2012 IEEE 18th International Conference on Parallel and Distributed Systems

OPAMP: Evaluation Framework for Optimal Page Allocation of Hybrid Main Memory Architecture

Jong-Hun Choi, Seong-Min Kim, Chulmin Kim, †Ki-Woomg Park and Kyu Ho Park
Computer Engineering Research Laboratory †Department of Computer Hacking and Information Security
Korea Advanced Institute of Science and Technology †Daejeon University
{jhchoi, smkim, cmkim}@core.kaist.ac.kr, and kpark@ee.kaist.ac.kr †woongbak@dju.kr

***Abstract*—Main memory as a hybrid between DRAM and non-volatile memory is rapidly considered as a basic building block of computing systems. Despite widely-performed researches no one can confirm whether hybrid memory is at its full performance in terms of energy consumption, time delay or both. The main problem is that evaluating their performance in comparison with the optimal performance is challenging since deriving the optimal value is NP-complete.**

In this paper, we design and implement an evaluation framework termed OPAMP, which calculates optimal performance of the hybrid memory environment. This system gathers workload, specification of DRAM and PRAM, and environmental parameters of the hybrid main memory. After that, it calculates the maximum performance under the corresponding conditions. We suggest the way of deriving the optimal value by profiling instead of page migration which is the mainstream of recent researches on hybrid main memory system. Also, proportion of DRAM's size to PRAM's and proportion of DRAM's usage space to PRAM's are impactive factors. While designing hybrid main memory, those two variables must be determined carefully and OPAMP gives the guideline to the researchers.

***Keywords*-PRAM; Energy Efficiency; Memory management**

I. Introduction

Dynamic random access memory (DRAM) has been located in the main memory of computer architecture for the last several decades. Recently, the position of DRAM is being threatened by limitation of DRAM scaling and its energy consumption [1]. In terms of power consumption, DRAM dominates a typical server system, accounting to 30-40% [2]. In future systems, as server system requires hundreds of GBs main memory, the memory will take a large part of the power consumption [3].

As a remedy for this problem, new memory technologies, such as Phase Change RAM (PRAM), Ferroelectric RAM (FRAM), and Magnetic RAM (MRAM) have been suggested; these can provide more memory capacity and less energy consumption than DRAM [1], [4], [5]. Among these memories, PRAM is the closest to being on sale [6]–[8].

The problem faced with PRAM is that it cannot entirely substitute DRAM in the main memory of computer architecture. This is because of its obstructive access latency, active energy, and low write endurance for each cell compared to DRAM. To solve this problem, many recent researchers suggest memory management policies enhanced with a new type of main memory called hybrid main memory which consists of both DRAM and PRAM [4], [9]–[12]. Basic concept of this architecture is to leverage the attractive attributes of both DRAM and PRAM while mitigating negative effects of both memories. Despite of those researches, there still remain several issues as follows.

Although many researches mentioned above have presented the hybrid memory management policies, they have following insufficiency.

Determination of the best performance of hybrid memory system: Researches about the hybrid memory architecture usually compare its performance with the main memory composed of DRAM only, hereinafter we will refer to it as the "DRAM-only memory" [4], [9]–[12]. They possibly show that the hybrid architecture has improved performance, but it is hard to say that proposed memory management policies operate at the maximum performance of the architecture. We can solve this problem by finding the maximum performance of the hybrid architecture.

Operation value of DRAM and PRAM: Above mentioned researches set its own hybrid memory environment, such as size of PRAM and DRAM, usable space of DRAM for each task, and acceptance of page migration between PRAM and DRAM to verify its memory management policy for different purposes. Those purposes are minimizing energy, time, energy delay product or the number of write on PRAM. This makes system designer harder to decide the suitable hybrid architecture with what kind of parameters. We can solve this problem by setting various parameters of the hybrid memory architecture and obtaining the maximum performance of several purposes.

In this work, we design and implement an evaluation framework termed OPAMP, which calculates optimal performance of the hybrid memory environment. This system gathers workload, specifications of PRAM and DRAM, and environmental parameters of the hybrid main memory from the user input and calculates the maximum performance under the corresponding conditions. We design OPAMP using pin tool [13] and MATLAB R2010a and this implementation can solve problems we mentioned above. Consequently, OPAMP can act as the key primitive for guideline of hybrid main memory. Specifically, OPAMP contains a potential usefulness and additional extensibility as follows:

OPAMP vs. Previous Works: We calculate the optimal minimizing energy using OPAMP in the same environment of past researches. Then, we use this data to evaluate the effectiveness of the past researches. This process provides the optimal value of each environment, just like the *Carnot*

1521-9097/12 $26.00 © 2012 IEEE
DOI 10.1109/ICPADS.2012.89

efficiency in thermodynamics [14] (Section IV-B).

Various Settings for the Hybrid Memory System: We design the experiment to find the influence of each parameter in the hybrid system (such as size of PRAM and DRAM, usage space of DRAM and so on) by changing those parameters in the same architecture of related works. From this, we conclude that the suitable parameters vary depending on the system characteristics (Section IV-C). For instance, the following cases show the usability of OPAMP:

• *Case 1 - Acceptance of page migrations between PRAM and DRAM:* By calculating the optimal performance of static and dynamic allocation, we conclude that the difference between above cases is less than 3%. This proves that the first well-allocated space is almost effective scheme compared to scheme which uses migration.

• *Case 2 - Ratio of PRAM and DRAM's size in the hybrid main memory:* We calculate the optimal values by changing the ratio of DRAM and PRAM for each workload. From this, it is hard to say that the energy saving is always assured in the decrease of DRAM size, at certain workload.

• *Case 3 - Limitation of DRAM's usage space:* After the ratio of PRAM and DRAM's size is determined, we calculate the optimal values while changing the DRAM usage space that the workload uses. From this, we know that in order to maximize the optimal values, DRAM usage space can vary for the characteristic of workload because PRAM has a large active power.

• *Case 4 - Various Objectives:* Performance and energy saving are important values in the hybrid memory system. Generally, to calculate the optimal values considering these two factors simultaneously, we use the concept of Energy-Delay product. Also, to reflect write endurance of PRAM, we propose the method minimizing the number of write operations on PRAM. We confirm that minimizing the number of write operation in PRAM make it as the near optimal energy saving of hybrid main memory architecture.

II. Problem Analysis

A. Background

1) Characteristics of PRAM: PRAM cells store data permanently and it sustains data even the power is off. Therefore, PRAM consumes no refresh power and lower leakage power than DRAM. Read and write latency of PRAM is longer than DRAM and these values are asynchronous. Also, the energy consumption is large while doing read and write operation. Finally, because of the nature of PRAM where the element is heated to change phase of material, there exists limitation on lifetime in the cell.

Energy consumption of DRAM and PRAM can be divided into three categories: background energy, activate energy and read/write energy [15]. Background energy is the energy to sustain the accessibility of each memory. Activate energy is consumed while the memory loads the data from the memory row to I/O buffer. Finally, read/write energy is the energy to read/write data which is stored in I/O buffer.

B. Challenging Key Issue

While allocating pages in the hybrid memory system, there are two methods which are static allocation and dynamic allocation. The difference between two methods is the acceptance of page migration between DRAM and PRAM. While finding optimal value of system's performance, consideration of page migration is treated as a challenging problem since the general memory allocation problem is NP-complete and static allocation is also complex enough [16], [17]. Because of the above mentioned reasons, previous researches focused on the static allocation to get the optimal values. To find the exact optimal value of hybrid memory, consideration of the dynamic case is needed and it is very challenging issue as we mentioned before. To solve this problem, we design a method which calculate the optimal value of dynamic allocation case.

C. Our Approach

We make three assumptions to calculate the optimal performance of the hybrid memory. First, we assume that the size of main memory is large enough. When processes try to access a page, the page must be always existed in the memory system. It can make us purely focus on the memory system's performance without overhead caused by the storage system. Time delay and energy consumption of the memory system must only be bounded by the latency and energy of PRAM and DRAM.

Second, CPU can access both DRAM and PRAM in a unified manner. There are some researches which propose architecture where CPU can only access to the DRAM. Thus, DRAM is used as the buffer of PRAM [4]. Our work can be applied to this architecture and we remain it as a further work. In this paper, we concentrate on the architecture where CPU can access to both DRAM and PRAM directly.

Finally, we do not consider memory I/O buffer hit in this paper. Since PRAM related researches induce the PRAM as the memory which has same structure with DRAM, two memories have similar latency and energy value when the memory accesses hit the I/O buffer. According to our experiments, the difference whether considering the I/O buffer hit or not is within 1%. Also, calculator module of OPAMP which consider I/O buffer hit is enormously complex and has three times longer calculation time.

III. Evaluation Framework

A. Design of OPAMP

Working flow of OPAMP has three phases; input, calculation, and result analysis. Figure 1 shows the outlook of OPAMP.

Phase1. Extracting memory access trace for a certain target application: By using the parameters of target environment and workload of users which are entered by them, OPAMP gathers the memory access trace in the DRAM-only system using pin-tool [13]. Each memory trace shows the page address and the type of operation whether it is read or write.

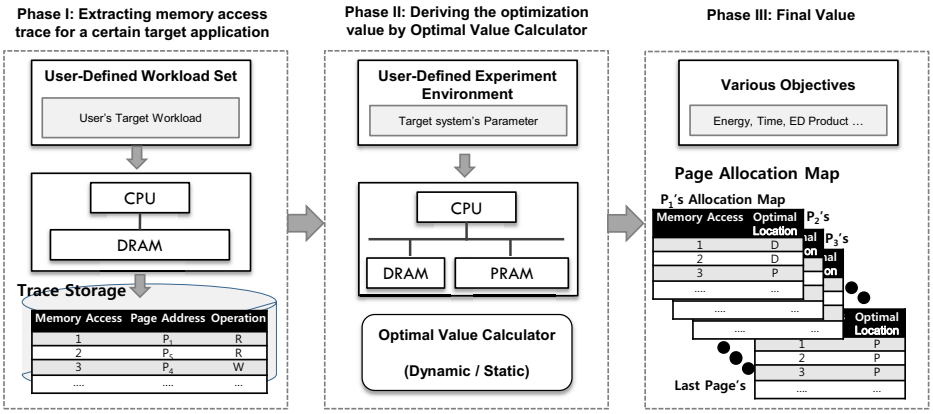

Fig. 1. Working Sequence of OPAMP

Phase2. Deriving the optimization value by Optimal Value Calculator: By using memory access trace and environmental variables, OPAMP calculates the several optimal values of memory. OPAMP uses the integer linear programming (ILP) to get the optimal values. ILP is the general method to get the optimal value in the NP-complete problems [16]–[18]. We construct the ILP formulation for hybrid memory system by using the parameters defined in Section III-B1.

Phase 3. Result analysis: It is possible to analyze the result of optimal values which are calculated by the above phases. To consider the overhead caused by wear-leveling, we add the average overhead which is proposed by [4] to our final result. OPAMP calculates the optimal energy E and latency T for the several purposes such as minimizing energy, time, energy delay product, or write operations on PRAM. Phase3 draws page allocation maps which minimize the above values, respectively. The page allocation map represents whole page's optimal location, DRAM or PRAM, for each memory accesses.

B. Optimal Value Calculator

1) General Parameters: Let us decide notations to express various hybrid memory's parameters to use ILP formulation. Each memory, DRAM and PRAM, has $\{T_r, T_w, E_r, E_w, S, B\}$ as a feature. T_r and T_w are respectively read and write latency of corresponding memory. E_r is the energy consumed by read operation and E_w is the energy consumed by write operation. S(D) is the available space that DRAM can use. S(P) has the same meaning for PRAM and its unit is page. For example, in the case of 4MB space in DRAM, S(D) is 1024 since the page size is 4KB. B(D) is the background energy of DRAM and B(P) is that of PRAM. Given a page set P, each page p_i is characterized as $N_r(p_i)$ and $N_w(p_i)$ where the number of page is m. $N_r(p_i)$ and $N_w(p_i)$ are the total number of times which p_i is read and written by cpu, respectively.

Related notations are arranged in Table 1. Also, the system needs several variables which can express the total energy consumption and execution time of main memory. Case of DRAM-only, E_D denotes the energy and T_D denotes the time

TABLE I
NOTATIONS OF GENERAL TERMS

Description	Notation
Set of Memory	{D, P}
Set of Page	P=$\{p_1, p_2, \ldots, p_m\}$
Number of pages	m
Number of times p_i is read	$N_r(p_i)$
Number of times p_i is write	$N_w(p_i)$
Latency to read DRAM, PRAM	$T_r(D)$, $T_r(P)$
Latency to write DRAM, PRAM	$T_w(D)$, $T_w(P)$
Energy consumption of read operation each DRAM, PRAM	$E_r(D)$, $E_r(P)$
Energy consumption of write operation each DRAM, PRAM	$E_w(D)$, $E_w(P)$
Size of DRAM, PRAM's free space	$S(D)$, $S(P)$ unit : page
Background Power of DRAM, PRAM	$B(D)$, $B(P)$

which are consumed while the workload of user is executed. In the hybrid memory, E_s and T_s denote the energy and time for the static allocation case. Similarly, E_{Dy} and T_{Dy} denote the energy and time for dynamic allocation case. For these two cases, minimal values for the objective equation will be denoted as O^{st} and O^{dy}.

2) Static Allocation: While the user's target workload is executed, the allocated memory of each page remains unchanged.

a) General Constraints: Each page p_i can only be assigned to one memory in whole memory system. We define $f(p_i)$ to express this,

$$f(p_i) = \begin{cases} 1 & \text{if } p_i \text{ is assigned to DRAM,} \\ 0 & \text{if } p_i \text{ is assigned to PRAM.} \end{cases} \quad (1)$$

When assigning pages to each memory, it cannot be allocated exceeding the memory's empty space. To guarantee this, we add a following inequality as conditions,

$$\sum_{i=1}^{m} f(p_i) \leq S(D). \quad (2)$$

For the proposed three cases, additional constraints should be considered.

b) Minimizing Energy Consumption or Time: The purpose is to find the each page's optimal fixed location which

minimizes energy consumption or execution time of target workload in the hybrid memory architecture. Total execution time of user's workload is expressed as

$$\begin{aligned} T_s &= T_D + \underbrace{\sum_{i=1}^{m}(1-f(p_i))\{N_r(p_i)T_r(P)+N_w(p_i)T_w(P)\}}_{\text{Total execution time of hybrid memory's PRAM}} \\ &+ \underbrace{\sum_{i=1}^{m} f(p_i)\{N_r(p_i)T_r(D)+N_w(p_i)T_w(D)\}}_{\text{Total execution time of hybrid memory's DRAM}} \\ &- \underbrace{\sum_{i=1}^{m}\{N_r(p_i)T_r(D)+N_w(p_i)T_w(D)\}}_{\text{Total execution time of DRAM-only}}. \end{aligned} \quad (3)$$

Let the total energy consumption be E_S,

$$\begin{aligned} E_s &= \sum_{i=1}^{m} f(p_i)\{N_r(p_i)E_r(D)+N_w(p_i)E_w(D)\} \\ &+ \sum_{i=1}^{m}(1-f(p_i))\{N_r(p_i)E_r(P)+N_w(p_i)E_w(P)\} \\ &+ T_s(B(D)+B(P)), \end{aligned} \quad (4)$$

where T_s is given by (3). Equation (3) is a linear function and (4) is sum of two linear functions which are solved by ILP. Above two formulations become objective equation of each case. While minimizing execution time of target workload, objective equation is

$$O_T^{st} = \min(T_s), \quad (5)$$

where T_s is given by (3). While minimizing energy consumption, objective equation is

$$O_E^{st} = \min(E_s), \quad (6)$$

where E_s is given by (4).

c) Minimizing Energy Delay Product: Equation (3) and (4) are linear equations. By multiplying those two functions to express energy delay product, it cannot be applied to ILP. To settle this, we put the energy term into constraints and we set the time term as objective equation. To put the energy term into constraints, it needs a threshold value. It can be solved by setting a limitation for energy consumption as a certain proportion of E_D. Added constraint is

$$E_s < E_D \times \alpha, \quad \alpha \in [0,1] \quad (7)$$

where E_s is given by (4). The Objective is

$$O_{E\cdot D}^{st} = \min(T_s) \qquad \text{under (7)},$$

where T_s is from (3).

d) Minimizing Write Operations in PRAM: To minimize number of writes on PRAM, the objective is

$$O_{PRAMwrite}^{st} = \min \sum_{i=1}^{m}(1-f(p_i))N_w(p_i). \quad (8)$$

3) Dynamic Allocation: For ILP formulation of dynamic page allocation, alternative analyzing method is needed. Let us define A as a set of all memory accesses. A_k denotes k-th memory access. It can be expressed as follows,

$$A = \{A_1, A_2, \ldots, A_l\}, \quad (9)$$

where l is the total number of memory accesses. A_k has features $\{P(A_K), R(A_k)\}$. $P(A_k)$ is the page address which is accessed at A_k and $R(A_k)$ shows whether A_k's operation is read or write.

$$R(A_k) = \begin{cases} 1, & \text{if } A_k\text{'s operation is read,} \\ 0, & \text{if } A_k\text{'s operation is write.} \end{cases} \quad (10)$$

a) General Constraint: p_i must be allocated to one memory when A_k is occurred,

$$f_k(p_i) = \begin{cases} 1, & \text{if } p_i \text{ is on DRAM when k-th access,} \\ 0, & \text{if } p_i \text{ is on PRAM when k-th access.} \end{cases} \quad (11)$$

To consider the limitation of DRAM's usage space,

$$\sum_{i=1}^{m} f_k(p_i) \le S(D), \quad \forall k \in \{1,2,\ldots,l\}. \quad (12)$$

We set variables to decide whether migration has occurred or not. While a page migrates from DRAM to PRAM, it is needed to read the page from DRAM and write it to PRAM. The latency and energy overhead is $C(T_r(D)+T_w(P))$ and $C(E_r(D)+E_w(P))$. where C is the value of page size divided by the size of memory row. If the page size is 4KB and the size of row is 512 bits, case of migration from PRAM to DRAM, the latency and energy overhead is $8(T_r(P)+T_w(D))$ and $8(E_r(P)+E_w(D))$.

To calculate overhead caused by migrations, the number of migrations is needed. This can be induced by finding case when $(f_k(p_i), f_{k+1}(p_i))$ is (0,1) or (1,0). However, it is hard to express the total number of (0,1) and (1,0) by ILP formulation for this case. As a result, in Figure 2, we change the case to finding (1,1). For this, we suggest function $g_k(p_i)$ and assume that $f_0(p_i)$, $f_{l+1}(p_i)$ is 1.

$$g_k(p_i) = f_k(p_i)f_{k+1}(p_i), \quad \forall k \in \{0,1,\ldots,l\}. \quad (13)$$

Those are designed to find $(f_k(p_i), f_{k+1}(p_i))$=(1,1) cases. Equation (13) is not a non-linear form. So it must be transformed into linear equation form.

$$g_k(p_i) \le (f_k(p_i)+f_{k+1}(p_i))/2, \quad \forall k \in [0,l] \quad (14)$$

$$g_k(p_i) \ge (f_k(p_i)+f_{k+1}(p_i)) - 1. \quad \forall k \in [0,l] \quad (15)$$

Equation (14) and (17) are suitable constraints of ILP formulation. Figure 2 describes the way to trace changes of page's location and the total number of migrations. The total number of migrations in DRAM can be induced from the total number of (1,1) because the total number of 1 s at DRAM can be calculated easily. [1]

The total number of 1 s in p_3's allocation map is $\sum_{k=0}^{l+1} f_k(p_3)$. The total number of (1,1) in p_3's allocation map is $\sum_{k=0}^{l} g_k(p_3)$. From above two equations and Figure 2, $2\sum_{i=1}^{m}(\sum_{k=0}^{l+1} f_k(p_i) - \sum_{k=0}^{l} g_k(p_i) - 1)$ can be induced, However, above equation contains additional migrations which are caused by attaching 1s of both sides. Except these inaccurate migrations, the number of total page migration is

$$M_N = 2\sum_{i=1}^{m}(\sum_{k=0}^{l+1} f_k(p_i) - \sum_{k=0}^{l} g_k(p_i) - 1) - (2l - \sum_{i=1}^{m} f_0(p_i) - \sum_{i=1}^{m} f_{l+1}(p_i)). \quad (16)$$

M_N is the number of total page migration. Let T_o and E_o be the total migration overhead of time and energy. Then, the total latency overhead caused by total migrations is

$$T_o = C(T_r(D)+T_w(D)+T_r(P)+T_w(P)) * M_N/2 \quad (17)$$

The total energy consumption overhead caused by total migrations is

$$E_o = C(E_r(D)+E_w(D)+E_r(P)+E_w(P)) * M_N/2 \quad (18)$$

Memory Trace

A	1	2	3	4	5	...	l-3	l-2	l-1	l
P(A)	p_3	p_3	p_3	p_2	p_2	…	p_2	p_2	p_2	p_3
R(A)	R	W	R	W	R	…	R	R	W	W

p_3's allocation map.

k	0	1	2	3	4	...	l-3	l-2	l-1	l	l+1
P/D	1	1	1	0	0	…	0	1	1	1	1

Let us Define k = 0, l+1→ $f_k(p_i)$=1 $\forall i \in [1, m]$,
Then, number of (1,0) = number of (0,1)

(1,0) : Overhead = DRAM Read + PRAM write
1 0 or **0 1** : Migration
(0,1) : Overhead = DRAM Write + PRAM read

Total number of migrations
= Total number of **1 0** & **0 1** ,
2*Total number of (1) -2
=Total number of (1,0) and (0,1) + 2*Total number of (1,1).
($\because$ *if* $k = 0$ *or* $l+1$, (1) *is counted once*)

Total number of migrations
= 2*{Total number of (1) - Total number of (1,1)-1}.

Fig. 2. How to find the number of migrations

The function of total time is given by

$$T_{Dy} = \underbrace{\sum_{k=1}^{l} f_k(P(A_k))\{R(A_k)T_r(D) + (1 - R(A_k))T_w(D)\}}_{\text{Total execution time of hybrid memory's DRAM}}$$
$$+ \underbrace{\sum_{k=1}^{l} (1 - f_k(P(A_k)))\{R(A_k)T_r(P) + (1 - R(A_k))T_w(P)\}}_{\text{Total execution time of hybrid memory's PRAM}}$$
$$- \underbrace{\sum_{k=1}^{l} (P(A_k))\{R(A_k)T_r(D) + (1 - R(A_k))T_w(D)\}}_{\text{Total execution time of DRAM-only}}$$
$$+ \underbrace{T_o}_{\text{Total migration overhead}} + T_D, \quad (19)$$

where T_o is given by (17). Also, function of total energy consumption is given by

$$E_{Dy} = \sum_{k=1}^{l} f_k(P(A_k))\{R(A_k)E_r(D) + (1 - R(A_k))E_w(D)\}$$
$$+ \sum_{k=1}^{l} (1 - f_k(P(A_k)))\{R(A_k)E_r(P) + (1 - R(A_k))E_w(P)\}$$
$$+ E_o + T_{Dy}(B(D) + B(P)), \quad (20)$$

where E_o is given by (18), and T_{Dy} comes from (19).

b) Minimizing Energy Consumption and Time: While minimizing Energy consumption, the objective is

$$O_E^{dy} = \min(E_{Dy}), \quad (21)$$

where E_{Dy} is defined at (20). While minimizing total needed time, the objective is

$$O_T^{dy} = \min(T_{Dy}), \quad (22)$$

where T_{Dy} is given by (19).

c) Minimizing Energy Delay Product: For the same reason in Section III-B2, we put the energy term into constraints. So we set a limitation for the energy consumption as a certain proportion of E_D. Added constraint is,

$$E_{Dy} < E_D \times \beta. \quad \beta \in [0, 1] \quad (23)$$

The objective is $O_{E \cdot D}^{dy}$=min(T_{Dy}), T_{Dy} is given by (19).

d) Minimizing Write Operations in PRAM: The objective is

$$O_{PRAMwrite}^{dy} = \min(\sum_{k=1}^{l} (1 - f_k(P(A_k))) \times (1 - R(A_k))), \quad (24)$$

where the first term's value is 1 when $P(A_k)$ is allocated in PRAM and the second term can be 1 when k-th memory operation is write.

IV. Case Study

A. Experimental Method

We utilize pin-tool to execute SPECCPU2000 benchmark to get traces of memory access. We use these traces as inputs of phase2 in OPAMP. Several variables (set of memory accesses A, set of pages P, N_r, N_w) of 0/1 ILP formulation in OPAMP are established by the traces of memory access. Other variables (T_r, T_w, E_r, E_w) which are defined in Section III-B1 comes from user's input. In this experiment, we use specification of [9] and those values are expressed in Table 2. We get T_D, E_D (Section III-B1) from the benchmark's execution results. Memory read access per instruction (RPI) and memory write access per instruction (WPI) of each benchmark are denoted in Table III.

B. OPAMP vs. Previous Work

To compare the energy saving, we setup the same environment of [9] where the size of hybrid memory as 1GB DRAM and 3GB PRAM and the cache size as 64KB of data and instruction L1 caches and 4MB of L2 cache. Firstly, we set the allocation of pages in DRAM as 25% from the total pages of workload. In this environment, we get the dynamic page allocation which minimizes the energy consumption of the hybrid memory system. We get the result by the objective equation (25) and constraint equation (12) in Section . From this allocation, we calculate the energy consumption by using (24). *Optimal approach rate* is calculated through dividing the previous work's energy consumption by optimal energy consumption. Like *Carnot efficiency* in thermodynamics [14], *optimal approach rate* gives whether additional improvement

TABLE II
Specifications of Target system

Parameter	DRAM	PRAM
Row read power	210 mW	78 mW
Row write power	195 mW	773 mW
Act Power	75 mW	25 mW
Standby Power	90 mW	45 mW
Refresh Power	4 mW	0 mW
Initial row read latency	15 ns	28 ns
Row write latency	22 ns	150 ns
Same row read/write latency	15 ns	15 ns

TABLE III
CHARACTERISTICS OF BENCHMARKS

Benchmark	RPI	WPI	Write ratio
applu	0.49%	0.37%	43.1%
gcc	0.03%	0.02%	34.3%
mcf	1.76%	0.09%	5.2%
mgrid	0.18%	0.03%	16.7%
gzip	0.02%	0.02%	55.1%
swim	1.83%	0.63%	25.3%

RPI: Memory read access per instruction, WPI: Memory write access per instruction, Write ratio: WPI/(RPI+WPI)

TABLE IV
ENERGY CONSUMPTION COMPARISON BETWEEN PREVIOUS WORK [9] AND OPAMP

	Previous work [9]	OPAMP	Optimal approach rate
Average	72%	64%	88.8%
Applu(Worst)	80%	68%	85.0%
gcc(Best)	65%	63%	96.9%

of previous work's memory management policy is possible or not.

In Table IV, it shows the energy consumptions of [9] and OPAMP which are normalized by that of DRAM-only memory. We get the values of four benchmarks, *applu*, *bzip2*, *facerec*, and *gcc*. Table IV shows the values of *gcc* which has the best *optimal approach rate*, applu which has the worst *optimal approach rate*, and average of four benchmarks. Case of [9], its memory controller allocates every page in PRAM firstly. If the write operation on the page of PRAM becomes larger than the threshold, controller moves the certain page to DRAM. Therefore, *applu* which has a lot of write operation has the worst *optimal approach rate* and *gcc* which has little write operation has the best *optimal approach rate*.

Since it is hard to know the specific DRAM usage ratio of the [9] while pages are allocated, we calculate the optimal energy consumption for the various DRAM's usage spaces.

C. Various Settings for the Hybrid Memory System

1) Acceptance of Page Migrations between PRAM and DRAM: To check the effect of page migration, we get the optimal page allocation for static and dynamic allocation cases under the same environment. Firstly, we draw the equation $E^{\alpha}T^{(1-\alpha)}$ which considers energy and time simultaneously. E and T are the energy consumption and execution time which are normalized by that of DRAM-only memory system. α is a parameter which can take any value between 0 and 1. By choosing the value of alpha, system designers can change the emphatic factor of the system.

In Figure 3, three curves are closely drawn. We get the average optimal values of all benchmarks for three cases which are minimizing energy, time with dynamic allocation and minimizing energy with static allocation, respectively. From the figure, two results using dynamic allocation have only 3% differences on energy and time term compared to that of static allocation. It means that the first appropriate allocation of pages gives similar results with page migration alternatively.

Still, current researches are more focusing on the page migration as a main factor to reduce energy consumption.

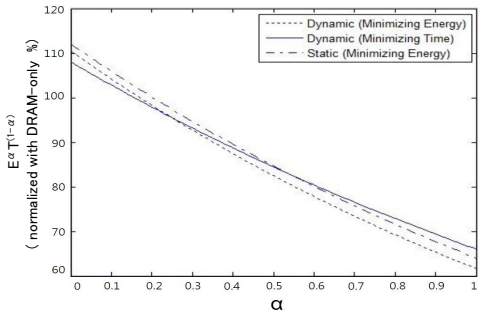

Fig. 3. Dynamic and Static optimal value's $E^{\alpha}T^{1-\alpha}$

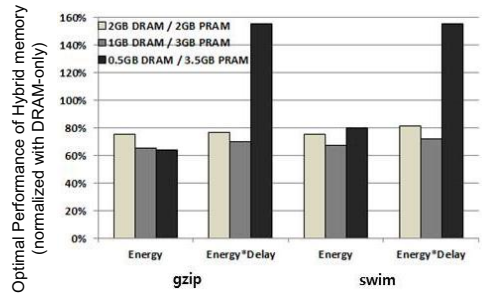

Fig. 4. Energy consumption and Energy delay product for various size of DRAM and PRAM

However, from this result, if profiling is possible (if the characteristics of pages are predictable), the system performs the near optimal performance by setting initial page allocation properly.

2) Ratio of PRAM and DRAM's Size in the Hybrid Main Memory: Figure 4 shows the calculated optimal energy consumption by finding the optimal page allocation which minimize the energy consumption of hybrid memory for two benchmarks, *gzip* and *swim*. We also calculate the product of the energy consumption and their execution time. The y-axis values are normalized by comparing certain calculated values with that of DRAM-only memory. The energy is minimized by (6) and the time is minimized by (3). For the various size combination of DRAM and PRAM, DRAM usage ratio is set as the size ratio of DRAM and PRAM. For instance, if the size of DRAM is 0.5GB and size of PRAM is 3.5GB, the usage ratio of DRAM is 12.5%. For the both cases, 1GB DRAM with 3GB PRAM hybrid memory system has lower optimal energy consumption and energy delay product than that of 2GB DRAM with 2GB PRAM hybrid memory system. By decreasing the portion of DRAM in hybrid memory system, effect of the background power of DRAM is decreased. However, it is impossible to say that decreasing portion of DRAM in hybrid memory system can improve the energy consumption and energy delay product. For *gzip*, the result shows that 0.5GB DRAM with 3.5GB PRAM hybrid memory system has lower energy consumption but definitely high energy delay product than that of 1GB DRAM with 3GB PRAM hybrid memory. Even though optimal energy consumption can be decreased by reducing the portion of DRAM in hybrid memory, energy delay product becomes high because of the performance overhead due to the lack of

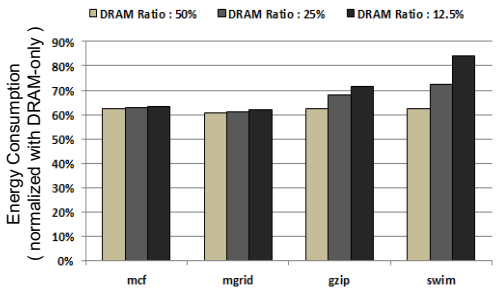

Fig. 5. Energy consumption under the limitation of DRAM's usage space

DRAM memory space. Case of *swim* which has high memory access operation per instruction, both energy and energy delay product become high.

3) Limitation of DRAM's Usage Space: We change the ratio of PRAM usage space and DRAM usage space as 1:1, 3:1, and 7:1 while we fix the size of DRAM as 1GB and PRAM as 3GB. We use the 0/1 ILP formulation of static allocation to calculate those values.

Figure 5 shows that the energy consumption of hybrid memory compared to DRAM-only memory increases from high DRAM usage ratio limitation (12.5%) to low limitation (50%) for the four benchmarks, *mcf*, *mgrid*, *gzip*, and *swim*. The result gives us that we get higher optimal energy value by increasing the limitation of DRAM usage in the hybrid memory. The change of optimal energy consumption caused by various DRAM usage limitations is small for both of *mcf* and *mgrid* which have many read operation than write operation. Since energy consumption during write operation is large for PRAM, the number of write operation is dominant term for the limitation of DRAM's space. The case of *gzip* and *swim* which have many write operation compared to previous two benchmarks show that the optimal energy consumption values are evidently changed by the limitation of DRAM's space.

4) Various Objectives: In this section, the size of DRAM and PRAM are set as 1GB and 3GB.

- ***Energy Delay Product:*** Table V shows the comparison between energy optimization case and energy delay product optimization case for *mgrid*. We can get the energy delay product optimal values by calculating optimal latency under the limited usage space of DRAM and energy consumption. We restrict the energy consumption of hybrid memory as 65% of DRAM-Only case and ratio of PRAM usage space and DRAM usage space as 3:1. The percentages are calculated by comparison between optimal values of hybrid memory and that of DRAM-only memory. Those values are calculated by the formulation of energy delay product in Section III-B2.
- ***Minimize PRAM Write vs. Maximize Energy Saving:***

TABLE V
ENERGY, DELAY, AND ENERGY DELAY PRODUCT UNDER ENERGY LIMITATION (NORMALIZED WITH DRAM-ONLY)

	Energy Optimization	E-D Product Optimization
Energy	61%	63%
Latency	111%	106%
E-D Product	68%	67%

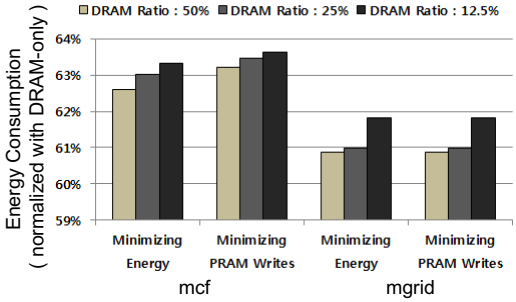

Fig. 6. Energy consumption when minimizing energy and PRAM writes

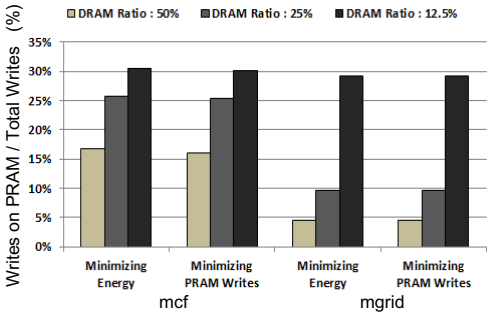

Fig. 7. Write on PRAM ratio when minimizing energy and PRAM writes

One of the most significant issues of PRAM is its poor write endurance. To consider the write endurance of PRAM, we find the minimum value of the number of PRAM's write operation when usage space of DRAM is limited. Those values are calculated by (25) and (28). Figure 6 shows the energy consumption for two cases of optimization which minimize energy consumption and minimize the number of write operations on PRAM compared to DRAM-only memory for two benchmarks, *mcf* and *mgrid*. Also, Figure 7 shows the ratio of number of write operation on PRAM in total write operations for the same cases. Both experimental results present that minimizing the number of write operation on PRAM case has almost near energy consumption and similar number of write operation on PRAM compared to minimizing energy case. Since the active energy of PRAM for write operation is dominantly high, both minimizing cases show similar result.

V. RELATED WORK

Research on finding optimal resource allocation using ILP is on progress. References [16]–[18] studied static optimal scheme that allocates variables using ILP in embedded system that uses scratch-pad memory which has similar properties with hybrid main memory. In this field, references [16]–[18] studied optimal memory allocation strategies from the result of first running and after that they adapt the strategies to their systems. In hybrid architecture using PRAM, references [19], [20] studied optimal allocation by using ILP. Reference [19] solved allocation problem of variable in DSP system. Another research [20] targeted optimal task allocation on hybrid main memory. Above researches are only progressed on static allocations. Meanwhile, other researches are progressing on finding ways to make hybrid main memory using PRAM

more competitive in real [4], [5]. To maximize power reducing, Park et al. [10] partially refreshed DRAM. Lee et al. [12], [21] design memory management policies based on reference bits of memory. To materialize hybrid main memory in real, there needs researches on how to adjust OS and hardware. Reference [5] had set system managing policies and designed memory controller for this. Other researches [21], [22] had adjusted OS to support the hybrid memory.

VI. Conclusion

In this paper, we propose an evaluation framework, OPAMP, which calculates the performance limitation of hybrid main memory architecture. From the result, well-designed hybrid memory decreases energy delay product over 20% on average compared to DRAM-only system while latency delay increases 12%. This means that hybrid memory architecture will be a future main memory system.

We conclude that the hybrid main memory system shows near optimal performance without page migration if it is possible to predict the behavior of each page whether it is hot or cold. Also, proportion of DRAM's size to PRAM's and proportion of DRAM's useful space to PRAM's are impactive factors. While designing hybrid main memory, those two variables must be determined carefully and OPAMP gives the guideline to the researchers. Finally, our dynamic ILP formulation can be applied to obtain real-optimal values in other heterogeneous memory systems, such as scratch-pad memory on embedded architecture.

References

[1] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," vol. 37. New York, NY, USA: ACM, June 2009, pp. 2–13. [Online]. Available: http://doi.acm.org/10.1145/1555815.1555758

[2] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. W. Keller, "Energy management for commercial servers," vol. 36. Los Alamitos, CA, USA: IEEE Computer Society Press, December 2003, pp. 39–48. [Online]. Available: http://dx.doi.org/10.1109/MC.2003.1250880

[3] "http://www.intalio.com/cloud-computing-is-memory-bound."

[4] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," vol. 37. New York, NY, USA: ACM, June 2009, pp. 24–33. [Online]. Available: http://doi.acm.org/10.1145/1555815.1555760

[5] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," vol. 37. New York, NY, USA: ACM, June 2009, pp. 14–23. [Online]. Available: http://doi.acm.org/10.1145/1555815.1555759

[6] "http://news.techeye.net/chips/samsung-throws-its-toys-into-pram."

[7] S. Kang, "A 0.1/spl mu/m 1.8v 256mb 66mhz synchronous burst pram," in *Solid-State Circuits Conference, 2006. ISSCC 2006. Digest of Technical Papers. IEEE International*, feb. 2006, pp. 487 –496.

[8] K.-J. Lee, "A 90 nm 1.8 v 512 mb diode-switch pram with 266 mb/s read throughput," vol. 43, no. 1, jan. 2008, pp. 150 –162.

[9] G. Dhiman, R. Ayoub, and T. Rosing, "Pdram: A hybrid pram and dram main memory system," in *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*, july 2009, pp. 664 –669.

[10] H. Park, S. Yoo, and S. Lee, "Power management of hybrid dram/pram-based main memory," in *Proceedings of the 48th Design Automation Conference*, ser. DAC'11. New York, NY, USA: ACM, 2011, pp. 59–64. [Online]. Available: http://doi.acm.org/10.1145/2024724.2024738

[11] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, "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*, ser. MICRO 42. New York, NY, USA: ACM, 2009, pp. 14–23. [Online]. Available: http://doi.acm.org/10.1145/1669112.1669117

[12] S. Lee, H. Bahn, and S. Noh, "Characterizing memory write references for efficient management of hybrid pcm and dram memory," in *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on*, july 2011, pp. 168 –175.

[13] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: building customized program analysis tools with dynamic instrumentation," vol. 40. New York, NY, USA: ACM, June 2005, pp. 190–200. [Online]. Available: http://doi.acm.org/10.1145/1064978.1065034

[14] O. M. Ibrahim, S. A. Klein, and J. W. Mitchell, "Optimum heat power cycles for specified boundary conditions," vol. 113, no. 4. ASME, 1991, pp. 514–521. [Online]. Available: http://link.aip.org/link/?GTP/113/514/1

[15] Micron, "http://www.micron.com/support/dram/power-calc," in *DDR3 TN-41-01*.

[16] O. Avissar, R. Barua, and D. Stewart, "An optimal memory allocation scheme for scratch-pad-based embedded systems," vol. 1. New York, NY, USA: ACM, November 2002, pp. 6–26. [Online]. Available: http://doi.acm.org/10.1145/581888.581891

[17] Y. Guo, Q. Zhuge, J. Hu, M. Qiu, and E.-M. Sha, "Optimal data allocation for scratch-pad memory on embedded multi-core systems," in *Parallel Processing (ICPP), 2011 International Conference on*, sept. 2011, pp. 464 –471.

[18] W. Che, A. Panda, and K. S. Chatha, "Compilation of stream programs for multicore processors that incorporate scratchpad memories," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '10. 3001 Leuven, Belgium, Belgium: European Design and Automation Association, 2010, pp. 1118–1123. [Online]. Available: http://dl.acm.org/citation.cfm?id=1870926.1871198

[19] Z. Wang and X. S. Hu, "Power aware variable partitioning and instruction scheduling for multiple memory banks," in *Proceedings of the conference on Design, automation and test in Europe - Volume 1*, ser. DATE '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 10 312–. [Online]. Available: http://dl.acm.org/citation.cfm?id=968878.969087

[20] T. Wanyong, L. Jianhua, Z. Yingchao, J. Chun, L. Minming, and C. Enhong, "Optimal task allocation on non-volatile memory based hybrid main memory." FL, USA: ACM, November 2011. [Online]. Available: http://dx.doi.org/10.1109/MC.2003.1250880

[21] Y. Park and K.-H. Park, "Linux kernel support to exploit phase change memory," in *Linux Symposium 2010*, 2010, pp. 217 –224.

[22] J. C. Mogul, E. Argollo, M. Shah, and P. Faraboschi, "Operating system support for nvm+dram hybrid main memory," in *Proceedings of the 12th conference on Hot topics in operating systems*, ser. HotOS'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 14–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855568.1855582

Notes

[1]Suggested method, pad 1 at both ends of each page's memory allocation map, can be adopted other resource allocation research area.