Cost-Aware Multimedia Data Allocation for Heterogeneous Memory Using Genetic Algorithm in Cloud Computing

Keke Gai, *Student Member, IEEE,* Meikang Qiu, *Member, IEEE,* Hui Zhao *Student Member, IEEE*

Abstract—Recent expansions of *Internet-of-Things* (IoT) applying cloud computing have been growing at a phenomenal rate. As one of the developments, heterogeneous cloud computing has enabled a variety of cloud-based infrastructure solutions, such as multimedia big data. Numerous prior researches have explored the optimizations of on-premise heterogeneous memories. However, the heterogeneous cloud memories are facing constraints due to the performance limitations and cost concerns caused by the hardware distributions and manipulative mechanisms. Assigning data tasks to distributed memories with various capacities is a combinatorial NP-hard problem. This paper focuses on this issue and proposes a novel approach, *Cost-Aware Heterogeneous Cloud Memory Model* (CAHCM), aiming to provision a high performance cloud-based heterogeneous memory service offerings. The main algorithm supporting CAHCM is *Dynamic Data Allocation Advance* (2DA) Algorithm that uses genetic programming to determine the data allocations on the cloud-based memories. In our proposed approach, we consider a set of crucial factors impacting the performance of the cloud memories, such as communication costs, data move operating costs, energy performance, and time constraints. Finally, we implement experimental evaluations to examine our proposed model. The experimental results have shown that our approach is adoptable and feasible for being a cost-aware cloud-based solution.

Index Terms—Cloud computing, genetic algorithm, heterogeneous memory, data allocation, multimedia big data

1 Introduction

The advance of cloud computing has motivated a variety of explorations in information retrieval for big data informatics in recent years. Heterogeneous clouds are considered a fundamental solution for the performance optimizations within different operating environments when the data processing task becomes a challenge in multimedia big data [1], [2].The growing demands of multimedia big data have driven the increasing amount of cloud-based applications in *Internet-of-Things* (IoT). Combining heterogeneous embedded systems with cloud-oriented services can enable various advantages in multimedia big data. Currently, cloud-based memories are mostly deployed in a non-distributive manner on the cloud side [3]. This deployment causes a number of limitations, such as overloading energy, additional communications, and lower performance resource allocation mechanism, which restricts the implementations of the cloud-based heterogeneous memories [4]–[6]. This paper concentrates on this issue and proposes an innovative data allocation approach for minimizing total costs of the distributed heterogeneous memories in cloud systems.

- *K. Gai is with the Department of Computer Science, Pace University, New York, NY 10038, USA, kg71231w@pace.edu.*
- *M. Qiu (Corresponding author) is with the Department of Computer Science, Pace University, New York, NY 10038, USA, mqiu@pace.edu.*
- *H. Zhao is with Software School, Henan University, Kaifeng, Henan, 475000, China, zhh@henu.edu.cn*

- *This work is supported by NSF CNS-1457506 and NSF CNS-1359557.*

Manuscript received October XX, 2015

Contemporary cloud infrastructure deployments mainly mitigate data processing and storage to the clouds [7]. *Central Processing Units* (CPU) and memories offering processing services are hosted by individual cloud vendors. This type of deployment can meet the processing and analysis demands for those smaller sized data [8]–[10]. The incontinuous or periodically changeable implementations of the big data-oriented usage have caused bottlenecks for constructing firm performances [11]. For example, some data processing tasks are tightly associated with the industrial trends or operations, such as annual accounting and auditing [12], [13]. Therefore, a flexible approach meeting dynamic usage switches has become an urgent requirement in high performance multimedia big data.

Moreover, another challenge is that deploying distributed memories in clouds is still facing a few restrictions [14], [15]. Allocating data to multiple cloud-based memories results in obstacles due to the diverse impact factors and parameters [16]. The divergent configurations and varied capabilities can limit the entire system's performance since a naive task separation is inefficient to fully operate heterogeneous memories. It implies that minimizing the entire cost of using cloud memories is restricted by multiple dimensional constraint conditions.

To address the main challenge, we propose a novel approach entitled *Cost-Aware Heterogeneous Cloud Memory* (CAHCM) Model, which aims to achieve a reduced data processing time

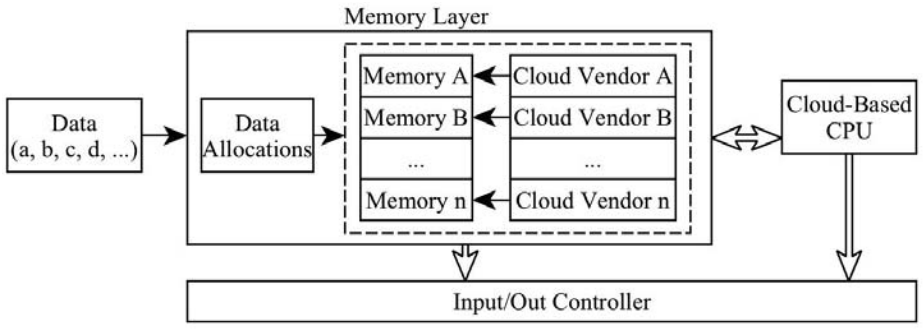


Fig. 1. The architecture of cloud-based heterogeneous memories using data allocation techniques in big data

through heterogeneous cloud memories for efficient *Memory-as-a-Service* (MaaS). The costs can be any expenditures during the operations, such as energy, time, and communication sources [17], [18]. Fig. 1 represents the architecture of cloud-based heterogeneous memory in which our proposed model will be applied. The operating principle of the proposed model is using genetic algorithm to minimize the processing costs via mapping data processing capabilities for different datasets inputs. The operating flow is based on the distributed parallel computations in heterogeneous memories located in various cloud vendors. The task assignments are completed by a data allocation operation in which tasks are assigned to cloud memories based on the mapping of the memory capabilities to the input data.

For supporting the proposed model, we propose an algorithm, *Dynamic Data Allocation Advance* (2DA) Algorithm, which is designed to obtain the minimum total costs using genetic algorithm. Implementing 2DA enables a dynamic data allocation to heterogeneous cloud memories with varied capacities. We consider a group of crucial factors that can impact on the costs when using cloud-based distributed heterogeneous memories, such as communication costs, data move operating costs, energy performance, and time constraints. The target of the proposed algorithm is to provide data allocation mechanism gaining the minimum computing costs.

The significance of this research is that the findings introduce a new approach for solving big data problems that request the scaled up capability memories deployed in clouds. Using our proposed scheme can facilitate a high performance of the efficiency in generating solutions to data allocations, which is a NP-hard problem. The main contributions of our research mainly include three aspects:

1) We venture to solve the cost optimization problem on heterogeneous memories in a polynomial time, which is a NP-hard problem. The outcomes are optimal solutions under certain constraints and suboptimal solutions with non-constraint.
2) The proposed model is an attempt in using heterogeneous cloud memories to solve multimedia big data problems by dynamically allocate data to various cloud resources.
3) We produce an approach that can be used to increase the entire usage rate of the cloud infrastructure along with the enhancement of the computation capability, which is also an approach for generating an optimized data processing mechanism in cloud-based multimedia data processing.

The remainder of the paper is organized by the follows. Section 2 reviews recent academic works in the relative fields. Next, we express a motivational example concerning the application of the proposed model in a simple implementation scenario in Section 3. Furthermore, main concepts and the proposed model statements are given in Section 4. Moreover, in Section 5, we provide detailed descriptions about the proposed algorithm. In addition, experimental evaluations are stated in Section 6, which configures the experimental settings and exhibits some experimental findings. Finally, we conclude the research in Section 7.

2 Related Work

This section has reviewed two crucial aspects related to our research, including multimedia big data and cloud resource management for multimedia in cloud computing. Investigations in these two hemispheres are theoretical supports for our research background.

2.1 Multimedia Big Data

Multimedia big data is a new technical term describing big data mechanisms applied in the multimedia field. We focus on the computation workload dimension even though a few other concentrations have been addressed by the prior researches.

First, the high performance-oriented data mining is one research direction in multimedia big data. One research was explored to prove that social images shared on cloud-based multimedia had a high level of similarities when users are socially connected [19], [20]. This research was an attempt of using multimedia to further expand the outcomes of big sized data mining. Next, data-oriented scheduling was also an alternative for enhancing the computation capability by applying resource scheduling algorithms [21], [22]. However, the computation workloads were not considered in this research, which was the main bottleneck for increasing the performance of the multimedia big data. Most data mining techniques can be directly used in multimedia big data field [23]. But the processing efficiency was a challenging issue due to the data size feature of the multimedia.

To address this challenge, a variety of preprocessing techniques have been examined by recent researches as well. One data preprocessing operation was configured in wireless multimedia sensor networks by deleting those data that have less impact on the results [24]. Computing the data having higher-level weights can obtain approximate solutions [25]. Some other data mining uses time series the determine the weights of the data by predictions [26]. However, this approach highly depends on the fault tolerance rate. The approach cannot ensure the accuracy if the rate is well configured even though the computation workload is reduced. Our work focuses on producing an innovative data allocation method for improving efficiency without reducing workloads.

Moreover, the feature selection was a research direction that could be used to eliminate those data with less data mining values. For example, an approach was proposed to use different constraints for re-ranking the image features [27]. This technique was also combined with Web services [23]. Despite many advantages of feature selections, this type of solutions is still attached to the software side. The computation speed was not increased, such that the balance of computation workload and outcome quality is still the critical part in this research direction. Our work targets at increasing the computation speed by optimizing the method of data allocations.

Therefore, most previous research work has addressed the software improvement for multimedia big data rather than the hardware side. Our research focuses on using cloud-based memories by deploying distributed heterogeneous computing resources.

2.2 Resource Management for Multimedia in Cloud Computing

Computing resources on cloud computing are deployed in a distributive manner. The approach of maximizing cloud resources has been explored by recent researches in different perspectives.

First, interconnecting various clouds has become a popular research topic in cloud computing. The relations between nodes in cloud computing are considered as one of the crucial aspects in increasing the entire system's performance [28], such as in *Internet of Things* (IoT). Prior researches have addressed a few dimensions increasing the system's performance. One of the crucial aspects in cost-aware approaches was saving energy by applying scheduling algorithms on *Virtual Machines* (VMs) [29], [30]. This type of optimizations mainly depends on the availability of the parameters for mapping cost requirements. Our proposed approach also requires a few parameters for the purpose of resource management, such as the amount of data reads and writes.

Moreover, from the perspective of security and forensics, resource management in cloud computing is considered an option of maximizing the security level by applying multiple constraints via VMs [31], [32]. Combining various inspection approaches on multiple VMs can increase the threat detection capability [33]. Correspondingly, the cost of the implementations will become greater while the amount of VMs increases [34]. The optimizations usually address the provisions of the heterogenous computing from distributed resources [35]. However, solving the resource management problem is still challenging when the number of the parameters or variables grows. It results in the difficulties of gain an adaptable solution that can be completed in a polynomial time. In the perspective, our research proposes an adaptive method that can ensure a high-efficiency of resource management method generations.

Furthermore, cloud task scheduling was also a research direction in resource management for big data. One recent optimization dimension was using iterative ordinal optimizations to adapt dynamic workloads in cloud computing [36]. This research aimed to gain sub-optimal scheduling solutions by optimizing each iteration. Meanwhile, some other scheduling optimizations focused on reducing the latency and constraints caused by bandwidths and data diversity [37]. A similar research focusing on reducing data transfer workloads used data partition techniques for optimizations. However, most prior researches did not consider implementing heterogeneous cloud computing such that the potential optimizations were ignored.

Therefore, in our research, we concentrated on a crucial research dimension that had rarely been addressed by the prior researches. Applying the heterogeneous memory in cloud computing is an approach to increase cloud service efficiency and its challenge is data allocations to diverse memories. The target addressed by our research is producing a method that can efficiently produce data allocation plans with minimum costs.

3 Motivational Example

We give a motivational example for clarifying the operational processes of CAHCM in this section. Assume that there are four cloud vendors offering MaaS with different performances, namely M_1, M_2, M_3, and M_4. Table 1 displays the costs for different cloud memory operations.

As shown in Table 1, we consider four main costs that include *Read* (R), *Write* (W), *Communications* (C), and *Move* (MV). R and W refer to the operation costs of reading and writing data. C means the costs happened to communication processes through the Internet. MV represents the costs taken place at the occasions when switching from one memory vendor to another set. β is a criterion for memory limits. There are two working modes based on βs, including a normal working status and an over limit status. Table 2 represents different performance critical points for different cloud memory limits.

According to Table 2, cloud memories have different memory service offerings. For example, M_1 has a critical capability

TABLE 1
The costs for different cloud memory operations. Four types of memories are M_1, M_2, M_3, and M_4; parameter (PAR) β represents the performance critical points; main costs derive from four aspects, including Read, Write, Communications (Com.), and Move.

Operations	PAR	M_1	M_2	M_3	M_4
Read (R)	β_1	4	2	1	50
	β_2	500	500	500	500
Write (W)	β_1	6	4	2	50
	β_2	500	500	500	500
Comm. (C)	-	10	10	10	10
Move (MV)	M_1	0	4	6	40
	M_2	3	0	5	40
	M_3	2	3	0	40
	M_4	40	40	40	0

TABLE 2
Performance critical points for different cloud memory limits and the number of memories.

	M_1	M_2	M_3	M_4
β_1 (GB)	<4	<3.2	<3.5	<4
β_2 (GB)	⩾4	⩾3.2	⩾3.5	⩾4
Number of Memories	2	2	3	3

point at 4 GB, which means the performance will be dramatically diminish when the input dataset is larger than 4 GB. The example configuration is that there are 2 M_1, 2 M_2, 3 M_3, and 3 M_4. Moreover, Table 3 represents the number of memory accesses and data sizes. There are 6 input data, including *A*, *B*, *C*, *D*, *E*, *F*, and *G* as well as their corresponded sizes. For instance, Data *A* are required to read 7 times and write 6 times and the data size is 2.35 GB.

TABLE 3
The number of memory accesses and data sizes

Data	Reads	Writes	Sizes (GB)
A	7	6	2.35
B	6	3	2.70
C	8	6	3.30
D	1	1	3.63
E	3	1	2.76
F	6	6	2.06
G	2	3	3.36

Assume that the initialized data allocations are $A \to M_2$, $B \to M_2$, $C \to M_4$, $D \to M_2$, $E \to M_3$, $F \to M_4$, and $G \to M_3$. The costs of allocating each data to memory units are given in Table 4. This table is called *B Table*, which is used to mapping all the costs for all potential activities of the data.

The mapped the costs and the number of accesses match the conditions of our proposed model. Thus, we remap the data that are shown in Table 5. The remapping table is named as *D Table*, which is used to generate optimized memory selections based on the data from *B Table*. There are two steps in the remapping process. First, we sort the data in an ascending order by summing up the number of *Read* and *Write*. For example, *A* is 13 deriving from (7+6), which is shown in the table. The next step is sorting the cost by the memory index number for each data. We assign an index number to each memory by $1 \to M_1$, $2 \to M_2$, $3 \to M_3$, and $4 \to M_4$. We sort the index for each data according to the costs of the data allocations given in Table 4. According to the memory availability, we always select the lowest costs first, from the left side to the right side in the table. For example, in Table 4, the data allocation costs for data *A* are respectively 77 (M_1), 48 (M_2), 34 (M_3), and 700 (M_4). The sorted order for *A* is M_3, M_2, M_1, and M_4 in Table 5.

TABLE 4
B Table: The costs of allocating each data to different memory units considering β.

Data	M_1	M_2	M_3	M_4
A	77	48	34	700
B	55	34	27	500
C	118	7050	70	710
D	23	1010	1015	150
E	30	23	15	250
F	110	86	68	610
G	38	2513	18	300

TABLE 5
D Table: Sorted memory indices deriving from Table 4

C	M_3 (70)	M_1 (118)	M_4 (710)	M_2 (7050)
A	M_3 (34)	M_2 (48)	M_1 (77)	M_4 (700)
F	M_3 (68)	M_2 (86)	M_1 (110)	M_4 (610)
B	M_3 (27)	M_2 (34)	M_1 (55)	M_4 (500)
G	M_3 (18)	M_1 (38)	M_4 (300)	M_2 (2513)
E	M_3 (15)	M_2 (23)	M_1 (30)	M_4 (250)
D	M_1 (23)	M_4 (150)	M_2 (1010)	M_3 (1015)

Based on the remapping, we select memories for each data, which starts with the top of the table to the bottom. According to the memory availability, we always select the lowest cost from the available memories, from the left to right shown in Table 5. Furthermore, an adjustment will be made after a calculation of the proposed genetic algorithm. In this motivational example, using our proposed approach can generate an optimal solution as $A \to M_3$, $B \to M_2$, $C \to M_3$, $D \to M_1$, $E \to M_2$, $F \to M_3$ and $G \to M_1$ with the total cost 290. Compare with *First-In-First-Out* (FIFO) algorithm, our approach reduces 96.5% cost in this case. Compare with Greedy algorithm, our approach can reduce 3.7% cost in this example.

4 Concepts and the Proposed Model

We demonstrate the proposed model and define the main concepts used in the model in this section. The operating principle of the model is given along with the proof in this section.

4.1 Execution Model and Definition

The execution model is based on the architecture represented in Fig. 1. Before the data are assigned to cloud memories, the entire input data need to be partitioned, which have been addressed by the prior researches [38]–[40]. In our proposed model, we assume that the data partitions are processed, after which the information of data operations in each cloud memory is gained. The task addressed by the proposed CAHCM is to execute data allocations for minimizing the total costs.

Definition 1. Cost Optimization Problem on Heterogeneous Memories (COPHM): *Given the initial status of input data and the cloud-based heterogeneous memories' capacities, including the number of data, the number of* Read *and* Write *accesses, the number of memories and availabilities, the costs of* Read *and* Write *for each memory, the costs of the* Move *between memories, critical points of the memories. The aimed problem is to find out the data allocation solution optimally minimizing the total cost.*

The inputs are the initial data allocations as well as memories' availabilities, capacities, and costs. Align with the problem definition, main inputs include the number of data, the required *Read* and *Write* access amounts, available memories' number as well as the costs for *Read*, *Write*, and the *Move* between memories, the initial locations of the data, data sizes, and critical points for each memory. The output is a data allocation scheme based on the available cloud memories provisioning the optimal solution for minimizing total costs.

Moreover, the crucial component of CAHCM is that the task assignments to heterogeneous memories offered by distinct cloud vendors based on the memories' availabilities, costs, and capabilities. Compare with the traditional on-premises memories, the cloud memories system provides a manageable data allocation platform, which allows tasks to be assigned to a distributed memory system. The capacities of the heterogeneous memories in cloud computing can be varying, which depends on the offerings of cloud providers. In our model, the task assignment process mainly includes two steps, which are mapping costs and task assignments. Fig. 2 represents an operation flow model of CAHCM model.

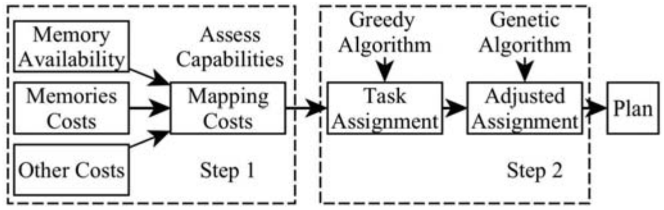


Fig. 2. Operation flow model of Cost-Aware heterogeneous cloud memory model

According to Fig. 2, at the first step, the costs are mapped mainly deriving from three categories of information. The *Memory Availability* refers to the availabilities of the memories, which may include amount, time, connection status, and locations, The *Memories Costs* is the consumption information corresponding to each available memory and the corresponding table in Section 3 is Table 1. *Other Costs* means the costs caused from other computing resources except the costs deriving from memories or operations between memories, such as transmission losses. For example, a *Move* cost is one of the costs that need to be considered. In addition, the data allocation plan is generated at the second step, which is mainly supported by our proposed algorithm, 2DA algorithm. Two subtasks form this step. First, an initial plan is created by using Greedy algorithm. Second, an improved data allocation plan is created by implementing the genetic algorithm, such that the task assignment may be adjusted. The following section provides the problem statement.

4.2 Problem Statement

The calculations of the total cost need to consider the cloud memories' capabilities [5]. The limits are determined by β that maps the critical capability points for different memories, which is aligned with COPHM problem defined by Definition 1 in Section 4.1. To address the cost minimizations, we formulate the cost calculation method shown in Eq. 2:

$$\begin{aligned} C_{Total} &= \sum_{i=1}^{n} C(i) \\ &= \sum_{i=1}^{n} (C_i^R \times N_i^R + C_i^W \times N_i^W + C_i^{MV} + C_i^{Other}) \end{aligned} \tag{1}$$

The total cost is a sum of all memories' costs with four types of costs. As shown in Eq. 2, C_{Total} refers to the total cost. The sum of all memories' costs is $\sum_{i=1}^{n} C(i)$ showing there are i memories operated in the data processing. C_i^R refers to the cost of *Read* and N_i^R refers to the number of the *Read*. Correspondingly, C_i^W refers to the cost of *Write* and N_i^W refers to the number of the *Write*. C_i^{MV} refers to the cost of data moves and C_i^{Other} refers to other costs, such as networking communications. The target of our proposed approach is minimizing the value of C_{Total}.

Moreover, for comparing the costs, normally a traversing search is required. Assume that there are m memories in total and d data. $P(m, d) = \frac{m!}{(m-d)!}$ where $d \leqslant m$. The minimum cost can be found from a permutation of *P(m, d)*. Using this information, the heterogeneous memories can construct a few bins such that each bin can refer to one memory. The volume of bins are memories' capacities, the items are input data, and the values are costs of data operations. In addition, we consider minimizing the total costs such that COPHM problem can be reducible to *Bin Packing Problem* (BPP) that is a combinatorial NP-hard problem [41], [42]. Therefore, COPHM problem is also a NP-hard problem, BPP$\leqslant_p$COPHM.

Our proposed approach can gain optimal solutions within the constraint conditions and suboptimal solutions under the non-constraint environment.

4.3 Optimal Constraints for Genetic Algorithm

Our work has found that using Greedy algorithm could provide optimal solutions under certain constraints and suboptimal solutions in other situations. Being aware of the optimal solution constraint is the critical component for achieving optimal solutions. It means that an adjustment needs to be done in order to improve the sub-optimal solution to being the optimal solution. We present an approach of overcoming the optimal solution constraint by applying genetic algorithm; thus, our proposed CAHCM is designed to achieve a high rate of producing optimal solutions. The proposed genetic algorithm is designed by considering the optimal solution constraints, which is represented in this section. The representation of the optimal solution constraints are illustrated as follows.

$\exists$ two memories M_1 and M_2, represented as $M_1 = \langle R_1, W_1\rangle$ and $M_2 = \langle R_2, W_2\rangle$. R_i refers to the *Read* cost and W_i refers to the *Write* cost. Assume that there are two data, A and B, need to be assigned, represented as $A = \langle a_1, b_1\rangle$ and $B = \langle a_2, b_2\rangle$. a_i is the access number of the *Read* and b_i is the number of the *Write*. We set $(a_1 + b_1) > (a_2 + b_2)$, $R_1 > R_2$, and $W_1 > W_2$. Therefore, data *A* have the selection priority according to our approach. We consider data *B* the data having lower selection priority than *A*, which locate at the right side of *A* in *D Table*. The table mapping is represented in Table 6.

TABLE 6
Table mapping for the cost comparisons between data allocations

B	C_2^b	C_1^b
A	C_2^a	C_1^a

First, our approach uses Greedy algorithm to determine the path $C_2^a \to C_1^b$. Since there are only two memories, the other path is $C_2^b + C_1^a$. Therefore, the crucial deterministic comparison for the optimal solution requirement is shown in Eq. 2, which shows a fundamental relation of obtaining an optimal solution.

$$C_2^b + C_1^a \geqslant C_1^b + C_2^a \Rightarrow C_1^a - C_2^a \geqslant C_1^b - C_2^b \tag{2}$$

In Eq. 2, C_i^j means the cost of using the i memory for data j. We use MC_i to present the sum of *Move* and *Communication* costs. The mathematical expressions of C_i^j are given as follows:

$$\begin{cases} C_1^a = R_1 \times a_1 + W_1 \times b_1 + MC_1 \\ C_2^a = R_2 \times a_1 + W_2 \times b_1 + MC_2 \\ C_1^b = R_1 \times a_2 + W_1 \times b_2 + MC_3 \\ C_2^b = R_2 \times a_2 + W_2 \times b_2 + MC_4 \end{cases}$$

Align with Eq. 2, we gain the following Eq. 3, which represents the constraints for obtaining optimal solutions.

$$\begin{cases} (R_1 - R_2)(a_1 - a_2) + (w_1 - w_2)(b_1 - b_2) \geqslant MC_{total} \\ MC_{total} = MC_2 + MC_3 - MC_1 - MC_4 \end{cases} \tag{3}$$

For satisfying the optimal solution requirements, we execute Eq. 3 and consider three situations based on the comparisons between variables a_1 and a_2 as well as variables b_1 and b_2. The results are given in Eq. 4.

$$\triangle R_{12} \triangle a_{12} + \triangle W_{12} \triangle b_{12} \geqslant MC_{total} \tag{4}$$

Consider the practical scenario, we define the process of data allocation as a P, in which there is a sorted data set. For any adjacent data A and B, the path is represented as $\overrightarrow{BA}$ meaning the data allocation order from data B to A. In any cases, $\forall$ four adjacent data allocations C_i^a, C_k^a, C_f^a, and C_j^a in *D Table*, $\exists$ $C_i^a < C_k^a$, $C_f^b < C_j^a$, and a path $a \to b$. For firming an optimal solution, it needs to ensure that $C_i^a + C_j^b - C_f^b - C_k^a \leqslant 0$. The values of the data allocations are:

$$\begin{cases} C_i^a = R_i \times a^R + W_i \times a^W + MC_i \\ C_j^b = R_j \times b^R + W_j \times b^W + MC_j \\ C_f^b = R_f \times b^R + W_f \times b^W + MC_f \\ C_k^a = R_k \times a^R + W_k \times a^W + MC_k \end{cases}$$

In the expression, R_i refers to the *Read* cost for memory i, a^R refers to the *Read* access number for data *A*, W_i refers to the *Write* cost for memory i, and a^W refers to the *Write* access number for data *A*. The path from data *A* to *B* selecting $C_i^a \to C_j^b$ means data allocations are $A \to$ *Memory i* and $B \to$ *Memory j*. We set the $\triangle R_{fj} = R_f - R_j$, $\triangle W_{fj} = W_f - W_j$, $\triangle R_{ki} = R_k - R_i$, $\triangle W_{ki} = W_k - W_i$, and $MC_{total} = MC_i + MC_j - MC_k - MC_f$ Therefore, the optimal solutions constraints are shown in Eq. 5:

$$\begin{cases} \triangle R_{fj} b^R + \triangle W_{fj} b^W + \triangle R_{ki} a^R + \triangle W_{ki} a^W \geqslant MC_{total} \\ MC_{total} = MC_i + MC_j - MC_k - MC_f \end{cases} \tag{5}$$

Applying Greedy algorithm can produce optimal solutions when the condition meets the requirements given in Eq. 5. Otherwise, using Greedy algorithm can generate suboptimal solutions that need adjustments. The accomplishment of the adjustment is processed by our proposed genetic algorithm. The theorem stating the optimal solutions restraints is given in Theorem 4.1, which focuses on the adjacent path comparisons and decisions in *D Table*.

Theorem 4.1. *$\exists$ a set memories $\mathbb{M}$, $\{M_i, M_j, M_f, M_k\} \subseteq \mathbb{M}$, data A=$\langle a^R, a^W\rangle$, B=$\langle b^R, b^W\rangle$, $C_i^a < C_k^a$ and $C_f^b < C_j^b$. The costs of the memories consist of R_x and W_y, $x = \{i, j, f, k\}$, $y = \{i, j, f, k\}$. Configure $\triangle R_{fj} = R_f - R_j$, $\triangle W_{fj} = W_f - W_j$, $\triangle R_{ki} = R_k - R_i$, $\triangle W_{ki} = W_k - W_i$, and $MC_{total} = MC_i + MC_j - MC_k - MC_f$. If satisfying $\triangle R_{fj} b^R + \triangle W_{fj} b^W + \triangle R_{ki} a^R + \triangle W_{ki} a^W \geqslant MC_{total}$ and priority selection principle, then $C_i^a + C_j^b \leqslant C_f^b + C_k^a$ and the data allocation $\langle A \to M_i, B \to M_j\rangle$ is an optimal solution for minimizing total costs under the four-memory configuration.*

We use contradiction proof to prove the correctness of Theorem 4.1. We use $\mathbb{P}$ to represent the satisfaction of Eq. 5, use $\mathbb{G}$ to denote the optimal solutions. The following proof proves that $\mathbb{G}$ is true when $\mathbb{P}$ is true, $\mathbb{P} \Rightarrow \mathbb{G}$.

Proof. Assume that $\mathbb{P}$ is true and $\mathbb{G}$ is false, when there exists memories M_i, M_j, M_f, M_k, data A=$\langle a^R, a^W\rangle$ and B=$\langle b^R, b^W\rangle$. Since $\mathbb{P}$ is true, $\mathbb{P} \Rightarrow C_i^a + C_j^b < C_f^b + C_k^a$. $\mathbb{G}$ is false $\Rightarrow$ there exists a solution $(C_x^a + C_y^b)$ offering a lower cost than $(C_i^a + C_j^b)$. $\neg\mathbb{G} \Rightarrow (C_i^a + C_j^b) > (C_x^a + C_y^b)$, where the adjacent paths formed by $C_i^a < C_x^a < C_k^a$, $C_f^b < C_j^b < C_y^b$, following the availability priority principle in *D Table*. Define the formulation calculating the total costs is $\mathbf{C}(m,n)$, as shown in Eq. 2. In the formulation $\mathbf{C}(m,n)$, *m* means the memory selection and *n* means the data.

Therefore, the following deduction is gained:

$$\begin{pmatrix} \mathbf{C}(f,b), \mathbf{C}(j,b), \mathbf{C}(y,b) \\ \mathbf{C}(i,a), \mathbf{C}(x,a), \mathbf{C}(k,a) \end{pmatrix} \text{ with } \begin{cases} C_x^a \in [C_i^a, C_k^a] \\ C_y^b \in [C_f^b, +\infty) \end{cases} \Rightarrow$$

$$\text{when } \neg\mathbb{G} \Rightarrow \exists \mathbf{C}(x,a) + \mathbf{C}(y,b) < \mathbf{C}(i,a) + \mathbf{C}(j,b)$$
$$< \mathbf{C}(f,b) + \mathbf{C}(k,a) \Rightarrow C_x^a - C_i^a < C_i^b - C_y^b$$
$$\text{Followed by } \mathbb{P}, \exists C_x^a - C_i^a > C_i^b - C_f^b$$
$$\Rightarrow C_i^b - C_y^b > C_x^a - C_i^a > C_i^b - C_f^b \Rightarrow C_y^b < C_f^b$$

which conflicts with $C_f^b < C_y^b$. Therefore, $\mathbb{P} \Rightarrow \mathbb{G}$ is true. Proved. □

Therefore, our genetic algorithm emphasizes the constraints and configure the evolutive principle according to the findings above. The initial plan created by greedy algorithm will be assessed by the optimal solution constraints and determine whether the plan needs to be adjusted. The next section provides the detailed description about our crucial algorithms.

5 Algorithms

In this section, we provide descriptions of two crucial algorithms of CAHCM. Section 5.1 illustrates 2DA algorithm and Section 5.2 exhibits the *B Table Generation Algorithm*.

5.1 Dynamic Data Allocation Advance (2DA) Algorithm

Aim to generate an adaptable data allocation solution, we propose *Dynamic Data Allocation Advance (2DA)* algorithm. This algorithm is designed to solve COPHM problem in a polynomial time. The outputs of implementing 2DA algorithm is producing optimal solutions under constraints or suboptimal solutions within non-constraint conditions. Pseudo codes of 2DA algorithm are given in Algorithm 5.1.

As mentioned in Section 4.1, inputs mainly include information about the data and memories. The notations used in algorithms' pseudo codes are given as follows. We use D_i denote the input data and i refers to the data index. The number of the input data is N_d. The data size for each data is $S_{d(i)}$. The required number of the *Read* and *Write* accesses for each input data is $R_{d(i)}$ and $W_{d(i)}$, which is aligned with D_i. The critical points for each memory is β. The cost of *Read* access is $C_r^{\beta} d(i)$ when $S_{d(i)} < \beta$. Otherwise, use $C_r^{!\beta} d(i)$ when $S_{d(i)} \geqslant \beta$. Correspondingly, The cost of each *Write* is $C_w^{\beta} d(i)$ when $S_{d(i)} < \beta$ and the use $C_w^{!\beta} d(i)$ when $S_{d(i)} \geqslant \beta$. The *Move* costs between memories are $M_{\overrightarrow{ab}}$, which means the data move from Memory a to b. The number of cloud memory type offered by one cloud vendor is N_{mt}. Each type has k available memories and we use $M_{mt}(g)$ to denote each memory.

In addition, the output is *Data Allocation Table* (DataAllocation). Two crucial procedures are involved in this algorithm.

Algorithm 5.1 Dynamic Data Allocation Advance (2DA) Algorithm

```
Require: B Table, $N_{mt}$, $k$, $\{M_{mt}(g)\}$
Ensure: DataAllocation
 1: Input B Table
 2: Initialize all availabilities of the cloud memories, $N_{mt}$, $k$, $\{M_{mt}(g)\}$
 3: $\forall$ data, sort data by the values of ($R_{d(i)}$ + $W_{d(i)}$) in a descending order
 4: for $\forall$ input data do
 5:     Sort memory allocation costs in an ascending order
 6:     /*According to B Table*/
 7:     for $\forall$ the cost for each data allocation manner do
 8:         if The memory ($M_{mt}(g)$) is available then
 9:             Allocate data to this memory
10:             /*$D_i \rightarrow M_{mt}(g)$*/
11:             The number of this type of memory -1
12:         end if
13:     end for
14: end for
15: for i=1 to Constant do
16:     for j=1 to data.size do
17:         if ($Cost_j(Plan_j)$+$Cost_{j-1}(Plan_{j-1})$)> ($Cost_{j-1}(Plan_j)$+$Cost_j(Plan_{j-1})$) then
18:             $Plan_j \leftrightarrow Plan_{j-1}$
19:             /*Switch the memory selections*/
20:         end if
21:     end for
22: end for
23: RETURN DataAllocation
```

First, we use Greedy algorithm to create a solution that can be either an optimal solution or a sub-optimal solution. Next, we use genetic algorithm to improve the solution, which applies the operating principle shown in Eq. (5) to evaluate the obtained results gained from the Greedy. The data allocation plan will be updated if lower costs are gained within the configured generation rounds. The main phases of Algorithm 5.1 are:

1) Input *B Table* generated by Algorithm 5.2. Initialize all availability information about the heterogeneous cloud memories. The example of *B Table* is Table 4.
2) Summing up $R_{d(i)} + W_{d(i)}$ for each memory and sort the values in a descending order.
3) Sort the memory options in an ascending order for each data according to the required costs, which are based on mappings of *B Table*. The sorting results example is illustrated by Table 5 in Section 3.
4) Allocate the data to the available memories that have the highest priority till the last data by which the initial *DataAllocation* strategy is made.
5) An adjustment is made by accomplishing a genetic algorithm, in which a series of comparisons are processed. The comparison is based on the optimal solution constraints defined by Eq. (5).

6) Update the plan if the better solution is found until the configured rounds end. This is the main procedure of applying our proposed genetic algorithm.
7) Return the completed *DataAllocation* strategy for data allocations.

In summary, 2DA algorithm is an approach using both Greedy and genetic algorithms, which can produce an efficient solution within a short period time. The next section describes the method of generating B Table generation, which is one of the inputs of 2DA algorithm.

5.2 B Table Generation Algorithm

As mentioned in Section 5.1, we need to map the costs for each data and the corresponding memories in a *B Table*, which is one of the main inputs of Algorithm 5.1. This algorithm considers the memory critical points, β, as well as the corresponding memory performances under different βs. The purpose of this algorithm is mapping the costs with the corresponding data allocations in a computing friendly manner. Algorithm 5.2 represents the *B Table Generation Algorithm*.

Algorithm 5.2 B Table Generation Algorithm

```
Require: $D_i$, $N_d$, $R_{d(i)}$, $W_{d(i)}$, $M_{\overrightarrow{ab}}$, $N_{mt}$, $M_{mt}(g)$, $k$, $M_j$, $S_{d(i)}$, $C_r^{\beta} d(i)$, $C_r^{!\beta} d(i)$, $C_w^{\beta} d(i)$, $C_w^{!\beta} d(i)$, $\beta$
Ensure: B Table
 1: Initialize all input data, $D_i$, $N_d$, $R_{d(i)}$, $W_{d(i)}$, $M_{\overrightarrow{ab}}$, $N_{mt}$, $M_{mt}(g)$, $k$, $M_j$, $S_{d(i)}$, $C_r^{\beta} d(i)$, $C_r^{!\beta} d(i)$, $C_w^{\beta} d(i)$, $C_w^{!\beta} d(i)$, $\beta$
 2: for $\forall$ data do
 3:     for $\forall$ memories $\{M_{mt}(g)\}$ do
 4:         Initialize $Cost_{temp} \leftarrow 0$
 5:         if $S_{d(i)} < \beta$ then
 6:             $Cost_{temp} \leftarrow Cost_{temp} + R_{d(i)} \times C_r^{\beta} d(i)$
 7:             $Cost_{temp} \leftarrow Cost_{temp} + W_{d(i)} \times C_w^{\beta} d(i)$
 8:         else
 9:             $Cost_{temp} \leftarrow Cost_{temp} + R_{d(i)} \times C_r^{!\beta} d(i)$
10:             $Cost_{temp} \leftarrow Cost_{temp} + W_{d(i)} \times C_w^{!\beta} d(i)$
11:         end if
12:         $Cost_{temp} \leftarrow Cost_{temp} + M_{ab}$
13:         $Cost_{temp} \leftarrow Cost_{temp}$ + communication costs
14:         Data allocation cost to $M_{mt}(g) \leftarrow Cost_{temp}$
15:     end for
16: end for
17: RETURN B Table
```

The main phases of Algorithm 5.2 include:

1) Input all required data, including D_i, N_d, $R_{d(i)}$, $W_{d(i)}$, $M_{\overrightarrow{ab}}$, N_{mt}, $M_{mt}(g)$, k, M_j, $S_{d(i)}$, $C_r^{\beta} d(i)$, $C_r^{!\beta} d(i)$, $C_w^{\beta} d(i)$, $C_w^{!\beta} d(i)$, β. Initialize a temporary variable $Cost_{temp}$ and assign an empty value.
2) Use Eq. (2) to calculate the value of total cost for each possible data allocations considering the memory capacities. Assign the values of the calculation outcomes to the temporary variable $Cost_{temp}$.
3) Generate the Table *B* by assigning the values of the $Cost_{temp}$ to the corresponding spots in $M_{mt}(g)$.
4) Return B Table used as the inputs of Algorithm 5.1.

In summary, two algorithms illustrated in this section support two main procedures of the model. The following section demonstrates our experimental evaluations.

6 Experiment and the Results

We examined our proposed approach through an experimental evaluation. Section 6.1 showed our experimental configurations, and Section 6.2 displayed partial our experimental results.

6.1 Experimental Configurations

We used experimental evaluations to assess the performance of the proposed scheme. For the purpose of simulating cloud systems, we used two servers to imitate the operations of distributed cloud servers. Two servers acted a remote server and an on-premise client end, respectively. The operating system configurations were Ubuntu 15.04 on bother servers. The input data were generated in a random setting in order to simulate the distributed data offload in practical scenarios. Two crucial aspects were examined in our evaluations, including efficiency and cost-saving performances.

First, we estimated the efficiency by evaluating the execution time. Two main comparisons were done in this evaluation. One comparison was investigating the execution time differences between the proposed scheme and the brute force method when various amounts of the cloud memories are executed. The other comparison assessed the execution time trend of the proposed model in various memory deployments.

In addition, our experiment also tested the performance of saving costs by evaluating the optimal and suboptimal solutions proportions out of a great number of experiments. We configured the difference ($\triangle Cost$) between the optimal solution costs ($Cost_{op}$) and the costs ($Cost_{CAHCM}$) required by our proposed approach, which is represented as $\triangle Cost$= $Cost_{CAHCM} - Cost_{op}$. We defined the *Accuracy Ratio* (AR) as the ratio between the optimal solution cost and $\triangle Cost$, which used the following formula: $AR = \triangle Cost / Cost_{op}$.

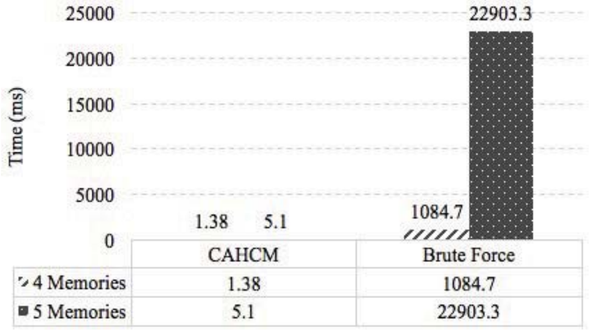
Moreover, we had two main experimental settings that were designed to assess the proposed approach's performance in different perspectives. The statement of two settings:

- Setting 1: We used a group of settings to evaluate the performance of CAHCM with different input data sizes, which included: Setting 1-1, 20 KB; Setting 1-2, 10 MB; Setting 1-3, 250 MB; Setting 1-4, 500 MB.
- Setting 2: We simulated multiple memory deployments, from 5 to 20, to assess the performance of data allocations in heterogeneous cloud memories.

These two settings were designed to examine the performance in adaptability and execution time while the input data sizes were varied.

6.2 Experimental Results

This section represented a few experimental results based on our experimental configurations. The performance comparisons were given between the proposed CAHCM and other available algorithms. The execution time examinations mainly focused on various input data sizes while deploying different amounts of the cloud memories.



	CAHCM	Brute Force
▨ 4 Memories	1.38	1084.7
■ 5 Memories	5.1	22903.3

Fig. 3. Time consumptions between CAHCM and Brute Force using 4and 5 memories under setting 1-1.

Fig. 3 represents the time consumptions between our proposed scheme and brute force method using 4 and 5 cloud memories under setting 1-1. The data allocation tasks using 6 memories cannot be accomplished in an acceptable polynomial time. As depicted in Fig. 3, CAHCM could complete the task in a shorter period. For example, using CAHCM could allocate data within 5.1 ms using 5 cloud memories, which was much shorter than brute force that required 22903.3 ms. The execution time differences between the proposed approach and normal method were great.

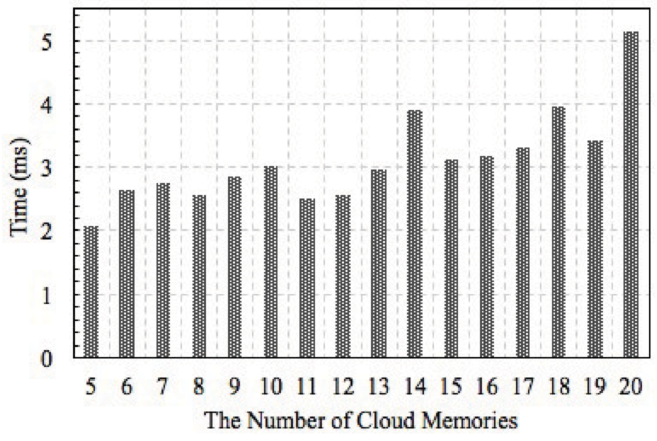


Fig. 4. Execution efficiency performance comparisons using CAHCM using different amounts of cloud memories under setting 1-1.

Moreover, we further evaluated the execution time by comparing various number of cloud memories, from 5 to 20, under settings 1-1, 1-2, 1-3, and 1-4. Fig. 4 exhibits the execution efficiency performance comparisons using CAHCM under different amounts of the cloud memories. As shown in the figure, the execution time of applying CAHCM only had a slight growth from 5 to 20 cloud memories. It implies that our proposed scheme could solve N-memory deployments in a dramatical short period, which met the heterogeneous cloud memories' needs. The average increase rate of the execution time was 7.8% when one cloud memory was added.

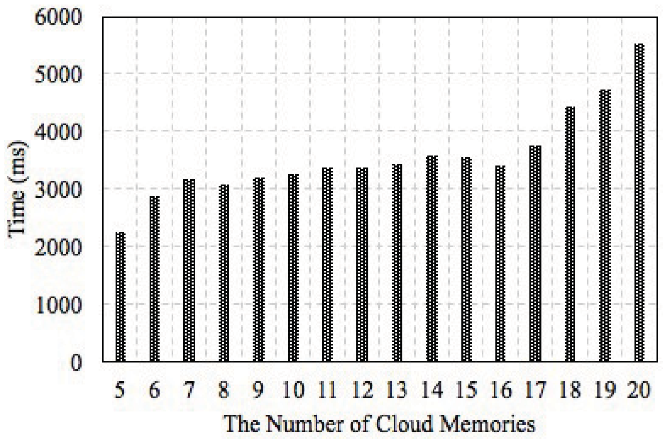


Fig. 5. Execution efficiency performance comparisons using CAHCM using different amounts of cloud memories under setting 1-2.

Fig. 5 represents the execution time evaluations under setting 1-2. The execution time had a positive relationship with the number of the memories. The average increase time was 218 ms. The average increase rate of the execution time was 6.5% along with the number of the cloud memories grew.

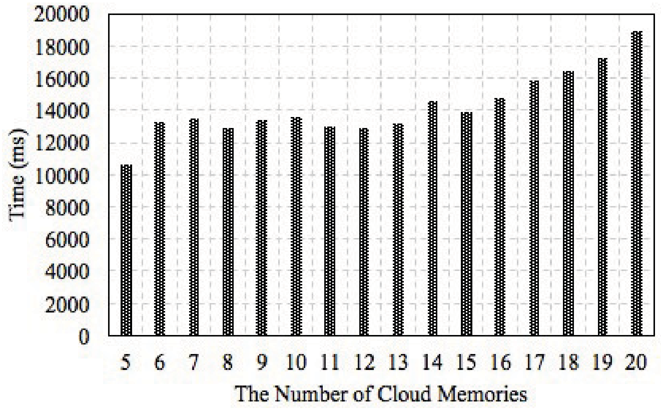


Fig. 6. Execution efficiency performance comparisons using CAHCM using different amounts of cloud memories under setting 1-3.

Fig. 6 shows the execution efficiency performance evaluations under setting 1-3. The movement had a similar trend line to Fig. 5. In general, the average increased execution time was 550 ms. The average execution time increase rate of each cloud memory increment was 4.1%.

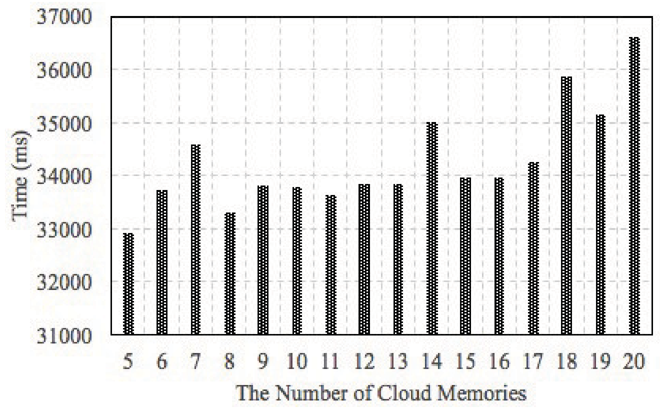


Fig. 7. Execution efficiency performance comparisons using CAHCM using different amounts of cloud memories under setting 1-4.

Furthermore, Fig. 7 displayed some experimental results for evaluating execution efficiency under setting 1-4. The average increase time was 246 ms. And the average increase rate was 0.7%. On the whole, the execution time was associated with the number of the cloud memories in our experimental evaluations. The input data sizes also had a positive relationship with the execution time. The average of the increase rate for inserting each cloud memory was 4.8%.

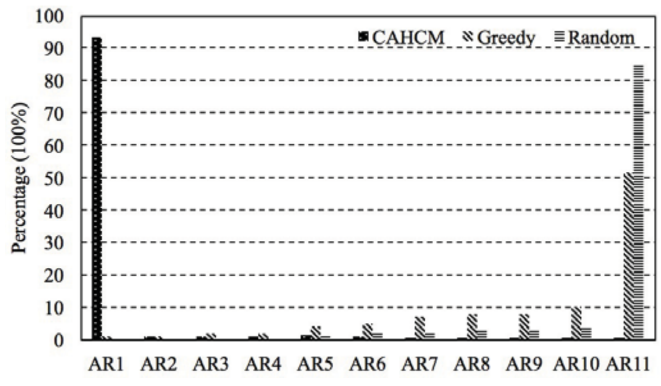


Fig. 8. Comparisons of the ratios of the optimal and suboptimal solutions examined by ratios for solving four dimensional data allocations for CAHCM, Greedy, and Random algorithms. AR1=0%, AR2=(0, 1%], AR3=(1%, 2%], AR4=(2%, 3%], AR5=(3%, 4%], AR6=(4%, 5%], AR7=(5%, 10%], AR8=(10%, 20%], AR9=(20%, 50%], AR10=(50%, 100%], AR11=(100%, +∞]

Next, Fig. 8 represents the comparisons of the optimal solutions proportions among CAHCM, Greedy, and Random algorithms. As defined in Section 6.1, we used AR to represent the portions of the optimal and suboptimal solutions. Our experimental results showed that the optimal solutions took move than 90% out of all the examinations, which was 93.61%. *AR1* depicts the percentage of the optimal solutions. From *AR2* to *AR11* illustrated the different portions under various settings, which were given in the figure's caption. The percentage of the acceptable solutions was 95.8%, if the acceptable scope of AR was [0, 2%] that could be considered the set of optimal and suboptimal solution. As the same configuration, Greedy algorithm could achieve 6.06% chances to reach the acceptable solutions at [0, 10%]. Random algorithm could achieve 5.05% at the scope [0, 10%]. The results depicted that our approach could obtain a dramatically high rate of obtaining optimal solutions.

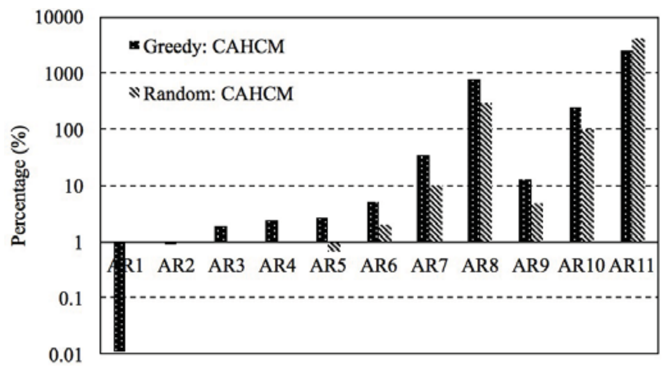


Fig. 9. Ratios of the optimal and suboptimal solutions examined by ratios for solving four dimensional data allocations. AR1=0%, AR2=(0, 1%], AR3=(1%, 2%], AR4=(2%, 3%], AR5=(3%, 4%], AR6=(4%, 5%], AR7=(5%, 10%], AR8=(10%, 20%], AR9=(20%, 50%], AR10=(50%, 100%], AR11=(100%, +∞].

In addition, Fig. 9 represents a comparison rate between CAHCM and other two algorithms. We defined *Comparison Rate = Algorithm/CAHCM*. Therefore, a higher value of the rate meant a greater difference gap when the rate was greater than 1. And the algorithm could have a higher chances than CAHCM. When the rate was less than 1, it meant the assessed algorithm had lower chances than CAHCM. As shown in the figure, our proposed CAHCM approach had an obvious advantage in gaining acceptable solutions.

In summary, according to our experimental results, our proposed approach had a great advantage in execution efficiency. The ratio of the optimal solutions was in an acceptable range.

7 Conclusions

This paper proposed a novel approach solving the problem of data allocations in cloud-based heterogeneous memories, which could be applied in big data for smart cities. The proposed model CAHCM was designed to enable to mitigate the big data processing to remote facilities by using cloud-based memories. The main algorithm was 2DA algorithm that could output optimal solutions at a high rate. Our proposed approach has been assessed in the experimental evaluations, in which performs reached the desired level.

References

[1] V. Aggarwal, V. Gopalakrishnan, R. Jana, K. Ramakrishnan, and V. Vaishampayan. Optimizing cloud resources for delivering IPTV services through virtualization. *IEEE Transactions on Multimedia*, 15(4):789–801, 2013.

[2] F. Hao, G. Min, J. Chen, F. Wang, M. Lin, C. Luo, and L. T. Yang. An optimized computational model for multi-community-cloud social collaboration. *IEEE Transactions on Services Computing*, 7(3):346–358, 2014.

[3] M. Qiu, M. Zhong, J. Li, K. Gai, and Z. Zong. Phase-change memory optimization for green cloud with genetic algorithm. *IEEE Transactions on Computers*, 64(12):3528 – 3540, 2015.

[4] Y. Song, Y. Sun, and W. Shi. A two-tiered on-demand resource allocation mechanism for vm-based data centers. *IEEE Transactions on Services Computing*, 6(1):116–129, 2013.

[5] Y. Guo, Q. Zhuge, J. Hu, M. Qiu, and E. Sha. Optimal data allocation for scratch-pad memory on embedded multi-core systems. In *2011 Int'l Conf. on Parallel Processing (ICPP)*, pages 464–471, 2011.

[6] N. Vujic, F. Cabarcas, M. Gonzalez, A. Ramirez, X. Martorell, and E. Ayguade. Dma++: On the fly data realignment for on-chip memories. *IEEE Transactions on Computers*, 61(2):237–250, 2012.

[7] K. Gai and S. Li. Towards cloud computing: a literature review on cloud computing and its development trends. In *2012 Fourth Int'l Conf. on Multimedia Information Networking and Security*, pages 142–146, Nanjing, China, 2012.

[8] J. Espadas, A. Molina, G. Jiménez, M. Molina, R. Ramírez, and D. Concha. A tenant-based resource allocation model for scaling Software-as-a-Service applications over cloud computing infrastructures. *Future Generation Computer Systems*, 29(1):273–286, 2013.

[9] L. Wu, S. Garg, S. Versteeg, and R. Buyya. SLA-Based resource provisioning for hosted Software-as-a-Service applications in cloud computing environments. *IEEE Transactions on Services Computing*, 7(3):465–485, 2014.

[10] C. Alcaraz and J. Aguado. MonPaaS: an adaptive monitoring Platform as a Service for cloud computing infrastructures and services. *IEEE Transactions on Services Computing*, 8(1):65–78, 2015.

[11] X. Wu, X. Zhu, G. Wu, and W. Ding. Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):97–107, 2014.

[12] C. Liu, J. Chen, L. T. Yang, X. Zhang, C. Yang, R. Ranjan, and K. Rao. Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates. *IEEE Transactions on Parallel and Distributed Systems*, 25(9):2234–2244, 2014.

[13] M. Sookhak, A. Gani, M. Khan, and R. Buyya. Dynamic remote data auditing for securing big data storage in cloud computing. *Information Sciences*, 2015.

[14] Z. Wei, G. Pierre, and C. Chi. CloudTPS: Scalable transactions for Web applications in the cloud. *IEEE Transactions on Services Computing*, 5(4):525–539, 2012.

[15] C. Wang, N. Cao, K. Ren, and W. Lou. Enabling secure and efficient ranked keyword search over outsourced cloud data. *IEEE Transactions on Parallel and Distributed Systems*, 23(8):1467–1479, 2012.

[16] M. Qiu, L. Chen, Y. Zhu, J. Hu, and X. Qin. Online data allocation for hybrid memories on embedded tele-health systems. In *IEEE Int'l Conf. on High Performance Computing and Communications*, pages 574–579, Paris, 2014.

[17] S. Yi, A. Andrzejak, and D. Kondo. Monetary cost-aware checkpointing and migration on Amazon cloud spot instances. *IEEE Transactions on Services Computing*, 5(4):512–524, 2012.

[18] J. Zhan, L. Wang, X. Li, W. Shi, C. Weng, W. Zhang, and X. Zang. Cost-aware cooperative resource provisioning for heterogeneous workloads in data centers. *IEEE Transactions on Computers*, 62(11):2155–2168, 2013.

[19] M. Cheung, J. She, and Z. Jie. Connection discovery using big data of user-shared images in social media. *IEEE Transactions on Multimedia*, 17(9):1417–1428, 2015.

[20] K. Aizawa, Y. Maruyama, H. Li, and C. Morikawa. Food balance estimation by using personal dietary tendencies in a multimedia food log. *IEEE Transactions on Multimedia*, 15(8):2176–2185, 2013.

[21] P. Zhang, Y. Gao, and M. Qiu. A data-oriented method for scheduling dependent tasks on high-density multi-GPU systems. In *2015 IEEE 17th International Conference on High Performance Computing and Communications*, pages 694–699, New York, NY, USA, 2015. IEEE.

[22] P. Zhang, L. Liu, and Y. Deng. A data-driven paradigm for mapping problems. *Parallel Computing*, 48:108–124, 2015.

[23] J. Yu, Y. Rui, and D. Tao. Click prediction for web image reranking using multimodal sparse coding. *IEEE Transactions on Image Processing*, 23(5):2019–2032, 2014.

[24] J. Park and J. Yoo. Preprocessing techniques for high-efficiency data compression in wireless multimedia sensor networks. *Advances in Multimedia*, 2015:1, 2015.

[25] M. Chen, J. Han, and P. Yu. Data mining: an overview from a database perspective. *IEEE Transactions on Knowledge and data Engineering*, 8(6):866–883, 1996.

[26] Y. Sakurai, Y. Matsubara, and C. Faloutsos. Mining and forecasting of big time-series data. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 919–922, Melbourne, Victoria, Australia, 2015. ACM.

[27] J. Yu, Y. Rui, and B. Chen. Exploiting click constraints and multi-view features for image re-ranking. *IEEE Transactions on Multimedia*, 16(1):159–168, 2014.

[28] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong. Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing. *Journal of Network and Computer Applications*, 59:46–54, 2016.

[29] A. Zhou and B. He. Transformation-based monetary cost optimizations for workflows in the cloud. *IEEE Transactions on Cloud Computing*, 2(1):85–98, 2014.

[30] X. Zhu, L. Yang, H. Chen, J. Wang, S. Yin, and X. Liu. Real-time tasks oriented energy-aware scheduling in virtualized clouds. *IEEE Transactions on Cloud Computing*, 2(2):168–180, 2014.

[31] Y. Gu, Y. Fu, A. Prakash, Z. Lin, and H. Yin. Multi-aspect, robust, and memory exclusive guest OS fingerprinting. *IEEE Transactions on Cloud Computing*, 2(4):380–394, 2014.

[32] M. Qiu, K. Gai, B. Thuraisingham, L. Tao, and H. Zhao. Proactive user-centric secure data scheme using attribute-based semantic access controls for mobile clouds in financial industry. *Future Generation Computer Systems*, PP:1, 2016.

[33] J. Li, X. Tan, X. Chen, D. Wong, and F. Xhafa. Opor: enabling proof of retrievability in cloud computing with resource-constrained devices. *IEEE Transactions on Cloud Computing*, 3(2):195–205, 2015.

[34] R. Duan, R. Prodan, and X. Li. Multi-objective game theoretic scheduling of bag-of-tasks workflows on hybrid clouds. *IEEE Transactions on Cloud Computing*, 2(1):29–42, 2014.

[35] Q. Zhang, M. Zhani, R. Boutaba, and J. Hellerstein. Dynamic heterogeneity-aware resource provisioning in the cloud. *IEEE Transactions on Cloud Computing*, 2(1):14–28, 2014.

[36] F. Zhang, J. Cao, K. Hwang, K. Li, and S. Khan. Adaptive workflow scheduling on cloud computing platforms with iterativeordinal optimization. *IEEE Transactions on Cloud Computing*, 3(2):156–168, 2015.

[37] R. Xie and X. Jia. Data transfer scheduling for maximizing throughput of big-data computing in cloud systems. *IEEE Transactions on Cloud Computing*, PP(99):1, 2015.

[38] J. Hu, C. Xue, Q. Zhuge, W. Tseng, and E. H. Sha. Data allocation optimization for hybrid scratch pad memory with SRAM and nonvolatile memory. *IEEE Transactions on Very Large Scale Integration Systems*, 21(6):1094–1102, 2013.

[39] Y. Guo, Q. Zhuge, J. Hu, J. Yi, M. Qiu, and E. Sha. Data placement and duplication for embedded multicore systems with scratch pad memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(6):809–817, 2013.

[40] P. Panda, N. Dutt, and A. Nicolau. On-chip vs. off-chip memory: the data partitioning problem in embedded processor-based systems. *ACM Transactions on Design Automations of Electronic Systems (TODAES)*, 5(3):682–704, 2000.

[41] C. Cicconetti, L. Lenzini, A. Lodi, S. Martello, E. Mingozzi, and M. Monaci. Efficient two-dimensional data allocation in IEEE 802.16 OFDMA. *IEEE/ACM Transactions on Networking*, 22(5):1645–1658, 2014.

[42] M. Vidyasagar. A metric between probability distributions on finite sets of different cardinalities and applications to order reduction. *IEEE Transactions on Automatic Control*, 57(10):2464–2477, 2012.