IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 8, 2019, accepted March 1, 2019, date of publication March 13, 2019, date of current version April 2, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2904803

WIRD: An Efficiency Migration Scheme in Hybrid DRAM and PCM Main Memory for Image Processing Applications

NA NIU[1], FANGFA FU[1], BING YANG[2], JIACAI YUAN[1], FENGCHANG LAI[1], AND JINXIANG WANG[1]

[1]Microelectronics Center, Harbin Institute of Technology, Harbin 150000, China
[2]Harbin University of Science and Technology, Harbin 150000, China

Corresponding author: Jinxiang Wang (jxwang@hit.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61504032.

ABSTRACT Using a hybrid main memory in embedded systems to process image processing applications has become an irresistible trend. However, the performance deficiencies (less write endurance and relative longer write latency) in phase change memory (PCM) have become the bottleneck for performance improvement in the whole system. To further improve the performance in the hybrid memory system, data migration schemes have been put forward to migrate the sequential data between DRAM and PCM. Nevertheless, the existing schemes are always universal type page migration schemes, which cannot be aware of the write hot pages in image processing applications accurately. In addition, a large number of unnecessary page migrations occurring has the potential to increase the latency overhead of page management and destroy the locality of the applications leading to performance degradation. In this paper, focusing on the image processing applications, we present a better estimator in predicting the future memory writes: inter-reference distance with history writes frequency. Based on this observation, we present a novel page migration scheme for hybrid DRAM and PCM memory architecture called write frequency with inter-reference distance (WIRD). The scheme monitors access patterns, migrates pages between DRAM and PCM relying on the memory controller. The experimental results show that our scheme minimizes the unnecessary page migrations which make most of the write hot pages absorbed by DRAM efficiently. Meanwhile, the high-efficiency character of WIRD has decreased the write to read switching ratio in the system which makes the PCM effective page write access time reduced to a large extent.

INDEX TERMS Hybrid memory, PCM, inter-reference distance, page migration, write frequency.

I. INTRODUCTION

Applying contemporary embedded systems (such as mobile phones, video players) to execute images processing applications is becoming one of the new approaches in image processing domain, which can significantly improve the capabilities of the data processing and communication in image processing [1]–[3]. Majority of the image processing operations are neighborhood oriented. In the execution of image processing, these operations are performed on stream of image data in sequence or on a group of images. These operations make image processing applications to be data dominated. The consequence is that the large storage space requirement for the pixel data and high power consumption during data storage and transfer in embedded systems [2]. Larger memory capacity consequently tends to incur larger access latency and needs more chip space. All of these factors degrade the performance of the embedded systems.

To satisfy the requirement capacity for the image processing applications in embedded systems without performance degradation, memory architecture researchers have attempted to use Phase Change Memory (PCM) as a potential replacement for DRAM, which can bridge the performance and capacity gaps. PCM, an emerging non-volatile memory technology, consumes little idle power, does not require refresh energy, and exhibits access latency in the nanosecond range.

The associate editor coordinating the review of this manuscript and approving it for publication was Zeev Zalevsky.

TABLE 1. The characteristics of DRAM and PCM.

attributes	DRAM	PCM
Read/write latency	50 / 50 (ns)	60 / 170 (ns)
Read/write energy	0.1 / 0.1 (nJ/bit)	0.2 / 1.0 (nJ/bit)
Static power	1 (W/GB)	0.1 (W/GB)
Retention	Volatile(64ms)	Non-volatile
Erase required	bit	bit
Endurance	∞	$10^6 \sim 10^8$

PCM is byte-addressable and high-density which can store more bits in the same physical area compared to DRAM cells. The characteristics comparison of DRAM and PCM is shown in Table 1. Recently, to exploit the advantages of the high capacity of PCM and the low latencies of DRAM, researchers have designed hybrid memory systems with a small amount of DRAM and a large amount of PCM combination. Recent researches have adequately illustrated hybrid memory system will be able to meet the current capacity requirements for the main memory system without performance degradation [3]–[5].

Unfortunately, PCM has performance deficiencies (less write endurance, relative longer write latency) to overcome before PCM becomes a part of the main memory system. Less endurance means the allowed number of write operations for each PCM cell is limited. Current write endurance times of a PCM cell is known to be about $10^6 \sim 10^8$ [4].

If the number of write operations performed on a PCM cell exceeds this limit lifetime threshold, it will end working [6]. More seriously, the other PCM memory cells could not work to their endurance lifetime threshold as well and cannot be used any longer [7]. Meanwhile, the write access time of PCM is expected to be about 6–10 times slower than that of DRAM and the write energy is 8-10 times higher than that of DRAM [4]. These limitations about PCM writes may significantly short the lifetime of memory system, restrict the usefulness of PCM in commercial system and impact system performance adversely. Therefore, the write traffic to these PCM devices must be reduced. Many researchers have tried to reduce the number of write operations in PCM using the page replacement policy based on the hybrid memory organization [8]–[12].

Focusing on the image processing applications, this paper explained two contributions. The first contribution of this paper is the conclusions that inter-reference distance with history write frequency may be a better estimator in predicting the future memory writes for image processing applications through analyzing these applications. Based on this observation, we present a new memory page replace management policy called WIRD (**W**rite Frequency with **I**nter-**R**eference **D**istance) implemented by the memory controller of hybrid system which is particular for image processing applications. This mechanism could hide the higher latency of PCM efficiently by migrating future write hot page to DRAM with limited migration times. Our policy minimizes the unnecessary page migrations and makes frequent memory writes absorbed by DRAM. This is our second contribution.

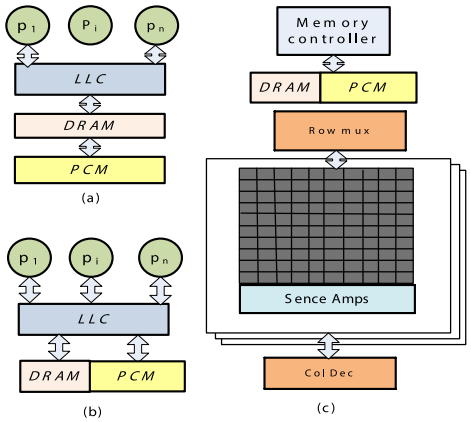


FIGURE 1. Hybrid memory architectures (a) DRAM cache architecture. (b) Parallel hybrid architecture. (c) Memory details.

The remainder of this paper is organized as follows. Section II introduces the recent hybrid memory architectures and related works about page migration policy briefly. In Section III, we capture the reference characteristics in image processing workloads. Then, we present the WIRD page migration policy for the hybrid memory architecture in section IV. Section V presents the evaluation of our experimental results. Finally, we conclude this paper in Section VI.

II. BACKGROUND AND RELATED WORK

A. HYBRID MEMORY SYSTEM ARCHITECTURE

Discussions regarding on exploiting DRAM and PCM combination architectures have gained much attention in recent years. Hybrid main memories architectures are classified in two organizational types: one is to use a small account of DRAM in front of PCM to hide the endurance and write latency problems of PCM, we call it DRAM cache architecture as depicted in Fig. 1(a). While accessing a page, the DRAM part is accessed as an inclusive hardware cache preferentially. The PCM part will be accessed only when the DRAM cache miss appears. However, for workloads with poor locality, the competition for DRAM cache resource may increase energy consumption and degrade the efficiency of the DRAM cache. Fig. 1(b) depicted another hybrid memory architecture. In this architecture, DRAM and PCM are at the same memory level and they share the physical memory address space. We call it parallel hybrid architecture [11], [15]. It makes both PCM and DRAM could be accessed by CPU directly and convenient to be managed for OS (operating system). In this organization, DRAM space can be used more efficiently which can meet the capacity requirement for image processing applications [4]. In this paper, the investigation is focusing on the parallel hybrid architecture. Fig. 1(c) displays the relations between memory controller and chips. Memory controller connects the last level cache and memory chips to ensure the communications between them.

B. RELATED WORK

Recently, prior researches generally confirm that page replacement policies can efficiently manage the hybrid main memory to enhance the write performance and endurance ability of PCM in hybrid memory system [4], [13]–[21]. Most policies are based on the CLOCK algorithm and LRU algorithm.

The most widely referred algorithms are elaborated below:

RaPP [4] is the most remarkable algorithm for memory controller hardware improvements in hybrid memory system. RaPP efficiently ranks pages according to popularity (access frequency) and migrates the top ranked pages to be absorbed by DRAM. But using the read and write frequency to predict the write frequency of PCM may not be quite accurately. If the access frequency of a PCM page has reached the migration threshold which is the contribution of the read operations, such migration may increase the power consumption and other overheads. In particular, the overhead to maintain multiqueues is relatively large.

To maintain the high hit ratio and stable time performance of the system, MHR-LRU [13] and APP-LRU [14] are proposed respectively. They are all LRU-based algorithms for hybrid memory. For maintaining a high hit ratio, MHR-LRU takes both page access patterns and the influences of the page faults into account to update pages in DRAM and performs page migration between DRAM and PCM. APP-LRU frees the page in the LRU position. It identifies the requested page as a DRAM page or a PCM page according to its historical access pattern and request type to reduce the write accesses in PCM and maintain stable time performance.

Lee *et al.* [15] proposed a page replacement policy called CLOCK-DWF algorithm, when a write operation occurs in an allocated page of PCM, the page will be immediately migrated to DRAM. The mechanism of CLOCK-DWF results in frequency migrations, which lead to a higher overhead.

M-CLOCK(migration optimized CLOCK) was proposed in [16].It focus on reducing the write operations in PCM by migration read intensive pages to PCM or evicting the unlikely referenced page in storage. However, M-CLOCK does not take the dynamic access patterns of the workloads into account which may lead to reallocate the evicted pages to DRAM.

However, MHR-LRU, APP-LRU, CLOCK-DWF and M-CLOCK are all designed to keep write-intensive pages in DRAM for absorbing future write operations, but they do not consider the influence of frequent page swap operations in the whole system.

Considering the migration efficiency in the system, Salkhordeh *et al.* [17] presented an efficient data migration scheme in OS level. In this policy, two LRU queues are used for the sake of data migration. In [18], refinery swap was proposed to solve the overhead of swap operations by entirely avoiding direct writes in PCM. It provides a multilevel priority scheme in choosing DRAM victim when swap-out operations occur. Kim *et al.* [20] presented Adaptive-Classification CLOCK (AC-CLOCK) page replacement policy for the hybrid main memory of embedded systems. It ameliorates request migration flaw of CLOCK-DWF. When PCM writes happens, instead of blindly starting the migration, it needs to judge whether the current PCM page is worth migrating. The algorithm uses a saturation counter to calculate the write frequency in the DRAM (cleared one turn), but this could not distinguish the write frequency level clearly. For write-intensive applications, it is possible to allocate the write hot pages into PCM.

For hybrid memory, three issues of the previous page replacement schemes can be concluded as follows:

(1) Previous allocation policies always proposed a universal page migration scheme. However, little research has been conducted to show policies particular in the image-processing domain.

(2) Previous allocation policies had the possibility to incur unnecessary page migrations, which had the potential to increase the overhead of page management.

(3) Almost all of the previous allocation policies required the intervention of OS more or less to record the migrated information even in RaPP. Although the OS-assisted page migration method may help conserve energy, compared with hardware-based design for memory controller, it is not on the path of most memory accesses and does not respond immediately and efficiently. The hardware-based design for memory controller could swap memory frames between fast and slow memory immediately without modifications to the OS and it is transparently to the OS. Applications can experience reasonable performance benefits without modifications to the operating systems [22], [23]. Of course, the hardware-based design for memory controller also needs the interference of the operating system while processing the workloads for the basic functions.

Thus, in our investigation, we delve into seeking the methods to solve the follows problems:

(1) Through analyzing the image processing applications, posing a better estimator to predict the future memory writes as accurately as possible.

(2) Basing on the observations, presenting an efficient memory page migration scheme implemented by the memory controller of hybrid system, which is particular for image processing applications and could place frequently written pages in DRAM with limited migrations (efficiently).

III. FEATURE EXPLORATION FOR IMAGE PROCESSING APPLICATIONS

To reduce the write traffic that occurs in PCM devices, DRAM should absorb as many write frequency pages as possible. For this purpose, it is necessary to find a good estimator for predicting future write references. Frequency and temporal locality are critical properties which are used to estimate future references [15]. In order to apply these practices to the design of memory controller, it is important to characterize the degree of locality presented in image processing applications.

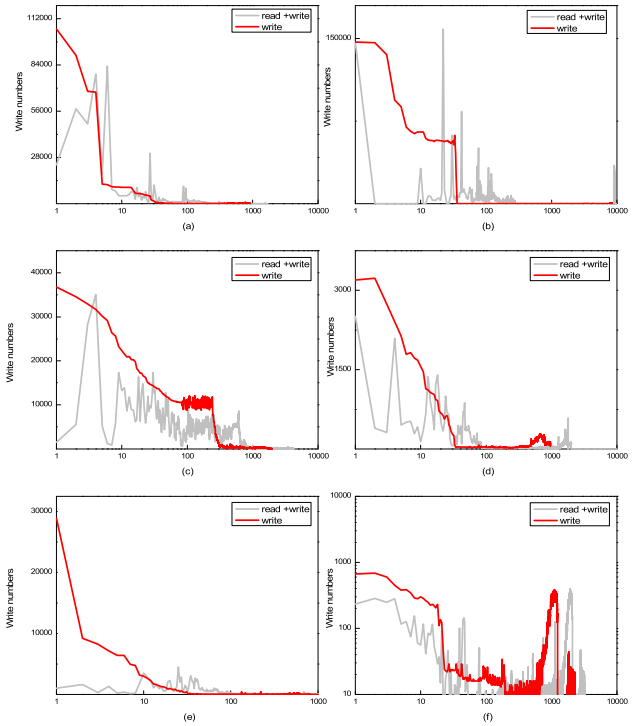


FIGURE 2. Write numbers that occur versus frequency of ranking write frequency only and read+write frequency. (a) Dijkstra pages ranking. (b) FFT pages ranking. (c) h264enc pages ranking. (c) h264enc pages ranking. (d) jpeg_dec pages ranking. (d) jpeg_dec pages ranking. (f) mpeg2dec pages ranking.

For our investigation, we captured the virtual memory access traces through a modified gem5 simulator [24]–[27]. The cache traces were filter out and only the main memory references were observed. We captured main memory access traces from six different image processing applications or algorithms: dikkstra, FFT, mpeg2dec, h264enc, jpeg_enc, jpeg_dec, which are from MediaBench [28] and Mibench [29]. These benchmarks are widely used in both industry and academia for evaluating the CPU, cache, and main memory performance of embedded systems.

A. WRITE REFERENCE COUNT

According to CLOCK-DWF [15], using write history alone leads to better estimation of future write references than using both read and write history. We use the number of accesses that counts to the current point which is called so-far frequency in [15] to curve Fig. 2 as CLOCK-DWF does. The x-axis represents the ranking of pages based on their so-far write frequency (red plot) and so-far read/write frequency (grey plot) respectively, implying that the page with the highest ranking is that has the highest frequency count. The y-axis represents the number of writes occurring in the page on the fixed ranking. The construction for Fig. 2 is elaborated as follows: when a write reference occurs on page A in the memory, the write counts of the ranking for page A will be increased by one. After page A has been written to the memory, the ranking for page A would move up to the corresponding ranking point basing on the reference count of page A. This procedure will iterate for each page in applications until the processing ends. In Fig. 2, the ranking in x-axis is basing on so-far frequency which meant the total accesses to the current timing point for this page. The red plots are located above the gray plots for most cases as can be seen from parts (a) to (f) of Fig. 2.The phenomena demonstrate the conclusion [15] that using write history alone is better than using both read/write history in estimating future write references (especially in top 10 rankings for image processing applications).

Except stated before, as is shown in Fig. 2, two noticeable phenomena can be demonstrated:

(1) Pages with higher history write frequency have a higher possibility to be rewritten in image processing applications. Consequently, using so-far write frequency to estimate future writes may be a favorable choice. As can be seen in Fig. 2, the shape of the curves in these figures can be modeled as a monotonic decreasing function. The write frequency decreases with the decreasing of the pages ranking. Most of the write operations take place on the pages that ranking before the inflection point. The write counts gap before and after the inflection point is in magnitude level. The steeply decline near the inflection point clearly divides the pages into write hot pages ranking and write cold pages ranking which implies that in image processing applications the distinction between write hot pages and write cold pages is obvious. A write hot page is likely to be rewritten with a higher probability. Specially, in Fig. 2(a)–(e), the huge gap before and after the inflection point infers that the write frequency in write hot pages and write cold pages is distinguished evidently in these applications. Compared with the applications in Fig. 2(a)-(e), the gap in mpeg2dec is not exceedingly obvious. This may because the images processing in mpeg2dec are interrelated and iterative. The pages used last time have a higher possibility to be reused in the next execution; consequently, the write frequency gap between write hot pages and write cold pages is not so obviously. However, the overall trend in Fig. 2(f) is almost identical to the others; the conclusion is also applicable to mpeg2dec.

(2) Typically, only a relatively small subset of pages is write hot pages during the execution of image processing applications. The location of the inflection point is noteworthy which appears around the ranking 100 (or less than 100) point. Before the inflection point, a narrow range of top ranking pages own most of the write operations in image processing applications. The observed result manifests that the write hot pages is only a small portion of the total pages, which does not need frequent migration times.

B. LOCALITY OF REFERENCE

The reason that system could get a good hit-rate in processing is that the memory reference steam is not random. Most applications have the natures of temporal locality and spatial locality [30], [31], [41]. A more recently referenced item has a high probability to be referenced in the immediate future [32].

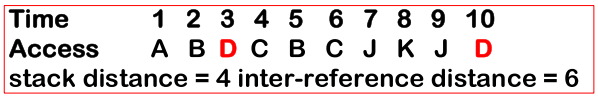


FIGURE 3. An example for stack distance and inter-reference distance.

This is called temporal locality. The items physically located near the most recently referenced item are likely to be re-referenced in the near future [33], [34]. This is called spatial locality. Therefore, keeping recently accessed items in the memory system can result in a remarkable number of hits. It is possible to make qualitative statements about locality. To take spatial locality into consideration, researchers perform migration in page granularity. In terms of the temporal locality, stack distance [35]–[39], could be a favorable choice. However, it is dynamically changing over time. Using hardware to calculate the stack distance for each page is ridiculous which would cause great hardware overhead. As mentioned above, it is indispensable to make qualitative statements about temporal locality. The number of other references between two references to the same address in the trace is called inter-reference distance [40]. In this paper, we use inter-reference distance to approximately exploit the temporal locality of image processing applications for the reason that the inter-reference distance can be denoted by the request number that the memory controller receives implemented by a global request counter. Fig. 3 shows an example of the discrimination between stack distance and inter-reference distance. The stack distance and inter-reference distance between two D is 4 and 6 respectively.

The number of pages with the same inter-reference distance is shown in Fig. 4. The x-axis represents the inter-reference distance in the image processing applications ranging from 1 to 10000. The y-axis represents the number of pages occurs in the fixed inter-reference distance.

Two main peaks are located in Fig. 4 (a) (b) and (c) and three main peaks appeared in the Fig. 4(d) (e) (f). The gap between main peaks and the rest peaks is huge, which indicates that most pages have the inter-reference distance value near the main peaks while accessing. Although the number of main peaks is different in these applications, the frontier peaks are located in the inter-reference distance ranging from 10-100 for all the cases. This phenomenon implies that an ocean of pages will be re-accessed when approximately 100 requests come. However, the last peaks for most cases are located between 100 and 1000 inter-reference distance (in particular, the location of the second peak in h264enc is near 1000), which infers that a multitude of pages would be re-accessed while 1000 requests occurs. In conclusion, most accessed pages will be re-accessed within 1000 inter-reference distance.

Based on the commonality characteristic observed in Fig. 2 and Fig. 4, we can conclude that if the so-far write frequency of a PCM page has exceeded the migration threshold and the dirty bit of this PCM page has been set in the previous 1000 memory accesses. At this time, it could

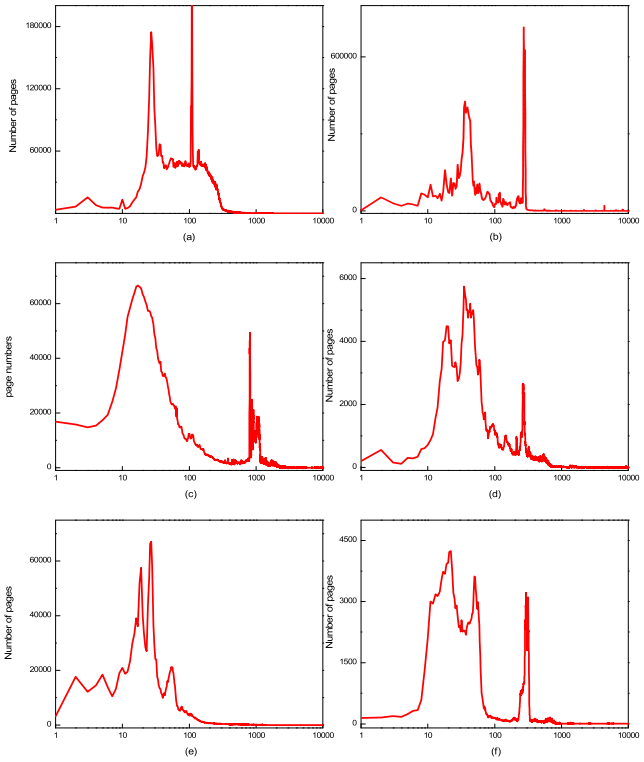


FIGURE 4. The number of pages with the same inter-reference distance. (a) Dijkstra inter-reference distance. (b) FFT inter-reference distance. (c) FFT inter-reference distance. (d) jpeg_dec inter-reference distance. (e) jpeg_enc inter-reference distance. (f) mpeg2dec inter-reference distance.

be judged as a write hot page. Then, if a pcm write occurs in this page, the system will start the migration. Meanwhile, the 1000 inter-reference distance is capable of filtering the write cold pages out. In memory controller, we can record this access information in an extra buffer.

In summary, combining inter-reference distance with history write frequency could be a better estimator in predicting future memory writes appropriately. Such method may present even better result.

IV. WIRD MIGRATION SCHEME

Basing on the observation in section III, in this section, we propose a novel page migration scheme named WIRD (write frequency with inter-reference distance) for the hybrid main memory implemented by memory controller. The proposed allocation scheme has the possibility of estimating the future write references accurately with limited migration times (we called it high efficiency) for image processing applications. Our scheme minimizes the unnecessary page migrations and makes frequent memory writes absorbed by DRAM. The high efficiency character has the potential to decrease the latency overhead that unnecessary page migrations cause. Our intention is to judge write intensive pages accurately and allocate these pages to DRAM with limited migration times.

As the conclusions stated in section III (A), the pages with higher history write frequency have a higher possibility to be

```
op :operation   p: page   Input op, p   Output NULL
procedure WIRD   (page p , operation op)
1.if ( p  ∈DRAM )
2.   then update_page_in_DRAM (p);
3.else        /*p is in PCM*/
4.    if (op is read )
5.          then no change in write_counter(p) and dirty_bit(p);
6.   else (op is write)
7.          write_counter(p)++;
8.          if   (write_counter(p)> = migration threshold
9.          && dirty_bit(p) = = 1
10.         && get_victim_page_in_DRAM()!=NULL)
11.               then swap p and victim_page ;
12.         else   no migration;
13.         endif
14.   endif
15.else no migration;
16.endif
17.request_counter++;
18.PCM_Reset();
19.if (op is write && p is in PCM )
20.   then dirty_bit(p)=1;
21.else no operation;
22.endif
end procedure
```

```
procedure   update_page_in_DRAM(p)
  23. expired time(p)=currenttime+8;
  24. dirty_bit(p)=1;
end procedure
procedure   PCM_Reset()
  25. if ( request_counter %1000 = = 0)
  26.       then reset dirty_bit in PCM each page;
  27. else
  28.       no change in dirty_bit;
  29. endif
end procedure
procedure   get_victim_page_in_DRAM()
  30. if (free page p in DRAM || dirty_bit (page p)==0
  31. || currenttime>expired time(p) )   then return p ;
  32. else return NULL;
  33. endif
 end procedure
```

FIGURE 5. WIRD algorithm procedure.

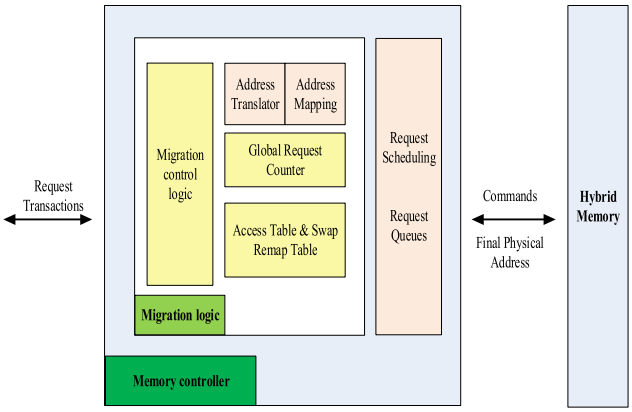


FIGURE 6. The structure of memory controller with WIRD.

rewritten in image processing applications while processing. A so-far global write counter is used to record the write counts for each page in PCM. For the conclusion obtained in section III (B), if the so-far write frequency of a PCM page has exceeded the migration threshold and the dirty bit of this PCM page has been set, it could be judged as a write hot page which has the opportunity to be scheduled for migrating to DRAM. We use a global request counter recording the request number that the memory controller receives to denote the inter-reference distance.

The details of WIRD algorithm scheme are organized in Fig. 5. In the WIRD page migration scheme, with the conclusion gotten from section III, inter-reference distance 1000 is used to tracking the temporal locality for distinct image processing applications. A global request counter is used in memory controller to indicate inter-reference distance. Dirty bit is also used in both PCM and DRAM per page, but the role dirty bit plays is different. The dirty bit in PCM records that whether this page has been written in the current 1000 requests while the dirty bit in DRAM indicates whether this page has been accessed in the long-term distance (including write and read). The dirty bit of each page in PCM will be reset every 1000 requests (line 25 to 29).

Every time the PCM block is writing accessed, its reference write counter is incremented (line 7). In addition, the dirty bit sets to 1 for this frame (line17-20). PCM_Reset function will check whether the dirty bit of each page in PCM should reset to 0 every 1000 quests (line25-29). After a certain number of accessing to the blocks, pages stored in PCM have become write hot pages and are to be scheduled for migration to the fast memory. The page whose write reference counter reaches the migration threshold and the dirty bit has been set to 1 in the current 1000 accesses has the opportunity to be scheduled for migrating to DRAM. A victim DRAM page destination must be selected for the migrating page (line 8-11). The free page, unreferenced page and the expired page in DRAM will be selected as a victim page (30-33). We deal with the DRAM page whose consecutive accesses number is more than 8 times as an expired page (line23-24). If such DRAM page does not exist, no victim destination is selected and the migration would be delayed until such page appears.

The memory controller designed with WIRD is depicted in Fig. 6. It receives the read/write requests via the request transactions. As presented in Fig. 6, a migration control logic in memory controller is implemented. In this particular case, the address is translated as normal in address translator which takes the request address and extracts the specific bits in the address to determine the destination of it. A global request counter is added to calculate the request number. An access table is added to record the information for the pages in hybrid memory. A swap remap table is added to make sure the correction of the final physical address. This table could record the original physical address and migrated address for migrated pages. When a request accessed a specific address, memory controller would look up the swap remap table in memory controller to see whether the page has been migrated or not. After checking the swap remap table, memory controller will send the corresponding commands with the correct address (final physical address) to the hybrid memory.

V. EXPERIMENT AND PERFORMANCE EVALUATION

In this section, we evaluate the performance of WIRD scheme using the gem5-nvmain simulator [24]–[27]which is a hybrid memory system simulator that could accurately characterize the performance of PCM and DRAM combination architecture. Syscall Emulation (SE) mode with out of order CPU is adopted in gem5-nvmain simulator, which could guarantee the system running environment. In this experimental environment, detailed simulation configurations for main memory are listed in Table 2.

We show the performance evaluation of WIRD compared with no migration (unmanaged), RaPP, M-CLOCK and AC-CLOCK. The symbols that we use are listed in Table 3.

A. EFFICIENCY MIGRATION

Our proposed scheme attempts to estimate the write references accurately with limited migration times (high

TABLE 2. Simulation configuration.

Parameter	value
L1 I / D cache	16KB 4-way set associative, 64-byte cache line size
L2 Cache(shared)	512KB 8-way set associative, 64-byte cache line size
Cache block size/OS page size	64B/1KB
Memory (DDR3 -1333)	DDR3 -1333 1152MB open page

TABLE 3. Symbol definitions.

symbol	description
$\mathbf{num_{pw\_algorithm}}$	number of write operations in PCM for the current algorithm
migration count	number of migrations in the current algorithm
migrationaccesses	number of references on the migrated pages after migration
$\mathbf{T_1}$	average **DRAM** read latency
$\mathbf{T_2}$	average **DRAM** write latency
$\mathbf{T_3}$	average **PCM** read latency
$\mathbf{T_4}$	average **PCM** write latency
$\mathbf{T_{ADRAM}}$	average **DRAM** access latency
$\mathbf{T_{APCM}}$	average **PCM** access latency
$\mathbf{T_A}$	average memory access latency
$\mathbf{N_{dram\_read}}$	number of read operations in **DRAM**
$\mathbf{N_{dram\_write}}$	number of write operations in **DRAM**
$\mathbf{N_{pcm\_read}}$	number of read operations in **PCM**
$\mathbf{N_{pcm\_write}}$	number of write operations in **PCM**
$\mathbf{N_{DRAM}}$	number of operations in **DRAM**
$\mathbf{N_{PCM}}$	number of operations in **PCM**

efficiency) for the image processing applications.

$$\text{AWRPM} = \frac{\mathbf{num_{pw\_umanaged} - num_{pw\_algotithm}}}{\mathbf{migrationcount}} \quad (1)$$

$$\text{MAPP} = \frac{\mathbf{migrationaccesses}}{\mathbf{migrationcount}} \quad (2)$$

Using the conclusions obtaining by analyzing the relation features in the image processing applications, WIRD reduces unnecessary page migrations by accurately picking the most write hot pages out. The migration count of WIRD is only 49.44%, 90.84%, 88.35% of RaPP, M-CLOCK and AC-CLOCK respectively on average. The high efficiency feature of WIRD can be thoroughly clarified in Fig. 7. Fig. 7(a) manifests the average write operations reduction per migration (AWRPM), which certifies the high efficiency of our proposed algorithm sufficiently. AWRPM is calculated in (1). In particular, for each migration, we reduce the average PCM write operation per migration by 2.57 times,1.48 times and 1.44 times on average than that of RaPP, M-CLOCK and AC-CLOCK respectively as depicted in Fig. 7(a). This presents a performance benefit (write operations reduction) that can be achieved in the hybrid main memory through one page migration from PCM. RaPP starts migrating based on both read and write references which could not pick the write hot pages out accurately and may bring extra migrations in the running procedure inevitably. M-CLOCK does not take the dynamic access patterns of the workloads into account

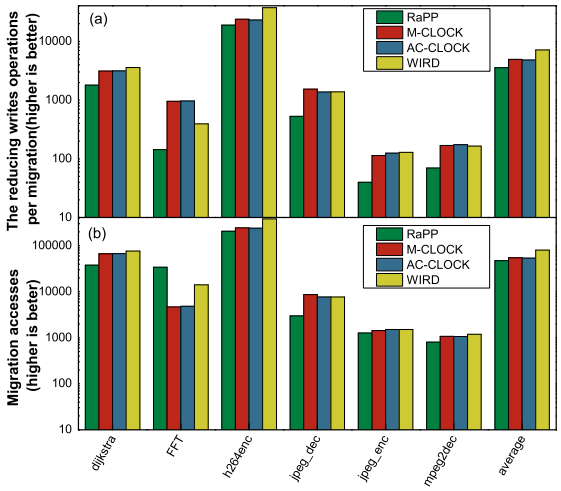


FIGURE 7. (a) Average write operations reduction per migration. (b) Migration accesses for per migrated page.

which may lead to the same problem that RaPP meets. AC-CLOCK uses a saturation counter to calculate the write frequency in the DRAM (cleared one turn) which could not distinguish the write frequency level clearly. It has a higher probability to allocate the write hot pages into PCM while finding victims. Excessive page migrations may result in various problems, such as the increasing of energy consumption while exchanging pages in hybrid memory system and the hit ratio decreasing in DRAM. More seriously, it may offset the benefits of the write reduction in PCM and adversely cause an increase of write operations while exchanging pages in PCM which would lead to a lifetime reduction for PCM.

Fig. 7 (b) presents the migration accesses per migrated page (MAPP: calculated in (2)) which evidences the popularity of the migrated page on average. One can find that the MAPP in WIRD is 1.69 times,1.46 times,1.49 times than that of RaPP, M-CLOCK and AC-CLOCK on average respectively. Such phenomenon supports the hypothesis that WIRD mechanism predicts the most popular write hot pages accurately and migrates them to DRAM with limited migration times. WIRD prevents the unnecessary page migrations from PCM to DRAM through a sum of contributions of write frequency and temporal locality representing by inter-reference distance; meanwhile, it allows essential pages (most write hot pages) migrate from PCM to DRAM.

B. AVERAGE WRITE ACCESS TIME IN PCM

Fig. 8 presents the detailed average memory access time for different benchmarks. While implementing efficient write migrations, our algorithm also reduces the average write latency of operations in PCM compared with other policies. There are four main factors affecting the average memory access delay: average DRAM read latency (T_1), average DRAM write latency(T_2), average PCM read latency(T_3), average PCM write latency(T_4). Formula (3)-(7) interpret the

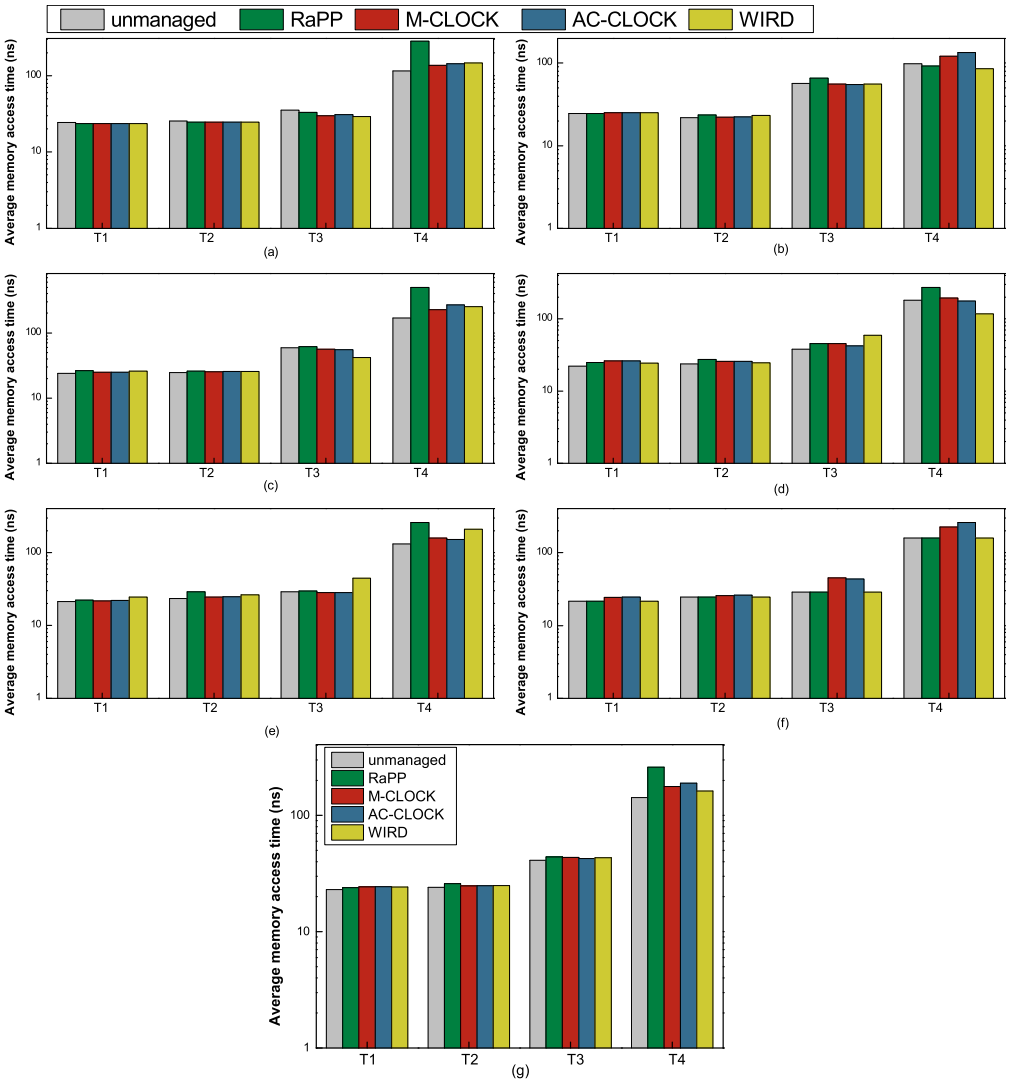


FIGURE 8. Detailed average memory access time. (a) Dijkstra. (b)FFT. (c) h264enc. (d) Peg_dec. (e) Peg_enc. (f) mpeg2dec. (g) Average.

interaction of these factors.

$$T_{ADRAM} = \frac{T_1 * N_{dram\_read} + T_2 * N_{dram\_write}}{N_{dram\_read} + N_{dram\_write}} \quad (3)$$

$$T_{APCM} = \frac{T_3 * N_{pcm\_read} + T_4 * N_{pcm\_write}}{N_{pcm\_read} + N_{pcm\_write}} \quad (4)$$

$$N_{DRAM} = N_{dram\_read} + N_{dram\_write} \quad (5)$$

$$N_{PCM} = N_{pcm\_read} + N_{pcm\_write} \quad (6)$$

$$T_A = \frac{T_{ADRAM} * N_{DRAM} + T_{APAM} * N_{PCM}}{N_{DRAM} + N_{PCM}} \quad (7)$$

As shown in Fig. 8, The DRAM effective page access time of WIRD is similar to that of the other policies. However, WIRD greatly reduces the PCM effective page write access time by an average of 37.68%, 8.46% and 14.43% compared with RaPP, M-CLOCK and AC-CLOCK respectively.

As stated before, WIRD predicts the future writes more accurately and well holds write intensive pages in DRAM. Most of the pages left in PCM are not written frequently. A large number of read hot pages will remain in PCM or migrate from DRAM to PCM. This makes the huge gap between read and write numbers in PCM. Fig. 9 presents the proportions of read and write operations in PCM and DRAM. As shown in Fig. 9, WIRD effectively holds read operation ratio by an average of 34.92% of the total read commands in PCM which is 1.91 times, 1.16 times and 1.19 times than that of RaPP, M-CLOCK and AC-CLOCK respectively. The interaction between the large number of read operations and the small number of write operations in PCM results in a reduction in the read/write switching rate. The read/write switch times has significant impact on the reduction of page write access time in PCM [42]–[44].

Fig. 10 presents the read/write switching rate for various algorithms. As shown in Fig. 10 (a) and (b), our algorithm reduces the read/write switching rate in PCM. Especially the write to read switching rate is greatly reduced by an

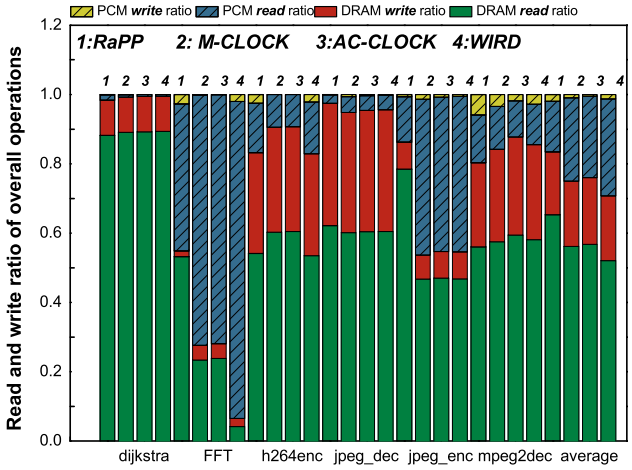


FIGURE 9. The proportion distribution of read and write operations in the hybrid system.

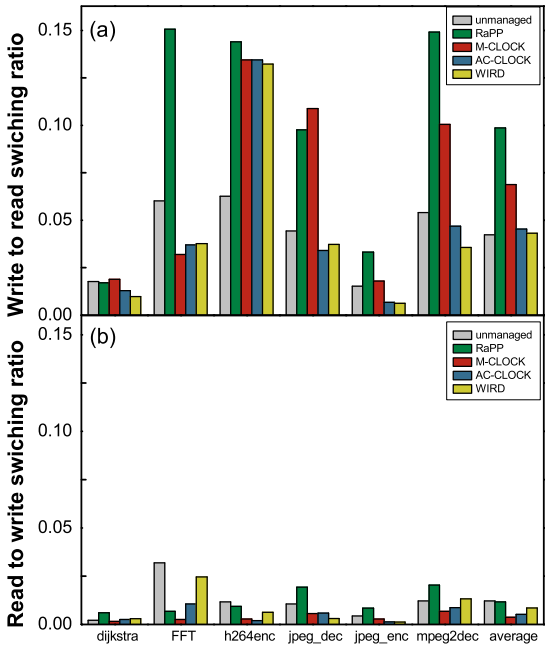


FIGURE 10. Read/write switching ratio.

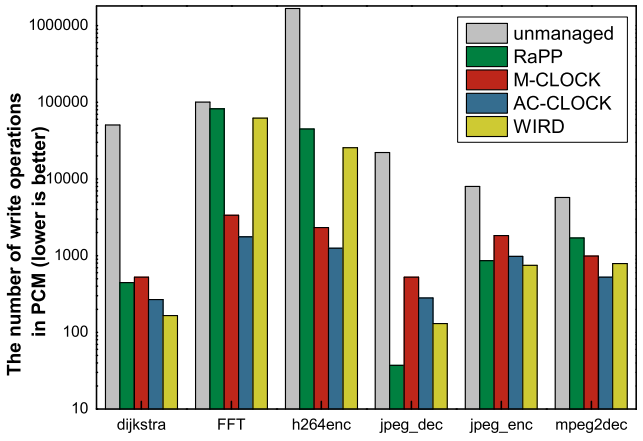


FIGURE 11. The number of write operations in PCM.

average of 56.18%, 37.17% and 2.23% compared with RaPP, M-CLOCK and AC-CLOCK respectively.

One can find that, in Fig. 10, no matter in which algorithm, the number of write to read switching operations is significantly higher than the number of read to write switching operations. The delay of read to write switching operation is significantly less than the delay of the write to read switching operation. To ensure the accuracy of subsequent read operations, the read-after-write operation must wait for the contents of the previous write operation to be stable in the row buffer (the date bus should wait for T_{WTR}). Then, the subsequent read operation can be performed. This mechanism opens the route to the realization of the data of previous write operation being correctly stored in the specific row. These operations greatly reduce the bus utilization. Therefore, in WIRD, the large drop in the write-to-read switching rate greatly reduces the read and write reverse pause delay of data bus. Fig. 10 provides the best agreement for the decreasing of PCM effective page write access time as shown in Fig. 8.

Fig. 11 shows the number of write operations in PCM for unmanaged, RaPP, M-CLOCK, AC-CLOCK and WIRD. As shown in Fig. 11, the PCM write count of WIRD is significantly decreased than the other algorithms for most cases. Specifically, WIRD reduces the write operations in PCM by an average of 33. 69%, 86.85%, and 63.08%, 99.67% at maximum compared with RaPP and umanaged respectively.

Maybe in Fig. 11, for jpeg_dec, the write count in WIRD(130) is a little higher than that of RaPP(37), but the gap between them is smaller than other applications. WIRD algorithm uses only 16 migrations which is much smaller than that of RaPP (42). This exactly implies that our proposed scheme estimates the write references accurately with limited migration times (high efficiency).

As to FFT, most pages in FFT have large read operations after the write operation. Then, the next write operation would occur in another page. This phenomenon indicates that in FFT there are a lot of distributed not-so-hot write pages. WIRD leaves large number of read operations in PCM (as shown in Fig. 9) and the MAPP in WIRD is much higher than that of the other policies. The results mentioned above certificate that WIRD accurately picks much hotter write pages out than M-CLOCK and AC-CLOCK do. Although other policies reduce much write operations than WIRD, the contribution to the reduction is the excessive migrations including many not-so-hot write pages. The reduction of PCM write access latency in WIRD (compared with M-CLOCK and AC-CLOCK) also provides the best agreement for the accuracy character in the proposed algorithm.

To workloads h264enc and mepg2dec, the capability in reducing the write operations of PCM may not as well as M-CLOCK and AC-CLOCK do. However, one can find that in Fig. 7, the MAPP and AWRPM in WIRD are much higher

than that of M-CLOCK and AC-CLOCK. This phenomenon indicates that, in M-LCOCK and AC-CLOCK, the migrated pages are not the so hot pages and the contributions to the reduction in operations are belonging to the excessive migrations in these two algorithms. In Fig. 8(c) and (f), the PCM average write access latencies of these two workloads are lower than that of M-CLOCK and AC-CLOCK. The reason for this phenomenon is that these applications running with M-CLOCK and AC-CLOCK have higher migration times which result in the increasing of PCM write queue latency while exchanging pages in PCM. The PCM write queue latency increasing leads to a higher PCM write access time. Consequently, WIRD can significantly reduce the write operations in PCM while obtaining lower PCM write access latency.

VI. CONCLUSION

PCM has been adopted as one of the most promising technologies to be incorporated into the memory hierarchy in order to enlarge the capacity of memory and reduce the processing latency of embedded systems.

In this paper, we analyzed the characteristics of image processing applications and found that inter-reference distance with history write frequency may be a better estimator in predicting the future memory writes in image processing applications. Based on these observations, we presented a novel page migration scheme for memory controller called WIRD.

Our scheme accurately picks the write hot page out and makes frequent memory writes absorbed by DRAM with limited migrations (high efficiency). Experimental results show that WIRD reduces AWRPM by 2.57 times, 1.48 times and 1.44 times on average than that of RaPP, M-CLOCK and AC-CLOCK respectively. The MAPP in WIRD is 1.69 times,1.46 times,1.49 times than that of RaPP, M-CLOCK and AC-CLOCK on average respectively. Such phenomena support the efficiency and accuracy characteristics of our proposed mechanism. Meanwhile, the high efficiency character of WIRD has decreased the read/write switching ratio in the system that makes the PCM effective page write access time reduced by an average of 37.68%, 8.46% and 14.43% compared with RaPP, M-CLOCK and AC-CLOCK respectively.

REFERENCES

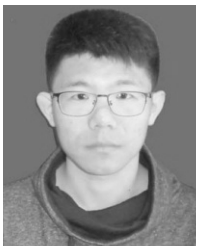
[1] R. Jothin and C. Vasanthanayaki, "High performance static segment on-chip memory for image processing applications," *J. Electron. Test.*, vol. 34, no. 4, pp. 389–404, 2018.
[2] N. Lawal and M. O'Nils, "Embedded FPGA memory requirements for real-time video processing applications," in *Proc. IEEE 23rd NORCHIP Conf.*, Nov. 2005, pp. 206–209.
[3] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," in *Proc. Int. Symp. Comput. Archit. (ISCA)*, vol. 37, no. 3, pp. 14–23, Jun. 2009.
[4] L. E. Ramos, E. Gorbatov, and R. Bianchini, "Page placement in hybrid memory systems," in *Proc. ACM Int. Conf. Supercomput.*, 2011, pp. 85–95.
[5] G. Wu, H. Zhang, Y. Dong, and J. Hu, "CAR: Securing PCM main memory system with cache address remapping," in *Proc. IEEE 18th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2012, pp. 628–635.
[6] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," *ACM SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 24–33, 2009.
[7] R. F. Freitas and W. W. Wilcke, "Storage-class memory: The next storage system technology," *IBM J. Res. Develop.*, vol. 52, no. 4.5, pp. 439–447, 2008.
[8] S. Mittal and J. S. Vetter, "A survey of software techniques for using non-volatile memories for storage and main memory systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1537–1550, May 2016.
[9] S. Eilert, M. Leinwander, and G. Crisenza, "Phase change memory: A new memory enables new memory usage models," in *Proc. IEEE Int. Memory Workshop (IMW)*, May 2009, pp. 1–2.
[10] S. Mittal and J. S. Vetter, "A survey of architectural approaches for data compression in cache and main memory systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1524–1536, May 2016.
[11] G. Dhiman, R. Ayoub, and T. Rosing, "PDRAM: A hybrid PRAM and DRAM main memory system," in *Proc. ACM 46th Annu. Design Automat. Conf.*, Jul. 2009, pp. 664–669.
[12] W. Zhang and T. Li, "Exploring phase change memory and 3D die-stacking for power/thermal friendly, fast and durable memory architectures," in *Proc. 18th Int. Conf. Parallel Archit. Compilation Techn. (PACT)*, Sep. 2009, pp. 101–112.
[13] K. Chen, P. Jin, and L. Yue, "A novel page replacement algorithm for the hybrid memory architecture involving PCM and DRAM," in *Proc. IFIP Int. Conf. Netw. Parallel Comput.* Berlin, Germany: Springer, 2014, pp. 108–119.
[14] Z. Wu, P. Jin, C. Yang, and L. Yue, "APP-LRU: A new page replacement method for PCM/DRAM-based hybrid memory systems," in *Proc. IFIP Int. Conf. Netw. Parallel Comput.* Berlin, Germany: Springer, 2014, pp. 84–95.
[15] 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 Trans. Comput.*, vol. 63, no. 9, pp. 2187–2200, Sep. 2014.
[16] M. Lee, D. H. Kang, J. Kim, and Y. I. Eom, "M-CLOCK: Migration-optimized page replacement algorithm for hybrid DRAM and PCM memory architecture," in *Proc. ACM Symp. Appl. Comput.*, 2015, pp. 2001–2006.
[17] R. Salkhordeh and H. Asadi, "An operating system level data migration scheme in hybrid DRAM-NVM memory architecture," in *Proc. Conf. Design, Automat. Test Eur.*, 2016, pp. 936–941.
[18] X. Chen, E. H.-M. Sha, W. Jiang, C. Yang, T. Wu, and Q. Zhuge, "Refinery swap: An efficient swap mechanism for hybrid DRAM–NVM systems," *Future Gener. Comput. Syst.*, vol. 77, pp. 52–64, Dec. 2017.
[19] Z. Zhang, Y. Fu, and G. Hu, "DualStack: A high efficient dynamic page scheduling scheme in hybrid main memory," in *Proc. Int. Conf. Netw., Archit., Storage (NAS)*, Aug. 2017, pp. 1–6.
[20] S. Kim, S.-H. Hwang, and J. W. Kwak, "Adaptive-classification CLOCK: Page replacement policy based on read/write access pattern for hybrid DRAM and PCM main memory," *Microprocessors Microsyst.*, vol. 57, pp. 65–75, Mar. 2018.
[21] J. Hu, W.-C. Tseng, C. J. Xue, Q. Zhuge, Y. Zhao, and E. H.-M. Sha, "Write activity minimization for nonvolatile main memory via scheduling and recomputation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 4, pp. 584–592, Apr. 2011.
[22] X. Dong, Y. Xie, N. Muralimanohar, and N. P. Jouppi, "Simple but effective heterogeneous main memory with on-chip memory controller support," in *Proc. ACM/IEEE Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2010, pp. 1–11.
[23] J. Sim, A. R. Alameldeen, Z. Chishti, C. Wilkerson, and H. Kim, "Transparent hardware management of stacked dram as part of memory," in *Proc. IEEE/ACM 47th Annu. Int. Symp. Microarchitecture*, 2014, pp. 13–24.
[24] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, 2011.
[25] A. Butko, R. Garibotti, L. Ost, and G. Sassatelli, "Accuracy evaluation of gem5 simulator system," in *Proc. 7th Int. Workshop Reconfigurable Commun.-Centric Syst.-Chip (ReCoSoC)*, 2012, pp. 1–7.
[26] M. Poremba and Y. Xie, "NVMain: An architectural-level main memory simulator for emerging non-volatile memories," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Aug. 2012, pp. 392–397.
[27] M. Poremba, T. Zhang, and Y. Xie, "NVMain 2.0: A user-friendly memory simulator to model (non-)volatile memory systems," *IEEE Comput. Archit. Lett.*, vol. 14, no. 2, pp. 140–143, Jul./Dec. 2015.

[28] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "MediaBench: A tool for evaluating and synthesizing multimedia and communications systems," in *Proc. ACM/IEEE 30th Annu. Int. Symp. Microarchitecture*, Dec. 1997, pp. 330–335.
[29] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *Proc. IEEE 4th Annu. Int. Workshop Workload Characterization (WWC)*, Dec. 2001, pp. 3–14.
[30] S. Lee, H. Bahn, and S. H. Noh, "Characterizing memory write references for efficient management of hybrid PCM and DRAM memory," in *Proc. IEEE 19th Annu. Int. Symp. Modelling, Anal., Simulation Comput. Telecommun. Syst. (MASCOTS)*, Jul. 2011, pp. 168–175.
[31] K. Namba and F. Lombardi, "On coding for endurance enhancement and error control of phase change memories with write latency reduction," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 2, pp. 230–238, Feb. 2018.
[32] S. Jin and A. Bestavros, "GreedyDual Web caching algorithm: Exploiting the two sources of temporal locality in Web request streams," *Comput. Commun.*, vol. 24, no. 2, pp. 174–183, 2001.
[33] E. Lee, J. E. Jang, T. Kim, and H. Bahn, "On-demand snapshot: An efficient versioning file system for phase-change memory," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 12, pp. 2841–2853, Dec. 2013.
[34] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, "Characterizing reference locality in the WWW," in *Proc. 4th Int. Conf. Parallel Distrib. Inf. Syst.*, Dec. 1996, pp. 92–103.
[35] C. Ding and Y. Zhong, "Predicting whole-program locality through reuse distance analysis," *ACM Sigplan Notices*, vol. 38, no. 5, pp. 245–257, 2003.
[36] M. Snir and J. Yu, "On the theory of spatial and temporal locality," Univ. Illinois Urbana Champaign, Champaign, IL, USA, Tech. Rep. UIUCDCS-R-2005-2611, 2005.
[37] S. Kumar and C. Wilkerson, "Exploiting spatial locality in data caches using spatial footprints," in *Proc. 25th Annu. Int. Symp. Comput. Archit.*, 1998, pp. 357–368.
[38] T. L. Johnson, M. C. Merten, and W. W. Hwu, "Run-time spatial locality detection and optimization," in *Proc. IEEE/ACM 30th Annu. Int. Symp. Microarchitecture*, Dec. 1997, pp. 57–64.
[39] H. Luo, P. Dai, L. Shi, C. J. Xue, Q. Zhuge, and E. H. M. Sha, "Write reconstruction for write throughput improvement on MLC PCM based main memory," *J. Syst. Archit.*, vol. 71, pp. 62–72, Nov. 2016.
[40] V. Phalke and B. Gopinath, "An inter-reference gap model for temporal locality in program behavior," Tech. Rep., 1995.
[41] G. Almási, C. Cascaval, and D. A. Padua, "Calculating stack distances efficiently," *ACM Sigplan Notices*, vol. 38, no. 2, pp. 37–43, 2002.
[42] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, "Memory access scheduling," *ACM SIGARCH Comput. Archit. News. ACM*, vol. 28, no. 2, pp. 128–138, 2000.
[43] M. K. Qureshi, M. M. Franceschini, and L. A. Lastras-Montaño, "Improving read performance of phase change memories via write cancellation and write pausing," in *Proc. 16th Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Jan. 2010, pp. 1–11.
[44] N. Chatterjee, N. Muralimanohar, R. Balasubramonian, A. Davis, and N. P. Jouppi, "Staged reads: Mitigating the impact of DRAM writes on DRAM reads," in *Proc. IEEE 18th Int. Symp. High-Perform. Comp Archit. (HPCA)*, Feb. 2012, pp. 1–12.

NA NIU received the M.S. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2016. She is currently pursuing the Ph.D. degree with the Microelectronics Center, Harbin Institute of Technology. Her research interests include hybrid memory design, very large-scale integration design, and system on chips (SoC).

FANGFA FU received the M.S. and Ph.D. degrees in microelectronics and solid-state electronics from the Harbin Institute of Technology, Harbin, China, in 2007 and 2012, respectively, where he has been a Lecturer with the Microelectronics Center, since 2012. His research interests include the system on chips (SoC), networks on chips (NoC), very large-scale integration design, and digital signal processing.

BING YANG received the M.S. and Ph.D. degrees in microelectronics and solid-state electronics from the Harbin Institute of Technology, Harbin, China, in 2002 and 2009, respectively. He is currently with the Harbin University of Science and Technology. His research interests include the system on chips (SoC), very large-scale integration design, and computer architecture.



JIACAI YUAN received the B.S. degree from the Harbin Institute of Technology, Harbin, China, in 2018, where he is currently pursuing the M.S degree with the Microelectronics Center. His research interests include hybrid memory design, very large-scale integration design, and system on chips (SoC).

FENGCHANG LAI received the B.S. degree from the Harbin Institute of Technology, Harbin, China, in 1984, where he is currently a Professor with the Microelectronics Center. His research interests include very large-scale integration design and SoC.

JINXIANG WANG received the B.S. and M.S. degrees in semiconductor physics and the Ph.D. degree in communication and information engineering from the Harbin Institute of Technology, Harbin, China, in 1990, 1993, and 1999, respectively, where he is currently a Professor with the Microelectronics Center. His research interests include very large scale integration design, wireless communication, SoC, and NoC.