Cost and Energy Aware Scheduling Algorithm for Scientific Workflows with Deadline Constraint in Clouds

Article *in* IEEE Transactions on Services Computing · January 2015

DOI: 10.1109/TSC.2015.2466545

6 authors, including:

Jidong Ge
Nanjing University
50 PUBLICATIONS **107** CITATIONS
SEE PROFILE

Haiyang Hu
Hangzhou Dianzi University
55 PUBLICATIONS **71** CITATIONS
SEE PROFILE

Wei Song
Nanjing University of Science and Techno…
44 PUBLICATIONS **146** CITATIONS
SEE PROFILE

Bin Luo
Nanjing University
41 PUBLICATIONS **86** CITATIONS
SEE PROFILE

Cost and Energy Aware Scheduling Algorithm for Scientific Workflows with Deadline Constraint in Clouds

Zhongjin Li, Jidong Ge, Haiyang Hu, Wei Song, Hao Hu, and Bin Luo

Abstract—Cloud computing is a suitable platform to execute the deadline-constrained scientific workflows which are typical big data applications and often require many hours to finish. Moreover, the problem of energy consumption has become one of the major concerns in clouds. In this paper, we present a cost and energy aware scheduling (CEAS) algorithm for cloud scheduler to minimize the execution cost of workflow and reduce the energy consumption while meeting the deadline constraint. The CEAS algorithm consists of five sub-algorithms. First, we use the VM selection algorithm which applies the concept of cost utility to map tasks to their optimal virtual machine (VM) types by the sub-makespan constraint. Then, two tasks merging methods are employed to reduce execution cost and energy consumption of workflow. Further, In order to reuse the idle VM instances which have been leased, the VM reuse policy is also proposed. Finally, the scheme of slack time reclamation is utilized to save energy of leased VM instances. According to the time complexity analysis, we conclude that the time complexity of each sub-algorithm is polynomial. The CEAS algorithm is evaluated using Cloudsim and four real-world scientific workflow applications, which demonstrates that it outperforms the related well-known approaches.

Index Terms—Workflow scheduling, cloud computing, energy aware, cost optimization, deadline constraint

1 Introduction

WORKFLOW is defined as a collection of tasks that are processed in a specific order to accomplish a real application [1], and scientific workflow has been proved an efficient and popular method to model various scientific computing problems in parallel and distributed systems. However, with the growing complexity of scientific computing systems, scientific workflows are typical big data applications which are becoming increasingly data-intensive, communication-intensive and computation-intensive, and often require many hours to execute [2]. Moreover, traditional on-premises systems such as clusters and grids are not only very expensive, but also inconvenient to expand resources. Therefore, it is imperative to perform workflows in cloud computing environments as their flexible utility-oriented pay-as-you-go pricing model [3].

Cloud computing is a large-scale distributed computing driven by economies, in which the resources and services are delivered on demand to external customers via the Internet [4]. Consumers and operators are generating a large amount of data on various services per minute in cloud environment, which increasingly comes to show all typical properties of big data [5]. The technological advantages of clouds, such as elasticity, scalability, accessibility and reliability, have made them an attractive alternative to replace private in-house IT infrastructures [6]. It has emerged as a promising computing paradigm for government, research institute and industry to solve ever-increasing computing and storage problems. Nowadays, there are a number of successful commercial cloud providers such as Amazon EC2 [7], Google App Engine [8], GoGrid [9], IBM "Blue Cloud" [10], Microsoft Azure Services Platform [11] and so on.

There are an infinite amount of resources can be accessed in the context of cloud computing environments. However, commercial cloud providers typically charge the users by an hourly-based pricing model. Therefore, the cost is not calculated based on the real amount of resources used, but according to the time unit model, meaning that the users have to pay for the whole leased hour even if they lease the instances for a second [12]. As far as we know, most of the commercial clouds offer various types of VM at different prices. Hence, it is difficult for cloud scheduler to devise an optimal scheme to execute scientific workflow applications within a specified time constraint.

The energy problem has become one of the major concerns in clouds. It is reported that about 50% management budget of Amazon's data center is used for powering and cooling the physical servers [13] and the energy consumption in cloud datacenters is now account for as much as 0.5 percent of the world's total power usage [14]. Besides

- *Z. Li and J. Ge are with both State Key Laboratory for Novel Software Technology, Software Institute, Nanjing University, China, and State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications), Beijing China. J. Ge is the corresponding author. E-mail: {lzjnju@126.com, gjd@nju.edu.cn}.*
- *H. Hu is with both School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China, and State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications), Beijing China. E-mail: huhaiyang@hdu.edu.cn.*
- *W. Song is with School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. E-mail: wsong@njust.edu.cn.*
- *H. Hu and B. Luo are with State Key Laboratory for Novel Software Technology, Software Institute, Nanjing University, China. E-mail: {myou@nju.edu.cn, luobin@nju.edu.cn}.*

xxxx-xxxx/0x$xx.00 © 200x IEEE Published by the IEEE Computer Society

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2466545, IEEE Transactions on Services Computing

the energy cost, datacenters also produce considerable amount of CO_2 emissions which significantly contribute to the growing environmental issue of global warming [15]. As a result, energy problem is receiving increasing attention due to the environmental and financial factors. In a word, it is necessary to develop an energy efficient workflow scheduling method to reduce energy consumption as much as possible.

Then, in this paper, we present a cost and energy aware scheduling (CEAS) algorithm for cloud schedulers to minimize the execution cost of scientific workflow and reduce the energy consumption. The proposed CEAS algorithm covers five sub-algorithms. First, the VM selection algorithm is used to apply the cost utility concept to map tasks to their optimal virtual machine (VM) types by the sub-makespan constraint. Then, two tasks merging methods are employed to reduce execution cost and energy consumption of workflow. In order to reuse the idle VM instances which have been leased, the VM reuse policy is also proposed to reuse these idle VMs. Finally, the task slacking algorithm is utilized to save the energy of leased VMs by DVFS technique. We can make a conclusion that time complexity of each sub-algorithm is polynomial from the analysis results. Simulation experiments with four well-known scientific workflows show that the proposed algorithm can reduce total execution cost and energy consumption efficiently. Our main contributions are summarized as below:

- We present a cost and energy aware scheduling algorithm, called CEAS, for cloud scheduler first to minimize the workflow cost for users, and on the basis of it, to reduce the energy consumption.
- We propose a concept of cost utility to map each task to an optimal instance type based on the hourly-based pricing model, which can save cost substantially.
- We introduce two tasks merging methods which can meet the performance requirements of workflow and reduce the cost and energy consumption.
- We propose a VM reuse sub-algorithm to reuse the idle VM instances which have been leased to further reduce execution cost and energy consumption of workflow.

The rest of this paper is organized as follows. Section 2 is devoted to the related work. We describe our sytem models and architecture in Sections 3. Section 4 introduces five sub-algorithms as well as the CEAS algorithm and analyses the time complexity of each sub-algorithm. The performance evaluation approaches, simulation details and results are conducted in Section 5. In Section 6, we conclude this paper.

2 Related Work

The term big data refers to large and complex data sets made up of a variety of structured and unstructured data which are too big, too fast, or too hard to be managed by traditional techniques. Many works have researched how to process, manage and apply the big data. In [16], authors aim to identify issues and challenges faced by MapReduce when confronted by the big data. Tan et al. [17] explore the personal ad hoc clouds comprised of individuals in social networks. In [18], authors propose an algorithm that allows users to express their interest in terms of constraints and use the MapReduce model to mine uncertain big data for frequent patterns.

A lot of work has been done in the past on multi-dimensional optimization problems in the context of databases [19], [20], [21], service selection [22], [23], [24], [25], and workflow scheduling [26], [27], [28], [29], [30]. All the multi-objective methods of workflow scheduling can be classified as bounded objective optimizations that consider one constrained objective and optimize another objective with respect to the defined constraints.

Lin and Wu [31] construct a set of analytical models to quantify the network performance of scientific workflows, and formulate a task scheduling problem to minimize the end-to-end delay of workflow under a budget constraint. They assume that a group of modules in the original workflow have been bundled together as one aggregate module in the resulted task graph, but it is difficult to gather the tasks into groups. Zhu and Agrawal [32] focus on using cloud resources for a class of adaptive applications. However, their work is based on the assumption that fine-grained allocation and pricing of resources is possible for the virtual environment. This is not true for the existing commercial cloud environments. An adaptive scheduling heuristic for parallel execution of scientific workflows is proposed in the cloud that is based on three criteria: total execution time (makespan), reliability and financial cost [33]. Although, both the cloud activities execution cost and data transfer cost have been considered, they do not calculate them based on the hourly-based pricing model. Moreover, all these methods mentioned above meet the QoS requirements for users which different from the perspective of this paper.

Other multi-objective scheduling algorithm like literature [12] proposes many basic rescheduling operations covering a broad set of scenarios for reducing the costs of running scientific workflows in clouds. In [34], a multi-objective list scheduling approach for workflow applications is presented. Based on a set of objectives constraints and weights defined by user, the algorithm attempts to find an appropriate Pareto solution: maximize the distance to a user-defined constraint vector for dominant solutions and minimizing it otherwise. Durillo et al. [13] intrduce a new Pareto-based list scheduling heuristic that provides the user with a set of tradeoff optimal solutions from where the one that better suits the user requirements can be manually selected. These scheduling approaches need to reserve a set of VM instances for the whole makespan, which will incur a lot of unnecessary cost when the leased VM instances are in idle. Moreover, as cloud datacenters offer various types of VM, it is difficult to know which types and how many VM instances should be reserved.

Numerous algorithms that aim to find multi-objective workflow scheduling scenarios with evolutionary methods have been studied. A meta-heuristic optimization technique, Particle Swarm Optimization (PSO), is devel-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2466545, IEEE Transactions on Services Computing

oped to minimize the overall workflow execution cost while meeting the deadline constraint in clouds [35]. It is devised to meet either the user's QoS requirements or incorporate some basic principles of cloud computing. Chen et al. [36] elaborate an approach based on Ant Colony Optimization (ACO) to meet various QoS requirements in a grid workflow, and Yu et al. [37] use Genetic Algorithms (GA) to implement a budget constrained scheduling of workflows on utility Grids. The main disadvantage of evolutionary algorithms is their long computational time due to their slow convergence rate.

Many efficient workflow scheduling techniques for the purpose of reducing the energy consumption have been researched [38], [39], [40]. Kimura et al. [46] present a slacking algorithm which extends the task execution time by reclaiming slack room to save energy. Then, an enhanced Energy-Efficient Scheduling (EES) algorithm is proposed to reduce the energy consumption while meeting the performance-based requirements [41]. Nevertheless, the two energy efficient algorithms mentioned above only use the slack time reclamation technique to reduce energy. Cao and Zhu [15] employ DVFS technique to lower the CPU frequencies of virtual machines as long as the finishing time is still before the specified deadline. But they ignore the fact that cloud datacenter charges the VM instances by hourly-based pricing model. Two energy-aware task scheduling algorithms [42], called the Enhancing Heterogeneous Earliest Finish Time (EHEFT) and the Enhancing Critical Path on a Processor (ECPOP), are proposed to address the time and energy efficient workflow scheduling. Then, Tang et al. [43] introduce a DVFS-enabled Efficient energy Workflow Task Scheduling (DEWTS) algorithm to obtain more energy reduction. However, the last two algorithms are belonging to the list scheduling method which will also have to reserve a set of VM instances for the whole makespan.

3 System Models and Architecture

Our problem consists in scheduling the big data workflow meeting the specified deadline in such a way that the monetary cost and energy consumption are minimized.In this section, we describe the workflow model, cloud datacenter model, energy model and cloud system architecture underneath our approach. For ease of understanding, we summarize the major notations and their meanings used throughout of this paper in TABLE 1.

3.1 Workflow Model

Generally, a workflow application model with precedence-constrained tasks is represented by a directed acyclic graph $DAG([t],[e])$, where $[t]=\{t_1,t_2,...,t_n\}$ is the set of n tasks; $[e]$ is the set of directed arcs or edges between the tasks to represent dependency. For example, edge $e(i,j)$ represents the precedence constraint that task t_i should complete its execution before task t_j starts. Case in which task t_i is called one of the direct predecessors of task t_j, and task t_j is called one of the immediate successors of task t_i. The set of predecessor tasks of t_i is denoted as $pre(t_i)$, and the set of successor tasks is $succ(t_i)$. A task is triggered to execute when all of its inputs are available. When the task t_i is finished, it generates the output O_i which is the typical big dataset in scientific workflow. A task with no predecessor tasks in *DAG* is called an entry task and a task with no successor tasks is called an exit task. We assume that *DAG* has exactly one entry task t_{entry} and one exit task t_{exit}. We denote the makespan as T_M which represents the entire execution time of the workflow. In addition, each workflow has a deadline T_D associated with it. A deadline is defined as a time constraint for the workflow execution.

TABLE 1
Major Notations

Symbol	Definition
t_i	The ith task of the workflow, $i=1,2,...,n$;
I_i	Total amount of input data of task t_i;
O_i	The amount of output data of task t_i;
w_i	Workload of task t_i;
$pre(t_i)$	Predecessor set of task t_i;
$succ(t_i)$	Successor set of task t_i;
t_{entry}	Entry task of the workflow;
t_{exit}	Exit task of the workflow;
vm_k	The VM of Type k;
p_k^{max}	Maximum computing capacity of vm_k;
p_k^{min}	Minimum computing capacity of vm_k;
p_k^j	Computing capacity of vm_k in jth level;
c_k	Cost per hour of vm_k;
v_k^j	Voltage in jth level;
f_k^j	CPU frequency in v_k^j;
P_k^j	Power of vm_k in jth voltage level;
E_k^j	Energy consumption in runtime;
E_{idle}	Energy consumption in idle time;
T_M	Makespan of the workflow;
T_D	Deadline of the workflow;
$T(t_i, vm_k)$	Execution time of task t_i on vm_k;
$T_{sub}(t_i)$	Sub-makespan of task t_i;
$u(t_i, vm_k)$	Cost utility of task t_i on VM of type k.

3.2 Cloud Datacenter Model

We assume that the cloud datacenter offers a set of K types of VM $VM=\{vm_1,vm_2,...,vm_k,...,vm_K\}$ with an hourly-based pricing model, without loss of the generality. For example, Amazon EC2 provides four types of VM instances [7]: small (S), medium (M), large (L) and extra-large (XL). A VM type vm_k is specified by several characteristics, such as maximum computing performance p_k^{max} in million instructions per second, cost per hour c_k and bandwidth B. Suppose that all the VMs in the cloud datacenter are with the same communication bandwidth. Moreover, all types of VM have been ascending sorted according to their maximum computing performances, that is $p_1^{max}<p_2^{max}<\cdots<p_k^{max}<\cdots p_K^{max}$, and then the prices are in accordance with the computing performances, that is $c_1<c_2<\cdots<c_k<\cdots c_K$. We also assume that the price and performance between VM types subject to the equation (1)

$$\frac{c_k}{p_k^{max}}<\frac{c_{k+1}}{p_{k+1}^{max}} \quad (1)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2466545, IEEE Transactions on Services Computing

Which means that the higher computing performance, the higher unit performance price. Thus, cloud scheduler has different choices to map tasks to the different VM types with the different workflow's deadline constraints. Therefore, the longer execution time of a task, the less cost will be. The DVFS technique operates CPU frequencies under multiple voltage levels. Generally, we consider the computing performance p_k^j, associating with the CPU frequency, is in the range of $[p_k^{min}, p_k^{max}]$, where j is the voltage level of CPU.

3.3 Energy Model

The power consumption for an application execution is composed of dynamic and static energy consumption: $E_{dynamic}$ and E_{static}. As the dynamic energy consumption is the most significant factor, the static energy consumption is ignored in this paper [43]. The dynamic power dissipation p_k^j of VM type vm_k in the voltage level j is described in formula (2).

$$P_k^j = \lambda_k \cdot (v_k^j)^2 \cdot f_k^j \quad (2)$$

Where the constant parameter λ_k is related to dynamic power depending on VM type and capacity; v_k^j is the supply voltage at level j on the VM of type k and parameter f_k^j is the frequency with matching v_k^j. We know that the parameter f_k^j and v_k^j are in proportion to the computing capacity p_k^j, and hence they are in the range $[f_k^{min}, f_k^{max}]$ and $[v_k^{min}, v_k^{max}]$ respectively. Then the energy consumption in runtime $t_{runtime}$ is computed by equation (3).

$$E_k^j = P_k^j \cdot t_{runtime} = \lambda_k \cdot (v_k^j)^2 \cdot f_k^j \cdot t_{runtime} \quad (3)$$

Energy is defined as power multiplied by time. In this situation, execution time is inversely proportional to the frequency. As a result, the energy required is proportional to voltage squared. Thus, decreasing the voltage can effectively reduce energy consumption. For example, if the voltage decreases to 70% then there is a 50% reduction in energy required. As the supply voltage and frequency cannot scale to zero during the processor's idle period, they will stay at their lowest voltage state for maximum energy saving. Hence, the energy consumption of idle time is defined by

$$E_{ilde} = \lambda_k \cdot (v_k^{min})^2 \cdot f_k^{min} \cdot t_{idletime} \quad (4)$$

The total energy consumption of a VM instance is represented by the equation (5).

$$E_{total} = E_k^j + E_{idle} \quad (5)$$

3.4 Cloud System Architecture

The aim of this paper is to minimize the execution cost of workflow for users, and then reduce the energy consumption for cloud datacenters. Note that the existing works [15], [44], [45] propose such an architecture that focuses on energy-efficient cloud computing. In these architectures, cloud scheduler can directly interact with users and infrastructures to perform workflow scheduling strategy. Inspired by these works, we also propose a cloud system architecture for cost and energy aware workflow scheduling which is shown in Fig. 1. There are basically four main entities involved:

1. **Cloud Users**: Submit workflow requests from anywhere in the world to the cloud.

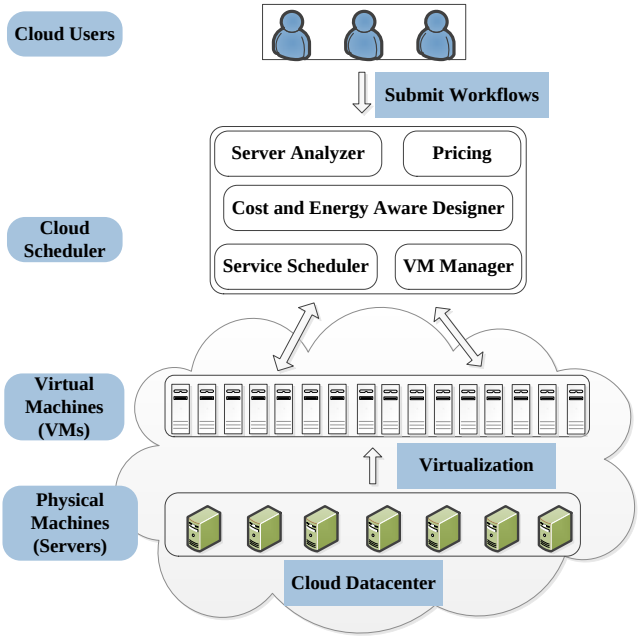


Fig. 1. Cloud System Architecture

2. **Cloud Scheduler**: Acts as the interface between the cloud infrastructure and users. It requires the interaction of the following components to support the cost and energy efficient workflow scheduling:
   a) *Service Analyzer*: Analyzes the service requirements of the workflow application.
   b) *Cost and Energy Aware Designer*: Devises a workflow scheduling strategy to minimize the execution cost of workflow while meeting the deadline constraint, and reduce the energy consumption.
   c) *Pricing*: Calculates the task execution cost based on the hourly-based pricing model.
   d) *Service Scheduler*: Assigns requests to VMs by the scheduling strategy, and it also decides when VMs are to be used or removed.
   e) *VM Manager*: Updates VMs status and provisions new VMs as well as reallocating VMs across physical machines.
3. **Virtual Machines (VMs)**: Multiple independent VMs can be created and deployed on a single physical server according to incoming requests.
4. **Physical Machines (Servers)**: The physical servers provide the hardware infrastructure for creating virtualized resources to meet service demands.

In the competitive cloud computing environment, cloud scheduler acts as the interface between the cloud infrastructure and users. It provides the minimum cost of workflows to attract more customers and users. Moreover, cloud scheduler also needs to earn money according to their charging model, and reduce energy consumption as much as possible. As for users, they will choose the cloud provider who has the minimum workflow execution cost in this commercial multicloud environment. However, after the completion of the workflow, they will be aware of the execution time of each task and the leased VM types. Then, according to the public pricing model of cloud providers, users can find if the cloud scheduler tells

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2466545, IEEE Transactions on Services Computing

the truth.

4 Algorithm Implementation

The proposed CEAS algorithm is capable of reducing cost and energy consumption while meeting the deadline. It consists of five sub-algorithms, such as VM selection algorithm, sequence tasks merging algorithm, parallel tasks merging algorithm, VM reuse algorithm and task slacking algorithm, to optimize the cost and energy consumption step by step. In this section, we present implementations of these sub-algorithms in detail as follows.

4.1 VM Selection Algorithm

Unlike the utility grids and clusters computing environment [38], [46], as long as there are services available, the scheduling mechanism can reserve a time slot and place the task on the service. In clouds, although there are unlimited resources, and cloud datacenter can offer various types of VM instance for users, it may not be always good to do so whenever a task needs to be processed, especially when the task could waste a significant portion of a purchased instance-hour. What is more, scientific workflows are typical big data applications, which often require high performance computing to run many hours. Therefore, in this section we introduce the VM selection method to map each task to an optimal VM type to minimize the monetary cost.

Before we introduce our VM selection algorithm, we first present a concept of sub-makespan that stands for the execution time of a task, which is similar to the makespan of the workflow. It is obvious that if the execution time of each task is no more than assigned sub-makespan, then the whole workflow will be finished within its deadline. First of all, we map each task to the VM of type K which has the maximum computing performance. Then, the fastest execution time of task t_i is calculated by equation (6).

$$T(t_i, vm_K) = w_i / p_K^{max} + I_i / B \tag{6}$$

Where $T(t_i, vm_K)$ is the execution time of task t_i on VM of type K which consists of the data transfer time and the effective execution time. We also denote the earliest start time and earliest finish time of task t_i as $EST(t_i)$ and $EFT(t_i)$ as follows.

$$EFT(t_i) = EST(t_i) + T(t_i, vm_K) \tag{7}$$

$$EST(t_i) = \begin{cases} 0, & if\ t_i = t_{entry} \\ \max\limits_{t_l \in pre(t_i)} EFT(t_l), & otherwise \end{cases} \tag{8}$$

The minimum makespan is the earliest finish time of task t_{exit}, that is $minT_M = EFT(t_{exit})$. Without loss of generality, we assume that the specified deadline T_D will no less than the minimum makespan, that is $T_D \geq minT_M$. We define the sub-makespan of task t_i as $T_{sub}(t_i)$ expanded proportionally according to its fastest execution time.

$$T_{sub}(t_i) = T(t_i, vm_K) \cdot T_D / minT_M \tag{9}$$

Then, we define the cost utility that means the workload can be finished per unit cost. Then, the cost utility of task t_i on VM of type k is computed by

$$u(t_i, vm_k) = \frac{w_i}{\lceil T(t_i, vm_k) \rceil \cdot c_k} \tag{10}$$

The value $\lceil T(t_i, vm_k) \rceil \cdot c_k$ is the total cost $c(t_i, vm_k)$ of the task t_i executing on vm_k since cloud datacenters charge users by hourly-based pricing model. We can see from equation (10) that the more cost utility, the less cost will pay for the task to execute. As each task has a sub-makespan constraint, we also have to select the VM type for each task to finish within its sub-makespan. A utility set of task t_i which contains all the cost utilities is denoted as $U(t_i) = \{u(t_i, vm_{k(1)}), ..., u(t_i, vm_{k(j)}), ..., u(t_i, vm_{k(K)})\}$ by descending sort. Hence, the first cost utility $u(t_i, vm_{k(1)})$ in $U(t_i)$ is the optimal one when task t_i is scheduled to $vm_{k(1)}$, and we mark $vm_{k(1)}$ as vm_k^{opt} for task t_i. Then, we can divide all tasks into two categories.

Category 1: For a task t_i, if $T(t_i, vm_k^{opt}) \leq T_{sub}(t_i)$, that means task t_i can be finished to meet its sub-deadline under the optimal cost utility. Hence, we map task t_i to the vm_k^{opt}, which is denoted as $map: t_i \rightarrow vm_k^{opt}$, and update the sub-makespan of task t_i as $T_{sub}(t_i) = T(t_i, vm_k^{opt})$. Thus, we can release some spare time for other tasks.

Category 2: Otherwise, we add task t_i into task set *TASK* for the further processing.

In terms of the tasks in *Category 2*, we apply the following steps to select an optimal VM type for each task.

Step a: Sort the tasks in set *TASK* according to their optimal cost utility in descending order.

Step b: Select the first task t_i in the set *TASK* which has the maximum cost utility.

Step c: Set $T_{sub}(t_i) = T(t_i, vm_k^{opt})$ and recalculate the makespan T_M of the workflow, and continue processing the sub-steps as below.

Step c-1: If $T_M \leq T_D$, that means enlarging the sub-makespan of task t_i without affecting the deadline, then set $T_{sub}(t_i) = T(t_i, vm_k^{opt})$ and map task t_i to vm_k^{opt} that is $map: t_i \rightarrow vm_k^{opt}$. Remove task t_i from the set *TASK*.

Step c-2: Otherwise, select the sub-optimal cost utility from utility set $U(t_i)$ of task t_i as the optimal one (e.g. if the current VM instance type is $k(j)$, then we set $vm_k^{opt} = vm_{k(j+1)}$). Resort the tasks in set *TASK* according to the new cost utility of task t_i.

Repeating the above steps until the task set *TASK* is empty. We can learn that the sub-makespan update of each task in *Step c* will not affect other tasks' sub-makespan, and each task can find an appropriate VM type to finish it. The time complexity of sorting cost utility for all tasks is $O(nKlogK)$. Actually, the worst time complexity of resorting in *Step c-2* is $O(n)$ as it only need insert operation, and the time complexity of makespan recalculating is $O(n+e)$ where e is the number of edges of workflow *DAG*. As *Step c* may iterate kn times, the worst time complexity of VM selection policy is $O(Kn^2(n+e))$.

4.2 Sequence Tasks Merging Algorithm

A workflow task partition method is developed in [38], which categorizes workflow tasks to be either a synchronization task or a simple task. The simple tasks, each of them has one parent and one child, are partitioned to a branch. In light of this, we extend this partition method as our sequence tasks merging algorithm.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2466545, IEEE Transactions on Services Computing

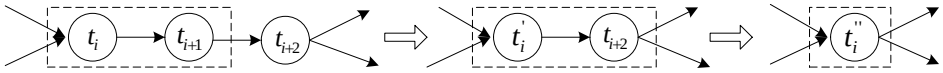


Fig. 2. The process of sequence tasks merging.

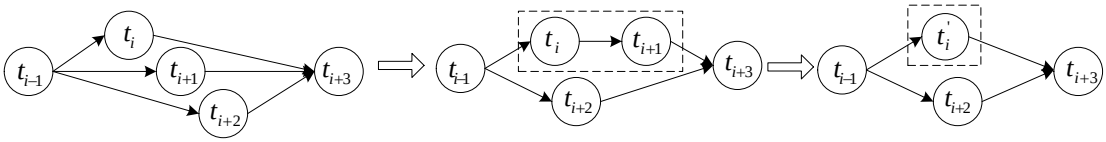


Fig. 3. The process of parallel tasks merging.

We assume two tasks t_i and t_{i+1} are sequence tasks which have been mapped to VM of type k^i and k^{i+1} respectively, and task t_{i+1} is the successor of task t_i. Then, the execution times of two tasks are $T(t_i, vm_{k^i})$ and $T(t_i, vm_{k^{i+1}})$, and the execution costs of two tasks are $c(t_i, vm_{k^i})$ and $c(t_i, vm_{k^{i+1}})$. If we map the two tasks into a VM of type k, the execution time of two tasks is calculated as follows according to formula (6).

$$T(t_{i,i+1}, vm_k) = (w_i + w_{i+1}) / p_k^{max} + I_i / B \quad (11)$$

Then, the execution cost of two tasks is computed by equation (12).

$$c(t_{i,i+1}, vm_k) = \left\lceil T(t_{i,i+1}, vm_k) \right\rceil \cdot c_k \quad (12)$$

From the equation (11), we can see that we do not need the data transfer time I_{i+1}/B any more as the two tasks are on the same VM instance. However, we can always find such a VM of type k which can just finish task t_i and t_{i+1} within the time $(T(t_i, vm_{k^i}) + T(t_i, vm_{k^{i+1}}))$ because we can take $k = max\{k^i, k^{i+1}\}$ at least. Although there may be many other types of VM instance can execute the two tasks while meeting the time constraint, they will result in more costs than the VM of type k on the basis of formula (1). Next, the implementation steps of sequence tasks merging are presented as below.

Step a: Put all tasks into the task set $TASK$.

Step b: For a task t_i in the set $TASK$, if it has only one successor and its successor has only one predecessor, or it has only one predecessor and its predecessor has only one successor. Then, get the VM of type k which can just finish task t_i and t_{i+1} in the time $(T(t_i, vm_{k^i}) + T(t_i, vm_{k^{i+1}}))$. Then, process the following *step b-1* and *step b-2*.

Step b-1: If $c(t_{i,i+1}, vm_k) \le c(t_i, vm_{k^i}) + c(t_{i+1}, vm_{k^{i+1}})$, merge the two tasks t_i and t_{i+1} as a new task t_i', map task t_i' to vm_k and set $T_{sub}(t_i') = T(t_{i,i+1}, vm_k)$. Update the predecessors, successors, and workload of this new task.

Step b-2: Otherwise, we do not merge them together and remove task t_i from task set $TASK$.

Fig. 2 depicts the sequence tasks merging process of three tasks, and only two tasks will be merged every time. The benefits of sequence tasks merging have threefold: 1) the merged tasks are all sequence tasks without affecting the tasks' precedence constraints; 2) reducing the execution cost of tasks; 3) We still merge the tasks even if $c(t_{i,i+1}, vm_k) = c(t_i, vm_{k^i}) + c(t_{i+1}, vm_{k^{i+1}})$ as we do not need to transfer some big dataset which can save a lot of energy.

The time complexity of finding a proper VM type for the merged task is $O(K)$. Then, we make a conclusion that the worst time complexity of sequence tasks merging algorithm is $O(Kn)$ when all tasks are sequence tasks.

4.3 Parallel Tasks Merging Algorithm

Generally, the parallel tasks have the same start execution time but with different finish time. For example in Fig. 3, the first workflow has three parallel tasks t_i, t_{i+1} and t_{i+2}. We assume that all the tasks have been mapped to the same type of VM instance, and the execution time of each task is 20min, 30min and 60min respectively. As these tasks are parallel tasks, task t_{i+3} cannot start until task t_{i+2} finished since task t_{i+2} has the longest execution time. But, if we change the parallel relation between task t_i and t_{i+1} as sequence which is depicted in second workflow of Fig. 3, the total execution time of task t_i and t_{i+1} is less than 50min which is not more than 60min. Therefore, if we merge task t_i and t_{i+1} together, which will not impact the makespan of workflow and only need fewer VMs to execute tasks that can reduce cost and energy consumption.

Workflow may have complex structure that is the reason why workflow scheduling is a challenging problem. Different from the literature [28], we regard two tasks with same predecessors and successors as the parallel tasks instead of having only one predecessor or successor. Suppose that there is a parallel tasks group g_g which contains more than or equal to two tasks, and $t_i, t_{i+1} \in g_g$. Let $maxT_{sub}$ as the maximum sub-makespan among the tasks in group g_g which is denoted by equation (13).

$$maxT_{sub} = \max_{t_i \in g_g} T_{sub}(t_i) \quad (13)$$

If the execution time of two merged tasks does not exceed the maximum sub-makespan $maxT_{sub}$, it will not affect the start execution time of their successors. In a word, the parallel tasks could be merged together.

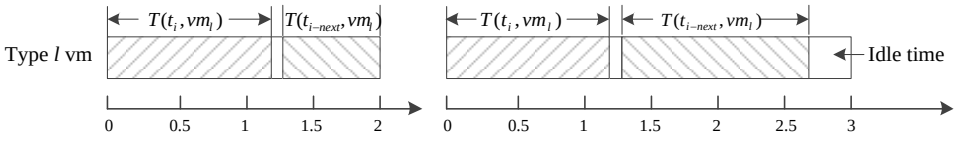
We assume parallel tasks t_i and t_{i+1} have been mapped to VM of type k^i and k^{i+1}, and the execution costs of two tasks are $c(t_i, vm_{k^i})$ and $c(t_i, vm_{k^{i+1}})$. If we map the two parallel tasks to the VM of type k, the execution time $T(t_{i,i+1}, vm_k)$ and the execution cost are the same with equation (11) and (12) respectively. In the same way that we also only need one copy of input data for the two tasks. Different from the sequence tasks merging that always has a VM of type k which can finish the merged task while meeting the time constraint, there may be not such a VM type for parallel tasks merging. Next, we apply the following steps to implement the parallel tasks merging algorithm.

Step a: For the whole workflow, find out all parallel tasks group set $G = \{g_1, g_2, ..., g_g, ..., g_G\}$.

Step b: For each group g_g, sort the tasks according to their minimum execution time by equation (6) in ascending order.

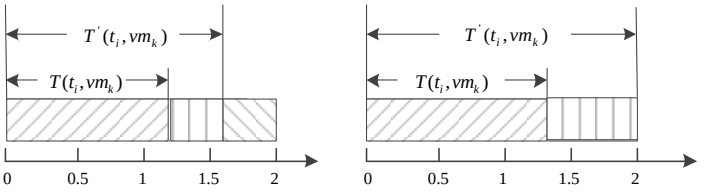
Step c: Take two tasks which have the minimum execution time. Find a VM of type k which can just satisfying the time

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2466545, IEEE Transactions on Services Computing



(a) Condition 1. (b) Condition 2.

Fig. 4. Two cases of VM reuse.



(a) VM is reused. (b) VM is not reused.

Fig. 5. Two cases of task slacking.

constraint $T(t_{i,i+1}, vm_k) \le maxT_{sub}$. If we cannot find such a VM instance type, stop merging and select the next group. Otherwise, continue processing the following sub-steps:

Step c-1: If $c(t_{i,i+1}, vm_k) \le c(t_i, vm_{k^i}) + c(t_{i+1}, vm_{k^{i+1}})$, merge the two tasks as a new task $t_i^{'}$. Map the new task $t_i^{'}$ to vm_k and set $T_{sub}(t_i^{'}) = T(t_{i,i+1}, vm_k)$. Update the predecessors, successors, and workload of this new task. Resort the tasks in group g_g according to the minimum execution time of new task $t_i^{'}$.

Step c-2: Otherwise, we do not merge them together, and select the next group.

Repeating *Step c* until the group set G is empty. The process of parallel tasks merging is depicted in Fig. 3. The benefits of parallel tasks merging have threefold: 1) it will not impact the execution time of successors; 2) it can reduce the execution cost of tasks; 3) it can save a lot of energy for cloud providers as we do not need to transfer some big dataset at least.

We can see that the worst time complexity of finding parallel tasks group set is $O(n^2)$ in *Step a* when there are no parallel tasks in the workflow. The worst time complexity of sorting task is $O(nlogn)$ in *Step b*. Then, the worst time complexity of *Step c* is $O(n^2 logn)$ when all the tasks are parallel tasks. Where $O(nlogn)$ is for tasking two tasks for the group and $O(n)$ is for resorting in *Step c-1*. As a result, the time complexity of parallel tasks merging algorithm is $O(n^2 logn)$.

4.4 VM Reuse Algorithm

If the execution time of task t_i is $T_{sub}(t_i) = 70$min, it still leases the VM instance for 2 hours. Thus, the VM will be in idle for 50min which results in cost waste for users. Nevertheless, we could reuse the idle time of leased VM to execute the current executable tasks. We denote the idle time of VM type vm_l as $T_{ilde}(vm_l)$ when the schedulable task t_{i-next} comes. If an idle VM instance can be reused by the task t_{i-next}, the *Condition 1* or *Condition 2* should be met, which are expressed as below.

Condition 1: The idle VM can totally finish the task t_{i-next} within the assigned sub-makespan, which is represented by equation (14).

$$T(t_{i-next}, vm_l) \le T_{sub}(t_{i-next}) \le T_{idle}(vm_l) \tag{14}$$

Condition 2: We keep on leasing the VM of type l integer hours for the task t_{i-next} that can save the monetary cost without exceeding the sub-makespan, which is described by equation (15).

$$\begin{cases} \left\lceil T(t_{i-next}, vm_l) - T_{idle}(vm_l) \right\rceil \cdot c_l \le \left\lceil T(t_{i-next}, vm_k) \right\rceil \cdot c_k \\ and \;\; T(t_{i-next}, vm_l) \le T_{sub}(t_{i-next}) \end{cases} \tag{15}$$

Equation (14) and (15) indicates that *Condition 1* and *Condition 2* break the original VM selection scheme of task t_{i-next} which has been derived in VM seletion policy. Therefore, we can see that the VM reuse method can take full advantage of leased VM instances to reduce monetary cost and energy consumption. The two cases of VM reuse are depicted in Fig. 4.

Unfortunately, there are more than one idle VM instances and schedulable tasks at the same time. Therefore, we present a simple but useful VM reuse method. We denote the idle VMs set as $IV = \{vm_1^{IV}, ..., vm_l^{IV}, ..., vm_L^{IV}\}$ and current schedulable tasks set as $ST = \{t_1^{ST}, ..., t_i^{ST}, ..., t_n^{ST}\}$. In order to reuse the idle VMs, the following steps will be implemented.

Step a: Select a task t_i^{ST} from the schedulable tasks set ST randomly.

Step b: Calculate the minimum execution cost of task $c(t_i^{ST}, vm_l^{IV})$ if we reuse the idle VM instance vm_l^{IV} according to the two conditions. Obviously, the cost is zero if the task t_i reuses the idle VM instance vm_l^{IV} satisfying the equation (14). Then, there is a cost set of task t_i^{ST}, that is $C(t_i^{ST}) = \{c(t_i^{ST}, vm_1^{IV}), ..., c(t_i^{ST}, vm_l^{IV}), ..., c(t_i^{ST}, vm_L^{IV})\}$. Get the minimum cost $minc(t_i^{ST}, vm_l^{IV})$ from the cost set.

Step b-1: If $minc(t_i^{ST}, vm_l^{IV}) \le \left\lceil T(t_i^{ST}, vm_k) \right\rceil \cdot c_k$, which means that the VM reuse method will cost less than original mapping scheme, and task t_i will reuse the idle VM instance vm_l^{IV} in the form of *Condition 1* or *Condition 2* and map the task t_i^{ST} to vm_l^{IV}. Then, remove the schedulable task t_i^{ST} and idle VM vm_l^{IV} from the set ST and IV respectively and update the sub-makespan of the task t_i^{ST} as $T_{sub}(t_i^{ST}) = T(t_i^{ST}, vm_l^{IV})$.

Step b-2: Otherwise, use the original mapping scheme, $map(t_i^{ST}) = vm_k$, and remove the task t_i^{ST} from the set ST.

Repeating the above steps until the task set ST or VM set IV is empty. We know that the energy consumption is reduced if the *Condition 1* is satisfied. Then we analyze the time complexity. Calculating the cost set is the most complex step in VM reuse algorithm that the time complexity is $O(L)$. Therefore, the time complexity of VM reuse algorithm is $O(nL)$ and the parameter L is the maximum number of idle VMs in the process of workflow scheduling.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2466545, IEEE Transactions on Services Computing

TABLE 2
THE CEAS ALGORITHM SUMMARY

	VM selection	Sequence tasks merging	Parallel tasks merging	VM reuse	Task slacking
Reduce cost	-	Yes	Yes	Yes	-
Reduce energy	-	Yes	Yes	Yes	Yes
Time complexity	$O(Kn^2(n+e))$	$O(Kn)$	$O(n^2 logn)$	$O(nL)$	$O(n^2)$

4.5 Task Slacking Algorithm

Energy consumption has become one of the major concerns to the cloud data centers. Many works have shown that energy consumption could be reduced by reclaiming slack time [41], [43]. The DVFS technique has been extensively applied to strike a tradeoff between performance and power consumption. Since non-critical tasks have some slack time in the context of workflow scheduling, we can reclaim slack time by lowering the voltage and frequency of leased VM instances to reduce energy consumption. Before we introduce our task slacking algorithm, we should calculate the latest start time $LST(t_i)$ of task t_i which is given by

$$LST(t_i)=\begin{cases} T_M - T(t_{entry}, vm_k), & t_i = t_{exit}, \\ \min\limits_{t_l \in succ(t_i)} LST(t_l) - T(t_l, vm_k), & otherwise \end{cases} \quad (16)$$

The slack time of task t_i is calculated by

$$T_{slacktime}(t_i) = LST(t_i) - EST(t_i) \quad (17)$$

We know that a task t_i belongs to the non-critical task when its slack time is greater than zero, that is $T_{slacktime}(t_i) > 0$. We present the non-critical task set as NT. Similar to literature [46], in order to lower the execution frequency of VM instances, we apply the following steps to slack the execution time of non-critical tasks but with more constrained conditions.

Step a: Calculate slack time by formula (16) and (17) for each task, and add all non-critical tasks into the task set NT. Mark all critical tasks as "extended" which means that task has no slack time to reclaim.

Step b: Pick a task t_i whose predecessors are all marked "extended", and calculate the longest path T_{path}. The longest path T_{path} is the sum of the execution time of the tasks on the path from t_i to the "extended" task.

Step c: Calculate the maximum execution time $T'_{sub}(t_i)$. We can see from the VM reuse algorithm that if the VM instance leased by task t_i is reused by task t_{i-next}, the latest finish time of task t_i cannot exceed the start time of task t_{i-next}; and if the VM instance is not reused, the latest finish time should less than the leased time. Hence,

Step c-1: If VM instance is reused, and then the extended sub-makespan of task t_i is calculated by equation (18).

$$T'_{sub}(t_i) = min\{T_{sub}(t_i) \cdot \frac{T_{path} + T_{slacktime}(t_i)}{T_{path}}, EST(t_{i-next})\} \quad (18)$$

An example of that is shown in Fig. 5a.

Step c-2: If the VM instance is not used, then extended sub-makespan of task t_i is calculated by equation (19).

$$T'_{sub}(t_i) = min\{T_{sub}(t_i) \cdot \frac{T_{path} + T_{slacktime}(t_i)}{T_{path}}, \lceil T(t_i, vm_k) \rceil\} \quad (19)$$

An example of that is depicted in Fig. 5b.

Step d: Change the VM instance frequency of task t_i by $f_k^j = f_k^{max} \cdot T_{sub}(t_i)/T'_{sub}(t_i)$, which means we can decrease the CPU frequency in proportion to the ratio of $T_{sub}(t_i)$ and $T'_{sub}(t_i)$. Mark task t_i as "extended" and remove it from non-critical task set NT.

The worst time complexity of calculating the slack time of all tasks is $O(n^2)$. It mainly depends on finding the longest path the worst time complexity of which is $O(n^2)$ by dynamic programming algorithm. Hence, the time complexity of task slacking algorithm is $O(n^2)$.

4.6 The CEAS Algorithm

We present the CEAS algorithm which consists of the five sub-algorithms mentioned above. The CEAS algorithm enable cloud scheduler expends less cost to complete a workflow and also reduce the energy consumption without exceeding the deadline. We have already allocated each task a sub-makepsan and mapped each task t_i to an appropriate VM type. Therefore, every task can be finished within its sub-makespan, and the whole workflow will be completed in time. The CEAS algorithm pseudo-code is depicted in Fig. 6. We schedule a task to the specified VM type to execute when all the predecessors of this task have been completed. When a task has finished, its successors may become the schedulable tasks and then map them to the assigned VM types. Obviously, we think that the entry task always belongs to the schedulable task. Repeating these processes until the workflow has been completed. The functions and time complexity of each sub-algorithm are summarized in TABLE 2. We can see that time complexity of each sub-algorithm is polynomial. In a word, we conclude that two tasks merging algorithm and VM reuse algorithm can reduce the monetary cost. Meanwhile,

The CEAS algorithm

```
BEGIN
1. call the VM selection algorithm;
2. call the sequence tasks merging algorithm;
3. call the parallel tasks merging algorithm;
4. call the VM reuse algorithm;
5. while the workflow is not completed
6.    map the schedulable task t_i to vm_k ;
7.    if the task is non-critical task
8.       call the task slacking algorithm;
9.    end if
10. end while
END
```

Fig. 6. The pseudocode of the CEAS algorithm.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2466545, IEEE Transactions on Services Computing

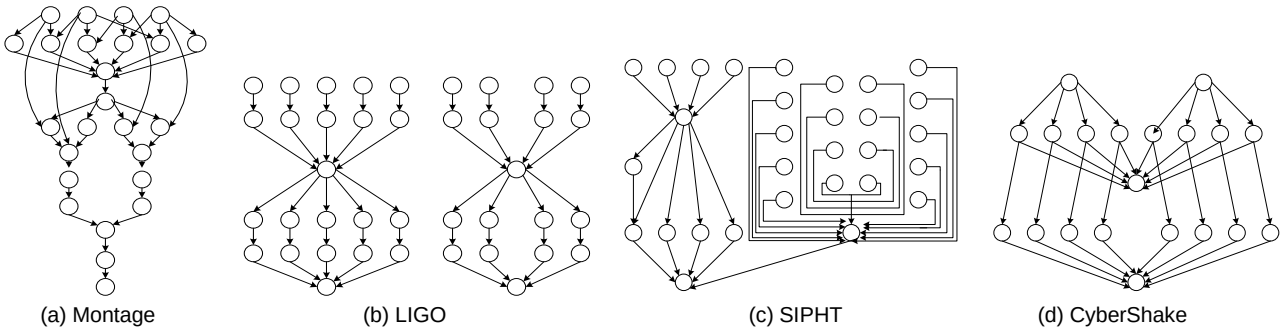


Fig. 7. The structure of four scientific workflows.

two tasks merging algorithm, VM reuse algorithm and task slacking algorithm can reduce the energy consumption.

5 Performance Evaluation

In this section we present the experiments in order to evaluate the performance of the proposed workflow scheduling approach. We use the CloudSim framework [47] to simulate a cloud environment and choose four different workflows from different scientific areas: Montage, LIGO, SIPHT, and CyberShake. These workflows are scientific applications which use large scale datasets. Each of these workflows has different structures as seen in Fig. 7 and different data and computational characteristics have been described in detail in [35], [39]. Suppose that the workload of each task in the workflow is in the range [1000, 5000] MIPS and the output size is in the range [100, 1000] GB only for experiments purpose. Due to the big data workflow applications, the deadline of workflow varies from 5.5 hours to 10 hours as the minimum makespan of them are approximate 5 hours.

We assume that the cloud datacenter has 10 VM types each of which has infinite instances, and bandwidth between VM instances is a constant 1GB. The computing performance, price per hour of each VM type are self-defined in TABLE 3. The relation between computing capacity and price should meet the formula (1). For convenience, we suppose that CPU frequency is direct proportional to computing performance, which is feasible from the practical point of view. Moreover, each VM type has the maximum and minimum CPU frequency used by DVFS technique to lower the execution frequency to reduce energy consumption.

TABLE 3
VM Instance Parameters

VM type	Cost ($/h)	Computing Capacity (MIPS)	CPU frequency level (GHz)	
			maximum	minimum
1	0.10	1000	1.0	0.50
2	0.20	1500	1.5	0.75
3	0.32	2000	2.0	1.00
4	0.46	2500	2.5	1.25
5	0.58	3000	3.0	1.50
6	0.73	3500	3.5	1.75
7	0.90	4000	4.0	2.00
8	1.05	4500	4.5	2.25
9	1.21	5000	5.0	2.50
10	1.40	5500	5.5	2.75

Our experiments focus on the cost and energy varying with the deadline. In order to prove the conformance effect of our sub-algorithms, we first conduct CEAS self-comparison experiments. Thus, the tasks merging algorithm and VM reuse algorithm are compared with VM selection algorithm to reveal the cost saving effectiveness of our CEAS. Note that the tasks merging algorithm consists of sequence tasks merging algorithm and parallel tasks merging algorithm. Then, on the purpose of evaluating the energy reduction behavior, the tasks merging algorithm, VM reuse algorithm and task slacking algorithms are simulated in contrast to the VM selection algorithm. What is more, we also perform four well-known scheduling algorithms, which will be described in detail in the following sub-section, to compare with our CEAS algorithm in terms of cost and energy consumption. First, two reference algorithms, Heterogeneous Earliest Finish Time (HEFT) algorithm [48] and Multi-Objective Heterogeneous Earliest Finish Time (MOHEFT) algorithm [49], are realized in comparison with our CEAS algorithm in the aspect of cost. Furthur, another two algorithms, enhanced Energy-Efficient Scheduling (EES) algorithm [41] and Enhancing HEFT (EHEFT) algorithm [42], are also simulated to demonstrate the energy saving performance of our algorithm.

5.1 Self-comparison Experiments

We conduct self-comparison experiments of cost and energy consumption in this section. In order to show the cost saving effect of our sub-algorithms, we use the VM selection algorithms as a baseline in contrast with our tasks merging algorithm and VM reuse algorithm. It is noted that tasks merging algorithm includes three sub-algorithms, i.e. VM selection algorithm, sequence tasks merging algorithm and parallel tasks merging algorithm. Similarly, the VM reuse algorithm comprises four sub-algorithms.

The simulation results of self-comparison are given in Fig. 8 for cost reduction of four scientific workflows. On the basis of VM selection algorithm, we can see from the figure that tasks merging algorithm indeed saves much monetary cost. This is because we always find a proper VM of type k to execute the merged tasks which can reduce the monetary cost. Moreover, the execution cost

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2466545, IEEE Transactions on Services Computing

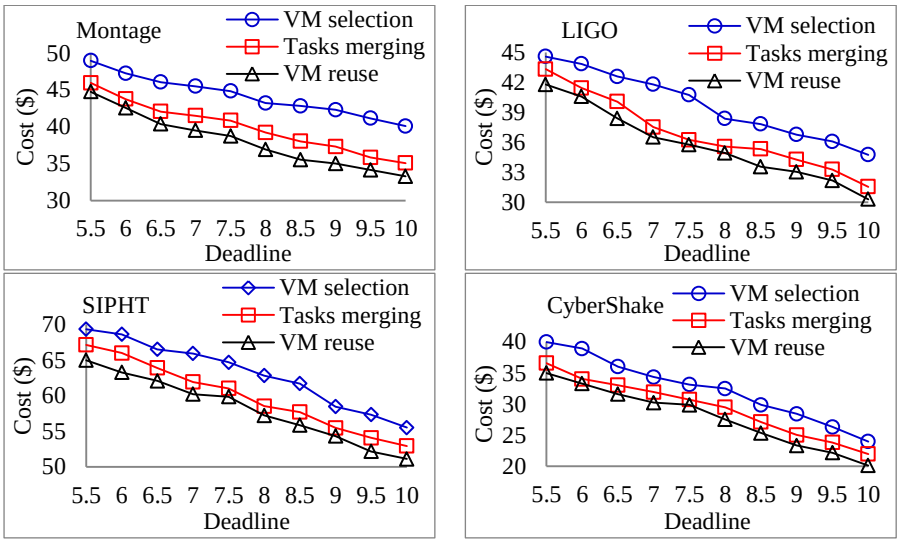


Fig. 8. Self-comparison experiments in cost.

of VM reuse algorithm is less than the tasks merging algorithm. This can be explained by the fact that some idle VM instances reused by other tasks, which break the original VM mapping schemes resulting in less cost. Furthermore, with the deadline of workflow growing, the monetary cost decreases with the same pace. This is because the relation between computing performance and cost satisfies the formula (1), and the assigned sub-makespan of each task will be larger when enlarging the deadline. Therefore, cloud scheduler will choose the VM type with lower computing performance and cost to execute the task when the sub-makespan has been extended. Thus, cloud scheduler could map different tasks to different VM types in terms of different sub-makespan constraints and workloads.

Then, we reveal the energy reduction results of self-comparison experiments. To show the energy saving effectiveness of tasks merging algorithm, VM reuse algorithm and task slacking algorithm, we compare them with our VM selection algorithms. Similarly, it is worthy to note that the task slacking algorithm contains all of the sub-algorithms.

Fig. 9 plots the energy consumption results of four algorithms on four scientific workflows. As we can see from the figures that tasks merging algorithm is superior to the VM selection algorithm, VM reuse algorithm is better than tasks merging algorithm, and task slacking algorithm performs the best. This is due to the fact that we do not need to transfer a lot of big datasets, and less VM instances are demanded when merging the sequence and parallel tasks. In the VM reuse algorithm, many tasks reuse the idle VM instances according to formula (14) which can definitely reduce the energy, and there is a strong likelihood that energy consumption can be decreased due to formula (15). As to task slacking algorithm, it is a common sense that workflows always have many non-critical tasks which have amount of slack time. Therefore, we can reduce energy consumption by DVFS technique without affecting the execution time of workflow. Furthermore, the energy consumption is decreasing when the deadline of workflow is growing. This can be explained by the reason that tasks can be mapped to the

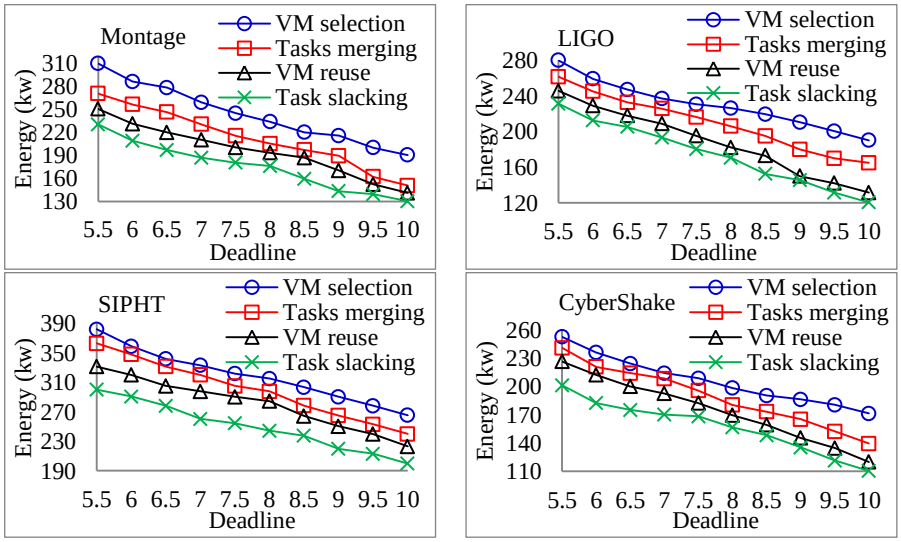


Fig. 9. Self-comparison experiments in energy consumption.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2466545, IEEE Transactions on Services Computing

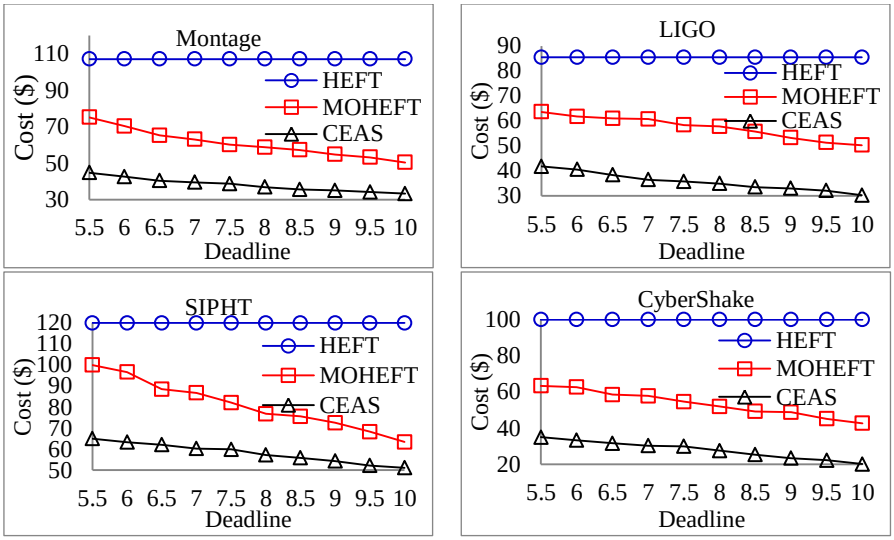


Fig. 10. Comparison experiments in cost.

VM types with lower computing performance that results in less energy consumption.

5.2 Comparison Experiments with Others

For purpose of revealing the strength of our CEAS algorithm on reducing cost and energy consumption, we perform four well-known scheduling algorithms to compare with our CEAS algorithm. First of all, two reference algorithms, namely HEFT algorithm and MOHEFT algorithm, are conducted in comparison with CEAS algorithm in the aspect of cost. The HEFT is a popular list-based heuristic scheduling algorithm for optimizing the makespan in workflow applications. It selects the task with the highest upward rank value at each step and assigns the selected task to the processor, which minimizes the earliest finish time with an insertion-based approach. The MOHEFT algorithm is a Pareto-based multi-objective list workflow scheduling method as an extension to the HEFT mono-objective workflow scheduling algorithm. It is a heuristic-based method that computes a set of tradeoff solutions with a small additional overhead compared with conventional single objective methods. Moreover, MOHEFT can build several workflow schedules in parallel, which has the potential for optimizing makespan and economical costs in a cloud computing system. In our comparison experiments, the MOHEFT algorithm is to minimize the cost in the presence of the specified deadline.

The simulation results of three algorithms on cost for four scientific workflows are given in Fig. 10. We can see that the monetary costs incurred by different algorithms are diverse. Overall, our proposed CEAS algorithm has the lowest cost overhead. The HEFT algorithm performs the worst. And the MOHEFT algorithm has less cost than HEFT algorithm. The flat curves in Fig. 10 demonstrate that the costs of HEFT algorithm are independent of the workflow's deadline. This is due to the fact that cloud datacenter has many types of VM and each type could provide infinite VM instances to the public. What is more, the HEFT assigns the task to the VM instance which can minimize the earliest finish time. Consequently, it always maps each task to the VM with the maximum computing performance without considering the cost utility, which results in the highest cost and hence will be not affected by workflow's deadline.

Considerable cost reduction has been made by MOHEFT algorithm in Fig. 10. Different from HEFT algorithm, due to the deadline, MOHEFT selects a proper VM type for each task to execute. Hence, the cost incurred by MOHEFT is less than HEFT. And the cost of MOHEFT algorithm is changing with the deadline, which means that the larger the deadline, the less the cost. This is because cloud scheduler will choose the VM type with lower computing performance and lower cost to execute a task when the deadline is enlarged. However, the MOHEFT algorithm assumes that VM instances are charged by the model of price for second. That is unreasonable in the context of cloud environment. Moreover, it leases VM instances for the whole makespan, which definitely expends more extra monetary cost. Our proposed CEAS algorithm leases the instances by the hourly-based pricing model, which conforms to the price model of cloud computing environment. Besides, it employs three policies to reduce the cost, such as sequence tasks merging, parallel tasks merging and VM reuse. Hence, we can see from the Figure that cost reduction effectiveness of CEAS algorithm is far more than the other two algorithms.

Now we evaluate the effectiveness of CEAS by comparing its performance with two well-known scheduling algorithms, called EES and EHEFT, in the energy saving potential. The two algorithms are briefly described as follows. The EES method is an HEFT-based enhanced energy-efficient scheduling algorithm to reduce energy consumption while meeting the performance-based Service Level Agreement (SLA). The main idea of EES is that as slacking non-critical jobs can achieve significant power saving, it exploits the slack time and allocates them in a global manner to save energy. The EHEFT algorithm is also a HEFT-based efficient energy aware task scheduling method, which has the objective of trying to sustain the makespan and energy consumption at the same time. In

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2466545, IEEE Transactions on Services Computing

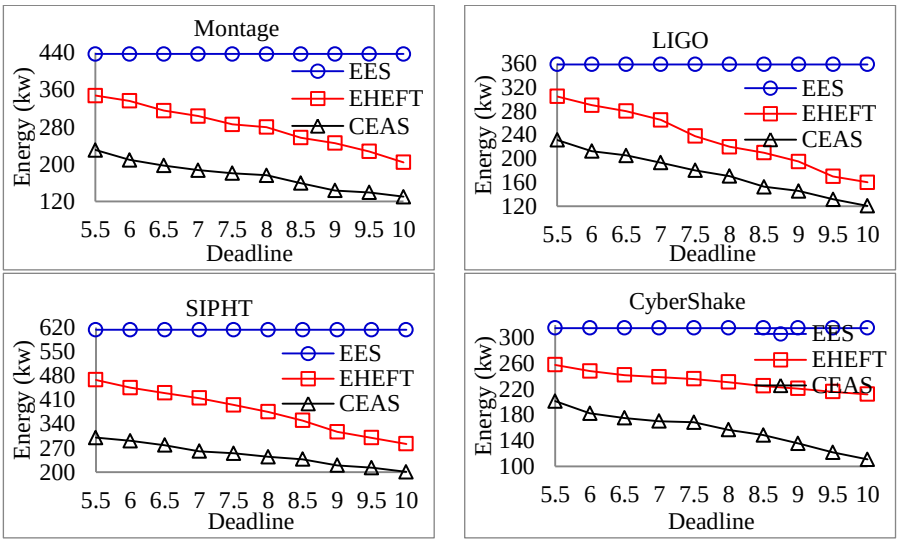


Fig. 11. Comparison experiments in energy consumption.

comparison with the EES algorithm, the EHEFT algorithm not only utilizes the slack time of non-critical task to save energy, but also uses a metric that identifies the inefficient processors and shuts them down to reduce energy consumption.

Fig. 11 shows the simulation results of energy consumption of the three algorithms on four scientific workflows. Among these algorithms, the CEAS algorithm has the least energy consumption, the EHEFT policy has the medium energy consumption, and the EES algorithm consumes the most energy. Similar to the cost performance of HEFT algorithm shown in Fig. 10, the energy consumption of EES algorithm does not change with the deadline. This is because the EES algorithm is based on the HEFT algorithm and always selects the best computing performance VM to execute the task, which does not concern about the cost and energy. As shown in Fig. 11 that the EHEFT algorithm consumes less energy than EES algorithm. This result can be explained by the fact that EHEFT algorithm uses a performance metrics called ratio of effectiveness to discover the inefficient processors and shut them down to reduce energy consumption. Different from our CEAS algorithm, the EES and EHEFT will reserve the VM instances for the whole makespan. Although they can lower the CPU frequency when processors are in idle, this will still consume massive energy. The CEAS algorithm leases the VM instances by hourly-based model, which has less idle time in the course of workflow executing. Furthermore, many energy reduction strategies, such as tasks merging, VM reuse and task slacking, are applied to optimize the energy. Therefore, the CEAS algorithm is superior to the other two algorithms. From a practical point of view, our approach is more suitable to use in the cloud environment.

6 Conclusions and Future Work

In this paper, we present a cost and energy aware scheduling algorithm for cloud scheduler to minimize the execution cost of workflow and reduce the energy consumption while meeting the deadline constraint. The proposed CEAS algorithm consists of five sub-algorithms. First, the VM selection algorithm is used to apply the cost utility concept to map tasks to their optimal virtual machine (VM) types. Then, two tasks merging methods are employed to reduce execution cost and energy consumption of workflow. In order to reuse the idle VM instances, the VM reuse policy is also proposed to reuse the idle VMs. Finally, the task slacking algorithm is utilized to save the energy by DVFS technique. In a word, sequence tasks merging, parallel tasks merging and VM reuse algorithms can reduce the monetary cost of workflow efficiently. Moreover, sequence tasks merging, the parallel tasks merging, VM reuse and task slacking policies can save considerable energy. We can make a conclusion that time complexity of each sub-algorithm is polynomial that is very suitable for the commercial multicloud environment. Experiment results demonstrate that our CEAS algorithm outperforms existing well-known approaches in both cost and energy consumption.

As a future work, we are going to consider the actual electricity cost for workflow scheduling, and we will incorporate the energy consumption of memory and hard drive into our energy model.

Acknowledgment

This work was supported by the 973 Program (2015CB352202, 2013CB329102), the National Natural Science Foundation, China (No.61321491, 91318301, 61272188, 61202003), National High-Tech Research & Development Program (863 program 2013AA01A213), the Natural Science Foundation of Jiangsu Province (No.BK20131277), the Open Foundation of State key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications)

(SKLNST-2013-1-14).

REFERENCES

[1] D. Kyriazis, K. Tserpes, A. Menychtas, A. Litke, and T. Varvarigou, "An Innovative Workflow Mapping Mechanism for Grids in the Frame of Quality of Service," *Future Generation Computer Systems*, vol. 24, no. 6, pp. 498-511, 2008.

[2] A. Kashlev and S.Y. Lu, "A System Architecture for Running Big Data Workflows in the Cloud," *2014 IEEE International Conference on Services Computing (SCC)*, pp. 51-58, 2014.

[3] Y. Zhao, X.B. Fei, I. Raicu, and S.Y. Lu, "Opportunities and Challenges for Running Scientific Workflows on the Cloud," *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp.455-462, 2011.

[4] I. Foster, Y. Zhao, I. Raicu, and S.Y Lu, "Cloud Computing and Grid Computing 360-Degree Compared," *IEEE Grid Computing Environments (GCE08) 2008 and IEEE/ACM Supercomputing 2008*.

[5] Z.B. Zheng, J.M. Zhu, and M.R. Lyu, "Service-Generated Big Data and Big Data-as-a-Service: An Overview," *2013 IEEE International Congress on Big Data*, pp. 403-410, 2013.

[6] G. Juve and E. Deelman, "Scientific Workflows in the Cloud, Computer Communications and Networks," *Springer*, pp. 71-91, 2010.

[7] Amazon EC2, http://aws.amazon.com/ec2/.

[8] Google App Engine, http://code.google.com/appengine/.

[9] GoGrid, http://www.gogrid.com/.

[10] Blue Cloud, http://www-03.ibm.com.

[11] Azure, http://www.microsoft.com/azure/default.mspx.

[12] H.M. Fard, T. Fahringer, and R. Prodan, "Budget-Constrained Resource Provisioning for Scientific Applications in Clouds," *2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 315-322, 2013.

[13] J.J. Durillo, V. Nae, and R. Prodan, "Multi-Objective Workflow Scheduling: An Analysis of the Energy Efficiency and Makespan Tradeoff," *2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 203-210, 2013.

[14] J. Shuja, S. Madani, K. Bilal, K. Hayat, S. Khan, and S. Sarwar, "Energy-Efficient Data Centers," *Computing*, vol. 94, no. 12, pp. 973-994, 2012.

[15] F. Cao and M.M. Zhu, "Energy-aware Workflow Job Scheduling for Green Clouds," *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pp. 232-239, 2013.

[16] K. Grolinger, M.A. Hayes, W.A. Higashino, A. L'Heureux, D.S. Allison, and M.A.M. Capretz, "Challenges for MapReduce in Big Data," *2014 IEEE 10th World Congress on Services*, pp. 182-189, 2014.

[17] W. Tan, M.B. Blake, I. Saleh, and S. Dustdar, "Social-Network-Sourced Big Data Analytics," *IEEE Internet Computing*, vol. 17, no. 5, pp. 62-69, 2013.

[18] C.K-S. Leung, R.K. MacKinnon, and F. Jiang, "Reducing the Search Space for Big Data Mining for Interesting Patterns from Uncertain Data," *2014 IEEE International Congress on Big Data*, pp. 315-322, 2014.

[19] R. Fagin, "Combining fuzzy information from multiple systems," *Journal of Computer and System Sciences*, vol. 58, no. 1, pp. 216-226, 1996.

[20] W.-T. Balke and U. Güntzer, "Multi-objective Query Processing for Database Systems," *Proceedings of the International Conference on Very Large Databases (VLDB)*, pp. 936-947, 2004.

[21] P. Godfrey, R Shipley, and J. Gryz, "Maximal Vector Computation in Large Data Sets," *Proceedings of the Conference on Very Large Data Bases (VLDB)*, pp. 229-240, 2005.

[22] Z. Lacroix, M.-E. Vidal, and C. Legendre, "Customized and Optimized Service Selection with ProtocolDB," *Proceedings of the Conference on Data Management in Grid and Peer-to-Peer Systems*, Lecture Notes in Computer Science, Vol. 5697, pp. 112-123, 2009.

[23] M. Goncalves and M.-E. Vidal, "Reaching the Top of the Skyline: An Efficient Indexed Algorithm for Top-k Skyline Queries," *Proceedings of the Database and Expert Systems Applications Conference*, Lecture Notes in Computer Science, Vol. 5690, pp. 471-485, 2009.

[24] L. Qu, Y. Wang, M. Orgun, L. Liu, H. Liu, and A. Bouguettaya, "CCCloud: Context-aware and Credible Cloud Service Selection based on Subjective Assessment and Objective Assessment," *IEEE Transactions on Services Computing*, vol. 8, pp. 369-383, 2015.

[25] L. Saranya, "Optimal top-K queries processing: Sampling and dynamic scheduling approach," *2014 International Conference on Computer Communication and Informatics (ICCCI)*, pp.1-3, 2014.

[26] Z. Lacroix, C. Legendre, and S. Tuzmen, "Reasoning on Scientific Workflows," *2009 IEEE Congress on Services*, pp.306-313, 2009.

[27] K. Agrawal, A. Benoit, L. Magnan, and Y. Robert, "Scheduling algorithms for linear workflow optimization," *2010 IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, pp.1-12, 2010.

[28] M. Mao and M. Humphrey, "Auto-Scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows," *2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 1-12, 2011.

[29] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Cost- and Deadline-Constrained Provisioning for Scientific Workflow Ensembles in IaaS Clouds," *2012 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 1-11, 2012.

[30] L. Zeng, B. Veeravalli, and X. Li, "Scalestar: Budget Conscious Scheduling Precedence-Constrained Many-Task Workflow Applications in Cloud," *26th International Conference on Advanced Information Networking and Applications (AINA)*, Fukuoka, Japan, pp. 534–541, 2012.

[31] X.Y. Lin and C.Q. Wu, "On Scientific Workflow Scheduling in Clouds under Budget Constraint," *2013 42nd International Conference on Parallel Processing (ICPP)*, pp. 90-99, 2013.

[32] Q. Zhu and G. Agrawal, "Resource Provisioning with Budget Constraints for Adaptive Applications in Cloud Environments," *IEEE Transactions on Services Computing*, vol. 5, no. 4, pp. 497-511, 2012.

[33] D. de Oliveira, K.A.C.S, Ocaña, F.A. Baião, and M. Marta, "A Provenance-based Adaptive Scheduling Heuristic for Parallel Scientific Workflows in Clouds," *Journal of Grid Computing*, vol. 10, no. 3, pp. 521-552, 2012.

[34] H.M. Fard, R. Prodan, J.J.D. Barrionuevo, and T. Fahringer, "A Multi-Objective Approach for Workflow Scheduling in Heterogeneous Computing Environments," *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 300-309, 2012.

[35] M.A. Rodriguez and R. Buyya, "Deadline based Resource Pro-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2015.2466545, IEEE Transactions on Services Computing

visioning and Scheduling Algorithm for Scientific Workflows on Clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 222-235, 2014.

[36] W.N. Chen and J. Zhang, "An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 39, no. 1, pp. 29-43, 2009.

[37] J. Yu and R. Buyya, "A budget constrained scheduling of workflow applications on utility grids using genetic algorithms," *Workshop on Workflows in Support of Large-Scale Science (WORKS)*, pp. 1-10, 2006.

[38] J. Yu, R. Buyya, and C.K. Tham, "Cost-based Scheduling of Scientific Workflow Applications on Utility Grids," *2005 First International Conference on e-Science and Grid Computing*, pp. 140-147, 2005.

[39] I. Pietri, M. Malawski, G. Juve, E. Deelman, J. Nabrzyski, and R. Sakellariou, "Energy-Constrained Provisioning for Scientific Workflow Ensembles," *2013 Third International Conference on Cloud and Green Computing*, pp. 34-41, 2013.

[40] A. Narayan and S. Rao, "Power-Aware Cloud Metering," *IEEE Transactions on Services Computing*, vol. 7, no. 3, pp. 440-451, 2014.

[41] Q. Huang, S. Su, and J. Li, "Enhanced Energy-Efficient Scheduling for Parallel Applications in Cloud," *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012)*, pp. 781-786, 2012.

[42] T. Thanavanich and P. Uthayopas, "Efficient Energy Aware Task Scheduling for Parallel Workflow Tasks on Hybrids Cloud Environment," *2013 International Computer Science and Engineering Conference (ICSEC)*, pp.37-42, 2013.

[43] Z. Tang, Z.Z. Cheng, K.L. Li, and K.Q. Li, "An Efficient Energy Scheduling Algorithm for Workflow Tasks in Hybrids and DVFS-enabled Cloud Environment," *2014 Sixth International Symposium on Parallel Architectures, Algorithms and Programming*, pp. 255-261, 2014.

[44] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," *2010 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2010)*, pp. 6-20, 2010.

[45] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Journal of Future Generation Computer Systems*, vol. 28, pp. 755-768, 2012.

[46] H. Kimura, M. Sato, Y. Hotta, T. Boku, and D. Takahashi, "Emprical Study on Reducing Energy of Parallel Programs using Slack Reclamation by DVFS in a Power-Scalable High Performance Cluster," *2006 IEEE International Conference on Cluster Computing*, pp. 1-10, 2006.

[47] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, and R. Buyya, "CloudSim: a Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23-50, 2010.

[48] H. Topcuoglu, S. Hariri, and M.Y. Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260-274, 2002.

[49] J. Durillo, H. Fard, and R. Prodan, "MOHEFT: A Multi-Objective List-based Method for Workflow Scheduling," *2012 IEEE 4th International Conference on Cloud Computing Technology and Science*, pp. 185-192, 2012.

Zhongjin Li is a PhD candidate at the Software Institute, Nanjing University, under the supervision of Prof. Jidong Ge and Prof. Bin Luo. His research interests include cloud computing, workflow scheduling.

Jidong Ge is an Associate Professor at Software Institute, Nanjing University. He received his PhD degree in Computer Science from Nanjing University in 2007. His current research interests include cloud computing, workflow scheduling, software engineering, workflow modeling, process mining. His research results have been published in more than 40 papers in international journals and conference proceedings including JASE, ESA, ICSE, APSEC, ICSSP, HPCC, SEKE etc.

Haiyang Hu is a full Professor at School of Computer, Hangzhou Dianzi University, Hangzhou, China. He received the BS, MS, and PhD degrees in computer science from Nanjing University, Nanjing, China, in 2000, 2003, and 2006, respectively. His research interests include cloud computing, computer network, workflow scheduling, mobile computing, and distributed computing. His research results have been published in more than 40 papers in international journals and conference proceedings including IEEE TVCG, ACM TALIP, JNCA, WPC, ISF, HPCC, DASFAA, ISF, APWeb etc.

Wei Song received the PhD degree from Nanjing University, China, in 2010. He is currently an associate professor in the School of Computer science and Engineering at Nanjing University of Science and Technology, China. His research interests include software engineering and methodology, services computing, and business process management. He is a member of the ACM.

Hao Hu is an Associate Professor at Department of Computer Science and Technology, Nanjing University. He received his PhD degree in Computer Science from Nanjing University in 2009. His current research interests include cloud computing, computer network, workflow scheduling, and software engineering, process modeling, process mining. His research results have been published in more than 40 papers in international journals and conference proceedings including ICSE, ICSM, ICSSP, SCC etc.

Bin Luo is a full Professor at Software Institute, Nanjing University. His main research interests include cloud computing, computer network, workflow scheduling, and software engineering. His research results have been published in more than 40 papers in international journals and conference proceedings. He is leading the institute of applied software

engineering at Nanjing University.