Journal of Computational Science 26 (2018) 402–408

Contents lists available at ScienceDirect

Journal of Computational Science

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

Cost-aware optimal data allocations for multiple dimensional heterogeneous memories using dynamic programming in big data

Hui Zhao [a], Meikang Qiu [b,*,1], Min Chen [c], Keke Gai [b]

[a] Software School, Henan University, Kaifeng, Henan 475000, China
[b] Department of Computer Science, Pace University, New York City, NY 10038, USA
[c] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China

ARTICLE INFO

Article history:
Received 26 April 2016
Received in revised form 21 May 2016
Accepted 12 June 2016
Available online 16 June 2016

Keywords:
Heterogeneous memories
Dynamic programming
Big data
Optimal approach
Data allocation
High performance

ABSTRACT

Multiple constraints in SPMs are considered a problem that can be solved in a nondeterministic polynomial time. In this paper, we propose a novel approach solving the data allocations in multiple dimensional constraints. For supporting the approach, we develop a novel algorithm that is designed to solve the data allocations under multiple constraints in a polynomial time. Our proposed approach is a novel scheme of minimizing the total costs when executing SPM under multiple dimensional constraints. Our experimental evaluations have proved the adaptation of the proposed model that could be an efficient approach of solving data allocation problems for SPMs.

1. Introduction

The dramatically booming requirements of high performance embedded systems have been driving multi-core system designs for big data processing in recent years [1]. As one of the popular approaches supporting software-controlled on-chip memories, *Scratch-Pad Memory* (SPM) has been broadly implemented in a variety of industries while the data volume becomes large [2,3]. An important benefit of using SPMs is reducing the total operating cost by allocating data for having less hardware overhead and more data controls [4]. The data accesses are controlled by a program stored in SPMs, which can process big data by dynamic manipulations. For gaining an efficient data processing, the critical issue of the cost optimizations is designing an approach of achieving optimal data allocations. Focusing on this issue, this paper proposes a novel approach named *Optimal Data Allocation in Heterogeneous Memories* (ODAHM) model, which sights at minimizing the total costs of SPMs by organizing and controlling the data allocations.

The recent development of the heterogeneous memories in big data mainly addressed the minimizations of the costs by balancing different dimensional consumptions, such as energy, performance, and time constraints [5–7]. The operating principle is that the input data are mapped onto difference spaces in SPMs. The challenging aspect of using this mechanism is that it is hard to guarantee real-time performance within the desired cost scope due to the complicated allocation processes [8,9]. The performance of heterogeneous memories can hardly reach the full-performance unless the efficient data allocation approach is applied. This restriction will become even more challenging when the data size turns into larger and the amount of the cost dimensions gets greater. Therefore, we consider this restriction a critical issue in improving heterogeneous memories in SPM and propose our approach to solve the data allocation problems with multiple cost dimensions.

The proposed model, ODAHM, defines the main operating procedure and manipulative principle, which deems multiple constraints influencing the costs while the data are allocated to memories. We modelize the operation of data allocations into a few steps, which include defining constraint dimensions, mapping the costs for each constraint, and producing optimal data allocation scheme according to the inputs. Implementing ODAHM can enable an advanced data allocation strategy for SPM since more impact factors can be involved for saving costs.

* Corresponding author. Tel.: +1 9147733253.
E-mail addresses: zhh@henu.edu.cn (H. Zhao), mqiu@pace.edu (M. Qiu), minchen2012@hust.edu.cn (M. Chen), kg71231w@pace.edu (K. Gai).

[1] This work was supported in part by the National Science Foundation under Grants CNS-1457506 and NSF CNS-1359557 (Prof. M. Qiu).

https://doi.org/10.1016/j.jocs.2016.06.002
1877-7503/

For reaching this goal, we propose an algorithm, *Optimal Multiple Dimensional Data Allocation* (OMDDA), which is designed to solve N-dimensional constraints data allocation problem. This algorithm uses dynamic programming and produces the optimal solution synchronously under a few constraints. The operation principle of this algorithm is using a deterministic approach to point at the cost caused by the corresponding constraints, which are mapped in the table. We produce local optimal solutions for each constraint and generate a global optimal solution deriving from the outcomes of the local optimal solutions.

Main contributions of this paper are threefold:

1 We propose a novel approach for solving the big data allocation problem with multiple dimensional constraints for heterogeneous memories in SPMs, which is a NP-hard problem. This approach supports the designs of the complex SPM systems considering all influencing elements.
2 This paper proposes a novel algorithm to solve the problem of data allocations in SPMs, which can be applied in solving other problems having the similar big data scenarios.
3 The proposed algorithm offers a method that can produce global optimal solutions that are executed by mapping local optimal solutions and using dynamic programming.

The remainder of this paper follows the order below: Recent related work in data allocations in SPM are reviewed in Section 2. We also present a motivational example in Section 3. Main concepts and the proposed model are exhibited in Section 4. In addition, main algorithms are given in Section 5. Moreover, we display a number of experimental results in Section 6. Finally, conclusions are stated in Section 7.

2. Related work

We have reviewed recent research work of data allocations in heterogeneous memories from different perspectives in this section, such as reducing energy costs and increasing working efficiency. The increasing demands of the Internet-based applications have driven a higher-level of memory requirements [10–13]. First, it has been proved that the power leakage consumption is a critical issue for heterogeneous memories with large scaled transistors due to the different memory capacities [14,15]. This has resulted in obstacles in applying heterogeneous memories since allocating data to different memories needs to assess various costs taken place during a few phases, such as data processing and data moves.

Recent research has been focusing on reducing costs using different techniques in various fields. Addressing the physical operating environment, a proposed approach was using temperature-aware data allocations in order to adjust the workload distributions between cache and SPM [16]. The optimization method was applying an energy-aware loop scheduling algorithm. The energy could be saved when the loop scheduling was improved by retiming-based methods. Meanwhile, considering the operating system environment, an approach was proposed to have an integrated real-time/non-real-time task execution environment for separately running the real-time or non-real-time nodes on OS and Linux [17,18]. This approach could reduce the total costs through a selective execution. However, the proposed approaches above mainly focused on software level optimizations, even though the energy consumptions and other costs can be managed or controlled by a software-based solution. There is little relationship with the manner of data allocations to memories.

Moreover, the volatile-related features of the memories provide optimizations with optional choices. One advantage of using heterogeneous memories is that the volatile memories can be combined with non-volatile memories, which can lead to saving energy [19]. The benefits of using non-volatile memories include low power leakage and high density, but the unbalanced usages and inefficient tasks scheduling [20]. A solution [21] was produced to being an efficient scheduling method by optimizing the global data allocations that are based on the regional data allocation optimizations. This approach can be further improved when more elements are considered. Our proposed approach can solve the problem covering multiple elements.

Next, for further optimizations, other constraints are also considered, such as reducing latency, increasing success probabilities, and hardware capacities. An approach [22] solving multidimensional resource allocations has been proposed, which combined a few low-complexity sub-optimal algorithms. This mechanism used the principle of integrating a few suboptimal solutions to form an adoptable solution. Our proposed scheme distinguishes from this approach since we generate global optimal solutions deriving from the local optimal solutions in different dimensions.

In addition, a genetic algorithm was proposed for data allocations of hybrid memories by configuring SRAM, MRAM, and Z-RAM [23]. This algorithm was designed to enhance the performance of SPM by dynamically allocate data to different memories having various usage constraints. Combing this genetic algorithm with dynamic programming, the memory costs can be reduced in terms of power consumption and latency [5]. Despite the memory performance can be increased, the implementations can only process the limited constraint dimensions due to the time complexity restrictions.

Another research direction focused on reducing costs on *Phase-Change Memory* (PCM) that could consist of multiple memories on different levels of the memory hierarchy [24,25]. The methods used in PCM are similar to heterogeneous memories in SPM. One approach was partitioning and allocating data to the multitasking systems depending on the memory types [26,27]. However, the solutions based on this research direction could hardly solve the problem of the computation complexity when aiming gain an optimal solution.

Furthermore, allocation optimizations have also been applied in saving energy consumptions, such as energy storage within the distributed networks [28]. Another approach of reducing the complexity was using the weighted sum of the usage from multiple channels [29]. Nonetheless, most optimizations could solve partial of the complexity problem within the constraints.

In summary, our approach is different from most prior research achievements. The proposed approach is a scheme of producing optimal solutions for multiple dimensional heterogeneous memories. We further expand the condition constraint from a limited amount to multiple dimensions.

3. Motivational example

In this section, we give a simple example of using our proposed approach to process data allocations to heterogeneous memories. In this example, assume that there are 4 ZRAM and 2 SRAM available. Main memory is always available for data. There are 7 input data that required different read and write accesses. The output is a data allocation plan that requires the minimum total cost. Table 1 represents the cost differences of data allocations to each memory.

Table 2 displays the number of the memory accesses, including *Read* and *Write*. 7 input data are A, B, C, D, E, F, and G. For instance, data A require 5 reads and 2 writes, according to the table. Our example's initial status is to allocate $A \rightarrow Main$, $B \rightarrow SRAM$, $C \rightarrow Main$, $D \rightarrow ZRAM$, $E \rightarrow Main$, $F \rightarrow Main$, and $G \rightarrow Main$.

Based on the conditions given by Tables 1 and 2, we map the costs of data allocations to heterogeneous memories in Table 3. For

Table 1
Cost differences for data allocations to heterogeneous memories.

Operation		SRAM	ZRAM	Main
Read		2	3	75
Write		2	7	75
Move	SRAM	0	10	72
	Z-RAM	6	0	72
	Main	70	76	0

Table 2
The number of the memory accesses.

Data	Read	Writes
A	5	2
B	10	8
C	8	5
D	4	4
E	2	10
F	3	15
G	15	4

Table 3
Mapping the costs for heterogeneous memories.

Data	SRAM	ZRAM	Main
A	84	105	525
B	36	96	1422
C	96	135	975
D	22	40	672
E	94	152	900
F	106	190	1350
G	108	149	1425

Table 4
Allocation cost for data A.

525 (Main)	105 (ZRAM)
84 (SRAM)	

Table 5
A partial D Table.

A		0	1	2	3	4	ZRAM
SRAM	0	525	105				
	1	84					
	2						

example, as shown in the table, data A costs 84 when it is allocated to SRAM, switched from Main memory. We are going to use Table 3 to calculate the total cost after the allocations. For obtaining the final optimal data allocation plan, we will produce a *D Table* showing the optimal data allocation plan. The generation process is given as follows, which consists of a few steps.

First, we only consider one input data A. Deriving from Table 3, we produce a four-cell grid that is shown Table 4. As shown in this grid, we mark the memory to the corresponding cost. Meanwhile, the D Table generation starts from adding Table 4 into the D Table. Table 5 represents the manner of the data insertions in D Table. As displayed in the table, A refers to the new added data A. In the first row, 0–4 means the availability of ZRAM. Correspondingly, 0–2 means the availability of SRAM in the second column. For instance, 525 is associated to (0, 0), which means neither ZRAM nor SRAM is available for data A. The cost of data A is 525 in this situation, which is allocated to Main memory.

Next, we consider adding data B to continue generating D Table. Fig. 1 represents the process of adding Data B into D Table. We mark the table showing the partial Table D with data A cost as DAC, table showing data B costs as DBC, and partial D Table for data A and B as DB. Since the maximum cases for each cell is 3, we alternative

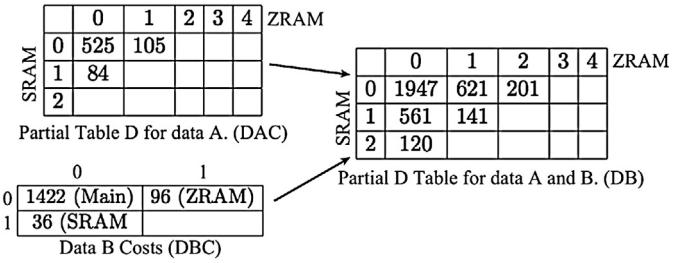

Fig. 1. Generating partial D Table for data A and B deriving from Tables 3 and 5.

the allocation plan with the lowest cost in each cell. For calculating the values, we use the memory availability values as coordinates. For example, DB(1, 1) refers to 141 in DB; DBC(1, 0) refers to 96 (ZRAM) in DBC. Therefore, the minimum cost for each cell in DB, DB(i, j) can be gained by $min\langle$[DBC(0, 0) + DAC(i, j)], [DBC(0, 1), DAC(i, $j-1$)], [DBC(1, 0), DAC($i-1$, j)]$\rangle$. For instance, DB(1, 1) = $min\langle$[DBC(0, 0) + DAC(1, 1)], [DBC(0, 1) + DAC(1, 0)], [DBC(1, 0) + DAC(0, 1)]$\rangle$ = $min\langle$[1422 + 0], [36 + 105], [96 + 84]$\rangle$ = $min\langle$1422, 141, 180$\rangle$ = 141.

Moreover, we add all other data by using the same method. A completed D Table is shown in Table 6. A, B, C, D, E, F, and G refer to the order of adding data. We bold the number that is corresponding to the optimal solution in this example. According to the table, the lowest data allocation cost is 1143 after data G are added to the table. Based on this finding, we move backward obtain all data allocations, which are 1143 → 994 → 888 → 736 → 696 → 561 → 525. We can gain the data allocation plan for each data from the calculation method mentioned above. Therefore, the data allocation plan is A → Main, B → SRAM, C → ZRAM, D → ZRAM, E → ZRAM, F → SRAM, G → ZRAM.

The detailed mechanism of our approach is given in Sections 4 and 5.

Table 6
D Table. The optimal data allocations are bolded.

		0	1	2	3	4	ZRAM
A							
SRAM	0	**525**	105				
	1	84					
	2						
B							
SRAM	0	1947	621	201			
	1	**561**	141				
	2	120					
C							
SRAM	0	2292	1596	756	336		
	1	1536	**696**	276			
	2	657	237				
D							
SRAM	0	3594	2268	1428	796	376	
	1	2208	1368	**736**	316		
	2	1329	697	277			
E							
SRAM	0	4494	3168	2328	1580	948	
	1	3108	2268	1520	**888**	468	
	2	2229	1462	830	410		
F							
SRAM	0	5844	4518	3358	2518	1770	
	1	4458	3274	2434	1686	1054	
	2	3214	2374	1626	**994**	574	
G							
SRAM	0	7269	5943	4667	3507	2667	
	1	5883	4607	3423	2583	1835	
	2	4566	3363	2523	1775	**1143**	

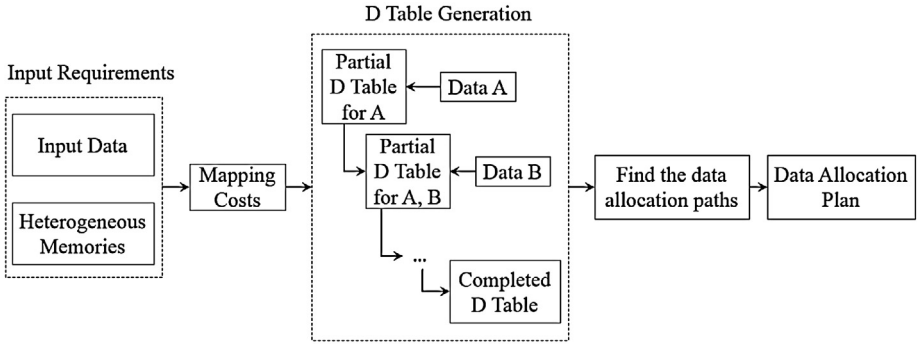

Fig. 2. Operating process of the proposed ODAHM model.

4. Concepts and the proposed model

We illustrate our proposed model by defining the main target problem and concepts used in the model.

4.1. System definition and problem statement

We assume that the data partition process of the input data completes before the data allocations. For generating the allocation plan, the target cost for saving purpose is determined and the information is available, such as computing cost per Read or per Write. The research problem is defined by the following Definition 1.

Definition 1. [Minimizing cost of data allocation on heterogeneous memories (MC-DAHM) problem] Inputs include a chain of data, initial data locations, the number of available memories for each type, the number of the memory access for each data, the cost of each data for Read, Write, and Move. The output is a data allocation plan. The research problem is finding out the data allocation plan that can minimize the total cost.

In this problem, inputs include the number of data, initial data locations, the number of memory types, the number of available memories for each type, the number of memory accesses for each data, the cost of data operations, including Read, Write, and Move. Our aim is to produce a data allocation plan that allocate the input data to the available memories by using the minimum total cost. The total cost calculation method is given in Eq. (1).

$$\text{Cost}_{\text{total}} = \sum_{i=1}^{n} (D_i \times \text{RC}_j + D_i \times \text{WC}_j + D_i \times \text{MV}_j) \quad (1)$$

Assume that there are n input data D. As shown in Eq. (1), $\text{Cost}_{\text{total}}$ refers to the total cost of data allocations. $D_i \times \text{RC}_j$ means the Read cost for a certain data and RC_j is the Read cost for data D_i each time. $D_i \times \text{WC}_j$ means the Write cost of each data and WC_j is the Write cost of D_i at each time. Similarly, $D_i \times \text{MV}_j$ is the Move cost and MV_j is the Move cost of D_i at each time. The total cost is summing up all n input data' costs.

4.2. Optimal data allocation in heterogeneous memories (ODAHM)

In our proposed model, there are mainly three steps for generating the data allocation plan. First, when the data were loaded, the cost of each data at each memory will be calculated. Three cost operations will be calculated, including Read, Write, and Move. The results of the costs will be mapped into a table. A sample is given in Section 3 as Table 3. Second, a D Table will be generated for gaining the optimal data allocation plans by using dynamic programming. A sample of D Table is given in Table 6. The creation process of D Table is the critical component of our model. We give the mathematical formulations about D Table generations in the late part of this section and describe computing algorithm in Section 5. Finally, we find out the optimal data allocation plan once the D Table is created.

Fig. 2 represents the operating process of the proposed ODAHM model that shows three main steps. As shown in the figure, input requirements include the information about both input data and memories, which will be used for mapping each data cost at each memory. At the phase of D Table generation, we create the table by adding data in a succession manner. Assume that there are m types of memories available. We define one type of the memory as one dimension. The definition of *N-Dimensional Heterogeneous Memories* is given by Definition 2.

Definition 2. [N-dimensional heterogeneous memories] $\exists$ n types of the memory available for alternatives, we define "n" as the number of dimensions and the available memory set is called N-Dimensional heterogeneous Memories. Each memory type can have different number of memories.

Once the D Table is accomplished, we can find the data allocation paths to determine the allocation plan. The detailed method description is given in Section 5.

5. Algorithms

In this section, we propose the algorithm to generate the optimal data allocation using dynamic programming. We define a few definitions used in our algorithm, including *B Table* defined by Definition 3 and *D Table* defined by Definition 4.

Definition 3. [B Table] We use a multiple-dimensional array to describe a table that stores the cost of each data at each memory. Inputs include each data's cost when it is assigned to a memory. The output will be a multiple-dimensional array. $\exists$ j types of memory, $\{M_j\}$, and each M_j has n_j memories available. The mathematical expression is $\text{BTab[i]}\langle(M_1, nb_1), (M_2, nb_2), \ldots, (M_j, nb_j)\rangle$, which represents the cost when $data_i$ use j types of memory and the number of each memory type.

Definition 4. [D Table] We use a multiple-dimensional array to describe a table that shows the total cost of data allocations by storing all task assignment plans. Input is the B Table. The output will be a multiple-dimensional array storing all task assignments as well as their costs. $\exists$ j types of memory, $\{M_j\}$, and each M_j has n_j memories available. The mathematical expression is $\text{DTab[d]}\langle(M_1, nd_1), (M_2, nd_2), \ldots, (M_j, nd_j)\rangle$, which represents the cost of all d *data* when using j types of memory. The tuple shows the assignment plan.

Algorithm 1. Optimal multiple dimensional data allocation (OMDDA) algorithm.

Require: The B Table *BTab*
Ensure: The optimal data allocation

```
input the B Table
DTab[0] ← BTab[0]
FOR ∀ rest cell in DTab,
    /* DTab[i]⟨(M1, nd1), (M2, nd2), ..., (Mj, ndj)⟩ */
  minCost ← ∞
  FOR ∀ cell in BTab[i],
    /* BTab[i]⟨(M1, nb1), (M2, nb2), ..., (Mj, nbj)⟩ */
    IF ∃ DTab[i-1]⟨(M1, nd1 − nb1), (M2, nd2 − nb2), ..., (Mj, ndj − nbj)⟩
      sum ← BTab[i]⟨(M1, nb1), (M2, nb2), ..., (Mj, nbj)⟩ + DTab[i-1]⟨(M1, nd1 − nb1), (M2, nd2 − nb2), ..., (Mj, ndj − nbj)⟩
      IF sum<minCost
        minCost ← sum
        update the data allocation plan
      ENDIF
    ENDIF
  ENDFOR
ENDFOR
RETURN the optimal data allocation by searching the D Table according to the memory condition.
```

Algorithm 1 shows our proposed algorithm and the main phases of our algorithm include:

1 Input *B Table*.
2 Start generating D Table. We copy BTab[0] to DTab[0] to produce the first partial D Table, which represents add data[0] to D Table.
3 We add data[1] to the D Table by calculating the data allocation costs based on the partial D Table generated by the last Step 5 and partial B Table BTab[1].
4 Add all data by applying the same method used in Step 5 until the D Table is generated.
5 Find the optimal data allocation by searching the lowest cost in D Table according to the memory condition. Output the task assignment plan.

6. Experiments and the results

The *n*-dimensional dynamic programming algorithm, OMDDA, is designed to find the optimal data allocation for multiple dimensional heterogeneous memories. Our experiments accomplished comparisons among Random algorithm, Greedy algorithm, DAHS algorithm [14], and our proposed algorithm. The comparisons focused on comparing four algorithms' data allocation costs and time consumptions. We implemented our experiments on a Host that ran Windows 8.1 64 bit OS. The main hardware configuration of the Host was: Intel Core i5-4210U CPU, 8.0 GB RAM.

We configured three experimental settings for assessing the performance of the proposed algorithm. There were three mainly variables in our experiments: the number of the data kind, the data size of every kind data, and the number of the available memories.

Three experimental settings are:

- Setting 1: We configured 7 kinds of data, each kinds data has 1 M size, 2 M available SRAM and 4 M available ZRAM.
- Setting 2: We configured 8 kinds of data, each kinds data has 1 G size, 1 G available SRAM and 5G available ZRAM.
- Setting 3: We configured 10 kinds of data, each kinds data has 1 G size, 1 G available SRAM and 6 G available ZRAM.

6.1. Experimental results

Fig. 3 shows the comparison among Random algorithm, Greedy algorithm, DAHS algorithm and OMDDA algorithm under setting 1. As shown in the figure, our proposed algorithm could accomplish all tasks with requiring the lowest cost. The Random algorithm demanded the largest cost because there were great chances to allocate data to those memories inquiring bigger costs. Greedy algorithm usually needed more costs than DAHS and OMDDA but there were some chances for Greedy to produce optimal solutions. The DAHS and OMDDA algorithm have the same solution since they both are optimal data allocation algorithm.

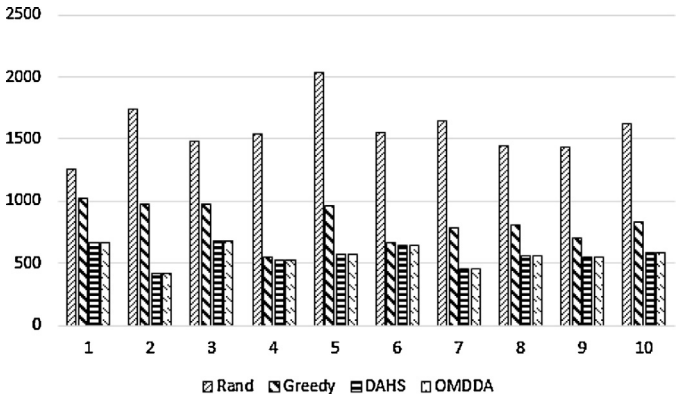

Fig. 3. Comparisons of costs under Setting 1.

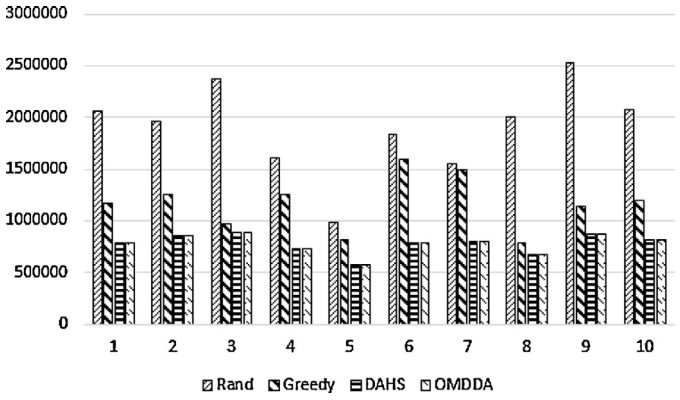

Fig. 4. Comparisons of costs under Setting 2.

Moreover, Fig. 4 shows the comparison among Random algorithm, Greedy algorithm, DAHS algorithm and OMDDA algorithm under setting 2. Our proposed scheme had a better performance than both Random and Greedy algorithms under this setting.

Fig. 5 shows the comparisons among Random, Greedy, DAHS and OMDDA algorithms under setting 3. Our proposed algorithm had an advantage in saving costs, according to the display of the

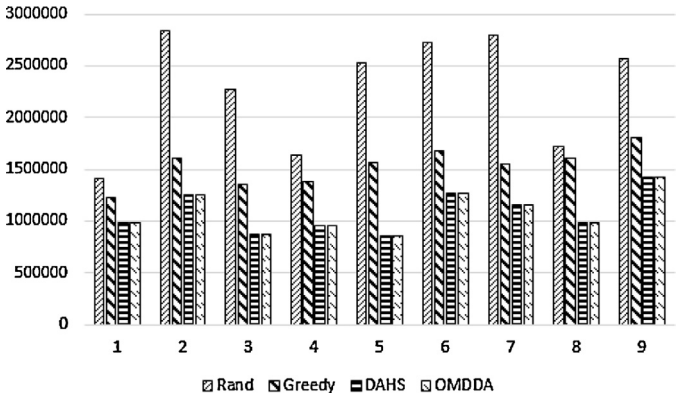

Fig. 5. Comparisons of costs under Setting 3.

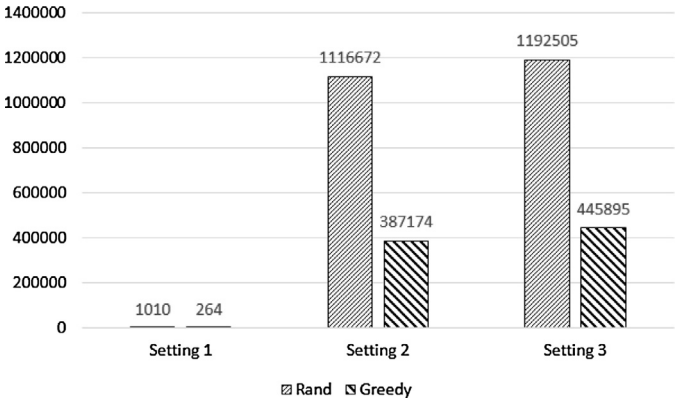

Fig. 6. Comparisons of the saved costs showing the difference between OMDDA and Random or Greedy algorithms.

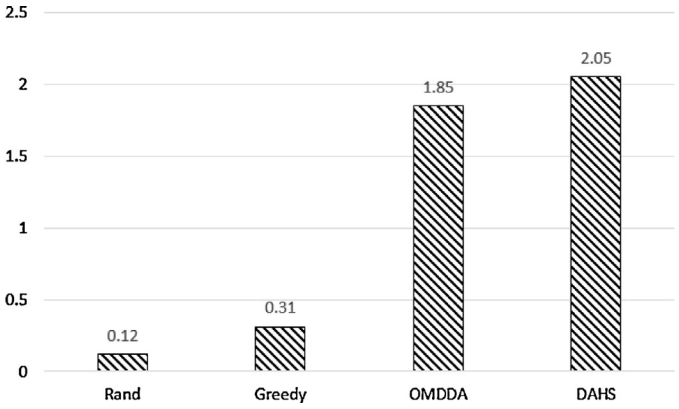

Fig. 7. Comparisons of execution time among Random, Greedy, DAHS and OMDDA algorithms.

figure. The vertical axis showed the cost levels for each algorithm. The advantage was obvious under this setting.

Fig. 6 shows how much cost OMDDA had saved comparing with Random and Greedy algorithms under each setting. As shown in Fig. 6, we could obtain the finding that the data size had a nearly positive relationship with the saved cost. It proved that our proposed scheme was suitable for the data processing in the big data context. The volume of the saved cost will become greater while the data volume becomes larger.

Fig. 7 shows the comparisons of execution time among Random algorithm, Greedy algorithm, DAHS algorithm and OMDDA algorithm. As shown in Fig. 7, we can see our proposed algorithm has less execution time than DAHS even through they have same optimal solution.

In summary, our experiments prove that the proposed algorithm, OMDDA performed better than Random algorithm and Greedy algorithm under all configured settings. Compare to DAHS, OMDDA algorithm has less consumption time to generate the optimal solution. Our findings also showed that the OMDDA had a better performance when the data volume is in a bigger size.

7. Conclusions

This paper addressed the issues of using heterogeneous memories to enable high performance big data processing and proposes an optimal solution that could generate the data allocation plans with the lowest cost. The proposed model was named as *Optimal Data Allocation in Heterogeneous Memories* (ODAHM) and the main algorithm in the model was *Optimal Multiple Dimensional Data Allocation* (OMDDA) algorithm. We had implemented experimental evaluations to examine our proposed model's performance and the collected results showed that our model had a great advantage in saving cost due to the optimal data allocations.

References

[1] N. Min-Allah, H. Hussain, S. Khan, A. Zomaya, Power efficient rate monotonic scheduling for multi-core systems, J. Parallel Distrib. Comput. 72 (1) (2012) 48–57.
[2] Y. Guo, Q. Zhuge, J. Hu, J. Yi, M. Qiu, E. Sha, Data placement and duplication for embedded multicore systems with scratch pad memory, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 32 (6) (2013) 809–817.
[3] M. Dorojevets, Z. Chen, C. Ayala, A. Kasperek, Towards 32-bit energy-efficient superconductor RQL processors: the cell-level design and analysis of key processing and on-chip storage units, IEEE Trans. Appl. Supercond. 25 (3) (2015) 1–8.
[4] G. Rodríguez, J. Touriño, M. Kandemir, Volatile STT-RAM scratchpad design and data allocation for low energy, ACM Trans. Arch. Code Optim. 11 (4) (2014) 38.
[5] M. Qiu, Z. Chen, M. Liu, Low-power low-latency data allocation for hybrid scratch-pad memory, IEEE Embed. Syst. Lett. 6 (2014) 69–72.
[6] D. Chang, C. Lin, Y. Chien, C. Lin, A. Su, C. Young, CASA: contention-aware scratchpad memory allocation for online hybrid on-chip memory management, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 33 (12) (2014) 1806–1817.
[7] M. Sabry, D. Atienza, F. Catthoor, OCEAN: an optimized HW/SW reliability mitigation approach for scratchpad memories in real-time SoCs, ACM Trans. Embed. Comput. Syst. 13 (4s) (2014) 138.
[8] B. Gaster, D. Hower, L. Howes, HRF-relaxed: adapting HRF to the complexities of industrial heterogeneous memory models, ACM Trans. Arch. Code Optim. 12 (1) (2015) 7.
[9] H. Waidyasooriya, Y. Ohbayashi, M. Hariyama, M. Kameyama, Memory allocation exploiting temporal locality for reducing data-transfer bottlenecks in heterogeneous multicore processors, IEEE Trans. Circuits Syst. Video Technol. 21 (10) (2011) 1453–1466.
[10] D. Kliazovich, P. Bouvry, S. Khan, GreenCloud: a packet-level simulator of energy-aware cloud computing data centers, J. Supercomput. 62 (3) (2012) 1263–1283.
[11] N. Fernando, S. Loke, W. Rahayu, Mobile cloud computing: a survey, Future Gener. Comput. Syst. 29 (1) (2013) 84–106.
[12] K. Gai, S. Li, Towards cloud computing: a literature review on cloud computing and its development trends., in: The 4th IEEE International Conference on Multimedia Information Networking and Security, IEEE, Nanjing, China, 2012, pp. 142–146.
[13] K. Gai, M. Qiu, H. Zhao, L. Tao, Z. Zong, Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing, J. Netw. Comput. Appl. 59 (2015) 46–54.
[14] J. Hu, C. Xue, Q. Zhuge, W. Tseng, E.H. Sha, Data allocation optimization for hybrid scratch pad memory with SRAM and nonvolatile memory, IEEE Trans. Very Large Scale Integr. Syst. 21 (6) (2013) 1094–1102.
[15] K. Gai, Z. Du, M. Qiu, H. Zhao, Efficiency-aware workload optimizations of heterogeneous cloud computing for capacity planning in financial industry, in: The IEEE 2nd International Conference on Cyber Security and Cloud Computing, IEEE, New York, NY, USA, 2015, pp. 1–6.
[16] Z. Jia, Y. Li, Y. Wang, M. Wang, Z. Shao, Temperature-aware data allocation for embedded systems with cache and scratchpad memory, ACM Trans. Embed. Comput. Syst. 14 (2) (2015) 30.
[17] H. Wei, Z. Shao, Z. Huang, R. Chen, Y. Guan, J. Tan, Z. Shao, RT-ROS: a real-time ROS architecture on multi-core processors, Future Gener. Comput. Syst. (2015).
[18] J. Li, M. Qiu, Z. Ming, G. Quan, X. Qin, Z. Gu, Online optimization for scheduling preemptable tasks on IaaS cloud systems, J. Parallel Distrib. Comput. 72 (5) (2012) 666–677.
[19] M. Qiu, L. Chen, Y. Zhu, J. Hu, X. Qin, Online data allocation for hybrid memories on embedded tele-health systems, in: 2014 IEEE International Conference on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst, IEEE, 2014, pp. 574–579.
[20] J. Hu, Q. Zhuge, C. Xue, W. Tseng, E.H. Sha, Management and optimization for nonvolatile memory-based hybrid scratchpad memory on multicore embedded processors, ACM Trans. Embed. Comput. Syst. 13 (4) (2014) 79.
[21] Q. Zhuge, Y. Guo, J. Hu, W. Tseng, C. Xue, E.H. Sha, Minimizing access cost for multiple types of memory units in embedded systems through data allocation and scheduling, IEEE Trans. Signal Process. 60 (6) (2012) 3253–3263.
[22] B. Han, M. Peng, Z. Zhao, W. Wang, A multidimensional resource-allocation optimization algorithm for the network-coding-based multiple-access relay channels in OFDM systems, IEEE Trans. Veh. Technol. 62 (8) (2013) 4069–4078.
[23] M. Qiu, Z. Chen, J. Niu, G. Quan, X. Qin, L. Yang, Data allocation for hybrid memory with genetic algorithm, IEEE Trans. Emerg. Top. Comput. (2015) 1–11.
[24] M. Qiu, M. Zhong, J. Li, K. Gai, Z. Zong, Phase-change memory optimization for green cloud with genetic algorithm, IEEE Trans. Comput. 64 (12) (2015) 3528–3540.

[25] Z. Wang, Z. Gu, Z. Shao, Optimizated allocation of data variables to PCM/DRAM-based hybrid main memory for real-time embedded systems, IEEE Embed. Syst. Lett. 6 (3) (2014) 61–64.
[26] P. Panda, N. Dutt, A. Nicolau, On-chip vs. off-chip memory: the data partitioning problem in embedded processor-based systems, ACM Trans. Des. Autom. Electron. Syst. (TODAES) 5 (3) (2000) 682–704.
[27] Y. Li, W. Dai, Z. Ming, M. Qiu, Privacy protection for preventing data over-collection in smart city, IEEE Trans. Comput. (2015) 1.
[28] Y. Zheng, Z. Dong, F. Luo, K. Meng, J. Qiu, K. Wong, Optimal allocation of energy storage system for risk mitigation of DISCOs with high renewable penetrations, IEEE Trans. Power Syst. 29 (1) (2014) 212–220.
[29] R. Fan, H. Jiang, Q. Guo, Z. Zhang, Joint optimal cooperative sensing and resource allocation in multichannel cognitive radio networks, IEEE Trans. Veh. Technol. 60 (2) (2011) 722–729.

Hui Zhao earned the B.E. and M.S. degrees from Xi'an Technology University, Shanxi and Henan University, Henan, China, in 2000 and 2008, respectively. He is a Ph.D. student at the Seidenberg School of Computer Science and Information Systems of Pace University. He is currently an associate professor in the Software School of Henan University.

Meikang Qiu earned BE and ME degrees from Shanghai Jiao Tong University, China in 1992 and 1998, respectively. He earned MS and PhD degrees in Computer Science from University of Texas at Dallas in 2003 and 2007, respectively. Currently, he is an associate professor of computer science at Pace University and adjunct professor at Columbia University. He is serving as a Chair of IEEE STC (Special Technical Community) in Smart Computing at IEEE Computer Society. He had worked at Chinese Helicopter R&D Institute, IBM, etc., for nine years. Currently, he is an IEEE senior member and ACM senior member. His research interests include cloud computing, big data storage and security, embedded systems, cyber security, hybrid memory, heterogeneous systems, mobile and sensor networks, operating systems, optimization, intelligent systems, cyber-physical systems, etc. A lot of novel results have been produced, and most of them have already been reported to the research community through high-quality journal and conference papers. He has published 4 books, 320 peer-reviewed journal and conference papers (including 150 journal articles, 170 conference papers, 40+ IEEE/ACM Transactions papers), and 3 patents. He has won ACM Transactions on Design Automation of Electrical Systems (TODAES) 2011 Best Paper Award. His paper about cloud computing has been published in JPDC (Journal of Parallel and Distributed Computing, Elsevier) and ranked No. 1 in Top Hottest 25 Papers of JPDC 2012. He has won another 6 Conference Best Paper Award (IEEE/ACM ICESS'12, IEEE GreenCom'10, IEEE EUC'10, IEEE CSE'09, IEEE CSCloud'15, IEEE BigDataSecurity'15) in recent years. Currently he is an associate editor of *IEEE Transactions on Computers* and *IEEE Transactions on Cloud Computing*. He is the General Chair of IEEE HPCC/ICESS/CSS 2015, IEEE CSCloud 2015 (Cyber Security and Cloud Computing), and NSS'15 (Network and System Security), Steering Committee Chair of IEEE BigData Security 2015, and Program Chair of IEEE SOSE/MobileCloud/BigData 2015. He won Navy Summer Faculty Award in 2012 and Air Force Summer Faculty Award in 2009. His research is supported by the US government, such as *National Science Foundation* (NSF), the Air Force, the Navy, and companies such as Nokia, TCL, and Cavium.

Min Chen (minchen2012@hust.edu.cn) [SM09] Min Chen is a professor in School of Computer Science and Technology at Huazhong University of Science and Technology (HUST). He is Chair of IEEE Computer Society (CS) Special Technical Communities (STC) on Big Data. He was an assistant professor in School of Computer Science and Engineering at Seoul National University (SNU) from September 2009 to February 2012. He worked as a Post-Doctoral Fellow in Department of Electrical and Computer Engineering at University of British Columbia (UBC) for three years. Before joining UBC, he was a Post-Doctoral Fellow at SNU for one and half years. He received Best Paper Award from IEEE ICC 2012, and Best Paper Runner-up Award from QShine 2008. He serves as editor or associate editor for Information Sciences, Wireless Communications and Mobile Computing, IET Communications, IET Networks, Wiley I. J. of Security and Communication Networks, Journal of Internet Technology, KSII Trans. Internet and Information Systems, International Journal of Sensor Networks. He is managing editor for IJAACS and IJART. He is a Guest Editor for IEEE Network, IEEE Wireless Communications Magazine, etc. He is Co-Chair of IEEE ICC 2012-Communications Theory Symposium, and Co-Chair of IEEE ICC 2013-Wireless Networks Symposium. He is General Co-Chair for the 12th IEEE International Conference on Computer and Information Technology (IEEE CIT-2012) and Mobimedia 2015. He is General Vice Chair for Tridentcom 2014. He is Keynote Speaker for CyberC 2012, Mobiquitous 2012 and Cloudcomp 2015. He has more than 260 paper publications, including 120+ SCI papers, 50+ IEEE Trans./Journal papers, 6 ISI highly cited papers and 1 hot paper. He has published two books: OPNET IoT Simulation (2015) and Big Data Inspiration (2015) with HUST Press, and a book on big data: Big Data Related Technologies (2014) with Springer Series in Computer Science. His Google Scholars Citations reached 6015+ with an h-index of 37. His top paper was cited 715+ times, while his top book was cited 420 times as of August 2015. He is an IEEE Senior Member since 2009. His research focuses on Cyber Physical Systems, IoT Sensing, 5G Networks, Mobile Cloud Computing, SDN, Healthcare Big Data, Medica Cloud Privacy and Security, Body Area Networks, Emotion Communications and Robotics, etc.

Keke Gai holds degrees from Nanjing University of Science and Technology (BEng), the University of British Columbia (MET) and Lawrence Technological University (MBA and MS). He is currently pursuing his PhD at Department of Computer Science at Pace University, New York, USA. Keke Gai has published more than 60 peer-reviewed journals or conference papers, 20+ journal papers (including ACM/IEEE Transactions), and 40+ conference papers. He has been granted three IEEE Best Paper Awards by IEEE conferences (IEEE SSC'16, IEEE CSCloud'15, IEEE BigDataSecurity'15) in recent years. His paper about cloud computing has been ranked as the "Most Downloaded Articles" of *Journal of Network and Computer Applications* (JNCA). He is involved in a number of professional/academic associations, including ACM and IEEE. Currently, he is serving as a Secretary/Treasurer of IEEE STC (Special Technical Community) in Smart Computing at IEEE Computer Society. He has worked for a few *Fortune 500* enterprises, including SINOPEC and GE Capital. His research interests include cloud computing, cyber security, combinatorial optimization, business process modeling, enterprise architecture, and Internet computing. He also served as a publicity chair/financial chair/web chair/TPC member in an amount of academic conferences, such as IEEE SmartCloud'16, SmartCom'16, IEEE CSCloud/SSC'15 '16, IEEE DataSec/HPSC/IDS'15 '16, IEEE HPCC/ICESS/CSS '15, and CONISAR'13.