Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Energy-aware page replacement and consistency guarantee for hybrid NVM–DRAM memory systems[☆]

Jinyu Zhan[*,a], Yiming Zhang[a], Wei Jiang[a], Junhuan Yang[a], Lin Li[b], Yixin Li[a]

[a] School of Information and Software, University of Electronic Science and Technology of China, China
[b] School of Computer Science and Engineering, University of Electronic Science and Technology of China, China

ARTICLE INFO

Keywords:
NVM
Page replacement
Energy consumption
Consistency guarantee

ABSTRACT

Non-Volatile Memory (NVM) has been widely used in hybrid memory architecture due to the promising advantages like high density and low power consumption. Most of existing page replacement methods focused on prolonging the life time of NVM memory systems, which either consider the energy cost or deal with consistency errors caused by system failures. This paper proposes a page replacement method based on NVM–DRAM hybrid main memory systems for low power and consistency guarantee, called Energy-Aware Page Replacement (EAPR). The energy consumption of page access in DRAM and NVM is modelled considering the memory access like reading, writing and idle. EAPR utilizes page grouping policy to improve the migration efficiency between NVM and DRAM. Page groups are chosen to migrate according to their energy-related substitution degrees. To guarantee the consistency during page migration, this paper designs two consistency guarantee schemes in page replacement phase and transaction committing phase. Instead of deleting logs directly, our approach guarantees the consistency of hybrid memory systems by modifying the data structures of logs after the transactions of applications are submitted. Based on existing benchmarks, we conduct extensive experiments to evaluate the proposed method. Experimental results show that EAPR can reduce the energy consumption up to 40.19% compared with the existing page replacement algorithms, and also guarantee the consistency of transactions in NVM–DRAM hybrid memory systems.

1. Introduction

In the past few decades, Dynamic Random Access Memory (DRAM) has been used as the main memory steadily due to its high performance and other excellent features. However, it has suffered some big problems such as high refresh power consumption and low capacity. High refresh power consumption leads the power usage of DRAM to be very high even when system is mostly idle. Specifically, refresh power consumption almost accounts up to 40% of the total system consumption in mobile devices [2]. In addition, low capacity of DRAM has been unable to meet the requirements of big data applications. Therefore, Non-Volatile Memory (NVM), with attractive characteristics of large capacity and low energy consumption, has obtained much attention from academic community. However, NVM also has certain disadvantages in writing operations [3–5]. For instance, the writing latency of NVM is 5 to 10 times higher than that of DRAM. Energy consumption of writing operation on NVM is much higher than DRAM. Besides, the writing times of NVM is limited.

There are mainly three kinds of design to apply NVM to the current computer architecture. The first one is to replace DRAM directly with NVM [6–8]. This method does not need to change the current computer architecture, which can minimize the energy consumption of memory systems because NVM's standby energy consumption is much lower than DRAM's. However, due to low writing speed and low lifetime of NVM, such architecture is a bit impractical, although it can use wear-leveling method to prolong the lifetime of NVM. The second method is to use NVM–DRAM hybrid memory to replace the traditional DRAM [9–11], where DRAM or NVM plays the role of buffering. The advantages of this architecture are that it can significantly improve the performance of the system and reduce the system energy consumption. But it is difficult to use NVM as the buffer, because it will put a great deal of writing pressure on the NVM and will eventually result in a very short lifetime of NVM. The third design is to combine NVM with DRAM as a hybrid main memory architecture [12–16], in which NVM and DRAM are both main memory. This architecture is generally the best choice, because it has no lasting writing pressure on NVM and can

☆ This paper is an extended version of a conference paper [1].
* Corresponding author.
E-mail address: zhanjy@uestc.edu.cn (J. Zhan).

https://doi.org/10.1016/j.sysarc.2018.07.004
Received 7 February 2018; Received in revised form 5 June 2018; Accepted 25 July 2018
Available online 27 July 2018

maximize the benefits of NVM. However, it also needs a main memory management to utilize the advantages of both DRAM and NVM. Researches on hybrid main memorys have predominantly pursued software-oblivious memory management executed by hardware [17] or involving OS [18]. Traditional file systems are not suitable for NVM-based hybrid memory systems since they cannot guarantee the consistency of hybrid memory systems.

In this paper, we tend to place NVM on the processor memory bus alongside the conventional DRAM to establish a hybrid volatile/non-volatile main memory system. We also focus on the memory management design. As a supplement to existing researches, we propose a Energy-Aware Page Replacement (EAPR) algorithm based on NVM–DRAM hybrid main memory systems for low power and consistency guarantees, which can reduce the energy consumptions of the NVM–DRAM hybrid systems and satisfy high consistency guarantees in both kernel and user mode. The primary contributions of this paper are listed as follows.

- Analyzing the features of energy consumption during migrating the pages between DRAM and NVM.
- Proposing an energy-aware page replacement algorithm for hybrid main memory architecture.
- Designing consistency guarantee schemes in page replacement and transaction committing phases to prevent system failures.
- Evaluating the effectiveness of the proposed techniques by extensive benchmark-based experiments.

The rest of the paper is organized as follows. Section 2 presents the system architecture and model for the page replacement. Section 3 presents the design of consistency improvement. Section 4 proposes our page replacement algorithm with consistency guarantees. Section 5 gives the experiments to validate the proposed design. In Section 6, mostly related works are reviewed. Finally, we conclude this paper in Section 7.

2. System model for page replacement

In this section, we first demonstrate the system architecture. We then present the energy model for page replacement. To improve the efficiency, we also present a page grouping method in this section. For further reference, most of the notations used in the paper are summarized in Table 1.

2.1. System architecture

Fig. 1 shows the target memory architecture of this paper. The system architecture has a hybrid main memory coloured with a blue box in Fig. 1, which is composed of two kinds of memory units, i.e., NVM and DRAM. CPU accesses the hybrid memory system via a memory controller. To improve the efficiency of the management of hybrid memory systems, in this paper we implement EAPR, a software component, between CPU and the hybrid memory system. In EAPR, we design two mechanisms (page replacement and consistency guarantee mechanism) to save energy and provide consistency guarantee, which are indicated in Fig. 1 with a red frame. First we focus on modeling the page energy consumption which can be calculated by the memory access, such as reading, writing and idle. To further improve the efficiency, we manage similar pages with continuous addresses as a page group, and we use a page grouping policy to migrate pages from DRAM to NVM and vice versa. For the consistency guarantee, the logs have to be stored in NVM because we need to recover the writing transactions according to these logs. We design EAPR with the consistency improvement in the page replacement phase and the user transaction committing phase, which can guarantee the consistency of writing transactions even if there exist page replacements.

Table 1
Notation table.

Symbol	Definition
P_i	A page in hybrid memory system
C_{NVM}	Energy cost of page P_i in NVM
C_{DRAM}	Energy cost of page P_i in DRAM
E_{NVM}^{read}	Reading energy consumption per page in NVM
E_{NVM}^{write}	Writing energy consumption per page in NVM
POW_{NVM}^{idle}	Idle power consumption for the pages in NVM
r_i	The number of readings for P_i in hybrid memory system
w_i	The number of writings for P_i in hybrid memory system
T	The time period for page replacement
E_{DRAM}^{read}	Reading energy consumption per page in DRAM
E_{DRAM}^{write}	Writing energy consumption per page in DRAM
POW_{DRAM}^{idle}	Idle power consumption for pages in DRAM
$C_{NVM--DRAM}$	Energy cost when P_i is replaced from NVM to DRAM
E_{extra}	Extra energy consumption during the migration of P_i
ΔE_{read}	$E_{NVM}^{read} - E_{DRAM}^{read}$
ΔE_{write}	$E_{NVM}^{write} - E_{DRAM}^{write}$
G_i	The ith page group
S_i^{group}	Substitution degree for page group G_i
V_i	Total size of page group G_i
N	Size of spare space in DRAM
S_i^*	Improved substitution degree for P_i in DRAM
R_i^N	Connectivity of page P_i in the transaction
λ	Coefficient of R_i^N

2.2. Energy model of page replacement

In this paper, we consider the energy cost when select pages to replace from NVM to DRAM. Energy consumption is mainly caused by reading and writing operations of hybrid memory and the idle duration.

Definition 1. Given page P_i in NVM, C_{NVM} is defined to denote the energy cost during the time period of T.

$$C_{NVM} = E_{NVM}^{read} * r_i + E_{NVM}^{write} * w_i + POW_{NVM}^{idle} * T \tag{1}$$

where E_{NVM}^{read} is the reading energy consumption of each page in NVM, E_{NVM}^{write} is the writing energy consumption of each page in NVM, r_i and w_i are the numbers of reading and writing P_i in NVM respectively. POW_{NVM}^{idle} is the idle power for each NVM page. In particular, POW_{NVM}^{idle} can be ignored because the idle energy consumption of NVM is much lower than that of DRAM [19,20].

Definition 2. Given page P_i in DRAM, C_{DRAM} is defined to denote the energy cost during the time period of T:

$$C_{DRAM} = E_{DRAM}^{read} * r_i + E_{DRAM}^{write} * w_i + POW_{DRAM}^{idle} * T \tag{2}$$

where E_{DRAM}^{read} is the reading energy consumption of each page in DRAM, E_{DRAM}^{write} is the writing energy consumption of each page in DRAM and POW_{DRAM}^{idle} is the idle power of each DRAM page.

Definition 3. $C_{NVM-DRAM}$, as the energy cost of P_i replaced from NVM to DRAM, is defined as:

$$\begin{aligned} C_{NVM-DRAM} \;=\; & E_{extra} + (E_{NVM}^{read} + E_{DRAM}^{write}) \\ & + (E_{DRAM}^{read} * r_i + E_{DRAM}^{write} * w_i \\ & + POW_{DRAM}^{idle} * T) + (E_{DRAM}^{read} \\ & + E_{NVM}^{write}) \end{aligned} \tag{3}$$

where E_{extra} is the extra energy consumption during migration process in addition to reading and writing consumptions. $(E_{NVM}^{read} + E_{DRAM}^{write})$ means the energy consumption of a page once migrating from NVM to DRAM. $(E_{DRAM}^{read} * r_i + E_{DRAM}^{write} * w_i + POW_{DRAM}^{idle} * T)$ represents the reading, writing as well as idle energy consumption in DRAM for a page, whereas $(E_{DRAM}^{read} + E_{NVM}^{write})$ is the energy consumption of being written back from DRAM to NVM. The energy consumption of being written

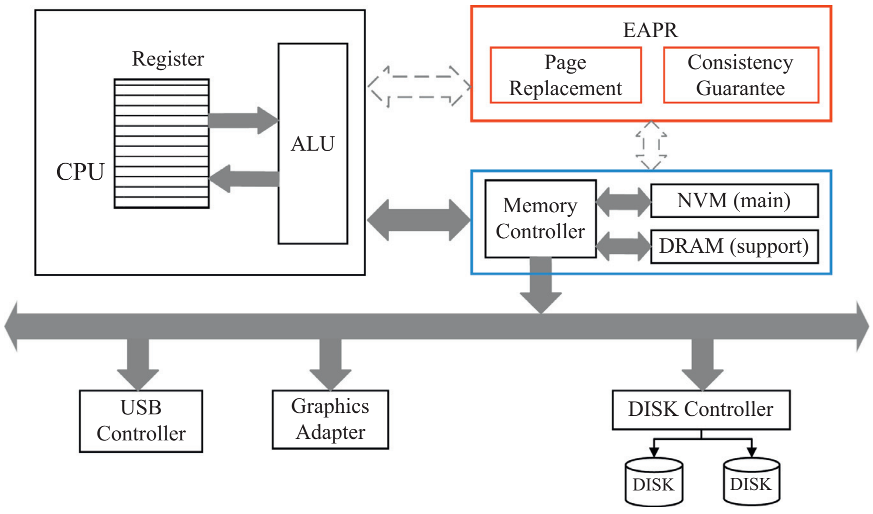


Fig. 1. System architecture.

back must be considered because the page will be replaced to NVM if it has been stored in DRAM. In particular, there are some extra energy consumptions in the page migration procedure, such as page wear energy consumption, cache energy consumption and so on. But these extra power consumptions are low enough to be ignored compared with the energy consumptions of page reading, writing and idle.

Lemma 1. *Page P_i is energy-effective enough to be replaced from NVM to DRAM if* $(C_{NVM} - C_{NVM-DRAM} > 0)$.

Based on Lemma 1, we can have the following inequalities:

$$\begin{aligned} C_{NVM} - C_{NVM-DRAM} \;>\; & 0 \Longrightarrow E^{read}_{NVM}*(r_i - 1) + E^{write}_{NVM}*(w_i - 1) \\ & - E^{write}_{DRAM}*(w_i + 1) - E^{read}_{DRAM}*(r_i + 1) \\ & + T*(POW^{idle}_{NVM} - POW^{idle}_{DRAM}) > 0 \end{aligned} \tag{4}$$

Since POW^{idle}_{NVM} is very low in the process, it can be ignored in In Eq. (4). Thus we have the following inequality.

$$\begin{aligned} & r_i*(E^{read}_{NVM} - E^{read}_{DRAM}) + w_i*(E^{write}_{NVM} \\ & - E^{write}_{DRAM}) - T*POW^{idle}_{DRAM} \\ & > E^{read}_{NVM} + E^{read}_{DRAM} + E^{write}_{NVM} + E^{write}_{DRAM} \end{aligned} \tag{5}$$

We define $(E^{read}_{NVM} - E^{read}_{DRAM})$ and $(E^{write}_{NVM} - E^{write}_{DRAM})$ as ΔE_{read} and ΔE_{write} respectively. Thus In Eq. (5) can be transformed as follows.

$$\begin{aligned} & r_i*\Delta E_{read} + w_i*\Delta E_{write} - T*POW^{idle}_{DRAM} \\ & > E^{read}_{NVM} + E^{read}_{DRAM} + E^{write}_{NVM} + E^{write}_{DRAM} \end{aligned} \tag{6}$$

The right side of In Eq. (6) is obviously constant. Hence we define the formula (the left side of In Eq. (6)) as the substitution degree of page P_i.

Definition 4. S_i, as the Substitution degree for P_i, is defined as follows.

$$S_i = r_i*\Delta E_{read} + w_i*\Delta E_{write} - T*POW^{idle}_{DRAM} \tag{7}$$

Lemma 2. *Page P_i is suitable to be moved to DRAM memory if* $S_i > E^{read}_{NVM} + E^{read}_{DRAM} + E^{write}_{NVM} + E^{write}_{DRAM}$.

Based on Lemma 1, In Eq. (6) and Definition 4, we can directly have the proof of Lemma 2.

Both ΔE_{read} and ΔE_{write} are constant and greater than 0 because the energy consumptions of reading and writing in NVM are both greater than DRAM[20,21]. In order to save energy, we can choose pages with higher substitution degrees (satisfying Lemma 2), and migrate them to DRAM. Otherwise the pages should be maintained in NVM memory if the page's substitution degrees can not satisfy Lemma 2.

2.3. Grouping model for page replacement

In hybrid main memory systems, single page migration should be avoided generally during the page replacement process because the replacement of a single page may cause different pages of one process to be stored in NVM and DRAM discretely and the addresses to be discontinuous, which will lead to lower speed and efficiency when reading or writing the pages of one process. Moreover, the pages will be swapped in and out very frequently if the basic unit of page replacement is a single page, which will also cause lower system performance. Therefore, we consider a page group as the basic unit of page replacement to avoid the frequent swapping and increase the efficiency of the system in this paper. We assume the pages of each group have continuous memory addresses and similar substitution degrees, which are assumed that the substitution degree error is ± 50% to the reference page in the group.

Fig. 2 is an example of the grouping model for page replacement. As shown in Fig. 2(a), when the system starts running after a time node, it is assumed that there are 5 processes and a total of 41 pages in NVM while no pages in DRAM at this time. The value of each unit represents the substitution degree of each page calculated by EAPR algorithm. At this point, according to the EAPR algorithm, some pages need to be replaced into DRAM. However, if page replacement is based on a single page, pages must be stored in DRAM in the descending order of their substitution degrees and pages of these 5 processes are particularly scattered, which will result in a significant decline in performance of reading or writing pages. Therefore, we have designed the page replacement based on page groups. Since those adjacent pages with similar substitution degrees of one process are considered as a page group, these 5 processes which have 41 pages are divided into 10 page groups. The EAPR algorithm performs the page replacement. As shown in Fig. 2(b), according to the selection of the simulated annealing algorithm, the page group 4, 5, 6, 8 and 9 are swapped into DRAM, which is completely occupied the free space of DRAM, and the sum of total substitution degrees of page groups in DRAM is the largest, satisfying the purpose of EAPR algorithm. Moreover, we can see that pages of each process are still continuous in each memory, which can improve the efficiency of subsequent reading or writing operations.

Definition 5. S^{group}_i, as the substitution degree of page group G_i, is defined as follows.

$$S^{group}_i = \sum_{P_i \in G_i} S_i \tag{8}$$

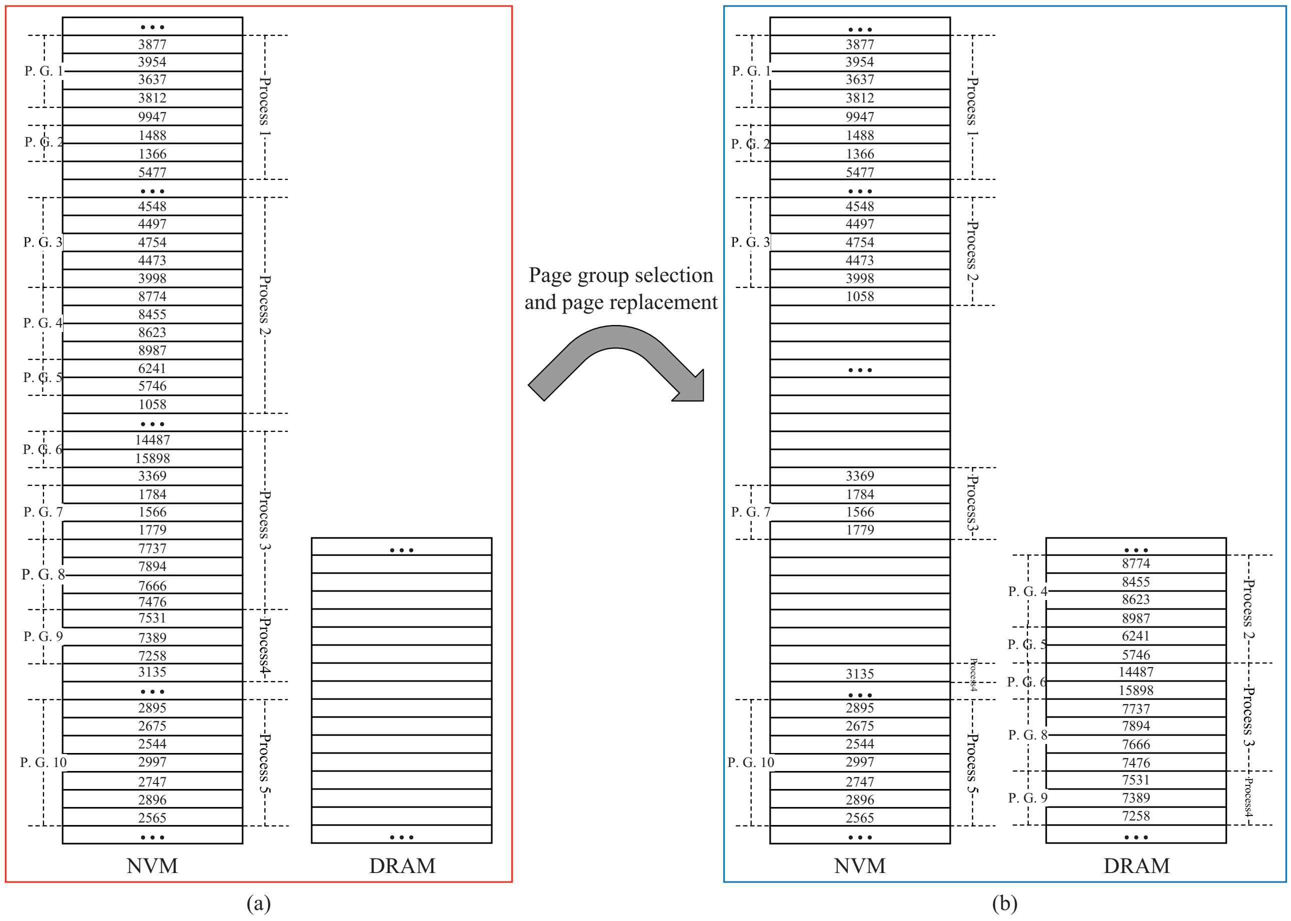


Fig. 2. An example of grouping model for page replacement.

In our NVM–DRAM main memory systems, NVM is the main memory and DRAM plays as the support memory. Therefore, NVM capacity is assumed to be large enough while the DRAM's is limited. That is to say, the size of DRAM which is used to store the replaced page group is limited in our hybrid main memory systems.

Because of the limited DRAM capacity, we have to migrate page groups which are most suitable from NVM to DRAM with higher substitution degrees. Page groups have different substitution degrees and group sizes. To obtain the optimal page selection is a linear programming problem, which is formulated as follows.

$$\begin{aligned} & Max \sum S_i^{group} \\ & S.\ T. \sum V_i \le N \end{aligned} \tag{9}$$

where V_i is the total size of page group G_i, and N is the size of spare DRAM which can be utilized to store pages moved from NVM.

To address the problem in Eq. (9) with optimal solution will result in high time complexity, which is not suitable during page replacement. In this paper, we design a greedy method to fast obtain page groups needed to be replaced. We use θ_i to denote the space efficiency of group G_i, which is defined as $\theta_i = S_i^{group}/V_i$. Page groups with higher space efficiency, are preferred to be selected. We just need to sort page groups by the order of θ_i and greedily replace the page group to DRAM with the highest efficiency value until DRAM is full.

3. Consistency improvement design

In this section, we first introduce the consistency method designed in EAPR. We then present the data structures and the improved page selection method designed for the realization of consistency guarantee.

3.1. Consistency method in EAPR

To guarantee the consistency of page replacement process, EAPR designs a writing-transaction committing mechanism between kernel mode and user mode. A writing transaction committing within NVM generates interrelated logs, and EAPR will be informed which pages have been changed. Any page exchanges between DRAM and NVM will prevent deleting logs until the pages are written back to NVM. Furthermore, EAPR will be informed which pages are log pages and the log pages can not be replaced to DRAM. In this way, user transactions can be recovered through logs even if the system failure occurs. For the recovery method, we can directly use the algorithm presented in paper of [22].

3.2. Data structures for consistence guarantee

To realize the consistency guarantee of hybrid memory systems, EAPR conducts three data structures in kernel mode, which are illustrated in Fig. 3.

- **TILL** (*Transaction Information Linked List*): In this linked list, each node corresponds to a writing transaction, which includes the pages written and replaced to DRAM. The nodes of this linked list are arrayed by the order of write transaction committing. EAPR can use this linked list to record the order of writing transactions and restore the hybrid memory system to a stable status if a system failure occurs.
- **PFHT** (*Page Frame Hash Table*): The keyword of hash table is the page frame number, which is the node position of writing transaction in the linked list. EAPR can find the writing transaction

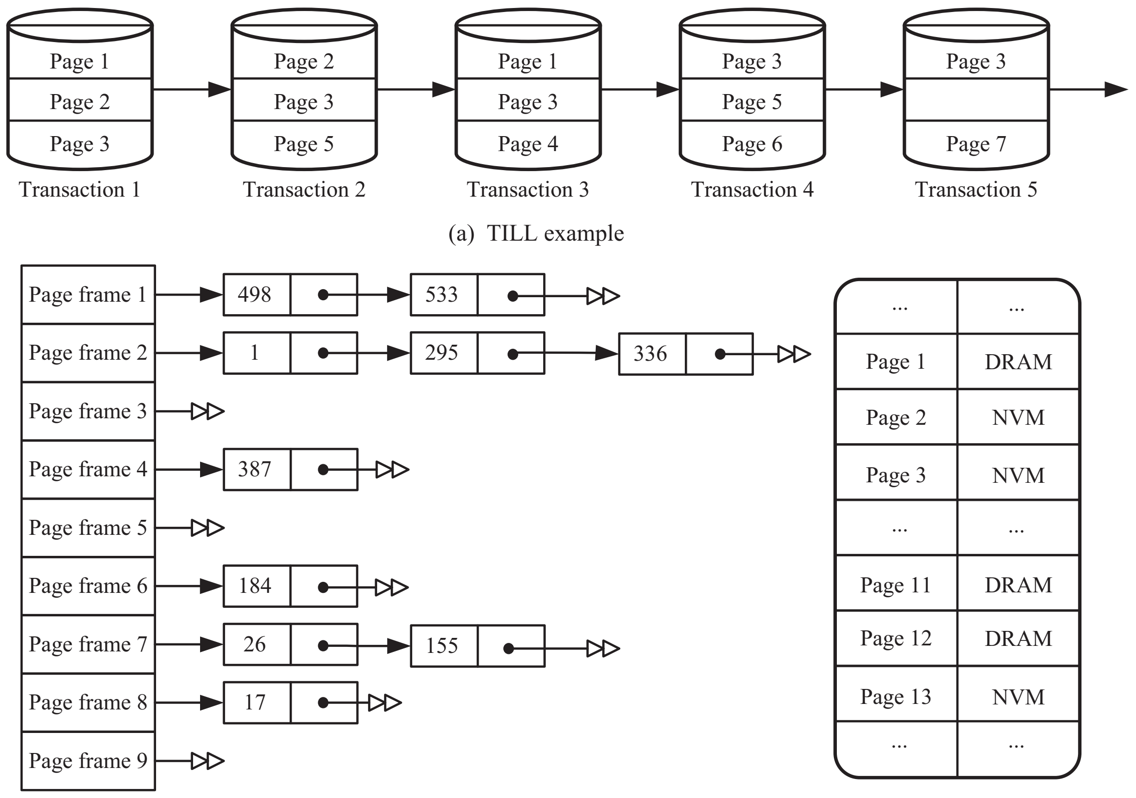


Fig. 3. Data structure for consistency improvement.

information of a page in the linked list through this hash table.

- **PLIT** (*Page Location Information Table*): PLIT is a simple table which mainly preserves the current location of each page, i.e., in NVM or DRAM.

3.3. Improved page selection with consistency

With the committing of user's writing transactions, the logs will become larger, leading to longer TILL. With long TILL, we have to take a lot of time to recover the system when failures occur. In order to prevent this situation, we modify the substitution degree to replace the most important pages from DRAM to NVM quickly.

In the TILL, each transaction concludes the information of many pages replaced to DRAM. We can construct a connected graph for these pages based on their relevancy. The connectivity of each page node in the graph is defined as R_i^N. As shown in Fig. 4, there are 5 transactions in the TILL. Each transaction has modified certain pages and we can create an undirected graph based on each page in a writing transaction. In the undirected graph, we can obtain that the degree of connectivity for page 1 is 4. Similarly, we can find the connectivity degrees of pages (2, 3, 4, 5, 6, 7) are (4, 9, 2, 4, 2, 7), respectively. The bigger R_i^N means that this page is the more important. Therefore, we choose to improve the substitution degree of each page by incorporating the connectivity related parameter.

Definition 6. S_i^*, as the improved substitution degree of P_i, is defined as follows.

$$S_i^* = r_i{*}\Delta E_{read} + w_i{*}\Delta E_{write} - T{*}POW_{DRAM}^{idle} - \lambda{*}R_i^N \tag{10}$$

where λ is the coefficient of R_i^N. λ influences the overall efficiency of hybrid memory systems. Large λ will result in migrating pages from DRAM to NVM quickly and lead to higher energy consumption. Although small λ obtains relative low energy consumption, the size of log files will be too large, which will reduce the operational efficiency of the hybrid memory systems. In this paper, we set $\lambda = \Delta E_{read} + \Delta E_{write}$ as a trade-off, which is also validated to be effective by the experiments.

According to Definition 6 and Lemma 2, we can have the following Lemma for pages (moved from DRAM to NVM) in order to improve the consistency.

Lemma 3. *Page P_i in DRAM is suitable to be moved to NVM memory if $S_i^* < E_{NVM}^{read} + E_{DRAM}^{read} + E_{NVM}^{write} + E_{DRAM}^{write}$.*

Page grouping policy in DRAM is the same as that in NVM. However we do not need to consider the space limitation of NVM since NVM is much larger than DRAM. We only use DRAM to store pages in order to reduce the energy consumption.

4. Page replacement method with consistency guarantee: algorithm description

In this section, we first describe our EAPR algorithm in details. We present our consistency guarantee mechanism in two phases respectively, i.e., page replacement phase and transaction committing phase.

4.1. Energy aware page replacement algorithm

The details of EAPR are shown in Algorithm 1. Before the main procedure of EAPR, we initialize the spare DRAM size which is predefined to the pages migrated from NVM, as is seen in Line 1. EAPR is set to conduct page replacement periodically, which is assumed to be executed at the beginning of each period T (Lines 2–25). At the first part, we manage the pages stored in NVM (Lines 3–10). Each page in NVM is given a metric called substitution degree by using Eq. (7) (Lines 4–6). Then the pages which have continuous addresses and similar substitution degrees are defined as a page group (Line 7). All page groups are sorted according to the decreasing order of their space efficiency factor (Line 8). EAPR will greedily select page groups with high

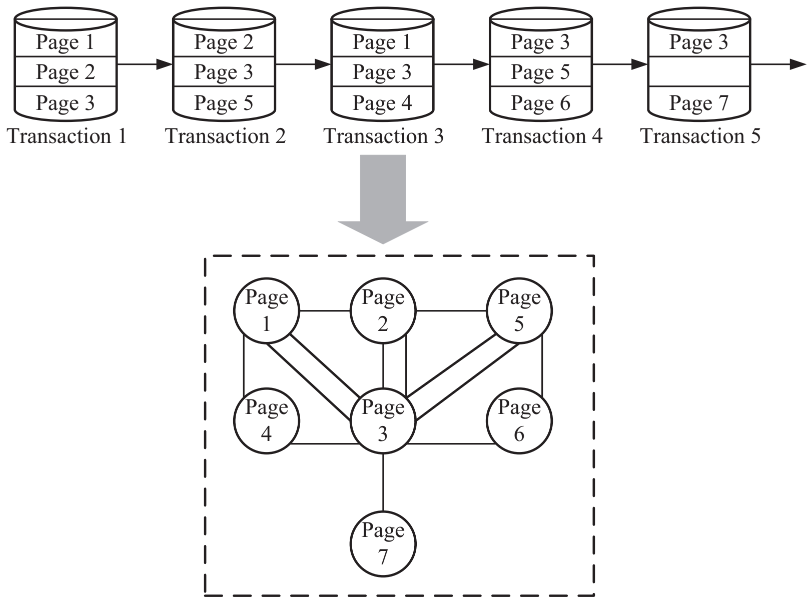


Fig. 4. Process of conducting undirected graph.

space efficiency until there is no spare DRAM space and migrate them to DRAM (Lines 9–10). Similarly, EAPR will manage the pages stored in DRAM at the second part (Lines 11–23). The improved substitution degree of each page in DRAM is calculated according to Eq. (10) (Lines 12–14). Then we group pages in DRAM and sort these page groups by the order of their substitution degrees (Lines 15–16). EAPR will select and migrate the pages suitable to be replaced from DRAM to NVM according to Lemma 3 (Lines 17–18). EAPR will conduct the consistency guarantee procedure at Line 19, due to the migration of pages from DRAM to NVM will cause consistency problem. The details of consistency guarantee procedure will be present in the following section. EAPR will also deal with the overflow problem of DRAM (Lines 20–23), where EAPR will replace the page group with lowest substitution degree from DRAM to NVM, then release the space of DRAM. Finally, EAPR will update the spare DRAM space (Line 24), which will be utilized to do page replacement next period.

4.2. Consistency procedure in page replacement phase

There are two situations needed to conduct consistency procedure, i.e., the situation of writing pages from DRAM to NVM in EAPR and the situation of user committing transaction. If the first situation happens during EAPR. We have to conduct a consistency procedure, which is shown in Fig. 5. In the process of writing back pages from DRAM to NVM, EAPR will first access the PFHT to find all transactions which have modified this page in the TILL, and then delete this page from

```
1:  Initialize $N$ (size of spare DRAM)
2:  while At the beginning of each $T$ do
3:      //Deal with pages within NVM
4:      for $\forall P_i$ in NVM do
5:          Calculate $S_i$ according to Eq.7
6:      end for
7:      Group pages in NVM with continuous addresses and similar substitution degrees
8:      Sort page groups by the decreasing order of space efficiency ($\theta$)
9:      Select page groups with highest $\theta$ satisfying Lemma 2 and $\sum V_i \le N$
10:     Migrate these selected page groups from NVM to DRAM
11:     //Deal with pages within DRAM
12:     for $\forall P_i$ in DRAM do
13:         Calculate $S_i^*$ according to Eq. 10
14:     end for
15:     Group pages in DRAM with continuous addresses and similar substitution degrees
16:     Sort groups by the increasing order of their substitution degree
17:     Select groups with low substitution degrees satisfying Lemma 3 for all pages of these groups
18:     Migrate these selected page groups from DRAM to NVM
19:     Implement the procedure of consistency guarantee
20:     if DRAM is full then
21:         Select page group $G_i$ with the lowest $S_i^{group*}$
22:         Migrate the chosen group from DRAM to NVM
23:     end if
24:     Update $N$ //spare DRAM size
25: end while
```

Algorithm 1. EAPR algorithm.

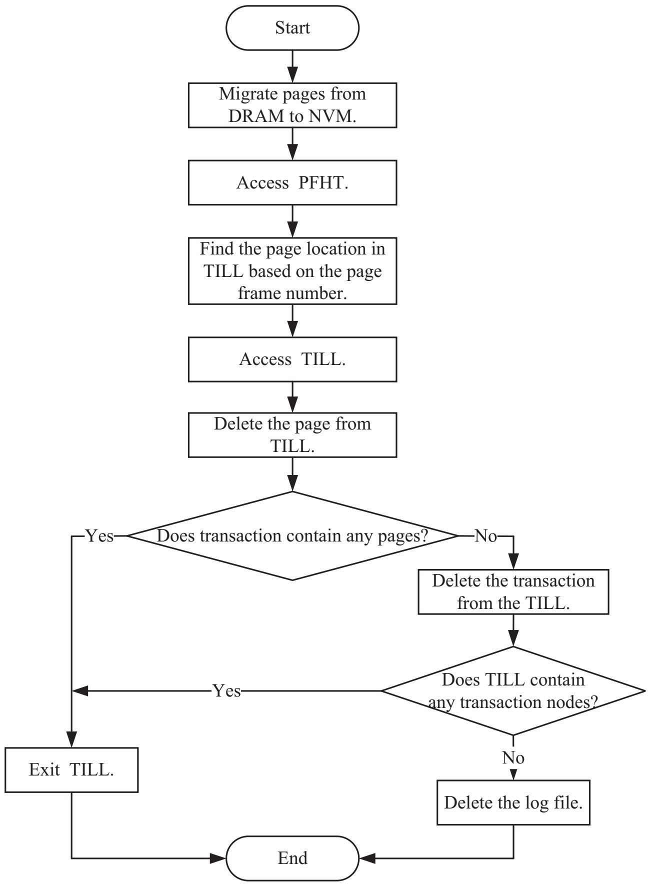

Start
Migrate pages from DRAM to NVM.
Access PFHT.
Find the page location in TILL based on the page frame number.
Access TILL.
Delete the page from TILL.
Does transaction contain any pages?
Yes
No
Delete the transaction from the TILL.
Does TILL contain any transaction nodes?
Yes
No
Exit TILL.
Delete the log file.
End

Fig. 5. Consistency guarantee in page replacement phase.

these transactions. After that, the transaction could be deleted from TILL if there is no page information in this transaction. Finally, the logs will be deleted if TILL is empty.

4.3. Consistency procedure in transaction committing phase

When user transactions are committed in user mode, EAPR (in kernel mode) will be informed which pages have been changed. Then EAPR will judge whether these pages are in the NVM according to PLIT. If these pages are all in NVM, none of the user transactions will be added to TILL because there is no page in DRAM which has been modified. At this time, if TILL is empty, the logs can be deleted from the main memory. If some pages are stored in NVM, the information of the modified pages in DRAM will be inserted to TILL. The consistency procedure of this phase is shown as Fig. 6. There might be a page replacement during modifying this page and informing EAPR by writing transaction, but it has no influence on the logic of our method. The final judgement result will keep correct.

5. Experiment evaluation

5.1. Experiment setup

For evaluating the performance of our EAPR, we use GEM5 combining with NVMAIN to simulate the hybrid memory systems. GEM5 simulator is a modular platform for computer architecture researches, including system-level architecture and processor's micro-architecture [23]. NVMAIN is a cycle accurate main memory simulator designed to simulate emerging non-volatile memories at the architectural level [24]. The configuration parameters of GEM5 which we used are listed in Table 2. The memory related energy parameters are listed in Table 3, according to [25]. We take the leakage power as the idle power. We conduct our EAPR algorithm with six workloads selected from SPEC CPU 2006 v1.2 [26] - *401.zlib, 403.gcc, 458.sjeng, 483.xalancbmk, 459.gemsDFTD, 429.mcf*, and the footprints of these chosen benchmarks are indicated in Table 4. The benchmarks in Table 4 focus on the performance of our EAPR in different DRAM spaces and the optimal ratios

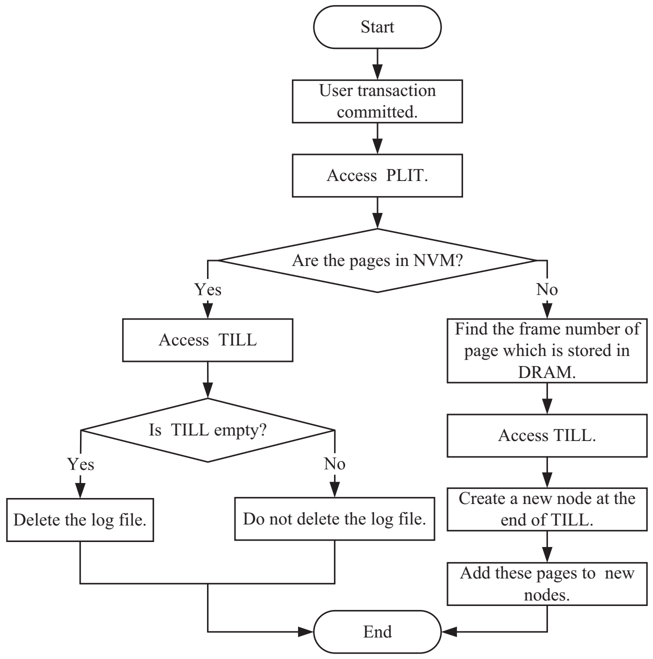

Start
User transaction committed.
Access PLIT.
Are the pages in NVM?
Yes
No
Access TILL
Find the frame number of page which is stored in DRAM.
Is TILL empty?
Yes
No
Access TILL.
Delete the log file.
Do not delete the log file.
Create a new node at the end of TILL.
Add these pages to new nodes.
End

Fig. 6. Consistency guarantee in transaction committing phase.

Table 2
Configuration of GEM5.

Processor	4 cores	
L1 cache		32 KB
L2 cache		6 MB
Memory apacity	NVM	4 GB
	DRAM	Based on benchmarks

Table 3
Parameters of PCM and DRAM.

	PCM	DRAM
Writing energy	418.6 nJ	12.7 nJ
Reading energy	80.41 nJ	5.9 nJ
Leakage power	4.23 mW/GB	451 mW/GB

Table 4
Footprints of benchmarks.

Benchmarks	Footprint (MB)	Describes
401.zlib	140	*Doing most work in memory, rather than doing I/O.*
403.gcc	290	*Based on gcc version 3.2, generates code for Opteron*
458.sjeng	690	*A highly-ranked chess program that plays several chess variants*
483.xalancbmk	720	*Transforming XML documents to other document types*
459.gemsFDTD	831	*Using the FDTD method to solve the Maxwell equations in 3D*
429.mcf	860	*Using a network simplx algorithm to schedule public transport*

Table 5
Three groups of PARSEC benchmarks.

Group #	Name	Description	R/W rate
1	*dedup*	*Compression with data deduplication*	1.28
	swaptions	*Pricing of a portfolio of swaptions*	1.36
	bodytrack	*Body tracking of a person*	1.41
2	*ferret*	*Content-similarity search server*	1.62
	fluidanimate	*Fluid dynamics for animation purpose*	1.67
	canneal	*Simulated cache-aware annealing*	1.99
3	*streamcluster*	*Online clustering of an input stream*	1.16
	blackscholes	*Option pricing with Black-Scholes PDE*	1.23
	vips	*Image processing*	1.27

of DRAM and NVM. However, the benchmarks have no detailed descriptions about the R/W ratio. Therefore, we add the new benchmarks in Table 5 to focus on the performance of our EAPR in different R/W ratios. As shown in Table 5, we evaluate EAPR with three groups of workloads selected based on PARSEC Benchmark Suite[27].

For comparison, we choose four existing algorithms as the candidates, i.e., RaPP[18], DMA[28], TBPI[29] and PMPC[30]. Furthermore, we also design NoManage as a intuitive candidate, in which without any page replacement method. We assume page allocation is random in NoManage, and the probability of assigning to NVM for each page is proportional to the ratio of NVM capacity to total memory size.

- **RaPP**[18]: Rank-based Page Placement algorithm;
- **DMA**[28]: Dynamic Measure and Allocation algorithm;
- **TBPI**[29]: Threshold-Based Pre-Invalidation algorithm;
- **PMPC**[30]: Page Migrated and Page Culling algorithm;
- **NoManage**: NVM–DRAM hybrid memory system without any page replacement method.

5.2. Energy cost comparison with DMA and RaPP

In the first experiment we make efforts to test the energy costs under various DRAM size. We conduct six benchmarks from SPEC CPU 2006 V1.2 for 100 trillion ticks in GEM5, each of them has seven different DRAM size settings. In this section, we compare our algorithm with RaPP, DMA and NoManage.

The normalized energy consumptions for all candidates are shown in Fig. 7. According to Fig. 7(a), we observe that EAPR can reduce energy consumption by 30.76%–83.38% compared with NoManage. EAPR also can reduce the energy consumption by 4.73%–21.18% compared with DMA. DMA only considers the number of page writing while ignoring the impact of page reading. Although EAPR has similar performance with RaPP, EAPR also gets a little lower energy consumption in some cases like DRAM size as 56 MB and 70 MB. EAPR also provides the guarantee of consistency for hybrid main memory systems, which has certain impact on the overall energy cost. Furthermore, we observe that EAPR gets the minimum energy cost when the DRAM space is about a half of the footprint of workload from Fig. 7(a–f). The reason is that more pages migrated to DRAM lead to the increase of DRAM capacity and DRAM standby energy consumption, which gradually increase the overall energy consumption. Therefore, we have a conclusion that the most efficient memory configuration is 1:2 for the ratio of DRAM space to NVM space.

Benchmarks with various characteristics may have impact on system performance. For example, some benchmarks which have many memory reading operations, such as *xalancbmk* and *gemsDFTD*, in such benchmarks energy consumption of EAPR is much lower than DMA. Compared with RaPP, energy consumption of EAPR is higher than RaPP in *zlib, sjeng* and *gemsDFTD*, while it is lower than RaPP in *gcc, xalancbmk* and *mcf*. The reason is that RaPP migrates popular pages to DRAM, which has substantially better performance in the presence of row buffer misses than NVM. In *zlib, sjeng* and *gemsDFTD*, there are a lot of repeated reading and writing at a certain area of main memory, which is more suitable for RaPP. However, in *gcc, xalancbmk* and *mcf*, the memory reading and writing are not repeated, especially in *xalancbmk*. For these cases, EAPR has higher energy saving than RaPP. Based on these observations, we conclude that EAPR is the best method. Although providing the consistency guarantee for hybrid memory systems has certain impact on the overall energy consumption, EAPR can save more energy consumption than the existing methods.

5.3. Energy cost comparison with TBPI and PMPC

In this experiment, we mainly compare EAPR with TBPI and PMPC. We conduct the experiment using the same benchmarks as the first experiment. Fig. 8 shows the energy consumptions of EAPR, TBPI, PMPC, and NoManage. According to the experimental results, we have following observations. The energy consumption of EAPR is lower than both TBPI and PMPC. Although EAPR has near performance as TBPI, EAPR obtains lower energy cost in all cases than TBPI. TBPI has considered the energy consumption of hybrid memory systems, but it does not take into account a variety of conditions, such as the energy influence of reading operations on pages. Compared with PMPC, EAPR has much better energy saving in all cases. This is because that PMPC is a simple method which only considers the hot/cold page's migration. We can have the conclusion that our EAPR algorithm is completely effective compared with TBPI and PMPC.

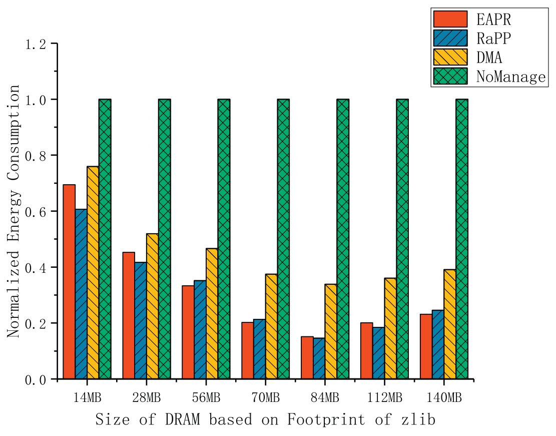
5.4. Impact of R/W rate on energy consumption

We conduct three groups of benchmarks based on PARSEC benchmark suite for 100 trillion ticks in GEM5. We set the ratio between DRAM space and NVM space as 1:2. The detailed R/W rate settings are indicated in Table 5. Energy costs of these three groups are shown as Fig. 9. We can observe that the energy consumptions of EAPR are much lower than other algorithms in all cases. The reason is that these three groups of benchmarks all have large R/W rates according to Table 5. EAPR can obtain higher efficiency for benchmarks with the larger R/W rate due to the consideration of reading and writing energy model. Thus we can conclude that our EAPR algorithm is more suitable for application scenarios with larger reading requests on main memory systems.

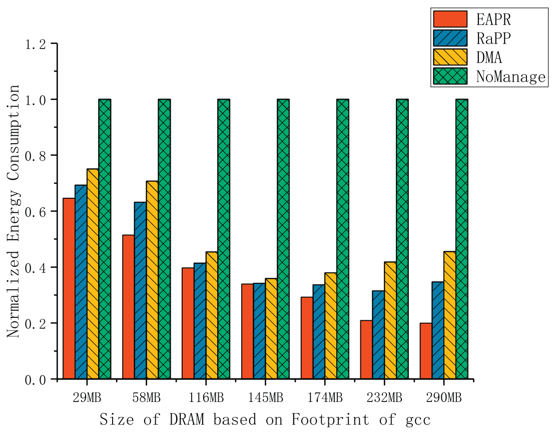
5.5. Time overhead comparison

In this section, we conduct a experiment to test the time overheads of these page replacement algorithms. We implement EAPR, DMA, RaPP, TBPI, PMPC and NoManage, and record their time overheads, which are demonstrated in Fig. 10. Based on the time overhead results, we obtain that the time overhead of EAPR is on average 32.14% more than NoManage, 21.43% more than DMA, 16.32% more than PMPC, 10.49% more than RaPP and 7.85% more than TBPI. We didn't give the experimental results of the overhead of single guarantee consistency because the cohesion of page replacement and consistency guarantee is too high in our EAPR algorithm which is based on logs. As is seen in Algorithm 1, the algorithm cannot be separated. So we conducted the experiment to test the overheads of our EAPR with consistency guarantee and other algorithms in this section. Although EAPR has the highest overhead, it is the only algorithm which can provide consistency guarantee for hybrid main memory systems. From Fig. 10, the time overhead of our EAPR is still relatively low compared with its energy saving.

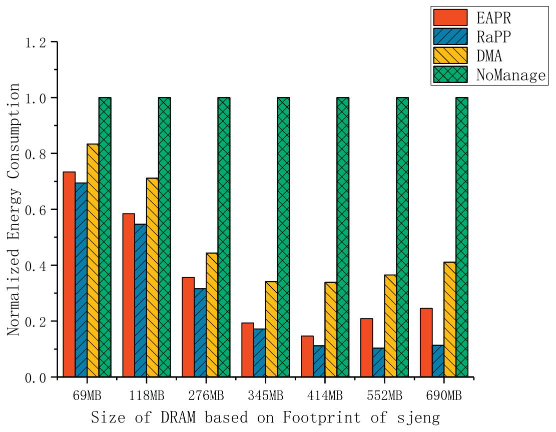
In addition, we conduct another experiment to test the memory costs of these three data structures (TILL, PLIT and PFHT) using the benchmark 458.sjeng. We test for the first 10 time nodes. As shown in Fig. 11, the memory costs of the TILL and the PLIT is not large at the beginning. This is because they are a linked list and an information table, and do not occupy too much memory space when there is not much stored data. However, PFHT has a relatively large memory



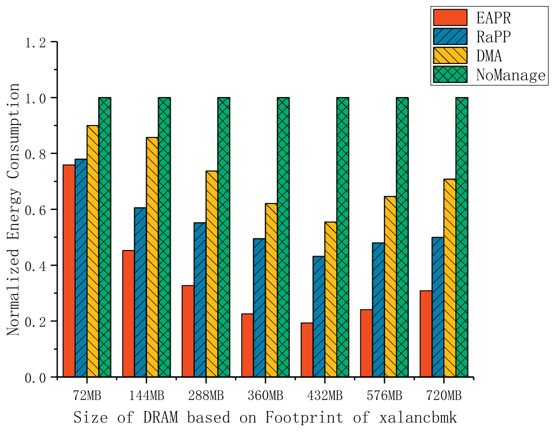
(a) zlib



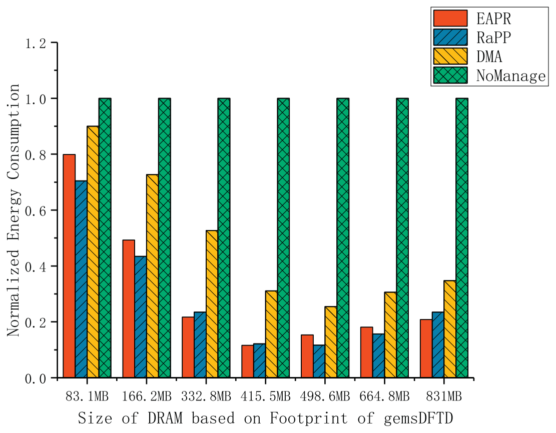
(b) gcc



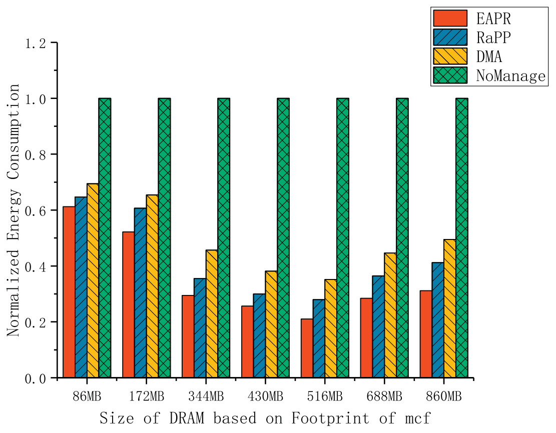
(c) sjeng



(d) xalancbmk



(e) gemsDFTD



(f) mcf

Fig. 7. Energy consumption comparison (Different DRAM space) - 1.

footprint, occupying 83 MB of memory space at the beginning. This is because PFHT is a hash table, which occupies a lot of memory space when the table is built. During this experiment, the memory costs of PLIT fluctuates a little. The memory cost of PLIT is low because the stored data is relatively simple and does not occupy much memory space. The memory cost of PFHT is also acceptable because there is no large fluctuation during system running. Although TILL occupies only 31 MB of memory during construction, the memory cost of TILL increases at a very high speed during the system running, reaching a maximum of 134 MB. The nodes of TILL increase rapidly due to a large number of writing transactions, which consumes a large amount of memory space. According to the characteristics of the log-based file system, the nodes of TILL will be reduced slowly because of the design of TILL in our EAPR.

5.6. Implementation and evaluation of consistency guarantee

In order to test the effectiveness of consistency guarantee, we implement EAPR on Linux 4.4.0, running on a computer with Intel Core i5-4460 3.20GHz processor and a hybrid memory including 8 GB DRAM

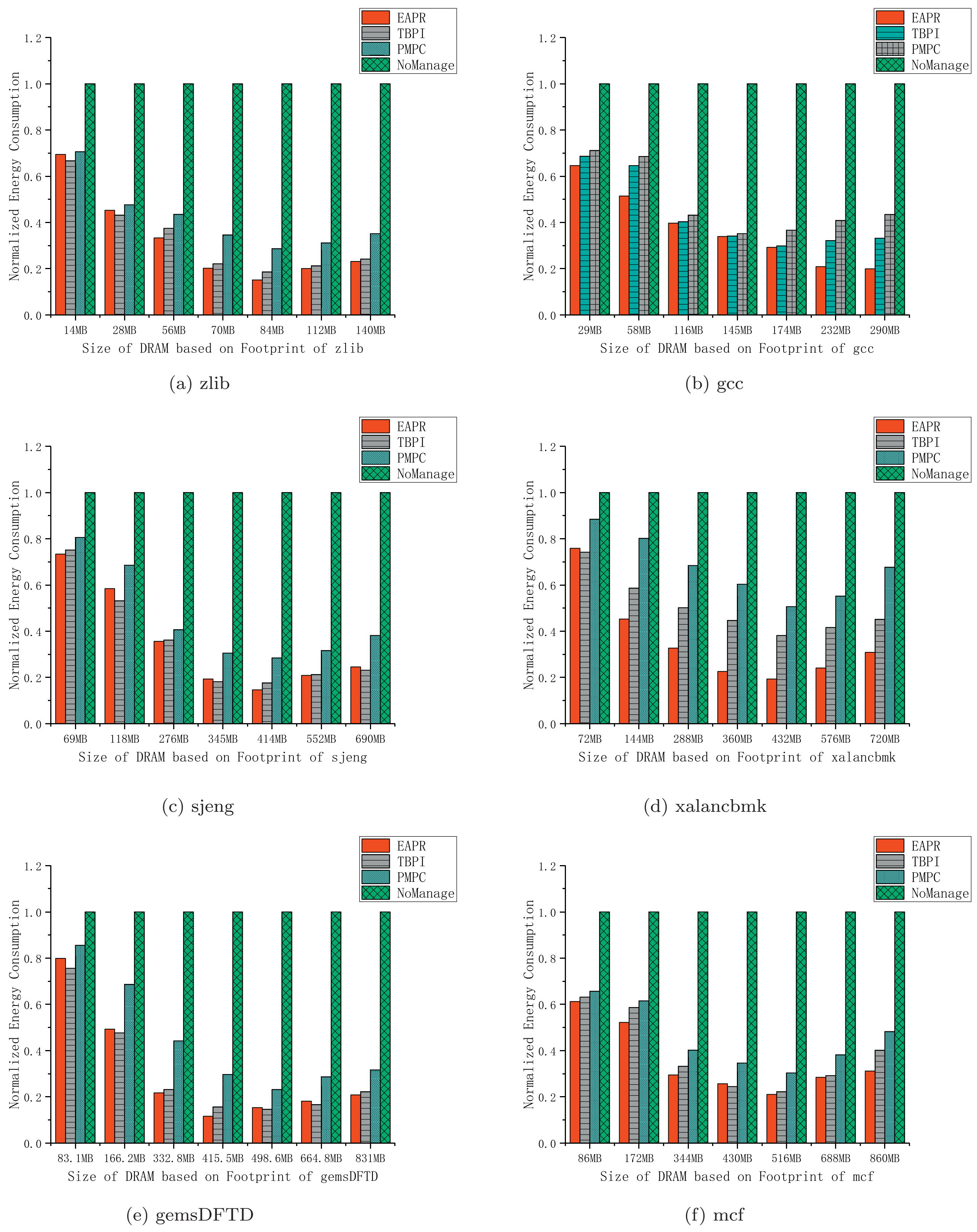


Fig. 8. Energy consumption comparison (Different DRAM space) - 2.

and 4 GB NVDIMM. NVDIMM is utilized to simulate NVM to test the consistency guarantee mechanism of EAPR.

In this experiment, the EAPR algorithm is implemented as a simple character driver that directly manages the memory space of NVDIMM. EAPR will allocate a certain number of DRAM pages directly from the buddy system as the replacement space. In addition to EAPR, we also implement the mapping of NVM page to the process's address space. That is to say, applications can apply pages in NVM directly. EAPR also provides I/O control interfaces for the application calls, including the notification interface of logs and the notification interface of user

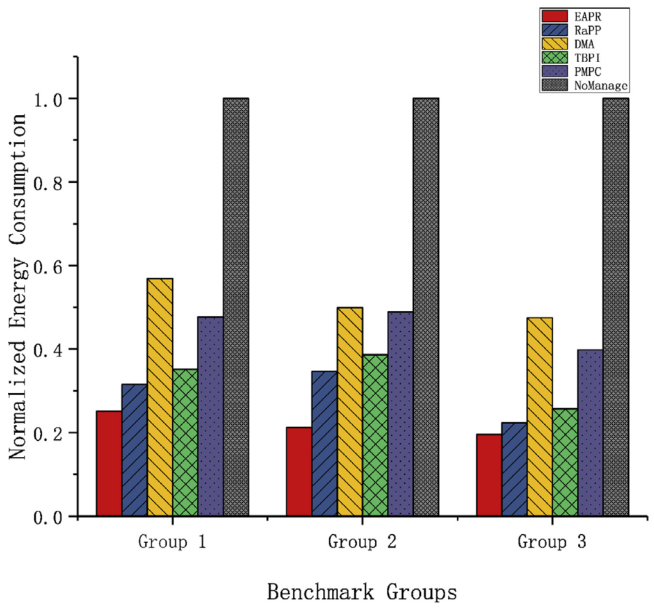


Fig. 9. Energy consumption comparison (Different R/W rate).

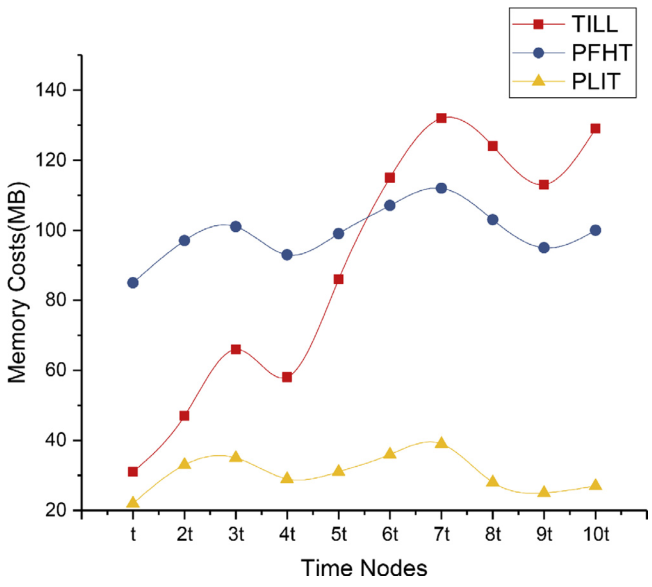


Fig. 11. Memory costs of data structures.

transaction submissions. In the notification interface of user transaction submissions, the application not only needs to provide the transaction information to modify the page information, but also needs to provide a status field address for EAPR calls. Then EAPR can use this status field to notify the application to delete logs. Furthermore, the status field should be stored in NVM, and EAPR will map the NVDIMM space in the kernel address space by using 1 GB page space. We need not to evaluate the application-level performance and the recovery performance of EAPR because we only focus on the effect of system recovery when a system failure occurs. Moreover, the system fault recovery algorithm of EAPR is based on NOVA [22], and the system failure recovery efficiency is not involved in EAPR.

We select several benchmarks used by NOVA to do the experiment. Two kinds of benchmarks are selected, i.e., micro benchmark and macro benchmark. Micro-benchmark creates 10,000 files, and calls *fsync* to persist the files, and finally deletes them. For macro-benchmarks, we select four *Filebench* workloads, i.e., *fileserver, webproxy, webserver* and *varmail* to run in EAPR. Furthermore, according to LSA algorithm given in [31], we implement a series of user transactions, such as data access, modification and deletion. We also implement the management of data structures on NVM including search, delete, insert and save operations on the linked list and hash tables.

We also realize a consistency check tool to verify the validity of consistency guarantee in EAPR. During the running period of those benchmarks, we simulate the occurrence of system failures through

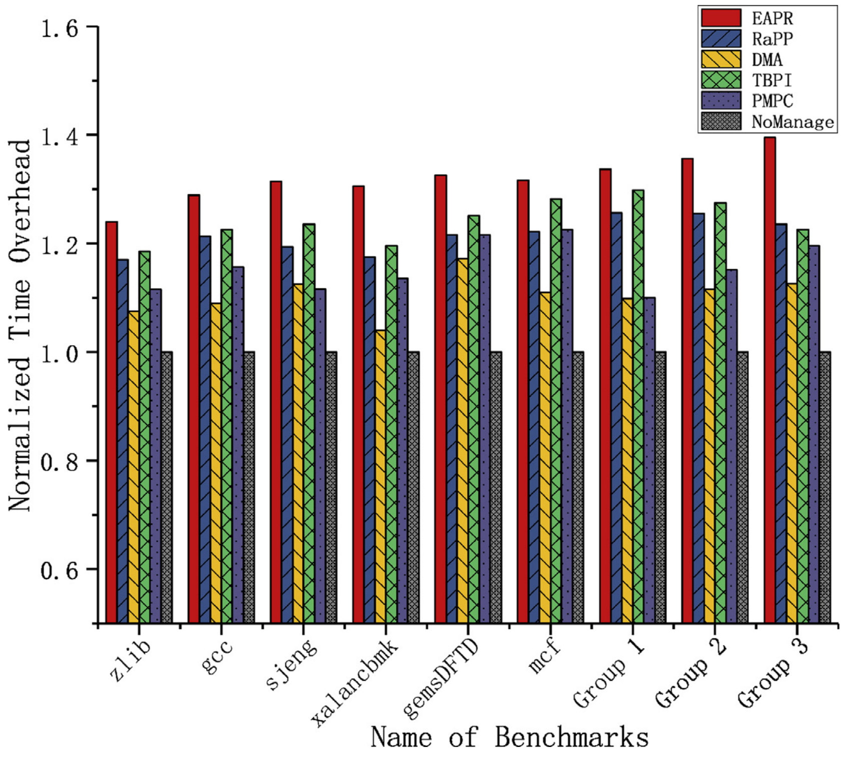


Fig. 10. Time overhead comparison.

power-off or other operations. Then the system will be reopened and restored by the method of NOVA. After the system has been restored, we use the tool to check whether the application is in an inconsistent state. Fortunately, the consistency is guaranteed in our experiments. Therefore EAPR can provide the consistency guarantee for hybrid memory systems and recover the system from a system failure state to the latest stable state.

6. Related work

In this section, we overview researches on hybrid memory systems from three perspectives: increasing lifetime of NVM, saving energy consumption of hybrid memory systems, and guaranteeing the consistency of hybrid memory systems.

There exist many studies to increase the life span of NVM memory systems. First, writing reduction techniques are utilized as effective methods to save energy, e.g., Lazy Write and Page Level Bypass [32], Partial Writes [17], and Redundant Bit-write Removal [33]. Lazy Write and Page Level Bypass took advantages of DRAM buffer to reduce writing traffic to NVM. Partial Writes aimed at reducing writing accesses to memory, via tracking cache modification and writing only dirty lines. Redundant Bit-write Removal was proposed to reduce writing frequency to NVM only if the stored value is different with the writing value. Moreover, Mittal and Vetter proposed a technique, named AYUSH, to enhance the lifetime of hybrid caches [34]. Preferentially using SRAM to store frequent-reused data during data migration, AYUSH can achieve larger improvement in lifetime and maintain system performance.

Researchers also made efforts to save energy consumptions of memory systems. Hassan and Vandierendonck proposed a hybrid memory system, consisting of DRAM and NVM, to reduce overall energy of the memory systems [25]. They found that placing application objects on hybrid memory got more energy savings than state-of-the-art OS-level page migration or hardware approaches. Salkhordeh and Asadi proposed an efficient data migration scheme at the Operating System Level in a hybrid DRAM-NVM memory architecture [35]. Two Least Recently Used (LRU) queues, one for DRAM section and one for NVM section, are devised for the sake of data migration. Zhang and Yang proposed a software system, named Mojim[36], to provide reliability and availability in large-scale NVM-based storage systems. Mojim used a two-tier architecture to achieve the goals of minimizing replication costs and exploiting as much of NVMs performance as possible.

There are also several studies focused on guaranteeing the consistency of NVM–DRAM main memory system. Xu and Swanson proposed a file system named NOVA [22], which was designed to maximize performance on hybrid memory systems while providing strong consistency guarantee. NOVA is a log-structure file system, which adapts conventional log-structured file system techniques to exploit the fast random access on NVM. The logs of NOVA provide metadata, data, and mmap atomicity focused on simplicity and reliability. Keeping complex metadata structures in DRAM can accelerate lookup operations in NOVA. Zhang and Ju presented a hybrid buffer cache architecture by combining NVM with DRAM to reduce the journaling overhead and overcome the disadvantages of NVM [37]. In order to better utilize such a novel architecture, they proposed a Journaling-Aware Page Management (JAPM) policy to put infrequently-update data in NVM to reduce the journaling overhead, whereas put frequently-update data in DRAM to improve the writing performance and lifetime. For guaranteeing the atomicity of the transactional execution in the hybrid cache architecture, a Partial In-Place committing (PIPC) journaling scheme is also proposed to coordinate different committing patterns.

Different to the existing researches, our method takes into account both reducing the energy consumption and guaranteeing the consistency of hybrid NVM–DRAM main memory systems.

7. Conclusions

In this paper we have made efforts to study the page replacement method on hybrid NVM–DRAM memory systems. We focused on the energy efficiency of page replacement. We established the energy cost model for page replacement and designed group-based page migration method to further improve the efficiency. We introduced the substitution degree for each page to indicate whether it is suitable to be moved between NVM and DRAM. To address the consistency problem within hybrid NVM–DRAM page migration, we have designed two consistency guarantee schemes in page replacement phase and transaction committing phase respectively. Our approach can not only choose the migrated pages according to the replacement energy consumption but also provide a log-based transaction mechanism to guarantee the consistency. We have conducted extensive experiments to evaluate the proposed method based on real benchmarks. The experimental results demonstrate that our method can efficiently reduce energy consumption up to 40.19% compared with the existing page replacement algorithms, and can also guarantee the consistency of transactions in NVM–DRAM hybrid memory systems with the acceptable time overhead. In the future, we will make efforts to separate the consistency guarantee from the page replacement to lower the cohesion of these two parts. We will also apply our EAPR approach to real applications to make it more practical.

Acknowledgements

This work was partly supported by the Research Fund of National Key Laboratory of Computer Architecture under Grant no. CARCH201501, and the Open Project Program of the State Key Laboratory of Mathematical Engineering and Advanced Computing under Grant no. 2016A09.

References

[1] Y. Zhang, J. Zhan, J. Yang, W. Jiang, L. Li, Y. Li, Energy-aware page replacement for nvm-based hybrid main memory system, the 23rd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), IEEE, 2017, pp. 1–6.
[2] F. Hameed, J. Castrillon, Rethinking on-chip DRAM cache for simultaneous performance and energy optimization, Proc. of Design, Automation Test in Europe Conference and Exhibition (DATE), IEEE, 2017, pp. 362–367.
[3] J.H. Choi, G.H. Park, Nvm way allocation scheme to reduce nvm writes for hybrid cache architecture in chip-multiprocessors, IEEE Trans. Parallel Distrib. Syst. 28 (10) (2017) 2896–2910.
[4] G. Jia, G. Han, H. Wang, F. Wang, Cost aware cache replacement policy in shared last-level cache for hybrid memory based fog computing, Enterpr. Inf. Syst. (2017) 1–17.
[5] G. Jia, G. Han, J. Jiang, L. Liu, Dynamic adaptive replacement policy in shared last-level cache of DRAM/PCM hybrid memory for big data storage, IEEE Trans. Ind. Inf. 13 (4) (2017) 1951–1960.
[6] M.K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, B. Abali, Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling, Proc. 42nd Annual IEEE/ACM Int. Symp. Microarchitecture (MICRO), IEEE, 2009, pp. 14–23.
[7] G. Sun, X. Dong, Y. Xie, J. Li, Y. Chen, A novel architecture of the 3D stacked mram l2 cache for cmps, Proc. IEEE 15th Int. Symp. High Performance Computer Architecture, IEEE, 2009, pp. 239–249.
[8] J. Zhao, C. Xu, P. Chi, Y. Xie, Memory and storage system design with nonvolatile memory technologies, IMT 10 (2015) 2–11.
[9] Y.J. Lin, C.L. Yang, H.P. Li, C.Y.M. Wang, A buffer cache architecture for smartphones with hybrid DRAM/PCM memory, Proc. IEEE Non-Volatile Memory System and Applications Symp. (NVMSA), IEEE, 2015, pp. 1–6.
[10] H. Yoon, J. Meza, R. Ausavarungnirun, R.A. Harding, O. Mutlu, Row buffer locality aware caching policies for hybrid memories, Proc. IEEE 30th Int. Conf. Computer Design (ICCD), IEEE, 2012, pp. 337–344.
[11] W. Zhang, T. Li, Exploring phase change memory and 3D die-stacking for power/thermal friendly, fast and durable memory architectures, Proc. 18th Int. Conf. Parallel Architectures and Compilation Techniques, IEEE, 2009, pp. 101–112.
[12] A. Hassan, H. Vandierendonck, D.S. Nikolopoulos, Software-managed energy-efficient hybrid dram/nvm main memory, Computing Frontiers, IEEE, 2015, pp. 1–8.
[13] S. Zheng, L. Huang, H. Liu, L. Wu, J. Zha, Hmvfs: A hybrid memory versioning file system, Proc. 32nd Symp. Mass Storage Systems and Technologies (MSST), IEEE, 2016, pp. 1–14.
[14] T. Liu, Y. Zhao, C.J. Xue, M. Li, Power-aware variable partitioning for dsps with

hybrid pram and DRAM main memory, IEEE Trans. Signal Process. 61 (14) (2013) 3509–3520.

[15] E. Chen, D. Apalkov, Z. Diao, A. Driskill-Smith, D. Druist, D. Lottis, V. Nikitin, X. Tang, S. Watts, S. Wang, S.A. Wolf, A.W. Ghosh, J.W. Lu, S.J. Poon, M. Stan, W.H. Butler, S. Gupta, C.K.A. Mewes, T. Mewes, P.B. Visscher, Advances and future prospects of spin-transfer torque random access memory, IEEE Trans. Magn. 46 (6) (2010) 1873–1878.

[16] S. Lee, H. Bahn, S.H. Noh, Characterizing memory write references for efficient management of hybrid PCM and DRAM memory, Proc. and Simulation of Computer and Telecommunication Systems 2011 IEEE 19th Annual Int. Symp. Modelling, Analysis, IEEE, 2011, pp. 168–175.

[17] B.C. Lee, E. Ipek, O. Mutlu, D. Burger, Architecting phase change memory as a scalable dram alternative, SIGARCH Comput. Archit. News 37 (3) (2009) 2–13.

[18] L.E. Ramos, E. Gorbatov, R. Bianchini, Page placement in hybrid memory systems, ICS ’11, ACM, 2011, pp. 85–95.

[19] E. Doller, Forging a future in memory: New technologies, new markets, new applications, Proc. IEEE Hot Chips 22 Symp. (HCS), IEEE, 2010, pp. 1–28.

[20] Micron, Micron TN-41-01: Calculating Memory System Power.

[21] X. Dong, C. Xu, Y. Xie, N.P. Jouppi, Nvsim: a circuit-level performance, energy, and area model for emerging nonvolatile memory, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 31 (7) (2012) 994–1007.

[22] J. Xu, S. Swanson, Nova: a log-structured file system for hybrid volatile/non-volatile main memories, Proceedings of the 14th USENIX Conference on File and Storage Technologies, IEEE, 2016.

[23] N. Binkert, B. Beckmann, G. Black, S.K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D.R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M.D. Hill, D.A. Wood, The gem5 simulator, SIGARCH Comput. Archit. News 39 (2) (2011) 1–7.

[24] M. Poremba, T. Zhang, Y. Xie, Nvmain 2.0: a user-friendly memory simulator to model (non-)volatile memory systems, IEEE Comput. Archit. Lett. 14 (2) (2015) 140–143.

[25] A. Hassan, H. Vandierendonck, D.S. Nikolopoulos, Energy-efficient hybrid DRAM/nvm main memory, PACT’15, IEEE, 2015, pp. 492–493.

[26] J.L. Henning, Spec cpu2006 benchmark descriptions, Acm Sigarch Comput. Architect. News 34 (4) (2006) 1–17.

[27] C. Bienia, S. Kumar, J.P. Singh, K. Li, The parsec benchmark suite: characterization and architectural implications, International Conference on Parallel Architectures and Compilation Techniques, IEEE, 2017, pp. 72–81.

[28] Y. Zhang, J. Zhan, J. Yang, W. Jiang, L. Li, L. Zhu, X. Tang, Dynamic memory management for hybrid DRAM-nvm main memory systems, Proc. 13th Int. Conf. Embedded Software and Systems (ICESS), IEEE, 2016, pp. 148–153.

[29] H.G. Lee, S. Baek, C. Nicopoulos, J. Kim, An energy- and performance-aware dram cache architecture for hybrid dram/pcm main memory systems, IEEE International Conference on Computer Design, IEEE, 2011, pp. 381–387.

[30] Y. Park, D.J. Shin, S.K. Park, K.H. Park, Power-aware memory management for hybrid main memory, The International Conference on Next Generation Information Technology, IEEE, 2011, pp. 82–85.

[31] P. Felber, C. Fetzer, P. Marlier, T. Riegel, Time-based software transactional memory, IEEE Trans. Parallel Distrib. Syst. 21 (12) (2010) 1793–1807.

[32] M.K. Qureshi, V. Srinivasan, J.A. Rivers, Scalable high performance main memory system using phase-change memory technology, Proceedings of the 36th Annual International Symposium on Computer Architecture, ACM, 2009, pp. 24–33.

[33] P. Zhou, B. Zhao, J. Yang, Y. Zhang, A durable and energy efficient main memory using phase change memory technology, ACM Sigarch Comput. Architect. News 37 (3) (2009) 14–23.

[34] S. Mittal, J.S. Vetter, Ayush: a technique for extending lifetime of sram-nvm hybrid caches, IEEE Comput. Archit. Lett. 14 (2) (2015) 115–118.

[35] R. Salkhordeh, H. Asadi, An operating system level data migration scheme in hybrid DRAM-nvm memory architecture, DATE 2016, IEEE, 2016, pp. 936–941.

[36] Y. Zhang, J. Yang, A. Memaripour, S. Swanson, Mojim: a reliable and highly-available non-volatile memory system, The 20th International Conference on Architectural Support for Programming Languages and Operating Systems, IEEE, 2015, pp. 3–18.

[37] Z. Zhang, L. Ju, Z. Jia, Unified DRAM and nvm hybrid buffer cache architecture for reducing journaling overhead, DATE 2016, IEEE, 2016, pp. 942–947.

Jinyu Zhan received the B.S. degree, the M.S. degree and the Ph.D. degree from the University of Science and Technology of China. She is currently an Associate Professor with the School of Information and Software Engineering, University of Electronic Scienc and Technology of China. Her research interests include embedded system design, optimization of computer architecture, storage and system co-design.

Yiming Zhang is a master student of University of Science and Technology of China. He received his B.S. degree from Chengdu University of Technology, Chengdu, China in 2014. He received his M.S. degree in software engineering from University of Science and Technology of China in 2017. His research interests include embedded systems and storage.

Wei Jiang received the B.S. degree, the M.S. degree and the Ph.D. degree from the University of Electronic Scienc and Technology of China. He is currently an Associate Professor with the School of Information and Software Engineering, University of Electronic Scienc and Technology of China. His research interests include real-time system, security/energy aware design, embedded system design, advanced computer architecture and reconfigurable computing. He is a member of ACM and IEEE.

Junhuan Yang is a master student of University of Science and Technology of China. He received his B.S. degree in software engineering from University of Science and Technology of China in 2015. His research interests include embedded systems and storage.

Lin Li received the B.S. degree, the M.S. degree and the Ph.D. degree from the University of Science and Technology of China. He is currently an Associate Professor with the School of Computer Science and Engineering, University of Science and Technology of China. His research interests include operating systems, distributed computing and storage.

Yixin Li is a master student of University of Science and Technology of China. She received her B.S. degree in software engineering from the University of Science and Technology of China in 2016. Her research interests include embedded systems and storage.