J Supercomput
DOI 10.1007/s11227-015-1567-9

An online greedy allocation of VMs with non-increasing reservations in clouds

Xiaohong Wu[1,2] · **Yonggen Gu**[2] · **Jie Tao**[2] · **Guoqiang Li**[3] · **Prem Prakash Jayaraman**[4] · **Daniel Sun**[5] · **Rajiv Ranjan**[6] · **Albert Zomaya**[7] · **Jingti Han**[1]

Abstract Dynamic VMs allocation plays an important role in resource allocation of cloud computing. In general, a cloud provider needs both to maximize the efficiency of resource and to improve the satisfaction of in-house users simultaneously. However, industrial experience has often shown only maximizing the efficiency of resources

✉ Guoqiang Li
li.g@sjtu.edu.cn

Xiaohong Wu
xhwu@hutc.zj.cn

Yonggen Gu
gyg68@hutc.zj.cn

Jie Tao
taojie@hutc.zj.cn

Prem Prakash Jayaraman
prem.jayaraman@rmit.edu.au

Daniel Sun
daniel.sun@rmit.edu.au

Rajiv Ranjan
rranjans@gmail.com

Albert Zomaya
albert.zomaya@sydney.edu.au

Jingti Han
hanjt@mail.shufe.edu.cn

1 Shanghai University of Finance and Economics, Shanghai, China

2 Huzhou University, Huzhou, China

3 Shanghai Jiao Tong University, Shanghai, China

4 RMIT Universtiy, Melbourne, VIC 3000, Australia

Published online: 28 November 2015

and providing poor or little service guarantee for users. In this paper, we propose a novel model-free virtual machine allocation, which is characterized by an online greedy algorithm with reservation of virtual machines, and is named OGAWR. We couple the greedy allocation algorithm with non-increasing reserving algorithms to deal with flexible jobs and inflexible jobs. With the OGAWR, users are incentivized to be truthful not only about their valuations, but also about their arrival, departure and the characters of jobs (flexible or inflexible). We simulated the proposed OGAWR using data from RICC. The results show that OGAWR can lead to high social welfare and high percentage of served users, compared with another mechanism that adopts the same method of allocation and reservation for all jobs. The results also prove that the OGAWR is an appropriate market-based model for VMs allocation because it works better for allocation efficiency and served users.

Keywords Cloud computing · Greedy allocation · Resource reservation · Online algorithm · Incentive compatible

1 Introduction

As the development of cloud computing technology, infrastructure-as-a-service has gained popularity in recent years due to its flexibility, scalability and reliability. A cloud provider needs to maximize the efficiency of resource allocation and meanwhile to improve the satisfaction of in-house users, instead of only to maximize the revenue of cloud provider. The efficiency of resource allocation can be described by the sum of all values of users, called social welfare in this paper. Thus, an IaaS cloud provider has to find an optimal resource allocation for all users.

To address above concerns, VMs allocation is applicable for cloud providers. There are mainly two kinds of allocation methods in public clouds: pay-as-you-go [3] and bid-based allocation [2]. Pay-as-you-go is a first-come first-served allocation mechanism which does not concern about the valuation of users for VMs. In fact, the efficiency of an allocation can be improved if the cloud allocates VMs to users with higher valuation by knowing user-centric valuation. Amazon has used bid-based mechanism in spot instance market to make up for this shortcoming, where users periodically submit bids to the provider, who in turn posts a series of spot prices. Users only hold resource access before the spot price rises above their bids. As a result, due to the dynamic changing of the spot price, it provides no service guarantee to those jobs which should be completed by VMs during multiple time units. To guarantee the service, users can choose subscription scheme. A subscription user always holds resource access during his subscription period. In spite of the fact that the user does not need the resource at some time units, he still has to pay a one-time subscription fee in subscription

[5] NICTA, Sydney, Australia

[6] Newcastle University, Tyne and Wear NE1 7RU, United Kingdom

[7] The University of Sydney, New South Wales 2006, Australia

period. Moreover, some other users might be rejected because they do not subscribe the resources while some reserved resources might be idle. All of these can decrease the efficiency of resources and are often adopted in public clouds.

In this paper, we propose an online greedy allocation with reservation, named OGAWR, for auction-driven clouds. This work includes a formal model for studying VMs allocation under an assumption that the users behave for their own self-interest, and a rationality-based approach using notions from algorithmic game theory [21] and particularly online game. OGAWR has three characteristics: first, the auction is carried out in each time unit as long as users come, while the auction in spot instance is carried out in each period that includes multiple time units; second, OGAWR provides service guarantees, that is, each job which should be processed during multiple time units will not be terminated until it is completed; third, we use different allocation methods of VMs for the flexible jobs and inflexible jobs. Here, flexible job refers to those jobs that users only care about whether they could be completed before their deadlines, while the process details are ignored. For example, a finance firm has to process the daily stock exchange data to guarantee the trading in next day. Obviously, the finance firm only cares about whether the jobs can be finished before the deadlines and does not care about how the jobs are processed. In contrast, inflexible job refers to those jobs that must be processed continuously when they start to be processed.

The contributions in this papers are as follows: first, we formalize the notions of online mechanism design for VMs allocation in cloud environments; second, we present an online greedy allocation with reservation (OGAWR) mechanism, which adopts different VMs reservation methods for flexible jobs and inflexible jobs. To improve the allocation efficiency, we propose two kinds of non-increasing reserving methods: discontinuous reserving based on reservation ratio and continuous reserving. Furthermore, we design a payment algorithm which ensures the truthfulness of the users together with the allocation and reservation policy. Third, we simulate the proposed OGAWR mechanism using data from RIKEN integrated cluster of clusters (RICC), comparing the mechanism which adopts the same method of allocation and reservation for all jobs, this mechanism can lead to higher social welfare and percentage of served user. This paper is an extension of our previous work [32].

The rest of paper is organized as follows. In Sect. 2 we discuss related work. After formalizing the problem model in Sect. 3, in Sect. 4 we design an online greedy allocation with reservation(OGAWR) mechanism and analyze the properties of OGAWR mechanism in Sect. 5. The performance evaluation are shown in Sect. 6. Finally, conclusions appear in Sect. 7.

2 Related work

Resource management and allocation have been considered an important problem in parallel and distributed systems for data-intensive applications, energy efficiency, massive simulations and so on [17–19,29]. The provision and allocation of VMs in cloud computing are closely related to the utility of both cloud users and service providers. There are mainly two types of work. One investigates the VMs allocation by solving an optimization problem [14,33,37]. This type focuses on the optimization of object functions, but generally without considering any strategic behaviours among users (e.g.the VM allocation approach for spot markets in paper [37]). The other is game theoretical approach. For instance, a cloud resource allocation approach assumes that the allocation would start after all users submit their requests. An alternative of game theoretical method is to design pricing mechanisms with maximized social welfare or profit [22,38]. Auction-driven mechanisms can combine the two approaches and simultaneously target both truthfulness and economic efficiency.

Classic applications of auctions can be found in a wide range of research areas, such as network bandwidth allocation [31] and wireless spectrum allocation [39]. Auctions have also been widely studied for scheduling and resource allocation [1,6,9,30]. Recently, a series of auction mechanisms are designed for VMs allocation in cloud computing. Wang et al. [28] apply the critical value method, and derive a mechanism that is collusion-resistant, an important property in practice. Their algorithm has a competitive ratio $O(\sqrt{k})$, where k is the number of VM instances. In addition, combinatorial auctions are supposed to apply in VMs allocation in some literatures [5,35]. Yet all these works only consider resource allocations in one time unit and restrict their discussions in a single offline auction period. However, in cloud computing, the cloud users arrive and leave randomly, so the statistic analysis and design-based game theory are not suitable for it.

Online mechanism is an important expansion of mechanism theory in the multi-agent and economics literature, and generally applied in dynamic environment, in which user population is dynamically changing, decisions must be made across time and there can be uncertainty about the set of feasible decisions in future periods [23]. That is consistent with the environment features in cloud computing. Consequently, some online methods used in other fields [4,26] are also applicable in cloud computing [13,38].

In online mechanism, Lavi and Nisan [16] coined the term online auction and initiated the study of incentive compatible mechanisms in dynamic environments. Friedman and Park [8] initiated the study of VCG-based online mechanisms and coined the term online mechanism design. According to the research on online mechanism, there are two frameworks of research. One is model-based approach which aims at developing online variants of Vickrey–Clarke–Groves (VCG) mechanisms [11,24]. These techniques rely on a model of future availability, as well as future supply (e.g., Parkes and Singh [24] used an MDP-type framework for predicting future arrivals). The other is model-free approach which requires fewer assumptions and makes computing allocations more tractable than the above [10,12,25].

The online mechanisms have been used in cloud computing [15,27,34,36]. [27,36] only introduce an online mechanism framework for cloud resources allocation without concrete allocation algorithms. The online mechanism in [15] is a resource allocation approach for batch jobs, and the value functions for users is continuous. Zaman et al. [34] designed a truthful mechanism that allocates the VMs to users by greedy allocation and allows those allocated users continuously use those VMs for the entire period requested. Ma et al. [20] focuses on designing dominant-strategy incentive compatible (DSIC) mechanisms with good competitive ratios, assuming that agents will not misreport their arrival times and deadlines.

Based on those work in [15,34,36], we also aim to design an online truthful mechanism for VMs allocation, but pay attention to the following points:

1. In our model, a user requests one VM for multiple time units to finish a job during the arrival-departure interval. According to the demand in process time, all jobs are classified into two classes: flexible jobs and inflexible jobs.
2. We choose different allocation methods for the two classes of jobs, especially design a discontinuous resource allocation based on reservation-ratio for the flexible jobs, by which the distribution of workload of users can be adjusted at their arrival–departure interval and the total workloads processed in cloud will be improved.
3. We focus on all the users with single-valued preference. That is, each user could get a non-zero constant value brought by the job only if it could be finished completely.

3 Modeling and notations

We consider a cloud provider who provides only one type of VM instances, and the total number of VM instances is denoted by C. Consider discrete time periods $T = 1, 2, \ldots$, indexed by t and possibly infinite.

Each user is represented by an agent. Let $N = 1, 2, \ldots, n$ denote the set of all agents. Agent submits its job to the cloud randomly, which can be characterized by the type $\theta_i = (a_i, d_i, l_i, e_i, V_i) \in \Theta_i$, where Θ_i is its type space. Here, a_i and d_i present the arrival and departure time of agent i, and l_i is its total computation workload, i.e, the job size. We assume that each agent requires at most one VM in each unit time. The workload l_i is the number of time units that agent i must be allocated one VM, so it also represents the resource demand. The last component of θ_i is V_i, the value agent i obtains if its job is completed, and $V_i \geq 0$.

As described in Sect. 1, the jobs are classified into flexible jobs and inflexible jobs. In order to distinguish the job classes, a character parameter e_i points out the agent is flexible ($e_i = 1$) or inflexible ($e_i = 0$).

We define $\pi_i = (\pi_i^{a_i}, \pi_i^{a_i+1}, \ldots, \pi_i^{d_i})$ as the allocation for agent i. $\pi_i^t = 1$ if agent i is allocated one VM at time $t \in [a_i, d_i]$, otherwise $\pi_i^t = 0$.

The allocation result for agent i is denoted by A_i.

$$A_i = \begin{cases} 1 & \text{if} \quad \pi_i^t \leq 1 \text{ and } \Sigma_t \pi_i^t \geq l_i, t \in [a_i, d_i] \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Each agent i is characterized by a valuation function v_i defined as follows:

$$v_i = \begin{cases} V_i & \text{if} \quad A_i = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The challenge of the cloud provider is to make allocation decisions π^t dynamically while trying to maximize the sum of agents value. The problem is described as follows:

Table 1 Symbols definition

Symbol	Definition
a_i	Arrival time of agent i
d_i	Departure time of agent i
l_i	Job size of agent i
e_i	Character parameter of agent i
θ_i	Type of agent, $\theta_i = (a_i, d_i, l_i, e_i, V_i)$
θ	Type set of all agents, $\theta_i \in \theta$
θ^t	Type set of agents which participate the allocation at t
π_i	Allocation of agent i, $\pi_i = \left(\pi_i^{a_i}, \pi_i^{a_i+1}, \ldots, \pi_i^{d_i}\right)$
π^t	Allocation decision at time t, $i \in \pi^t$ if i is allocated at t
π_{-i}^t	Allocation decision at time t if agent i were not present before time t (including time t)
A	Allocation results, $A = (A_1, A_2, \ldots, A_n)$
A_i	Allocation result of agent i. If i is allocated, $A_i = 1$
S^t	Supply about future period, $S^t = (s(t), s(t+1), \ldots)$
S_{-i}^t	Supply about future period if agent i were not present before time t (including time t)
$s(t)$	Supply at time t
$r(t)$	Reservation-ratio at time t, $r(t) = s(t)/C$
p_i	Payment of agent i

$$\begin{aligned} \max \quad & \Sigma_{i\in N} v_i \\ \text{s.t.} \quad & \pi_i^t \le 1, \forall t \in T \\ & \Sigma_{i=1}^n \pi_i^t \le C, \forall t \in T \end{aligned} \tag{3}$$

Table 1 lists main symbols defined in this paper.

4 The online greedy allocation with reservation mechanism

4.1 Description of mechanism

In this section we design a model-free online mechanism for the above setting.

Denote the number of idle VMs at time t by $s(t)$. The definition of greedy allocation is as follows.

Definition 4.1 (*Greedy allocation*) At each step t allocate the $s(t)$ VMs to the active agents with the highest valuations.

If all agents request one VM only for one time unit, greedy allocation with appropriate payment could constitute a truthful mechanism [21]. However, in the case of multiple time units demands, according to Eq. (2), whether agent i could get the value V_i is decided by all of its allocation in period $[a_i, d_i]$. That is, the value brought by one

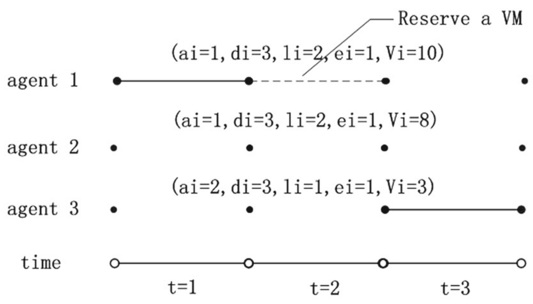
Reserve a VM
(ai=1, di=3, li=2, ei=1, Vi=10)
agent 1
(ai=1, di=3, li=2, ei=1, Vi=8)
agent 2
(ai=2, di=3, li=1, ei=1, Vi=3)
agent 3
time
t=1
t=2
t=3

Fig. 1 An example for multi-time unit demand ($C = 1$)

VM at some time unit cannot be decided at first, so greedy allocation for each time unit can not be performed. In order to maintain incentive compatibility, we extend the greedy allocation policy by allowing the system to reserve VMs for agents. By such allocation approach, the agent is not only allocated one VM at current time t but also reserved VMs in future time units. Define unit valuation as the valuation of one VM per unit time, and it is expressed as V_i/l_i for agent i.

Definition 4.2 [*Online greedy allocation with reservation* (*OGAWR*)] At each step t allocate the $s(t)$ VMs to the active agents with the highest unit valuations, at the same time, make the VM reservation for allocated agent i during period $[t+1, d_i]$ if $l_i > 1$.

Consider an example with 3 time units and 3 agents in Fig. 1, where $\theta_1 = (1, 3, 2, 1, 10), \theta_2 = (1, 3, 2, 1, 8), \theta_3 = (2, 3, 1, 1, 3)$ showing the agents arrival, departure, job size, job class and valuation. Suppose furthermore that $C = 1$. Sort the agents by their unit valuation V_i/l_i. Because agent 1 has the highest unit valuation at time 1, OGAWR method would allocate the VM to agent 1 and reserve one VM for it. Since it is a flexible job ($e_1 = 1$), there are two choices for reserving: reserving at time 2 or at time 3. In Fig. 1, the VM in time unit 2 is chosen to reserve for agent 1, so there is no idle VM to be allocated at time 2. At time 3, although agent 2 has higher unit valuation than agent 3, the VM is still allocated to agent 3, because agent 2 has no sufficient time to finish the job at that time.

It is worth to note that OGAWR might not be performed in some cases. That is, an agent with highest unit valuation cannot be allocated although there is sufficient time to finish the job. In the above example, if the VM in time unit 3 is reserved for agent 1 at time 1. At time 2, although agent 2 has higher unit valuation than agent 3 and there is sufficient time to process, agent 2 still cannot be allocated. The reason for this result is that the supply in future is less than that of current time. Therefore, the OGAWR can be realized only if it makes 'non-increasing reserving'.

Definition 4.3 (*non-increasing reserving*) Non-increasing reserving refers to a class of reserving schemes which always satisfies $s(t) \leq s(t+1) \leq s(t+2) \leq \cdots$ after allocation at each time t.

In OGAWR mechanism, the agent participating the allocation at time t satisfies three conditions: (1) it arrives before time t. (2) Its departure time is longer than $t + l_i - 1$. (3) It is still unallocated. The OGAWR mechanism consists of allocation rule and payment rule described as follows.

- *Allocation rule*
  At each time t, it makes allocation as follows.
  Stage 1 *Greedy allocation* Allocate the $s(t)$ VMs using greedy allocation, breaking ties at random.
  Stage 2 *Non-increasing reservation* Make non-increasing reservation for agents who are allocated in stage 1 if necessary. If one VM is reserved for agent i at time k, $\pi_i^k = 1$.
  Let $\theta^t = (\theta_1, \theta_2, \ldots, \theta_n)$ denote the set of agent types participating the allocation at time t, and π^t denotes the decision policy at time t. The mechanism makes a sequence of allocation decisions $(\pi^1, \pi^2, \ldots)$, and π^t includes all those agents allocated at time t. For example, in Fig. 1, $\pi_1 = (1, 1, 0)$, $\pi^1 = \{1\}$ and $A_1 = 1$ after allocation at time 1, and it is not changed at time 2. After allocation at time 3, $\pi_3 = (0, 1)$, $\pi^3 = \{3\}$, $A_3 = 1$.
- *Payment rule*
  We design a critical payment which is equal to the critical value for allocated agents, and the definition of critical value is as follows.
  Given type $\theta_i = (a_i, d_i, l_i, e_i, V_i)$, the critical value for agent i is defined as

$$V^c_{(a_i,d_i,l_i,e_i)}(\theta_{-i}) = \begin{cases} \min V_i' & \text{s.t. } A_i(\theta_i', \theta_{-i}) = 1, \\ & \quad \text{for } \theta_i' = (a_i, d_i, V_i') \\ \infty & \text{no such } V_i' \text{ exists} \end{cases} \tag{4}$$

  where $\theta_{-i} = (\theta_1, \theta_2, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots)$.
  We define payment policy $p_i(\theta)$ as

$$p_i(\theta) = \begin{cases} V^c_{(a_i,d_i,l_i,e_i)}(\theta_{-i}) & \text{if } A_i = 1 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

4.2 The algorithm design of OGAWR mechanism

In this section, the algorithm based on the proposed rules for allocation and payment is designed. First, we introduce two reserving methods for inflexible agents and flexible agents respectively.

Continuous reserving Continuous reserving is suitable for inflexible agents, which is similar to the allocation in MOVMPA mechanism proposed in paper [34]. If agent i wins the auction at time t, one VM will be reserved continuously for it in next $l_i - 1$ units. That is, $\pi_i^k = 1, k = t + 1, \ldots, t + l_i - 1$, if $\pi_i^t = 1$.

Discontinuous reserving based on reservation-ratio (discontinuous reserving) This reserving method reserves one VM for agent i in next $l_i - 1$ time units with lowest reservation-ratio, and reserves the VM in earliest time unit if there are multiple time units with same reservation ratio.

Reservation ratio denoted by $r(k)$ is the ratio of the number of reserved VMs and total supply capacity C at future time k expressed as $r(k) = s(k)/C$. Obviously, $r(k)$ is changed with time.

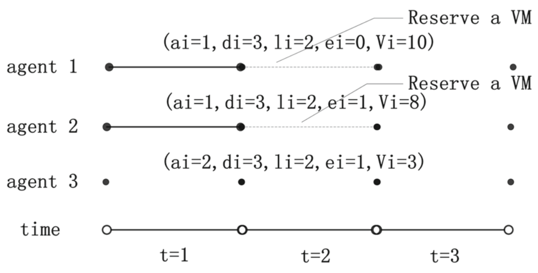
Reserve a VM
(ai=1, di=3, li=2, ei=0, Vi=10)
agent 1
Reserve a VM
(ai=1, di=3, li=2, ei=1, Vi=8)
agent 2
(ai=2, di=3, li=2, ei=1, Vi=3)
agent 3
time
t=1
t=2
t=3

Fig. 2 An example for continuous allocation ($C = 2$)

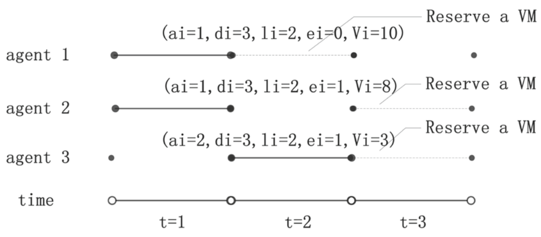
Reserve a VM
(ai=1, di=3, li=2, ei=0, Vi=10)
agent 1
(ai=1, di=3, li=2, ei=1, Vi=8)
Reserve a VM
agent 2
Reserve a VM
(ai=2, di=3, li=2, ei=1, Vi=3)
agent 3
time
t=1
t=2
t=3

Fig. 3 An example for noncontinuous allocation based on reservation-ratio ($C = 2$)

For inflexible agents, continuous reserving and discontinuous reserving both could be used. In the next example, we compare the allocation results by using above two reserving methods respectively. Consider the example with 3 time units and 3 agents in Figs. 2 and 3, where $\theta_1 = (1, 3, 2, 0, 10)$, $\theta_2 = (1, 3, 2, 1, 8)$, $\theta_3 = (2, 3, 2, 1, 3)$, and $C = 2$. Figure 2 shows the results by continuous reserving for all agents. At time 1, agent 1 and agent 2 are allocated and reserved VMs continuously. At time 2 there is no idle VMs because two VMs are both reserved at time 1, so agent 3 cannot be allocated. At time 3, agent 3 still cannot be allocated because it has no sufficient time from departure to complete. Figure 3 shows another results by discontinuous reserving. At time 1, agent 1 and agent 2 all be allocated one VM. Then it reserves VM at time 2 for agent 1 and reserves VM at time 3 for agent 2 based on reservation-ratio. We can see all agents could be allocated by this method, and the sum of users value is higher than that of continuous reserving. In our mechanism, we choose discontinuous reserving for inflexible agents. At each time unit t, the steps of allocation shown by Algorithm 1 are as follows:

Step 1 Sort all agents which participate the allocation at time t in non-increasing order of V_i/l_i.

Step 2 Allocate $s(t)$ idle VMs to $s(t)$ agents with highest valuation, breaking ties at random.

Step 3 Choose a suitable reserving method for each agent allocated at step 2. Continuous reserving is chosen if $e_i = 0$ and discontinuous reserving is chosen if $e_i = 1$.

Define a status vector $S^t = (s(t), s(t+1), s(t+2), \ldots, s(t+m-1))$ as the VM supplies in period $[t, t+m-1]$ before allocation at time t, where $s(t+k)$ is denoted as the supply at future time $(t+k) \in T$, and m satisfies $s(t+m-1) < C$ and $s(t+m) = C$. For computing critical value, we define $v_{-i,t}^{(m)}$ to be the mth highest of unit valuations V_j/l_j from all agents j in θ^t, $j \neq i$. Then $v_{-i,t}^{s(t)}$, for supply $s(t)$, is the

```
input  : $\theta^t, S^t, t$
output: $S^t, \pi^t, A$
if $s(t) > 0$ then
    sort all $\theta^t \subseteq \Theta$ in non-increasing order of $V_i/l_i$;
    $(\pi^t, A) = greedyallocate(\theta^t, S^t)$;
    sort all $i \in \pi^t$ in non-decreasing order of $d_i$;
    foreach $i \in \pi^t$ do
        if $e_i = 1$ then
            $(\pi_i, S^t) = DiscontinuousReserve(l_i - 1)$;
        end
        else
            $(\pi_i, S^t) = ContinuousReserve(l_i - 1)$;
        end
    end
end
$S^{t+1} \leftarrow S^t \setminus \{s(t)\}$;
```
Algorithm 1: Allocation algorithm: *Allocate*

lowest value that is still allocated a unit, if agent i were not present not only at current t but also before t. Henceforth, we refer to $v_{-i,t}^{(s(t))}$ as the marginal clearing value of the idle VM for agent i at time t.

OGAWR Mechanism designed in Algorithm 2 runs in each time unit t, the algorithm is described as follows:

Step 1 According to the allocation rule of OGAWR, the allocation is performed based on S^t, θ^t, which generates an allocation set π^t and a new status set S^{t+1}, and updates allocation result A.

Step 2 For each agent i who got its first unit at step 1, the critical value for agent i at time t is computed as equation $v_{i,t}^c = v_{-i,t}^{s(t)}$. Then, we execute a suppositional allocation in which i is not present and get suppositional results π_{-i}^t and S_{-i}^{t+1}.

Step 3 For each i who got the allocation before time t and $t \leq d_i - l_i + 1$, according to the suppositional result S_{-i}^t which suppose that i had been not present before t, we compute the critical value for agent i as $v_{i,t}^c = v_{-i,t}^{s_{-i}(t)}$. We also execute a suppositional allocation in which i is not present based on the suppositional status S_{-i}^t and get suppositional results π_{-i}^t and S_{-i}^{t+1}.

Step 4 For each i who satisfies $t = d_i - l_i$, the payment p_i is computed. If $A_i = 0$, the payment p_i is zero, otherwise the payment can be computed as

$$p_i = \left(\min_{t \in [a_i, d_i - l_i + 1]} v_{i,t}^c \right) \cdot l_i$$

Lemma 4.1 *The payment in above algorithm is a critical payment. That is,* $(\min_{t \in (a_i, d_i - l_i + 1)} v_{i,t}^c) \cdot l_i = V_{(a_i, d_i, V_i, l_i, e_i)}^c(\theta_{-i})$ *for each allocated agent i.*

Proof Follows directly from the definition of critical value, expressed by Eq. (4). □

```
input  : $t, \theta^t = \{\theta_1, \theta_2, \ldots, \theta_n\}, S^t$
output: $S^{t+1}, \pi^t, p_i, A$
$\pi^t = \emptyset$;
$(S^{t+1}, \pi^t, A)$=$Allocate(\theta^t, S^t, t)$;
foreach $\pi_i \in \pi^t$ do
    $v^c_{i,t} = v^{s(t)}_{-i,t}$;
    $(S^{t+1}_{-i}, \pi^t_{-i}, A_{-i})$=$Allocate(\theta^t_{-i}, S^t, t)$;
end
foreach $i \notin \pi^t$ and $A_i = 1$ and $t \leq d_i - l_i$ do
    $v^c_{i,t} = v^{s_{-i}(t)}_{-i,t}$;
    $(S^{t+1}_{-i}, \pi^t_{-i}, A_{-i}) = Allocate(\theta^t_{-i}, S^t_{-i}, t)$;
end
foreach $i: t = d_i - l_i + 1$ do
    if $A_i = 0$ then $p_i = 0$ else $p_i = \min_{t \in [a_i, d_i - l_i + 1]} v^c_{i,t} \cdot l_i$
end
```
Algorithm 2: Mechanism algorithm: OGAWR

5 Analysis of OGAWR mechanism

We assume no early-arrival no late-departure misreports with $a_i \leq a'_i \leq d'_i \leq d_i$, because generally agent i does not know its type until a_i and the value of agent will be zero if it is finished after d_i. We also assume no less job size misreports with $l'_i \geq l_i$ because the agent i will have no sufficient time to process if $l'_i < l_i$.

Definition 5.1 (*Monotonic with resource demand*) An allocation policy is monotonic with resource demand l_i if for any arrival-departure interval $[a_i, d_i]$, any valuation V_i and any job size report $l'_i \geq l_i$, we have $A_i(a_i, d_i, V_i, l'_i) = 1 \Rightarrow A_i(a_i, d_i, V_i, l_i) = 1$.

Definition 5.2 (*Monotonic with arrival-departure interval*) An allocation policy is monotonic with arrival-departure time if for any job size l_i, any valuation V_i and any arrival-departure time report $a'_i \geq a_i$ and $d'_i \leq d_i$, we have $A_i(a'_i, d'_i, V_i, l_i) = 1 \Rightarrow A_i(a_i, d_i, V_i, l_i) = 1$.

Lemma 5.1 *The allocation policy in OGAWR mechanism is monotonic with resource demand and arrival–departure interval.*

Proof First, we prove the monotonicity with resource demand l_i.

Assume for contradiction that $l'_i \geq l_i$ and $A_i(a_i, d_i, l'_i) = 1$, but $A_i(a_i, d_i, l_i) = 0$. Because $A_i(a_i, d_i, V_i, l'_i) = 1$, it must satisfy $V_i \geq V^c_{(a_i,d_i,V_i,l'_i,e_i)}(\theta_{-i})$. Since $V^c_{(a_i,d_i,V_i,l'_i,e_i)}(\theta_{-i}) = (\min_{t\in[a_i,d_i-l'_i+1]} v^c_{i,t}) \cdot l'_i$, it follows that $V_i/l'_i \geq \min_{t\in[a_i,d_i-l'_i+1]} v^c_{i,t}$. On the other side, because $A_i(a_i, d_i, V_i, l_i) = 0$, it must satisfy

$$V_i < V^c_{(a_i,d_i,V_i,l_i,e_i)}(\theta_{-i}) = \left(\min_{t\in[a_i,d_i-l_i+1]} v^c_{i,t}\right) \cdot l_i.$$

Thus it follows that $V_i/l_i < \min_{t\in[a_i,d_i-l_i+1]} v^c_{i,t}$.

Since $[a_i, d_i - l'_i + 1] \subseteq [a_i, d_i - l_i + 1]$ for $l'_i \geq l_i$,

$$\min_{t \in [a_i, d_i - l_i + 1]} v^c_{i,t} \leq \min_{t \in [a_i, d_i - l'_i + 1]} v^c_{i,t}.$$

So,

$$V_i / l_i < \min_{t \in [a_i, d_i - l_i + 1]} v^c_{i,t} \leq \min_{t \in [a_i, d_i - l'_i + 1]} v^c_{i,t} \leq V_i / l'_i.$$

Then it must satisfy $l'_i < l_i$. That is a contradiction with assumption.

Second, we prove the monotonicity on arrival-departure interval. we say that an arrival-departure interval $[a_i, d_i]$ is tighter than $[a'_i, d'_i]$ if $a'_i \geq a_i$ and $d'_i \leq d_i$.

Also assume for contradiction that $[a_i, d_i]$ is tighter than $[a'_i, d'_i]$, and $A_i(a'_i, d'_i, V_i, l_i) = 1$, but $A_i(a_i, d_i, V_i, l_i) = 0$. Because $A_i(a'_i, d'_i, V_i, l'_i) = 1$, $V_i \geq V^c_{(a'_i, d'_i, V_i, l_i, e_i)}(\theta_{-i})$. According to the computation of critical value,

$$V^c_{(a_i, d_i, V_i, l_i, e_i)}(\theta_{-i}) \leq V^c_{(a'_i, d'_i, V_i, l_i, e_i)}(\theta_{-i}),$$

so $A_i(a_i, d_i, V_i, l_i) = 1$ and a contradiction with the assumption. □

Next, we discuss whether an agent's utility is improved by misreporting e_i or not. First, an inflexible agent would not misreport $e_i = 1$ because discontinuous allocation for this class job will cause zero value. Second, we find there is no difference in allocation and payment to flexible agent between reporting $e_i = 1$ and reporting $e_i = 0$. For allocation, due to the greedy allocation and non-increasing reserving, whether an agent can be allocated is only decided by the order of its valuation and not related with e_i. For payment, according to the critical value Eq. (4), the critical value of agent i would not changes when e_i changes, and the payment of the agent is equal to the critical value which is also not related to e_i. We assume that each agent is rational, that is, the agent will choose to report true type when misreport cannot improve its utility.

Theorem 5.1 *The OGAWR mechanism is incentive compatible with no-early arrival, no-late departure misreports and no less job size misreports.*

According to the Lemma 5.1, given valuation report V_i, the allocation policy is monotonic with resource demand and arrival–departure time, so agents will report true arrival–departure interval and job size with no-early arrival, no-late departure misreports and no less job size misreports. Assume agent i is truthful. Next, we prove by case analysis that the agent i would get more utility if it is truthful to report the valuation V_i. (a) If agent i is not allocated, $V^c_{(a_i, d_i, V_i, l_i, e_i)}(\theta_{-i}) > V_i$ and to be allocated, the agent must report some $V'_i > V_i$. But since the critical value is greater than the true valuation V_i, it will cause to negative utility if i wins for V'_i.

$$u_i = V_i - V^c_{(a_i, d_i, V_i, l_i, e_i)}(\theta_{-i}) < 0.$$

(b) If agent i is allocated, the utility is nonnegative since $V^c_{(a_i,d_i,V_i,l_i,e_i)}(\theta_{-i}) \leq V_i$, and agent i does not want to report a lower valuation V'_i for which it would not be allocated. Consider any V'_i for which agent i continue to allocated. The critical value is not changed since it is independent of the reported valuation V'_i. Therefore, OGAWR is incentive compatible.

5.1 Competitive analysis

We define the competitive ratio on social welfare as follows. An auction mechanism Misc-competitive with respect to the social welfare if for every bidding sequence θ, $E_M(\theta) \geq E_{\text{opt}}(\theta)/c$. Accordingly, c is the competitive ratio of M. Where, E_M is the sum of agents value in mechanism M and E_{opt} denotes the sum of agents value by the optimal algorithm.

Assume that VM to all agents has a same maximal unit valuation $v_{\max}$ and same minimal unit valuation $v_{\min}$, i.e, $v_i \in [v_{\min}, v_{\max}]$. Define $N = \frac{v_{\max}}{v_{\min}}$. At the same time, we assume the maximal job size is L and $L \geq 2$.

For example, consider the case $C = 1$, and three bids: $\theta_1 = (1, L+1, 2, 1, 2(v_{\min}+\epsilon))$, $\theta_2 = (2, L+1, L, 1, L \cdot v_{\max})$ and $\theta_3 = (1, 1, 1, 1, v_{\min})$, where ε is a positive quantity, and $v_{\min} - \epsilon > 0$. In OGAWR agent 1 would be allocated one VM at time 1 and reserved one VM at time 2 because it has highest unit valuation at that time. But this reserving makes agent 2 lose the allocation, although agent 2 has higher unit valuation. Because $l_i = L$ for θ_2, agent 2 should get the allocation at time 2, otherwise it has no sufficient time to process. Therefore, VMs is only allocated to agent 1 and get the total value $2(v_{\min}+\epsilon)$ in OGAWR. On the other hand, an optimum offline algorithm will allocate resources to agent 2 and agent 3 and get the total value $(L \cdot v_{\max} + v_{\min})/(2(v_{\min} + \epsilon))$.

Theorem 5.2 *OGAWR mechanism has a competitive ratio on social welfare* $\frac{N\cdot(L+1)}{2}$ *if* $C = 1$ *and* $L > 1$.

Proof Consider a period of time $[t, t + L] \in T$. Obviously, the potential maximal value of allocation is $v_{\max} \cdot L$. Now, we consider the potential worst case with minimal value of allocation. The worst case is that the VM during L time units is only reserved in one time unit and due to the reserving VMs during other $L - 1$ units could not be allocated. The potential minimal value of allocated agents in $[t, t + L]$ is $v_{\min}$. Moreover, if the minimal valuation is $v_{\min}$, it must exits another time unit t_1 before time t that the agent gets allocation and reservation by bids $v_{\min}$.

Now, we consider the efficiency during the time period: $t_1 \cup [t, t + L]$. From above analysis, we can compute the minimal value by our allocation in $t_1 \cup [t, t + L]$, i.e.,

$$\min E_{\text{OGAWR}}(\theta) = 2 \cdot v_{\min}.$$

The potential maximal value of allocation in $t_1 \cup [t, t + L]$ is

$$\max E_{\text{opt}}(\theta) = v_{\max} \cdot (L + 1).$$

So, $E_{\mathrm{OGAWR}}(\theta) \geq E_{\mathrm{opt}}(\theta)/c$, where $c = \frac{N \cdot (L+1)}{2}$ $\square$

If $L = 1$, the case is similar to the online auction of expiring items described in [21], and the competitive ratio is 2.

Consider there are C VMs and $L > 1$. We can easily deduce that the competitive ratio is $C \cdot \frac{N \cdot (L+1)}{2}$.

6 Performance evaluation

As analysed above, the competitive ratio c of OGAWR mechanism might be very large because it is decided by L and N. That is, it may cause to very low social welfare at the worst case. In this section, we will present the simulation results and compare the OGAWR mechanism with two allocation methods. One is a good online mechanism (MOVMPA) designed in paper [34]. The main difference between MOVMPA mechanism and OGAWR mechanism is that MOVMPA uses same continuous allocation for all jobs while OGAWR mechanism adopts different reserving methods for flexible jobs and inflexible jobs. The other method compared is an offline optimal approach designed under the assumption that we know all the agents valuation beforehand and completely ignore the allocation time constraint in $[a_i, d_i]$. Although it is not reasonable that OGAWR mechanism is compared with the offline allocation without time constraint, but we can understand the actual level of proposed mechanism on social welfare and percentage of served agents by comparing the curves.

As same as [34], the input data of the experiments are collected from the parallel workload archive [7], where collect many workload logs from large scale parallel systems in various places around the world. We select 10 thousands continuous records from log RICC-2010-2. In the log, the minimal time unit recorded is one second. In our experiment, we choose 10 min as one time unit, and all records selected are distributed randomly from time 0 to time 8000. Each record in the log corresponds to one task in out experiment, and the information of a task includes arrival time, wait time, runtime, number of allocated processors corresponding to the record information. Each task is processed by at most 8 thousands processors. According to the number of allocated processors k, a task can be divided into k subtasks, each of which need to be processed serially in one processor. That is, one subtask requests at most one VM in each time unit which is consistent with the assumption in our model. Let each agent present one subtask(also is one job). After the step of task decomposition, there are about 285 thousands agents in these records.

Next, the type of each agent θ_i can be obtained, $\theta_i = (a_i, d_i, l_i, e_i, V_i)$. First, a_i and l_i can be got from the log, where the real arrival time of the record is a_i and the runtime can be converted to the size l_i. As described above, if k agents are generated from one same record, they will have same arrival time and job size. Second, we produce the other information d_i and V_i. Assume that the deadline and the valuation are exponential distribution. Deadline d_i and valuation V_i/l_i are computed as $d_i = a_i + l_i + l_i \cdot \exp(d_{\mathrm{avg}})$ and $V_i/l_i = \exp(v_{\mathrm{avg}})$. Finally, the parameter e_i is generated randomly. The Table 2 shows the simulation parameters.

Figures 4 and 5 show the distribution of all those subtasks we selected. Figure 4 shows the number of arrival subtasks at each time unit, while Fig. 5 is the size distrib-

Table 2 Simulation parameters

Type	Notaion	Value	Parameter
Arrival time	a_i	Form workload archive	
Departure time	d_i	$a_i + l_i + l_i \cdot \exp(d_{\text{avg}})$	$d_{\text{avg}} = 2$
Job size	l_i	From workload archive	
Valuation	V_i	$V_i / l_i = \exp(v_{\text{avg}})$	$v_{\text{avg}} = 50$
Job character (flexible)	e_i	1 or 0, generate randomly	

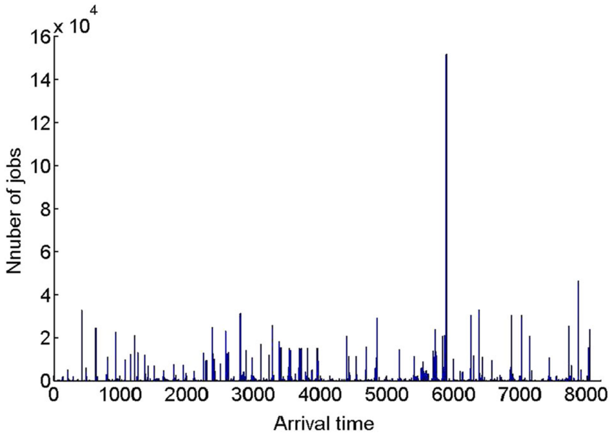

Fig. 4 Distribution of arrival time

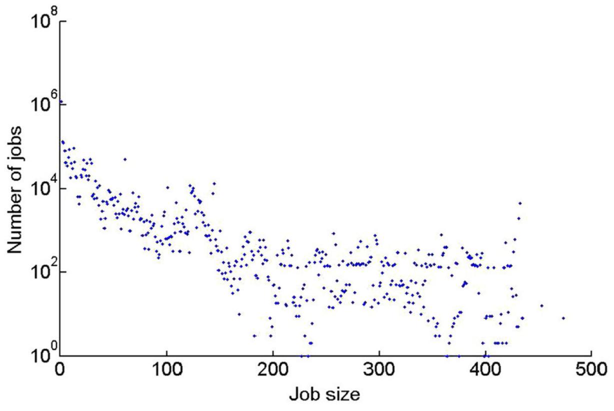

Fig. 5 Distribution of job size

ution of all agents. Before running of the mechanism, we initialize the supply, the total number of VMs, which is closely related to the allocation results. Define an initial supply C_0 is equal to average requirement for each time unit, i.e., $C_0 = \Sigma_{i=1}^{n} l_i / |T|$, where $|T| = 8000$ is total time units we select.

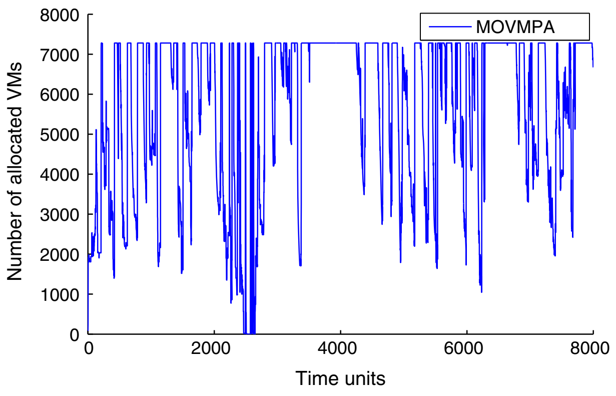

Fig. 6 The distribution of allocated VMs under MOVMPA mechanism

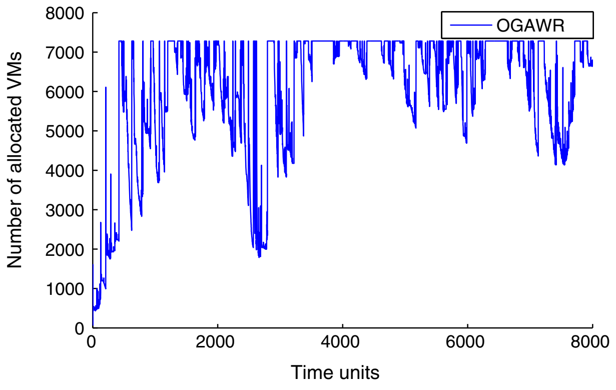

Fig. 7 The distribution of allocated VMs under OGAWR mechanism

We mainly compare the results from three aspects: the percentage of allocated VMs, the allocation efficiency and the percentage of completed jobs.

We compare the allocated VMs distribution under two online mechanisms, MOVMPA and OGAWR with the same input data. Figures 6 and 7 show the distribution of allocated VMs in each time unit with supply capacity $C = 7280$. Comparing the two figures, OGAWR presents higher number of allocated VMs than MOVMPA, because OGAWR can adjust the distribution of demands when the workloads have arrival bursts. To compare it in different capacities, we set the capacity C from 2600 to 7800, that is, C changes from $0.5 \cdot C_0$ to $1.5 \cdot C_0$. Figure 8 shows the percentage of allocated VMs. Obviously, OGAWR has higher VMs utilization rate than MOVMPA in different capacities.

Although OGAWR has higher VMs utilization, how about the efficiency of the allocated VMs? To answer this question, we compare the social welfare under different mechanisms. Figure 9 shows the social welfare, the sum of agents value under different C and C changes from $0.5 \cdot C_0$ to $1.5 \cdot C_0$. First, we note that the trends for the two scenarios are different—when supply is low and close to $0.5 \cdot C_0$, the OGAWR

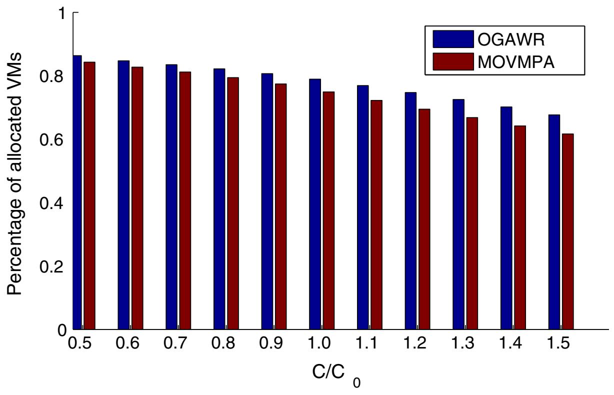

Fig. 8 The percentage of allocated VMs under OGAWR mechanism

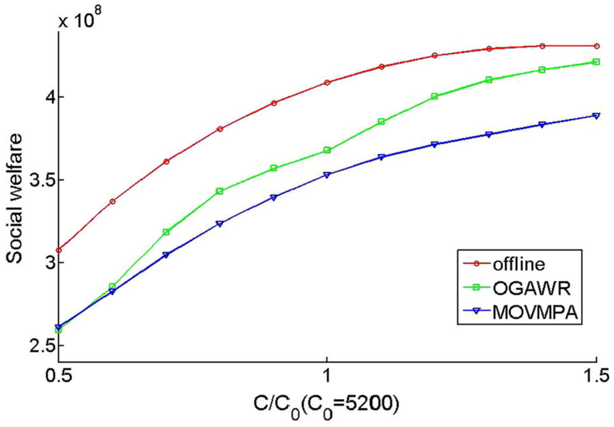

Fig. 9 The social welfare under three mechanisms

mechanism results only in a small overall improvement in social welfare, However, when it grows to more than C_0, there is a very marked improvement. Especially, when $C = 1.5 \cdot C_0$, the social welfare under OGAWR mechanism is very close to it under offline allocation, while it still keep a low level in MOVMPA mechanism.

With respect to satisfaction of in-house users, the number of completed jobs of individual agents is discussed. The results are shown as Fig. 10. The percentage of completed jobs increases with the increase of supply in all allocation approaches, and in OGAWR, it is obviously higher than that in MOVMPA when they are in the same supply capacity.

7 Conclusion

In this study, we address the problem of dynamic VMs allocation, payment determination in IaaS clouds in order to maximize the efficiency of resource and improve

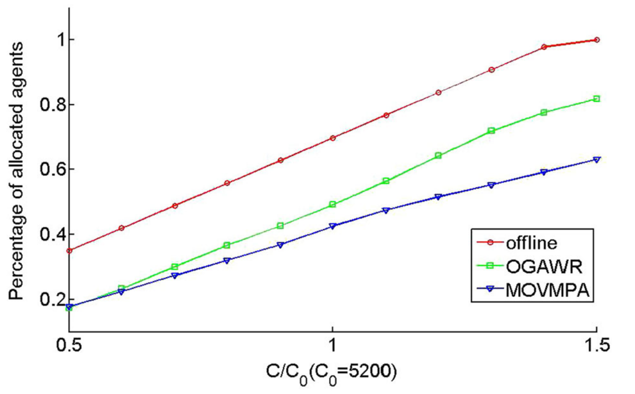

Fig. 10 The percentage of completed jobs under three mechanisms

the satisfaction of users simultaneously. We construct an online resource allocation model in which jobs are divided into two classes: inflexible jobs and flexible jobs. Then, an online greedy allocation with reservation (OGAWR) mechanism in dynamic cloud environments is designed. In proposed mechanism, users are incentivized to be truthful not only about their valuations, but also about their arrival, departure and the characters of jobs (flexible or inflexible). Through extensive experiments that evaluate our technique, we show that OGAWR is better than the allocation which does not distinguish the inflexible agents from flexible agents, from the perspective of improving social welfare and the number of completed jobs. In future work, we plan to implement our mechanism in an experimental cloud computing system, as part of an integrated solution for the management of resources.

Acknowledgments The workload log from the RICC cluster was graciously provided by Motoyoshi Kurokawa. This work has been supported by the National Natural Science Foundation of China (No. 61170029, 61472240, 61373032 and 71271126), Doctoral Fund of Ministry of Education of China under Grant No. 20120078110002, and Zhejiang Provincial Science and Technology Plan of China under Grant No. 2013C31097.

References

1. Abramson D, Buyya R, Giddy J (2002) A computational economy for grid computing and its implementation in the nimrod-g resource broker. Future Gener Computer Syst 18(8):1061–1074
2. Agmon Ben-Yehuda O, Ben-Yehuda M, Schuster A, Tsafrir D (2013) Deconstructing amazon ec2 spot instance pricing. ACM Trans Econ Comput 1(3):16
3. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I et al (2010) A view of cloud computing. Commun ACM 53(4):50–58
4. Constantin F, Feldman J, Muthukrishnan S, Pál M (2009) An online mechanism for ad slot reservations with cancellations. In: Proceedings of the twentieth annual ACM-SIAM symposium on discrete algorithms, pp 1265–1274. Society for Industrial and Applied Mathematics
5. Danak A, Mannor S (2010) Resource allocation with supply adjustment in distributed computing systems. In: IEEE 30th international conference on distributed computing systems (ICDCS), 2010, pp 498–506. IEEE
6. Dash RK, Vytelingum P, Rogers A, David E, Jennings NR (2007) Market-based task allocation mechanisms for limited-capacity suppliers. IEEE Trans Syst Man Cybern Part A Syst Humans 37(3):391–405

7. Feitelson DG (2015) Parallel workloads archives: logs. http://www.cs.huji.ac.il/labs/parallel/workload/logs.html
8. Friedman EJ, Parkes DC (2003) Pricing wifi at starbucks: issues in online mechanism design. In: Proceedings of the 4th ACM conference on electronic commerce, pp 240–241. ACM
9. Garg SK, Venugopal S, Broberg J, Buyya R (2013) Double auction-inspired meta-scheduling of parallel applications on global grids. J Parallel Distrib Comput 73(4):450–464
10. Gerding EH, Robu V, Stein S, Parkes DC, Rogers A, Jennings NR (2011) Online mechanism design for electric vehicle charging. In: The 10th international conference on autonomous agents and multiagent systems, vol 2, pp 811–818. International Foundation for Autonomous Agents and Multiagent Systems
11. Gershkov A, Moldovanu B (2010) Efficient sequential assignment with incomplete information. Games Econ Behav 68(1):144–154
12. Hajiaghayi MT (2005) Online auctions with re-usable goods. In: Proceedings of the 6th ACM conference on electronic commerce, pp 165–174. ACM
13. Hao F, Kodialam M, Lakshman T, Mukherjee S (2014) Online allocation of virtual machines in a distributed cloud. In: INFOCOM, 2014 proceedings IEEE, pp 10–18. IEEE
14. Hsu CH, Slagter KD, Chen SC, Chung YC (2014) Optimizing energy consumption with task consolidation in clouds. Inf Sci 258:452–462
15. Jain N, Menache I, Naor JS, Yaniv J (2014) A truthful mechanism for value-based scheduling in cloud computing. Theory Comput Syst 54(3):388–406
16. Lavi R, Nisan N (2000) Competitive analysis of incentive compatible on-line auctions. In: Proceedings of the 2nd ACM conference on electronic commerce, pp 233–241. ACM
17. Lizhe W, Hao G, Peng L, Ke L, Joanna K, Rajiv R, Y ZA (2015) Particle swarm optimization based dictionary learning for remote sensing big data. J Knowl Based Syst 79:43–50
18. Lizhe W, Jie T, Rajiv R, Holger M, Achim S, Jingying C, Dan C (2013) G-hadoop: Mapreduce across distributed data centers for data-intensive computing. J Future Gener Compter Syst 29(3):739–750
19. Lizhe W, Samee UK, Dan C, Joanna K, Rajiv R, Cheng-Zhong X, Y ZA (2013) Energy-aware parallel task scheduling in a cluster. J Future Gener Compter Syst 29(7):1661–1670
20. Ma W, Zheng B, Qin T, Tang P, Liu T (2014) Online mechanism design for cloud computing. arXiv:1403.1896
21. Nisan N, Roughgarden T, Tardos E, Vazirani VV (2007) Algorithmic game theory, vol 1. Cambridge University Press, Cambridge
22. Niu D, Feng C, Li B (2012) Pricing cloud bandwidth reservations under demand uncertainty. In: ACM SIGMETRICS performance evaluation review, vol 40, pp 151–162. ACM
23. Parkes DC, Duong Q (2007) An ironing-based approach to adaptive online mechanism design in single-valued domains. In: AAAI, vol 7, pp 94–101
24. Parkes DC, Singh SP (2003) An mdp-based approach to online mechanism design. In: Advances in neural information processing systems
25. Porter R (2004) Mechanism design for online real-time scheduling. In: Proceedings of the 5th ACM conference on electronic commerce, pp 61–70. ACM
26. Robu V, Stein S, Gerding EH, Parkes DC, Rogers A, Jennings NR(2012) An online mechanism for multi-speed electric vehicle charging. In: Auctions, market mechanisms, and their applications, pp 100–112. Springer, New York
27. Shi W, Zhang L, Wu C, Li Z, Lau F (2014) An online auction framework for dynamic resource provisioning in cloud computing. In: The 2014 ACM international conference on measurement and modeling of computer systems, pp 71–83. ACM
28. Wang Q, Ren K, Meng X (2012) When cloud meets ebay: towards effective pricing for cloud computing. In: INFOCOM, 2012 proceedings IEEE, pp 936–944. IEEE
29. Weijing S, Lizhe W, Rajiv R, Joanna K, Dan C (2015) Towards modeling large-scale data flows in a multidatacenter computing system with petri net. IEEE Syst J 9(2):416–426
30. Wolski R, Plank JS, Brevik J, Bryan T (2001) Analyzing market-based resource allocation strategies for the computational grid. Int J High Perfor Comput Appl 15(3):258–281
31. Wu C, Li Z, Qiu X, Lau F (2012) Auction-based p2p vod streaming: incentives and optimal scheduling. ACM Trans Multim Comput Commun Appl (TOMCCAP) 8(1S):14
32. Wu X, Gu Y, Li G, Ma X, Tao J (2014) Online mechanism design for vms allocation in private cloud. In: The 11th IFIP international conference on network and parallel computing (NPC'14), pp 234–246
33. Xu Y, Li K, Hu J, Li K (2014) A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. Inf Sci 270:255–287

34. Zaman S, Grosu D (2012) An online mechanism for dynamic vm provisioning and allocation in clouds. In: IEEE 5th international conference on cloud computing (CLOUD), 2012, pp 253–260. IEEE
35. Zaman S, Grosu D (2013) Combinatorial auction-based allocation of virtual machine instances in clouds. J Parallel Distrib Comput 73(4):495–508
36. Zhang H, Li B, Jiang H, Liu F, Vasilakos AV, Liu J (2013) A framework for truthful online auctions in cloud computing with heterogeneous user demands. In: INFOCOM, 2013 Proceedings IEEE, pp 1510–1518. IEEE
37. Zhang Q, Zhu Q, Boutaba R (2011) Dynamic resource allocation for spot markets in cloud computing environments. In: 2011 Fourth IEEE international conference on utility and cloud computing (UCC), pp 178–185. IEEE
38. Zhao J, Li H, Wu C, Li Z, Zhang Z, Lau F (2014) Dynamic pricing and profit maximization for the cloud with geo-distributed data centers. In: INFOCOM, 2014 Proceedings IEEE, pp 118–126. IEEE
39. Zhu Y, Li B, Li Z (2012) Truthful spectrum auction design for secondary networks. In: INFOCOM, 2012 Proceedings IEEE, pp 873–881. IEEE