Int J Parallel Prog (2017) 45:4–16
DOI 10.1007/s10766-015-0382-5

An Optimal Page-Level Power Management Strategy in PCM–DRAM Hybrid Memory

Jinbao Zhang[1] · Xiaofei Liao[1] · Hai Jin[1] · Dong Liu[1] · Li Lin[1] · Kao Zhao[1]

Received: 30 March 2015 / Accepted: 20 May 2015 / Published online: 3 October 2015

Abstract The new emergence of data-intensive applications raises a huge requirement on the capacity of memory. However, an obvious increase of memory makes the corresponding energy consumption unacceptable in practice. To address this question, any effort only to reduce the energy consumption of *dynamic random access memory* (DRAM) is ineffective. Recently, combining DRAM and *phase change memory* (PCM) to construct a hybrid main memory is recognized as a promising solution to distinctly reduce the energy consumption. In this paper, we propose a new page-level energy management strategy to optimize the energy consumption of the hybrid main memory. The proposed strategy records pages' local and global access information by a new data structure, and then classifies pages by the access history, at last adaptively places PCM or DRAM pages according to the memory characteristics and remaps the migrated pages. Our experimental results show that our strategy can achieve 9.4 % of energy saving and 9.6 % of performance improvement at most compared with APG and PDRAM, which were proposed respectively by conferences RACS'12 and DAC'09.

Keywords Energy consumption · Hybrid memory · PCM

✉ Hai Jin
hjin@mail.hust.edu.cn; hjin@hust.edu.cn

Xiaofei Liao
xfliao@hust.edu.cn

[1] Cluster and Grid Computing Lab, Services Computing Technology and System Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

1 Introduction

A series of complex and data-intensive applications have emerged since the big data era arrives. These applications bring an irreversible escalation of memory demands. The capacity and speed of memory need to escalate. Traditional *dynamic random access memory* (DRAM) has a good access performance, but it is limited to its high refresh power. Energy consumption of main memory increases as the capacity of memory grows, which has become a critical issue in modern computer system.

Fortunately, the emergence of a new storage hardware, PCM [1,2], provides a way to reduce the energy consumption of memory. PCM is a nonvolatile memory without refresh power. Consequently, a hybrid memory which combines DRAM and PCM is recognized as a promising solution to distinctly reduce the energy consumption [3]. However stacking the two materials simply cannot enable their advantages effectively since traditional memory controller does not distribute the request according to the characteristics of each memory.

PCM has a lower read operation energy consumption compared with DRAM, while its access speed is slower, and the power of write operation is higher than DRAM [4]. Moreover, when a page in PCM is written frequently, it will consume a very high power leading to a loss of its native low power consumption characteristic [5]. In this paper, we propose a strategy to place pages according to the access information. The strategy intends to reduce the energy of memory through migrating pages from PCM to DRAM.

To achieve this goal, we introduce a new data structure which records the local and global access information in page level. Then we classify the pages into hot pages and cold pages [1] according to the frequency of write operation: the pages which are accessed infrequently in DRAM are cold pages; the pages which are written frequently in PCM are hot pages. Moreover, we build a mathematical model for energy consumption, by which the saving energy can be estimated. Then pages saving the most energy in PCM (the hottest pages) will be migrated to DRAM.

The contributions in this paper are as follows: first, we propose a method that classifies pages according to the local and global access information; second, we build an energy model to estimate the power consumption of pages; finally, we evaluate our strategy and the results show that it achieves a better performance.

The rest of paper is organized as follows. Section 2 shows an overview of the related work about hybrid memory. In Sect. 3, we describe the hybrid memory architecture and our mechanism. Comprehensive evaluations are gave in Sect. 4. Section 5 concludes our work.

2 Related Work

Since PCM appeared and Dhiman et al. [3] proposed the PCM and DRAM hybrid memory architecture, several strategies have been proposed to manage this hybrid main memory by researchers. In this architecture researchers intended to exploit the benefits of DRAM and PCM. Some researches focus on improving the system performance [4, 6] and enhancing the lifetime of PCM [7–9]. Other works aim to reduce the energy consumption of hybrid main memory [5,6].

Dhiman et al. [3] proposed a parallel architecture which combined DRAM and PCM, namely PDRAM. In the paper, challenges in hybrid main memory have been evaluated by the authors, and they proposed low overhead policies to manage this hybrid main memory system and improve the system performance. The paper also designed a wear-leveling scheme and page swapper to enhance the lifetime of PCM. The work mainly enhanced the lifetime of PCM and simply switched write-heavy pages and free pages. However, the improvement of performance was limited.

A new mechanism of hybrid memory management, *Adaptive Page Grouping* (APG), was introduced by Shin et al. [10]. They suggested that the mechanism for recording access information of pages should not add other additional hardware. APG extended the TLB entry and recorded the access information in these extra spaces. They calculated the access properties through access information. The method managed hybrid memory pages in groups which was classified by similar access properties. Although the strategy can effectively avoid Ping-Pong migrations [10], it lost the flexibility.

Rank-based Page Placement (RaPP) [11] was another hybrid memory management mechanism. Ramos et al. introduced a multi-level queue to store the access information. The record of each page will be set a survival time, during the time all access information will be recorded. The record in a higher level of queue is the pages accessed more frequently. Although the mechanism provided a flexibly migration strategy but it did not taking global access information and energy into consideration.

So far, existing works have not provided an efficient policy to distinguish hot and cold pages according to local and global access information at the same time. APG and RaPP only considered local information and other mechanisms considered less information. Our work aims to make up for this deficiency. We propose a method that can distinguish pages more reasonably, and it can also reduce the energy consumption of main memory effectively.

3 System Design

Figure 1 depicts the hybrid memory architecture [3], and we build our system based on that architecture. In our system, PCM and DRAM are parallel structure and directly connected to the memory controller. The address of hybrid memory is linear encoding, of which DRAM is in low end and PCM is in high end.

However, stacking two kinds of memories simply cannot play their respective advantages. We propose an optimal page-level power strategy which manages pages in memory controller shown in Fig. 1. It includes four modules: page access recording, page classifying, energy profiling, page migration and remapping. The page access recording and classifying modules record page access information and classify hot pages and cold pages by their access history. The energy profiling module calculates the saving energy of hot pages and selects the most appropriate pages to be migrated. At last, we do the migration and remap the migrated pages, thus the requests come after the migration can access memory correctly.

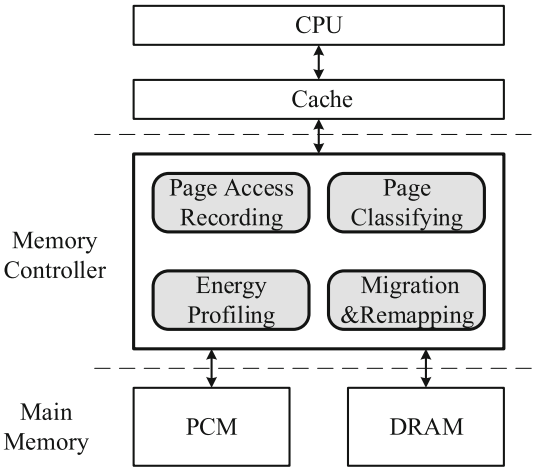

Fig. 1 Memory controller modules

3.1 Page Access Recording and Classifying

It is a key point to place pages into appropriate kind of memory according to the access characteristics. To make it simple, pages which are written frequently should be placed into DRAM, while pages accessed infrequently should be placed in PCM. Other pages that are not hot pages nor cold pages will not be moved. To classify the hot pages and cold pages we record the access information of memory in page level using a data structure shown in Fig. 2, including a *Page Record* and a *Record Queue*. In the *page record*, *TW* and *TR* record the total number of write and read when the page is accessed for the first time. *RW* and *RR* record the write and read access information in a period of time called $valid\ lifetime$. The nearest 32 accesses are recorded by *RU* field. In addition, the $flag$ records whether the page is selected as a hot page or not last time.

In order to distinguish cold and hot pages accurately, we need an effective sorting mechanism according to local and global access information. The global access information refers to the write and read frequency of pages when they are accessed from the first time until the current time, and the local access information refers to the

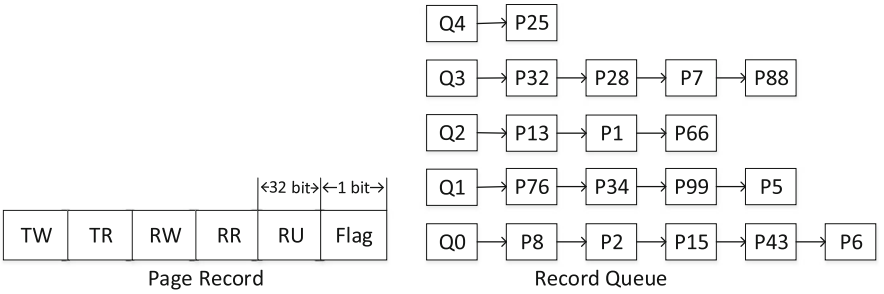

Fig. 2 Record struct and record queue

write and read frequency of pages in a valid lifetime. In our method, we define three policies: (1) *TW* has a higher weight; (2) *RW* should reach a threshold value; (3) the number of 1 value bit in *RU* is a decisive factor.

In this paper, we introduce a multi-level queue (on the right of Fig. 2). First, the mechanism sorts the record queue according to the global access information. When the system is running, the value of record changes, then it will be moved to a higher queue if the *TW* reaches the threshold ($\alpha = 2^{i+1}$, i indicates the number of queue). As shown in Fig. 2, the page record will be placed in number 0 queue (Q0) at the first accessed, and we set the survival time t to a certain number. *RW* and *RR* record the write and read frequency until t change to 0, then t, *RW* and *RR* are reset. When the page is accessed, the fields of record will change. We manage the record in this way that the page having a high total write number is more likely to be a hot page. Moreover, *RU* records the recent 32 access information, which considers the locality of program. When the system starts, the *RU* field of page is set to 0. Once the page is written, we shift the *RU* one bit to the left and set the last bit to 1.

Algorithm 1 Algorithm for picking hot pages

```
Require: Queue PCM :Record queue in PCM
Ensure: hot pages
  function PICK HOT PAGES FROM PCM(Queue pcm)
    i ← Q_size
    for i = Q_size → 1 do
      if pcm[i].next != NULL then
        find ← pcm[i].next
        while find! = NULL do
          TW ← find.totWrite
          RW ← find.recentWrite/(RR + RW)
          RU ← GETVALUE(find.RU)
          if TW > α and RW > β then
            result ← find.page
          else
            if RU > μ then
              result ← find.page
            end if
          end if
          find ← find.next
        end while
      end if
    end for
    return result
  end function
```

At last, the probability to be a hot page reduces from the highest queue to the lowest. At the same time, we record the max *RR* and *RW* of each queue in the head record. We traverse the queue from highest one and calculate the value of $RW/(RR + RW)$ in the record and we denote it by k. If k is less than the threshold β, we ignore this record and calculate the next one, because it is unlikely to be a hot page. Otherwise, we test the *RU* field. If the number of 1 value bit in *RU* field is greater than the threshold μ, it is a hot page. If not, we calculate the last 16 bits of *RU*, and the number of 1 value

bit will decide the page whether it is a hot page or not. The procedure of algorithm is shown in algorithm 1. The combination of α, β and μ can control the number of page migrations, and they are relatively independent. Therefore these thresholds can be determined by controlling variables method. First, we fix the values of α and μ to test the appropriate value of β. Then test the value of μ in the similar way. The typical values of β and μ in our measurements are 0.76 and 1734. These values leverage the access time and energy consumption.

3.2 Energy Profiling Model

In this section, we establish an energy model for hybrid memory. We build an energy consumption model for pages in order to select pages more intuitively. The power of one page can be estimated according to its access history. To make our model straightforward, we make the following definitions:

- E_{pi} indicates the power of page i in PCM and E_{dj} denotes the power of page j in DRAM.
- E_{pr} indicates the power of reading PCM page once and E_{pw} denotes the power of writing PCM page once.
- E_{dr} indicates the power of reading DRAM page once and E_{dw} denotes the power of writing DRAM page once.
- The number of read operation in the page i is denoted as N_{ri} and write operation as N_{wi}.

We build the model that: the energy of page i in PCM is:

$$E_{pi} = E_{pr} \cdot N_{ri} + E_{pw} \cdot N_{wi}$$

The energy of page j in DRAM is:

$$E_{dj} = E_{dr} \cdot N_{rj} + E_{dw} \cdot N_{wj}$$

Besides, if the PCM has n pages and DRAM has m pages we get the total energy of PCM E_P:

$$E_p = \sum_{i=1}^{n} E_{pi} = \sum_{i=1}^{n} E_{pr} \cdot N_{ri} + \sum_{i=1}^{n} E_{pw} \cdot N_{wi}$$

Similarly, the total energy of DRAM E_d is:

$$E_d = \sum_{j=1}^{m} E_{dj} = \sum_{j=1}^{m} E_{dr} \cdot N_{rj} + \sum_{j=1}^{m} E_{dw} \cdot N_{wj}$$

In Sect. 3.1, we have recorded the access information of pages to calculate the energy consumption through the formula. The migration energy can be calculated as the difference before and after the page migrated, which is E minus E' where E and E' denote the energy consumption before and after page migration respectively. For example,

the page in PCM is migrated to DRAM. According to the principle of locality [12], the page access information is similar to the previous survival time. The energy of PCM page after migration is

$$E' = E_{dr} \cdot N_{ri} + E_{dw} \cdot N_{wi}$$

The power before migration is

$$E = E_{pr} \cdot N_{rj} + E_{pw} \cdot N_{wj}$$

The migration energy equals to the energy before migration minus the energy after migration. We get the formula for calculating the saving energy

$$\begin{aligned}
\Delta E_{ij} &= \Delta E_{pi} + \Delta E_{dj} \\
\Delta E_{ij} &= E_{pi} - E'_{pi} + E_{dj} - E'_{dj} \\
&= \left(E_{pr} - E_{dr}\right)\left(N_{ri} - N_{rj}\right) + \left(E_{pw} - E_{dw}\right)\left(N_{wi} - N_{wj}\right) \\
&= \Delta E_r \cdot \Delta N_r + \Delta E_w \cdot \Delta N_w
\end{aligned}$$

This formula can estimate the saving energy for a page after it is migrated. At last, we migrate k pages from PCM to DRAM, and get the total energy of main memory E as follow:

$$\begin{aligned}
E &= E_p + E_d - \sum_{i,j}^{k} \Delta E_{ij} \\
&= \sum_{i=1}^{n} E_{pi} + \sum_{j=1}^{m} E_{dj} - \sum_{i,j}^{k} \Delta E_{ij}
\end{aligned}$$

This formula shows that to reduce the energy consumption of memory it is necessary to improve the migration energy. In the system, we make full use of the access history and calculate the saving energy of every page. Then we pick the pages that earn the most energy saving of memory.

3.3 Page Migration and Remapping

When performing page migration, we just pick up the pages to be migrated. Fortunately, we have got these pages in Sect. 3.2. We just pick the pages from the front of the energy queue and migrate those pages until the queue is empty. In order to complete the migration, It takes three steps: (1) after picking one element from the queue, we build the PCM and DRAM read packets with migration flag; (2) send these read packets to the memory entity; (3) swap data of two read migration responses and send the write migration packets if we receive the read migration responses.

We remap the migrated pages to ensure the system access those pages correctly. At beginning of the migration operation, we build a map containing a source page,

Table 1 Power characteristics

Parameter	DRAM (mW)	PRAM (mW)
Row read power	210	78
Row write power	195	773
Active power	75	25
Standby power	90	45
Refresh power	4	0

a destination page and a migration $flag$. We change the $flag$ as the migration is running. After that, if a packet wants to access the address which has been remapped, we change the address to the mapped destination address. When the access completes, the address is reset to the primary value. If the address has not been remapped, the packet is sent to the memory directly.

4 Evaluation

In this section we evaluate the memory access time and energy consumption of our mechanism for the hybrid memory. We first introduce our experimental setup and configurations, then analyze the results of experiments.

4.1 Experimental Setup

In order to verify our energy management strategy, we build our system based on a full system simulator—gem5 [13]. Gem5 includes DRAM models but does not provide PCM models. We adopt nvmain simulator [14] as a memory model combining with gem5. Nvmain has a detailed PCM memory structure, and it also contains energy and timing models.

We extend the structure of gem5 simulator to make it support hybrid main memory. We build the modified hybrid memory structure in gem5 simulator. The DRAM module is built on gem5, and PCM module is introduced from nvmain. The delay and power parameters of DRAM and PCM used in the experiments are shown in the Tables 1 and 2 [3].

4.2 Experimental Configurations and Benchmark

The system configurations of gem5 simulator are as follows: we set CPU with the *timing* type, that is gem5 uses the Timing Simple CPU, and the simulator then runs on time sequence mode. Two levels of cache are applied—a 64KB first level and a 2MB second level. The first level cache contains a four-way set-associative session level instruction cache and a 32KB four-way set-associative session level data cache. The size of cache line in second level cache is 64B. The hybrid memory combines PCM with DRAM, and the size of PCM is four times bigger than that of DRAM [10,15].

Table 2 Timing characteristics

Parameter	DRAM (ns)	PRAM (ns)
Initial row read	15	28
Same row read	15	15
Initial row write	22	150
Same row write	15	15

Table 3 Read and write percentage of DRAM and PCM

Benchmark	Read (DRAM) (%)	Write (DRAM) (%)	Read (PCM) (%)	Write (PCM) (%)
Sjeng	13.8	13.6	36.4	36.2
Calculix	24.1	7.9	55.4	12.6
CactusADM	30.5	13.3	42.9	13.3
Gobmk	6.5	6.4	45.3	41.8
Bzip2	11.2	10.9	48.9	29.0
Leslie3d	13.1	12.9	41.4	32.6

We select six typical frequent memory access programs from SPEC CPU 2006 to evaluate the performance of our system. Because not all of the programs occupy the same size of memory, the simulator should be set with different size of memory for each program. Consequently, we make access requests distributed evenly between PCM and DRAM in hybrid main memory.

4.3 Results and Analysis

In this section, we analyze the performance and energy consumption of our system. To make it simple we name our mechanism EMH—energy management for hybrid memory. We implement the mechanism PDRAM, APG and our EMH in gem5. Table 3 shows the benchmark characteristics. it includes write and read access proportion respectively in DRAM and PCM for all the benchmarks. Figure 3 shows the hit rates of DRAM and PCM in hybrid memory across all the benchmarks.

4.3.1 Memory Access Time

Figure 4 illustrates the memory access time of PDRAM, APG and EMH. The data show that EMH achieves a maximum 9.6 % reduction of memory access time compared with PDRAM, and an also 9.3 % reduction compared with APG. On average, EMH can obtain 5.2 % performance improvement compared with PDRAM, and 2.5 % compared with APG across all the benchmarks.

Specially, in Fig. 4, APG has a better memory access time performance over EMH on *gobmk*, that is because *gobmk* has a higher write proportion and hit rate (see in Table 3 and Fig. 3). The more we migrate pages in PCM, the better performance the system

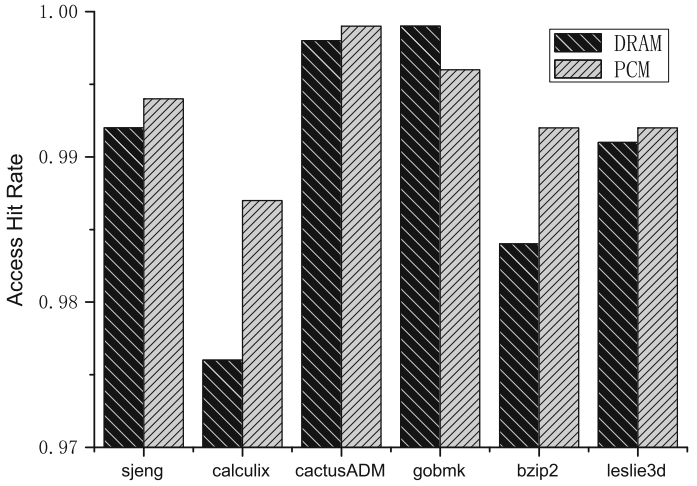

Fig. 3 The hit rate of hybrid memory

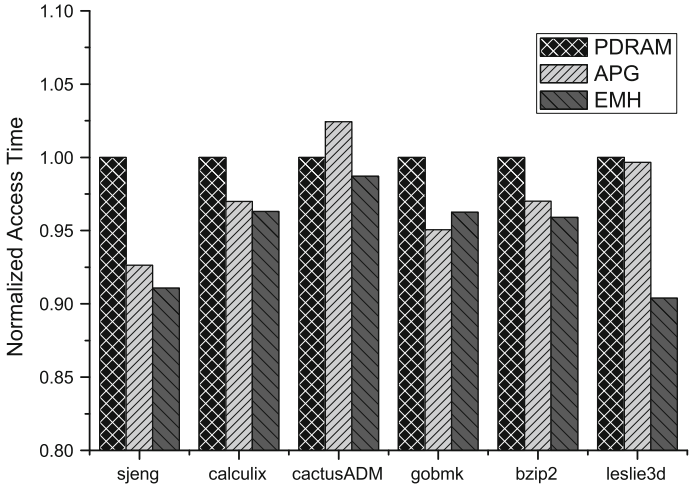

Fig. 4 The normalized memory access time of hybrid memory

get. For *cactusADM*, APG has an extraordinarily high access time. As migrating page by group, APG will spend time to migrate pages which are unnecessary to be migrated, and this defect behaves even worse when the hit rate is high (see in Fig. 3). Therefore its access time is higher than PDRAM and EMH on *cactusADM*.

4.3.2 Energy Consumption

The energy consumptions of main memory for all benchmarks are shown in Fig. 5, including standby power, refresh power, active power, read power and write power. It is obviously that EMH achieves an effective energy saving compared with PDRAM

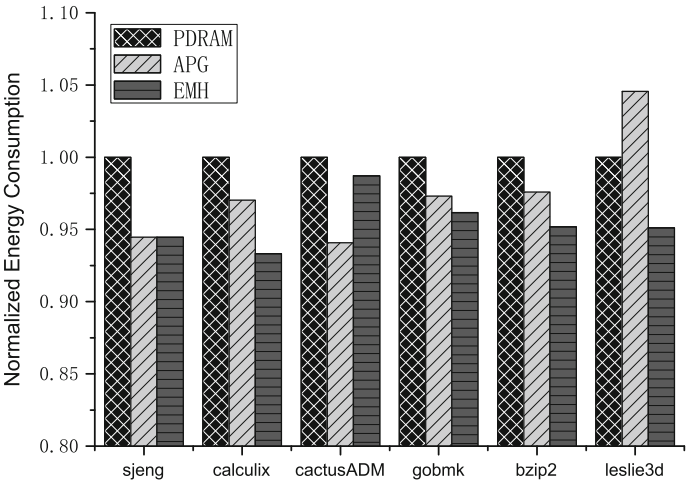

Fig. 5 The normalized energy consumption of hybrid memory

and APG, the maximum 6.7 and 9.4 % respectively. On average, our method reduces about 4.5 % energy consumption compared with PDRAM and 2.0 % compared with APG across all the benchmarks.

More specially, APG has the highest energy consumption on *leslie3d*. For *leslie3d*, it behaves a relatively higher write percentage and even larger read access (see in Table 3). Moreover, *leslie3d* has a good access hit rate. Since APG migrates the pages by group it will migrate read frequency pages to DRAM, and from Table 1 we know that the energy of read operation in PCM is higher than in DRAM. In addition, for *cactusADM*, APG has a lower energy consumption than EMH. That is because *cactusADM* has a relative low PCM write operation percentage but a high DRAM read operation proportion, and these access behaviors match the nature of APG mechanism. EMH considers not only the local access information but also the global's, which seems overly strict in this case and leads to a performance degradation. While APG takes into account more local information to migrate more proper pages then gets a better energy saving.

In this section, we compare the memory access performance and energy consumption among the three mechanism. As the results shown, EMH performs better on both of the two metrics. EMH also demonstrates its advantages of low write power in DRAM and low read power in PCM, as well as low write latency.

4.3.3 Space Overhead

In this section, we discuss the space overhead of our mechanism for hybrid memory. First, we analyze the space overhead of access history record structure. One record structure takes up 28 bytes. That means n pages need $28n$ bytes. If our system divides pages of memory by 4096 bytes, the size of our memory is $4096n$ bytes in total. The migration structure takes 16 bytes. Besides, the remap structure takes $16k$ bytes, where k expresses the number of pages migrated. Then the total space is $28n + 32k$ bytes. The

proportion of space overhead in the memory size is $(28n+32k)/(4096n)$. Because the k is far smaller than n, then the proportion is $28/4096 = 0.68\,\%$. The overhead is $<1\,\%$.

5 Conclusions

In this paper, we propose a power management strategy for PCM and DRAM hybrid memory to reduce energy consumption and improve system performance. We record the local and global access information of hybrid main memory and use a multi-level queue to manage these records. Then we select hot pages and cold pages according to the local and global access information to calculate the energy saving. Our strategy also provides a model for selecting the migration pages. At last we migrate the hot pages from PCM to DRAM and remap the pages that have been migrated.

We implement our strategy in gem5 simulator. We evaluate our strategy, APG and PDRAM. The results show that our strategy reduces the energy consumption more efficiently and achieves better performance.

Acknowledgements This paper is supported by China National Natural Science Foundation under Grants Nos. 61322210, 61379135, Doctoral Fund of Ministry of Education of China under Grant No. 20130142110048, National High-tech Research and Development Program of China (863 Program) under Grant No. 2015AA015303.

References

1. Zhou, P., Zhao, B., Yang, J., Zhang, Y.: A durable and energy efficient main memory using phase change memory technology. In: Proceedings of the 46th Annual Design Automation Conference, pp. 14–23. ACM (2009)
2. Chung, L.: Phase change memory. Sci. China (Inf. Sci.) **2011**(5), 1061–1072 (2011)
3. Dhiman, G., Ayoub, R., Rosing, T.: PDRAM: a hybrid PRAM and DRAM main memory system. In: Proceedings of the 46th Annual Design Automation Conference, pp. 664–469. ACM (2009)
4. Qureshi, M.K., Srinivasan, V., Rivers, J.A.: Scalable high performance main memory system using phase-change memory technology. In: Proceedings of the 36th Annual International Symposium on Computer Architecture, pp. 24–33. ACM (2009)
5. Mangalagiri, P., Sarpatwari, K., Yanamandra, A., Narayanan, V., Xie, Y., Irwin, M.J. Karim, O.A.: A low-power phase change memory based hybrid cache architecture. In: Proceedings of the 18th ACM Great Lakes Symposium on VLSI, pp. 395–398. ACM (2008)
6. Park, H., Yoo, S., Lee, S.: Power management of hybrid DRAM/PRAM-based main memory. In: Proceedings of the 48th Design Automation Conference, pp. 59–64. ACM (2011)
7. Baek, S., Lee, H.G., Nicopoulos, C., Kim, J.: A dual-phase compression mechanism for hybrid DRAM/PCM main memory architectures. In: Proceedings of the Great Lakes Symposium on VLSI, pp. 345–350. ACM (2012)
8. Qureshi, M.K., Karidis, J., Franceschini, M. Srinivasan, V. Lastras, L. Abali, B.: Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, pp. 14–23. ACM (2009)
9. Ferreira, A.P., Zhou, M., Bock, S.: Childers, B., Melhem, R., Mossé, D.: Increasing PCM main memory lifetime. In: Proceedings of the Conference on Design, Automation and Test in Europe, DATE '10, pp. 914–919. European Design and Automation Association (2010)
10. Shin, D.-J., Park, S.K., Kim, S.M., Park, K.H.: Adaptive page grouping for energy efficiency in hybrid PRAM–DRAM main memory. In: Proceedings of the 2012 ACM Research in Applied Computation Symposium, pp. 395–402. ACM (2012)
11. Ramos, L.E., Gorbatov, E., Bianchini, R.: Page placement in hybrid memory systems. In: Proceedings of the International Conference on Supercomputing, pp. 85–95. ACM (2011)

12. Lee, B.C., Ipek, E., Mutlu, O., Burger, D.: Phase change memory architecture and the quest for scalability. Commun. ACM **53**(7), 99–106 (2010)
13. Binkert, N., Dreslinski, R., Hsu, L., Lim, K., Saidi, A., Reinhardt, S.: The M5 simulator: modeling networked systems. IEEE Micro **26**(4), 52–60 (2006)
14. Poremba, M., Xie, Y.: Nvmain: an architectural-level main memory simulator for emerging non-volatile memories. In: Proceedings of the 2012 IEEE Annual Symposium on VLSI, pp. 392–397. IEEE (2012)
15. Lee, S., Bahn, H., Noh, S.: Clock-dwf: a write-history-aware page replacement algorithm for hybrid PCM and DRAM memory architectures. IEEE Trans. Comput. **63**(9), 2187–2200 (2014)