are frequently read but rarely written, much higher priority to be selected as migration victims, this is handled by a swap clock structure.
4) We conduct trace-driven experiments in a simulated NVM-based hybrid memory system. We compare our proposed memory management policy with three alternative methods, i.e., CLOCK [11], CLOCK-DWF [9], and D-CLOCK [10]. Through extensive experiments over several traces, we show that the proposed mechanism can effectively reduce the total write count to NVM without reducing the hit ratio.

Theremainder of this paper is organized as follows. Section 2 briefly describes the problems of page management policy in NVM-based hybrid memory systems. In Section 3, we presents the details of the proposed W-HCLOCK virtual memory management policies, including page partition, replacement and migration policies. Experimental results are shown in Section 4. And finally, we conclude this paper in Section 5.

2. Related Work

Traditional virtual memory management policies, such as CLOCK [11], CLOCK-pro[13]focus on how to improve the hit ratio. While in NVM-based hybrid memory management, reducing the total write count to NVM should also be taken into account. Several memory management policies have been proposed for NVM-based hybrid memory systems.

RaPP [14] uses a modified version of the Multi-Queue [15] algorithm to manage the pages and try to place performance-critical pages and frequently written pages in DRAM, which relies on a sophisticated memory controller monitoring access patterns and ranking pages according to access frequency and write intensity.APG [16] is a page migration policy aiming at increasing energy efficiency and minimizing performance degradation. APG maintains access information in page table entry and moves adaptive number of pages which is decided bythe grouping technique.

LRU-WPAM[6] uses four monitoring queues (DRAM read queue, DRAM write queue, NVM read queue and NVM write queue) to monitor the write and read operations of each pages, so that read-tendency pages are migrated to NVM and write-tendency pages are migrated to DRAM, but the migrations may promote the system to evict a page from the buffer and decrease the hit ratio. To this end, APP-LRU [8] devotes to classifying pages into read-tendency group and write-tendency group without affecting the hit ratio. When a page fault occurs, APP-LRU decides whether the missed page is write-tendency or read-tendency based on a history information table. MHR-LRU [7] records the write operations to DRAM using a DRAM write list, write heat is judged by the location in the write list. If an empty DRAM frame is needed, MHR-LRU migrate a write-cold page from DRAM to NVM. However, both APP-LRU and MHR-LRU migrate pages only when page faults occur, which cannot migrate pages to the right medium immediately when access pattern changes.

Another group of memory management policies are designed based on CLOCK algorithm. Basically, there is no frequently list operation in these algorithms. AIMR [17] identifies write-intensive pages by virtue of two CLOCK lists, which manages the pages with “recency” feature and pages with “frequency” feature, separately.However,since the replacement policy replaces pages which are selected from one of the CLOCK lists, there is a risk that the replaced pages are warmer than some pages in the other CLOCK list. Lee et al. proposed CLOCK-DWF [9], whichimplements an improved CLOCK policy and normal CLOCK policy to manage DRAM pages and NVM pages, respectively. The improved CLOCK policy utilizes the frequency and recency of write operations to classify pages. CLOCK-DWF make sure that all the write operations are performed in DRAM, that means, the pages should be copied to DRAM before being written. To make room for these pages, write-cold DRAM pages are migrated to NVM, this action may beperformed along with an eviction of PCM page. Similarly, D-CLOCK [10] takes the same mechanism to deal with the write operations. Meanwhile, all the DRAM pages and NVM pagesare managed by a global CLOCK list, so that all the pages of similar access pattern have the same chance the be evicted. In addition, in order to limit the writes to PCM incurred by page faults, D-CLOCK may force to evict a DRAM page instead of PCM page. These policies put effort into reducing writes to PCM, but they ignore another important goal of memory management, hit ratio.

3. W-HCLOCK

Higher hit ratio is the most important goal in all of the memory management policies.Besides, when designingthe memory management policies for NVM-based hybrid memory system, reducing the total writes to NVM must also be taken into consideration.

The researches for NVM-based hybrid memory system in the literaturelearned from LRU policy, in which the write hotness of pagesis predicted bythe write recency and the write count. Unfortunately, LRU behaves poorly when handling the access patterns with low locality, such as sequential scan and cyclic pattern that the footprints are slightly larger than the buffer size [12]. To fundamentally solve the problems of LRU, LIRS uses IRR in the replacement decision, where IRR of a page refers to the number of other pages accessed between two consecutive operations to the page.Similarly, the researches for NVM-based hybrid memory system also suffer from memory missing when coping with sequential writes and cyclic write patterns. Inspired by LIRS, W-HCLOCKintroduces a new concept called Recent Rewrite Distance (RRD)to maintain the recorded history write information of each pages. RRD of a page refers to the number of other pages written between two recent write operations to the page, if a page has been written only once or never been written, the RRD of that page is set to infinite.

W-HCLOCKclassifies pages either as write-hot pagesor write-cold pages based on the RRD and recency of pages.Note that, it is meaningless to classify a page with low RRDto write-hot group, which has not been written for a long time. Then we mark the status of pages as write-hot or write-cold. The goal of the proposal is trying to place all write-hot pages in DRAM, and write-cold pages in NVM. Therefore, the maximum number of write-hot pages is set to the number of pages that DRAM can hold.

Table 1 shows an example of how a write-cold page becomesa write-hot one. Initially, the recency of write-hot page D is larger than a write-cold page E, but the RRD of page D is smaller than page E. Then, there is a write operation to page E, and the RRD of page E is changed from 8 to 2, which is smaller than page D. Obviously, the new RRD of page E is smaller than the recency of page D, which means even if page D is updated in the near future, the new RRD of page D is still larger than the one of page E. Therefore, it is believed thatpage E is warmer than page D in writing.W-HCLOCK records the write information of pages, but maintaining the write recency of write-cold pages which is larger than the one of write-hot pages is insignificantbecause these pages have no chance to become write-hot. Therefore, we give the write-cold pages a chance to compete with the write-hot pages.On the other hand, we also grant a write-cold page a test period. If a write-cold page is not accessed during the test period, it is discarded. But if it is accessed in the test period, the write-cold page can turn into write-hot.

Table 1. An Example of Write Hotness Change

Page id	**Before writing page E**			**After writing page E**		
	Write hotness	Recency	RRD	Write hotness	Recency	RRD
A	Write-hot	3	4	Write-hot	4	4
B	Write-hot	2	3	Write-hot	3	3
C	Write-hot	4	2	Write-hot	5	2
D	Write-hot	5	5	Write-cold	6	5
E	Write-cold	1	8	Write-hot	0	2

W-HCLOCK inherits traditional CLOCK replacement policy to replace pages when page fault occurs. The difference is that W-HCLOCK maintains the write information of pages even if they are evicted from the memory. If there is a page fault incurred by a read operation, W-HCLOCK decides where to place the faulted page via the write hotness of that pageshownin the historical write information. If the page is a write-hot page, it is placed in DRAM. Otherwise, the page is placed in where the evicted page is. However, W-HCLOCK places all the faulted pages which are incurred by write operations to DRAM since there are two writes (one for loading to memory and one for write operation) when such page fault occurs. It is also possible that the evicted page releases a NVM frame, but the faulted page requires a DRAM frame. For such case, W-HCLOCK will moveaDRAM write-cold page to NVM. If we store a faulted page which is incurred by a write operation in NVM, there are two writes performed on NVM as mentioned above, but if we store it in DRAM, there is only one write implemented on NVM at worse, which is incurred by a migration from DRAM to NVM.If a write operation hits on NVM and the historical write information of that hit page exists (the write information is in its test period), W-HCLOCK regards it as a write-hot page and migrate it to DRAM.

3.1. Data Structure

W-HCLOCK maintains three clock-aware lists: general clock,writing clock and DRAM swapping clock, as shown in Fig. 1. General clock maintains all the memory pages (NVM and DRAM pages) and performs page replacement like traditional CLOCK policy do (reference bit of a page is set or reset). Thus W-HCLOCK can ensure that there is no hit ratio decrease. The write clock records the most recent writes information, which is used to decide whether a page is write-hot or write-cold. In Fig. 1, shadowed circles represent write-hot pages and white circles represent write-cold pages.In the write clock, some of records belong to thepages stored in NVM or DRAM and the others belong to pages not in the memory, which are marked with "D", "N" or blank.Additionally,W-HCLOCK designs a DRAM swapping clock to manage all the DRAM pages discarded from write clock andthe totally clean DRAM pages. If a page in DRAM swapping clock is hit by a write operation, we move that hit page to writing clock. Note that the DRAM swapping clock may be empty if all the DRAM pages exist in writing clock, which means all the DRAM pages have been accessed recently or all of them are write-hot pages.

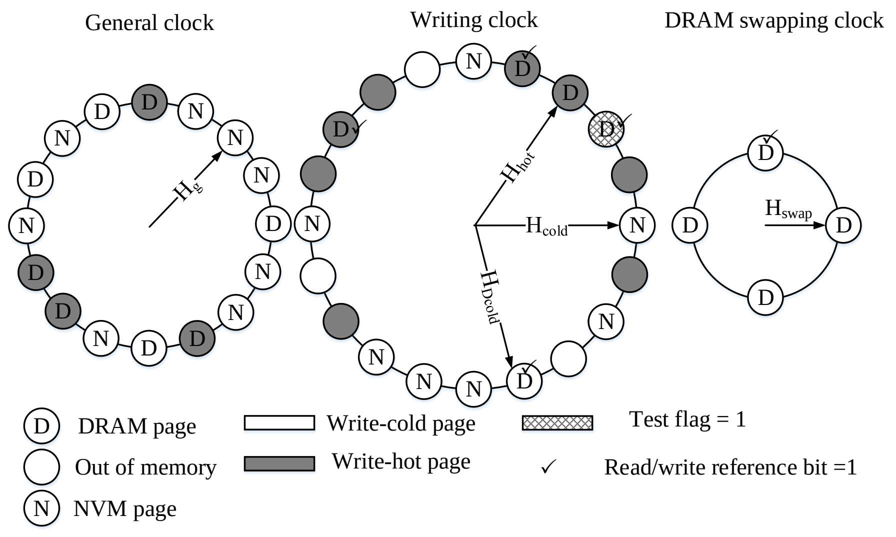

Figure 1. Data structure of W-HCLOCK

In W-HCLOCK, there are several hands that should be introduced. H_g in general clock points to the least recently used pages, which equivalent to the hand in CLOCK. H_{hot} points to the write-hot page with the largest write recency, which serves as a threshold of being a write-hot page. The page pointed by $H_{hot}$is the tail of write clock list. We move H_{hot} to turn a write-hot page into write-cold. H_{cold} points to the write-cold page which is the nearest one toH_{hot} in the clockwise direction.The page pointed by H_{cold} is the least recently written cold page. We limit the number of write information to reduce space cost. When the number of pages in the write clock exceeds a predetermined upper limit, we sweepH_{cold} to find a write-cold victim, and remove the victim from write clock, this also means that we terminate the removed victim's test period.H_{Dcold} points to theDRAM page which is the nearest write-cold page from H_{hot}. This pointer is used to quickly locate a DRAM write-cold page when a migration performs. But H_{Dcold} may be null if there is no DRAM write-cold page.H_{swap}points to the tail of DRAM swap clock, which is the least recently accessed (read) page.

In CLOCK, there are no operations for page hits, only reference bits of the accessed pages are set. Similarly,write clockalso introduces a write reference bit to approximate list operations when write operations hit on it, so the pages whose write reference bit is set are much warmer than the one whose write reference bit is not set, if their write hotness is the same.Once a page in write clock is written, its write reference bit is set (marked by "✓" in Fig. 1), and page migration may happen if it is a NVM pages (this page is regarded as a write-hot page, although we do not change its write hotness immediately). As mentioned before, if a write-cold page is not accessed during the test period, it will be discarded from write clock (i.e., it is out of test period); but we give the write-cold page whose write reference bit is set a second chance to be discarded by using a test flag (set to "1", the page is filled with lattice in Fig. 1) since it has been accessed before.Note that, if a page is not in the write clock or its test flag is set, that means it is out of its test period, and cannot be turned into write-hot page. DRAM swap clock also uses a read reference bit to distinguish hot pages from cold ones;if a page in DRAM swapping clock is read, its read reference bit is set (marked by "✓" in Fig. 1).

After inducing the test flag, we redefine H_{cold} as the nearest write-cold page to the list tail, whose test flag is unset, and H_{Dcold} as the nearest DRAM write-cold page to the list tail, whose write reference bit is unset or both write reference bit and test flag are set. When H_{cold} is triggered, we keep moving it until it finds a write-cold page eligible for discarding, and stop at the next write-cold page whose test flag is reset. During moving H_{cold}, if the write reference bit of the write-cold page pointed by H_{cold} is set, we set its test flag to mark this page out of test period. Otherwise, we found a write-cold page to discard.

3.2. Page Migration

Next we will describe how these hands are coordinated with each other to search a write-cold page for migration.

If W-HCLOCK decides to put a page in DRAM, but there is no empty DRAM frame for it, a migration from DRAM to NVM should be performed. First, we examine whether the DRAM swap clock is empty. If it is not empty, the migrated victim is selected from DRAM swap clock. We start from the counterclockwisedirection of $H_{swap}$to look for a page whose read reference bit is set, since such page is more read intensive than others (all of them are write-cold) and such migration benefits more (i.e., the probability of migratingthe page back to DRAM is lower). If all the read reference bits in DRAM swap clock are unset, the page before H_{swap} in the clockwise direction is selected as a victim. Otherwise, if there is no page in DRAM swap clock, the migrated page is searched from write clock. The purpose is to find a DRAM write-cold page, so we first check whether $H_{Dcold}$is null. If it is not null, the migrated victim is found and H_{Dcold} is moved to next DRAM page which meets the requirement. Note that, if the write reference bit and test flag of the victim are set, we reset them and move the victim to the head of the write clock (immediately beforeH_{hot} in the clockwise direction). If $H_{Dcold}$is null, which means all the pages in DRAM are write-hot or will soon be write-hot. At this moment, we start the cold-to-hot procedure thatis used to find a write-cold page to turn into write-hot.

Cold-to-hot procedure enables a temporary hand, which starts from the position of $H_{cold}$and moves in the clockwise direction,to find the target page. The temporary hand skips all the write-hot pages and the write-cold pages whose write reference bit is unset. If the write reference bit of the page pointed by the temporary hand is set and the page is in its test period (test flag is unset), we turn the page into write-hot page, reset its write reference bit and move it to the head of the write clock. Meanwhile, if the number of write-hot pages exceeds the size of DRAM memory, we ask H_{hot} for its actions and the procedure ends. However, if both of the write reference bit and test flag are set, there is no status change as well as H_{hot} actions, but the two bit are reset and we move the page to the head of the write clock. The temporary hand keeps moving until it turns a write-cold page into write-hot or meets with H_{cold}again. The situation that the temporary hand meets with

H_{cold}means that no write-cold pages need to turn into write-hot (i.e., all the write-hot pages are DRAM pages). When the temporary hand meets H_{cold}again, we ask H_{hot} for its actions too.

We trigger the movement of H_{hot} when a write-cold page turns into write-hot and the number of write-hot pages exceeds the determined upper limit. The duty of moving H_{hot} is to turns a write-hot page with the largest write recency into a write-cold one. If the write reference bit of the hot page swept by H_{hot} is unset, we simply change its write hotness to write-cold and stop H_{hot} at the next write-hot page. Otherwise, if its bit has been set before, we keep it as write-hot page, but reset the write reference bit. However, the test period of any write-cold pages swept by H_{hot} will be terminated, which means H_{hot}removes the write-cold pages from write clock whose write reference bits are unset, or sets the test flags if their write reference bits are set, just like H_{cold} does.

All of the actions described before are to find a DRAM write-cold page for migration. We trigger the cold-to-hot procedure and the movement of H_{hot} when all the pages in DRAM are write-hot or will soon be write-hot. Figure 2 shows an example of status change and page migration. Initially, the DRAM memory size and the maximize number of write-hot pages is 6;H_{Dcold} is null sinceall the pages in DRAM are write-hot or will soon be write-hot (shown in Fig. 2(a)); H_{hot} points to page B. Then a write request on page A is implemented (its write reference bit is set as shown in Fig. 2(b)). Since page A is accessed during its test period, we think it is a write-hot page and should be put into DRAM. However, there is no empty DRAM frame for it, so a DRAM write-cold page should be migrated to NVM. The cold-to-hot procedure changes the write-cold page C to write-hot, and put it after page B in the clockwise direction which is pointed by H_{hot}at that moment. But the number of write-hot pages exceeds 6, the movements of H_{hot} are triggered. As a result, page B turns to write-cold page and is pointed by H_{Dcold}, H_{hot} and H_{cold} stop at write-hot page F and write-cold page A, respectively, as shown in Fig. 2(b). Note that write-cold page E is discarded by H_{hot} because it is out of test period. Finally, we have found a DRAM write-cold page B(pointed by H_{Dcold}), and swap it with the NVM page A (shown in Fig. 2(c)).

Since the page replacement of memory is controlled by the general clock, some of the write-hot pages may be evicted from the memory, thus the situation may takes place that H_{hot} turns a non-resident write-hot page into write-cold one. As shown in Fig. 2(d), the writing operation on page *G* triggers the cold-to-hot procedure and the movements of H_{hot}. So page A turns into write-hot page and is moved to the head of write clock (after page F in the clockwise direction),the non-resident page F pointed by H_{hot} before turns to write-cold page. Unfortunately, we fail to find a DRAM write-cold page to swap with page G. The migration of page *G* is delayed until it is written again during its test period in the future. Figure 2(d) also shows that we set the test flag of page *H*, rather than discard this page like we did in Fig. 2(b), this is because page *H* has been written before during its test period, but have not had time to turns to write-hot.

3.3. Page Management

This section we will introduce the page management policy in details, including when to migrate pages and how to change status bits.

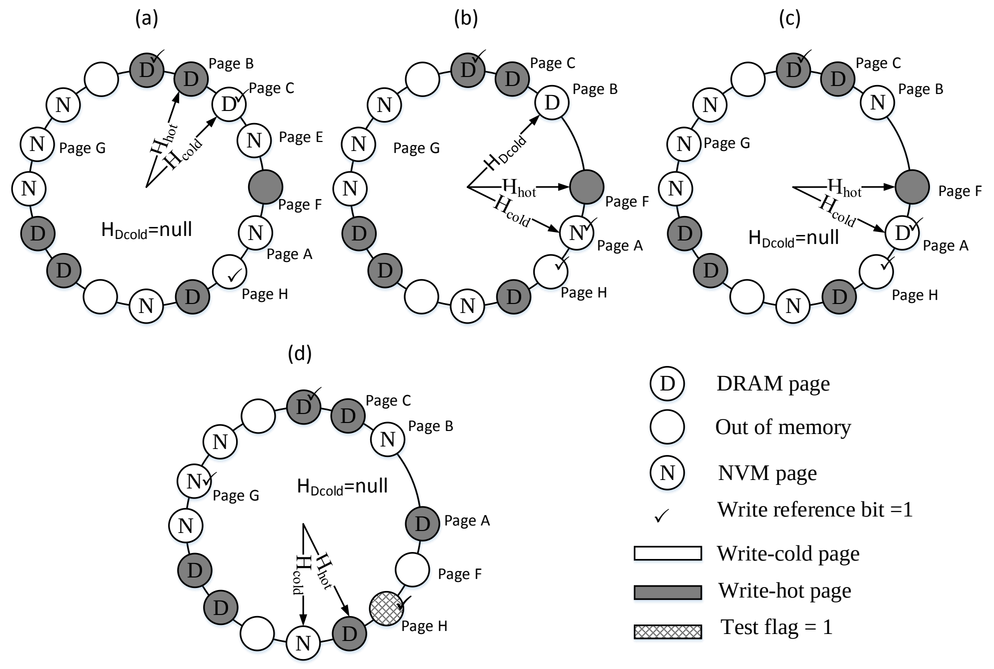

Figure 2. An Example of Page Migration

```
Algorithm 1: page hit policy
Input: requested page p, operation type op
1   p.reference_bit=1;
2   if(op is read and p exists in DRAM swap clock)then
3   p.read_reference_bit=1;
4   else if (op is write)then
5   if (p exists in write clock)then
6   p.write_reference_bit =1;
7   if(p is stored in NVM)then
8   call the page migration procedure to find a DRAM write-cold page q;
9   if ($q \neq$null)then swap q and p;
10  else/*p does not exist in write clock*/
11  if (p exists in DRAM swap clock) then remove p from DRAM swap clock;
12  add p to the head of the write clock ;
            /*immediately before $H_{hot}$ in the clockwise direction)*/
13  set p to write-cold; p.write_reference_bit=0; p.test_flag=0;$N_{write\_clock}$++;
14  if ($N_{write\_clock}$>upper limit) then trigger the movements of $H_{cold}$;
```

Algorithm 1 shows the processing procedure when a request hits on memory. When a page is accessed, W-HCLOCK first set its reference bit (*line 1*). If the operation type is read and the requested page exists in DRAM swap clock (it is a write-cold page), the read reference bit of that page is set (*line 2-3*), which indicates that the page is more read intensive than other write-cold pages. If a write operation hits on memory, W-HCLOCK first check whether the requested page *p* exists in write clock. If not, we add it to the head of write clock and initial its status bits (*line 12-13*). Before that, page *p* is removed from DRAM swap clock if it exists in DRAM swap clock (*line 11*). We keep track of the total number of pages in write clock (N_{write_clock}). Once it exceeds 2**m*, where *m* is the memory size in the number of pages, the movements of H_{cold} are triggered and a page will be discarded from the write clock (*line 14*). If the discarded page is a DRAM page, we added it to the head of DRAM swap clock and its read reference bit is reset. If page *p* already exists in write clock, its write reference bit is set. W-HCLOCK regards such page as

Copyright ⓒ 2016 SERSC

write-hot page (although it may have not been changed to write-hot). If it is an NVM page, W-HCLOCK tries to find a DRAM write-cold page*q*by activating the page migration procedure, and swap *q* with *p* if found (*line 8-9*). Since we may fail to find the DRAM write-cold page, the swapping may not be performed.

Algorithm 2 describes how to manage pages when a page fault occurs. When a page fault occurs,first, W-HCLOCK selects a victim *q*from the general clock to replace based on traditional CLOCK algorithm (*line 1*). Second, W-HCLOCK decides where to place the faulted based on the write information in write clock.

If it is a read operation and the requested page *p* is a write-hot page, we place it to DRAM, which can avoid unnecessary migration since there may be many write operations to page *p* in the future. However, if the evicted page *q* is a NVM page, W-HCLOCK tries to call the page migration procedure to find a DRAM write-cold page *r*, and migrates*r* to *q* if found (*line 4-6*). Then we load the requested page *p* from disk to *r*if the migration from DRAM to *q* is performed successfully or to *q* if the migration fails (*line 7*). If the requested page is a write-cold page, we simply store it in where the evicted page is (*line 9*).

Algorithm 2: *page fault policy*

Input: requested page *p*, operation type *op*

```
 1  Replace a page q from general clock using CLOCK policy;
 2  if(op is read) then
 3     if (p exists in write clock and p is a write-hot page) then
 4     if (qbelongs to NVM) then
 5     call the page migration procedure to find a DRAM write-cold page r;
 6     if (r≠null) then migrate r from DRAM to q;
 7     load p to r or q; /*load to the frame where r or q is stored before*/
 8     else
 9     load p to q;
10  else /*write operation*/
11  if (p exists in write clock) then
12  p.write_reference_bit =1;
13  else
14  add p to the head of the write clock;
15  set p to write-cold; p.write_reference_bit=0; p.test_flag=0; $N_{write\_clock}$++;
16  if ($N_{write\_clock}$>upper limit) then trigger the movements of $H_{cold}$;
17  if (q belongs to NVM)then
18  call the page migration procedure until W-HCLOCK finds a write-cold page r;
19  migrate r from DRAM to q;
20    load p to DRAM;
21  add p to the head of general clock; p.reference_bit=1;
```

If the operation is a write request, we need to store it in DRAM since there are two operations as mentioned before. If the requested page *q* exists in the write clock, its write reference bit is set (*line 11-12*). Otherwise, W-HCLOCK adds it to the head of the write clock, reset the status flags and activated the movements of H_{cold} to discard a write-cold page from the write clock if needed (*line 13-16*). Here, we force to store page *p* in DRAM, which make us call the page migration procedure repeatedly until a DRAM write-cold page *r* is found, if the evicted page *q* belongs to NVM (*line 18*). After finding page *r*, we move it from DRAM to the NVM frame where page *q* was stored before (*line* 19). The requested page *p*is loaded to DRAM. Finally, we add page *p* to the head of general clock (immediately before H_g in the clockwise direction) and set its reference bit (*line 21*).

4. Experiments

In this section, we present the evaluation results of our proposal described in Section 3. We first introduce the experimental setting on the workloads and competitive algorithms. Then we present the comparative results with these competitive algorithms.

4.1. Experimental Setup

We have implemented a simulated NVM based hybrid memory system, which adopts unified addressing mode, DRAM takesthe low-end addresses and PCM takes the high-end addresses. The page size is set to 4KB. We compare our proposal with four alternatives: CLOCK [11], CLOCK-DWF [9], and D-CLOCK [10]. Both CLOCK-DWF and D-CLOCK are designed based on CLOCK

Both synthetic and realistic traces in the following experiments are used, which have been shown in table 2. We use DiskSim [18] to generate two groups of traces, i.e., T5582 and T5555, by setting different read/write ratio and locality. For the four ZIPF traces [19], the probability of accessing the i^{th} page among a totality of N pages, $p_i$is calculated based on the following formula:

$$p_i = \frac{1}{H_N^{1-\theta} \times i^{1-\theta}} (1)$$

$\theta = \log a \,/\log b$ and $H_N^{1-\theta}$ is the Nth harmonic number of order $(1-\theta)$. When$a$ is 0.6 and b is 0.4, the distribution means that 60% of the references are located on the most active 40% of the pages. There are total 10000 different pages for each synthetic trace. The locality of 80%/20% means that 20% of pages absorb 80% of requests. The other group of trace, OLTP, is collected from a real bank database system, which has also been used in APP-LRU [8] and ADLRU [20]. This trace contains 607390 references to a CODASYL database with a total size of 20 Gigabytes.

Table 1. Parameters of the Zipf and OLTP Traces

Traces	Footprints	Read/write ratio	Locality	Total Requests
Zipf1982	10000	10%/90%	80%/20%	400000
Zipf1955	10000	10%/90%	50%/50%	400000
Zipf2873	10000	20%/80%	70%/30%	400000
Zipf4682	10000	40%/60%	80%/20%	400000
OLTP	10000	53%/47%	~	607390

4.2. NVM Write Count with Zipf Traces

Figure 3 shows the effect of different NVM size ratio with a fixed memory size, to investigate the feasibility of different memory management policies, since NVM includes several new memory devices, which have different scalability. The y-axis denotes to the total write count on NVM and the x-axis shows the different proportion of DRAM and NVM. The total size of memory is set to 2000 pages, and the write count is induced by page faults, write operations of traces and migrations from DRAM to NVM. When increasing the size of DRAM, the write count on NVM increases for different traces and different policies, since NVM becomes larger than before. Figure 3 also shows that the write count on NVM is less when implementing the proposed W-HCLOCK in most cases, which means our proposal can adapt to a variety of hybrid memory configuration and perform well in reducing the NVM write count. However, when the ratio of DRAM becomes smaller (such as 1:5 and 1:6), the NVM write count of W-HCLOCK is larger than D-CLOCK for Zipf1982 and Zipf4682. Since W-HCLOCK sets the maximum number of write-hot pages as the size of DRAM memory, which is smaller than the actual number of write-hot pages, thus some of the write-hot pages are migrated between NVM and DRAM, frequently.

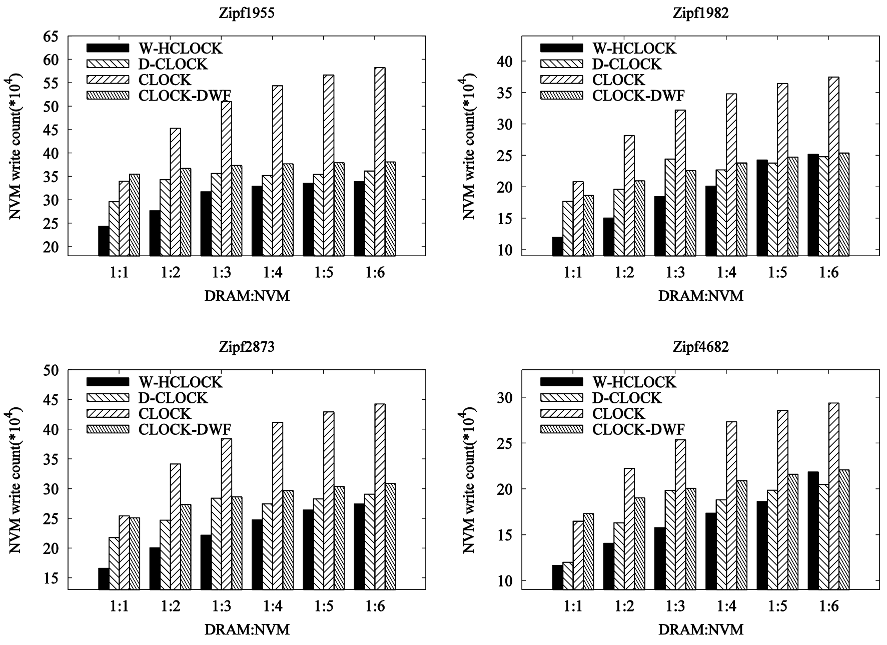

Figure 3. NVM write Count for Different DRAM/NVM Size Ratio

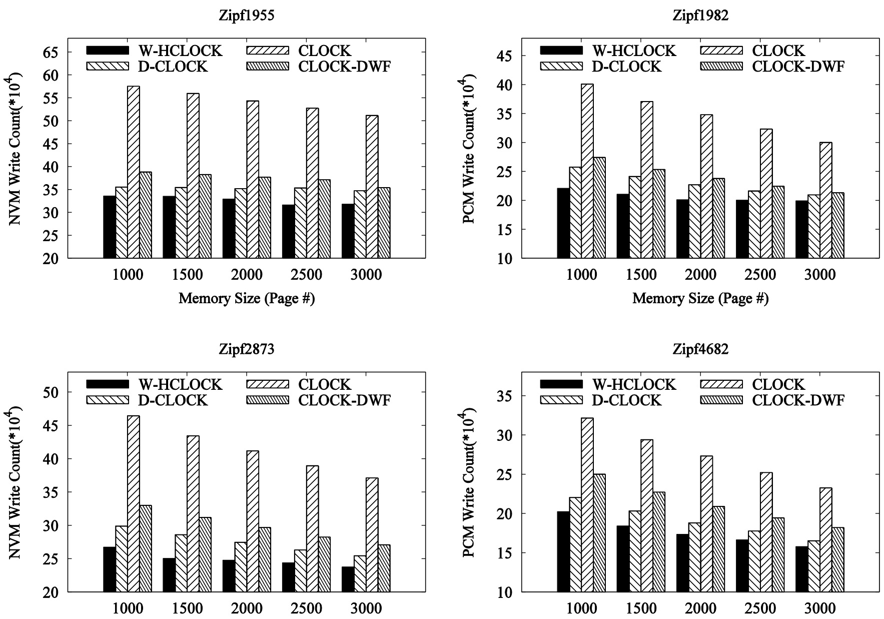

Figure 4. NVM Write Count for Different Memory Size

Reducing the total write count on NVM is one of the most important goal in NVM-based hybrid memory management design. Figure 4 shows the total write count on NVM induced by page faults, write operations of traces and migrations from DRAM to NVM. In the experiment, the ratio of DRAM and NVM is kept to 1:4. We make two observations. First, a large memory size yields less NVM write count for the four memory management policies. This is because as the memory size increases, the DRAM size increases too, which serves more write operations than before. Second, the write count of our proposed W-HCLOCK is consistently smaller than the other three competitors over all traces and with different buffer sizes. Actually, the NVM write count of W-HCLOCK is only about 55%-66% of the one of CLOCK, while both D-CLOCK and CLOCK-DWF incur more NVM write count than the proposal. This is because,in D-CLOCK and CLOCK-DWF, each time the write operation hits on NVM, page migration between NVM and DRAM will be performed, which can incur a large number of page migrations. However, W-HCLOCK decides whether to move pages from NVM to DRAM based on the write hotness (write-hot or write-cold) of the pages, so that unnecessary page migrations can be avoided.

4.3. Page Faults with Zipf Traces

Another goal of NVM-base hybrid memory management design is keeping or improving the hit ratio of memory. Figure 5 shows the number of page fault for the four memory management policies when varying the memory size. We make four observations. First, with a larger memory size, the page fault number decreases, as expected. Second, the page faults of W-HCLOCK keep the same with CLOCK. Third, CLOCK-DWF incurs more page faults when the references concentrate on a small number of pages (such as Zipf1982, Zipf4682 and Zipf2873). The reason is that, when CLOCK-DWF moves pages from DRAM to NVM and there is no empty frame for the migrated pages, CLOCK-DWF will call CLOCK policy to evited a NVM pages out of memory, which increase the page faults. Fourth,actually the experimental data also shows that, the page faults of D-CLOCK is nearly the same with other three policies in most cases, but D-CLOCK has a little more page faults than CLOCK in some cases. Since D-CLOCK will replace DRAM pages forcibly to store the pages loaded from disk and prevent these pages from writing NVM frequently, which may evict a warmer page from memory.

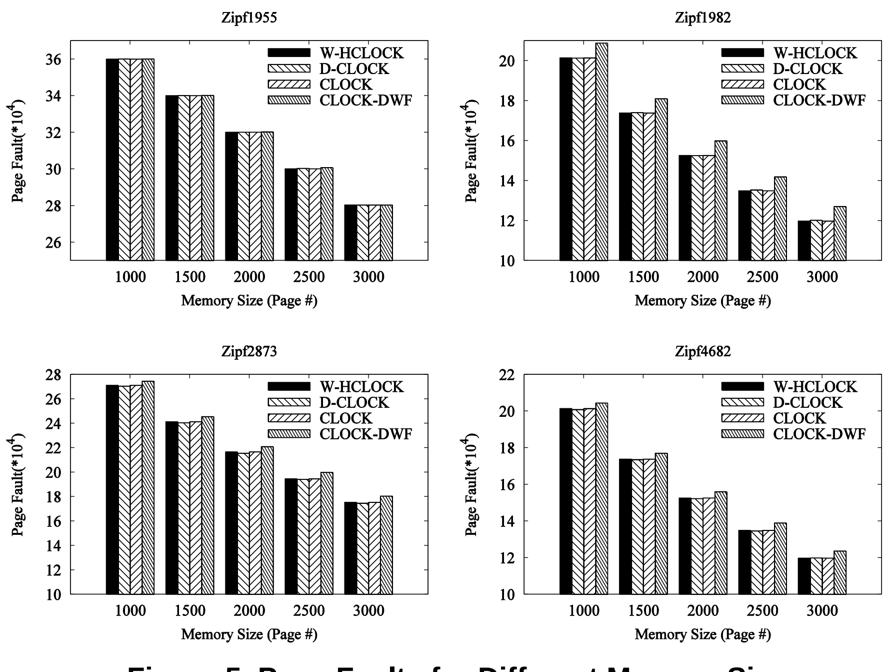

Figure 5. Page Faults for Different Memory Size

4.4. Experimental Result withthe OLTP trace

This section describes the experimental comparative results with the OLTP trace. Figure 6(a) and Fig. 6(b) describe comparative results of NVM write count and page fault, separately. We make several observations. First, from Fig. 5(a), we can see that NVM write count of D-CLOCK is smaller than other three policies. This is because the memory size is not large enough to accommodate the total memory footprint of OLTP, thus there are a large number of page faults. Meanwhile, there are a lot of read requests, W-HCLOCK will not place write-cold pages to DRAM forcibly, but in order to prevent new pages loaded from disk being written to NVM too often, D-CLOCK will place them in DRAM by evicting DRAM pages forcibly, which will lower the hit ratio of memory. Therefore, the second observation is both D-CLOCK and CLOCK-DWF have higher page faults than other two policies, as shown in Fig. 6(b). When there is no empty frame in target memory medium, D-CLOCK and CLOCK-DWF will replace pages by forcibly. Since there are several order of magnitudes between memory and secondary storages, a larger number page faults will seriously hinder the system performance.

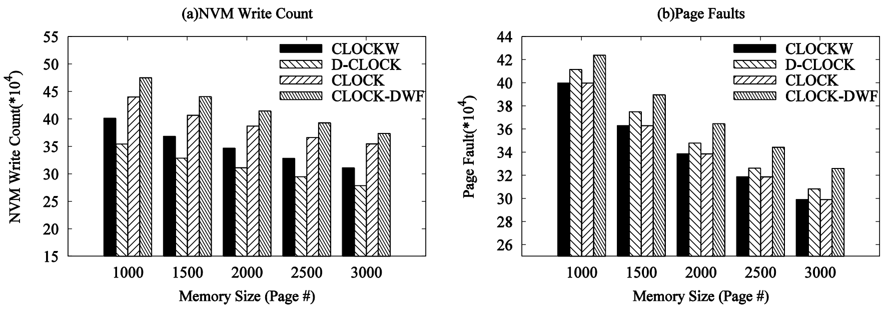

Figure 6. Page Fault Count and NVM Write Count for the OLTP Trace

5. Conclusions

In this paper, we propose a CLOCK-based virtual memory management policy called W-HCLOCK for NVM-based hybrid memory system. W-HCLOCK introduces RRD to decide the write hotness(write-hot or write-cold) of pages, so that write-hot pages can be placed to DRAM in time and the writes on NVM can be reduced substantially. Meanwhile, when a page is evicted from memory, W-HCLOCK maintainsthe corresponding write information of that page for a while, which can be used to predict the page’s write hotness in case of it is accessed in the near future. Therefore, based on the predicted write hotness, W-HCLOCK determines to put pages in NVM or DRAM, and unnecessary page migration can be avoided. W-HCLOCK uses CLOCK policy to perform page replacement, thus NVM writes can be reduced without lowering the hit ratio of memory. Through comprehensive experiments on several traces, we demonstrate that W-HCLOCK can reduce NVM write count without hit ratio degradation, by accurately predicting pages’ write hotness and moving them between NVM and DRAM.

Acknowledgments

This paper is partially supported by the National Science Foundation of China under grant (61379037 and 61472376) and the Fundamental Research Funds for the Central Universities.

References

[1] R. Fang, H. Hsiao, B. He, C. Mohan and Y. Wang, “High performance database logging using storage class memory”, Proceeding of ICDE, **(2011)**; Hannover.
[2] J. A. Mandelman, R. H. Dennard, G. B. Bronner, J. K. DeBrosse, R. Divakaruni, Y. Li and C. J. Radens, “Challenges and future directions for the scaling of dynamic random-access memory (DRAM)”, IBM Journal of Research and Development, vol. 46, no. 2, **(2002)**, pp. 187-212.
[3] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler and T. W. Keller, “Energy management for commercial servers”, Computer, vol. 36, no. 12, **(2003)**, pp. 39-48
[4] S. Viglas, “Data Management in Non-Volatile Memory”, Proceedings of SIGMOD, **(2015)**.
[5] S. Chen, P. Gibbons, and S. Nath, “Rethinking database algorithms for phase change memory”, Proceedings of CIDR, **(2011)**.

[6] H. Seok, Y. Park, K. W. Park and K. H. Park, "Efficient page caching algorithm with prediction and migration for a hybrid main memory", ACM SIGAPP Applied Computing Review, vol. 11, no. 4, **(2011)**, pp. 38-48.
[7] K. Chen, P. Jin and L. Yue, "A novel page replacement algorithm for the hybrid memory architecture involving PCM and DRAM", Proceedings of NPC, LNCS 8707, **(2014)**.
[8] Z. Wu, P. Jin P, C. Yang and L. Yue, "APP-LRU: A New Page Re-placement Method for PCM/DRAM-Based Hybrid Memory Systems", Proceedings of NPC, LNCS 8707, **(2014)**.
[9] S. Lee, H. Bahn and S. H. Noh, "CLOCK-DWF: A write history-aware page replacement algorithm for hybrid PCM and DRAM memory architectures", IEEE Transactions on Computers, vol. 63, no. 9, **(2014)**, pp. 2187-2200.
[10] K. Chen, P. Jin and L. Yue, "Efficient Buffer Management for PCM-Enhanced Hybrid Memory Architecture", Proceedings of APWeb, LNCS 9313, **(2015)**.
[11] F. J. Corbato, "A Paging Experiment with the Multics System", MIT Project MAC Report MAC-M-384, **(1968)**.
[12] S. Jiang and X. Zhang, "LIRS: an efficient low inter-reference recency set replacement policy to improve buffer cache performance", ACM SIGMETRICS Performance Evaluation Review, vol. 30, no. 1, **(2002)**, pp. 31-42.
[13] S. Jiang, F. Chen and X. Zhang, "CLOCK-pro: An effective improvement of the CLOCK replacement", Proceedings of USENIX ATC, **(2005)**.
[14] L. E. Ramos, E. Gorbatov and R. Bianchini, "Page placement in hybrid memory systems", Proceedings of SC, **(2011)**.
[15] Y. Zhou, P. Chen and K. Li, "The Multi-Queue Replacement Algorithm for Second-Level Buffer Caches", Proceedings of USENIX ATC, **(2001)**.
[16] D. Shin, S. K. Park, S. Kim and K. H. Park, "Adaptive page grouping for energy efficiency in hybrid PRAM-DRAM main memory", Proceedings of SAC, **(2012)**.
[17] Z. Sun, Z. Jia, X. Cai, Z. Zhang and L. Ju, "AIMR: an adaptive page management policy for hybrid memory architecture with NVM and DRAM", Proceedings of HPCC/CSS/ICESS, **(2015)**.
[18] DiskSim, Available at Hyperlink, http://www.pdl.cmu.edu/DiskSim.
[19] D. Knuth, "The Art of Computer Programming", Volume 3: Sorting and Searching, Addison-Wesley, **(1973)**.