IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 3, 2018, accepted April 8, 2018, date of publication April 11, 2018, date of current version May 2, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2825648

Reducing Energy Consumption With Cost Budget Using Available Budget Preassignment in Heterogeneous Cloud Computing Systems

YUEKUN CHEN, GUOQI XIE, (Member, IEEE), AND RENFA LI, (Senior Member, IEEE)
Key Laboratory for Embedded and Network Computing of Hunan Province, College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China

Corresponding author: Guoqi Xie (xgqman@hnu.edu.cn)

This work was supported in part by the National Key Research and Development Plan of China under Grant 2016YFB0200405, in part by the National Natural Science Foundation of China under Grant 61702172 and Grant 61672217, in part by the Natural Science Foundation of Hunan Province, China, under Grant 2018JJ3076, in part by the China Postdoctoral Science Foundation under Grant 2016M592422, in part by the CCF-Venustech Open Research Fund under Grant CCF-VenustechRP2017012, in part by the Open Research Project of the State Key Laboratory of Industrial Control Technology, Zhejiang University, China, under Grant ICT1800399, and in part by the Fundamental Research Funds for the Central Universities.

ABSTRACT Energy consumption and cost are the important factors in executing applications in grid or cloud computing systems, because they directly affect resource consumption and economic benefits. This paper solves the problem of reducing energy consumption of a cost budgeted directed acyclic graph (DAG) application in heterogeneous computing systems. The state-of-the-art work has studied the cost budgeted scheduling for a DAG application in the heterogeneous computing systems by proposing the budget level-based preassignment method; however, this paper is merely to reduce the schedule length without involving energy consumption, and its preassignment method is still pessimistic although it improves the existing method. In this paper, we first propose an available budget preassignment method to further improve it. We then introduce the available budget preassignment method to reduce the energy consumption. We propose minimizing energy consumption using the available budget preassignment (MECABP) algorithm based on the two steps. Experiments on three types of DAG applications with different parallelism degrees confirm the effectiveness of the proposed MECABP algorithm compared with existing algorithms.

INDEX TERMS Cost budget, budget preassignment, heterogeneous cloud computing systems.

I. INTRODUCTION

A. MOTIVATION

Cloud computing is a promising, effective computing model for supporting scientific applications [1]–[4], which are frequently used in modeling scientific applications in the fields of bioinformatics, astronomy, and physics [5]. Heterogeneous cloud computing systems are special systems in which different virtual machines (VMs) have varying computation capacities because old and slow machines are continuously being replaced by new and fast machines [6]. In heterogeneous cloud computing systems, an application is commonly modeled as a set of tasks interconnected via data or computing dependencies and is described as a directed acyclic graph (DAG) [7]–[10] Examples of DAG applications are Gaussian elimination and fast Fourier transform [8], [9], [11].

Energy consumption and cost are important factors in executing DAG applications in cloud computing systems because they directly affect resource utilization and economic benefits. Users and service providers are the types of roles in cloud computing systems, and conflicting requirements exist between them by the server-level agreement (SLA) [8]. An user submits an application with a given cost budget as its quality of service (QoS) requirement to the cloud data center supplied by service providers. Cost budgeted application refers to an application with a limited cost budget, which is the maximum cost that the user can pay. Many cost budgeted scheduling algorithms have been extensively studied in computing systems [5], [7], [8], [10], [12], [13]. However, these studies do not involve energy consumption. As the user tends to focus on low cost and service providers tend to focus on low energy consumption, combining the two factors is essential. In cloud computing systems, cost calculation typically uses pay-as-you-go mode [8], [12], [14], and energy consumption reduction by using dynamic voltage and frequency

scaling (DVFS) [11], [15], [16]. However, dynamically scaling down the voltage to reduce consumption will increase execution time, thereby increasing execution costs.

This study aims to solve the problem of reducing energy consumption of a cost budgeted application in heterogeneous cloud computing systems. The state-of-the-art work has studied the cost budgeted scheduling for a DAG application by proposing the budget level-based preassignment method [8]. However, [8] is merely to reduce the schedule length without involving energy consumption, and its preassignment method is still pessimistic although it improves the existing method of [7] (refer to Section IV.A for more details).

B. OUR CONTRIBUTION

This study aims to reduce the energy consumption of a cost budgeted application in heterogeneous cloud computing systems. We divide the problem into two sub-problems: 1) satisfying cost budget, and 2) reducing energy consumption. The main contributions of this study are as follows:

(1) Satisfying cost budget. We first propose an available budget preassignment method to improve the existing budget preassignment methods. In this step, we transfer the cost budget of the application to each task by using the available budget preassignment.

(2) Reducing energy consumption. We introduce the available budget preassignment method to reduce the energy consumption. In this step, we allow each task to select the combination assignment of VM and frequency that has the minimum energy consumption while satisfying the cost budget of the task.

(3) We propose the minimizing energy consumption using available budget preassignment (MECABP) algorithm based on the two steps.

(4) Experiments on three types of DAG applications with different parallelism degrees confirm the effectiveness of the proposed MECABP algorithm compared with existing algorithms.

The rest of this paper is organized as follows. Section II reviews related works. Section III shows related models and problem statement. Section IV presents the MECABP algorithm. Section V evaluates the performance of the MECABP algorithm. Section VI concludes this study.

II. RELATED WORK

Considering that this study involves reducing energy consumption with cost budget of a DAG application, we review related works organized by cost scheduling and energy scheduling, respectively.

A. COST SCHEDULING OF A DAG APPLICATION

Mao and Humphrey [17] proposed the autoscaling computational instances approach to optimize the cost of a schedule length constrained DAG application in cloud computing systems. Abrishami *et al.* [18] solved the problem of minimizing the cost of a schedule length constrained DAG application on homogeneous computing systems. However, both [17] and [18] aim to minimize cost with schedule length constraint rather to minimize schedule length with cost budget. Wu *et al.* [19] studied the problem of minimizing the schedule length of a cost budgeted DAG application by proposing a critical-greedy approach and introducing a budget level (BL); however BL is only used in homogeneous cloud computing systems. Arabnejad and Barbosa [7] studied the same problem as that reported in [19] on heterogeneous computing systems and proposed the heterogeneous budget constrained scheduling (HBCS) algorithm. HBCS is a pessimistic algorithm because it preassigns the minimum costs to unassigned tasks. Chen *et al.* [8] extended the BL in [19] to heterogeneous computing systems and proposed the minimizing schedule length using BL (MSLBL) algorithm. However, as mentioned in Section I.A, MSLBL is merely to reduce the schedule length without involving energy consumption, and its preassignment is still pessimistic although it improves the HBCS algorithm.

B. ENERGY SCHEDULING OF A DAG APPLICATION

DVFS-based energy design techniques have been used to execute a DAG application. These techniques and algorithms aim to minimize energy consumption with schedule length budget or minimize schedule length with energy consumption budget. Lee and Zomaya [20] presented energy-conscious scheduling to implement joint reduction between the schedule length and energy consumption of a DAG application on heterogeneous computing systems. Xiao *et al.* [21] studied the problem of reducing schedule length of an energy consumption constrained DAG application on heterogeneous computing systems by preassigning minimum energy values to unassigned tasks. Huang *et al.* [22] studied the problem of reducing energy consumption of a schedule length constrained DAG application on heterogeneous computing systems by reclaiming the slack time using upward approach [22]. Xie *et al.* [23] studied the same problem as that reported in [22] by proposing a combination of downward and upward approaches. Tang *et al.* [24] proposed the method of turning off inefficient VMs on the basis of [22]. Xie *et al.* [25] improved the limitations of [24] by proposing energy-aware VM merging algorithms.

Unlike the above related works, this study will study the new problem of minimizing energy consumption of a cost budgeted application. Particularly, we will propose a new budget preassignment method to improve the existing methods.

III. MODELS

A. DAG APPLICATION MODEL

A targeted cloud computing platform is composed of a set of heterogeneous VMs to provide services with different capabilities and costs. Let $U = \{u_1, u_2, \ldots, u_{|U|}\}$ denote a set of heterogeneous VMs, where $|U|$ represents the size of set U. For any set X, $|X|$ is used to denote its size. We assume that communication can overlap with computation, which means

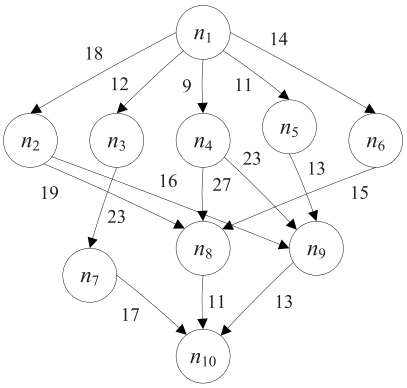


FIGURE 1. Motivational example of a DAG application with ten tasks [8], [11], [23], [25], [26].

that data can be transmitted from one VM to another while a task is being executed on the recipient VM. A work running on VMs is represented by a DAG $G = (N, W, M, C)$ [7]–[9], [21], [25], [26].

(1) N represents a set of nodes in G, and each node $n_i \in N$ represents a task. $pred(n_i)$ represents the set of the immediate predecessor tasks of n_i. $succ(n_i)$ represents the set of the immediate successor tasks of n_i. The task that has no predecessor task is denoted as n_{entry}, and the task that has no successor task is denoted as n_{exit}. If a DAG application has multiple entry or exit tasks, then a dummy entry or exit task with zero-weight dependencies is added to the graph.

(2) W is a $|N| \times |U|$ matrix, where $w_{i,k}$ denotes the execution time of n_i running on u_k [7]–[9], [21], [25], [26]. Each task has different execution time values on different VMs due to the heterogeneity of the VMs.

(3) M is a set of communication edges, and each edge $m_{i,j} \in M$ represents the communication message from n_i to n_j.

(4) $c_{i,j} \in C$ represents the communication time of $m_{i,j}$ if n_i and n_j are not assigned to the same VM. All computation and storage services are assumed to be in the same physical region. Hence, communication time $c_{i,j}$ depends on the amount of data to be transferred between task n_i and task n_j, which is independent of the computation service on the VMs [8]. When tasks n_i and task n_j are assigned to the same VM, $c_{i,j}$ is zero because the intra-VM communication cost can be ignored.

Fig. 1 shows a motivational example of a DAG application. Table 1 shows a matrix of execution time values in Fig. 1. The example shows 10 tasks executed on 3 VMs $\{u_1, u_2, u_3\}$. The weight 16 of n_1 and u_2 in Table 1 represents the execution time with the maximum frequency denoted by $w_{1,2} = 16$. The weight 18 of the edge between n_1 and n_2 represents the communication time denoted as $c_{1,2}$ if n_1 and n_2 are not assigned to the same VM [7], [8], [11], [23], [25], [26]. All the time unit is omitted in this example for simplicity.

TABLE 1. Execution time values of tasks on different VMs with the maximum frequencies of the DAG application in Fig. 1 [8], [11], [23], [25], [26].

Task	u_1	u_2	u_3
n_1	14	16	9
n_2	13	19	18
n_3	11	13	19
n_4	13	8	17
n_5	12	13	10
n_6	13	16	9
n_7	7	15	11
n_8	5	11	14
n_9	18	12	20
n_{10}	21	7	16

B. ENERGY MODEL

Considering the linear relationship between voltage and frequency, DVFS decreases the voltage and frequency to save energy. Similar to [11], [21], [23], and [25], we use the term frequency change to represent the process of changing the voltage and frequency simultaneously. Considering a DVFS-capable system, we also adopt the system-level power model that is widely used in [11], [21], [23], and [25], in which the power consumption at frequency f is given by

$$P(f) = P_{\text{s}} + h(P_{\text{ind}} + P_{\text{d}}) = P_{\text{s}} + h(P_{\text{ind}} + C_{\text{ef}} f^m),$$

where P_s represents the static power that can only be removed by powering off the entire system. P_{ind} represents frequency-independent dynamic power that can be removed by putting the system into the sleep state. P_{d} represents frequency-dependent dynamic power. h represents system states and indicates whether dynamic power is currently consumed by the system. When the system is active, $h = 1$; otherwise, $h = 0$. C_{ef} represents an effective switching capacity, and m represents the dynamic power exponent that is not less than 2. Both C_{ef} and m are VM-dependent constants.

In this study, we assume that the system is always turned on because turning on/off a system causes excessive overhead. In other words, P_{s} is always consumed and is unmanageable. Similar to [11], [21], [23], and [25], this study concentrates on dynamic power (e.g., P_{ind} and P_{d}) and ignores the P_{s} in the calculation. Less P_{d} does not result in less energy consumption because of the P_{ind}, that is, a minimum energy-efficient frequency f_{ee} exists [11], [21], [23], [25] and is denoted as

$$f_{\text{ee}} = \sqrt[m]{\frac{P_{\text{ind}}}{(m-1)C_{\text{ef}}}}. \tag{1}$$

Assuming the frequency of a VM varies from a minimum available frequency $f_{\min}$ to the maximum frequency $f_{\max}$, the lowest frequency to execute a task is

$$f_{\text{low}} = \max(f_{\min}, f_{\text{ee}}). \tag{2}$$

Therefore, any actual effective frequency f_h should belong to the scope of $f_{\text{low}} \leqslant f_{\text{h}} \leqslant f_{\max}$.

Considering that the number of VMs is $|U|$ in the system and these VMs are completely heterogeneous,

each VM should have individual power parameters. Here, we define frequency-independent dynamic power set

$$\{P_{1,\text{ind}}, P_{2,\text{ind}}, \ldots, P_{|U|,\text{ind}}\},$$

frequency-dependent dynamic power set

$$\{P_{1,\text{d}}, P_{2,\text{d}}, \ldots, P_{|U|,\text{d}}\},$$

effective switching capacitance set

$$\{C_{1,\text{ef}}, C_{2,\text{ef}}, \ldots, C_{|U|,\text{ef}}\},$$

dynamic power exponent set

$$\{m_1, m_2, \ldots, m_{|U|}\},$$

minimum energy-efficient frequency set

$$\{f_{1,\text{ee}}, f_{2,\text{ee}}, \ldots, f_{|U|,\text{ee}}\},$$

and actual effective frequency set

$$\left\{\begin{array}{l}\{f_{1,\text{low}}, f_{1,\alpha}, f_{1,\beta}, \ldots, f_{1,\max}\},\\ \{f_{2,\text{low}}, f_{2,\alpha}, f_{2,\beta}, \ldots, f_{2,\max}\},\\ \ldots,\\ \{f_{|U|,\text{low}}, f_{|U|,\alpha}, f_{|U|,\beta}, \ldots, f_{|U|,\max}\}\end{array}\right\}.$$

We let $E(n_i, u_k, f_{k,h})$ represent the energy consumption of the task n_i on the VM u_k with frequency $f_{k,h}$. This set is calculated by

$$E(n_i, u_k, f_{k,h}) = \left(P_{k,\text{ind}} + C_{k,\text{ef}} \times {f_{k,h}}^{m_k}\right) \times w_{i,k} \times \frac{f_{k,\max}}{f_{k,h}}. \tag{3}$$

The energy consumption of the DAG application G is calculated by

$$E(G) = \sum_{i=1}^{|N|} E(n_i) = \sum_{i=1}^{|N|} E\left(n_i, u_{pr(i)}, f_{pr(i),hz(i)}\right), \tag{4}$$

where $u_{pr(i)}$ and $f_{pr(i),hz(i)}$ represent the assigned VM and frequency of n_i, respectively. In this study, the overheads of the frequency transitions are ignored because of the negligible amount of time (e.g., 10 μs-150 μs [20], [25]).

C. COST MODEL

The cost model in this study is based on a pay-as-you-go mode, and users are charged based on to the amount of time that they used VMs according to the current commercial clouds [8], [12]. Each VM has an individual unit price because VMs in the system are completely heterogeneous [7], [8], [10]. We let r_k be the unit price of the assigned computing service on VM u_k. Accordingly, we formally define the cost $cost(n_i, u_k, f_{k,h})$ of task n_i on VM u_k with frequency $f_{k,h}$ and is calculated by

$$cost(n_i, u_k, f_{k,h}) = w_{i,k} \times \frac{f_{k,\max}}{f_{k,h}} \times r_k. \tag{5}$$

The total cost of the DAG application G is calculated by

$$cost(G) = \sum_{i=1}^{|N|} cost(n_i) = \sum_{i=1}^{|N|} cost(n_i, u_{pr(i)}, f_{pr(i),hz(i)}). \tag{6}$$

D. COST BUDGET SCOPE

As the maximum frequency means minimum execution time, and lowest frequency means the maximum execution time for a task on the same VM, we can obtain the minimum and maximum costs of task n_i, denoted by $cost_{\min}(n_i)$ and $cost_{\max}(n_i)$, respectively, by traversing all the VMs

$$cost_{\min}(n_i) = \min_{u_k \in U} cost(n_i, u_k, f_{k,\max}), \tag{7}$$

and

$$cost_{\max}(n_i) = \max_{u_k \in U} cost(n_i, u_k, f_{k,\text{low}}). \tag{8}$$

Then, the total cost of the DAG application G is the sum of that of each task, and the minimum and maximum costs of the DAG application are

$$cost_{\min}(G) = \sum_{i=1}^{|N|} cost_{\min}(n_i), \tag{9}$$

and

$$cost_{\max}(G) = \sum_{i=1}^{|N|} cost_{\max}(n_i), \tag{10}$$

respectively.

The cost budget of the DAG application $cost_{budget}(G)$ must be larger than or equal to $cost_{\min}(n_i)$; otherwise, $cost_{budget}(G)$ is always unsatisfied. Meanwhile, $cost_{budget}(G)$ should be less than or equal to $cost_{\max}(n_i)$; otherwise, $cost_{budget}(G)$ is always satisfied. Therefore, this study assumes that $cost_{budget}(G)$ belongs to the scope of $cost_{\min}(G)$ and $cost_{\max}(G)$:

$$cost_{\min}(G)) \leq cost_{budget}(G) \leq cost_{\max}(G). \tag{11}$$

E. PROBLEM STATEMENT

The problem to be solved is to assign a combination of VM and frequency for each task to reduce the energy consumption of the DAG application

$$E(G) = \sum_{i=1}^{|N|} E(n_i) = \sum_{i=1}^{|N|} E\left(n_i, u_{pr(i)}, f_{pr(i),hz(i)}\right), \tag{12}$$

subject to its cost budget

$$cost(G) \leqslant cost_{\text{budget}}(G). \tag{13}$$

Considering that scheduling tasks with quality of service (QoS) requirement for optimality on multiprocessors is known to be an NP-hard optimization problem [27], the problem to be solved in this study is also an NP-hard optimization problem.

IV. REDUCING ENERGY CONSUMPTION WITH COST BUDGET

A. EXISTING COST BUDGETED SCHEDULING

Although no research is exactly the same as this paper (i.e., reducing energy consumption with cost budget), the idea

of reducing the schedule length with cost budget can be referred to [7] and [8].

Assuming that the task to be assigned is $n_{s(y)}$, where $n_{s(y)}$ represents the yth assigned task, then $\{n_{s(1)}, n_{s(2)}, \ldots, n_{s(y-1)}\}$ represents the task set where the tasks have been assigned (i.e., high-priority tasks correspond to $n_{s(y)}$), and $\{n_{s(y+1)}, n_{s(y+2)}, \ldots, n_{s(|N|)}\}$ represents the task set where the tasks are unassigned (i.e., low-priority tasks correspond to $n_{s(y)}$) [8]. Initially, all tasks of the DAG application are unassigned. Tasks are prioritized according to the decreasing order of upward rank value ($rank_{\mathrm{u}}$) [26]

$$rank_{\mathrm{u}}(n_i) = \overline{w_i} + \max_{n_j \in succ(n_i)} \{c_{i,j} + rank_{\mathrm{u}}(n_j)\}, \tag{14}$$

which is considered the de facto prioritizing task criterion of list scheduling for a DAG application on heterogeneous computing systems because it is widely used in energy and cost budget scheduling [7], [8], [11], [23], [25].

To ensure that the cost budget of the DAG application is satisfied at each task assignment, the HBCS algorithm proposed in [7] presents the preassignment to each unassigned task $n_{s(z)}$ in $\{n_{s(z+1)}, n_{s(z+2)}, \ldots, n_{s(|N|)}\}$ with minimum cost of $cost_{\min}(n_{s(z)})$. When assigning $n_{s(y)}$, the cost budgeted value of the DAG application is expressed as follows:

$$cost(G) = \sum_{x=1}^{y-1} cost(n_{s(x)}) + cost(n_{s(y)}) + \sum_{z=y+1}^{|N|} cost_{\min}(n_{s(z)}),$$

where $cost(n_{s(x)})$ is the actual used cost of $n_{s(x)}$ and $cost_{\min}(n_{s(z)})$ is the preassigned cost of $n_{s(z)}$.

$cost(G)$ must be less than or equal to $cost_{\mathrm{budget}}(G)$ according to the problem statement. Therefore, we have

$$\sum_{x=1}^{y-1} cost(n_{s(x)}) + cost(n_{s(y)}) + \sum_{z=y+1}^{|N|} cost_{\min}(n_{s(z)}) \leqslant cost_{\mathrm{budget}}(G).$$

Therefore, the cost value of the task $n_{s(y)}$ should have the following constraint:

$$cost(n_{s(y)}) \leqslant cost_{\mathrm{budget}}(G) - \sum_{x=1}^{y-1} cost(n_{s(x)}) - \sum_{z=y+1}^{|N|} cost_{\min}(n_{s(z)}).$$

The cost budget of task $n_{s(y)}$ is

$$cost_{\mathrm{budget}}(n_{s(y)}) = cost_{\mathrm{budget}}(G) - \sum_{x=1}^{y-1} cost(n_{s(x)}) - \sum_{z=y+1}^{|N|} cost_{\min}(n_{s(z)}). \tag{15}$$

Eq. (15) indicates that the cost budget of the application is transferred to each task. As long as each task satisfies its cost budget of

$$cost(n_{s(y)}) \leqslant cost_{\mathrm{budget}}(n_{s(y)}),$$

then the cost budget of the application can also be satisfied.

The minimum cost preassignment method is pessimistic because preassigning with the minimum cost to unassigned low-priority tasks will cause the cost budgets of the high-priority tasks to be large, such that this method is severely pessimistic toward unfair frequency assignment and cost usage among tasks (see more details in Table 6), and thus results in limited reduction of the schedule length. Therefore, a more fair method called budget level preassignment was proposed in [8].

The main idea of the budget level preassignment is that the budget preassignment value of $n_{s(z)}$ is changed to $cost_{\mathrm{bl}}(n_{s(z)})$, which is calculated by

$$cost_{\mathrm{bl}}(n_{s(z)}) = cost_{\min}(n_{s(z)}) + \big(cost_{\max}(n_{s(z)}) - cost_{\min}(n_{s(z)})\big) \times BL(G), \tag{16}$$

where $BL(G)$ represents the budget level of the DAG application and is calculated by

$$BL(G) = \frac{cost_{\mathrm{budget}}(G) - cost_{\min}(G)}{cost_{\max}(G) - cost_{\min}(G)}. \tag{17}$$

When assigning $n_{s(y)}$, its cost budget is correspondingly changed to

$$cost_{\mathrm{budget}}(n_{s(y)}) = cost_{\mathrm{budget}}(G) - \sum_{x=1}^{y-1} cost(n_{s(x)}) - \sum_{z=y+1}^{|N|} cost_{\mathrm{bl}}(n_{s(z)}). \tag{18}$$

The budget level preassignment in [8] can greatly reduce the pessimism compared with the minimum cost preassignment [7]. However it is still pessimistic due to the following reason.

We can see that $cost_{\mathrm{bl}}(n_{s(z)})$ is related to both $cost_{\max}(n_{s(z)})$ and $cost_{\max}(G)$ in Eq. (16). However, these two values are unnecessary in DVFS-enabled systems due to the large frequency scaling scope, because

$$cost_{\max}(n_{s(z)}) \gg cost_{\min}(n_{s(z)}),$$

and

$$cost_{\max}(G) \gg cost_{\min}(G),$$

Therefore, $cost_{\mathrm{bl}}(n_{s(z)})$ will degenerate to

$$cost_{\mathrm{bl}}(n_{s(z)}) = cost_{\max}(n_{s(z)}) \times \frac{cost_{\mathrm{budget}}(G)}{cost_{\max}(G)}.$$

In this case, the high-priority tasks will have relative low cost usage, and low-priority tasks will have relative high cost usage (see more details in Table 7). In other words, the budget level preassignment is still pessimistic and can be improved further.

B. AVAILABLE BUDGET PREASSIGNMENT

Given the limitations in preassignment to unassigned tasks using MSLBL, this subsection presents the improved budget preassignment method. We first give a new definition as follows.

Definition 1 (Available Budget): Available energy is the available budget between $cost_{\text{budget}}(G)$ and $cost_{\min}(G)$ of the DAG application, as shown in Eq. (19):

$$AB(G) = cost_{\text{budget}}(G) - cost_{\min}(G). \tag{19}$$

Compared with the budget level preassignment, the main improvement in this subsection is that the preassignment of $n_{s(z)}$ is changed from $cost_{\text{bl}}(n_{s(z)})$ to $cost_{\text{avail}}(n_{s(z)})$, and thus, $cost_{\text{bl}}(n_{s(z)})$ is changed to $cost_{\text{avail}}(n_{s(z)})$ in calculating $cost_{\text{budget}}(n_{s(y)})$:

$$cost_{\text{budget}}(n_{s(y)}) = cost_{\text{budget}}(G) - \sum_{x=1}^{y-1} cost(n_{s(x)}) - \sum_{z=y+1}^{|N|} cost_{\text{avail}}(n_{s(z)}). \tag{20}$$

$cost_{\text{avail}}(n_{s(z)})$ is calculated by

$$cost_{\text{avail}}(n_{s(z)}) = \frac{cost_{\min}(n_{s(z)})}{cost_{\min}(G)} \times AB(G) + cost_{\min}(n_{s(z)}), \tag{21}$$

$cost_{\text{avail}}(n_{s(z)})$ can be further updated to the following by substituting Eq. (19) to Eq. (21):

According to Eq. (21), we let the available budget be preassigned to each task based on its cost proportion $\frac{cost_{\min}(n_{s(z)})}{cost_{\min}(G)}$.

Eq. (21) can be further updated to the folllowing by substituting Eq. (19) into it.

$$\begin{aligned} cost_{\text{avail}}(n_{s(z)}) &= \frac{cost_{\min}(n_{s(z)})}{cost_{\min}(G)} \times \left(cost_{\text{budget}}(G) - cost_{\min}(G)\right) \\ &\quad + cost_{\min}(n_{s(z)}) \\ &= cost_{\min}(n_{s(z)}) \times \frac{cost_{\text{budget}}(G)}{cost_{\min}(G)}. \end{aligned} \tag{22}$$

According to the Eq. (22), we distribute the available budget to unassigned tasks according to the cost proportion $\frac{cost_{\min}(n_{s(z)})}{cost_{\min}(G)}$. In this manner, the cost budget of the DAG application can still be transferred to each task while $cost_{\max}(G)$ does not appear in Eq. (22).

When assigning $n_{s(y)}$, its cost budget is correspondingly changed to

$$cost_{\text{budget}}(n_{s(y)}) = cost_{\text{budget}}(G) - \sum_{x=1}^{y-1} cost(n_{s(x)}) - \sum_{z=y+1}^{|N|} cost_{\text{avail}}(n_{s(z)}). \tag{23}$$

Compared with the minimum cost preassignment $cost_{\min}(n_{s(z)})$ and budget level preassignment $cost_{\max}(n_{s(z)}) \times \frac{cost_{\text{budget}}(G)}{cost_{\max}(G)}$, the available budget preassignment $cost_{\min}(n_{s(z)}) \times \frac{cost_{\text{budget}}(G)}{cost_{\min}(G)}$ only accepts the advantages of both, but also comprehensively improves the pessimism of both. The detailed comparison can be found in Section IV.E. We prove the correctness of the proposed available budget preassignment in Theorem 1.

Theorem 1: Each task in the DAG application G can always find an assignment of VMs to satisfy

$$\begin{aligned} cost(G) &= \sum_{x=1}^{y-1} cost(n_{s(x)}) + cost(n_{s(y)}) + \sum_{z=y+1}^{|N|} cost_{\text{avail}}(n_{s(z)}) \\ &\leqslant cost_{\text{budget}}(G). \end{aligned}$$

Proof: Considering that available budget preassignment $cost_{\text{avail}}(n_{s(z)})$ is considered the minimum cost of $n_{s(z)}$), if we can prove that the sum of the available budget preassignment of all tasks is less than or equal to the given cost budget of the DAG application, then the theorem is proven.

First, let the sum of the available budget preassignments of all tasks be

$$cost_{\text{avail}}(G) = \sum_{z=1}^{|N|} cost_{\text{avail}}(n_{s(z)}). \tag{24}$$

Then, substituting Eq. (22) into Eq. (24) obtains

$$\begin{aligned} &\prod_{z=1}^{|N|} cost_{\text{avail}}(n_{s(z)}) \\ &= \prod_{z=1}^{|N|} \left(cost_{\min}(n_{s(z)}) \times \frac{cost_{\text{budget}}(G)}{cost_{\min}(G)}\right) \\ &= \frac{cost_{\text{budget}}(G)}{cost_{\min}(G)} \times \prod_{z=1}^{|N|} cost_{\min}(n_{s(z)}) \\ &= \frac{cost_{\text{budget}}(G)}{cost_{\min}(G)} \times cost_{\min}(G) \\ &= cost_{\text{budget}}(G). \end{aligned}$$

Given that $cost_{\text{avail}}(G)$ is equal to $cost_{\text{budget}}(G)$ under the available budget preassignment, we can find assigned VMs to satisfy $cost_{\text{budget}}(G)$. Thus, Theorem 1 is proven. ■

C. REDUCING ENERGY CONSUMPTION

The MSLBL algorithm in [8] aims to reduce the schedule length rather than the energy consumption. In this subsection, we solve the problem of reducing energy consumption.

After transferring the cost budget of the DAG application to each task, these tasks are assigned to the VM with the minimum energy consumption while satisfying the cost budget of each task. The strategy is as follows: we simply need to traverse all the VMs and frequencies to assign the current task n_i to the combination of VM and frequency with the minimum energy consumption while satisfying the cost

budget of $cost(n_i, u_k, f_{k,h}) \leqslant cost_{\text{budget}}(n_i)$:

$$E(n_i, u_{pr(i)}, f_{pr(i),hz(i)}) = \min_{\substack{u_k \in U, f_{k,h} \in [f_{k,\text{low}}, f_{k,\max}], \\ cost(n_i,u_k,f_{k,h}) \leqslant cost_{\text{budget}}(n_i)}} \left\{E(n_i, u_k, f_{k,h})\right\} \quad (25)$$

The above expression is a heuristic search although it traverses all the VMs and frequencies. We find an efficient, good solution instead of the best solution because finding the best solution needs global search and is an NP-hard optimization problem. In this study, we conduct a heuristic search to find a partial optimization result. Given that the VM number and frequency levels are bounded, our heuristic search is effective and fast.

Considering that low frequency means low energy consumption and high cost for a task in the same VM, we can traverse the frequencies from $f_{k,\text{low}}$ to $f_{k,\max}$ on the VM u_k. When a frequency is found in a VM that has the minimum energy consumption, then the remaining frequencies in this VM can be skipped.

D. THE MECABP ALGORITHM

In the analysis of Sections IV.B and IV.C, the MECABP algorithm is presented, as shown in Algorithm 1.

Algorithm 1 The MECABP Algorithm

Input: $G = (N, W, M, C)$, U, $cost_{\text{budget}}(G)$
Output: $E(G)$, $cost(G)$ and its related values

1: Prioritize the tasks in a list *rank_list* based on $rank_{\text{u}}$ values;
2: **while** (*rank_list* is not null) **do**
3: $n_i \leftarrow n_{s(y)} \leftarrow rank\_list.out()$;
4: Calculate $cost_{\min}(G)$ by using Eq. (9);
5: Calculate available budget $AB(G)$ by using Eq. (19);
6: Calculate $cost_{\text{budget}}(n_i)$ by using Eq. (20);
7: $E(n_i) = +\infty$;
8: **for** (each VM $u_k \in U$) **do**
9: **for** (each frequency $f_{k,h}$ in $[f_{k,\text{low}}$ and $f_{k,\max}]$) **do**
10: Calculate $cost\left(n_i, u_k, f_{k,h}\right)$ for the task n_i;
11: **if** $(cost(n_i, u_k, f_{k,h}) \leqslant cost_{\text{budget}}(n_i))$ **then**
12: Calculate $E(n_i, u_k, f_{k,h})$ by using Eq. (3);
13: **if** $(E(n_i, u_k, f_{k,h}) < E(n_i))$ **then**
14: $E(n_i) \leftarrow E(n_i, u_k, f_{k,h})$;
15: $cost(n_i) \leftarrow cost(n_i, u_k, f_{k,h})$;
16: **end if**
17: **break**;
18: **end if**
19: **end for**
20: **end for**
21: **end while**
22: Calculate $cost(G)$ by using Eq. (6);
23: Calculate $E(G)$ by using Eq. (4);

The main idea of MECABP is that the cost budget of the DAG application is transferred to each task by preassigning available budget preassignment values to unassigned tasks.

TABLE 2. Unit prices of VMs [8].

u_k	r_k
u_1	3
u_2	5
u_3	4

TABLE 3. Power parameters of VMs (u_1, u_2, and u_3).

u_k	$P_{k,\text{ind}}$	$C_{k,\text{ef}}$	m_k	$f_{k,\text{low}}$	$f_{k,\max}$
u_1	0.03	0.8	2.9	0.26	1.0
u_2	0.04	0.7	2.5	0.27	1.0
u_3	0.07	1.0	2.5	0.29	1.0

(1) MECABP prioritizes the tasks based on $rank_{\text{u}}$ values in Line 1.

(2) MECABP iteratively assigns each task to the VM with the minimum energy consumption of the DAG application in Lines 2-21.

(3) MECABP gets the cost budget of n_i by using Eq. (20) in Lines 4-6.

(4) MECABP selects the effective combination assignment of VM and frequency for n_i while satisfying its cost budget in Lines 8-20.

(5) In Lines 22 and 23, MECABP calculates the related values of the DAG application.

The time complexity of the MECABP algorithm is $O(|N|^2 \times |U| \times |F|)$. The details are analyzed as follows:

(1) The energy consumption of the application is the sum of those of all the tasks, which can be performed within $O(|N|)$ time (the While loop in Lines 2-21).

(2) Selecting the combination assignment of VM and frequency needs $O(|N| \times |U| \times |F|)$ time (the For loop in Lines 8-20). $|F|$ means the maximum frequencies number on the VM.

E. EXAMPLE OF THE ALGORITHMS

This subsection illustrates the results of the motivational DAG application using HBCS, MSLBL. We assume that the unit prices of VMs are shown in Table 2 [8]. We assume that the power parameters for all VMs are known and shown in Table 3. The maximum frequency $f_{k,\max}$ for each VM is 1 and the frequency precision is 0.01. All the parameter units are ignored in the example for simplicity. The lowest energy-efficient frequency $f_{k,\text{low}}$ for each VM can be obtained according to Eq. (2). The minimum and maximum cost values are $cost_{\min}(G) = 344$ and $cost_{\max}(G) = 2681$ by using Eqs. (9) and (10), respectively, for the motivational DAG application. We set the cost budget of the motivational DAG application as $cost_{\text{budget}}(G) = 400$.

The original HBCS and MSLBL algorithms aim to minimize the schedule length and do not involve energy consumption. The results using HBCS and MSLBL have been shown in [8]. HBCS and MSLBL can also be applied to this study as long as the objective of reducing schedule length is changed to reducing energy consumption according to

TABLE 4. Available budget preassignment values of the tasks of the motivational application.

Task	n_1	n_2	n_3	n_4	n_5
$cost_{\text{avail}}(n_{s(z)})$	41.8605	45.3488	38.3721	45.3488	41.8605
Task	n_6	n_7	n_8	n_9	n_{10}
$cost_{\text{avail}}(n_{s(z)})$	41.8605	24.4186	17.4419	62.7907	40.6977

the guide of Section IV.C. The new algorithms are named as energy-efficient HBCS (EHBCS) and minimizing energy consumption using budget level (MECBL) in this study. The algorithm descriptions of the EHBCS and MECBL are show in Algorithms 2 and 3, respectively. The main difference among three algorithms is in calculating $cost_{\text{budget}}(n_i)$, which is calculated by Eqs. (20), (15), and (23) for MECAB, EHBCS, and MECBL, respectively.

Algorithm 2 The EHBCS Algorithm

Input: $G = (N, W, M, C)$, U, $cost_{\text{budget}}(G)$
Output: $E(G)$, $cost(G)$ and its related values

Prioritize the tasks in a list *rank_list* based on $rank_{\text{u}}$ values;
while (*rank_list* is not null) **do**
  $n_i \leftarrow n_{s(y)} \leftarrow rank\_list.out()$;
  Calculate $cost_{\text{budget}}(n_i)$ by using Eq. (15);
  $E(n_i) = +\infty$;
  for (each VM $u_k \in U$) **do**
    for (each frequency $f_{k,h}$ in $[f_{k,\text{low}}$ and $f_{k,\text{max}}]$) **do**
      Calculate $cost\left(n_i, u_k, f_{k,h}\right)$ for the task n_i;
      if $(cost(n_i, u_k, f_{k,h}) \leqslant cost_{\text{budget}}(n_i))$ **then**
        Calculate $E(n_i, u_k, f_{k,h})$ by using Eq. (3);
        if $(E(n_i, u_k, f_{k,h}) < E(n_i))$ **then**
          $E(n_i) \leftarrow E(n_i, u_k, f_{k,h})$;
          $cost(n_i) \leftarrow cost(n_i, u_k, f_{k,h})$;
        end if
        break;
      end if
    end for
  end for
end while
Calculate $cost(G)$ by using Eq. (6);
Calculate $E(G)$ by using Eq. (4);

Algorithm 3 The MECBL Algorithm

Input: $G = (N, W, M, C)$, U, $cost_{\text{budget}}(G)$
Output: $E(G)$, $cost(G)$ and its related values

Prioritize the tasks in a list *rank_list* based on $rank_{\text{u}}$ values;
while (*rank_list* is not null) **do**
  $n_i \leftarrow n_{s(y)} \leftarrow rank\_list.out()$;
  Calculate $cost_{\text{budget}}(n_i)$ by using Eq. (23);
  $E(n_i) = +\infty$;
  for (each VM $u_k \in U$) **do**
    for (each frequency $f_{k,h}$ in $[f_{k,\text{low}}$ and $f_{k,\text{max}}]$) **do**
      Calculate $cost\left(n_i, u_k, f_{k,h}\right)$ for the task n_i;
      if $(cost(n_i, u_k, f_{k,h}) \leqslant cost_{\text{budget}}(n_i))$ **then**
        Calculate $E(n_i, u_k, f_{k,h})$ by using Eq. (3);
        if $(E(n_i, u_k, f_{k,h}) < E(n_i))$ **then**
          $E(n_i) \leftarrow E(n_i, u_k, f_{k,h})$;
          $cost(n_i) \leftarrow cost(n_i, u_k, f_{k,h})$;
        end if
        break;
      end if
    end for
  end for
end while
Calculate $cost(G)$ by using Eq. (6);
Calculate $E(G)$ by using Eq. (4);

1) RESULTS OF THE MECABP ALGORITHM

Table 5 shows the combination assignment of VM and frequency of each task by using MECABP. The available budget $AB(G)$ of the DAG application is $400 - 344 = 56$ calculated by Eq. (19).

1) We calculate that the available budget preassignment $cost_{\text{avail}}(n_{s(z)})$ for each task is shown in Table 4 by using Eq. (22).

2) Each row in Table 5 represents the combination assignment of VM and frequency, cost value, and energy consumption of each task. For the first task n_1, its cost budget is 41.8605 calculated by Eq. (20) as follows:

$$\begin{aligned} &cost_{\text{budget}}(n_1) \\ &= cost_{\text{budget}}(G) - \sum_{z=2}^{10} cost_{\text{avail}}(n_{s(z)}) \\ &= 400 - 45.3488 - 38.3721 - 45.3488 - 41.8605 \\ &\quad - 41.8605 - 24.4186 - 17.4419 - 62.7907 - 40.6977 \\ &= 41.8605. \end{aligned}$$

We find that $cost_{\text{budget}}(n_1)$ is equal to $cost_{\text{avail}}(n_1)$, which indicates that these two values are equal for the first assigned task.

3) Then, n_1's combination assignment of VM and frequency is u_3 and 0.87, because this combination can generates the minimum energy consumption of 8.0275 calculated by Eq. (3) while satisfying its cost budget of 41.8605. In this case, the actual cost value is 41.3793.

4) The remaining tasks use the same pattern as n_1, as shown in Table 5. For example, the cost budget for n_3 is 38.8532 calculated by Eq. (20); n_3's combination assignment of VM and frequency is u_1 and 0.85.

5) The actual cost and final energy consumption of the DAG application G are 399.9325 and 63.8819 calculated by Eqs. (6) and (4), respectively.

To get intuitive comparison, we also list the results of the motivational DAG application using the EHBCS and MECBL algorithms.

TABLE 5. Combination assignments of VM and frequency for tasks of the motivational DAG application by using MECABP.

n_i	Task's cost budget $cost_{\text{budget}}(n_i)$	Combination assignment of VM and frequency $< u_{pr(n_i)}, f_{pr(n_i),hz(n_i)} >$	Actual cost $cost(n_i)$	Energy consumption $E(n_i)$
n_1	41.8605	$< u_3, 0.87 >$	41.3793	8.0275
n_3	38.8532	$< u_1, 0.85 >$	38.8235	6.8504
n_4	45.3786	$< u_2, 0.89 >$	44.9438	5.0614
n_2	45.7836	$< u_1, 0.86 >$	45.3488	8.2622
n_5	42.2952	$< u_1, 0.86 >$	41.8605	7.6267
n_6	42.2952	$< u_3, 0.86 >$	41.8605	7.9103
n_9	63.2254	$< u_2, 0.95 >$	63.1579	8.2832
n_7	24.4861	$< u_1, 0.86 >$	24.4186	4.4489
n_8	17.5094	$< u_1, 0.86 >$	17.4419	3.1778
n_{10}	40.7652	$< u_2, 0.86 >$	40.6977	4.2335
$cost(G) = 399.9325 < cost_{\text{budget}}(G) = 400, E(G) = 63.8819$				

TABLE 6. Combination assignments of VM and frequency for tasks of the motivational DAG application by using EHBCS.

n_i	Task's cost budget $cost_{\text{budget}}(n_i)$	Combination assignment of VM and frequency $< u_{pr(n_i)}, f_{pr(n_i),hz(n_i)} >$	Actual cost $cost(n_i)$	Energy consumption $E(n_i)$
n_1	92	$< u_1, 0.46 >$	91.3043	3.4743
n_3	33.6957	$< u_1, 0.98 >$	33.6735	8.8053
n_4	39.0222	$< u_1, 1.00 >$	39.0	10.79
n_2	39.0222	$< u_1, 1.00 >$	39	10.79
n_5	36.0222	$< u_1, 1.00 >$	36	9.96
n_6	36.0222	$< u_3, 1.00 >$	36	9.63
n_9	54.0222	$< u_1, 1.00 >$	54	14.94
n_7	21.0222	$< u_1, 1.00 >$	21	5.81
n_8	15.0222	$< u_1, 1.00 >$	15	4.15
n_{10}	35.0222	$< u_2, 1.00 >$	35	5.18
$cost(G) = 399.9778 < cost_{\text{budget}}(G) = 400, E(G) = 83.5297$				

2) RESULTS OF THE EHBCS ALGORITHM

Table 6 shows the combination assignment of VM and frequency of each task by using EHBCS. The actual cost and final energy consumption of the DAG application G using EHBCS are 399.9778 and 83.5297, respectively. We can see from Table 6 that only the high-priority tasks n_1 and n_3 achieve the frequency reduction adjustment while other tasks are all executed at the highest frequency of 1.0. EHBCS shows visible pessimism and its effect on energy reduction is limited in this example.

3) RESULTS OF THE MECBL ALGORITHM

Table 7 shows the combination assignment of VM and frequency of each task by using MECBL. The actual cost and final energy consumption of the DAG application G using MECBL are 399.7195 and 67.0192, respectively. We can see from Table 7 that the frequency scope is between 0.77 and 0.91 for all the tasks. Therefore, MECBL can reduce more energy consumption than EHBCS in general. However, compared with that the frequency scope is between 0.85 and 0.89 in Table 5 using MECABP, MECBL has certain disadvantage in energy reduction. MECABP achieves a more fair frequency assignment, thus effectively reducing the energy consumption.

F. SUMMARY OF ALGORITHMS

Table 8 shows the results of the motivational DAG application using EHBCS, MECBL, and MECABP algorithms.

TABLE 7. Combination assignments of VM and frequency for tasks of the motivational DAG application by using MECBL.

n_i	Task's cost budget $cost_{\text{budget}}(n_i)$	Combination assignment of VM and frequency $< u_{pr(n_i)}, f_{pr(n_i),hz(n_i)} >$	Actual cost $cost(n_i)$	Energy consumption $E(n_i)$
n_1	42.2363	$< u_3, 0.86 >$	41.8605	7.9103
n_3	38.8639	$< u_1, 0.85 >$	38.8235	6.8504
n_4	43.7238	$< u_2, 0.92 >$	43.4783	5.2895
n_2	46.741	$< u_1, 0.84 >$	46.4286	7.9316
n_5	41.2177	$< u_1, 0.88 >$	40.9091	7.939
n_6	42.5449	$< u_3, 0.85 >$	42.3529	7.7941
n_9	59.5074	$< u_1, 0.91 >$	59.3407	12.631
n_7	27.3187	$< u_1, 0.77 >$	27.2727	3.6809
n_8	19.567	$< u_1, 0.77 >$	19.4805	2.6292
n_{10}	40.0532	$< u_2, 0.88 >$	39.7727	4.3632
$cost(G) = 399.7195 < cost_{\text{budget}}(G) = 400, E(G) = 67.0192$				

TABLE 8. Results of the motivational DAG application using EHBCS, MECBL, and MECABP.

Algorithm	Method	$cost(G)$	$E(G)$	Time complexity
EHBCS	minimum cost preassignment	399.9778	83.5297	$O(\|N\|^2 \times \|U\| \times \|F\|)$
MECBL	budget level preassignment	399.7195	67.0192	$O(\|N\|^2 \times \|U\| \times \|F\|)$
MECABP	available budget preassignment	399.9325	63.8819	$O(\|N\|^2 \times \|U\| \times \|F\|)$

In Table 8, the following results can be observed:

(1) All the algorithms can satisfy the cost budget of the application, and the actual cost value is very close to the cost budget.

(2) EHBCS generates the highest energy consumption because its cost budget preassignment is pessimistic resulting in frequency reduction adjustment does not work for low-priority tasks.

(3) MECBL generates less energy consumption than EHBCS because it improves the cost budget preassignment compared with EHBCS. However, MECBL can be further improved.

(4) MECABP generates the lowest energy consumption among them because it not only accepts the advantages of EHBCS and MECBL, but also comprehensively improves the pessimism of both.

(5) All the algorithms have the time complexity of $O(|N|^2 \times |U| \times |F|)$.

V. EXPERIMENTAL RESULTS AND DISCUSSION

The metrics are the actual cost value $cost(G)$ (calculated by Eq. (6)) and final energy consumption $E(G)$ (calculated by Eq. (4)) of the DAG application. The algorithms compared with the proposed MECABP algorithm are aforementioned EHBCS [7] and MECBL [8].

The simulated heterogeneous cloud computing platform contains 64 VMs with different computing abilities and unit prices. As this study uses the VM specification of short-term lease (i.e., pay-as-you-go), the prices for VMs range from \$0.095 to \$0.38 per hour are based on the Amazon EC2 [28]. The execution time values of tasks and communication time values of messages could be within the scope of 1 h $\leq w_{i,k} \leq$ 128 h, 1 h $\leq c_{i,j} \leq$ 128 h [8]. The VM parameters taken from [25] are as follows: 10 ms$\leqslant w_{i,k} \leqslant$ 100 ms, 10 ms$\leqslant c_{i,j} \leqslant$ 100 ms, $0.03 \leqslant P_{k,\text{ind}} \leqslant 0.07$, $0.8 \leqslant C_{k,\text{ef}} \leqslant 1.2$, and

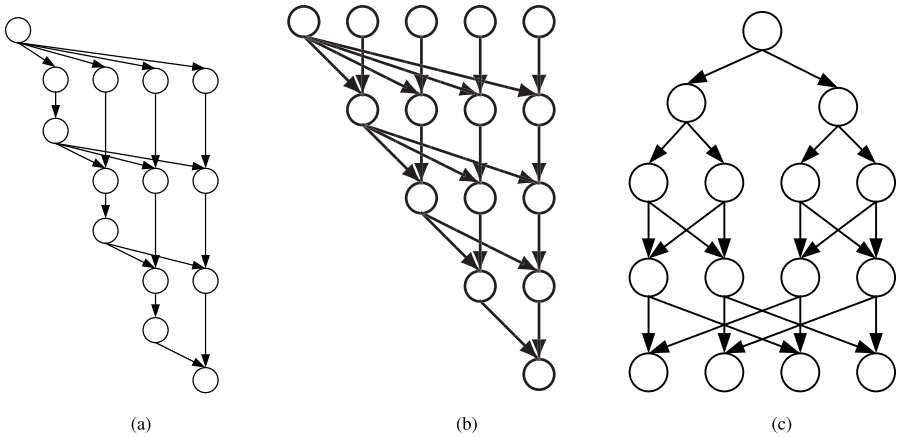


FIGURE 2. Three DAG applications. (a) Gaussian elimination. (b) Linear algebra. (c) Fast Fourier transform.

TABLE 9. Average schedule lengths (unit: h) of DAG applications using HEFT.

DAG application	Task number	Average schedule length (unit:h)
Gaussian elimination	1,175	5,484
Linear Algebra	1,176	3,973
Fast Fourier transform	1,151	1198

$2.5 \leqslant m_k \leqslant 3.0$. The aforementioned values are generated with uniform distribution. All frequencies are discrete, and the resolution is 0.01 GHz.

Many DAG applications are executed in cloud computing systems. We selected three representative DAG applications, namely, Gaussian elimination, linear algebra, and fast Fourier transform [11], [25], because they represent low-parallelism, middle-parallelism, and high-parallelism DAG applications, respectively. The parallelism degree can be identified by calculating the schedule lengths for approximate equal scales of DAG applications using the well-studied and commonly used heterogeneous earliest finish time (HEFT) algorithm [26], as shown in Table 9. A short schedule length corresponds to a high parallelism. The structures of the three DAG applications are shown in Figs. 2(a), 2(b), and 2(c), respectively.

Note that the values of experimental results are obtained by executing one run for one function. Many tests with the same parameter values and scales are preformed and show the same regular pattern and relatively stable results. In other words, experiments are repeatable and do not affect the consistency of the results.

A. GAUSSIAN ELIMINATION

The results of the Gaussian elimination application with 1,175 tasks are shown in Table 10. The cost budgets are changed from \$2,000 to \$10,000 with \$2,000 increments. The following observations are summarized and analyzed as follows:

(1) The actual costs are less than the given cost budgets in all the cases using all the three algorithms. All the actual costs obtained by all the algorithms are very close to the given cost budgets. These results confirm that all the three algorithms can satisfy the given cost budgets.

TABLE 10. Actual cost (unit: \$) and final energy consumption values (unit: GWh) of the Gaussian elimination with $\rho = 48$ ($|N| = 1,175$) for varying cost budgets (Experiment 1).

$\|N\|$	$cost_{\text{budget}}(G)$	EHBCS [7]		MECBL [8]		MECABP	
		$cost(G)$	$E(G)$	$cost(G)$	$E(G)$	$cost(G)$	$E(G)$
1,175	2,000	1,999.99	12,846.42	1,999.99	9,687.35	1,999.99	8,331.46
1,175	4,000	3,999.99	10,754.09	3,999.93	4,938.37	3,999.99	3,613.15
1,175	6,000	5,999.99	8,759.40	5,999.58	3,584.28	5,999.80	3,015.30
1,175	8,000	7,999.99	6,597.57	7,999.88	3,021.27	7,999.88	2,809.49
1,175	8,000	10,000	4,337.82	9,999.76	2,747.99	9,993.07	2,685.97

(2) With the increase of cost budgets, the energy consumptions are reduced gradually using all the algorithms. These results confirm that dynamically scaling down the voltage to reduce consumption will increase execution time, thereby increasing execution costs.

(3) MECABP obtains the minimum energy consumptions, flowed by MECBL and EHBCS in all the cases. When the cost budget is \$4,000. MECABP and MECBL can reduce 66.4% and 54.1% energy consumptions compared with EHBCS. MECABP can reduce 35.1% ($cost_{\text{budget}}(G) = 2,000$) to 66.4% ($cost_{\text{budget}}(G) = 4,000$) energy consumptions compared with EHBCS and can reduce 2.2% ($cost_{\text{budget}}(G) = 10,000$) to 26.8% ($cost_{\text{budget}}(G) = 4,000$) energy consumptions compared with EHBCS. These results confirm that the budget preassignment method of MECABP is more efficient than MECBL and EHBCS.

B. LINEAR ALGEBRA

Linear Algebra is a middle-parallelism DAG application. The results of the linear Algebra application with 1,176 tasks are shown in Table 11. The cost budgets are also changed from \$2,000 to \$10,000 with \$2,000 increments. A comparison between Tables 10 and 11 indicate that middle-parallelism linear Algebra shows an approximately equal regular pattern as low-parallelism Gaussian elimination in obtained

TABLE 11. Actual cost (unit: $) and final energy consumption values (unit: GWh) of the fast Fourier transform with $\rho = 128$ ($|N| = 1,176$) for varying cost budget (Experiment 2).

$\|N\|$	$cost_{\text{budget}}(G)$	EHBCS [7]		MECBL [8]		MECABP	
		$cost(G)$	$E(G)$	$cost(G)$	$E(G)$	$cost(G)$	$E(G)$
1,176	2,000	1,999.99	13,876.12	1,999.99	11,973.68	1,999.99	10,966.55
1,176	4,000	3,999.99	11,316.50	3,999.26	5,885.28	3,999.98	4,091.03
1,176	6,000	5,999.99	8,781.91	5,997.52	3,928.31	5,999.98	3,007.25
1,176	8,000	7,999.99	6,226.04	7,995.77	3,086.76	7,998.55	2,716.31
1,176	10,000	9,999.99	3,711.70	9,961.53	2,722.94	9,995.95	2,621.80

TABLE 12. Actual cost (unit: $) and final energy consumption values (unit: GWh) of the fast Fourier transform with $\rho = 128$ ($|N| = 1,151$) for varying cost budgets (Experiment 3).

$\|N\|$	$cost_{\text{budget}}(G)$	EHBCS [7]		MECBL [8]		MECABP	
		$cost(G)$	$E(G)$	$cost(G)$	$E(G)$	$cost(G)$	$E(G)$
1,151	2000	1,999.99	12,282.71	1,999.99	9,596.77	1,999.98	8,626.68
1,151	4,000	3,999.99	10,329.29	3,999.97	5,057.54	3,999.99	3,491.30
1,151	6,000	5,999.99	83,32.08	5,995.89	3,649.34	5,999.95	2,754.76
1,151	8,000	8,000.00	6,406.57	7,996.43	2,966.66	7,998.87	2,545.85
1,151	10,000	9,999.99	4,121.27	9,986.80	2,574.55	9,999.58	2,422.44

actual cost values and final energy consumptions. MECABP still generates the minimum energy consumption, followed by MECBL and EHBCS. MECABP can reduce 20.1% ($cost_{\text{budget}}(G) = 2,000$) to 65.7% ($cost_{\text{budget}}(G) = 6,000$) energy consumptions compared with EHBCS and can reduce 8.4% ($cost_{\text{budget}}(G) = 1,000$) to 30.5% ($cost_{\text{budget}}(G) = 2,000$) energy consumptions compared with MECBL. These results further indicate that MECABP is more energy-efficient than MECBL and EHBCS.

C. FAST FOURIER TRANSFORM

High-parallelism fast Fourier transform is used in an experiment. The results of the fast Fourier transform application with 1,151 tasks are shown in Table 12. The cost budgets are also changed from $2,000 to $10,000 with $2,000 increments. On the basis of the experimental results (Tables 10-12) of three types of DAG applications with different parallelism degrees, the following findings were obtained:

(1) Regardless of the parallelism degrees of DAG applications, the actual cost values basically have the same rule pattern in approximately equal task scales. MECABP, MECBL, and EHBCS can satisfy the cost budgets, and the actual cost values obtained by them are close to the cost budgets.

(2) Regardless of the parallelism degrees of DAG applications, the final energy consumption values basically have the same rule pattern in approximately equal task scales, namely, MECABP is the most energy-efficient, followed by MECBL and EHBCS.

VI. CONCLUSIONS

This study proposed an algorithm called MECABP to reduce the energy consumption of a cost budgeted DAG application in heterogeneous cloud computing systems. MECABP attempts to minimize energy consumption while satisfying the cost budget of the DAG application by proposing the available budget preassignment method. MECABP decomposes the problem into two sub-problems to implement the heuristic algorithm with low time complexity. MECABP demonstrates its effectiveness compared with existing algorithms through experiments based on three types of DAG applications with different parallelism degrees.

ACKNOWLEDGMENT

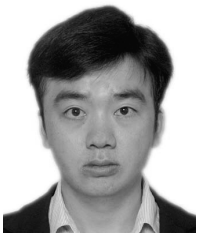
The authors would like to express their gratitude to the associate editor and three anonymous reviewers for their constructive comments which have helped to improve the quality of the paper.

REFERENCES

[1] K. Keahey, I. Raicu, K. Chard, and B. Nicolae, "Guest editors introduction: Special issue on scientific cloud computing," *IEEE Trans. Cloud Comput.*, vol. 4, no. 1, pp. 4–5, Jan./Mar. 2016.
[2] H. Li, K. Ota, M. Dong, A. Vasilakos, and K. Nagano, "Multimedia processing pricing strategy in GPU-accelerated cloud computing," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2017.2672554.
[3] T. Kumrai, K. Ota, M. Dong, J. Kishigami, and D. K. Sung, "Multiobjective optimization in cloud brokering systems for connected Internet of Things," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 404–413, Apr. 2017.
[4] K. Xie *et al.*, "Distributed multi-dimensional pricing for efficient application offloading in mobile cloud computing," *IEEE Trans. Services Comput.*, to be published, doi: 10.1109/TSC.2016.2642182.
[5] M. A. Rodriguez and R. Buyya, "Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 222–235, Apr./Jun. 2014.
[6] G. Xie *et al.*, "Minimizing redundancy to satisfy reliability requirement for a parallel application on heterogeneous service-oriented systems," *IEEE Trans. Services Comput.*, to be published, doi: 10.1109/TSC.2017.2665552.
[7] H. Arabnejad and J. G. Barbosa, "A budget constrained scheduling algorithm for workflow applications," *J. Grid Comput.*, vol. 12, no. 4, pp. 665–679, 2014.
[8] W. Chen, G. Xie, R. Li, Y. Bai, C. Fan, and K. Li, "Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems," *Future Generat. Comput. Syst.*, vol. 74, pp. 1–11, Sep. 2017.
[9] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 682–694, Mar. 2014.
[10] H. Arabnejad, J. G. Barbosa, and R. Prodan, "Low-time complexity budget–deadline constrained workflow scheduling on heterogeneous resources," *Future Generat. Comput. Syst.*, vol. 55, pp. 29–40, Feb. 2016.
[11] G. Xie, Y. Chen, X. Xiao, C. Xu, R. Li, and K. Li, "Energy-efficient fault-tolerant scheduling of reliable parallel applications on heterogeneous distributed embedded systems," *IEEE Trans. Sustain. Comput.*, to be published, doi: 10.1109/TSUSC.2017.2711362.
[12] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds," *Future Generat. Comput. Syst.*, vol. 29, no. 1, pp. 158–169, 2013.
[13] T. Wei *et al.*, "Cost-constrained QoS optimization for approximate computation real-time tasks in heterogeneous MPSoCs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, to be published, doi: 10.1109/TCAD.2017.2772896.
[14] H. Li, M. Dong, K. Ota, and M. Guo, "Pricing and repurchasing for big data processing in multi-clouds," *IEEE Trans. Emerg. Topics Comput.*, vol. 4, no. 2, pp. 266–277, Apr./Jun. 2016.
[15] S. Wang, Z. Qian, J. Yuan, and I. You, "A DVFS based energy-efficient tasks scheduling in a data center," *IEEE Access*, vol. 5, pp. 13090–13102, 2017.
[16] W. Huang, Z. Wang, M. Dong, and Z. Qian, "A two-tier energy-aware resource management for virtualized cloud computing system," *Sci. Programm.*, vol. 2016, p. 6, Oct. 2016.
[17] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal. (SC)*, Nov. 2011, pp. 1–12.
[18] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Cost-driven scheduling of grid workflows using partial critical paths," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1400–1414, Aug. 2012.

[19] C. Q. Wu, X. Lin, D. Yu, W. Xu, and L. Li, "End-to-end delay minimization for scientific workflows in clouds under budget constraint," *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 169–181, Apr./Jun. 2015.
[20] Y. C. Lee and A. Y. Zomaya, "Energy conscious scheduling for distributed computing systems under different operating conditions," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 8, pp. 1374–1381, Aug. 2011.
[21] X. Xiao, G. Xie, R. Li, and K. Li, "Minimizing schedule length of energy consumption constrained parallel applications on heterogeneous distributed systems," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Aug. 2016, pp. 1471–1476.
[22] Q. Huang, S. Su, J. Li, P. Xu, K. Shuang, and X. Huang, "Enhanced energy-efficient scheduling for parallel applications in cloud," in *Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGrid)*, May 2012, pp. 781–786.
[23] G. Xie, J. Jiang, Y. Liu, R. Li, and K. Li, "Minimizing energy consumption of real-time parallel applications using downward and upward approaches on heterogeneous systems," *IEEE Trans. Ind. Informat.*, vol. 13, no. 3, pp. 1068–1078, Mar. 2017.
[24] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment," *J. Grid Comput.*, vol. 14, no. 1, pp. 55–74, 2016.
[25] G. Xie, G. Zeng, R. Li, and K. Li, "Energy-aware processor merging algorithms for deadline constrained parallel applications in heterogeneous cloud computing," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 2, pp. 62–75, Apr./Jun. 2017.
[26] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002.
[27] J. D. Ullman, "*NP*-complete scheduling problems," *J. Comput. Syst. Sci.*, vol. 10, no. 3, pp. 384–393, Jun. 1975.
[28] G. Koslovski, W.-L. Yeow, C. Westphal, T. T. Huu, J. Montagnat, and P. Vicat-Blanc, "Reliability support in virtual infrastructures," in *Proc. IEEE 2nd Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2010, pp. 49–58.

YUEKUN CHEN is currently pursuing the Ph.D. degree in computer science with Hunan University. Her research interests include cost design optimization, cost-aware computing, and energy-efficient computing.



GUOQI XIE (M'15) received the Ph.D. degree in computer science and engineering from Hunan University, China, in 2014. From 2014 to 2015, he was a Post-Doctoral Researcher with Nagoya University, Japan. From 2015 to 2017, he was a Post-Doctoral Researcher with Hunan University, where he is currently an Associate Professor of computer science and engineering. His current research interests include embedded and cyber-physical systems, parallel and distributed systems, and design automation of electronic systems. He is a member of ACM and CCF. He has received the Best Paper Award from ISPA 2016.

RENFA LI (M'05–SM'10) is currently a Professor of computer science and electronic engineering with Hunan University, China. He is the Director of the Key Laboratory for Embedded and Network Computing of Hunan Province, China. His major research interests include computer architectures, embedded computing systems, cyber-physical systems, and Internet of Things. He is a member of the council of CCF and a Senior Member of ACM. He is also an Expert Committee Member of National Supercomputing Center, Changsha, China.