J Supercomput (2013) 65:866–885
DOI 10.1007/s11227-013-0869-z

Efficient resource allocation for optimizing objectives of cloud users, IaaS provider and SaaS provider in cloud environment

Chunlin Li · Layuan Li

Published online: 7 February 2013

Abstract The cloud architecture is usually composed of several XaaS layers—including Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). The paper studies efficient resource allocation to optimize objectives of cloud users, IaaS provider and SaaS provider in cloud computing. The paper proposes the composition of different layers in the cloud, such as IaaS and SaaS, and its joint optimization for efficient resource allocation. The efficient resource allocation optimization problem is conducted by subproblems. The proposed cloud resource allocation optimization algorithm is achieved through an iterative algorithm. The experiments are conducted to compare the performance of proposed joint optimization algorithm for efficient resource allocation with other related works.

Keywords Cloud computing · Resource allocation · IaaS provider · SaaS provider

1 Introduction

Cloud computing represents a state-of-the-art “computing as a service” paradigm, where the configurable computing resources are pooled and shared among multiple users and are efficiently provisioned to them on-demand through a broadband network access [1]. The deployment of cloud services promises enterprises a number of benefits, such as faster time to market and improved scalability, as well as cost benefits in terms of lower start-up and operations costs. By providing virtualized computing resources as a service in a pay-as-you-go manner, cloud computing enables new business models and cost-effective resource usage. Instead of having to maintain

C. Li (✉) · L. Li
Department of Computer Science, Wuhan University of Technology, Wuhan 430063, P.R. China
e-mail: chunlin74@tom.com

L. Li
e-mail: jwtu@public.wh.hb.cn

their own data center, companies can concentrate on their core business and purchase resources when needed. In cloud computing, the consumer is able to exploit virtual machines (VMs) running on top of the cloud providers' physical infrastructure. The consumer then can deploy and run arbitrary software including operating systems and applications. Computing resources can be acquired on a pay-per-use basis; hence resource costs for customers can be reduced. Because of the pay-as-you-go model, the goal of resource allocation should be to keep the resource budget to a minimum, while meeting an application's needs.

Cloud computing can be divided into three main categories: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). SaaS providers allow their customers to use specific applications over the cloud. These applications can be anything from web-based email (like G-Mail) to real applications like twitter. SaaS providers derive their profits from the margin between the operational cost of infrastructure and the revenue generated from customers. Therefore, SaaS providers are looking into solutions that minimize the overall infrastructure cost without adversely affecting the customers. PaaS providers is one step further than SaaS as it allows users to create their own SaaS applications using certain software API and tools. These applications run on the platforms provided by these cloud IaaS providers. Examples of PaaS are Google AppEngine and Microsoft Azure. IaaS provides users with full control over virtual machines that run over the cloud which the user can configure and customize to fit their needs. Examples are Amazon EC2 and GoGrid.

The paper studies efficient resource allocation to optimize objectives of cloud users, IaaS provider and SaaS provider in cloud computing. The paper proposes the composition of different layers in the cloud such as IaaS and SaaS and its joint optimization for efficient resource allocation. The efficient resource allocation optimization problem is conducted by subproblems. The proposed cloud resource allocation optimization algorithm is achieved through an iterative algorithm. The experiments are conducted to compare the performance of proposed joint optimization algorithm for efficient resource allocation with other related work.

The rest of the paper is structured as followings. Section 2 discusses the related works. Section 3 presents efficient resource allocation for optimizing the objectives of cloud users, IaaS provider and SaaS provider in cloud computing. Section 4 proposes cloud resource allocation algorithm. In Sect. 5 the experiments are conducted. Section 6 gives the conclusions to the paper.

2 Related works

A number of research efforts address dynamic resource provisioning of cloud computing [3–8]. Saurabh Kumar Garg et al. [9] proposed near-optimal scheduling policies that exploit heterogeneity across multiple data centers for a cloud provider. They consider a number of energy efficiency factors (such as energy cost, carbon emission rate, workload, and CPU power efficiency) which change across different data centers depending on their location, architectural design, and management system. Linlin Wu et al. [2] proposed resource allocation algorithms for SaaS providers who want to minimize infrastructure cost and SLA violations. In their proposed algorithms, SaaS providers manage the dynamic change of customers, mapping customer requests to infrastructure level parameters and handling heterogeneity of Virtual Machines. They also take into account the customers' Quality of Service parameters. Bernardetta Addis et al. [12] proposed resource allocation policies for the management of multi-tier virtualized cloud systems with the aim at maximizing the profits associated with multiple-class SLAs. A heuristic solution based on local-search which provides also availability guarantees for the running applications to have been developed. Martin Randles et al. [5] presented a comparative study of three distributed load-balancing-based resource provisioning algorithms for cloud computing. Denis Saure et al. [13] considered a class of pricing policies called Time-of-Use (ToU), and proposed a simple and intuitive algorithm. They evaluate the performance of the approach numerically and discuss the implementation of the proposed approach for cloud computing. Yousri Kouki et al. [10] studied an automated and dynamic resource allocation problem in a cloud environment. They consider the problem where there is a fixed time-limit as well as a resource budget for a particular task. Within these constraints, an adaptive application needs to maximize the Quality of Service (QoS) metric. They also present a dynamic resource provisioning algorithm, which is based on control theory. I. Rodero et al. [14] presented an energy-aware online provisioning approach for HPC applications on consolidated and virtualized computing platforms. Energy efficiency is achieved using a workload-aware, just-right dynamic provisioning mechanism and the ability to power down subsystems of a host system that are not required by the VMs mapped to it. Hady S. Abdelsalam et al. [15] created a mathematical model for power management for a cloud computing environment that primarily serves clients with interactive applications such as web services. The mathematical model computes the optimal number of servers and the frequencies at which they should run. In ref. [16], a new framework is presented that provides efficient green enhancements within a scalable cloud computing architecture. Using power-aware scheduling techniques, variable resource management, live migration, and a minimal virtual machine design, overall system efficiency is improved in data center based cloud with minimal overhead.

Resource allocation in cloud computing is widely studied. Young Choon Lee et al. [19] addressed the problem of service request scheduling in cloud computing systems. They consider a three-tier cloud structure, which consists of infrastructure vendors, service providers and consumers; the latter two parties are of particular interest to us. They develop a pricing model for clouds and present two sets of profit-driven scheduling algorithms. Fangzhe Chang et al. [17] studied optimal resource allocation in clouds. They formulate demand for computing power and other resources as a resource allocation problem with multiplicity. They present an approximation algorithm with a proof of its approximation bound that can yield close to optimal solutions in polynomial time. In [18], Stefano Ferretti et al. described middleware architecture to respond to the Quality of Service (QoS) requirements of the cloud customer applications. The proposed architecture incorporates a load-balancer that distributes the computational load across the platform resources, and monitors the QoS the platform delivers. If this deviates from that specified in the SLA, so as to violate it, the platform is reconfigured dynamically in order to incorporate additional resources from the cloud. Fei Teng and Frédéric Magoulès [20] studied resource pricing and equilibrium allocation policy in cloud computing. They propose a new resource pricing and allocation policy where users can predict the future resource price as well as satisfy budget and deadline constraints. In [21], Yağız Onat Yazır et al. introduced a new approach for dynamic autonomous resource management in computing clouds. Their approach consists of a distributed architecture of NAs that performs resource configurations using MCDA with the PROMETHEE method.

Existing approaches on cloud resource allocation are locally optimally based. Although these approaches take requirements of cloud users and cloud providers into account, they cannot resolve the problem of global optimization in cloud. The main contribution of this paper is to study efficient resource allocation to optimize objectives of cloud users, IaaS provider and SaaS provider and achieve global optimization in cloud system, the existing works being relatively limited.

3 Efficient resource allocations for optimizing objectives of cloud users, IaaS provider and SaaS provider in cloud environment

In this section, we will give the model description of efficient resource allocation, and then illustrate how to achieve optimize the objectives of cloud users, IaaS provider and SaaS provider in cloud computing, also give the mathematic solution of the optimization of the problem.

3.1 Model description

The cloud systems can be modeled by IaaS provider, SaaS provider and cloud users running on those virtual machines. The resources in cloud computing system include the computation resource, communication and storage resources. A customer sends requests for utilizing enterprise software services offered by a SaaS provider, who uses three layers, namely application layer, platform layer and infrastructure layer, to satisfy the customer's request. Figure 1 shows resource allocation for optimizing the objectives of cloud users, IaaS provider and SaaS provider in cloud computing. Our efficient resource allocation method in cloud computing is studied by dividing the global cloud utility optimization problem into subproblems, which are conducted by IaaS provider, SaaS provider, and cloud users. The proposed optimization decomposition policy produces an optimal set of cloud user's payments, SaaS service allocation, payments of SaaS providers and cloud resources at the application layer, service layer and resource layer respectively to maximize cloud system's utility. The decomposition of global cloud utility optimization problem is according to the vertical organization of cloud computing layers. The IaaS provider, SaaS provider and cloud users obtain inputs from other participants, try to maximize their own utility and provide outputs back to other participants.

The proposed global optimization strategy for resource allocation mechanisms enables IaaS providers and SaaS providers to partition their resource and services based on some criteria such as profit and resource efficiency. Global optimization-based cloud resource allocation is to allocate cloud resources and services such

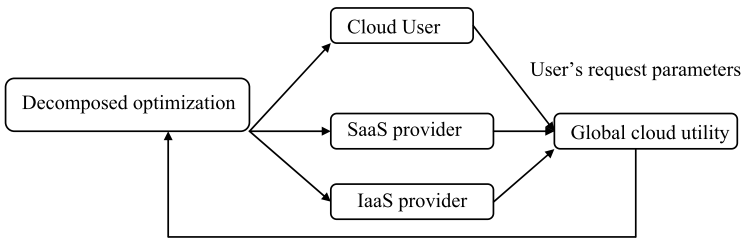


Fig. 1 Resource allocation considering cloud users, IaaS provider and SaaS provider

that the cloud system utility U_{Cloud} be maximized subject to resource constraints of IaaS provider, QoS constraints of cloud users, and SaaS service constraints of SaaS provider. The resource allocation method for optimizing the objectives of cloud users, IaaS provider and SaaS provider in cloud computing is formulated as follows:

$$\begin{aligned}&\text{Max}\, U_{\text{Cloud}}\\ &\text{s.t.}\quad B_m \geq \sum_i q_m^i, \quad T_m \geq \sum_{n=1}^{N} t_m^n, \quad C_j^{\text{cpu}} \geq \sum_i x_i^{j\,(\text{cpu})}, \quad C_j^{\text{ram}} \geq \sum_i x_i^{j\,(\text{ram})}\\ &\qquad SB_i \geq \sum_j P_i^{j\,(\text{cpu})} + \sum_j P_i^{j\,(\text{ram})}, \quad L_i^{(\text{ram})} \geq \sum_j D_i^{j\,(\text{ram})},\\ &\qquad L_i^{(\text{cpu})} \geq \sum_j D_i^{j\,(\text{cpu})}\end{aligned}$$

The resource allocation method in cloud computing aims to maximize U_{Cloud} subject to the constraints of IaaS provider, SaaS provider, and cloud users. In this problem, the first type of the constraints is related with IaaS provider. Further, $x_i^{j\,(\text{cpu})}$ is CPU required by a VM for SaaS provider i from the IaaS provider $j, x_i^{j\,(\text{ram})}$ is the memory required by a VM for SaaS provider i from the IaaS provider j. The constraint implies that the aggregate CPU does not exceed the total capacity C_j^{cpu} of CPU of IaaS provider j, aggregate memory units do not exceed the total resource C_j^{ram} of memory of IaaS provider j. The second type of constraints is related with cloud users. Cloud users should complete all its jobs under time limits and certain payment. Cloud users needs to complete a sequence of jobs in a specified amount of time, T_m, while the payment overhead accrued cannot exceed the budget B_m, q_m^i being the payment of the cloud users m to the SaaS provider i. $P_i^{j\,(\text{cpu})}$, $P_i^{j\,(\text{ram})}$ are the payments of the SaaS provider i to the IaaS provider j for CPU and memory required by a VM, respectively. SB_i is the budget of SaaS provider i. $D_i^{j\,(\text{ram})}$, $D_i^{j\,(\text{cpu})}$ represent resource allocation delays from IaaS provider j for memory and CPU required by SaaS provider i supporting a VM, respectively. $L_i^{(\text{ram})}$, $L_i^{(\text{cpu})}$ are upper limits of allocation delay of memory and CPU required by SaaS provider i for supporting VM.

$$\begin{aligned}\text{Max}\, U_{\text{Cloud}} = &\sum \left(P_i^{j\,(\text{cpu})} \log x_i^{j\,(\text{cpu})} + P_i^{j\,(\text{ram})} \log x_i^{j\,(\text{ram})}\right)\\ &+ \sum_m q_m^i \log v_m^i + \left(E_m - \sum_{n=1}^{N} e_m^n\right)\end{aligned}$$

$$\begin{aligned} \text{s.t.}\quad & B_m \geq \sum_i q_m^i, \quad T_m \geq \sum_{n=1}^{N} t_m^n, \quad C_j^{\text{cpu}} \geq \sum_i x_i^{j\,(\text{cpu})}, \quad C_j^{\text{ram}} \geq \sum_i x_i^{j\,(\text{ram})} \\ & SB_i \geq \sum_j P_i^{j\,(\text{cpu})} + \sum_j P_i^{j\,(\text{ram})}, \quad L_i^{(\text{ram})} \geq \sum_j D_i^{j\,(\text{ram})}, \\ & L_i^{(\text{cpu})} \geq \sum_j D_i^{j\,(\text{cpu})} \end{aligned}$$

In U_{Cloud}, $P_i^{j\,(\text{cpu})} \log x_i^{j\,(\text{cpu})} + P_i^{j\,(\text{ram})} \log x_i^{j\,(\text{ram})}$ represent the revenue of IaaS provider j. $\sum_m q_m^i \log v_m^i$ represents the revenue obtained by SaaS provider i from cloud users m. $(E_m - \sum_{n=1}^{N} e_m^n)$ is remaining energy of cloud users. In the paper, maximization formulation of the cloud system utility adopts a network utility maximization (NUM) framework [23] in which each application has an associated utility function. In ref. [23], an optimization framework leads to a decomposition of the overall system problem into a separate problem for each user, in which the user chooses a charge per unit time that the user is willing to pay, and one for the network.

We can apply the Lagrangian method to the above problem [24]. Lagrangian relaxation is a relaxation technique which works by moving hard constraints into the objective so as to exact a penalty on the objective if they are not satisfied. Let us consider the Lagrangian form of this optimization problem:

$$\begin{aligned} L = {} & \sum \big(P_i^{j\,(\text{cpu})} \log x_i^{j\,(\text{cpu})} + P_i^{j\,(\text{ram})} \log x_i^{j\,(\text{ram})}\big) \\ & + \lambda\left(SB_i - \left(\sum_j P_i^{j\,(\text{cpu})} + \sum_j P_i^{j\,(\text{ram})}\right)\right) \\ & + \left(E_m - \sum_{n=1}^{N} e_m^n\right) + \sum_m q_m^i \log v_m^i + \beta\left(B_m - \sum_i q_m^i\right) + \gamma\left(T_m - \sum_{n=1}^{N} t_m^n\right) \end{aligned}$$

Since the Lagrangian is separable, the maximization of the Lagrangian can be processed in parallel by IaaS provider, SaaS provider and cloud users, respectively. The optimization-based approach for resource allocation and service provisioning naturally leads to a decomposition of problem MaxU_{Cloud} among three participants. The problem can be decomposed into three subproblems F_1, F_2 and F_3, which are respectively conducted by IaaS provider, SaaS provider and cloud users as follows:

$$\begin{aligned} & F_1 = \text{Max} \sum \big(P_i^{j\,(\text{cpu})} \log x_i^{j\,(\text{cpu})} + P_i^{j\,(\text{ram})} \log x_i^{j\,(\text{ram})}\big) \\ & \text{s.t.}\quad C_j^{\text{cpu}} \geq \sum_i x_i^{j\,(\text{cpu})}, \quad C_j^{\text{ram}} \geq \sum_i x_i^{j\,(\text{ram})} \\ & F_2 = \text{Max}\left\{\left(SB_i - \sum_j P_i^{j\,(\text{cpu})} - \sum_j P_i^{j\,(\text{ram})}\right) + \sum_m q_m^i \log v_m^i\right\} \\ & \text{s.t.}\quad S_i \geq \sum_m v_m^i, \\ & \qquad L_{\text{i}}^{(\text{ram})} \geq \sum_j D_i^{j\,(\text{ram})}, \end{aligned}$$

$$L_{\mathrm{i}}^{(\mathrm{cpu})} \geq \sum_j D_i^{j\,(\mathrm{cpu})}$$

$$F_3 = \mathrm{Max}\left\{\left(B_m - \sum_i q_m^i\right) + \left(T_m - \sum_n t_m^n\right) + \left(E_m - \sum_{n=1}^{N} e_m^n\right)\right\}$$

Problem F_1 is conducted by the IaaS provider at resource layer; different IaaS providers compute optimal resource allocation for maximizing the revenue of their own. Problem F_2 is conducted by SaaS provider at the SaaS service layer, the SaaS provider pays IaaS provider for available cloud resources to run VMs and also provides software services for cloud users to maximize the benefits. SaaS provider i submits the payment $P_i^{j\,(\mathrm{cpu})}$, $P_i^{j\,(\mathrm{ram})}$ to IaaS provider j for CPU and memory required by a VM. $SB_i - \sum_j P_i^{j\,(\mathrm{cpu})} - \sum_j P_i^{j\,(\mathrm{ram})}$ represents surpluses of SaaS provider's budget. Further, q_m^i stands for the payments of the cloud users m to the SaaS provider i. $\sum_m q_m^i \log v_m^i$ presents the revenue obtained by SaaS provider i from cloud users m. The objective of Problem F_2 is to maximize the surplus of SaaS providers that pays IaaS provider at the resource layer for available resources to run VMs and also revenue that is obtained by providing software services for cloud users. Problem F_3 is conducted by cloud users; the cloud users give the unique optimal payment to SaaS provider under the deadline constraint to maximize the cloud user's satisfaction. $B_m - \sum_i q_m^i$ represents the surplus of cloud users, which is obtained by budgets subtracting the payments to SaaS providers. $(T_i - \sum_{n=1}^{N} t_m^n)$ represents the saving times for cloud users, which is gotten by time limit subtracting actual spending time. $(E_m - \sum_{n=1}^{N} e_m^n)$ is remaining energy of cloud users. So, the objective of Problem F_3 is to get more surpluses of money and energy, and complete the task for cloud users as soon as possible.

3.2 Solutions to efficient cloud resource allocations

Optimization-based cloud resource allocation problem can be decomposed into a sequence of three subproblems at three layers. Interactions between the three subproblems are through optimal variables for capacities of cloud computing resources and service demand. Resource allocation problem is conducted by IaaS provider. Different IaaS providers compute optimal resource allocation for maximizing the revenue of their own under constrains of cloud resource capacity. Different IaaS providers compute optimal resource allocation for maximizing the revenue of their own, the objective of IaaS providers is to maximize $P_i^{j\,(\mathrm{cpu})} \log x_i^{j\,(\mathrm{cpu})} + P_i^{j\,(\mathrm{ram})} \log x_i^{j\,(\mathrm{ram})}$ under the constraints of their provided amounts.

$$F_1 = \mathrm{Max} \sum \left(P_i^{j\,(\mathrm{cpu})} \log x_i^{j\,(\mathrm{cpu})} + P_i^{j\,(\mathrm{ram})} \log x_i^{j\,(\mathrm{ram})}\right)$$

$$\text{s.t.} \quad C_j^{\mathrm{cpu}} \geq \sum_i x_i^{j\,(\mathrm{cpu})}, \quad C_j^{\mathrm{ram}} \geq \sum_i x_i^{j\,(\mathrm{ram})}$$

$$U_{\mathrm{IaaS}}\left(x_i^{j\,(\mathrm{cpu})}, x_i^{j\,(\mathrm{ram})}\right) = \sum \left(P_i^{j\,(\mathrm{cpu})} \log x_i^{j\,(\mathrm{cpu})} + P_i^{j\,(\mathrm{ram})} \log x_i^{j\,(\mathrm{ram})}\right)$$

The Lagrangian for F_1 problem is $L_1(x_i^{j\,(\mathrm{cpu})}, x_i^{j\,(\mathrm{ram})})$:

$$\begin{aligned} L_1\big(x_i^{j\,(\mathrm{cpu})}, x_i^{j\,(\mathrm{ram})}\big) &= \sum \big(P_i^{j\,(\mathrm{cpu})} \log x_i^{j\,(\mathrm{cpu})} + P_i^{j\,(\mathrm{ram})} \log x_i^{j\,(\mathrm{ram})}\big) \\ &\quad + \beta\Big(C_j^{\mathrm{cpu}} - \sum_i x_i^{j\,(\mathrm{cpu})}\Big) + \gamma\Big(C_j^{\mathrm{ram}} - \sum_i x_i^{j\,(\mathrm{ram})}\Big) \\ &= \sum \big(P_i^{j\,(\mathrm{cpu})} \log x_i^{j\,(\mathrm{cpu})} + P_i^{j\,(\mathrm{ram})} \log x_i^{j\,(\mathrm{ram})} \\ &\quad - \beta x_i^{j\,(\mathrm{cpu})} - \gamma x_i^{j\,(\mathrm{ram})}\big) + \gamma C_j^{\mathrm{cpu}} + \beta C_j^{\mathrm{ram}} \end{aligned}$$

where β, γ are the Lagrangian constants. From Karush–Kuhn–Tucker theorem [25] we know that the optimal solution is given by $\partial L_1(x_i^{j\,(\mathrm{cpu})}, x_i^{j\,(\mathrm{ram})})/\partial x_i^{j\,(\mathrm{cpu})} = 0$ for $\lambda > 0$.

Let $\partial L_1(x_i^{j\,(\mathrm{cpu})}, x_i^{j\,(\mathrm{ram})})/\partial x_i^{j\,(\mathrm{cpu})} = 0$ to obtain $x_i^{j\,(\mathrm{cpu})} = P_i^{j\,(\mathrm{cpu})}/\beta$.

Using this result in the constraint equation $C_j^{\mathrm{cpu}} \geq \sum_i x_i^{j\,(\mathrm{cpu})}$, we can determine β as

$$\beta = \frac{\sum_{d=1}^{n} P_d^{j\,(\mathrm{cpu})}}{C_j^{\mathrm{cpu}}}$$

We substitute β into $x_i^{j\,(\mathrm{cpu})}$ to obtain

$$x_i^{j\,(\mathrm{cpu})*} = \frac{P_i^{j\,(\mathrm{cpu})} C_j^{\mathrm{cpu}}}{\sum_{d=1}^{n} P_d^{j\,(\mathrm{cpu})}}$$

Here $x_i^{j\,(\mathrm{cpu})*}$ is the unique optimal CPU allocation to support VM for SaaS provider i, and it maximizes the revenue of IaaS provider j.

Let us consider memory allocation optimization problem of IaaS provider j, using the similar method.

Let $\partial L_1(x_i^{j\,(\mathrm{cpu})}, x_i^{j\,(\mathrm{ram})})/\partial x_i^{j\,(\mathrm{ram})} = 0$ to obtain

$$x_i^{j\,(\mathrm{ram})*} = \frac{P_i^{j\,(\mathrm{ram})} C_j^{\mathrm{ram}}}{\sum_{k=1}^{n} P_k^{j\,(\mathrm{ram})}}$$

Here $x_i^{j\,(\mathrm{ram})*}$ is unique optimal memory allocation to support VM for SaaS provider, and it maximizes the revenue of IaaS provider j.

Service provisioning of SaaS provider refers to providing a specific service which runs on VMs by leasing various cloud resources from IaaS provider to construct SaaS service for cloud users to achieve their goals. The utility function for SaaS provider is maximally optimized with specific constraints. The SaaS provider not only acts as a consumer which pays IaaS provider at the resource layer, but also acts as a supplier which provides SaaS services for cloud users at application layer.

$$\begin{aligned} F_2 &= \mathrm{Max}\Big\{\Big(SB_i - \sum_j P_i^{j\,(\mathrm{cpu})} - \sum_j P_i^{j\,(\mathrm{ram})}\Big) + \sum_m q_m^i \log v_m^i\Big\} \\ \text{s.t.}\quad & S_i \geq \sum_m v_m^i, \end{aligned}$$

$$L_i^{(\mathrm{ram})} \geq \sum_j D_i^{j\,(\mathrm{ram})},$$
$$L_i^{(\mathrm{cpu})} \geq \sum_j D_i^{j\,(\mathrm{cpu})}$$

In the above formula, v_m^i is SaaS service sold to cloud users m by SaaS provider i. Further, q_m^i stands for the payments of the cloud users m to the SaaS provider i. $\sum_m q_m^i \log v_m^i$ represents the revenue obtained by SaaS provider i from cloud users m. SaaS provider cannot sell SaaS software service to cloud users by more than S_i, which is the upper limit of service owned by SaaS provider i. We assume that SaaS provider i submits payment $P_i^{j\,(\mathrm{cpu})}$, $P_i^{j\,(\mathrm{ram})}$ to the IaaS provider j for CPU and memory required by a VM, respectively. Let $m_i = P_i^{j\,(\mathrm{cpu})} + P_i^{j\,(\mathrm{ram})}$, m_i being the total payment of the ith SaaS provider. N SaaS providers compete for cloud computing resources with finite capacity. $D_i^{j\,(\mathrm{ram})}$, $D_i^{j\,(\mathrm{cpu})}$ represent resource allocation delays from IaaS provider j for memory and CPU required by SaaS provider i supporting a VM, respectively. $L_i^{(\mathrm{ram})}$, $L_i^{(\mathrm{cpu})}$ are allocation delay limit of memory and CPU required by SaaS provider i for supporting VM. The cloud resource of IaaS provider is allocated using a market mechanism, where the partitions depend on the relative payments sent by the SaaS providers. The Lagrangian associated with problem F_2 for the SaaS provider's utility is $L(P_i^{j\,(\mathrm{cpu})}, P_i^{j\,(\mathrm{ram})}, v_m^i)$:

$$\begin{aligned}
L\left(P_i^{j\,(\mathrm{cpu})}, P_i^{j\,(\mathrm{ram})}, v_m^i\right) &= SB_i - \sum_j P_i^{j\,(\mathrm{cpu})} - \sum_j P_i^{j\,(\mathrm{ram})} \\
&\quad + \sum_m q_m^i \log v_m^i + \delta\left(S_i - \sum_m v_m^i\right) \\
&\quad + \eta\left(L_{\mathrm{i}}^{(\mathrm{ram})} - \sum_j D_i^{j\,(\mathrm{ram})}\right) + \beta\left(L_{\mathrm{i}}^{(\mathrm{cpu})} - \sum_j D_i^{j\,(\mathrm{cpu})}\right) \\
&= \left(SB_i - \sum_j P_i^{j\,(\mathrm{cpu})} - \sum_j P_i^{j\,(\mathrm{ram})}\right. \\
&\quad \left. - \eta \sum_k D_i^{j\,(\mathrm{ram})} - \beta \sum_l D_i^{j\,(\mathrm{cpu})}\right) \\
&\quad + \sum_m \left(q_m^i \log v_m^i - \delta v_m^i\right) + \delta S_i + \eta L_{\mathrm{i}}^{(\mathrm{ram})} + \beta L_{\mathrm{i}}^{(\mathrm{cpu})}
\end{aligned}$$

Since the Lagrangian is separable, this maximization of the Lagrangian over $P_i^{j\,(\mathrm{cpu})}$, $P_i^{j\,(\mathrm{ram})}$, v_m^i can be conducted in parallel as the following two problems:

$$L_{\mathrm{saas1}} = \mathrm{Max}\left(SB_i - \sum_j P_i^{j\,(\mathrm{cpu})} - \sum_j P_i^{j\,(\mathrm{ram})} - \eta \sum_j D_i^{j\,(\mathrm{ram})} - \beta \sum_j D_i^{j\,(\mathrm{cpu})}\right)$$
$$L_{\mathrm{saas2}} = \mathrm{Max} \sum_m \left(q_m^i \log v_m^i - \delta v_m^i\right)$$

The first subproblem is related to interaction between the IaaS provider and SaaS provider, the SaaS provider acting as a consumer, which pays IaaS provider for available cloud resources to run VMs. The second subproblem is conducted with interaction between the cloud users and SaaS provider, the SaaS provider acting as a supplier, which provides SaaS software services for cloud users.

Firstly, we consider the first subproblem. Let r_j^{cpu} and r_j^{ram} denote the price of CPU and memory of the IaaS provider j, respectively. Let the pricing policy of IaaS providers $r^{\text{cpu}} = (r_1^{\text{cpu}}, r_2^{\text{cpu}}, \ldots, r_j^{\text{cpu}})$ be a set of unit prices of CPU, $r^{\text{ram}} = (r_1^{\text{ram}}, r_2^{\text{ram}}, \ldots, r_j^{\text{ram}})$ be a set of unit prices of memory. The ith SaaS provider receives cloud resources for supporting VMs proportional to its payment relative to the sum of the IaaS provider's revenue. Let $x_i^{j\,(\text{cpu})}, x_i^{j\,(\text{ram})}$ be the units of CPU and memory allocated to SaaS provider i for VMs by IaaS provider j:

$$x_i^{j\,(\text{cpu})} = C_j^{\text{cpu}} \frac{P_i^{j\,(\text{cpu})}}{r_j^{\text{cpu}}}, \qquad x_i^{j\,(\text{ram})} = C_j^{\text{ram}} \frac{P_i^{j\,(\text{ram})}}{r_j^{\text{ram}}}$$

Resource allocation delays from IaaS provider j for CPE and memory required by SaaS provider i supporting a VM are:

$$D_i^{j\,(\text{cpu})} = \frac{r_j^{\text{cpu}}}{P_i^{j\,(\text{cpu})} C_j^{\text{cpu}}}, \qquad D_i^{j\,(\text{ram})} = \frac{r_j^{\text{ram}}}{P_i^{j\,(\text{ram})} C_j^{\text{ram}}}$$

We reformulate first subproblem as follows:

$$L_{\text{saas1}} = \text{Max}\Bigg(SB_i - \sum_j P_i^{j\,(\text{ram})} - \sum_j P_i^{j\,(\text{cpu})} - \eta \sum_k \frac{r_j^{\text{ram}}}{P_i^{j\,(\text{ram})} C_j^{\text{ram}}} - \beta \frac{r_j^{\text{cpu}}}{P_i^{j\,(\text{cpu})} C_j^{\text{cpu}}} \Bigg)$$

Here η, β are the Lagrangian constants. From Karush–Kuhn–Tucker theorem [25] we know that the optimal solution is given by $\partial L_{\text{saas1}} / \partial P_i^{j\,(\text{ram})} = 0$.

We can get $P_i^{j\,(\text{ram})} = (\frac{\eta r_j^{\text{ram}}}{C_j^{\text{ram}}})^{1/2}$.

Using this result in the constraint equation, we can determine η as

$$(\eta)^{-1/2} = \frac{L_i^{(\text{ram})}}{\sum_{j=1}^{N} (\frac{r_j^{\text{ram}}}{C_j^{\text{ram}}})^{1/2}}$$

We substitute η to obtain

$$P_i^{j\,(\text{ram})*} = \left(\frac{r_j^{\text{ram}}}{C_j^{\text{ram}}} \right)^{1/2} \frac{\sum_{j=1}^{N} (\frac{r_j^{\text{ram}}}{C_j^{\text{ram}}})^{1/2}}{L_i^{(\text{ram})}}$$

It means that SaaS provider i wants to pay $P_i^{j\,(\text{ram})*}$ to IaaS provider j for memory to support VMs and maximizes the benefits of SaaS provider i.

Using a similar method, we can solve CPU allocation of IaaS provider j. $P_i^{j\,(\mathrm{cpu})*}$ is unique optimal payment to IaaS provider j for CPU and maximizes the benefits of SaaS provider:

$$P_i^{j\,(\mathrm{cpu})*} = \left(\frac{r_j^{\mathrm{cpu}}}{C_j^{\mathrm{cpu}}}\right)^{1/2} \frac{\sum_{j=1}^{N} (\frac{r_j^{\mathrm{cpu}}}{C_j^{\mathrm{cpu}}})^{1/2}}{L_i^{(\mathrm{cpu})}}$$

Secondly, we consider the second subproblem, $L_{\mathrm{saas2}} = \mathrm{Max} \sum_m (q_m^i \log v_m^i - \delta v_m^i)$.

Here δ is the Lagrangian constant. From Karush–Kuhn–Tucker theorem we know that the optimal solution is given by $\partial L_{\mathrm{saas2}}/\partial v_m^i = 0$ for $\delta > 0$.

Let $\partial L_{\mathrm{saas2}}/\partial v_m^i = 0$ to obtain $v_m^i = q_m^i/\delta$.

Using this result in the constraint equation, we can determine δ as

$$S_i = \frac{1}{\delta}\sum_{k=1}^{n} q_k^i, \quad \delta = \frac{\sum_{k=1}^{n} q_k^i}{S_i}$$

We substitute δ to obtain

$$v_m^{i*} = \frac{q_m^i S_i}{\sum_{k=1}^{n} q_k^i}$$

Here v_m^{i*} is the unique optimal solution to the subproblem 2. It means that SaaS provider allocates v_m^{i*} to cloud users to maximize its revenue.

$$F_3 = \mathrm{Max}\left\{\left(B_m - \sum_i q_m^i\right) + \left(T_m - \sum_n t_m^n\right) + \left(E_m - \sum_{n=1}^{N} e_m^n\right)\right\}$$
$$\text{s.t.} \quad T_m \ge \sum_n t_m^n$$

$$\mathrm{U}_{\mathrm{clouduser}} = \left(B_m - \sum_i q_m^i\right) + \left(T_m - \sum_n \frac{y_m^n}{v_m^i}\right) + \left(E_m - \sum_{n=1}^{N} e_m^n\right)$$

Here y_m^n is the SaaS software requirement of the mth cloud user's nth job. Let τ_i denote the price of the SaaS service unit of SaaS provider i. Let the pricing policy, $\tau = (\tau_1, \tau_2, \ldots, \tau_i)$, denote the set of service unit prices of all the SaaS providers. The cloud users m receives software services proportional to its payment relative to the sum of the SaaS provider's revenue. Let v_m^ibe the fraction of service allocated to the cloud users m by SaaS provider i. The SaaS software service v_m^i allocated to cloud users m is

$$v_m^i = S_i \frac{q_m^i}{\tau_i}$$

The time taken by the mth cloud users to complete the nth job is: $t_m^n = \frac{y_m^n \tau_i}{S_i q_m^i}$. We reformulate

$$\mathrm{Max}\left\{\left(B_m - \sum_i q_m^i\right) + \left(T_m - \sum_{n=1}^{N} \frac{y_m^n \tau_i}{S_i q_m^i}\right) + \left(E_m - \frac{\gamma y_m^n \tau_i}{S_i q_m^i}\right)\right\}$$

The Lagrangian for the cloud user's utility is $L(q_m^i)$:

$$L(q_m^i) = \left(B_m - \sum_i q_m^i\right) + \left(T_m - \sum_{n=1}^{N} \frac{y_m^n \tau_i}{S_i q_m^i}\right) + \left(E_m - \frac{\gamma y_m^n \tau_i}{S_i q_m^i}\right) + \lambda\left(T_m - \sum_{n=1}^{N} t_m^n\right)$$

Here λ is the Lagrangian constant. From Karush–Kuhn–Tucker theorem we know that the optimal solution is given by $\partial L(q_m^i)/\partial q_m^i = 0$ for $\lambda > 0$.

Let $\partial L(q_m^i)/\partial q_m^i = 0$ to obtain $q_m^i = (\frac{(1+\lambda+\gamma)y_m^n \tau_i}{S_i})^{1/2}$.

Using this result in the constraint equation, we can determine $\theta = 1 + \lambda + \gamma$ as

$$(\theta)^{-1/2} = \frac{T_m}{\sum_{n=1}^{N} (\frac{\tau_n y_m^n}{S_n})^{1/2}}$$

We substitute θ to q_m^i to obtain

$$q_m^{i*} = \left(\frac{y_m^n \tau_i}{S_i}\right)^{1/2} \frac{\sum_{k=1}^{N} (\frac{y_m^n \tau_k}{S_i})^{1/2}}{T_m}$$

Here q_m^{i*} is the unique optimal payment of cloud users m to SaaS provider i under the deadline to maximize the cloud user's benefits.

4 Efficient resource allocations algorithms in cloud computing

The proposed efficient resource allocation for optimizing the objectives of cloud users, IaaS provider and SaaS provider in cloud computing is conducted by sub-problems. In each iteration, the cloud users compute the unique optimal payment to SaaS provider under the deadline constraint to maximize the cloud user's satisfaction. The cloud users individually solves its fees to pay for SaaS services to complete its all jobs, adjusts its SaaS service demand and notifies the SaaS provider about this change. After the new SaaS service demand is observed by the SaaS provider, it updates its price accordingly and communicates the new prices to the cloud users. The SaaS provider never sells a SaaS service for less than the cost paid to the IaaS providers for supporting VMs. So SaaS provider adjusts its cloud resource demand for running VMs under budget constraint. Different IaaS providers compute optimal resource allocation for maximizing the revenue of their own. The iterative Algorithm 1 that achieves resource allocation for optimizing cloud users, IaaS provider and SaaS provider (RASP) is described as follows.

5 Experiments

In this section, the performance evaluation of proposed *RASP* algorithm is conducted. There are a total of 2 IaaS providers, 10 SaaS providers, and 100 cloud users taken for

Algorithm 1 Resource allocation algorithm for cloud users, IaaS provider and SaaS provider (RASP)

```
Subroutine 1
Step 1: cloud users receive from the SaaS provider $i$ the price$\tau_i^{(n)}$;
Step 2: cloud users calculate its optimal payments to SaaS provider to maximize the
        utility of the users;
        $$q_m^{i*} = \text{Max}\left\{\left(B_m - \sum_i q_m^i\right) + \left(T_m - \sum_n t_m^n\right) + \left(E_m - \sum_{n=1}^{N} e_m^n\right)\right\};$$
Step 3: cloud users compute new cloud SaaS service
            If $B_m \geq \sum_i q_m^i$
            Then $v_m^{i(n+1)} = q_m^{i*(n)}/\tau_i^{(n)}$;
            Else Return Null;
Step 4: send SaaS service demand to SaaS providers

Subroutine 2
Step 1: SaaS provider receives SaaS service demand $v_m^i$ from cloud users $m$;
Step 2: SaaS provider receives the prices of CPU and memory of the IaaS provider
        $j, r_j^{\text{cpu}}, r_j^{\text{ram}}$;
Step 3: SaaS provider calculates new price of SaaS service
            If $S_i \geq \sum_m v_m^i$
            Then
            $\tau_i^{(n+1)} = \max\{\varepsilon, \tau_i^{(n)} + \eta(\sum_m v_m^i - S_i)\}$;
            // $\eta > 0$ is a small step size parameter, $n$ is iteration number.
            Else Return Null;
Step 4: SaaS provider calculates optimal payment to maximize the benefit of SaaS
        provider;
        $$P_i^{j\,(\text{cpu})*}, P_i^{j\,(\text{ram})*} = \text{Max}\left\{\left(SB_i - \sum_j P_i^{j\,(\text{cpu})} - \sum_j P_i^{j\,(\text{ram})} - \eta\sum_j D_i^{j\,(\text{ram})} - \beta\sum_j D_i^{j\,(\text{cpu})}\right)\right\};$$
Step 5: send SaaS service price to all cloud users;
Step 6: send the payment to IaaS providers;

Subroutine 3
Step 1: IaaS provider receives the payments $P_i^{j\,(\text{cpu})*}, P_i^{j\,(\text{ram})*}$ from SaaS provider
        $i$;
Step 2: IaaS provider calculates optimal resource allocation to provide VM for SaaS
        provider
        $$x_i^{j\,(\text{cpu})*}, x_i^{j\,(\text{ram})*} = \text{Max}\{\text{U}_{\text{IaaS}}\};$$
```

```
Step 3: IaaS provider computes a new resource price for SaaS providers;
    If $C_j^{\text{cpu}} \geq \sum_i x_i^{j\,(\text{cpu})}$, $C_j^{\text{ram}} \geq \sum_i x_i^{j\,(\text{ram})}$
    Then
        $$r_j^{\text{cpu}(n+1)} = \max\left\{\varepsilon, r_j^{\text{cpu}(n)} + \eta\left(\sum_i x_i^{j\,(\text{cpu})} - C_j^{\text{cpu}}\right)\right\};$$
        $$r_j^{\text{ram}(n+1)} = \max\left\{\varepsilon, r_j^{\text{ram}(n)} + \eta\left(\sum_i x_i^{j\,(\text{ram})} - C_j^{\text{ram}}\right)\right\};$$
  Else Return Null;
Step 4: send new prices $r_j^{\text{cpu}(n+1)}$, $r_j^{\text{ram}(n+1)}$ to SaaS providers
```

Table 1 Simulation parameters

Simulation parameter	Value
Total number of cloud users	100
Total number of SaaS providers	10
Total number of IaaS providers	2
Initial price of VM	[10, 500]
Deadline	[100, 400]
Expense Budget	[100, 1500]
Electrical energy	[0.1, 1.0]
Bandwidth	[100, 1000]
Computing power	[100, 1000]
RAM	[100, 2000]
Energy price	[1, 100]

experimental evaluation of the system. In the experiments, the cost of cloud resource and the energy are expressed in dollars that can be defined as unit resource or energy processing cost. The initial price of electrical energy is set from 1 to 100 dollars. The initial price of VM is set from 10 to 500 dollars. Cloud users submit their jobs with varying deadlines. The deadlines of cloud users are chosen from 100 to 400 ms. The budgets of cloud users are set from 100 to 1500 dollars. The simulation results shown in the figures represent mean values. Simulation parameters are listed in Table 1. Each experiment is repeated 6 times and 95 % confidence intervals are obtained. The simulation results shown in the figures represent mean values.

The experiments aimed at comparing our *RASP* with dynamic resource provisioning algorithm [11] proposed by Qian Zhu and Gagan Agrawal. In [11], the authors use a fine-grained pricing model where a higher allocation of CPU cycle percentage or memory is associated with a higher cost for each time unit. They use two different pricing models: a linear pricing model and an exponential pricing model. Their price-based dynamic resource provisioning algorithms are denoted as *DRP_linear pricing* and *DRP_exponential pricing*. To evaluate the performance of our *RASP* against dynamic resource provisioning algorithm [11], we adopt the metrics: resource cost, ex-

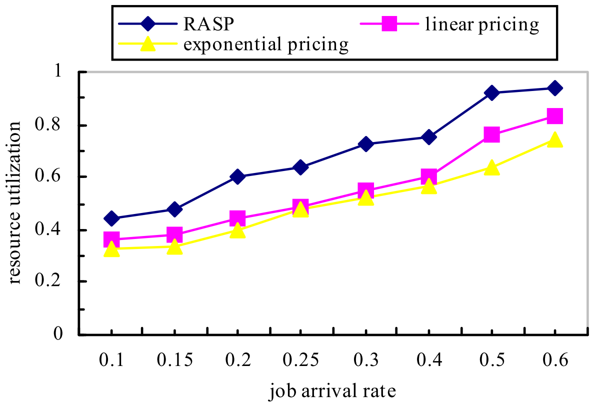


Fig. 2 Resource utilization at different job arrival rate

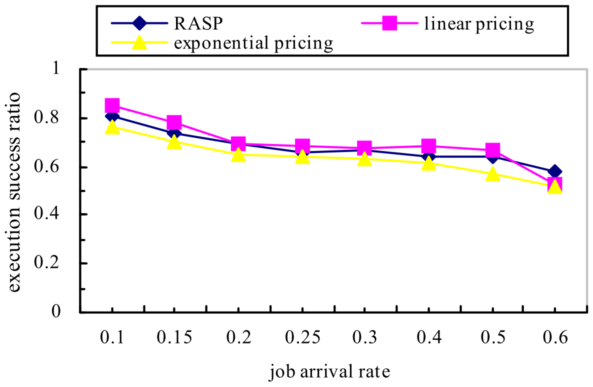
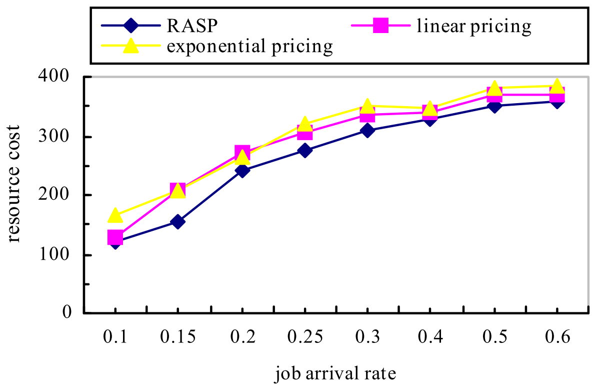


Fig. 3 Execution success ratio at different job arrival rate

ecution success ratio and resource utilization. *Resource cost* which is used in [11] is the price charged for the CPU cycles as well as memory used by the execution of the application. In the experiment, some simulation data is selected according to the ref. [11] which was our compared work. We construct the VM types by examining the maximum resource usage of CPU and memory and match them with the available VM types offered by SpotCloud [22], a cloud computing company that provides a commodity-based market for trading computing resources. In the experiment, some metrics are designed to test the performance of cloud users, such as execution success ratio and user satisfaction ratio. Other metrics are used to test the performance of IaaS provider, such as the resource utilization and resource cost. We compare our method with the compared method which only considers cloud user's benefit and maximizes application QoS.

The impacts of job arrival rate on resource cost, resource utilization and execution success ratio are illustrated in Figs. 2, 3, 4, respectively. As Fig. 2 shows, as job arrival rate (a) increases, resource utilization ratio increases. When $a = 0.4$, the resource utilization of *RASP* is as much as 16 % more than utilization *DRP_exponential pricing*. When job arrival rate is very large, many jobs are sent to cloud system; cloud resources of IaaS provider are busier. Under low job arrival rate ($a = 0.1$), the resource utilization of *RASP* is as much as 9 % larger that *DRP_linear pricing*. When

Fig. 4 Resource cost at different job arrival rate



increasing the job arrival rate by $a = 0.4$, the resource utilization of *DRP_linear pricing* is as much as 14 % less than that using the *RASP*. Figure 3 shows that the execution success ratio decreases when job arrival rate increases. When $a = 0.6$, execution success ratio of *RASP* is as much as 13 % lower than that by $a = 0.10$. The smaller is a, sufficient VMs are available for SaaS providers. The requirements of the cloud users can be processed on time and these cloud users experience higher user satisfaction. On the other hand, the larger is a, the lower is execution success ratio. When job arrival rate reaches 0.2 ($a = 0.20$), the execution success ratio of *DRP_linear pricing* is as much as 4 % more than that using *RASP*. When job arrival rate increases, execution success ratio of *DRP_linear pricing*, *DRP_exponential pricing* and *RASP* deteriorates. When the job arrival rate increases, fewer requests from cloud users can be accepted by the cloud system due to the increase of system burden; so, fewer requests from cloud users can be executed successfully before their deadline. But under the same job arrival rate, execution success ratio of *DRP_linear pricing* is slightly better than *RASP*. Figure 4 is to show the effect of job arrival rate on resource cost. When increasing the job arrival rate, the impact on the resource cost is obvious. A higher job arrival rate brings out higher resource cost. With the same job arrival rate, *DRP_linear pricing* has higher resource cost than *RASP*. *RASP* considers the benefit of the cloud users and IaaS provider. Cloud users calculate suitable payments for SaaS services and maximize the profit function. The *RASP* considers the trade-off between profits of IaaS providers and the cloud users. *DRP_linear pricing* algorithm adopts a feedback-control-based approach for maximizing application QoS, while meeting both the time constraint and the resource budget limit. When job arrival rate increases ($a = 0.3$), resource cost of *RASP* is as much as 9 % less than *DRP_linear pricing*.

The impacts of price increase ratio on user satisfaction ratio, the revenue for IaaS providers, execution success ratio and resource utilization are illustrated in Figs. 5, 6, 7, 8, respectively. User satisfaction ratio measures the level of satisfying cloud user's requests. It is computed as the proportion of cloud user's jobs that required QoS to be fulfilled out of all cloud user's jobs. When p is set from 0.1 to 0.8, the smaller is p, the earlier the system reaches the steady state and the higher is the user satisfaction as shown in Fig. 5. Since, when price of SaaS service becomes high, cloud users will afford more payment to obtain the SaaS service, some tasks of cloud

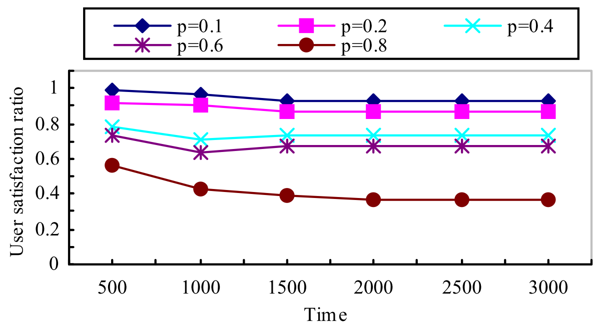


Fig. 5 User satisfaction ratio vs. price increase ratio

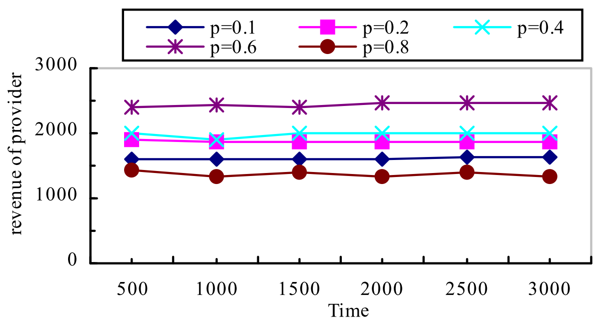


Fig. 6 Revenue of IaaS provider vs. price increase ratio

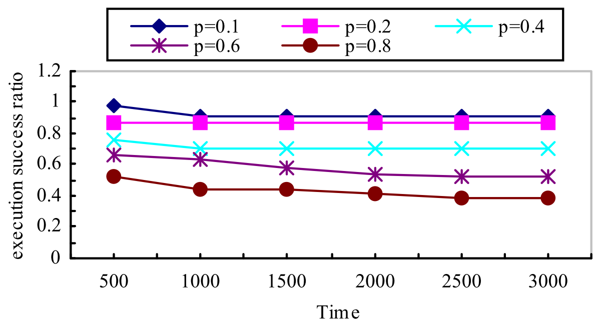


Fig. 7 Execution success ratio vs. price increase ratio

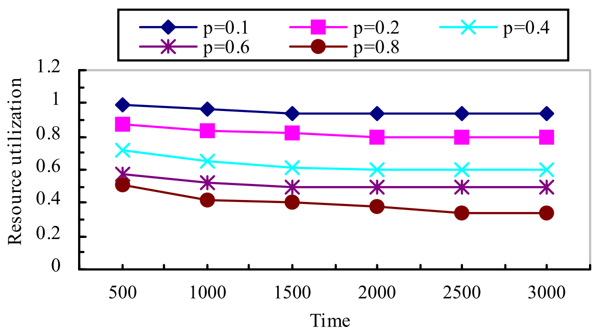


Fig. 8 Resource utilization vs. price increase ratio

users cannot be completed before their deadlines. Increasing price quickly leads to situations when some cloud users with low budget cannot be satisfied to fulfill their achievements. From the results of Fig. 6 we can find that there exists a price increase ratio ($p = 0.6$) to maximize the revenue of IaaS provider. The price variations take effect on the revenue of IaaS providers by means of the cloud user's reaction depicted in *RASP* by calculating its SaaS software service demand according to optimal payments for SaaS provider to maximize cloud user's utility function. There is also related with the SaaS provider's decisions in allocating SaaS service to the cloud users. SaaS provider will pay more to IaaS providers for cloud resources. The maxim of the curve is the optimal revenue point for the IaaS provider. The highest value of the revenue is determined by both acceptable price and cloud resource allocation. Considering the execution success ratio, the results of Fig. 7 show that when increasing p, the execution success ratio becomes lower. Increasing prices of IaaS provider will prevent cloud users from being processed. When increasing price increase ratio by $p = 0.4$, the execution success ratio is as much as 21 % less than that with $p = 0.1$. When $p = 0.8$, the execution success ratio reduces to nearly 45 %. Considering the resource utilization, from the results in Fig. 8, as price increase ratio is higher, the resource utilization becomes lower. When $p = 0.60$, the resource utilization is as much as 28 % less than utilization by $p = 0.2$. Since, when the price increases quickly, the cloud users will afford more payment to obtain the cloud resource, some cloud users with low budget cannot buy the cloud resources.

6 Conclusions

The paper studies efficient resource allocation to optimize objectives of cloud users, IaaS provider and SaaS provider in cloud computing. The paper proposes the composition of different layers in the cloud such as IaaS and SaaS and its joint optimization for efficient resource allocation. The efficient resource allocation optimization problem is conducted by subproblems. The proposed cloud resource allocation optimization algorithm is achieved through an iterative algorithm. The experiments are conducted to compare the performance of proposed joint optimization algorithm for efficient resource allocation with other related works. From the simulation results, the resource utilization of *RASP* is higher than *DRP_linear pricing* and *DRP_ exponential pricing*. When job arrival rate is 0.4, the resource utilization of *RASP* is as much as 16 % more than utilization *DRP_exponential pricing*. But, the execution success ratio of *RASP* is not better than some compared algorithm; when job arrival rate reaches 0.2, the execution success ratio of *DRP_linear pricing* is as much as 4 % more than *RASP*. *RASP* jointly considers both the benefits of the cloud users and cloud providers.

Acknowledgements The work was supported by the National Natural Science Foundation (NSF) under Grants Nos. 61272116 and 61171075, Specialized Research Fund for the Doctoral Program of Higher Education under Grant No. 20120143110014, National Key Basic Research Program of China (973 Program) under Grant No. 2011CB302601, and the Open Fund of the State Key Laboratory of Software Development Environment. Any opinions, findings, and conclusions are those of the authors and do not necessarily reflect the views of the above agencies.

References

1. Gulati A, Shanmuganathan, G, Holler A (2011) Cloud scale resource management: challenges and techniques. In: USENIX HotCloud, Portland
2. Wu L, Garg SK, Buyya R (2011) SLA-based resource allocation for a software as a service provider in cloud computing environments. In: Proceedings of the 11th IEEE/ACM international symposium on cluster computing and the grid (CCGrid 2011), Los Angeles, USA, 23–26 May 2011
3. Chaisiri S, Lee B, Niyato D (2012) Optimization of resource provisioning cost in cloud. IEEE Trans Serv Comput 5(2):164–177
4. Rimal BP, Choi E (2012) A service-oriented taxonomical spectrum, cloudy challenges and opportunities of cloud computing. Int J Commun Syst 25(6):796–819
5. Randles M, Lamb D, Taleb-Bendiab A (2010) A comparative study into distributed load balancing algorithms for cloud computing. In: IEEE 24th international conference on advanced information networking and applications workshops, pp 551–556
6. Jayasinghe D, Pu C, Eilam T, Steinder M, Whally I, Snible E (2011) Improving performance and availability of services hosted on IaaS clouds with structural constraint-aware virtual machine placement. In: Proceedings of the IEEE international conference on services computing, Washington, USA. IEEE Press, New York, pp 72–79
7. Mohd Yusoh Z, Tang M (2010) A cooperative coevolutionary algorithm for the composite SaaS placement problem in the cloud. In: Proceedings of the neural information processing, theory and algorithms, pp 618–625
8. Kwok T, Mohindra A (2008) Resource calculations with constraints, and placement of tenants and instances for multi-tenant SaaS applications. In: Sixth international conference on service-oriented computing, Sydney, Australia. Springer, Berlin, pp 633–648
9. Garg SK, Yeo CS, Anandasivam A, Buyya R (2011) Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers. J Parallel Distrib Comput 71(6):732–749
10. Kouki Y, Ledoux T, Sharrock R (2011) Cross-layer SLA selection for cloud services. In: First international symposium on network cloud computing and applications, pp 143–147
11. Zhu Q, Agrawal G (2012) Resource provisioning with budget constraints for adaptive applications in cloud environments HPDC. In: Proceedings of the 19th ACM international symposium on high performance distributed computing, pp 304–307
12. Addis B, Ardagna D, Panicucci B (2010) Autonomic management of cloud service centers with availability guarantees. In: IEEE 3rd international conference on cloud computing, pp 220–227
13. Saure D, Sheopuri A, Qu H, Jamjoom H, Zeevi A (2010) Time-of-use pricing policies for offering cloud computing as service. In: IEEE SOLI 2010, pp 300–305
14. Rodero I, Jaramillo J, Quiroz A, Parashar M, Guim F (2010) Energy-efficient application-aware online provisioning for virtualized clouds and data centers. In: International conference on green computing (GREENCOMP ’10)
15. Abdelsalam HS, Maly K, Kaminsky D (2009) Analysis of energy efficiency in clouds. In: Computation world: future computing, service computation, cognitive, adaptive, content, patterns, pp 416–422
16. Younge AJ, von Laszewski G, Wang L (2012) Efficient resource management for cloud computing environments. In: IEEE international green computing conference (IGCC), pp 357–364
17. Chang F, Ren J, Viswanathan R (2010) Optimal resource allocation in clouds. In: IEEE 3rd international conference on cloud computing, pp 418–425
18. Ferretti S, Ghini V, Panzieri F, Pellegrini M, Turrini E (2010) QoS–aware clouds. In: IEEE 3rd international conference on cloud computing, pp 321–328
19. Lee YC, Wang C, Zomaya AY, Zhou BB (2010) Profit-driven service request scheduling in clouds. In: 10th IEEE/ACM international conference on cluster, cloud and grid computing, pp 15–24
20. Teng F, Magoulès F (2010) Resource pricing and equilibrium allocation policy in cloud computing. In: 10th IEEE international conference on computer and information technology (CIT 2010), pp 195–202
21. Yazır YO, Matthews C, Farahbod R (2010) Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis. In: IEEE 3rd international conference on cloud computing, pp 91–98
22. Spotcloud (2012) Cloud capacity clearing house/spot market. Home, http://www.spotcloud.com

23. Kelly F, Maulloo A, Tan D (1998) Rate control for communication networks: shadow prices, proportional fairness and stability. J Oper Res Soc 49(3):237–252
24. Luh PB, Hoitomt DJ (1993) Scheduling of manufacturing systems using the Lagrangian relaxation technique. IEEE Trans Autom Control 38(7):1066–1079
25. Kuhn HW, Tucker AW (1951) Nonlinear programming. In: Proceedings of 2nd Berkeley symposium. University of California Press, Berkeley, pp 481–492