Task Allocation on Nonvolatile-Memory-Based Hybrid Main Memory

Wanyong Tian, Yingchao Zhao, Liang Shi, Qingan Li, Jianhua Li, Chun Jason Xue, *Member, IEEE*, Minming Li, and Enhong Chen, *Senior Member, IEEE*

***Abstract*— In this paper, we consider the task allocation problem on a hybrid main memory composed of nonvolatile memory (NVM) and dynamic random access memory (DRAM). Compared to the conventional memory technology DRAM, the emerging NVM has excellent energy performance since it consumes orders of magnitude less leakage power. On the other hand, most types of NVMs come with the disadvantages of much shorter write endurance and longer write latency as opposed to DRAM. By leveraging the energy efficiency of NVM and long write endurance of DRAM, this paper explores task allocation techniques on hybrid memory for multiple objectives such as minimizing the energy consumption, extending the lifetime, and minimizing the memory size. The contributions of this paper are twofold. First, we design the integer linear programming (ILP) formulations that can solve different objectives optimally. Then, we propose two sets of heuristic algorithms including three polynomial time offline heuristics and three online heuristics. Experiments show that compared to the optimal solutions generated by the ILP formulations, the offline heuristics can produce near-optimal results.**

***Index Terms*— Hybrid main memory, integer linear programming (ILP), nonvolatile memory (NVM).**

I. Introduction

Energy consumption is an important issue in the design of embedded systems. While the main processor has always been the primary energy-consuming device, the main memory has become a significant energy dissipator in recent years. Research [1] shows that the main memory accounts for up to 40% of total energy consumption on modern server systems. Thus, optimizing energy consumption of the main memory is crucial to the overall energy budget of the system. Dynamic random access memory (DRAM) has been widely used as the main memory of computer systems for decades. Recently, several emerging nonvolatile memory (NVM) technologies such as phase change random access memory (PRAM) [2], [3] and spin transfer torque RAM (STT-RAM) [4], [5] have been proposed. Compared to DRAM, these emerging NVMs have more promising characteristics for future universal memory. PRAM has better power efficiency but shorter write endurance and longer access latency. STT-RAM is faster than PRAM. The write endurance of STT-RAM is about 10^{15}, which is much longer than that of PRAM. Sun *et al.* [6] and Li *et al.* [7] have proposed the STT-RAM-based hybrid cache which can reduce energy consumption and improve performance. Hu *et al.* [8] have designed a novel NVM-based scratch pad memory to take advantage of the ultralow leakage consumption of NVM. Dong *et al.* [9] have presented the NVM cache model to exploit the low leakage dissipation of NVM. Table I shows the comparison of the characteristics of DRAM and PRAM.

Since NVM has the disadvantages of long write latency and limited lifetime, it is not desirable to be directly employed as the main memory. Otherwise, the system performance will be adversely impacted and lifetime will be severely reduced. Recently, hybrid memory has become a hot research topic. As NVMs usually have limited endurance based on the number of writes, different mechanisms to reduce writes on NVM in order to extend the lifetime have been proposed in [10]–[12]. Qureshi *et al.* [13] have analyzed a PRAM-based hybrid main memory composed of PRAM storage with a small DRAM buffer. They showed that such an architecture has not only the latency benefit of DRAM but also the capacity benefit of PRAM. Dhiman *et al.* [14] have proposed a novel energy-efficient main memory architecture based on PRAM and DRAM, and designed a low-overhead hybrid hardware/software solution to manage this new main memory. Their experimental results showed that the new memory system could save 30% energy on average at negligible overhead compared with traditional DRAM-based main memory. Mogul *et al.* [15] have

Manuscript received December 15, 2011; revised March 21, 2012; accepted June 12, 2012. Date of publication August 14, 2012; date of current version June 21, 2013. This work was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Project CityU 123609 and Project CityU 123811, the City University of Hong Kong under Project 7002611, the Natural Science Foundation of China under Grant 61073110, the National Major Special Science and Technology Project under Grant 2011ZX04016-071, the HeGaoJi National Major Special Science and Technology Project under Grant 2012ZX01029001-002, and the Research Fund for the Doctoral Program of Higher Education of China under 20113402110024.

W. Tian, L. Shi, and J. Li are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China, with the Department of Computer Science, City University of Hong Kong, Hong Kong, and also with USTC-CityU Joint Research Institute, Suzhou 215123, China.

Y. Zhao is with the Department of Computer Science, Caritas Institute of Higher Education, Hong Kong.

C. J. Xue, and M. Li are with the Department of Computer Science, City University of Hong Kong, Hong Kong.

Q. Li is with the Computer School, Wuhan University, Wuhan 430072, China.

E. Chen is with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China (e-mail: cheneh@ustc.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2012.2208129

TABLE I
CHARACTERISTICS OF DRAM AND PRAM (SOURCES INCLUDE NVSIM [16], CACTI, [17])

Tech.	Write Endurance (cycles)	Access latency (ns)		Access energy (nJ/bit)		Leakage power (mW)
		Read	Write	Read	Write	
DRAM	∞	104.4	104.4	3.26	3.26	1924
PRAM	$10^8 - 10^9$	143.5	270.53/135.05 (SET/RESET)	0.043	10.05/10.98 (SET/RESET)	194

presented an OS-level management policy for hybrid memory to hide the disadvantages of PRAM while exploiting its ideal attributes.

In this paper, we consider the task allocation problem on hybrid main memory as follows: given a set of tasks to be placed in the hybrid main memory for execution, each task is characterized by arrival time, finish time, size, number of reads, and number of writes, and these tasks need to be allocated to the hybrid main memory to optimize different given objectives. The objectives of this paper are: 1) minimizing energy consumption; 2) minimizing number of writes in NVM; and 3) minimizing the NVM size of the main memory. According to the properties of the different parts of the hybrid main memory, this paper proposes techniques that strive to assign each task to the most suitable memory location to achieve different objectives. This paper targets the embedded systems that often use physical addresses and do not apply techniques such as virtual memory. The proposed algorithms will calculate the allocated memory locations to determine the physical address of each task. Using the proposed management mechanisms, we can reduce energy dissipation while extending the lifetime of NVMs. To the best of our knowledge, this is the first paper on task allocation level to exploit the hybrid main memory consisting of NVM and DRAM. The main contributions of this paper are as follows.

1) We propose an integer linear programming (ILP) model for optimizing NVM-based hybrid main memory. Different objectives such as minimizing energy consumption, NVM writes, and NVM size are considered.
2) We present efficient offline and online heuristic algorithms for different objectives. The offline heuristics could produce near-optimal results compared to the optimal solutions generated by the ILP formulations.

In the rest of this paper, PRAM will be used as a representative of NVM in the hybrid main memory. However, the methodologies introduced in this paper are also suitable for other types of NVMs.

The remainder of this paper is organized as follows. Section II presents the problem description and gives three motivational examples to show the effectiveness of the proposed algorithms. In Section III, the ILP formulations and optimal solutions of the target problem are proposed. Section IV presents the heuristics to minimize energy consumption, number of writes on PRAM, and PRAM size. Section V shows the experimental results. In Section VI, we present the related work on hybrid memory problem. Finally, we conclude this paper and point out the future work in Section VII.

II. PROBLEM ANALYSIS

In this section, we first give the background information of PRAM, and then present the problem description and task allocation principle. Finally, we give a motivational example to show the effectiveness of the proposed offline heuristic MinE.

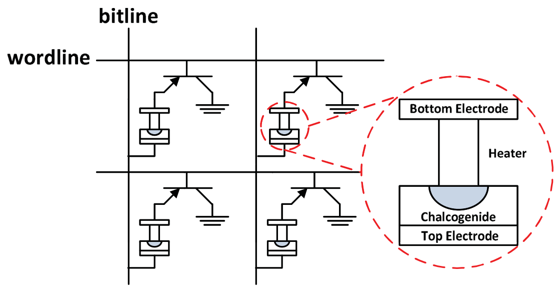


Fig. 1. PRAM cell array [18].

A. PRAM Background Information

1) PRAM Basis: PRAM is one type of NVM that exploits the unique behavior of a chalcogenide alloy to store information [2]. A PRAM cell usually consists of a thin layer of chalcogenide such as Ge2Sb2Te5 (GST) and two electrodes attached to the chalcogenide from each side (Fig. 1). The chalcogenide has two stable states, i.e., crystalline and amorphous.

Two operations can change the resistance of a PRAM cell and thus its stored information. As shown in Fig. 2, the SET operation has the GST heated above the crystallization temperature (300 °C) but below the melting temperature (600 °C) over a period. This turns the GST into the low-resistance crystalline state (logic "1"); the RESET operation gets GST heated above the melting temperature and quenched quickly. This places the GST in the high-resistance amorphous state (logic "0").

2) PRAM Write: With repeated heat stress applied to the phase change material, a PRAM cell survives only a limited number of write cycles. Referred to as write endurance, it is a key parameter in designing PRAM-based memory systems. A typical PRAM cell can sustain 10^8–10^9 writes before it gets stuck at the SET or RESET states [2]. PRAMs write endurance is worse than that of DRAM (10^{15} or ∞ write cycles before failure).

PRAM write requires a higher voltage than conventional V_{dd}. Therefore, a PRAM chip usually separates read and write circuits, and integrates charge pumps to boost the write voltage [19]. Thus, PRAM has larger peripherals and consumes more write energy than DRAM. Fortunately, technology scaling decreases the volume of the phase change material. It requires smaller current and less energy to program future PRAM cells.

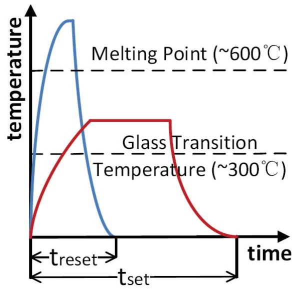


Fig. 2. RESET and SET operations [18].

B. Problem Description

The problem that this paper targets to solve is described as follows: We are given a task set $T = \{\tau_1, \tau_2, \ldots, \tau_n\}$, and each task τ_i is characterized by $\langle a_i, f_i, s_i, Nr_i, Nw_i\rangle$, where a_i means the arrival time, f_i means the finish time, s_i denotes the size, Nr_i represents the number of reads, and Nw_i represents the number of writes of the task. In the time interval $[a_i, f_i)$, τ_i should be allocated s_i continuous memory bytes. In this paper, we assume that each task could be entirely allocated in one part or occupy continuous physical locations across the two parts. Besides, τ_i will be read Nr_i times and written Nw_i times. Reads and writes are evenly distributed across the address space of a task. If any of the above constraints is violated, τ_i fails.

The notations used in this paper are summarized in Table II. The energy consumptions[1] for each read and write on PRAM are denoted by Erp and Ewp. The energy consumptions for each read and write on DRAM are denoted by Erd and Ewd. Several prototypes of PRAM have been presented recently [3], [13], [20]. To be independent of specific PRAM and DRAM prototypes, we utilize abstract PRAM and DRAM models with the parameters set as follows: $Ewd = Erd = 5$; $Ewp = 15$; $Erp = 1$.

We will strive to allocate each task in proper locations of PRAM or DRAM to ensure no task failure such that the following objectives will be achieved.

1) *Case 1:* Given the sizes of PRAM and DRAM and a threshold of number of writes on PRAM, minimize the energy consumption.
2) *Case 2:* Given the sizes of PRAM and DRAM and a threshold of energy consumption, minimize number of writes on PRAM.
3) *Case 3:* Given the size of DRAM, a threshold of energy consumption, and number of writes on PRAM, minimize the PRAM size.

C. Task Allocation Principle

The hybrid memory consists of two parts: DRAM and PRAM. The two parts form a combined homogeneous address space. The size of the hybrid memory is $Dt + Pt$. The first Dt bytes belong to DRAM, while the last Pt bytes belong to PRAM. A task can either be entirely allocated in one part, or can occupy continuous locations across the two parts.

[1]In this paper, the two parts of the main memory are always in "active" mode, i.e., the leakage energies of PRAM and DRAM are fixed. Thus, only access energy (write/read) is considered.

TABLE II
NOTATIONS USED IN THIS PAPER

Notation	Description
T	The task set.
τ_i	The ith task.
a_i	Arrival time of τ_i.
f_i	Finish time of τ_i.
s_i	Size of τ_i.
Nr_i/Nw_i	Number of reads/writes of τ_i.
x_i/z_i	Tag of whether τ_i is totally assigned in DRAM/PRAM.
y_i	Starting location of τ_i.
$f_{i,j}$	Spatial order of τ_i and τ_j that have temporal overlaps. 0 represents $y_i > y_j$; 1 represents $y_i < y_j$.
Erp/Ewp	Energy consumption of each read/write on PRAM.
Erd/Ewd	Energy consumption of each read/write on DRAM.
Ed_i	Energy consumption of τ_i when entirely allocated on DRAM. $Ed_i = Nr_i \cdot Erd + Nw_i \cdot Ewd$
Ep_i	Energy consumption of τ_i when entirely allocated on PRAM. $Ep_i = Nr_i \cdot Erp + Nw_i \cdot Ewp$
ΔE_i	Energy consumption difference of τ_i on PRAM and DRAM. $\Delta E_i = Ed_i - Ep_i$
D-task	Task τ_i with $\Delta E_i < 0$.
P-task	Task τ_i with $\Delta E_i \geq 0$.
d_i/p_i	Size of τ_i allocated in DRAM/PRAM.
E_i	Energy consumption of τ_i when allocated across DRAM and PRAM. $E_i = (d_i/s_i) \cdot Ed_i + (p_i/s_i) \cdot Ep_i$
N_i	Number of writes on PRAM of τ_i. $N_i = (p_i/s_i) \cdot Nw_i$
P/D	Used size of PRAM/DRAM.
Pt/Dt	Threshold of PRAM/DRAM size.
E	Energy consumption of all tasks.
Et	Threshold of energy consumption
N	Number of writes on PRAM.
Nt	Threshold of number of writes on PRAM.

For each task τ_i entirely located in PRAM or DRAM, there are two energy values: Ed_i, Ep_i

$$Ed_i = Nr_i \cdot Erd + Nw_i \cdot Ewd$$
$$Ep_i = Nr_i \cdot Erp + Nw_i \cdot Ewp$$

where Ed_i (Ep_i) represents the energy consumption of τ_i when τ_i is entirely assigned to DRAM (PRAM).

If task τ_i is assigned across DRAM and PRAM, the energy consumption (E_i) of τ_i is

$$E_i = \frac{d_i}{s_i} \cdot Ed_i + \frac{p_i}{s_i} \cdot Ep_i \quad (1)$$

and the number of writes on PRAM (N_i) of τ_i is

$$N_i = \frac{p_i}{s_i} \cdot Nw_i \quad (2)$$

where d_i (p_i) represents the size in DRAM (PRAM). Obviously, $0 \leq d_i, p_i \leq s_i$, $d_i + p_i = s_i$.

We define $\Delta E_i = Ed_i - Ep_i$. Two notations are given as follows.

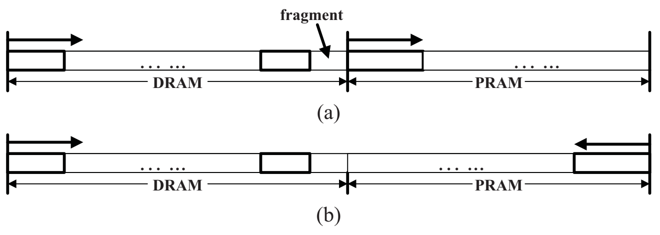


Fig. 3. Task allocation principle.

1) *D-task:* If $Ed_i < Ep_i$ ($\Delta E_i < 0$), τ_i is a D-task.
2) *P-task:* If $Ed_i \geq Ep_i$ ($\Delta E_i \geq 0$), τ_i is a P-task.

If the two parts of the memory both start allocation from the lowest free address, there might be fragments in the last bytes of DRAM [as shown in Fig. 3(a)]. To reduce fragmentation, location of each task assigned in DRAM and PRAM begins from different addresses; tasks assigned to DRAM begin from the lowest free address, while tasks assigned to PRAM begin from the highest free address [as shown in Fig. 3(b)]. Thus, the used PRAM size P could be calculated as follows:

$$P = Pt - P_{lu}$$

where P_{lu} is the largest unallocated address.

In the rest of this paper, task allocation will follow the above principle.

D. Motivational Example

In this section, a motivational example is presented to show the effectiveness of the proposed offline heuristic MinE. Notice that a task τ_i is characterized as $\tau_i = \langle a_i, f_i, s_i, Nr_i, Nw_i \rangle$, $\Delta E_i = Ed_i - Ep_i = Nr_i \cdot (Erd - Erp) + Nw_i \cdot (Ewd - Ewp)$. The parameters are set as follows: $Ewd = Erd = 5$, $Ewp = 15$, $Erp = 1$.

For Case 1, assume $Nt = 10$, $Pt = 20$, $Dt = 20$, $T = \{\tau_1, \ldots, \tau_8\}$. The tasks shown in Fig. 4(a) are as follows:

$$\begin{aligned}
\tau_1 &= \langle 1, 8, 4, 7, 2 \rangle, \quad \Delta E_1 = 8\\
\tau_2 &= \langle 0, 10, 2, 10, 3 \rangle, \quad \Delta E_2 = 10\\
\tau_3 &= \langle 2, 8, 5, 13, 4 \rangle, \quad \Delta E_3 = 12\\
\tau_4 &= \langle 1, 6, 2, 14, 4 \rangle, \quad \Delta E_4 = 16\\
\tau_5 &= \langle 1, 12, 3, 1, 1 \rangle, \quad \Delta E_5 = -6\\
\tau_6 &= \langle 5, 13, 6, 1, 1 \rangle, \quad \Delta E_6 = -6\\
\tau_7 &= \langle 4, 14, 12, 0, 1 \rangle, \quad \Delta E_7 = -10\\
\tau_8 &= \langle 7, 13, 2, 1, 2 \rangle, \quad \Delta E_8 = -16.
\end{aligned}$$

We compare the solutions generated by the proposed offline heuristic MinE and the simple heuristic MinE_CP. The principles of these two algorithms are similar: all P-tasks and D-tasks are originally assigned to PRAM and DRAM, respectively, which might lead to $N > Nt$, $P > Pt$, $D > Dt$. Then tasks are iteratively migrated between PRAM and DRAM until none of the three former inequalities holds. The difference lies in the following: MinE considers not only parameter ΔE_i but also s_i and Nw_i. It migrates the task with the smallest $(|\Delta E_i|/Nw_i)$ or $(|\Delta E_i|/s_i)$ iteratively. MinE_CP merely migrates the task with the smallest $|\Delta E_i|$ at each step.

Fig. 4(a) lists the attributes of all tasks. The two algorithms proceed as follows. First, as shown in Fig. 4(b), all P-tasks and D-tasks are originally assigned to PRAM and DRAM, respectively. Fig. 4(c) depicts that MinE_CP sequentially migrates τ_1, τ_5, τ_2, and τ_6 between PRAM and DRAM, energy consumption $E_{\text{MinE\_CP}} = E_o + |\Delta E_1| + |\Delta E_5| + |\Delta E_2| + |\Delta E_6| = 309$. From Fig. 4(d) we can see MinE only requires to migrate τ_3 (with the smallest $(|\Delta E_i|/Nw_i)$) and τ_7 (with the smallest $(|\Delta E_i|/s_i)$), energy consumption $E_{\text{MinE}} = E_o + |\Delta E_3| + |\Delta E_7| = 301$, which is smaller than $E_{\text{MinE\_CP}}$.

III. ILP Formulations

In this section, the ILP formulations of the three cases are given. General constraints are presented in Section III-A. Constraints for the three cases are proposed in Section III-B.

A. General Constraints

The size of PRAM is Pt and the size of DRAM is Dt. We introduce a constant $M = n \sum_{i=1}^{n} s_i$, which is a large enough number. The objective is to find the optimal location for each task.

For each task τ_i, there are three variables x_i, y_i, and z_i. Variable y_i represents the starting location of task τ_i. Variable x_i represents whether task τ_i is totally located in DRAM. Variable z_i represents whether task τ_i is totally located in PRAM

$$x_i = \begin{cases} 0, & \text{if task } \tau_i \text{ is not totally assigned in DRAM} \\ 1, & \text{if task } \tau_i \text{ is totally assigned in DRAM} \end{cases} \quad (3)$$

$$z_i = \begin{cases} 0, & \text{if task } \tau_i \text{ is not totally assigned in PRAM} \\ 1, & \text{if task } \tau_i \text{ is totally assigned in PRAM.} \end{cases} \quad (4)$$

The two memories can be considered together as a whole memory with the size $Dt + Pt$. The first Dt bytes belong to DRAM, while the last Pt bytes belong to PRAM. A natural constraint for y_i is as follows:

$$y_i \geq 0 \quad \forall i \in \{1, \ldots, n\}. \quad (5)$$

Another natural constraint is

$$0 \leq x_i + z_i \leq 1. \quad (6)$$

To make sure that each task is assigned within the suitable part of the memory, we give the following constraints:

$$\begin{cases} y_i \leq Dt - s_i + (1 - x_i) \cdot M \\ y_i \geq Dt + (z_i - 1) \cdot M \\ y_i \leq Dt + Pt - s_i \\ y_i > Dt - s_i - (x_i + z_i) \cdot M \\ y_i < Dt + (x_i + z_i) \cdot M \end{cases} \quad \forall i \in \{1, \ldots, n\}. \quad (7)$$

In the above constraints, if $x_i = 1$ and $z_i = 0$, which means that task τ_i is assigned to DRAM, the first inequality implies that $y_i \leq Dt - s_i$ and the second inequality will always hold because of inequality (5). If $x_i = 0$ and $z_i = 1$, which means that task τ_i is assigned to PRAM, the first inequality will always hold because M is very large while the second inequality implies that $y_i \geq Dt$. The fourth and fifth inequalities deals with the case where one task crosses two types of memories.

τ_i	$\frac{\lvert\Delta E_i\rvert}{Nw_i}$	$\frac{\lvert\Delta E_i\rvert}{s_i}$	ΔE_i
τ_1	8/2	8/4	8
τ_2	10/3	10/2	10
τ_3	12/4	12/5	12
τ_4	16/4	16/2	16
τ_5	6/1	6/3	−6
τ_6	6/1	6/6	−6
τ_7	10/1	10/12	−10
τ_8	16/2	16/2	−16

(a)

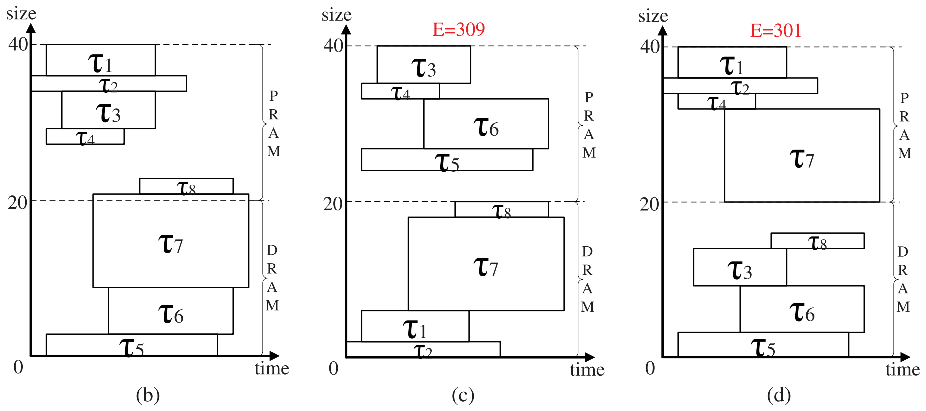


Fig. 4. Motivational example to minimize energy consumption. (a) Attributes of all tasks. (b) Original tasks. (c) Result of "MinE_CP." (d) Result of "MinE."

Next, we introduce another two variables for each task. Let d_i be the size of τ_i allocated in DRAM and p_i be the size of τ_i allocated in PRAM. First, we have

$$d_i + p_i = s_i. \tag{8}$$

We enforce the following two sets of constraints which will guarantee that the values of d_i and p_i are correctly decided for different combinations of x_i and z_i

$$\begin{cases} d_i \geq 0 \\ d_i \geq x_i \cdot s_i + (x_i - 1) \cdot M \\ d_i \geq (Dt - y_i) - (x_i + z_i) \cdot M \\ p_i \geq 0 \\ p_i \geq z_i \cdot s_i + (z_i - 1) \cdot M \\ p_i \geq s_i - (Dt - y_i) - (x_i + z_i) \cdot M \end{cases} \quad \forall i \in \{1, \ldots, n\}. \tag{9}$$

We can see that, if $x_i + z_i = 0$, which means task τ_i is crossing the boundary of two types of memories, the above constraints give $d_i \geq Dt - y_i$ and $p_i \geq s_i - (Dt - y_i)$ which forces $d_i + p_i = s$. When $x_i = 1$, the above constraints give $d_i \geq s_i$ and $p_i \geq 0$. When $z_i = 1$, the above constraints give $d_i \geq 0$ and $p_i \geq s_i$. Therefore, it fits all the three cases.

To make sure that any two tasks overlapping in time do not overlap in the memory space, we introduce a new variable f_{ij} to indicate the spatial order between tasks that have temporal overlap. For any two tasks τ_i and τ_j such that their execution times have overlap, the variable

$$f_{ij} = \begin{cases} 0, & \text{if } y_i > y_j \\ 1, & \text{if } y_i < y_j. \end{cases} \tag{10}$$

From the definition of f_{ij}, we can get the following property directly:

$$f_{ij} + f_{ji} = 1. \tag{11}$$

Since f_{ij} indicates the location order for tasks that have time overlap, variable f_{ij} and y_i, y_j have the following relation:

$$y_j - y_i < f_{ij} \cdot M. \tag{12}$$

For two tasks τ_i and τ_j that have overlap in their execution time, if $y_i < y_j$, then there must be $y_i + s_i \leq y_j$. We use the following inequality to guarantee this property:

$$y_j \geq y_i + s_i + (f_{ij} - 1) \cdot M. \tag{13}$$

Notice that we do not consider f_{ij} if task i and task j have no overlap in their execution time and we even do not care about the relation between corresponding y_i and y_j.

The above constraints (3)–(13) restrict the spatial relations of tasks according to their execution times and ensure that no violation will take place. For the proposed three problems, additional constraints should be added.

B. Three Cases

1) Case 1: Minimizing Energy Consumption: To minimize the energy consumption (E) when Dt, Pt, and Nt are given, the added constraint is

$$\sum_{i=1}^{n} \frac{p_i}{s_i} \cdot Nw_i \leq Nt. \tag{14}$$

The objective is

$$\min \sum_{i=1}^{n} \left(\frac{d_i}{s_i} \cdot (Nr_i \cdot Erd + Nw_i \cdot Ewd) + \frac{p_i}{s_i} \cdot (Nr_i \cdot Erp + Nw_i \cdot Ewp) \right). \tag{15}$$

2) Case 2: Minimizing Number of Writes on PRAM: To minimize the number of writes on PRAM (N) when Dt, Pt, and Et are given, the added constraint is

$$\sum_{i=1}^{n} \left(\frac{d_i}{s_i} \cdot (Nr_i \cdot Erd + Nw_i \cdot Ewd) + \frac{p_i}{s_i} \cdot (Nr_i \cdot Erp + Nw_i \cdot Ewp) \right) \leq Et. \tag{16}$$

The objective is

$$\min \sum_{i=1}^{n} \frac{p_i}{s_i} \cdot Nw_i. \tag{17}$$

3) Case 3: Minimizing PRAM Size: To minimize the PRAM size (P) when Dt, Nt, and Et are given, the added constraints include (14) and (16). Besides, the third inequality of constraint (7) should be modified as follows:

$$y_i \leq Dt + P - s_i \quad \forall i \in \{1, \ldots, n\}. \tag{18}$$

The objective is

$$\min P. \tag{19}$$

IV. Algorithms

The objective is to exploit the advantages of DRAM and PRAM while hiding their disadvantages. However, such objectives are conflicting with the task allocation problem. For instance, minimizing energy consumption prefers to place all P-tasks into PRAM, the side effect of which is increasing the writes on PRAM. Therefore, careful balancing is crucial.

There are four constraints in total: 1) a threshold of energy consumption (Et); 2) a threshold of PRAM size (Pt); 3) a threshold of DRAM size (Dt); and 4) a threshold of the number of writes on PRAM (Nt). By fixing any three of the constraints, we can optimize the fourth one. Thus, naturally four problems exist. Since it is similar to minimizing the PRAM size or the DRAM size, we only select to minimize the PRAM size.

The three problems are listed as follows.

1) To minimize energy consumption (E) when Pt, Dt, and Nt are given.
2) To minimize the number of writes on PRAM (N) when Pt, Dt, and Et are given.
3) To minimize the PRAM size (P) when Dt, Et, and Nt are given.

In this section, two sets of algorithms are proposed to tackle the above problems. The first set consists of three offline algorithms: MinE, MinN, and MinP. The second set includes three online algorithms: MinE′, MinN′, and MinP′.

A. *Offline Algorithms*

Three offline heuristics MinE, MinN, and MinP are presented in this section. All these heuristics are polynomial time solvable. The most time-consuming part lies in computing *P/D*, i.e., the PRAM/DRAM size used by all allocated tasks, which is a well-known NP-hard problem—the dynamic storage allocation (DSA) problem [21]. We use the polynomial time approximation algorithm proposed by Buchsbaum *et al.* [22], which can present the best and worst case performance among all existing algorithms.

1) Minimizing Energy Consumption: Algorithm 1 presents the methodology to minimize energy consumption offline. First, all P-tasks and D-tasks are assigned in PRAM and DRAM, respectively. Thus, the energy consumption E will be the lower bound of this problem, while the used PRAM size P, used DRAM size D, and the number of writes on PRAM N might surpass the thresholds Nt, Pt, and Dt, respectively. To avoid violating the constraints, some tasks need to be migrated between PRAM and DRAM. Naturally, the tasks needing large sizes or many writes will be selected. Since N, P, and D might all be larger than the thresholds Nt, Pt, and Dt, each time the most "urgent" task should be selected for migration. Two ratios are defined: $R_1 = (N/Nt)$, $R_2 = (P/Pt)$, $R_3 = (D/Dt)$. Comparing R_1–R_3, we determine which task should be migrated first. For example, if R_1 is the largest, we would prefer to select the task with the largest Nw_i to migrate. But it is not enough to only consider the parameter Nw_i; energy consumption E_i should also be taken into account with regard to the objective of "minimizing energy consumption." As a result, P-task τ_i with the smallest ratio $(|\Delta E_i|/Nw_i)$

Algorithm 1 MinE (offline): Minimizing Energy Consumption

```
Require: Task set $T = \{\tau_1, \tau_2, \ldots\}$, $Nt$, $Pt$, $Dt$.
Ensure: Energy consumption ($E$) of the hybrid memory.
1: Place all P-tasks in PRAM and all D-tasks in DRAM,
   Compute $P$, $D$, $N$;
2: $R_1 \leftarrow (N/Nt)$, $R_2 \leftarrow (P/Pt)$, $R_3 \leftarrow (D/Dt)$;
3: while $R_1 > 1$ or $R_2 > 1$ or $R_3 > 1$ do
4:   if $R_1 \geq R_2$ and $R_1 \geq R_3$ then
5:     Migrate P-task $\tau_i$ with the smallest $(|\Delta E_i|/Nw_i)$ from
       PRAM to DRAM;
6:   end if
7:   if $R_2 > R_1$ and $R_2 \geq R_3$ then
8:     Migrate P-task $\tau_i$ with the smallest $(|\Delta E_i|/s_i)$ from
       PRAM to DRAM;
9:   end if
10:  if $R_3 > R_1$ and $R_3 > R_2$ then
11:    Migrate D-task $\tau_i$ with the smallest $(|\Delta E_i|/s_i)$ from
       DRAM to PRAM;
12:  end if
13:  Recompute $R_1$, $R_2$, $R_3$;
14: end while
15: Compute $E$;
16: Return $E$;
```

will be migrated from PRAM to DRAM.[2] Similarly, if R_2 (R_3) is the largest, P-task (D-task) τ_i with the smallest ratio $(|\Delta E_i|/s_i)$ will be migrated from PRAM to DRAM (from DRAM to PRAM). In each iteration, the most "urgent" task is selected, and R_1–R_3 are recomputed for the next iteration. The process will not terminate until R_1–R_3 are all smaller than 1. Finally, the resulting E will be the desired energy consumption.

2) Minimizing Number of Writes on PRAM: The heuristic MinN with the objective to minimize number of writes on PRAM offline is shown in Algorithm 2, the principle of which is as follows. First, all tasks are assigned in DRAM. Thus, $D > Dt$ and $E > Et$ might hold. Then, tasks need to be iteratively migrated from DRAM to PRAM until neither of these inequalities holds. At each iteration step, if $R_1 > R_2$, the task τ_i with the largest (s_i/Nw_i) will be migrated from DRAM to PRAM. Else, ($R_1 \leq R_2$), P-task with the largest $(|\Delta E_i|/Nw_i)$ will be selected for migration from DRAM to PRAM. When the iteration terminates, we check if $P \leq Pt$ holds. If yes, N will be the desired solution. Otherwise, no result can be derived.

3) Minimizing PRAM Size: For Case 3, the heuristic MinP with the objective to minimize the PRAM size offline is similar to MinN. Thus, MinP is omitted here. The approach proceeds as follows. All the tasks are initially assigned to DRAM, and two inequalities $D > Dt$ and $E > Et$ might hold. Then, at each iteration, the task τ_i with the smallest size s_i (if $(D/Dt) > (E/Et)$) or the P-task with the largest $(|\Delta E_i|/s_i)$ (if $(D/Dt) \leq (E/Et)$) is iteratively migrated from the DRAM

[2] P-task τ_i in PRAM could be entirely or partially allocated in PRAM. For P-task τ_i which is partially in PRAM, $s_i = p_i$, E_i, and N_i could be computed as (1) and (2). $|\Delta E_i| = |E_i - Ed_i|$, $Nw_i = N_i$. This principle also holds in the following heuristic algorithms.

Algorithm 2 MinN (offline): Minimizing Number of Writes on PRAM

```
Require: Task set $T = \{\tau_1, \tau_2, \ldots\}$, $Et$, $Pt$, $Dt$.
Ensure: Number of writes on PRAM ($N$).
1: Place all tasks in DRAM, compute $D$, $E$;
2: $R_1 \leftarrow (D/Dt)$, $R_2 \leftarrow (E/Et)$;
3: while $R_1 > 1$ or $R_2 > 1$ do
4:   if $R_1 > R_2$ then
5:     Migrate $\tau_i$ with the largest $(s_i/Nw_i)$ from DRAM to PRAM;
6:   else
7:     Migrate P-task $\tau_i$ with the largest $(|\Delta E_i|/Nw_i)$ from DRAM to PRAM;
8:   end if
9:   Recompute $R_1$, $R_2$;
10: end while
11: Compute $N$, $P$;
12: if $P \leq Pt$ then
13:   Return $N$;
14: end if
```

to the PRAM until neither of the inequalities holds. When the iteration stops, check if N is smaller than Nt. If yes, P will be the solution. Otherwise, no result can be derived.

4) Minimizing Both E and N:

$$f = \alpha \cdot E + \beta \cdot N. \tag{20}$$

The combined overhead of E and N is defined as (20). To minimize both E and N, i.e., to minimize f, we can proceed as follows. For each N, invoke Algorithm MinE, so we can get an energy consumption E, Then the overhead f can be computed from (20). Given a series of E, we can get a series of N and f. Among all the values of f, the minimum value will be the desired objective. Notice that the coefficients α and β are set according to the relative importance of N and E.

B. Online Algorithms

In this section, three online algorithms are devised for different objectives. Due to lack of global information of tasks, a different mechanism—a first-fit policy—is adopted to place all the tasks. The first-fit policy proceeds as follows. For all tasks to be assigned in DRAM (PRAM), the task with earlier start time is allocated to lower (higher, see Section II-C) address, and all later tasks will be assigned one by one to the address consecutively after (before) the prior one. At each time instant t, each task τ_i with finish time f_i equal to t will be freed and the occupied memory locations will be available to other tasks from then on. Then, each task τ_j with start time equal to t will be allocated to memory according to the First-fit policy.

As shown in Table I, the access latency of reads and writes on the two memories is different. Considering the different read/write performance on the two memories, we make an assumption as follows. For each task τ_i, the given finish time f_i refers to the time instant τ_i should finish if τ_i is totally or partly assigned to PRAM (the slower memory). Otherwise (τ_i is totally assigned to DRAM), the finish time f_i will be brought forward to

$$f_i = a_i + (f_i - a_i) \cdot 0.8 \tag{21}$$

where a_i is the arrival time of τ_i. We applied this principle in the online algorithms.

Before presenting the algorithms, two notations are defined.

1) $T_s[t] = \{\tau_i | s_i = t\}$, refers to the set of tasks with start time $s_i = t$.
2) $T_f[t] = \{\tau_i | f_i = t\}$, refers to the set of tasks with finish time $f_i = t$.

1) Minimizing Energy Consumption: To minimize energy consumption online, all D-tasks and P-tasks are preferentially assigned to the DRAM and PRAM, respectively.[3] The process is shown in Algorithm 3. At each time instant t, first, tasks of which the finish time equals t are removed from the PRAM and DRAM, and the locations occupied by these tasks are freed and available to other tasks. Then, tasks of which the start time equals t are assigned to the PRAM or DRAM according to first-fit policy. Compute N, P, and D if tasks in $T_s[t]$ are (virtually) assigned to PRAM. If $R_1 = (N/Nt) \leq 1$, $R_2 = (P/Pt) \leq 1$, and $R_3 = (D/Dt) \leq 1$, all P-tasks and D-tasks in $T_s[t]$ are actually assigned to PRAM and DRAM, respectively. Otherwise ($R_1 > 1$ or $R_2 > 1$ or $R_3 > 1$), P-task $\tau_i \in T_s[t]$ with the largest $(|\Delta E_i|/N_i)$ (if R_1 is the largest) or P-task with the largest $(|\Delta E_i|/s_i)$ (if R_2 is the largest) or D-task with the largest $(|\Delta E_i|/s_i)$ (if R_3 is the largest) is preferentially assigned to PRAM (if R_1 or R_2 is the largest) or DRAM (if R_3 is the largest). If τ_i cannot be accommodated in the preferential memory, it is allocated to the other part. Repeat the allocation process until all tasks in $T_s[t]$ are allocated. Algorithm MinE′ terminates when the task with the largest finish time completes execution. If all tasks can be accommodated in PRAM or DRAM, the resulting E will be the desired energy consumption.

2) Minimizing Number of Writes on PRAM: The algorithm to minimize the number of writes on PRAM online is shown in Algorithm 4. At each time instant t, tasks in $T_f[t]$ are freed. Then, task τ_i with the smallest (s_i/Nw_i) is preferentially allocated to DRAM (if $R_1 = (D/Dt) > R_2 = (E/Et)$) or P-task with the largest $(|\Delta E_i|/Nw_i)$ is preferentially allocated to PRAM (if $R_1 \leq R_2$). Iterations are similar to those of MinE′. If all tasks can be accommodated to PRAM or DRAM, N will be the solution.

3) Minimizing PRAM Size: To minimize PRAM size online, tasks are preferentially assigned to DRAM. Algorithm MinP′ is similar to MinN′. Thus, the details are omitted here. MinP′ proceeds as follows. At each time t, first, remove all tasks in $T_f[t]$. Then, preferentially allocate task $\tau_i \in T_s[t]$ with the largest s_i to DRAM (if $R_1 = (D/Dt) > R_2 = (E/Et)$) or P-task with the largest $(|\Delta E_i|/s_i)$ to PRAM (if $R_1 \leq R_2$). If τ_i cannot be accommodated in its preferential part, it is assigned to the other part. If all tasks can be held in PRAM or DRAM, the computed P is the solution.

[3]Each task can be entirely or partially allocated in the PRAM/DRAM. If τ_i is partially in PRAM, s_i, $|\Delta E_i|$, and N_i could be calculated as in footnote 3.

Algorithm 3 MinE′ (online): Minimizing Energy Consumption

```
Require: Task set $T = \{\tau_1, \tau_2, \ldots\}$, $Nt$, $Pt$, $Dt$.
Ensure: Energy consumption ($E$) of the hybrid memory.
1: for $t = 0, 1, 2, \ldots$ do
2:   Free PRAM and DRAM locations occupied by tasks in $T_f[t]$;
3:   Compute $N$, $P$, $D$ if all P-tasks and D-tasks in $T_s[t]$ are assigned to PRAM and DRAM, respectively;
4:   $R_1 \leftarrow (N/Nt)$, $R_2 \leftarrow (P/Pt)$, $R_3 \leftarrow (D/Dt)$;
5:   while $R_1 > 1$ or $R_2 > 1$ or $R_3 > 1$ do
6:     if $R_1 \geq R_2$ and $R_1 \geq R_3$ then
7:       P-task $\tau_i \in T_s[t]$ with the largest $(|\Delta E_i|/N_i)$ is preferentially assigned to PRAM. If $\tau_i$ cannot be accommodated in PRAM, then it is assigned to DRAM;
8:     end if
9:     if $R_2 > R_1$ and $R_2 \geq R_3$ then
10:      P-task $\tau_i \in T_s[t]$ with the largest $(|\Delta E_i|/s_i)$ is preferentially assigned to PRAM. If $\tau_i$ cannot be accommodated in PRAM, then it is assigned to DRAM;
11:    end if
12:    if $R_3 > R_1$ and $R_3 > R_2$ then
13:      D-task $\tau_i \in T_s[t]$ with the largest $(|\Delta E_i|/s_i)$ is preferentially assigned to DRAM. If $\tau_i$ cannot be accommodated in DRAM, then it is assigned to PRAM;
14:    end if
15:    $T_s[t] \leftarrow T_s[t] \setminus \{\tau_i\}$;
16:    if $\tau_i$ is totally assigned to DRAM then
17:      $f_i \leftarrow a_i + (f_i - a_i) \cdot 0.8$;
18:    end if
19:    Compute $N$, $P$, $D$ if all P-tasks and D-tasks in $T_s[t]$ are assigned to PRAM and DRAM, respectively;
20:    $R_1 \leftarrow (N/Nt)$, $R_2 \leftarrow (P/Pt)$, $R_3 \leftarrow (D/Dt)$;
21:  end while
22:  if $T_s[t] \neq \emptyset$ then
23:    Allocate all P-tasks and D-tasks in $T_s[t]$ to PRAM and DRAM respectively, update each $f_i$ as (21) if $\tau_i$ is totally or partly assigned to DRAM;
24:  end if
25: end for
26: Compute $E$;
27: Return $E$;
```

Algorithm 4 MinN′(online): Minimizing Number of Writes on PRAM

```
Require: Task set $T = \{\tau_1, \tau_2, \ldots\}$, $Et$, $Pt$, $Dt$.
Ensure: Number of writes on PRAM ($N$).
1: for $t = 0, 1, 2, \ldots$ do
2:   Free PRAM and DRAM locations occupied by tasks in $T_f[t]$;
3:   Compute $D$ and $E$ if all tasks in $T_s[t]$ are assigned to DRAM;
4:   $R_1 \leftarrow (D/Dt)$, $R_2 \leftarrow (E/Et)$;
5:   while $R_1 > 1$ or $R_2 > 1$ do
6:     if $R_1 > R_2$ then
7:       The task $\tau_i$ in $T_s[t]$ with the smallest $(s_i/Nw_i)$ is preferentially assigned to DRAM. If $\tau_i$ cannot be accommodated in DRAM, then it is assigned to PRAM;
8:     else
9:       P-task $\tau_i$ in $T_s[t]$ with the largest $(|\Delta E_i|/Nw_i)$ is preferentially assigned to PRAM. If $\tau_i$ cannot be accommodated in PRAM, then it is assigned to DRAM;
10:    end if
11:    $T_s[t] \leftarrow T_s[t] \setminus \{\tau_i\}$;
12:    if $\tau_i$ is totally to DRAM then
13:      $f_i \leftarrow a_i + (f_i - a_i) \cdot 0.8$;
14:    end if
15:    Compute $D$ and $E$ if all tasks in $T_s[t]$ are assigned to DRAM;
16:    $R_1 \leftarrow (D/Dt)$, $R_2 \leftarrow (E/Et)$;
17:  end while
18:  if $T_s[t] \neq \emptyset$ then
19:    Allocate each task $\tau_i \in T_s[t]$ to DRAM, update $f_i$ as (21);
20:  end if
21: end for
22: Compute $N$, $P$;
23: if $P \leq Pt$ then
24:   Return $N$;
25: end if
```

V. Experiments

In this section, the experimental evaluation of the proposed algorithms are presented. We know that in real applications the number of read operations is often more than the number of write operations. During random generation of task sets, we assign the number of reads/writes based on sampling from MiBench [23] where the number of reads are $3-10$ times more than the number of writes. There are four sets of tasks, with the task numbers 40, 60, 80, and 100. Table III shows the attributes of all tasks. $E_{\text{ave}} = (1/n)\sum_{i=1}^{n}(Ep_i + Ed_i)$ represents the average energy consumption when all tasks are assigned to PRAM and DRAM. MaxN refers to the sum of number of writes of all tasks, MaxN $= \sum_{i=1}^{n} Nw_i$. S_{LB} corresponds to the largest sum of concurrent task sizes among all time instants, which is the lower bound of the required memory size.

The techniques under comparison are: 1) the ILP formulations for the three cases; 2) the proposed offline heuristics, including MinE, MinN, and MinP; 3) the online heuristics, including MinE′, MinN′, and MinP′; and 4) the simple heuristics for the three cases, including MinE_CP, MinN_CP, and MinP_CP, which will be illustrated later in this section.

The experiments were conducted on a desktop computer with an Intel Pentium 4 processor (3.39 GHz) and 2 GB memory. The LP solver used to solve the ILP formulations is Lingo [24]. We find that, when the constraints are too tight or the solution space is too large, the ILP formulations cannot

TABLE III
ATTRIBUTES OF FOUR TASK SETS

Task no.	E_{ave}	MaxN	S_{LB}
40	4124	195	160
60	5984	294	234
80	8023	383	310
100	9904	490	385

be optimally solved in finite time. All heuristics in this paper can be solved in seconds.

A. Simple Heuristics for Comparison

Since task allocation optimization problems on hybrid memory are brand new, no method exists that can solve these problems directly. We devise a set of simple algorithms for comparison: MinE_CP, MinN_CP, and MinP_CP. They are used to show the effectiveness of the proposed heuristics MinE, MinN, and MinP, respectively.

Take MinE_CP for example. The procedure is as follows.

Step 1: Allocate all P-tasks and D-tasks into PRAM and DRAM, respectively.

Step 2: In this case, constraints $N \leq Nt$, $P \leq Pt$ and $D \leq Dt$ might be violated. We iteratively migrate each task with the smallest ΔE_i among all unmigrated tasks between PRAM and DRAM. The migration process will not terminate until all the three above constraints hold.

Step 3: Compute the energy consumption $E_{\text{MinE\_CP}}$.

MinN_CP is analogous to MinE_CP. MinN_CP first assigns all tasks in DRAM, and then migrates each task τ_i with the smallest Nw_i from DRAM to PRAM until both $D \leq Dt$ and $E \leq Et$ hold. Finally, check if $P \leq Pt$ holds. If yes, compute N.

The only difference between MinP_CP and MinN_CP is as follows: in step 2 of MinP_CP, we migrate task τ_i with the smallest s_i at each iteration step.

B. Results Comparison

Tables IV–VI depict the results generated by all the ILP formulations and heuristics. Input parameters Dt, Pt, Nt, Et represent the threshold of DRAM size, PRAM size, number of writes on PRAM, and energy consumption, respectively. Since the ILP formulations are time-consuming to solve while all heuristics can be solved in negligible time, we only list the column "time" for ILP solutions. "*" in column "MinP_CP" and row "80" of Table V means that the heuristic MinP_CP has no solution, while "*" in column "ILP" of these three tables means that the ILP program cannot generate optimal solution in limited time.

In Table IV, E_{MinE}, $E_{\text{MinE\_CP}}$, and E_{ILP} refer to the energy consumption generated by heuristic MinE, $E_{\text{MinE\_CP}}$, and the ILP formulations.

1) $\text{Dif}e1 = \frac{E_{\text{MinE}} - E_{\text{ILP}}}{E_{\text{ILP}}} \cdot 100\%$.
2) $\text{Dif}e2 = \frac{E_{\text{MinE\_CP}} - E_{\text{ILP}}}{E_{\text{ILP}}} \cdot 100\%$.
3) $\text{Dif}e3 = \frac{E_{\text{MinE}'} - E_{\text{ILP}}}{E_{\text{ILP}}} \cdot 100\%$.

In Table V, N_{MinN}, $N_{\text{MinN\_CP}}$, and N_{ILP} represent the number of writes generated by heuristic MinN, $E_{\text{MinN\_CP}}$, and the ILP formulations.

1) $\text{Dif}n1 = \frac{N_{\text{MinN}} - N_{\text{ILP}}}{N_{\text{ILP}}} \cdot 100\%$.
2) $\text{Dif}n2 = \frac{N_{\text{MinN\_CP}} - N_{\text{ILP}}}{N_{\text{ILP}}} \cdot 100\%$.
3) $\text{Dif}n3 = \frac{N_{\text{MinN}'} - N_{\text{ILP}}}{N_{\text{ILP}}} \cdot 100\%$.

In Table VI, P_{MinP}, $P_{\text{MinP\_CP}}$, and P_{ILP} correspond to the PRAM size generated by heuristic MinP, MinP_CP, and the ILP formulations.

1) $\text{Dif}p1 = \frac{P_{\text{MinP}} - P_{\text{ILP}}}{P_{\text{ILP}}} \cdot 100\%$.
2) $\text{Dif}p2 = \frac{P_{\text{MinP\_CP}} - P_{\text{ILP}}}{P_{\text{ILP}}} \cdot 100\%$.
3) $\text{Dif}p3 = \frac{P_{\text{MinP}'} - P_{\text{ILP}}}{P_{\text{ILP}}} \cdot 100\%$.

We can conclude from Tables IV–VI that all the proposed offline heuristics perform better than the simple heuristics. Besides, solutions of the offline heuristics are all within 10% of the optimal solutions on average while dramatically reducing the solving time. Performances of the online heuristics are all within 20% of the optimal solutions.

C. Parameter Sensitivity Study

In this section, we present the parameter sensitivity study of the proposed heuristics. We conduct experiments on the 40-task set. For each heuristic, there are three inputs. By fixing two inputs and changing the third one, we can derive different outputs. Note that in Fig. 5(a) no solutions can be derived from MinE, MinE_CP, and MinE$'$ when $Dt = 80$. The same scenario appears in Figs. 5(b) and 6(a). From Figs. 5–7, the following conclusions can be drawn.

1) The offline heuristics perform better than the simple heuristics and are close to the optimal ILP formulations, while the online heuristics behave worst among all heuristics.
2) In each group of experiments, our method proceeds by fixing two parameters and changing the remaining one. A notable phenomenon is that, as the free parameter increases, the result decreases.
3) The parameters have different impacts in the experiments. As depicted in Fig. 7(a), Dt has the largest impact on P. This reason is obvious: the hybrid memory consists of PRAM and DRAM. A small DRAM will require a large PRAM to accommodate all the tasks. For Fig. 6(c), the hybrid memory is large enough to hold the tasks, so the energy consumption Et will obviously impact N.

VI. RELATED WORK

As the size of main memory continually increases, more and more energy will be consumed by the main memory subsystem. The conventional DRAM-based main memory has contributed to as much as 40% of the total system power on some server machines [1]. Emerging NVMs such as STT-RAM and PRAM are promising candidates to be employed as main memory. Take PRAM for example: it has the advantages of nonvolatility, excellent energy economy, and high density, and the disadvantages of limited write endurance and long access

TABLE IV
COMPARISON OF MINE′, MINE_CP, MINE, AND ILP

Task	Parameter setting			MinE′	MinE_CP	MinE	ILP		Dife1	Dife2	Dife3
no.	Dt	Pt	Nt	$E_{\text{MinE}'}$	$E_{\text{MinE\_CP}}$	E_{MinE}	E_{ILP}	Time (s)	(%)	(%)	(%)
40	100	100	75	4597	4297	4167	4023	7	3.6	6.8	14.3
40	100	100	100	4354	4187	4090	3910	15	4.6	7.1	11.4
40	100	100	150	4137	4043	3943	3675	34	7.3	10	12.6
40	150	150	150	4003	3903	3791	3587	12	5.7	8.8	11.6
60	150	150	200	6487	6402	6144	5742	103	7	11.5	13.0
60	200	200	150	6245	6184	5727	5475	87	4.6	12.9	14.1
60	150	150	150	6795	6502	6379	5985	43	6.6	8.6	13.5
60	200	200	200	5993	5824	5606	5314	92	5.5	9.6	12.8
80	200	200	300	8643	8328	7692	7403	25	3.9	12.5	16.7
80	250	250	250	8502	8301	7879	7357	11	7.1	12.8	15.6
80	200	200	200	8943	8549	8208	7765	>7200	5.7	10.1	15.2
80	250	250	200	8449	8302	7874	7513	465	4.8	10.5	12.5
100	200	200	300	10 987	10 749	10 154	9634	1864	5.6	11.6	14.0
100	200	200	400	10 574	10 346	9957	*	–	–	–	–
100	250	250	300	10 112	10 043	9646	9023	>7200	6.9	11.3	12.1
100	250	250	400	9986	9787	9513	*	–	–	–	–
Ave. (%)	–	–	–	–	–	–	–	–	5.6	10.3	13.5

TABLE V
COMPARISON OF MINN′, MINN_CP, MINN, AND ILP

Task	Parameter setting			MinN′	MinN_CP	MinN	ILP		Difn1	Difn2	Difn3
no.	Dt	Pt	Et	$N_{\text{MinN}'}$	$N_{\text{MinN\_CP}}$	N_{MinN}	N_{ILP}	Time (s)	(%)	(%)	(%)
40	150	150	3600	135	131	123	116	267	6.0	12.9	16.4
40	100	100	3800	118	118	112	105	34	6.7	12.4	12.3
40	100	100	4000	96	90	88	81	22	8.6	11.1	18.5
40	150	100	4000	89	85	81	76	14	6.6	11.8	17.1
60	150	150	5800	215	211	200	189	65	5.8	11.6	13.8
60	150	150	6000	167	158	156	143	14	7.7	10.5	16.8
60	200	200	5400	197	195	191	174	32	9.8	12.1	13.2
60	200	200	5600	153	150	143	134	>7200	6.7	11.9	14.2
80	200	200	7500	325	311	301	289	153	4.2	7.6	12.5
80	200	200	8000	200	194	190	178	43	6.7	9.0	12.4
80	250	250	7500	214	210	198	187	13	5.9	12.3	14.4
80	250	250	8000	149	142	135	128	157	5.5	10.9	16.4
100	250	250	9000	280	273	260	245	>7200	6.1	11.4	14.3
100	250	250	9500	189	186	178	165	435	7.9	12.7	14.5
100	300	300	9000	254	245	231	*	–	–	–	–
100	300	300	9500	177	173	163	*	–	–	–	–
Ave. (%)	–	–	–	–	–	–	–	–	6.7	11.3	14.8

latency. Limited by these disadvantages, PRAM is not suitable to be directly employed as main memory.

Active investigations on the hybrid main memory have been conducted recently. From different levels, the authors have proposed efficient management mechanisms and evaluated their performances. From operating system level, Mogul *et al*. [15] showed how to manage hybrid memory in order to benefit from the ideal characteristics and hide the nonideal attributes of the two parts of hybrid memory. Dhiman *et al*. [14] proposed a hybrid hardware/software solution to manage the hybrid memory which is referred to as PDRAM. Due to the limitation of PRAM write endurance, they introduced a cost-efficient book that stores the write frequency to PRAM. Additionally, they presented an operating-system-level page manager that exploits the write frequency information provided by the hardware to perform wear leveling on all the PRAM pages. They could save 37% energy at a negligible overhead compared to DRAM architecture and behave better on energy and performance efficiency over homogeneous PRAM main memory.

Qureshi *et al*. [13] studied PRAM-based main memory integrated with a small DRAM buffer. They showed that DRAM

TABLE VI
COMPARISON OF MINP′, MINP_CP, MINP, AND ILP

Task	Parameter setting			MinP′	MinP_CP	MinP	ILP		Difp1	Difp2	Difp3
no.	Dt	Et	Nt	$P_{\mathrm{MinP'}}$	$P_{\mathrm{MinP\_CP}}$	P_{MinP}	P_{ILP}	Time (s)	(%)	(%)	(%)
40	100	3500	150	137	133	130	123	6	5.7	8.1	11.4
40	100	3800	150	108	107	102	94	15	8.5	13.8	14.9
40	100	4000	150	100	99	91	86	432	5.8	15.1	16.3
40	150	3500	150	116	112	105	101	54	4.0	10.9	14.9
60	150	5500	200	215	210	202	189	23	6.9	11.1	13.8
60	200	5500	200	130	123	122	113	156	8.0	8.8	15.0
60	150	6000	200	170	170	166	151	45	9.9	12.6	12.6
60	200	6000	200	109	107	107	99	654	8.1	8.1	10.1
80	200	7500	300	213	206	199	188	>7200	5.9	9.6	13.3
80	250	8000	300	145	145	140	130	>7200	7.7	11.5	11.5
80	250	7500	200	176	173	169	156	327	8.3	10.9	12.8
80	200	8000	200	203	199	189	176	32	7.4	13.1	15.3
100	200	9000	300	285	284	284	265	563	7.2	7.2	7.5
100	200	9500	350	220	218	213	201	>7200	6.0	9.5	11.4
100	250	9000	300	271	260	251	*	–	–	–	–
100	250	9500	350	210	210	201	*	–	–	–	–
Ave. (%)	–	–	–	–	–	–	–	–	7.2	10.7	12.8

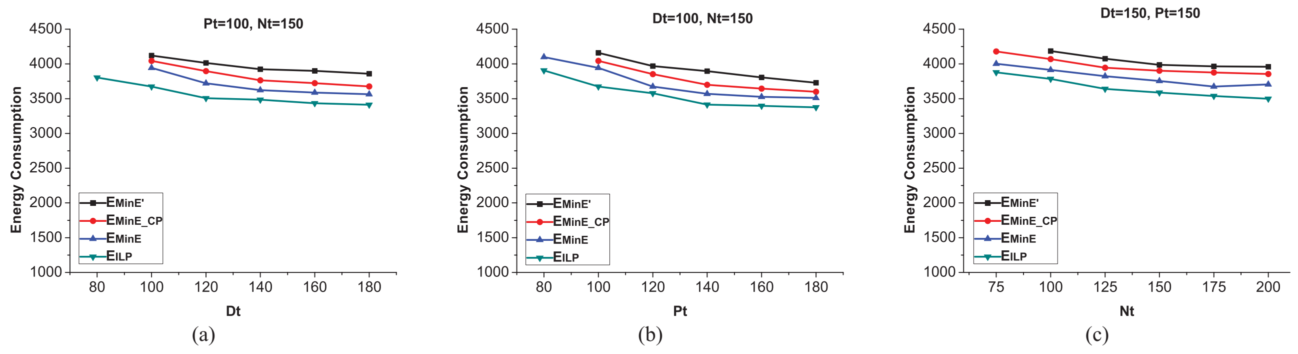


Fig. 5. Parameter sensitivity study of MinE on the 40-task set. (a) Changing Dt. (b) Changing Pt. (c) Changing Nt.

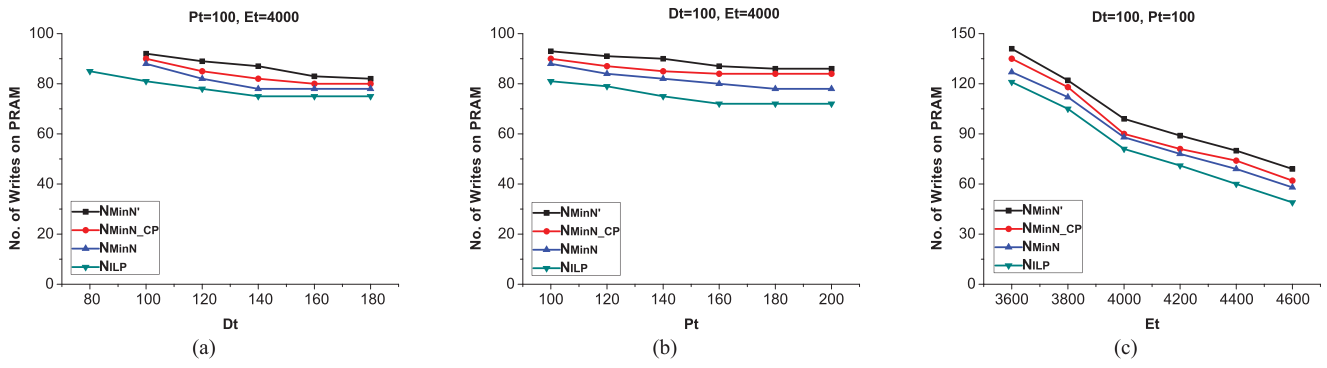


Fig. 6. Parameter sensitivity study of MinN on the 40-task set. (a) Changing Dt. (b) Changing Pt. (c) Changing Et.

buffer with only 3% of PRAM size can effectively bridge the speed gap between PRAM and DRAM. To reduce the write traffic on PRAM, they proposed three techniques: lazy write, line level writeback, and page level bypass. These techniques could significantly extend the average life expectancy of PRAM. In addition to reducing write traffic, a low-overhead technique—fine grain wear leveling—was proposed to make the wearout uniform among all lines in a page. Chen *et al.* [25] discussed the optimization techniques for PRAM-based main memory on database systems. They presented analytic metrics for PRAM endurance, energy, and latency. Furthermore, they demonstrated that current approaches such as B^+-trees and Hash-joins are suboptimal for PRAM and proposed improved algorithms to reduce both the execution time and energy dissipation while increasing write endurance on PRAM.

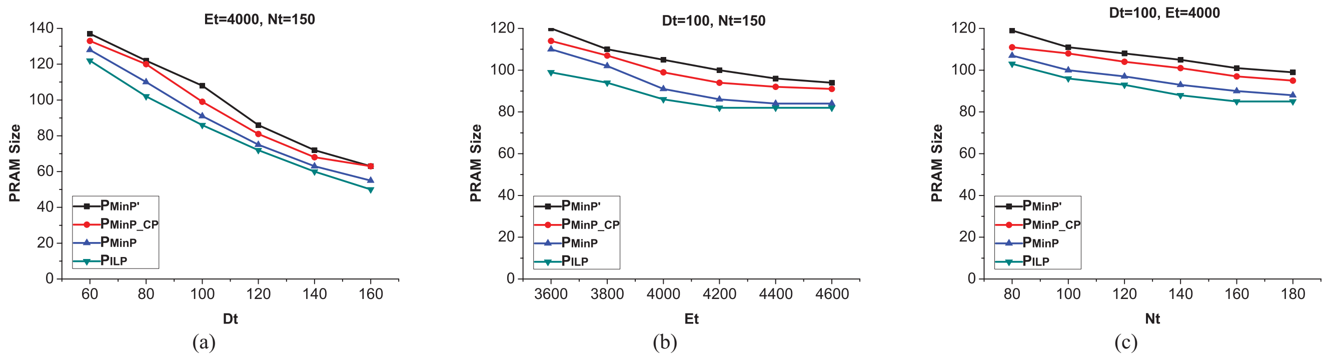


Fig. 7. Parameter sensitivity study of MinP on the 40-task set. (a) Changing Dt. (b) Changing Et. (c) Changing Nt.

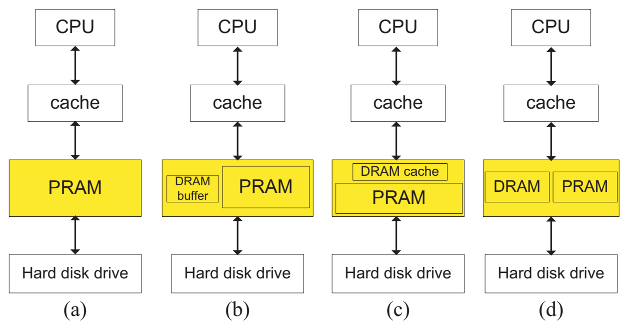


Fig. 8. Architecture designs of integrating PRAM into main memory [24]. (a) Pure PRAM is used as main memory. (b) Hybrid memory manages a small DRAM buffer by software. (c) Small DRAM is integrated in the hybrid memory as a transparent hardware cache. (d) Sizes of PRAM and DRAM are of the same level.

In modern computer systems, a great portion of the main memory is used for cache to hide disk access latency. Many conventional caching algorithms, such as least recently used (LRU), low inter-reference recency set, etc., have been proposed and showed good performance. But these algorithms are only suitable for DRAM-based main memory whose access latency is uniform and write endurance is unlimited. For the new hybrid main memory consisting of PRAM and DRAM, Seok *et al.* [26] designed an LRU-based page caching algorithm which adopts page monitoring and migration schemes to keep read-bound access pages to PRAM. This algorithm could minimize the write access of PRAM, thereby extending its life expectation while maintaining high cache hit ratio.

Fig. 8 illustrates the architectures integrating PRAM into the main memory system in recent investigations [13], [20], [25], [27]. The main difference is whether a transparent or software-controlled DRAM buffer is contained in the main memory. Fig. 8(a) is proposed in [20], which directly employs PRAM as main memory. By smart optimizations, the authors could reduce application execution time on PRAM to within a factor of 1.2 compared with DRAM-based main memory. Both [13] and [27] integrate a small DRAM with PRAM with the purpose of keeping frequently accessed data in the DRAM buffer to improve performance and reduce PRAM writes. The difference is that (b) manages the DRAM buffer by software [27] while (c) controls the DRAM buffer as another level of transparent hardware cache [13]. Recent work [25] considered an abstract framework that captures all (a)–(c) for different algorithm purposes. In [28] and this paper, a hybrid DRAM and PRAM main memory is adopted as shown in Fig. 8(d). Considering their different characteristics, we propose heuristics for optimization according to the different objectives.

VII. Conclusion

In this paper, we studied the task allocation problem on the hybrid main memory composed of PRAM and DRAM. We exploited the energy efficiency of PRAM and the long write endurance of DRAM. The objectives were to minimize the energy consumption, number of writes on PRAM, and the PRAM size. Two sets of heuristics to solve these problems were proposed. The experimental results showed that compared with the simple heuristics, the proposed offline heuristics perform better. Moreover, the offline heuristics could obtain near-optimal solutions but consume much less time compared with the ILP formulations.

References

[1] L. A. Barroso and U. Hölzle, “The case for energy-proportional computing,” *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[2] G. W. Burr, M. J. Breitwisch, M. Franceschini, D. Garetto, K. Gopalakrishnan, B. Jackson, B. Kurdi, C. Lam, L. A. Lastras, A. Padilla, B. Rajendran, S. Raoux, and R. S. Shenoy, “Phase change memory technology,” *J. Vac. Sci. Technol. B*, vol. 28, no. 2, pp. 223–262, 2010.

[3] S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y.-C. Chen, R. M. Shelby, M. Salinga, D. Krebs, S.-H. Chen, H.-L. Lung, and C. H. Lam, “Phase-change random access memory: A scalable technology,” *IBM J. Res. Develop.*, vol. 52, nos. 4–5, pp. 465–479, 2008.

[4] M. Hosomi, H. Yamagishi, T. Yamamoto, K. Bessho, Y. Higo, K. Yamane, H. Yamada, M. Shoji, H. Hachino, C. Fukumoto, H. Nagao, and H. Kano, “A novel nonvolatile memory with spin torque transfer magnetization switching: Spin-RAM,” in *Proc. IEEE IEDM*, 2005, pp. 459–462.

[5] Y. Huai, “Spin-transfer torque MRAM (STT-MRAM): Challenges and prospects,” *AAPPS Bullet.*, vol. 18, no. 6, pp. 350–355, 2008.

[6] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, “A novel architecture of the 3D stacked MRAM L2 cache for CMPs,” in *Proc. 15th Int. Symp. High Perform. Comput. Arch.*, 2009, pp. 239–249.

[7] J. Li, C. Xue, and Y. Xu, “STT-RAM based energy efficiency hybrid cache for CMPs,” in *Proc. 19th IEEE/IFIP VLSI Syst. Chip Conf.*, Oct. 2011, pp. 31–36.

[8] J. Hu, C. Xue, Q. Zhuge, W. Tseng, and E. H.-M. Sha, “Toward energy efficient hybrid on-chip scratch pad memory with non-volatile memory,” in *Proc. Conf. Des., Autom. Test Eur.*, 2011, pp. 1–6.

[9] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li, and Y. Chen, “Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement,” in *Proc. 45th Des. Autom. Conf.*, 2008, pp. 554–559.

[10] J. Hu, C. Xue, W.-C. Tseng, Y. He, M. Qiu, and E. H.-M. Sha, "Reducing write activities on non-volatile memories in embedded CMPs via data migration and recomputation," in *Proc. 47th Des. Autom. Conf.*, 2010, pp. 350–355.

[11] Y. Huang, T. Liu, and C. Xue, "Register allocation for write activity minimization on non-volatile main memory," in *Proc. 16th Asia South Pacific Des. Autom. Conf.*, 2011, pp. 129–134.

[12] L. Shi, C. Xue, and X. Zhou, "Cooperating write buffer cache and virtual memory management for flash memory based systems," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp.*, Apr. 2011, pp. 147–156.

[13] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," in *Proc. 36th Int. Symp. Comput. Arch.*, 2009, pp. 24–33.

[14] G. Dhiman, R. Ayoub, and T. Rosing, "PDRAM: A hybrid PRAM and DRAM main memory system," in *Proc. 46th Des. Autom. Conf.*, 2009, pp. 664–669.

[15] J. C. Mogul, E. Argollo, M. Shah, and P. Faraboschi, "Operating system support for NVM+DRAM hybrid main memory," in *Proc. 12th Workshop Hot Topics Operat. Syst.*, 2009, pp. 14–21.

[16] X. Dong, N. P. Jouppi, and Y. Xie, "PCRAMsim: System-level performance, energy, and area modeling for phase-change RAM," in *Proc. Int. Conf. Comput.-Aided Des.*, 2009 pp. 269–275.

[17] *CACTI* [Online]. Available: http://quid.hpl.hp.com:9081/cacti/

[18] C. J. Xue, Y. Zhang, Y. Chen, G. Sun, J. J. Yang, and H. Li, "Emerging non-volatile memories: Opportunities and challenges," in *Proc. 7th IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codes. Syst. Synth.*, Oct. 2011, pp. 325–334.

[19] T. Tanzawa and T. Tanaka, "A dynamic analysis of the dickson charge pump circuit," *IEEE J. Solid-State Circuits*, vol. 32, no. 8, pp. 1231–1240, Aug. 1997.

[20] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable DRAM alternative," in *Proc. 36th Int. Symp. Comput. Arch.*, 2009, pp. 2–13.

[21] M. R. Garey and D. S. John, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman, 1979.

[22] A. L. Buchsbaum, H. Karloff, and C. Kenyon, "OPT versus LOAD in dynamic storage allocation," in *Proc. 35th Annu. ACM Symp. Theory Comput.*, 2003, pp. 556–564.

[23] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Proc. 4th Annu. IEEE Int. Workshop Workload Charact.*, 2001, pp. 3–14.

[24] *Lingo* [Online]. Available: http://www.lindo.com/index.php?option=com_content&view=article&id=2&Itemid=10

[25] S. Chen, P. B. Gibbons, and S. Nath, "Rethinking database algorithms for phase change memory," in *Proc. 5th Biennial Conf. Innov. Data Syst. Res.*, 2011, pp. 21–31.

[26] H. Seok, Y. Park, and K. H. Park, "Migration based page caching algorithm for a hybrid main memory of DRAM and PRAM," in *Proc. 26th ACM Symp. Appl. Comput.*, 2011, pp. 595–599.

[27] J. Condit, E. B. Nightingale, C. Frost, E. Ipek, B. C. Lee, D. Burger, and D. Coetzee, "Better I/O through byte-addressable, persistent memory," in *Proc. 22nd ACM Symp. Operat. Syst. Principles*, 2009, pp. 133–146.

[28] T. Liu, C. Xue, Y. Zhao, and M. Li, "Power-aware variable partitioning for DSPs with hybrid PRAM and DRAM main memory," in *Proc. 48th Des. Autom. Conf.*, 2011, pp. 405–410.

Wanyong Tian received the B.E. degree from the Department of Computer Science and Technology, Northwest University, Xi'an, China, in 2007. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, China.

His current research interests include algorithm design and analysis and embedded systems.

Yingchao Zhao received the B.E. and Ph.D. degrees from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2004 and 2009, respectively.

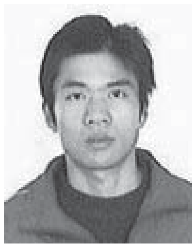
She is currently a Lecturer with the Department of Computer Science, Caritas Institute of Higher Education, Hong Kong. Her current research interests include algorithmic game theory, algorithm designs, and computational complexity analysis and scheduling.

Liang Shi received the B.S. degree in computer science from the Xi'an University of Post and Telecommunication, Xi'an, China, in 2008. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, University of Science and Technology of China, Hefei, China.

His current research interests include embedded systems and emerging non-volatile memory technology.

Qingan Li received the B.E. degree from the Computer School of Wuhan University, Wuhan, China, in 2008, where he is currently pursuing the Ph.D. degree.

His current research interests include compiler optimization and program analysis for embedded systems.



Jianhua Li received the B.S. degree from the Department of Computer Science and Technology, Anqing Teachers' College, Anhui, China, in 2007. He is currently pursuing the Ph.D. degree with the School of Computer Science, University of Science and Technology of China, Hefei, China.

His current research interests include computer architecture, multi-core memory system, and on-chip networks.

Chun Jason Xue (M'12) received the B.E. degree from the Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, in 1997, and the M.E. and Ph.D. degrees from the Department of Computer Science, University of Texas at Dallas, Dallas, in 2003 and 2007, respectively.

He is currently an Assistant Professor with the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong. His current research interests include optimization for parallel embedded systems, optimization for DSPs with VLIW or multi-core architecture, hardware and software co-design.

Minming Li received the B.E. and Ph.D. degrees from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2002 and 2006, respectively.

He is currently an Assistant Professor with the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong. His current research interests include algorithm design and analysis in wireless networks and energy efficient scheduling, combinatorial optimization and computational economics.

Enhong Chen (SM'12) received the Ph.D. degree from the University of Science and Technology of China (USTC), Hefei, China, in 1996.

He is currently a Professor with the School of Computer Science and Technology, USTC. His current research interests include data mining, personalized recommendation systems, and web information processing.