Energy-Aware Page Replacement for NVM-based Hybrid Main Memory System

Yiming Zhang*, Jinyu Zhan*, Junhuan Yang*, Wei Jiang*‡, Lin Li† and Yixin Li*
*School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, China
†School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China
‡State Key Lab of Mathematical Engineering and Advanced Computing, Wuxi, China
Email: zhangluforever@hotmail.com; zhanjy@uestc.edu.cn (corresponding author);
yjhherny@163.com; weijiang@uestc.edu.cn; lilin@uestc.edu.cn; liyiyi1001@163.com

***Abstract*—With the advantage of low power consumption, Non-Volatile Memories (NVMs) has been widely used in hybrid memory architecture. This paper presents a page replacement method based on NVM-DRAM hybrid main memory system for low power and consistency guarantee, called EAPR The energy consumption of page access in DRAMs and NVMs can be calculated according to the memory access, and the pages are migrated according to their energy consumption, by which the pages are determined to migrate from NVM to DRAM or from DRAM to NVM. Instead of deleting the logs directly, our approach guarantees the consistency of the hybrid memory architecture by optimizing the structure of logs after the transactions of the applications are submitted. Finally, the experimental results show that EAPR can not only reduce the energy consumption at least 20% compared with other page replacement algorithms but also guarantee the consistency of transactions in the NVM-DRAM hybrid memory system.**

***Index Terms*—NVM; page replacement; energy consumption; consistency guarantee**

I. Introduction

NVM, with attractive characteristics of large capacity and low energy consumption, has obtained much attention from academic community. However, NVM has certain disadvantages for write operations. For instance, the write latency of NVM is 5 to 10 times higher than that of DRAM, and the write energy consumption of NVM is much higher. Besides, the write times of NVM is limited. Hence, we tend to place NVMs on the processor memory bus alongside conventional DRAM, leading to hybrid volatile/non-volatile main memory systems.

Research on hybrid main memory has predominantly pursued software-oblivious memory management executed by the hardware [1] or involving the OS [2]. In both cases, the memory access properties of large chunks of data are analyzed on-line and chunks are migrated to the most appropriate memory type. Zhang and Ju presented a hybrid buffer cache architecture by combing NVM with DRAM to reduce the journaling overhead and overcome the constraints of NVM [3]. For energy saving, Hassan and Vandierendonck propose a hybrid memory system, consisting of DRAM and NVM, to reduce overall energy of the memory system [4].

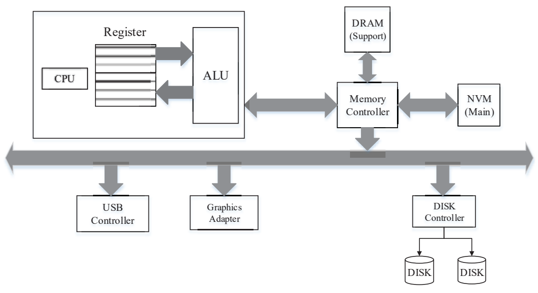


Fig. 1: System Architecture

Meanwhile, conventional file systems are not suitable for NVM-based hybrid memory system, which cannot address the problem of consistency guarantee for hybrid memory system, such as data missed and system crashes. The reason is that the data of DRAM will be lost when system failure occurred. System failures include power-fail, process error and so on. Thus Xu and Swanson proposed a file system named NOVA, which designed to maximize performance on hybrid memory systems while providing strong consistency guarantee [5]. NOVA is a log-structure file system, which uses transactions mechanism and log structure to guarantee the consistency of hybrid memory system.

In this paper, we present a page replacement method based on NVM-DRAM hybrid main memory system for low power and consistency guarantee, called EAPR (Energy-Aware Page Replacement) . Fig. 1 shows the target architecture focused in this paper. The architecture has a hybrid memory including 2 memory units, one of them is NVM and the other one is DRAM. Our approach focuses on modeling the page energy consumption, and the similar pages with continuous address are accessed as a page group. Then the page groups are chosen to migrate from DRAMs to NVMs or vice versa. Furthermore, for the consistency guarantee, we added three data structures to a log-structure file system. Thus EAPR can guarantee the consistency of transactions even if there are some page replacements.

The primary contributions of this paper are listed as follows.

- Analyzing the features of energy consumption during migrating the pages between DRAMs and NVMs.

- Proposing a energy-based page replacement algorithm for hybrid main memory architecture.
- Presenting an improved transaction scheme for satisfying the consistency guarantee when system failure occurs.
- Demonstrating the superiority of the proposed mechanism by experiments.

II. Energy Consumption Based Page Selection

In this section, we present an algorithm to choose the pages based on the energy consumption. Firstly, we introduce the replacement model by giving certain definitions. And then, we present the scheme of page group.

A. Energy Model of Replacement

Our algorithm is used to select the pages to replace them from NVM to DRAM, mainly consider from the viewpoint of energy consumption. The energy consumption here includes energy consumption of read and write, and the energy consumption of idle duration.

Definition 1: Given page P_i in the NVM, we use C_{NVM} to denote the energy consumption during the time period of T:

$$C_{NVM} = E^{read}_{NVM} * r_i + E^{write}_{NVM} * w_i + POW^{idle}_{NVM} * T \quad (1)$$

where E^{read}_{NVM} is the read energy consumption of each page in NVM, E^{write}_{NVM} is the write energy consumption of each page in NVM(nJ), r_i and w_i are the number of reads and writes in NVM of each page respectively, and P^{idle}_{NVM} is unit time idle power of each NVM page(mW/GB). In particular, as Table II shows, POW^{idle}_{NVM} can be ignored because the idle energy consumption of NVM is much lower than DRAM [6] [7].

Definition 2: Given page P_i in the DRAM, we use C_{DRAM} to denote the energy consumption during the time period of T:

$$C_{DRAM} = E^{read}_{DRAM}*r_i+E^{write}_{DRAM}*w_i+POW^{idle}_{DRAM}*T \quad (2)$$

where E^{read}_{DRAM} is the read energy consumption of each page in DRAM, E^{write}_{DRAM} is the write energy consumption of each page in DRAM and POW^{idle}_{DRAM} is unit time idle power of each DRAM page.

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

$$\begin{aligned} C_{NVM-DRAM} =& E_{extra} + (E^{read}_{NVM} + E^{write}_{DRAM})+ \\ & (E^{read}_{DRAM} * r_i + E^{write}_{DRAM} * w_i + \\ & POW^{idle}_{DRAM} * T) + (E^{read}_{DRAM}+ \\ & E^{write}_{NVM}) \end{aligned} \quad (3)$$

where E_{extra} is the extra energy consumption of the migration process in addition to the read and write consumption, $(E^{read}_{NVM} + E^{write}_{DRAM})$ means the energy consumption of a page once migrating to the DRAM, $(E^{read}_{DRAM} * r_i + E^{write}_{DRAM} * w_i + POW^{idle}_{DRAM} * T)$ represents the reads, writes as well as idle energy consumption in DRAM for a page under the same conditions, and $(E^{read}_{DRAM}+E^{write}_{NVM})$ is the energy consumption of writing back from DRAM to NVM, this write back energy consumption must be considered, the reason is that the page will be replaced to NVM if it has been stored in DRAM.

Definition 4: If P_i will be replaced from NVM to DRAM, it is energy-effective for P_i if $(C_{NVM} - C_{NVM-DRAM} > 0)$. Then we have following inequality:

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

Since POW^{idle}_{NVM} is very small in the process, it can be ignored in Definition 4. The Inequality 4 can be simplified as follows.

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

We define $(E^{read}_{NVM} - E^{read}_{DRAM})$ and $(E^{write}_{NVM} - E^{write}_{DRAM})$ as ΔE_{read} and ΔE_{write} respectively, then the Inequality 5 is changed as follows.

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

In Inequality 6, the right side of the Inequality 6 is a constant, so we define the formula on the left side of the Inequality 6 as the substitution degree of page P_i.

Definition 5: We use S_i to denote the substitution degree for P_i, and it can be defined as:

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

Because the energy consumption of reads and writes in NVM are both greater than DRAM [7], [8], the ΔE_{read} and ΔE_{write} both are constants and greater than 0. Accordingly, we get a conclusion that in the condition of $S_i > E^{read}_{NVM} + E^{read}_{DRAM} + E^{write}_{NVM} + E^{write}_{DRAM}$, the greater S_i is, the more suitable to replace into DRAM for a page. In this way, EAPR can reduce the energy consumption of hybrid memory system in terms of memory reads and writes.

B. Page Group

In hybrid main memory system, the page replacement process generally does not choose single page to migrate. This is because the replacement of individual page may cause different pages of the same process to be stored in NVM and DRAM discretely, resulting in address discontinuities, and it seriously affects the read, write and execution efficiency of the hybrid main memory system. Thus in EAPR, some continuous pages with similar substitution degree are defined as a page group, and the page replacement unit is the page group not a single page.

Definition 6: We use S^{group}_i to denote the substitution degree of page group G_i, which is defined as:

$$S^{group}_i = \sum_{P_i \in G_i} S_i \quad (8)$$

In the NVM-DRAM hybrid main memory system, NVM is the main memory and DRAM plays the auxiliary memory. So NVM capacity is larger, while the DRAM‘s is limited. That is to say, in a hybrid main memory system, the size of the DRAM capacity space that can be used to store the replaced page group is limited.

If we fix the DRAM space capacity to N, and assume to migrate different amount with different substitution degree pages each time. There is a linear programming problem that under the condition of the space of DRAM is fixed, migrated the page groups combinations with the largest overall substitution degree and take up all DRAM space as much as possible.

$$\begin{aligned} &Max(X = \sum G_i) \\ &S.T.\ (\sum V_i \leq N) \end{aligned} \tag{9}$$

where V_i is the total size of page group G_i and N is the size of space in DRAM.

III. Consistency Improvement

A. Consistency Guarantee Method for EAPR

To guarantee the consistency of page replacement process, EAPR designed a writing transaction commit mechanism between user state and kernel state. A write transaction submit within NVM will generate interrelated logs, and EAPR will be informed that which pages have been changed. Any pages exchange between DRAM and NVM will prevent deleteing these logs, until they were writen back to NVM. Furthermore, EAPR will be informed that which pages are journal pages because the log pages can not be replaced to DRAM. In this way, user transactions can be recovered through journals even if the system failure occurs and the method of recovery can use the algorithm which is presented by paper [5].

B. Data Structure

For realizing consistency guarantee of hybrid memory systems, EAPR conducts three data structures in kernel state, called TILL, PFHT and PLIT. The data structures are shown in Fig. 2.

- **TILL** (*Transaction Information Linked List*): In this linked list, each node corresponds to a write transaction, and this transaction includes the pages which have been written and replaced to DRAM. Then the nodes of this linked list are arrayed by the order of write transaction committing.
- **PFHT** (*Hash Table with a Keyword of Page Frame Number*): As described, the keyword of hash table is the page frame number, and the value is the node position of write transaction in the linked list. Thus EAPR can find the write transaction information of a page in the linked list through this hash table.
- **PLIT** (*Page Location Information Table*): This table is a simple table which mainly preserved each page current location in the NVM or DRAM.

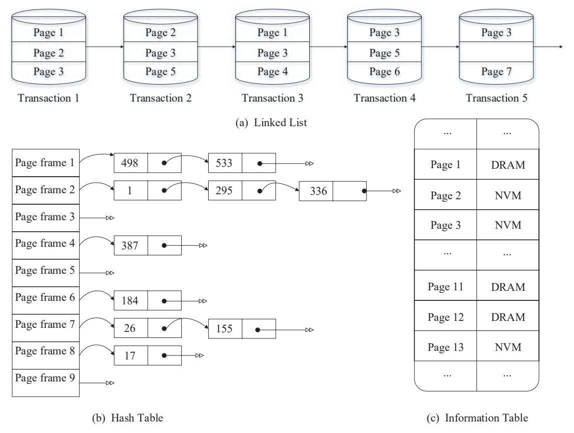


Fig. 2: Data Structure for Consistency Improvement

C. Optimization of Page Replacement

As the submission of user write transactions, the logs become larger and larger and the transaction information linked list becomes longer. These causes a problem that it will cost a lot of time to recover the system if system failures occurred. In order to solve this problem, we optimized the substitution degree for replacing the important pages from DRAM to NVM quickly. And the method for defining the important pages is shown as follows.

In the transaction information linked list, each transaction concludes many pages information, these pages are modified by the transaction and have been replaced to DRAM. Then we can conduct a graph of these pages based on their relevancy with other pages. The connectivity of each node in the graph is defined as R^N, and bigger R^N means that this page is more important. Then we give a coefficient to R^N and add them to the substitution degree of each page. As a result, the substitution degree has been optimized.

Definition 7: We use S_i^* to denote the optimized substitution degree of P_i which combines with consistency improvement process, and it is defined as:

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

where R^N is the connectivity of each node in the graph and λ is the coefficient of R^N. Furthermore, the determination of λ is required to go through the experiment, and in the experiment, by constantly changing the value of λ, the current λ value is the best lambda value when the system achieves maximum recovery efficiency.

IV. Page Replacement Method with Consistency Guarantee

In this section, we will describe our algorithmic in details. EAPR mainly contains two phases, ie., page replacement phase and transaction commit phase.

Page Replacement Algorithm

```
Input: $r_i$, $w_i$,$T$ of page $P_i$
begin
  if $P_i$ is in NVM then
    for $P_i$ do
      $S_i$
      j=i+1
      $S_j$
      if $S_i \approx S_j$ then
        $S_i^{group}$=$S_i$+$S_j$
      else then
        $S_i^{group}$=$S_i$
        $S_j^{group}$=$S_j$
      end if
    end for
    select $G_i$
    migrate $G_i$ to DRAM
    modify PLIT
  else if $P_i$ is in DRAM then
    for $P_i$ do
      $S_i^*$
      j=i+1
      $S_j^*$
      if $S_i^* \approx S_j^*$ then
        $S_i^{group*}$=$S_i^*$+$S_j^*$
      else then
        $S_i^{group*}$=$S_i^*$
        $S_j^{group*}$=$S_j^*$
      end if
    end for
    if $S_i^{group*} < E_{NVM}^{read} + E_{DRAM}^{read} + E_{NVM}^{write} + E_{DRAM}^{write}$ then
      migrate $G_i$ to NVM
      access PFHT then
        search TILL then
          delete $P_i$ from transaction
          if transaction is empty then
            delete transaction
          end if
        end search
      end access
      if TILI=$\emptyset$ then
        delete logs
      end if
    end if
  if DRAM is full then
    select $G_i$
    change $G_i$ at DRAM and NVM
  end if
end
```

A. Page Replacement Phase

At the beginning, all of the pages will be stored in NVM and given a attribute called substitution degree by using Equation 7. And then, the pages which have consecutive address and similar substitution degree are defined as a page group. Next, all of these page groups will be sorted according to their substitution degree from big to small by using Equation 8. And then, owing to the size of DRAM is fixed, EAPR must choose a set of page groups which have the biggest overall substitution degree and the largest DRAM space utilization. The constraint condition of this question is as the Equation 9 shown.

After a set of page groups been migrated to DRAM, EAPR will modify PLIT with these pages current location. In this process, when a page is written back to NVM from DRAM, EAPR will first access the PFHT to find all of the transactions which have been modified this page in TILL, and then this page will be deleted from these transactions. After this step, the transactions could be deleted from TILL if there is not any page information in this transaction. Finally, the logs will be deleted when TILL is empty.

B. Transaction Commit Phase

When user transactions are submitted, EAPR will be informed that which pages have been changed, and then EAPR will judge whether these pages are in the NVM based on PLIT. If these pages are all in the NVM, none of the user transaction will be added to TILL because there are not any pages of DRAM has be modified. At this time, if TILL is empty, the logs can be deleted from the main memory. On the other hand, if some of these pages are stored in the NVM, the information of the modified pages which are in the DRAM will be inserted to TILL.

实验配置：GEM5+NVMAIN
工作负载：SPEC CPU2006

V. Experiment

A. Experiment Setup

1) Page Replacement Method: For simulating realistic environment, we use a computer with Intel Core i5-4460 3.20GHz processor and 8GB memory. The operating system which we use is Ubuntu 16.04 LTS with a Linux kernel 4.4.0.

For evaluating the performance of our replacement algorithm, we use GEM5 combining with NVMAIN to simulate the hybrid memory system environment. The GEM5 simulator is a modular platform for computer-system architecture research, encompassing system-level architecture as well as processor micro-architecture [9]. And NVMAIN is a cycle accurate main memory simulator designed to simulate emerging non-volatile memories at the architectural level [10]. The detail parameters of GEM5 which we used are shown as Table I, and Table II shows that the parameters of NVM (PCM) and DRAM which we used [4]. We evaluate our replacement algorithm with six workloads which are selected from SPEC CPU 2006 v1.2 [11] and the footprints of the chosen benchmarks are shown as Table I.

The following schemes are evaluated and compared, RaPP [2], DMA [12] and noManage are chosen to compare with EAPR.

- **DMA** [12]: Dynamic measure and allocation algorithm;
- **RaPP** [2]: Rank-based page placement algorithm;
- **noManage**: NVM-DRAM hybrid memory system without page replacement method, the ratio of each page stored in NVM or DRAM in the system is proportional to the capacity of NVM and DRAM.

2) Consistency Guarantee Method: In order to test the consistency guarantee in EAPR, we implemented the EAPR algorithm on Linux 4.4.0. For the experimental environment, we use a computer with Intel Core i5-4460 3.20GHz processor and a hybrid memory including 8GB DRAM and 4GB NVDIMM. The operating system which we use is Ubuntu

GEM5设置：8GBDRAM +4GBNVDIMM
内核：Linux4.4

TABLE I: Configuration of GEM5 and Benchmarks

Processor	4 cores		
L1 Cache	32KB		
L2 Cache	6MB		
Memory Capacity	DRAM	bzip	140MB
		gcc	290MB
		sjeng	690MB
		xalancbmk	720MB
		gemsFDTD	831MB
		mcf	860MB
	NVM	4GB	

TABLE II: Parameters of PCM and DRAM

	PCM	DRAM
Write Energy	418.6nJ [8]	12.7nJ [7]
Read Energy	80.41nJ [8]	5.9nJ [7]
Leakage	4.23mW/GB [6]	451mW/GB [7]

16.04 LTS with a Linux kernel 4.4.0. Obviously, this paper will use NVDIMM to simulate NVM to test the consistency guarantee function of the EAPR algorithm.

In this consistency guarantee experiment, the EAPR algorithm is implemented as a simple character driver that directly manages the storage space of NVDIMM. EAPR will allocate a certain number of DRAM pages directly from the buddy system as the replacement space for the pages in NVM. In addition to the page replacement function, this experiment also implemented the mapping of the NVM page to the process address space, that is to say, the applications can apply the pages in NVM directly. EAPR also provides an I/O control interface for application calls, and these interfaces mainly include the log notification interface and the user transaction submission notification interface, etc. It is worth mentioning that in this paper, we need not to evaluate the application-level performance and the recovery performance of EAPR. This is because we only focus on the effect of system recovery after a system failure occurred. And after adding the consistency guarantee function, it also has little effect on the system operation efficiency.

In order to verify the effectiveness of EAPR, we have written a tool for consistency checking and simulated the power-off situation under the real system environment. And then, we use the consistency check tool to determine whether the system page is in an inconsistent state after the system has recovered.

B. Comparison of Energy Consumption

We run the six benchmarks for 100 trillion ticks in GEM5, and these benchmarks are under different size of DRAM. Then we compared our algorithm with DMA, RaPP and noManage. Fig. 4 depicts the proportion of energy consumption of EAPR, DMA and RaPP to noManage of these benchmarks.

From Fig. 4(a), the experimental results show that EAPR can reduce the energy consumption by 20%∼80% compared with noManage depending on the size of the DRAM space for application, and EAPR also can reduce the energy consumption by almost 20%∼40% compared with DMA because DMA only considers the number of writes in pages without thinking about the number of reads. Compared with RaPP, EAPR has a similar performance with RaPP and EAPR gets a better efficiency in some cases. The reason for this result is that EAPR can guarantee the consistency of the hybrid main memory system and this function has a little influence with the overall performance.

Compared to RaPP, Fig. 4 shows that the EAPR and RaPP have their own applicable scenarios, for instance, the energy consumption of EAPR is higher than RaPP in *bzip*, *sjeng* and *gemsDFTD*, and it is lower than RaPP in *gcc*, *xalancbmk* and *mcf*. The reason for this phenomenon is that RaPP migrated popular pages to the DRAM area, which has substantially better performance in the presence of row buffer misses than NVM. And *bizp*, *sjeng* and *gemsDFTD* all have a lot of repeated read and write at a certain area of main memory data, this feature of them is more suitable for RaPP. Furthermore, EAPR provides a function that it can guarantee the consistency of the hybrid memory system, it also influenced the energy consumption of EAPR.

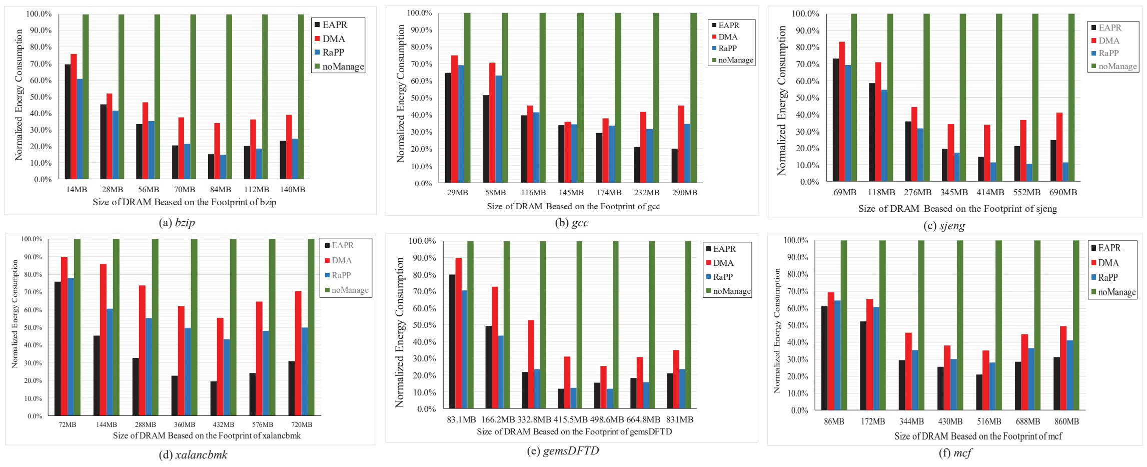
C. Comparison of Time Overhead

There are some extra overheads causing by EAPR, one of which is record overhead. Record overhead is to get the number of writes for our test programs, and the overhead of recording data is directly reflected in the time overhead. So we do some experiments for time operational efficiency of EAPR, DMA, RaPP and noManage, and the time operational efficiency is shown as Fig. 5. The results show that the time consumption of EAPR is around 25% more than noManage, around 20% more than DMA and 10% more than RaPP. Although EAPR has the highest overhead, EAPR is the only one algorithm which can provide consistency guarantee of the hybrid main memory system in these four methods. However, compared with the savings in energy consumption of memory accesses, EAPR has a relatively small time overhead.

D. Effectiveness of Consistency Guarantee

In this paper, according to the LSA algorithm given in paper [13], we implemented a series of user transactions, such as data access, modification and deletion. As well as the management of the data structure on the NVM, including the search, delete, insert and save operations on the linked list, tree and hash tables.

In this experiment, we have written a consistency check tool to verify the validity of the consistency guarantee function in EAPR. During the running period of those benchmarks and transactions, we simulate the occurrence of system failure through the power-off or other operations, and then the system will be reopened and restored by the method of NOVA. After the system has been restored, we use the consistency check tool to determine whether the application is in an inconsistent state. Experiment results show that EAPR can provide consistency guarantee for hybrid memory system and recover the system from a system failure state to the lasted stable state.



(a) *bzip* (b) *gcc* (c) *sjeng*

(d) *xalancbmk* (e) *gemsDFTD* (f) *mcf*

Fig. 3: Energy Comparison

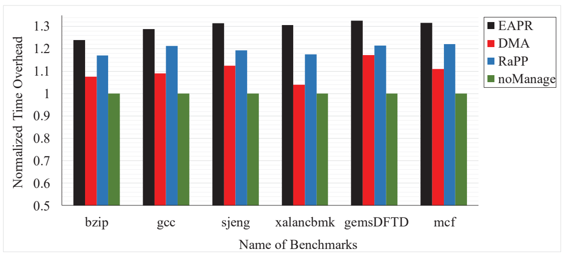


Fig. 4: Time Overhead Comparison

VI. Conclusion and Future Work

This paper presented EAPR, which is an energy-aware page replacement algorithm with strong consistency guarantee. In EAPR, every page has a substitution degree, which can be calculated by write times, read times and the storage time in DRAMs to measure the replacement energy consumption. The experimental results show that EAPR can save energy consumption by up to 80% comparing to traditional algorithms. Besides, our algorithm can also guarantee the consistency of data in NVM-DRAM hybrid memory system with acceptable time overhead.

In the future, we will optimize the substitution degree and apply our approach in some consumer electronics. Meanwhile, there are many outstanding research work, such as PCM-FTL [14] and [15], and we will also do more experiments to compare with them in the future.

VII. Acknowledge

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] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," *SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 2–13, Jun. 2009. [Online]. Available: http://doi.acm.org/10.1145/1555815.1555758

[2] L. E. Ramos, E. Gorbatov, and R. Bianchini, "Page placement in hybrid memory systems," ser. ICS '11. New York, NY, USA: ACM, 2011, pp. 85–95. [Online]. Available: http://doi.acm.org/10.1145/1995896.1995911

[3] Z. Zhang, L. Ju, and Z. Jia, "Unified DRAM and nvm hybrid buffer cache architecture for reducing journaling overhead," in *DATE 2016*, Mar. 2016, pp. 942–947.

[4] A. Hassan, H. Vandierendonck, and D. S. Nikolopoulos, "Energy-efficient hybrid DRAM/nvm main memory," in *PACT'15*, Oct. 2015, pp. 492–493.

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

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

[7] Micron, *Micron TN-41-01: Calculating memory system power*, 2014.

[8] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, Jul. 2012.

[9] 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, and D. A. Wood, "The gem5 Simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2024716.2024718

[10] M. Poremba, T. Zhang, and Y. Xie, "Nvmain 2.0: A user-friendly memory simulator to model (non-)volatile memory systems," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 140–143, Jul. 2015.

[11] J. L. Henning, "Spec cpu2006 benchmark descriptions," *Acm Sigarch Computer Architecture News*, vol. 34, no. 4, pp. 1–17, 2006.

[12] Y. Zhang, J. Zhan, J. Yang, J. Wei, L. Li, and L. Zhu, "Dynamic memory management for hybrid dram-nvm main memory systems," in *13th International Conference on Embedded Software and Systems*, 2016, pp. 148–153.

[13] P. Felber, C. Fetzer, P. Marlier, and T. Riegel, "Time-based software transactional memory," *IEEE Transactions on Parallel & Distributed Systems*, vol. 21, no. 12, pp. 1793–1807, 2010.

[14] D. Liu, K. Zhong, T. Wang, Y. Wang, Z. Shao, H. M. Sha, and J. Xue, "Durable address translation in pcm-based flash storage systems," *IEEE Transactions on Parallel & Distributed Systems*, vol. 28, no. 2, pp. 475–490, 2017.

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