Managing Deadline-constrained Bag-of-Tasks Jobs on Hybrid Clouds with Closest Deadline First Scheduling

Bo Wang[1 2], Ying Song[2], Yuzhong Sun[2] and Jun Liu[1]
[1] SPKLSTN Lab, Department of Computer Science and Technology, Xi'an Jiaotong University
Shaanxi, 710049, China
[e-mail: wangbo2012@stu.xjtu.edu.cn; liukeen@mail.xjtu.edu.cn]
[2] State Key Laboratory of Computer Architecture, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, 100190, China
[e-mail: songying, yuzhongsun}@ict.ac.cn]
*Corresponding author: Bo Wang

Received February 7, 2016; revised May 22, 2016; accepted June 10, 2016; published July 31, 2016

Abstract

Outsourcing jobs to a public cloud is a cost-effective way to address the problem of satisfying the peak resource demand when the local cloud has insufficient resources. In this paper, we studied the management of deadline-constrained bag-of-tasks jobs on hybrid clouds. We presented a binary nonlinear programming (BNP) problem to model the hybrid cloud management which minimizes rent cost from the public cloud while completes the jobs within their respective deadlines. To solve this BNP problem in polynomial time, we proposed a heuristic algorithm. The main idea is assigning the task closest to its deadline to current core until the core cannot finish any task within its deadline. When there is no available core, the algorithm adds an available physical machine (PM) with most capacity or rents a new virtual machine (VM) with highest cost-performance ratio. As there may be a workload imbalance between/among cores on a PM/VM after task assigning, we propose a task reassigning algorithm to balance them. Extensive experimental results show that our heuristic algorithm saves 16.2%-76% rent cost and improves 47.3%-182.8% resource utilizations satisfying deadline constraints, compared with first fit decreasing algorithm, and that our task reassigning algorithm improves the makespan of tasks up to 47.6%.

Keywords: Bag-of-tasks, cloud computing, hybrid cloud, task scheduling, resource management

A preliminary version of this paper is accepted by and presented in the 2016 Spring Simulation Multi-Conference featured with the 24th High Performance Computing Symposium (HPC/SpringSim 2016), April 3-6, 2016, Pasadena, CA, USA. This version includes a task reassigning algorithm to banlance workload imbalances between/amongst cores on a physical machine in the local cloud/cluster or a rented virtual machine from the public cloud and the performance evaluation for the task reassignment. The research was supported by National Science Foundation of China under Grant Nos. 61202060, 61428206, 61221063, 912183001, 61521092 and 61221062; National High Technology Research and Development Program 863 of China under Grant No. 2012AA011003; Cheung Kong Scholar's Program; Ministry of Education Innovation Research Team No. IRT13035.

http://dx.doi.org/10.3837/tiis.2016.07.005 **ISSN : 1976-7277**

1. Introduction

Hybrid cloud, combining a local cloud (private cloud) and a public cloud, is a cost-efficient way to address the problem of insufficient resources in the local cloud when its users have a peak resource demand as the peak load is much larger than average, but transient [1]. Surveyed by the European Network and Information Security Agency (ENISA), most of the small to medium enterprises prefer a mixture of cloud computing models (public cloud, private cloud) [2].

Reducing total capital expenditure on resources is a main objective on hybrid clouds for a provider owning local resources. Generally, using the resources of the local cloud is costless or cheaper, considering that investment costs for the physical infrastructures are "sunk costs", compared with leasing the resources from a public cloud. Thus minimizing the costs for a private cloud provider on a hybrid cloud is the integration of maximizing the resource utilizations of the local cloud and minimizing the rent cost from the public cloud.

There are various researches on scheduling the scientific computing applications on hybrid clouds. A few works focus on minimizing the rent cost with deadline constraints [3-9] or minimizing the makespan [10-12] for scientific computing applications by deciding which tasks should be outsourced to the public cloud. While, these works do not consider how the local cloud/cluster provisions resources, i.e. they do not provide the mapping between physical machines (PM) and provisioned resources in the local cloud.

Only a few hybrid cloud managements [13-15] have coordinated dynamic provisioning and scheduling that is able to cost-effectively complete applications within their respective deadlines. While these works separately scheduled tasks and provisioned resources, i.e., they first decided how many resources, each of which is either a virtual machine (VM) in public cloud or a PM in local cloud(s)/cluster(s), used for running tasks and then provisioned the resources from the resource pool, considering that all of the resources are homogeneous.

Different from these existing works, we study on cost-efficiently mapping the tasks to the resources for deadline-constrained Bag-of-Tasks (BoT) jobs, a kind of very common application in the parallel and distributed systems [16, 17], such as parallel image rendering, data analysis, and software testing [18-20], on a hybrid cloud with heterogeneous local resources. BoT jobs are often composed of hundreds of thousands of independent tasks and are CPU-intensive.

In this paper, we modeled the task and resource managements of hybrid clouds into a binary nonlinear programming (BNP) model. The model minimizes the cost for the resources leased from the public cloud satisfying the deadline constraints of jobs. As BNP is NP-hard problem [21], we proposed a heuristic algorithm to solve this BNP problem. In brief, the contributions of this paper can be summarized as follows:

- We modeled the hybrid cloud management which minimizes the cost for renting the resources of the public cloud for BoT jobs with deadline constraints into a BNP problem.
- To solve the BNP problem in polynomial time, we proposed a heuristic algorithm. The algorithm assigns a task to a core such that the difference between the task's finish time and its deadline is minimum in all assignments between unassigned tasks and cores of used PMs. If there is no such assignment, which means that there is no task can be completed within the deadline by the resources already used, the algorithm adds an available PM with most capacity or leases a VM with most cost-performance ratio from

the public cloud when there is no available PM in the local cloud, and assigns tasks to the cores of the added PM/VM as the previous step.

- After assigning tasks, the workloads on a PM/VM may be imbalance between cores. We proposed a task reassigning algorithm to balance them, which may improve the makespan of tasks or/and the lease time of VMs.
- We conducted extensive simulation experiments using two real work traces to investigate the effectiveness and efficiency of the proposed algorithm. The experiments results show that our heuristic algorithm saves 16.2%-76% cost for finishing jobs within their respective deadlines, improves 47.3%-182.8% resource utilizations, and assigns 52.6%-231.1% more load to local resources, compared with first fit decreasing (FFD) algorithm, and that our reassigning algorithm can decrease makespans up to 47.6%.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents our unified model, the heuristic algorithm for solving the model, and our task reassigning algorithm. Section 4 evaluates our work and Section 5 concludes this paper.

2. Related Works

There are various researches on scheduling scientific applications on hybrid clouds.

To minimize the cost for leased resources from public clouds, W. Z. Jiang and Z. Q. sheng [22] modelled the mapping of tasks and VMs as a bipartite graph. The two independent vertex sets of the bipartite graph are task and VM collections, respectively. The weight of an edge is the VM cost of a discrete task, i.e. the product of the running time of the task and the cost of the VM per unit time. Then the problem minimizing the cost is to find a subset of the edge set, where the weighted sum of all the edges in the subset is the minimum. The authors used the Hopcroft-Karp algorithm [23] to solve the minimum bipartite match problem. This work does not consider whether a task could be finish within the deadline.

Some existing work studied on minimizing cost with deadline constraints in hybrid clouds. Van den Bossche et al. [3-5] proposed a set of algorithms to cost-efficiently schedule the deadline-constrained BoT applications on both public cloud providers and private infrastructure while taking into account data constraints, data locality and inaccuracies in task runtime estimates. The Hybrid Cloud Optimized Cost (HCOC) scheduling algorithm [6, 7] tried to optimize the monetary execution costs resulting from the public nodes only while maintaining the execution time fitting deadline. HCOC first made an initial schedule using the Path Clustering Heuristic (PCH) algorithm [24] to schedule the tasks to the private cloud and then rescheduled some tasks to the public cloud if the deadline is missed. The algorithm can achieve cost optimization for workflow scheduling. Genez et al. [8] presented an integer linear program model. The numbers of each type of VM instances in both private and public clouds are obtained by solving this model to minimize cost without missing its deadline for a workflow. In this work, the authors considered the VM instances with same type being homogeneous, which is not apply to a private cloud with heterogeneous PMs. For completing the job on time and with minimum cost, Chu and Simmhan [9] first modelled a time-vary spot VM price as a Markov chain and then established a reusable table with three-tuple elements consisted of job compute requirement, deadline constraint of the job and a series of actions with minimal cost based on the price model. The subsequent action was decided by performing a simple table look-up. In this work, they considered that utilizing off local machines does not incur expense.

These above work studied on either resource provisioning or task scheduling. Moreover, most approaches for dynamic provisioning operate in a per-job level, and thus they are inefficient because they fail in consider that other tasks could utilize idle cycles of cloud resources. To address these problems, Aneka [13-15] coordinated dynamic provisioning and scheduling that is able to cost-effectively complete applications within their deadlines by considering the whole organization workload at individual tasks level when making decisions and an accounting mechanism to determine the share of the cost of utilization of public cloud resources to be assigned to each user. While Aneka separately scheduled tasks and provisioned resources, i.e., Aneka first decided how many resources, each of which is either a VM in public cloud or a PM in local cloud/grids, used for running tasks and then provisioned the resources from the resource pool, considering that these resources are homogeneous.

Besides minimizing cost, a few work focused on minimizing the makespan of scientific applications by cloud bursting. FermiCloud [10] despatched a VM on the PM that has the highest utilization but still have enough resource for the VM in private cloud. Only when all the resources in private cloud are consumed, VM are deployed on a public cloud. A new VM would be launched in a public cloud only when adding the VM can reduce the average job running time. Kailasam et al. [11, 12] proposed four cloud bursting schedulers whose main ideas are outsourcing a job to a public cloud when the estimated time between now and beginning execution of the job is greater than the estimated time consumed by migrating the job to the public cloud.

These aforementioned work studied on the task and/or resource management with one objective minimizing financial cost for private cloud providers or minimizing the makespan of applications. There are a few work focusing on a balance between two objectives. Taheri et al. [25] proposed a bi-objective optimization model minimizing both the execution time of a batch of jobs and the transfer time required to deliver their required data for hybrid clouds, and used a PSO-based approach to find the relevant pareto frontier. They do not take the finance expenditure into account. V. A. Leena et al. [26] proposed an algorithm for the simultaneous optimization of execution time and cost, in hybrid cloud, by determining whether tasks have to be scheduled to either the private cloud or the public cloud, employing a genetic algorithm. This work does not consider the mapping between VMs and PMs in the private cloud. The VM instances of the same type are homogeneous, which is not true in a heterogeneous data center. Wang et al. [27] proposed a dual-objective multi-dimension multichoice knapsack problem to model the task scheduling with the two objectives of minimizing the cost and minimizing the makespan in hybrid clouds. As the high complexity of solving the problem, the adaptive scheduling algorithm (AsQ) was proposed. AsQ used MAX-MIN strategy [28] to schedule task in private cloud and outsourcing the smallest task to the public cloud when the private cloud has insufficient resources. AsQ allocated the public resource slot with minimal cost to a task, fitting the deadline constraint of the task, considering that using extra public resource slots does not incur extra expense. HoseinyFarahabady et al. [29-31] studied on the balance between the makespan and the cost for BoT applications in hybrid clouds. They first established a BNP model with the objective of minimizing the sum of the weighted costs, i.e., the product of cost per unit time of a task and the running time raised to the power of a predefined factor of the task, and then relaxed the model by removing the binary constraints. By Lagrange multiplier method, the relaxed model was solved to get the workload assigned to each resource (PM in the private cloud or VM in public clouds). At last, they used FFD algorithm [32] to assigned tasks to resources so that the total workload of a resource are close to the value obtained from the last step.

In this paper, we studied on cost-efficiently mapping the tasks to the resources for deadline-constrained BoT applications on a hybrid cloud with heterogeneous local resources. Our work has the following main differences from these above works. (I) We considered that the resources are heterogeneous in the local cloud, which is very common as PMs get installed and replaced over the lifetime of a data center. (II) We directly provided the mapping of tasks to cores of PMs/VMs, instead of only mapping tasks to PMs/VMs as done by above works which did not provide the mapping between tasks and cores in a PM/VM.

3. Hybrid Cloud Management

In a hybrid cloud, as shown in **Fig. 1**, a task of jobs runs on a core of a PM on the local cloud/cluster or of a VM leased from the public cloud. In this paper, the objective is to cost-efficiently provision resources to tasks and assign the tasks to the resources to meet the complete time within corresponding deadline in the hybrid cloud environment, i.e., to provide the mapping between the PMs or rented VMs and the tasks with minimal cost while fitting deadlines.

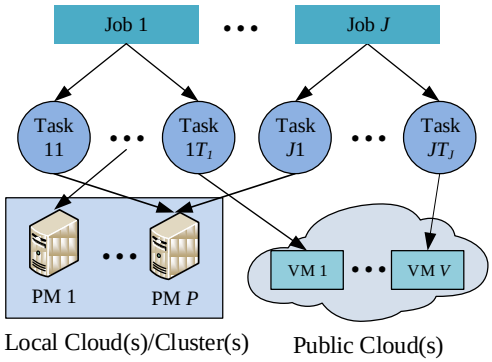

Fig. 1. Hybrid cloud environment. A task of jobs is assigned to either a PM in local cloud(s)/cluster(s) or a VM leased from public cloud(s).

3.1 Problem Formulation

We consider a hybrid cloud consisted of a local cloud/cluster and a public cloud. Multiple public clouds in a hybrid cloud can be seen as one big public cloud including the resources provisioned by these public clouds. **Table 1** summarizes notations used in this paper.

Table 1. Notations

Notations	Description
J	The number of jobs running on the hybrid cloud.
T_i	The number of tasks constituting job i.
T	The number of tasks, $T = \sum_{i=1}^{J} T_i$.
$t_{i,j}$	The j th task of job i.
d_i	The deadline of job i.
$r_{i,j}$	The resource amount needed to complete $t_{i,j}$.
P	The number of PMs in the local cloud.
V	The maximal number of VMs leased from the public cloud.

p_f	The price per unit time of VM f provisioned by the public cloud.
$\tau_{i,j,k}$ ($\tau_{i,j,P+f}$)	The running time of $t_{i,j}$ on PM k (VM f).
τ_k / τ_{P+f}	The usage time of PM k (VM f).
c_f	The cost for renting VM f .
N_k (N_{P+f})	The number of cores on PM k (VM f).
r_k (r_{P+f})	The capacity of each core on PM k (VM f).
$x_{i,j,k,l}$ ($x_{i,j,P+f,l}$)	The binary variable representing whether $t_{i,j}$ is assigned to core l of PM k (VM f).
N_{RV}	The number of rented VMs.

There are J jobs running on the hybrid cloud. Job i ($i=1,...,J$) is composed of T_i independent tasks, $\{t_{i,j} \mid j=1,...,T_i\}$. $T=\sum_{i=1}^{J} T_i$ represents the number of all tasks. It needs $r_{i,j}$ resource amounts to complete $t_{i,j}$. Job i must be completed before d_i. Without loss of generality, we assume that $d_1 \le d_2 \le \cdots \le d_J$.

In the local cloud/cluster, there are P PMs. In the public cloud, at most V VMs are rented. The price per unit time of VM f ($f=1,...,V$) is p_f. PM k ($k=1,...,P$) and VM f ($f=1,...,V$) has N_k and N_{P+f} cores, respectively. Each core of PM k (VM f) has r_k (r_{P+f}) capacity. It takes $\tau_{i,j,k}=r_{i,j}/r_k$ ($\tau_{i,j,P+f}=r_{i,j}/r_{P+f}$) time for completing $t_{i,j}$ when running on a core of PM k (VM f). If all the tasks scheduled to a core can be completed within respective deadlines, respectively, they can run in sequential order by their deadlines in ascending fashion to meet the deadlines. Then the deadline constraints can be formulated as follow (noticing that $d_1 \le d_2 \le \cdots \le d_J$):

$$\sum_{i'=1}^{i}\sum_{j=1}^{T_{i'}}(x_{i',j,k,l}\cdot\tau_{i',j,k}) \le d_i, \forall i=1,...,J, \forall l=1,...,N_k, \forall k=1,...,P+V, \tag{1}$$

and

$$\sum_{i'=1}^{i}\sum_{j=1}^{T_{i'}}(x_{i',j,P+f,l}\cdot\tau_{i',j,P+f}) \le d_i, \forall i=1,...,J, \forall l=1,...,N_{P+f}, \forall f=1,...,V, \tag{2}$$

where the binary variable $x_{i,j,k,l}$ ($i=1,...,J$, $j=1,...,T_i$, $k=1,...,P$, $l=1,...,N_k$) or $x_{i,j,P+f,l}$ ($i=1,...,J$, $j=1,...,T_i$, $f=1,...,V$, $l=1,...,N_{P+f}$) represents whether $t_{i,j}$ is assigned to core l of PM k or VM f . If so, $x_{i,j,k,l}=1$ or $x_{i,j,P+f,l}=1$, otherwise, $x_{i,j,k,l}=0$ or $x_{i,j,P+f,l}=0$. The left sides of Inequations (1) and (2) represent the finish times of $t_{i,j}$, $i=1,...,J$, $j=1,...,T_i$, respectively. The total time using a PM/VM is the maximum finish time of the task running on it,

$$\tau_k = \max_{1\le l\le N_k}\sum_{i=1}^{J}\sum_{j=1}^{T_i}(x_{i,j,k,l}\cdot\tau_{i,j,k}), \forall k=1,...,P, \tag{3}$$

$$\tau_{P+f} = \max_{1 \le l \le N_{P+f}} \sum_{i=1}^{J} \sum_{j=1}^{T_i} (x_{i,j,P+f,l} \cdot \tau_{i,j,P+f}), \forall f = 1,...,V. \quad (4)$$

Thus, the costs for leasing VMs from public cloud respectively are

$$c_f = \lceil \tau_{P+f} \rceil \cdot p_f, \forall f = 1,...,V, \quad (5)$$

where $\lceil \tau_{P+f} \rceil$ is the ceiling integer of τ_{P+f}.

We formulate the problem of hybrid cloud management as a BNP as follows:

$$\text{Minimize} \sum_{f=1}^{V} c_f \quad (6)$$

subject to:

$$\sum_{k=1}^{P} \sum_{l=1}^{N_k} x_{i,j,k,l} + \sum_{f=1}^{V} \sum_{l=1}^{N_{P+f}} x_{i,j,P+f,l} = 1, \forall j = 1,...,T_i, \forall i = 1,...,J, \quad (7)$$

Inequations (1) and (2), and Equations (3), (4) and (5),

$$x_{i,j,k,l} \in \{0,1\}, \forall j = 1,...,T_i, \forall i = 1,...,J, \forall l = 1,...,N_k, \forall k = 1,...,P, \quad (8)$$

$$x_{i,j,P+f,l} \in \{0,1\}, \forall j = 1,...,T_i, \forall i = 1,...,J, \forall l = 1,...,N_{P+f}, \forall f = 1,...,V, \quad (9)$$

The decision variables are $x_{i,j,k,l}$ ($j = 1,...,T_i$, $i = 1,...,J$, $l = 1,...,N_k$, $k = 1,...,P$) and $x_{i,j,P+f,l}$ ($j = 1,...,T_i$, $i = 1,...,J$, $l = 1,...,N_{P+f}$, $f = 1,...,V$). The objective (6) of this model is minimizing the rent cost ($\sum_{f=1}^{V} c_f$) for VMs to complete jobs within respective deadlines. Constraints (7) ensure that each task must be assigned to exactly one core. Constraints (8) and (9) represent the binary requirements for the decision variables. After solving this model, we achieve the task assignments, $x_{i,j,k,l}$ ($i = 1,...,J$, $j = 1,...,T_i$, $k = 1,...,P$, $l = 1,...,N_k$) and $x_{i,j,P+f,l}$ ($i = 1,...,J$, $j = 1,...,T_i$, $f = 1,...,V$, $l = 1,...,N_{P+f}$), and the renting time for each VM, c_f / p_f ($f = 1,...,V$).

3.2 The Heuristic Algorithm

As BNP is NP-hard [21], we propose a heuristic algorithm to solve the model presented in Section 3.1 in polynomial time. The main idea of the algorithm is to assign the task to a core so that the finish time of the task is closest to its deadline. If there is no enough resource, the algorithm adds an available PM with most capacity or leases a VM with best cost-performance ratio from the public cloud when there is no available PM in the local cloud. The details of the algorithm are described as follows, outlined in Algorithm 1.

Algorithm 1 assigning tasks to PMs or/and rented VMs

$\mathcal{T}$: the set of unassigned tasks, 3-tuples: (an unassigned task, its requirement of resource amount, its deadline);
$\mathcal{C}$: the set of available cores, 4-tuples: (a core, its capacity, the finish time of all tasks assigned to it, the PM/VM containing it);
$\mathcal{PM}$: the set of idle PMs, 3-tuples: (an idle PMs, its core number, the capacity of a core);
$\mathcal{VT}$: the set of available VM instance types, 4-tuples: (an available VM instance type, its core number, the capacity of a core, its price per unit time);
$\mathcal{A}$: the set of assignments, 5-tuples: (a task, the core running it, the PM/VM containing the core, its start time).

Input: $\mathcal{T}$; $\mathcal{PM}$; $\mathcal{VT}$
Output: $\mathcal{A}$

1: **while** $\mathcal{T} \neq \phi$ **do**
2: **if** $\mathcal{C} = \phi$ **then** */*There is no available resource*/*
3: **if** $\mathcal{PM} \neq \phi$ **then** */*Adding the idle PM with maximum capacity*/*
4: $pm \leftarrow p : (p \in \mathcal{PM}) \wedge (p(2) \cdot p(3) = \max_{p' \in \mathcal{PM}} (p'(2) \cdot p'(3)))$;
/$p(i)$ is ith element in tuple p.*/*
5: $\mathcal{PM} \leftarrow \mathcal{PM} \setminus \{pm\}$;
6: **for** $i = 1$ to $pm(2)$ **do**
7: $\mathcal{C} \leftarrow \mathcal{C} \cup \{(\text{new core}, pm(3), 0, pm(1))\}$;
8: **end for**
9: **else** */*Renting an VM with highest cost-performance ratio*/*
10: $vm \leftarrow v : (v \in \mathcal{VT}) \wedge \text{C1}(v) \wedge \text{C2}(v) \wedge \text{C3}(v)$;
/* $\text{C1}(v) : (\forall t)((t \in \mathcal{T}) \wedge (t(2)/v(3) \leq t(3)))$,
// satisfying deadline constraints of tasks
$\text{C2}(v) : v(2) \cdot v(3)/v(4) = \max_{(v' \in \mathcal{VT}) \wedge \text{C1}(v')} (v'(2) \cdot v'(3)/v'(4)))$,
//having best cost-performance ratio with condition C1
$\text{C3}(v) : v(4) = \min_{(v' \in \mathcal{VT}) \wedge \text{C1}(v') \wedge \text{C2}(v')} v'(4)$
*//having minimal price per unit time with C1 and C2 */*
11: **for** $i = 1$ to $vm(2)$ **do**
12: $\mathcal{C} \leftarrow \mathcal{C} \cup \{(\text{new core}, vm(3), 0, \text{a new VM instance of type } vm(1))\}$;
13: **end for**
14: **end if**
15: **else** */*Scheduling a task on an available core*/*
16: $(task, core) \leftarrow (t, c) : \text{C4}(t, c) \wedge \text{C5}(t, c) \wedge \text{C6}(t, c)$;
/* $\text{C4}(t, c) : (t \in \mathcal{T}) \wedge (c \in \mathcal{C}) \wedge (t(3) \geq c(3) + t(2)/c(2))$
//satisfying the deadline constraint of task t
$\text{C5}(t, c) : t(3) - (c(3) + t(2)/c(2)) = \min_{\text{C4}(t', c')} (t'(3) - (c'(3) + t'(2)/c'(2)))$
//selecting the task closest to its deadline with C4
$\text{C6}(t, c) : t(2) = \max_{\text{C4}(t', c') \wedge \text{C5}(t', c')} (t'(2))$
*//selecting the longest task with C4 and C5 */*
17: **if** $\{(task, core)\} \neq \phi$ **then**
18: $\mathcal{A} \leftarrow \mathcal{A} \cup \{(task(1), core(1), core(4), core(3), core(3) + task(2)/core(2))\}$
//adding the assignment (task, core) to $\mathcal{A}$;
19: $core(3) \leftarrow core(3) + task(2)/core(2)$;
20: $\mathcal{T} \leftarrow \mathcal{T} \setminus \{task\}$;
21: **else** */*There is no available resource in $\mathcal{C}$ for any task*/*
22: $\mathcal{C} \leftarrow \phi$;
23: **end if**
24: **end if**
25: **end while**

Step 1: If there is no available resource (line 2), the algorithm would add an available PM (lines 3-8) or lease a VM (lines 9-14) from public cloud when there is no available PM in local cloud. The selection principle of a PM is to selecting the PM with most capacity (line 4). The selection principle of rented VM is that the selected VM has the capacity to complete any task when running alone (C1 in line 10) and has best cost-performance ratio (C2 in line 10). If there are multiple types of VMs having same cost-performance ratio, the algorithm selects the VM with minimal price per unit time (C3 in line 10). After selection, the algorithm adds the cores of selected PM (lines 6-8) or rented VM (lines 11-13) to the pool of available cores.

Step 2: When there are one or more available cores (line 15), the algorithm assigns the unassigned tasks to these cores (lines 15-24). For all of assignments between unassigned tasks and available cores, the algorithm examines the finish times of the tasks and selects the assignment that the difference between the finish time of the task and its deadline is minimal and that the task is finished within its deadline (lines 16-20). If no assignment that the task can be finished within its deadline, there is no available resource for the unassigned task (lines 21-22).

Step 3: The algorithm repeats Steps 1 and 2 until there is no unassigned task (line 1).

The computing resource consumed by the algorithm are mainly composed of the selection of an available PM or VM and the decision of an assignment between a task and a core. In real world, the numbers of VM instance types and cores in a PM/VM both are a few tens or fewer, thus the selection of a PM or VM and the decision of an assignment are $O(P)$ and $O(T)$, respectively, in time complexity. Therefore, assigning tasks to PMs is $O(T \cdot (P+T))$ in time complexity. We assume that there are N_{RV} VMs leased from public cloud for completing the tasks within their respective deadlines. Assigning tasks to a VM is $O(T)$ in time complexity. Thus, assigning tasks to the rented VMs is $O(T \cdot N_{RV})$ in time complexity. Hence, the algorithm is $O(T \cdot (P+T+N_{RV}))$ in time complexity, overall.

3.3 Improvement on the Heuristic Algorithm

After task assignment, there may be some PMs/VMs on which the workloads are imbalance among cores. For these unbalanced PMs/VMs, the finish times of tasks or/and the lease time can be improved by balancing the workloads. For example, as shown in **Fig. 2**, four tasks are assigned to a VM with two cores, and the first three tasks are assigned to one core while the fourth task are assigned to another core. These four tasks will be finished within first, second, third, and second unit time, respectively. The lease time of the VM is 3 unit time. While if Task 3 is assigned to another core, the lease time would be reduced to 2 unit time and the finish time of Task 3 would be reduced to within second unit time.

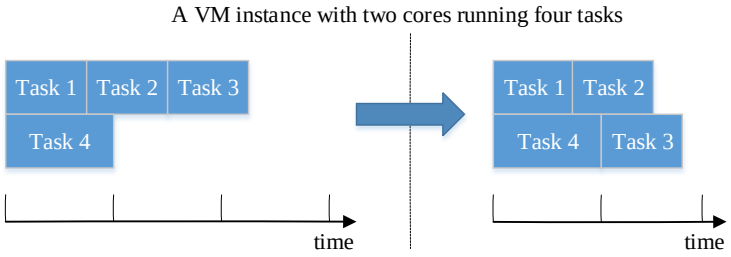

Fig. 2. An example that the lease time of a VM instance and the finish time of a task are improved without additional cost by reassigning tasks.

Algorithm 2 The algorithm reassigning tasks to cores on a PM/VM to improve the imbalance of a task assignment.

Input: $\mathcal{A}'$: The initial task assignment.
Output: $\mathcal{A}$: The task assignment after reassignment.

1: **for** each $a \in \mathcal{A}'$ **do**
2: $\mathcal{B} \leftarrow \{b \mid (b \in \mathcal{A}') \wedge (b(3) = a(3))\}$;
3: $\mathcal{A}' \leftarrow \mathcal{A}' \setminus \mathcal{B}$;
4: **for** each $core$ in $a(3)$ **do**
5: $lft(core) \leftarrow \max_{(b \in \mathcal{B}) \wedge (b(2) = core)} b(5)$;
/*latest finish time of tasks assigned to core*/
6: **end for**
7: **while** true **do**
8: $ft_{max} \leftarrow \max_{core \text{ in } a(3)} lft(core)$;
9: $C_{max} \leftarrow core$: $lft(core) = ft_{max}$;
10: $A_{max} \leftarrow b$: $(b(5) = ft_{max}) \wedge (b \in \mathcal{B}) \wedge (b(2) = C_{max})$;
11: $ft_{min} \leftarrow \min_{core \text{ in } a(3)} mft(core)$;
12: $C_{min} \leftarrow core$: $lft(core) = ft_{min}$;
13: **if** $A_{max}(5) < A_{max}(5) - A_{max}(4) + ft_{min}$ **then**
14: $lft(C_{max}) \leftarrow lft(C_{max}) - (A_{max}(5) - A_{max}(4))$;
15: $lft(C_{min}) \leftarrow lft(C_{min}) + (A_{max}(5) - A_{max}(4))$;
16: $A_{max}(2) \leftarrow C_{min}$;
17: $A_{max}(5) \leftarrow A_{max}(5) - A_{max}(4) + ft_{min}$;
18: $A_{max}(4) \leftarrow ft_{min}$;
19: $\mathcal{B} \leftarrow \mathcal{B} \setminus \{b\}$;
20: $\mathcal{B} \leftarrow \mathcal{B} \cup \{A_{max}\}$;
21: **else**
22: **goto** line 3;
23: **end if**
24: **end while**
25: $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{B}$;
26: **end for**

For improving the imbalance on a PM/VM, we present the reassignment algorithm (RA), outlined in Algorithm 2. For each PM/VM (lines 1-3), the algorithm examines whether the finish time of the task with latest finish time (lines 8-10) could be reduced by reassigning the task to the core with the lightest load (lines 11-12) on a PM/VM, and, if so, reassigns the task to the core (lines 13-20). The algorithm repeats the step until the latest finish time can not be reduced (lines 21-22). The time complexity of the algorithm is no more than the number of tasks because only some of tasks are considered to reduce the latest finish time for a PM/VM, thus RA is $O((P + N_{RV}) \cdot T)$ in time complexity at worst.

4. Experiments Results and Analysis

In this section, we introduce our testbed and experiment design, and then discuss the experimental results.

4.1 Testbed and Experiments Design

We use a 3-month trace collected from the University of Luxemburg *Gaia* cluster system and a 2-month trance collected from NASA Ames iPSC/860, i.e., *UniLu Gaia log* and *NASA iPSC log* in Parallel Workloads Archive [33], to evaluate the performance of our algorithms. We assume that the trace data are the information of tasks running on 1 GHz cores. The features of these two traces are shown in **Table 2** where short and long tasks are considered as tasks

whose running time shorter than and equal to 1 hour and longer than 1 hours, respectively. As shown in the table, we can see that the proportion of short tasks in Gaia trace is smaller than that in NASA trace. We set the deadline of each task as α (1, 2, 3, or 4) times of its run time on a 2 GHz core.

Table 2. The number of tasks and the proportions of short and long tasks in Gaia and NASA traces.

Trace	Task Number	Proportion of Short Tasks	Proportion of Long Tasks
Gaia	38642	69.49%	30.51%
NASA	18066	71.11%	28.89%

The PMs used as the local resources are shown in **Table 3**. In public cloud, we use a compute optimized instance type, c3.large in EC2 [34], because it has the best cost-performance ratio for CPU-intensive applications, compared with other instances provisioned by EC2. Each VM instance has 2 vCPUs with 2.7GHz. The price of a VM instance is $0.105 per hour (in US east (N. Virginia)).

Table 3. The CPU configurations and numbers of PMs used for local resources.

Type	CPU Frequency × #cores	#PM
1	2.378 GHz × 8	5
2	2.33 GHz × 8	5
3	2.216 GHz × 4	5

We compare our algorithm (HA) against a baseline algorithm, First Fit Decreasing (FFD) [32], one of the most popular task scheduling algorithm. FFD is assigning the longest task to the first core on which it will fit.

We compare task management algorithms in the following aspects:

- *rent cost*: the cost for leasing VMs from the public cloud;
- *local load*: the load assigned to local resources, $\sum_{k=1}^{P}\sum_{l=1}^{N_k}\sum_{i=1}^{J}\sum_{j=1}^{T_i}(x_{i,j,k,l}\cdot\tau_{i,j,k})$. More local load reflects less loads assigned to public cloud and thus less rented resources;
- *makespan*: the latest finish time of tasks;
- *resource utilization*: the overall utilization of used PMs and rented VMs;
- *energy consumption*: the energy consumed by PMs;
- *overhead*: the CPU times consumed by task management algorithms;
- *scalability*: the growth trends of overhead as the PM and task numbers increase.

4.2 Comparison of Task Management Algorithms

Figs 3, **4**, **5**, **6**, **7**, and **8** show rent costs, local loads, makespans of tasks, overall resource utilizations, energy consumption and overheads for finishing the tasks, managed by FFD and HA running on a Intel(R) Xeon(R) CPU E5410 @ 2.33GHz core, within their respective deadlines, respectively.

For completing Gaia and NASA tasks within their respective deadlines, respectively, as shown in **Fig. 3**, HA consumes less 16.2%-76% cost than FFD for leasing VMs from public cloud. The reason is that HA uses more local resources than FFD, reflected by **Fig. 4** showing that HA assigns more 52.6%-231.1% loads to local resources than FFD, and thus consumes less resources leased from the public cloud. While HA postpones the finish time of these tasks to 11.8%-82.6% later, as shown in **Fig. 5**, compared with FFD, with completing the tasks

within their respective deadlines. The less rent cost and longer finish time of HA imply that HA makes better use of resources, compared with FFD. From **Fig. 6**, we can see that the resource utilization of HA is 47.3%-182.8% more than that of FFD. These can be explained as follows. FFD schedules longest task first, and thus there will be small tasks left to be scheduled after assigning most of tasks. For finishing the left small tasks within their respective deadlines, only a few small tasks are assigned to one rented VM. Therefore, these VMs running only small tasks are under-utilised as rented VMs are charged by unit time, leading to high costs and low utilizations. While HA schedules the task closest to its deadline first, and thus rarely has the problem.

Fig.s 3 and **5** show that the rent cost is decreased with increasing the deadline while the finish time increases with deadline. The reason is that tasks are allocated less resources, which leads to longer waiting time of tasks and thus longer time to finish these tasks, if their deadlines are postponed. From **Fig. 3**, we can also see that, compared with FFD, HA has decreasing ratios increasing from 24.2% to 43.2% for Gaia trace and 16.2% to 62.5% for NASA trace in rent cost as the deadline factor α increases from 1 to 4, i.e., the cost improved by HA is more and more better than that by FFD as deadlines are postponed overall. The reason is that, when deadlines are postponed, more short tasks are consolidated in both PMs with long tasks and VMs with idle rent time which is less than 1 hour to reduce rent costs for HA, while for FFD, short tasks are hardly consolidated in PMs as FFD schedules long tasks to PMs first to increase resource utilizations. We also get the result that the increasing extent of decreasing ratios of rent costs for NASA trace is greater than for Gaia trace as increasing deadlines for HA compared with FFD, from **Fig. 3**. This can be explained as follows. The proportion of short tasks in NASA trace is larger than that in Gaia trace, as shown in **Table 2**. More short tasks mean more VMs with idle rent time when deadlines are short, and thus more room for improving rent costs by consolidating short tasks in PMs or VMs with idle rent time when postponing deadlines, and HA makes better usage of these improvements than FFD as analyzed above.

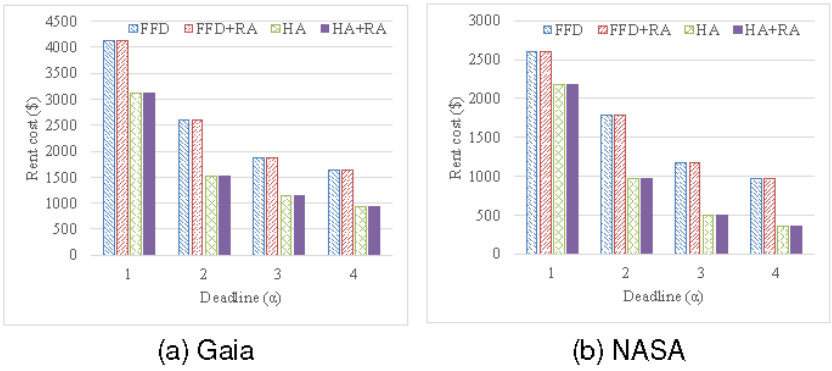

Fig. 3. The rent costs for finishing Gaia (a) and NASA (b) tasks within their deadlines

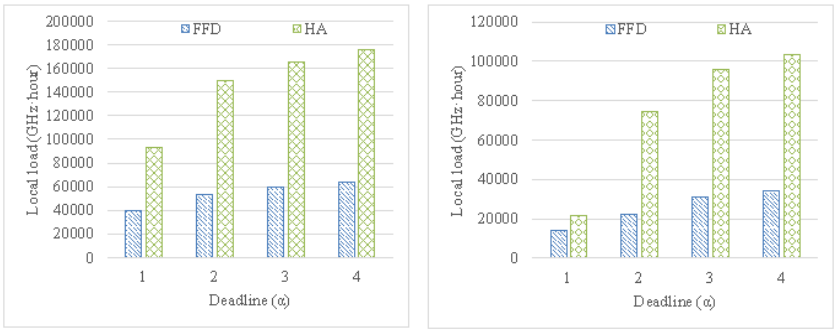

Fig. 4. The load assigned to local resources

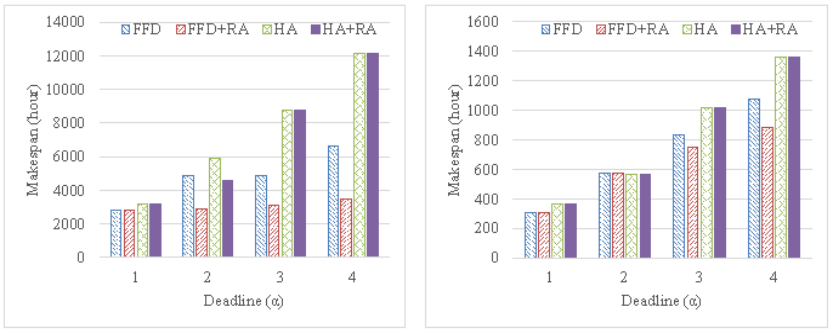

Fig. 5. The times finishing Gaia (a) and NASA (b) tasks

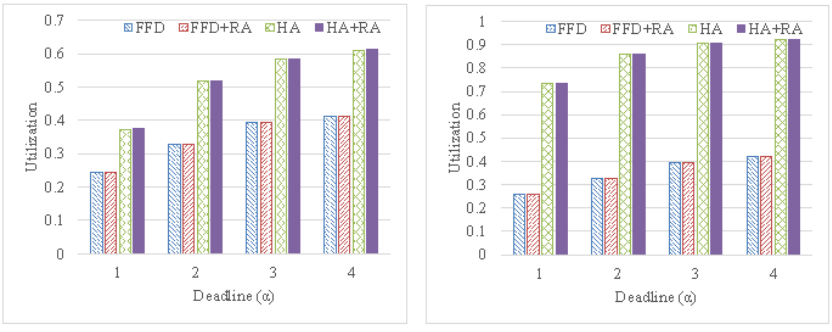

Fig. 6. The utilizations of PMs/VMs running Gaia (a) and NASA (b) tasks

Assigning more loads to local resources increasing the energy consumed by local cloud. Here we examine the energy consumption of PMs for HA and FFD. To calculate energy consumption, we use the prevalent linear model [35, 36] and the parameters of a modern mid-rang computer of which consumed powers with idle and full load respectively are $70W$ and $110W$, given in [36], i.e. the energy consumption of a PM is

$$E = \int_{\Delta\tau} ((110-70)\cdot u(\tau)+70)\mathrm{d}\tau \times 10^{-3} KW \cdot H \tag{10}$$

where $\Delta\tau$ is the time interval of using the PM to run tasks and $u(\tau)$ is the resource utilization

at time τ. The results are shown in **Fig. 7**. As shown in this figure, HA consumes only less than 50% more energy than FFD. Compares with FFD, the increase rates of energy consumptions are much less than that of loads assigned to local resources, respectively, for HA. This implies that HA uses local resources much more effectively than FFD.

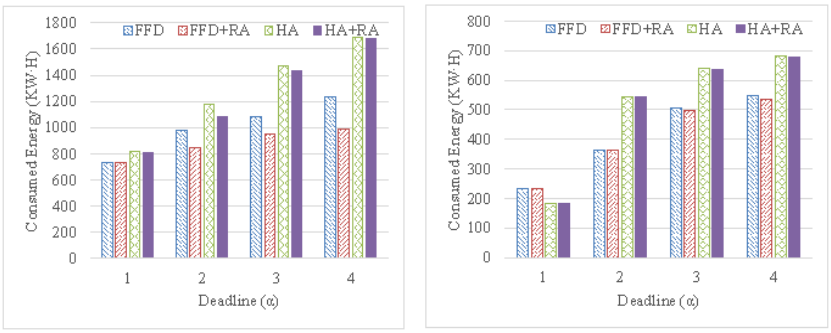

Fig. 7. The estimated energy consumed by PMs

As shown in **Fig. 8**, HA consumes less time than FFD when the deadline is early (α = 1 or 2), while it consumes more time than FFD when it is late (α = 3 or 4). The reasons are as follows. The time complexities of HA and FFD are $O(T \cdot (P + T + N_{RV}))$ and $O(T \cdot (P + N_{RV}))$, respectively. When the deadline is earlier, the number of rented VMs is larger. When α = 1 or 2, the number of rented VMs used by FFD is larger than 4000, as shown in **Fig. 9**, which is above 288% more than that by HA and is larger than the number of tasks for both Gaia and NASA traces, and thus the time consumed by FFD is more than HA. When $\alpha \geq 3$, the rented VM numbers are smaller than task numbers, therefore, FFD consumes less time than HA.

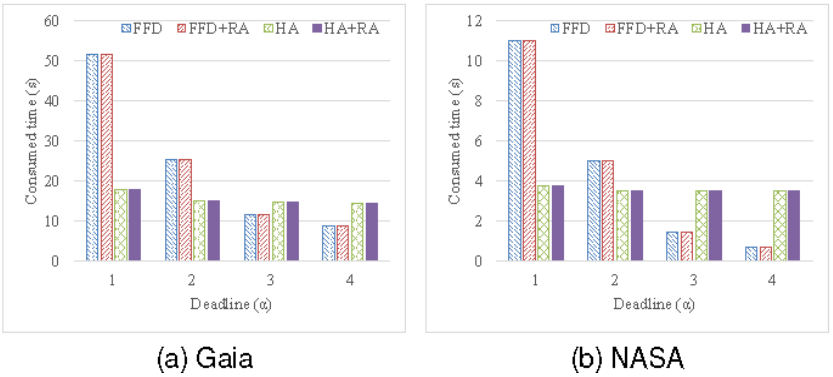

Fig. 8. The CPU times consumed by task assignments

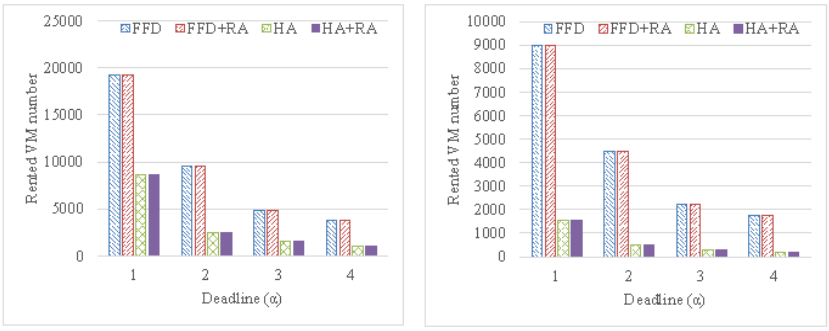

Fig. 9. The numbers of rented VMs for completing tasks within respective deadlines

Thus, HA is much better than FFD in minimizing costs and improving resource utilizations with only a few overheads for resource and task managements of hybrid clouds.

Next, we examine the scalabilities of HA and FFD on the number of local PMs. We scale the PM resources by a factor ranging from 1 to 10. For example, when the scale factor is 1, 15 PMs described in **Table 2** are used for running tasks, while there would be 150 PMs when the scale factor is 10. Fig.s 10 and 11 show the changes of costs and time consumed by HA and FFD with the scale factor, respectively. We present the results of the case of $\alpha = 4$ here. Other cases have similar results. As shown in **Fig. 10**, the cost is decreased with increasing of PM numbers because the resources should be leased from the public cloud are reduced in amount as the amount of local resources increases. From **Fig. 11**, we can see that HA costs less 24.1%-75.1% and about 20% than FFD for Gaia and NASA traces, respectively, for any scale of local cloud, i.e., HA always consumes less cost than FFD. As shown in **Fig. 11**, the times consumed by FFD and HA both are stable as the increase of PM number. The reason is that the increase of PM number results in the decreasing of rented VM number, leading to the total number of the PMs and rented VMs ($P + N_{RV}$) having almost no change as the PM number increases.

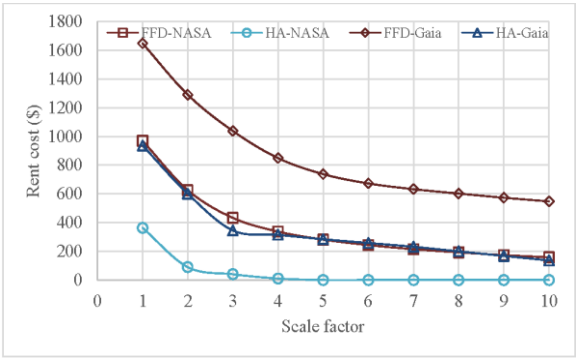

Fig. 10. The rent costs decreasing with increasing of PM numbers

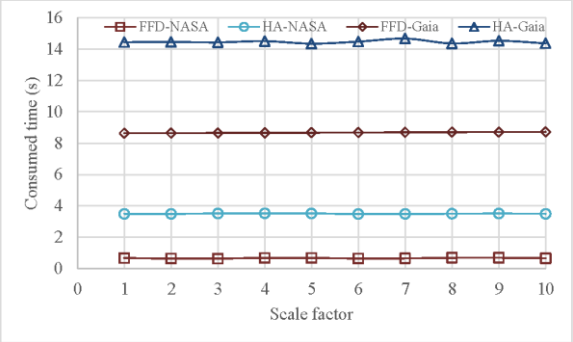

Fig. 11. The times consumed by task assignments, stable when the local PM number is increasing

Now, we examine the scalabilities of HA and FFD on the number of tasks by changing the task scale from 1000 to 35000 in number using Gaia trace. **Fig. 12** presents the result of the case of $\alpha = 1$. Other cases have similar results. As shown in the figure, the consumed times of HA and FFD both are quadratically increasing with the task number, which is consistent with their respective time complexities. **Fig. 12** shows that the costs have nearly linear increases with the increase of task number as more resources required for processing more tasks and that HA saves 25.7%-58.1% costs compared with FFD.

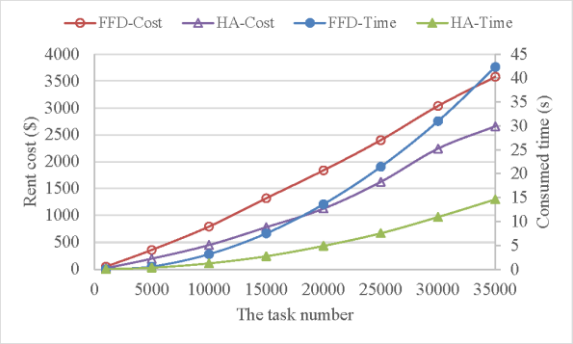

Fig. 12. The variations of rented costs and consumed times by task assignments with increasing the task number

These results above indicate that HA is better than FFD in minimizing rent cost and that both HA and FFD have good scalability.

4.3 Performance of the Improvement Method

In this section, we experimentally study on the improvement of our improvement method (RA presented in Section 3.3) on the performance of task managements by comparing these two task managements with (+RA) and without combining RA.

As shown in **Fig. 3**, we can see that RA is hardly improving rent costs. This is because that the imbalances mostly exist in PMs, leading to hardly reduction of the lease time of VMs by reassigning.

Fig. 5 shows that RA improves 36%-47.6% task makespans for FFD when $\alpha \geq 2$ while 22.3% for HA only when $\alpha = 3$, which indicates that the workloads managed by HA are much more balance than that by FFD on a PM/VM. RA also improves up to 20% and up to 7.8% energy consumptions for FFD and HA, respectively, as shown in **Fig. 7**, by reducing the finish time of the tasks running on PMs.

As shown in **Fig. 6**, the resource utilizations are almost not improved by HA. This is because the lease times of VMs are almost not reduced by HA, leading to almost no improvement of resource utilizations for VMs. The numbers of VMs are much more than that of PMs, respectively, resulting in negligible improvement of overall resource utilizations when the resource utilizations of PMs are improved.

RA consumes only less than 0.005s CPU time which is negligible, as shown in **Fig. 8**, and does not change the number of rented VMs for hybrid cloud managements, as shown in **Fig. 9**.

These above observations indicates that RA improves the hybrid cloud managements in improving rent costs, makespans of tasks, and resource utilizations with negligible overheads.

5. Conclusion

In this paper, we studied the hybrid cloud management for Deadline-constrained BoT jobs. We first model the hybrid cloud management to a BNP problem which minimizes the rent costs for leasing VMs from public cloud. As BNPs are NP-hard problems, we propose the heuristic algorithm to solve the BNP problem in polynomial time. The algorithm's main idea is assigning a task to a core such that the difference between the finish time of the task and its deadline is minimal in all of feasible assignments. If none of unassigned tasks can be completed within its deadline, the algorithm adds an available PM with most capacity or rents a new VM with highest cost-performance ratio and assigns tasks to the new PM/VM as previous step. As there are workload imbalances between cores on a PM/VM after task assigning, we propose a task reassigning algorithm to balance them. Extensive experiments using real world traces have been conducted to study on the effectivenesses and efficiencies of our heuristic algorithm and reassigning algorithm.

In this paper, we focused on BoT jobs consisted of independent tasks. In the future, we will study the management of workflow jobs containing tasks which can be started only when the tasks they depend on are finished on hybrid clouds.

References

[1] Xiaozhu Kang, Hui Zhang, Guofei Jiang, Haifeng Chen, Xiaoqiao Meng, and K. Yoshihira, "Measurement, Modeling, and Analysis of Internet Video Sharing Site Workload: A Case Study," in *Proc. of Web Services, 2008. ICWS '08. IEEE International Conference on*, pages 278-285, Sept 2008. Article (CrossRef Link).

[2] Raouf Boutaba and Nelson LS da Fonseca, "Cloud Architectures, Networks, Services, and Management," *Cloud Services, Networking, and Management*, pages 1-22, 2015. Article (CrossRef Link).

[3] Ruben Van den Bossche, Kurt Vanmechelen, and Jan Broeckhove, "Online Cost-efficient Scheduling of Deadline-constrained Workloads on Hybrid Clouds," *Future Generation Computer Systems*, 29(4):973 - 985, 2013. Article (CrossRef Link).

[4] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads," in *Proc. of Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 228-235, July 2010. Article (CrossRef Link).

[5] Ruben Van den Bossche, Kurt Vanmechelen, and Jan Broeckhove, "Cost-Efficient Scheduling Heuristics for Deadline Constrained Workloads on Hybrid Clouds," in *Proc. of Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 320-327, Nov 2011. Article (CrossRef Link).

[6] LuizFernando Bittencourt and EdmundoRobertoMauro Madeira, "HCOC: a Cost Optimization Algorithm for Workflow Scheduling in Hybrid Clouds," *Journal of Internet Services and Applications*, 2(3):207-227, 2011. Article (CrossRef Link).

[7] L.F. Bittencourt, C.R. Senna, and E.R.M. Madeira, "Scheduling Service Workflows for Cost Optimization in Hybrid Clouds," in *Proc. of Network and Service Management (CNSM), 2010 International Conference on*, pages 394-397, Oct 2010. Article (CrossRef Link).
[8] Thiago A. L. Genez, Luiz F. Bittencourt, and Edmundo R. M. Madeira, "On the Performance-Cost Tradeoff for Workflow Scheduling in Hybrid Clouds," in *Proc. of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, UCC '13*, pages 411-416, Washington, DC, USA, 2013. IEEE Computer Society. Article (CrossRef Link).
[9] Hsuan-Yi Chu and Y. Simmhan, "Cost-Efficient and Resilient Job Life-Cycle Management on Hybrid Clouds," in *Proc. of Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pages 327-336, May 2014. Article (CrossRef Link).
[10] Hao Wu, Shangping Ren, G. Garzoglio, S. Timm, G. Bernabeu, H.W. Kimy, K. Chadwick, Haengjin Jang and Seo-Young Noh, "Automatic Cloud Bursting under FermiCloud," in *Proc. of Parallel and Distributed Systems (ICPADS), 2013 International Conference on*, pages 681-686, Dec 2013. Article (CrossRef Link).
[11] S. Kailasam, N. Gnanasambandam, J. Dharanipragada, and N. Sharma, "Optimizing Ordered Throughput Using Autonomic Cloud Bursting Schedulers," *Software Engineering, IEEE Transactions on*, 39(11):1564-1581, Nov 2013. Article (CrossRef Link).
[12] S. Kailasam, N. Gnanasambandam, J. Dharanipragada, and N. Sharma, "Optimizing Service Level Agreements for Autonomic Cloud Bursting Schedulers," in *Proc. of Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, pages 285-294, Sept 2010. Article (CrossRef Link).
[13] Rodrigo N. Calheiros, Christian Vecchiola, Dileban Karunamoorthy, and Rajkumar Buyya, "The Aneka Platform and QoS-driven Resource Provisioning for Elastic Applications on Hybrid Clouds," *Future Generation Computer Systems,* 28(6):861-870, 2012. Article (CrossRef Link).
[14] Christian Vecchiola, Rodrigo N. Calheiros, Dileban Karunamoorthy, and Rajkumar Buyya, "Deadline-driven Provisioning of Resources for Scientific Applications in Hybrid Clouds with Aneka," *Future Generation Computer Systems*, 28(1):58 - 65, 2012. Article (CrossRef Link).
[15] RodrigoN. Calheiros and Rajkumar Buyya, "Cost-Effiective Provisioning and Scheduling of Deadline Constrained Applications in Hybrid Clouds," *X.Sean Wang, Isabel Cruz, Alex Delis and Guangyan Huang, editors*, *Web Information Systems Engineering - WISE* 2012, pages 171-184. Springer Berlin Heidelberg, 2012. Article (CrossRef Link).
[16] Alexandru Iosup, Mathieu Jan, Ozan Sonmez, and Dick Epema, "The Characteristics and Performance of Groups of Jobs in Grids," in *Proc. of Euro-Par 2007 Parallel Processing*, pages 382-393. 2007. Article (CrossRef Link).
[17] Tran Ngoc Minh, L.Wolters, and D. Epema, "A Realistic Integrated Model of Parallel System Workloads," in *Proc. of Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 464-473, May 2010. Article (CrossRef Link).
[18] Persistence of Vision Raytracer Pty Ltd, "Persistence of vision raytracer," http://www.povray.org/, 2013.
[19] Stephen Olivier, Jun Huan, Jinze Liu, Jan Prins, James Dinan, P. Sadayappan, and Chau-Wen Tseng, "UTS: An Unbalanced Tree Search Benchmark," in *Proc. of Languages and Compilers for Parallel Computing*, volume 4382 of Lecture Notes in Computer Science, pages 235-250. Springer Berlin Heidelberg, 2007. Article (CrossRef Link).
[20] Stefan Bucur, Vlad Ureche, Cristian Zamfir, and George Candea, "Parallel symbolic execution for automated real-world software testing," in *Proc. of the Sixth Conference on Computer Systems, EuroSys* '11, pages 183-198, New York, NY, USA, 2011. ACM. Article (CrossRef Link).
[21] Duan Li and Xiaoling Sun, "Nonlinear integer programming," volume 84. Springer Science & Business Media, 2006. Article (CrossRef Link).
[22] Wang Zong Jiang and Zheng Qiu Sheng, "A New Task Scheduling Algorithm in Hybrid Cloud Environment," in *Proc. of Cloud and Service Computing (CSC), 2012 International Conference on,* pages 45-49, Nov 2012. Article (CrossRef Link).

[23] Norbert Blum, "A New Approach to Maximum Matching in General Graphs," *MichaelS. Paterson, editor, Automata, Languages and Programming*, volume 443 of Lecture Notes in Computer Science, pages 586-597. Springer Berlin Heidelberg, 1990. Article (CrossRef Link).
[24] Luiz F. Bittencourt and Edmundo R. M. Madeira, "A Performance-oriented Adaptive Scheduler for Dependent Tasks on Grids," *Concurrency and Computation: Practice and Experience*, 20(9):1029-1049, 2008. Article (CrossRef Link).
[25] Javid Taheri, Albert Y. Zomaya, Howard Jay Siegel, and Zahir Tari, "Pareto Frontier for Job Execution and Data Transfer Time in Hybrid Clouds," *Future Generation Computer Systems*, 37:321 - 334, 2014. Article (CrossRef Link).
[26] VA Leena and MS Rajasree, "Genetic Algorithm Based Bi-Objective Task Scheduling in Hybrid Cloud Platform," *International Journal of Computer Theory and Engineering*, 8(1):7, 2016. Article (CrossRef Link).
[27] Wei-Jen Wang, Yue-Shan Chang, Win-Tsung Lo, and Yi-Kang Lee, "Adaptive Scheduling for Parallel Tasks with QoS Satisfaction for Hybrid Cloud Environments," *The Journal of Supercomputing*, 66(2):783-811, 2013. Article (CrossRef Link).
[28] Min-Yi Tsai, Ping-Fang Chiang, Yen-Jan Chang, and Wei-Jen Wang, "Heuristic scheduling strategies for linear-dependent and independent jobs on heterogeneous grids," *Grid and Distributed Computing*, volume 261, pages 496-505. Springer Berlin Heidelberg, 2011. Article (CrossRef Link).
[29] M.R. HoseinyFarahabady, H.R.D. Samani, L.M. Leslie, Young Choon Lee, and A.Y. Zomaya, "Handling Uncertainty: Pareto-Efficient BoT Scheduling on Hybrid Clouds," in *Proc. of Parallel Processing (ICPP), 2013 42nd International Conference on*, pages 419-428, Oct 2013. Article (CrossRef Link).
[30] M.R. HoseinyFarahabady, Young Choon Lee, and A.Y. Zomaya, "Pareto-Optimal Cloud Bursting," *Parallel and Distributed Systems, IEEE Transactions on,* 25(10):2670-2682, Oct 2014. Article (CrossRef Link).
[31] MohammadReza HoseinyFarahabady, YoungChoon Lee, and AlbertY. Zomaya, "Randomized Approximation Scheme for Resource Allocation in Hybrid-cloud Environment," *The Journal of Supercomputing,* 69(2):576-592, 2014. Article (CrossRef Link).
[32] López, José María, José Luis Díaz, and Daniel F. García, "Utilization Bounds for EDF Scheduling on Real-Time Multiprocessor Systems," *Real-Time Systems*, 28(1):39-68, 2004. Article (CrossRef Link).
[33] "Parallel Workloads Archive," http://www.cs.huji.ac.il/labs/parallel/workload/, 2015.
[34] "Amazon Elastic Compute Cloud (Amazon EC2)," http://aws.amazon.com/ec2/, 2015.
[35] L.A. Barroso and U. Holzle, "The Case for Energy-Proportional Computing," *Computer*, 40(12):33-37, Dec 2007. Article (CrossRef Link).
[36] J. Baliga, R.W.A. Ayre, K. Hinton, and RodneyS. Tucker, "Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport," in *Proc. of the IEEE*, 99(1):149-167, Jan 2011. Article (CrossRef Link).

Bo Wang received the BS degree in computer science and technology from Northeast Forestry University (NEFU), China, in 2010. He is currently working toward the PhD degree in computer science at Xi'an Jiaotong University (XJTU), China. His research interests include performance modeling, large-scale distributed systems, cloud computing and datacenter, and virtualization technology.

Ying Song received the PhD degree in computer engineering from Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS). She is an Assistant Professor in ICT, CAS. With main research interests in computer architecture, parallel and distributed computing, virtualization technology and big data technologies, her work thus far has covered topics such as: performance modeling, resource management and cloud computing. She has authored or coauthored more than 10 publications in these areas since 2007; served in various academic conferences.

Yunzhong Sun received the PhD degree in computer engineering from Institute of Computing Technology (ICT), Chinese Academy of Sciences. He is a professor in the State Key Laboratory of Computer Architecture at ICT. His research interests focus on distributed system software and computing/programming models. He has authored and co-authored more than 50 publications, and he has served in various academic conferences and journals. He is a member of the IEEE and the IEEE Computer Society.

Jun Liu received the BS, MS, and PhD degrees from Xi'an Jiaotong University, China, in 1995, 1998, and 2004, respectively, all in computer science. He is currently a professor in the Department of Computer Science, Xi'an Jiaotong University. His main research interests include text mining, data mining and e-learning. He has published more than 40 research papers in various journals and conference proceedings.