Motavaselalhagh *et al. Human-centric Computing and Information Sciences* (2015) 5:16
DOI 10.1186/s13673-015-0031-4

Human-centric Computing and Information Sciences
a SpringerOpen Journal

RESEARCH Open Access

Knowledge-based adaptable scheduler for SaaS providers in cloud computing

Farzaneh Motavaselalhagh[1*], Faramarz Safi Esfahani[1] and Hamid Reza Arabnia[2]

* Correspondence: farzane.mgh@gmail.com
[1]Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Isfahan, Iran
Full list of author information is available at the end of the article

Abstract

Software as a Service (SaaS) in Cloud Computing offers reliable access to software applications for end users over the Internet without direct investment in infrastructure and software. SaaS providers utilize resources of internal datacenters or rent resources from a public Infrastructure as a Service (IaaS) provider in order to serve their customers. Internal hosting can increase cost of administration and maintenance, whereas hiring from an IaaS provider can impact quality of service due to its variable performance. To surmount these challenges, we propose a knowledge-based admission control along with scheduling algorithms for SaaS providers to effectively utilize public Cloud resources in order to maximize profit by minimizing cost and improving customers' satisfaction level. In the proposed model, the admission control is based on Service Level Agreement (SLA) and uses different strategies to decide upon accepting user requests for that minimal performance impact, avoiding SLA penalties that are giving higher profit. However, because the admission control can make decisions optimally, there is a need of machine learning methods to predict the strategies. In order to model prediction of sequence of strategies, a customized decision tree algorithm has been used. In addition, we conducted several experiments to analyze which solution in which scenario fit better to maximize SaaS provider's profit. Results obtained through our simulation shows that our proposed algorithm provides significant improvement (up to 38.4 % cost saving) compared to the previous research works.

Keywords: Cloud computing; Service Level Agreement (SLA); Admission control; Software as a service; Scalability of application services; Knowledge-based; Data mining; Decision tree algorithm

Introduction

Cloud computing has been recognized as one of the new prominent computing paradigms. The ability of cloud to provide on demand access to software, application platforms and infrastructure in the form of scalable services, has attracted considerable interest in academic communities and difference industries. It can be viewed as the transformation into reality of a long held dream called "Utility Computing", it also emerged into the market with a huge potential to fulfill this dream. In its fold, companies do not even need to plan for their IT growth in advance with this new "pay as you go" system. The Cloud model is cost-effective because customers pay for their actual usage without upfront costs, and scalable because it can be used more or less depending on the customers' needs. A set of applications is managed and hosted externally by a third partner and it is delivered over a secure high quality network. It is

also available anywhere with an internet connection, even when on the move. Cloud computing is an internet technology that utilizes both central remote servers and internet to manage the data and applications. This technology allows many businesses and users to use the data and application without an installation. Users and businesses can access the information and files at any computer system having an internet connection. Generally, according to the most popular cloud service providers, for example Microsoft [3], Amazon [7] and salesforce [4]; Cloud-based services can be categorized as: Application (Software as a Service-SaaS), Platform (Platform as a Service-PaaS) and Hardware resources (Infrastructure as a Service-IaaS).

Here, we focus on the SaaS layer that allows end users to reliably access applications over the Internet without the burden of software related cost and annoying effort (such as software licensing and upgrade). The primary objective of SaaS providers is to minimize cost and maximize Customer Satisfaction Level (CSL). The above mentioned cost includes the administration operation cost, infrastructure cost and, finally, penalty cost incurred by SLA violations. CSL depends on the degree SLA is satisfied. In general, SaaS providers utilize internal resources of their own datacenters or rent additional resources from another specific IaaS provider. Internal hosting can create administration and maintenance cost, while renting resources from a single IaaS provider can impact the service quality offered to SaaS customers due to the variable performance. To overcome the above constraints, multiple IaaS providers are considered over here and since acquiring resources from multiple IaaS providers lends a large amount of resources with different usage policies, price models, performance patterns and availability to satisfy Service Level Agreement (SLA), so an admission control has been used as a general mechanism to avoid overloading of resources and violation of SLAs. Most current SaaS providers do not contain admission control mechanism and their method of scheduling is not known publicly. Thus, the following facts need to be considered to allow efficient use of resources that is offered by multiple IaaS providers, where the resources can be dynamically expanded and reduced on demand. The facts are: 1) accepting new requests without impacting accepted requests, 2) mapping various user requests with different QoS parameters to VMs, 3) deciding upon whether the new request should be assigned to available resources or new VM must be initiated. In some existing systems such as [14, 11, 18], they proposed mechanisms which considered this facts but there are some drawbacks in their methods which must be solved. For example in [14] they use various profit maximization algorithms such as maximizing the profit by minimizing the number of VMs (ProfminVM), maximizing the profit by rescheduling (ProfRS), maximizing the profit by exploiting the penalty delay (ProfPD). In admission control phase of these algorithms, they use the following four strategies for request acceptance:

1. Initiate new VM strategy

   In this strategy, first checks for each type of VMs in each resource provider in order to determine whether the deadline of new request is long enough comparing to the estimated finish time. The estimated finish time depends on the estimated start time, request processing time, and VM initiation time. If the new request can be completed within the deadline, the investment return is calculated. If there is value added according to the investment return, and then all related

information (such as resource provider ID, VM ID, start time and estimated finish time) are stored into the potential schedule list.

2. Wait strategy
   In this strategy, first verifies each VM in each resource provider if the deadline time of new request is enough to wait all accepted requests in vm_{il} to complete. If new request can wait for all accepted requests to complete, then the investment return is calculated and the remaining steps are the same as those in initiate new VM strategy.
3. Insert strategy
   In this strategy, first checks verifies if any accepted request u^k according to latest start time in vm_{il} can wait the new request to finish. If there is an already accepted request u^k that is able to wait for the new user request to complete, the strategy checks if the new request can complete before its deadline. If so, unew gets priority over u^k, then the algorithm calculates the investment return and the remaining steps are the same as those in initiate new VM strategy.
4. Penalty delay strategy
   In this strategy, first checks if the new user request's budget is enough to wait for all accepted user requests in vm_i to complete after its deadline, and then the investment return is calculated and the remaining steps are the same as those in initiate new VM strategy.

Between their proposed algorithms, the "ProfPD" algorithm is more efficient. The Pseudo-code of this algorithm is showed in Fig. 1.

One of the drawbacks in [14] is strength of the algorithms by handling errors in dynamic scenario of cloud environment, another drawback is that their algorithms used a static sequence of strategies for each request in each resource provider to check whether new request can be accepted or not. While, maybe, running another sequence of strategies for a request can have a better answer. For example, "Initiate new VM" strategy in resource provider 1 is runnable for the new request and for this request in resource provider 2, "Wait" strategy is runnable too. So in this situation, if the "ProfPD" algorithm runs, the first solution is selected for the new request, but is this really the best solution? Which of them is the best? So this algorithm cannot always have an optimal solution. In [18] there is a drawback too.

They used a machine learning method to predict the strategies and produce the according resources but since the status of each resource provider is not static all time and in a period of time, the predicted resource provider may not be available so this method cannot be good in every time, too.

In this paper, we provide a solution to the above problems by proposing an adaptive and cost-effective knowledge-based admission control technique and scheduling algorithm to maximize the profit of SaaS provider. This proposed solution is aimed to maximize the number of efficient placement of user requests on VMs rented from several IaaS providers. We consider various customers' QoS requirements and heterogeneity of infrastructure.

The key contributions of this study can be considered as follows: 1) a system model is provided for SaaS providers to satisfy customer's requirements, 2) an admission control and scheduling algorithm for maximizing the SaaS provider's thereby minimizing cost and maximizing CSL is proposed, 3) a Data Mining method such as Microsoft Decision Tree algorithm to train up the system for dynamic scenario improving the

Pseudo-code for *ProfPD* algorithm

```
Input: New user's request parameters (u_new), expInvRet_ij^new
Output: Boolean
Functions:
AdmissionControl( ) {
1.      If ( there is any initiated VM ) {
2.          For each vm_i in resource provider rp_j {
3.                  If (! canWait ( u_new, vm_i ) ) {
4.                      If (! canInsert ( u_new, vm_i ) ) {
5.                          If (! canInitiateNew(u_new, rp_j))
6.                              continue;
7.                          If (! canPenaltyDelay(u_new, rp_j))
8.                              continue;
9.                      }
10.                 }
11.         }
12.     }
13.     Else If (! canInitiateNew(u_new, rp_j))
14.         Return reject
15.     If (PotentialScheduleList is empty)
16.         Return reject
17.     Else { Get the max[ret_ij^new, SD_ij ] in PotentialScheduleList
18.             If ( max(ret_ij^new) = expInvRet_ij^new )
19.                 Return accept
20.             Else
21.                 Return reject
22.     }
    }

schedule ( ) {
23. Get the [ret_max^new, SD_max ] in maxRet(PotentialScheduleList)
24. If ( SD_max is initiateNewVM)
25. initiateNewVM in rp_j
26. Schedule the u^new in VM_max in rp_max according to SD_max.
}
```

Fig. 1 The Pseudo-code of "ProfPD" algorithm [14]

performance rate of the system is utilized, 4) the system is evaluated, 5) conclusion and future direction are discussed.

Related works

Research on market driven resource allocation and admission control has started as early as 1981 [14]. Most of the market-based resource allocation methods are either non-profit-based [13] or designed for fixed number of resources. Fig. 2, showed a category of the resource management techniques.

In Cloud, IaaS providers focusing on maximize profit and many works [9, 8, 2] proposed market based scheduling approaches. For instance, Amazon [10] introduced spot instance way for customers to buy those unused resources at bargain prices. This is a way of optimizing resource allocation if customers are happy to be terminated at any

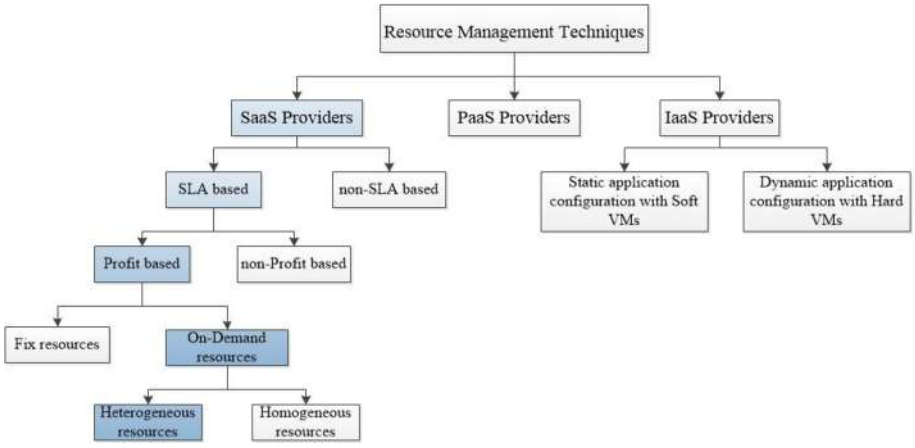

Fig. 2 A category of the resource management techniques

time. However, our goal is not only to maximize profit but also satisfy the SLA agreed with the customer. However, research at the SaaS provider level is still in its infancy, because many works do not consider maximizing profit and guaranteeing SLA with the leasing scenario from multiple IaaS providers, where resources can be dynamically expanded and contracted on demand. Yeo and Buyya in [1] presented algorithm to handle penalties in order to enhance the utility of the cluster based on SLA, although they have outlined a basic SLA with four parameters in cluster environment, multiple resources and multiple QoS parameters from both user and provider sides are not explored. Jaideep and Varma in [6] proposed learning-based admission control in Cloud computing environments. Their work focuses on the accuracy of admission control but does not consider software service providers' profit. Bichler and Setzer in [7] proposed an admission control strategy for media on demand services, where the duration of service is fixed. Our approach allows a SaaS provider to specify its expected profit ratio according to the cost, for example; the SaaS provider can specify that the service request, which can increase the profit in two times, will be accepted. Popovici et al in [4], mainly focused on QoS parameters on resource provider's side such as price and offered load. However, our work differs on QoS parameters from both users' and SaaS providers' point of view, such as budget, deadline, and penalty rate. In [2] they first establish a cloud service request model with SLA constraints and then present a new optimization algorithm for profit driven service request scheduling based on dynamic reuse, which takes account of the personalized SLA characteristics of user requests and current system workload. Their proposed algorithm constructs an on demand resource pool of dynamic virtual machines, attains optimal cloud service request scheduling in sensible time and thus considerably reduces operational costs of cloud service providers thereby increase profits of CSPs. In their model, one resource provider is used and their work is not price-based.

Linlin et al. in [14] propose an innovative admission control and scheduling algorithm for SaaS providers to effectively utilize public Cloud resources to maximize profit by minimizing cost and improving customer satisfaction level. In their system, they use various profit maximization algorithms such as Maximizing the profit by minimizing the number of VMs (ProfminVM), Maximizing the profit by rescheduling (ProfRS),

Maximizing the profit by exploiting the penalty delay (ProfPD). Between these algorithms, the algorithm "ProfPD" is more efficient, but this algorithm is not optimized because it always use a static sequence of strategies for each request in each IaaS provider for request acceptance. While, running another sequence of strategies for a request may have better answer. So we use a knowledge-based admission control and scheduling algorithms in order to cause more efficiently and can provide substantial improvement. Our admission control uses knowledge process to decide which sequence of strategies can be run for a new request to checking it can be accepted or not. Mohana et al. in [18] present a dynamically adaptable admission control and scheduling algorithms for efficient resource allocation to maximize profit and CSL for SaaS providers. They use a machine learning technique such as SVM and Artificial Neural Network (ANN) to train up the system for dynamic scenario improving the performance rate of the system. They use the same strategies of us, In their work, the machine learning method used to predict the strategies and produce the according resources but since the status of each resource provider is not static in all time and the predicted resource provider may not be available in a period of time, so this method cannot be good in every time too. In the Table 1, the summary of related works are presented.

System Model

In this section, a model of the SaaS provider will be introduced which consists of the actors and admission control and scheduling and knowledge process components. In this model, the name of "adaptable scheduler", refers to the last three components depicted in Fig. 3. The actors are users, SaaS providers, and IaaS providers. The system consists of application layer and platform layer functions. Users request the software from a SaaS provider by submitting their QoS requirements. The platform layer uses admission control to interpret and analysis the user's QoS parameters and decides whether to accept or reject the request based on the capability, availability and price of VMs. Admission control decisions are based on its knowledge, which gathers from knowledge process component. Then, the scheduling component is responsible for allocating resources based on admission control decision and sending decision results to knowledge process to be saved in knowledge database.

Actors

The participating actors involved in the process are discussed below along with their objectives and constraints:

User: On users' side, a request for an application is sent to a SaaS provider's application layer with QoS constraints, such as, deadline, budget and penalty rate. Then, the platform layer utilizes the admission control and scheduling algorithms to admit or reject this request. If the request is accepted, a formal agreement (SLA) will be signed between both parties to guarantee QoS requirements such as response time, etc. The SLA with Users includes the following properties:

- Deadline: Maximum time user would like to wait for the result.
- Budget: How much user is willing to pay for the requested services.

Table 1 The summery and comparison of the related work in cloud computing area

Related works	Admission control	Resource management (scheduling)	Profit driven	Resource characteristics	Knowledge based	Price-based	SLA oriented	QoS	
								For users	For saas provider
Reig,.et al. [17]	Yes	Yes	No	NA	No	No	Yes	Deadline	Nothing
Bicher, Setzer [7]	Yes	No	Yes	NA	No	No	Yes	Budget	Nothing
Wu et al. [14]	Yes	Yes	Yes	Multiple resource provider	No	No	Yes	Deadline, Budget, Request Length, Penalty rate	Vm Initiation time, Data Transfer Time
Wu. et al. [15]	No	Yes	Yes	Multiple Resource Provider	No	No	Yes	Request type, product type, account type, contract length, number of accounts	VM Type, Product Type, Account Type
Zhipiao Liu, Qibo Sun [16]	No	No	Yes	One resource provider	No	No	Yes	Budget, Deadline	VM Type, Vm Price
N. Ani Brown [11, 20]	Yes	Yes	Yes	Multiple resource provider	No	Yes	Yes	Deadline, Budget, Request Legnth, Penalty Rate	Vm Initiation time, Data Transfer Time
Mohana. et al. [18]	Yes	Yes	Yes	Multiple resource provider	No	Yes	Yes	Deadline, Budget, Request Legnth, Penalty Rate	Vm Initiation time, Data Transfer Time
Choi and Lim [19]	No	No	Yes	One resource provider	No	Yes	Yes	Nothing	total expence
Our Work	Yes	Yes	Yes	Multiple resource provider	Yes	No	Yes	Deadline, Budget, Request Legnth, Penalty Rate	Vm Initiation time, Data Transfer Time

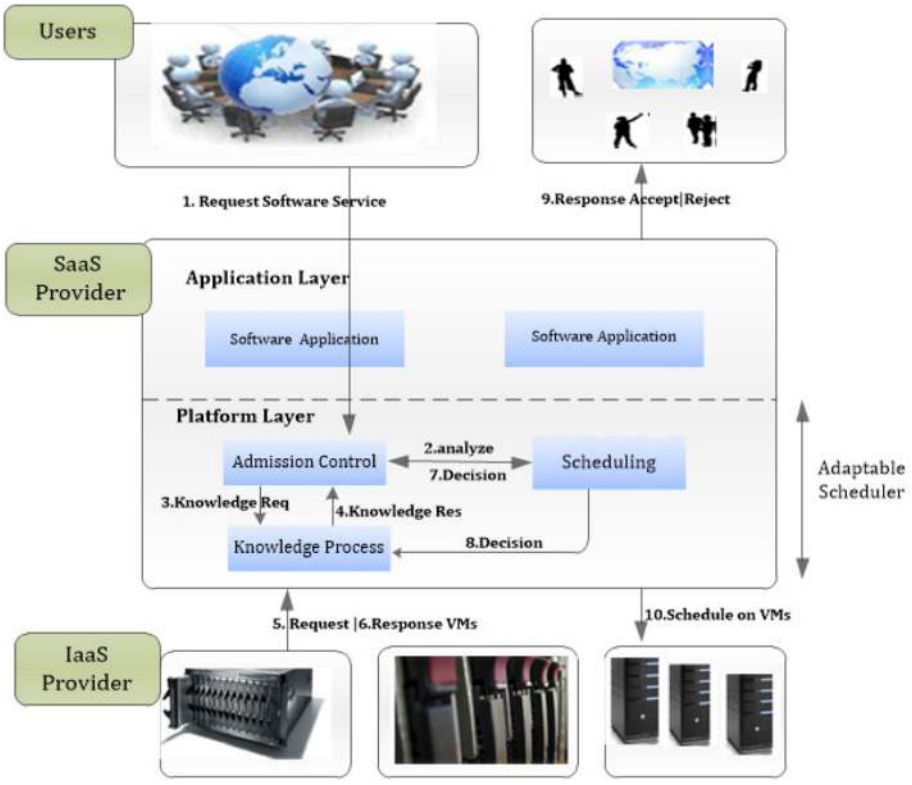

Fig. 3 A high level KBRP system for application service scalability using multiple IaaS providers in Cloud

- Penalty Rate Ratio: A ratio for consumers' compensation if the SaaS provider misses the deadline.
- Input File Size: The size of input file provided by users.
- Request Length: How many Millions of Instructions (MI) are required to be executed to serve the request?

SaaS provider: a SaaS provider rents resources from IaaS providers and leases software as services to users. SaaS providers aim at minimizing their operational cost by efficiently using resources from IaaS providers, and improving Customer Satisfaction Level (CSL) by satisfying SLAs, which are used to guarantee QoS requirements of accepted users. From SaaS provider's point of view, there are two layers of SLA with both users and resource providers. It is important to establish two SLA layers, because the SLA with users can help the SaaS provider to improve the customer satisfaction level by gaining users' trust of the quality of service; SLA with resource providers can enforce resource providers to deliver the satisfied service. If any party in the contract violates its terms, the defaulter has to pay for the penalty according to the clauses defined in the SLA.

IaaS provider: an IaaS provider offers VMs to SaaS providers and is responsible for dispatching VM images to run on their physical resources. The platform layer of SaaS provider uses VM images to create instances. It is important to establish SLA with a resource provider, because it enforces the resource provider to guarantee service quality. Furthermore, it provides a risk transfer for SaaS providers, when the terms are violated by resource providers. In this work, the compensation given by the resource

provider is not considered because 85 % resource providers do not in fact currently provide penalty enforcement for SLA violation [12]. The SLA with IaaS providers includes the following properties:

- Service Initiation Time: How long it takes to deploy a VM.
- Price: How much a SaaS provider has to pay per hour for using a VM from a resource provider?
- Input Data Transfer Price: How much SaaS providers has to pay for data transfer from local machine (their own machine) to resource provider's VM.
- Output Data Transfer Price: How much a SaaS provider has to pay for data transfer from resource provider's VM to local machine.
- Processing Speed: How fast can the VM process. Machine Instruction per Second (MIPS) of a VM as processing speed is used.
- Data Transfer Speed: How fast the data is transferred. It depends on the location distance and the network performance.

Adaptable Scheduler

The adaptable scheduler is used to analyze whether or not a new request can be accepted based on the QoS requirements and resource capabilities. This scheduler contains the components such as admission control, scheduling and knowledge process. The admission control uses different strategies to decide which of the users' requests can be accepted in order to cause minimal performance impact, avoiding SLA penalties. For each new request, it sends the request's characteristic to knowledge process, then the knowledge process uses a decision tree algorithm to decide which sequence of strategies should be run for this request for getting maximum profit for SaaS provider and then returns the best sequence's number of strategies to the admission control. Afterward, the admission control based on the result returned from the knowledge process, runs strategies for the request. The scheduling part of this system determines where and which type of VMs will be used by incorporating the heterogeneity of IaaS providers in terms of their price, service initiation time, and data transfer time. After that, it will send the result of scheduling to the knowledge process to be stored in the knowledge base.

Algorithm and strategies

In this section, we use the four strategies which is presented in [14] to analyze whether a new request can be accepted or not based on the QoS requirements and resource capability and an algorithm has been introduced that utilizes these strategies based on its knowledge to allocate resources. In this algorithm, the admission control part uses different strategies for each request to decide which user requests would be accepted. The knowledge process component forecasts which sequence of these strategies should be run for each request and it is contributed to the admission control component to run the best sequence, and finally the scheduling component is responsible for allocating resources and scheduling based on the admission control result.

Strategies

The strategies that admission control uses to decide which of the users' requests can be accepted are introduced as follows: 1) Initiate new VM 2) Wait Strategy 3) Insert

Before strategy and 4) Penalty Delay Strategy. Inputs of all strategies are QoS parameters of the new request and resource providers' related information. Outputs of all strategies are admission control and scheduling related information, for example, which VM and in which resource provider the request can be scheduled. In our algorithm, first strategy is represented as "canInitiateNewVM ()" and second strategy is represented as "canWait ()" and third strategy is represented as "canInsertBefor ()" and finally, fourth strategy is represented as "canPenaltyDelay ()".

Different Sequences of Strategies

In this section, we will present three different sequences of strategies which are used by admission control for user requests. Three different sequences of strategies in each resource provider is depicted in Fig. 4.

As seen in Fig. 4, in sequence1, in each resource provider, four strategies are checked to find the best strategy to processing the new request, in other words, in the first resource provider, first it checks if the new request can wait all accepted requests to complete in any initiated VM in this resource provider, If the request cannot wait, then it checks if the new request can be inserted before any accepted request in any already initiated VM, otherwise the algorithm checks if the new request can be accepted by initiating a new VM provided by any resource provider or by delaying the new request with penalty compensation; if a strategy is found, checking is stopped and the result will pass to

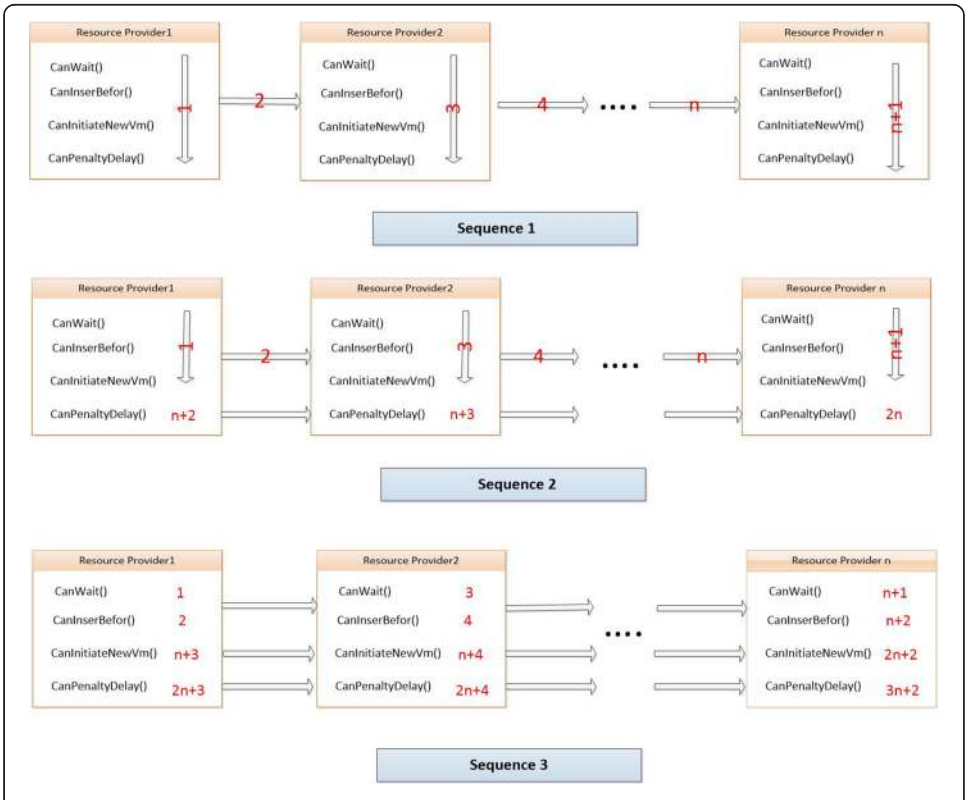

Fig. 4 Three sequence of running strategies in resource providers

scheduling part, but if no strategy is found, the same scenario runs in the next resource provider to find a suitable strategy. If a SaaS provider does not make sufficient profit by any strategy, the new request will be rejected. Otherwise, the request is accepted and scheduled based on the entry in "PotentialScheduleList" which gives the maximum return.

In sequence 2, in each resource provider for each VM type, the first three strategies (canWait (), canInsertBefor () and canInitiateNewVM ()) are checked to find the best strategy for processing the new request. If a strategy is found, checking is stopped and the result will pass to scheduling part, but if it ends up in not finding any strategies, it will check the last strategy (canPenaltyDelay ()) in each resource provider and so on.

In sequence3, in each resource provider, the first strategy (canWait ()) is checked, if it is suitable for processing the new request in any resource providers, the second strategy is checked in each resource provider and so on.

Proposed Algorithm

A service provider can maximize the profit by reducing the infrastructure cost, which depends on the number and type of initiated VMs in IaaS providers' datacenter. Therefore, our algorithms are designed in a way that minimize the number of VMs by maximizing the utilization of already initiated VMs. In this section, based on the above strategies, we propose an algorithm, which is a knowledge-based Profit Maximization (KBPM). Algorithm 1 in Fig. 5, describes the knowledge-Based Profit Maximization algorithm, which involves three main phases: 1) admission control, 2) knowledge process, and 3) scheduling.

In the admission control phase, the new user request will be checked if it can be processed by any strategies. Hence, firstly, it sends the new request's characteristics to knowledge process for getting the sequence' number of running strategies. When the value is returned from the knowledge process, based on the returned result, it runs strategies. For example if the returned value is equal to '1', firstly the algorithm checks if the new request can wait all accepted requests to complete in any initiated VM – invoking Wait Strategy. If the request cannot wait, then it checks if the new request can be inserted before any accepted request in any already initiated VM – using InsertBefor Strategy. Otherwise the algorithm checks if the new request can be accepted by initiating a new VM provided using Initiate New VM Strategy or by delaying the new request with penalty compensation – using Penalty Delay Strategy.

If a SaaS provider does not make sufficient profit by any strategy, the algorithm rejects the new request.

Otherwise, the request is accepted and scheduled based on the entry in PotentialScheduleList which gives the maximum return.

The scheduling phase is the actual resource allocation and scheduling based on the admission control result; if the algorithm accepts the new request, the algorithm first finds out in which IaaS provider (rp_j) and which VM (vm_i) a SaaS provider can gain the maximum investment return by extracting information from PotentialScheduleList . If the maximum investment return is gained by initiating a new VM, then the algorithm initiates a new VM in the referred resource provider (rp_j), and schedule the request to it. Finally, the algorithm schedules the new request on the referred VM (vm_i).

It should be noted that in this algorithm supposed, the available IaaS providers are ordered by distance. In other words, the IaaS Provider with a minimum distance is consider the first provider.

Algorithm 1 : Pseudo-code for Knowledge-based *Profit Maximization* Algorithm

Input: New user's request parameters (u^{new}), $expInvRet_{ij}^{new}$
Output: Boolean
Functions:
AdmissionControl( ){

```
1.    Way= GetWayNumberFromKnowledgeProcess(u^new)
2.    If(way=1){
3.        Run sequence1(u^new);
4.    }
5.    Else If(way=2){
6.        Run sequence2 (u^new);
7.    }
8.    Else If(way=3){
9.        Run sequence3(u^new);
10.   }
11.   If (PotentialScheduleList is empty)
12.           Return reject
13.   Else { Get the max[ret_ij^new, SD_ij ] in PotentialScheduleList
14.       If ( max(ret_ij^new) = expInvRet_ij^new )
15.           Return accept
16.       Else
17.           Return reject
18.   }
19. }
20. schedule ( ) {
21.       Get the [ret_max^new, SDmax ] in maxRet(PotentialScheduleList)
22.       If ( SD_max is initiateNewVM)
23.           initiateNewVM in rp_j
24.       Schedule the u_new in VM_max in rp_j according to SD_max.
25.   }
```

Fig. 5 Pseudo-code for knowledge based profit maximization algorithm

Decision Tree Algorithm

The Microsoft Decision Tree algorithm is a classification and regression algorithm provided by Microsoft SQL Server Analysis Services to be used in the predictive modeling of both discrete and continuous attributes. For discrete attributes, the algorithm makes predictions based on the relationships between input columns in a dataset. It uses the values, known as states, of those columns to predict the states of a column that is designated as predictable. Specifically, the algorithm identifies the input columns that are correlated with the predictable column. For example, in a scenario to predict which sequence of strategies should be run for the new request, if eight out of ten relax requests run with sequence 1 and two out of ten tight requests run with sequence 3, so the algorithm infers that deadline of requests is a good predictor of selecting sequence of strategies.

Working of Decision Tree

The Microsoft Decision Trees algorithm builds a data mining model by creating a series of splits in the tree. These splits are represented as nodes. The algorithm adds a node to the model every time that an input column is found to be significantly correlated with the predictable column. The way that the algorithm determines a split is different depending on whether it is predicting a continuous column or a discrete column. The predictable column in our system is the sequence's number of strategies and input columns are request's characteristics such as Deadline, Budget, FileSize, and Length. Fig. 6 shows the used decision tree, it is based on the knowledge database.

Experimental Results

In this paper, CloudSim [13] is used as a Cloud environment simulator and implements our algorithms within this environment. We observe the performance of the proposed algorithm from both users' and SaaS providers' perspectives. From users' perspective, we observe how many requests are accepted and how fast user requests are processed (we call it average response time). From SaaS providers' perspective, we observe how much profit they gain and how many VMs they initiate. Therefore, we use four performance measurement metrics: total profit, average request response time, number of initiated VMs, and number of accepted users. All the parameters from both users' and IaaS providers' side used in the simulation study are given as [14]. Table 2 shows the parameter's value which we used from user's side and the detail resource's characteristics which are used for modelling IaaS providers are shown in Table 3. We examine our algorithm with the total of 300 users. The experimental environment is showed in Fig. 7. As shown in it, our algorithms is written in java language and so the knowledge process part to use decision tree, run a "DMX query" for connecting to Analysis Service and scheduling part for saving the decision of admission control in SQL database used a "SQL Query".

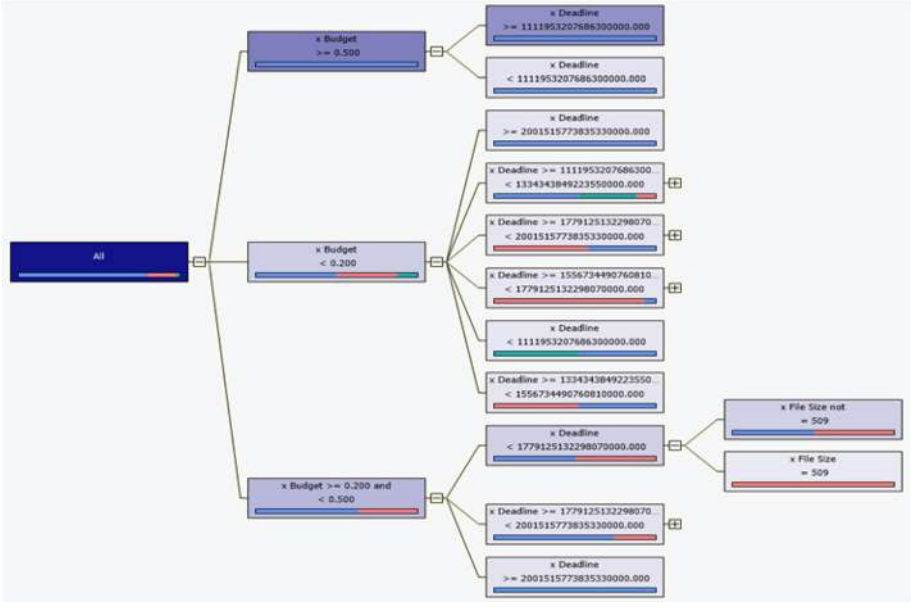

Fig. 6 The decision tree model for predictiong the sequence number of strategies

Table 2 All the parameters from users' side used in the simulation study

Number of users	Deadline	Budget	Length	Penalty rate factor
Vary from 50-300	vary the deadline from "very tight" to "very relax" ($\alpha = 0.5$ to $\alpha = 2.5$)	Random from "0.1$" to "1$"	Vary from "10^5 MI" to "$5*10^5$ MI"	Vary the mean of *r* from "very small" (4) to "very large" (44).

Performance results

In this section, we compare our proposed algorithm with reference algorithm by varying the number of users. Then, the impact of QoS parameters on the performance metrics is evaluated. Then, the impact of QoS parameters on the performance metrics is evaluated. All of the results present the average obtained by 6 experiment runs. In each experiment we vary one parameter, and others are given constant mean value. The constant mean, which are used during experiment, are as follows: arrival rate = 300 requests/sec, deadline = 2*estprocT, budget = 1 $, request length = 3×10^5 MI, and penalty rate factor (r) =10.

Comparison with Reference Algorithm

To observe the overall performance of our algorithm, we vary the number of users from 50 to 300 without varying other factors such as deadline and budget. Fig. 8 presents the comparison of our proposed algorithm with reference algorithm, ProfPD [14], in terms of the four performance metrics.

When the number of user requests varies from 50 to 300, for each algorithm the total profit and average response time has increased, because of more user requests. Fig. 8a shows that our algorithm achieves (30 %) more profit over ProfPD when number of users changes from 50 to 300. This is because when the number of requests increased, the number of users being accepted increased by utilizing initiated VM.

Fig. 8b shows that our algorithms' trends of response time increase from 50 users to 300 users because of increasing in processing of user requests per VM. When there is smaller number of requests, the difference between different algorithm's response times becomes significant. For example, with 50 requests, our algorithm gives users 23 % lower response time ProfPD, and even accept more requests. This is because in our algorithm, a request can run on an idle Initiated Vm on a resource provider instead of waiting on a VM of a resource provider. So the response time will be decrease.

Figure 8c shows that our algorithm initiates 22.22 % less number of VMs and Fig. 8d shows that our algorithm accepts 12.23 % more user requests. That is because in our algorithm a request can run on a Vm in a resource provider instead of waiting on a VM in other resource provider and complete. So other new requests are accepted.

Table 3 The summary of resource provider characteristics

Provider	VM types	VM price ($/hour)
Amazon EC2	Small/Large	0.12/0.48
RackSpace	Windows	0.32

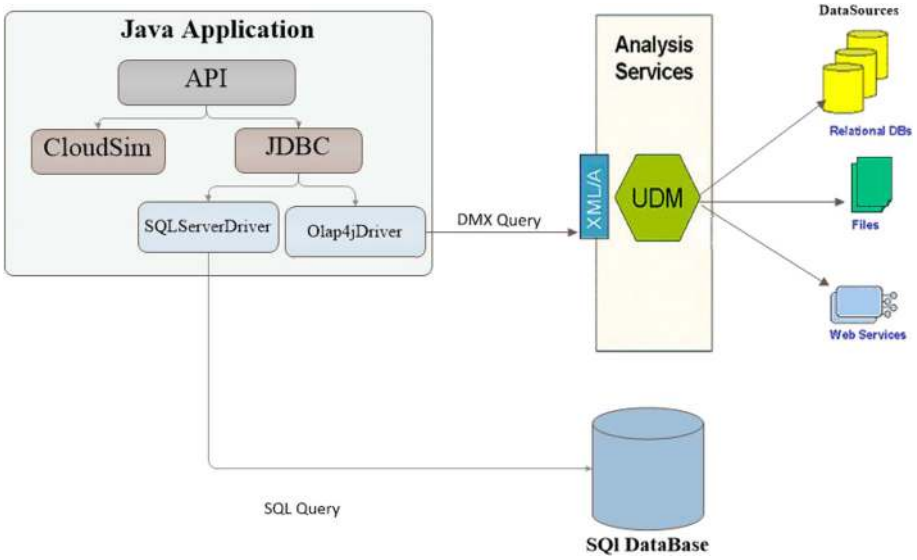

Fig. 7 The view of experimental environment

Impact of QoS parameters

In the following sections, we examine various experiments by varying both user and resource provider side's SLA properties to analyze the impact of each parameter.

1 Impact of variation in deadline

To investigate the impact of deadline in our algorithms, we vary the deadline, while keeping all other factors such as budget fixed. Fig. 9a shows that our algorithm

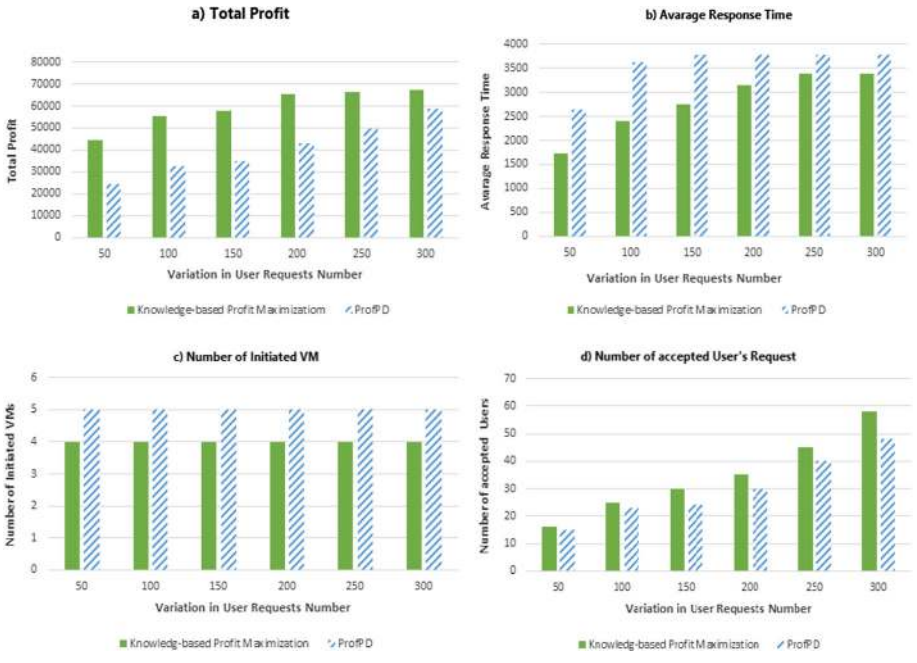

Fig. 8 Algorithm's performance during variation in number of user requests

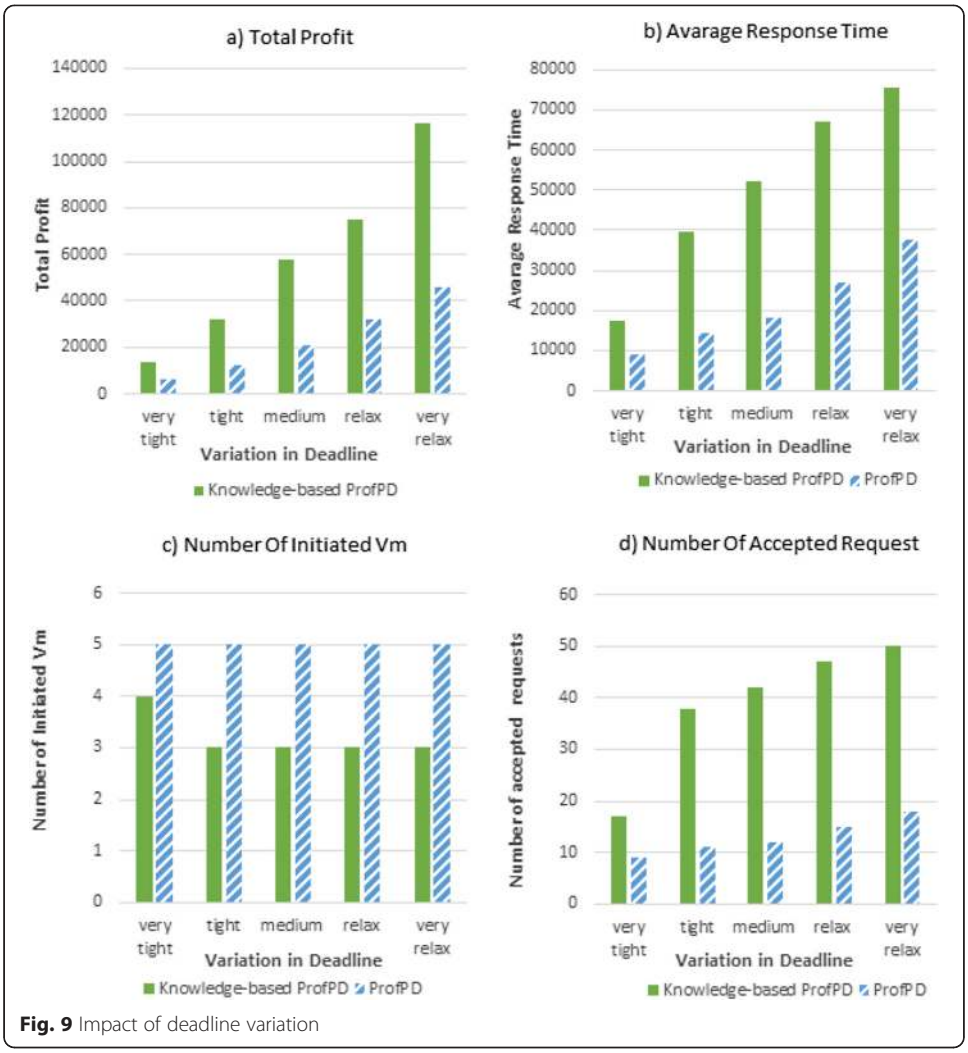

Fig. 9 Impact of deadline variation

achieved the highest profit (59.43 % over ProfPD) by accepting 64 % more user requests (Fig. 9d) and initiating 22.22 % less VMs (Fig. 9c)". Fig. 9b shows that when deadline is relaxed, our algorithm results in 22.07 % higher average response time than in the case of ProfPD.

Our Algorithm has larger response time because of the two factors governing response time, i.e., request's service time and VM initiation time. It can be seen from Fig. 9d that our algorithm always requires less VMs, to process more requests. Thus, when service time is comparable to the VM initiation time, the response time will be lower. When the VM initiation time is larger than the service time, the response time is affected by the number of initiated VMs.

2 Impact of variation in Budget

Figure 10 shows variation of budget impacts on our algorithm, while keeping all other factors such as deadline fixed. This figure shows that when budget varies from "very small" to "large", in average all the factors except initiated VMs, by two algorithms has slightly increased, and number of initiated VMs is almost constant.Our

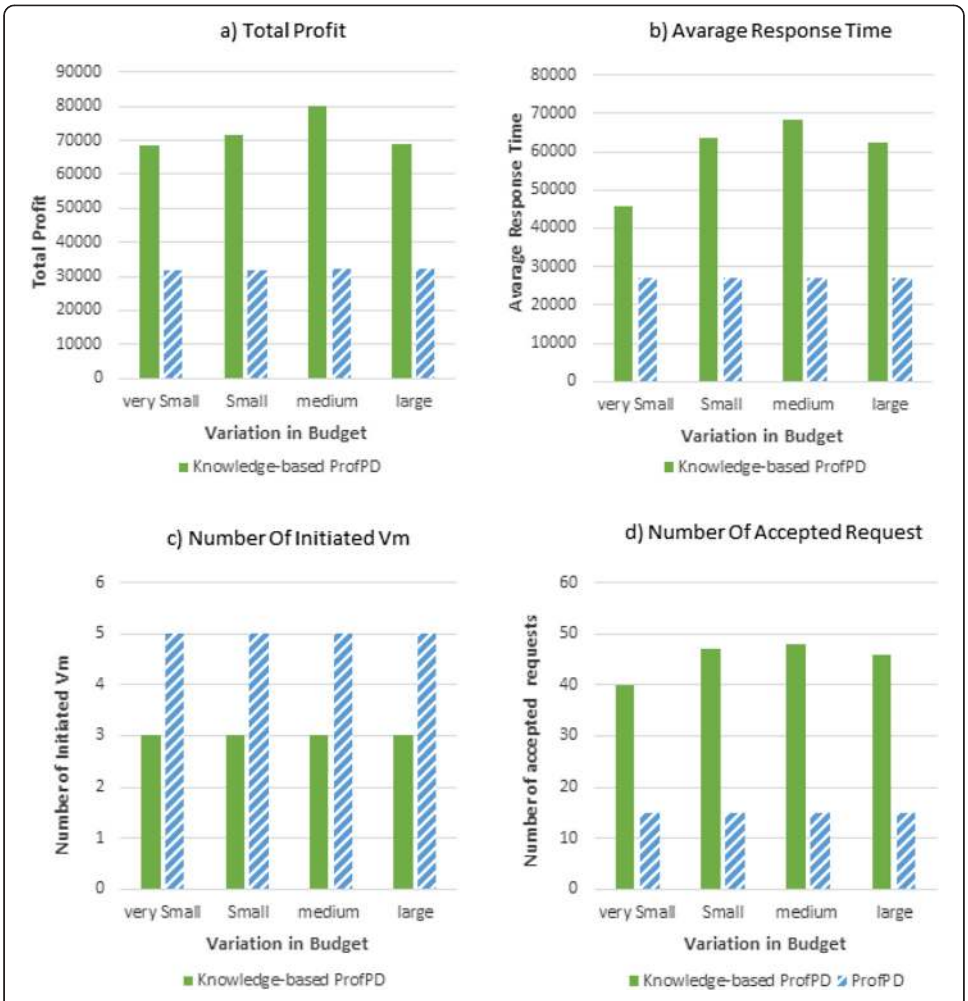

Fig. 10 Impact of budget variation

algorithm compared with ProfPD achieved the highest profit (59.43 % over ProfPD) by accepting 66 % more user requests (Fig. 10d) and initiating 22.22 % less VMs (Fig. 10c)". Response time is too much higher than ProfPD because of more accepted requests. Fig. 10b shows that in all budget variations, our algorithm results in 53.97 % higher average response time than in the case of ProfPD. That is because of request's service time and VM initiation time. It can be seen from Fig. 10d that our algorithm always requires less VMs, to process more requests.

3 Impact of variation in service time

Figure 11 shows how service time impacts our algorithm, while keeping all other factors such as deadline as the same. In order to vary the service time, five classes of request length (MI) are chosen from "very small" (10^5MI) to "very large" (5×10^5MI).

Figure 11a shows that the total profit by two algorithms has slightly increased but response time increased rapidly when the request length varies from "very small" to "very large". Our algorithm achieves the highest profit among ProfPD. For example,

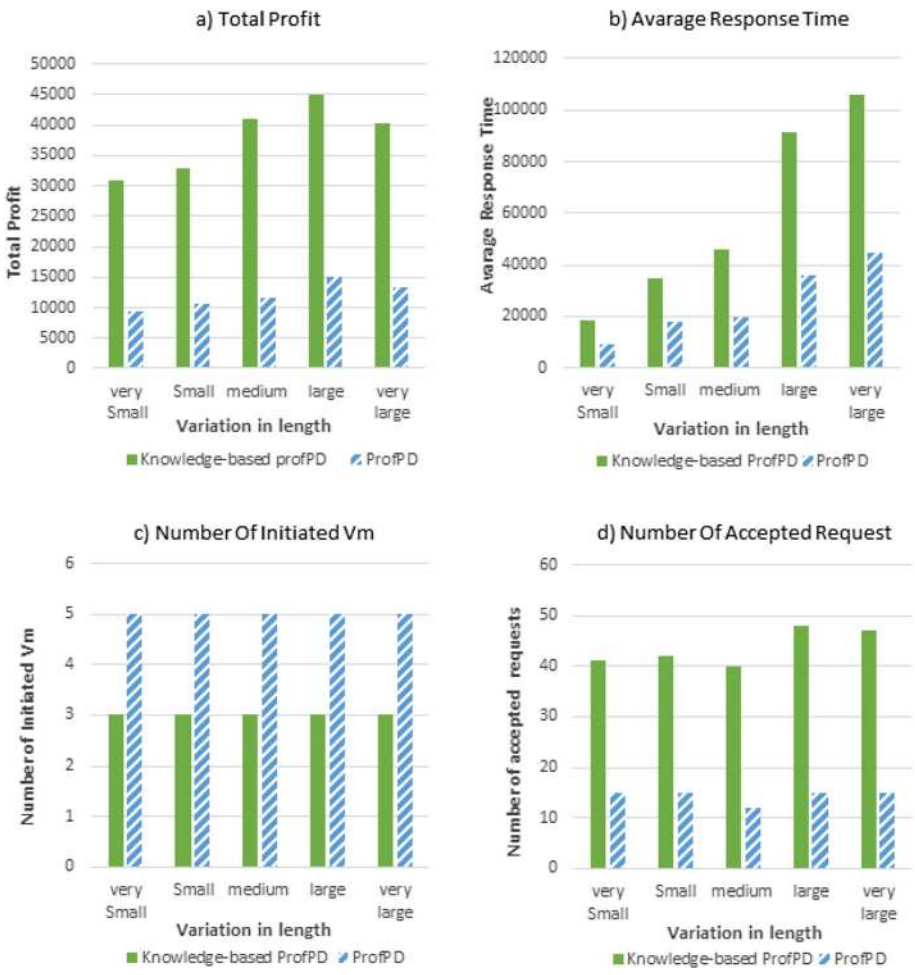

Fig. 11 Impact of request length variation

in the case of "large" request length scenario, our algorithm generated about 44.26 % more profit than ProfPD by accepting 31.25 % more requests (Fig. 11d) and initiating 40 % (Fig. 11c) less VMs. Therefore, our algorithm is the best solution for any size of requests .Moreover, it can be observed from Fig. 11b that our algorithm provides higher response time (39 %) than ProfPD, because it accepts more user requests with less VMs, leading to more requests waiting for processing on each VM.

Conclusions and future directions

In this work, we presented a knowledge based adaptable admission control and scheduling algorithms for efficient resource allocation to maximize profit and CSL for SaaS providers. Through simulation, we showed the proposed algorithm well in a different kind of scenarios. Our simulation results show that in average our algorithm with reduced SLA violation and SaaS provider's cost gives the maximum profit among all other techniques that ultimately focus on fastest response time. In the future we will increase the robustness of our algorithms by handling such errors dynamically. In addition, due to this performance degradation error, we will consider SLA negotiation in Cloud computing environments to

improve the robustness. We will also add different type of services and other pricing strategies such as spot pricing to increase the profit of service provider.

Abbreviations
QoS: Quality of Service; SLA: Service level agreement; KBRP: Knowledge-Based Resource Provisioning.

Competing interests
The Authors declare that they have no competing interests.

Authors' contributions
FM carried out all the development, implementation, experimental data collection and analysis, and manuscript drafting phases of the above work. FS was the work supervisor and critically revising the manuscript. HR.A was the work advisor and reviewed the final manuscript. All authors read and approved the final manuscript.

Author details
[1]Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Isfahan, Iran. [2]Department of Computer Science, University of Georgia, Georgia, USA.

Received: 11 December 2014 Accepted: 4 May 2015
Published online: 20 June 2015

References

1. Yeo CS, Buyya R (2005) Service level agreement based allocation of cluster resources: Handling penalty to enhance utility. In: Proceedings of the 7th IEEE International Conference on Cluster Computing (Cluster 2005), Boston, MA, USA
2. Lee YC, Wang C, Zomaya AY, Zhou BB (2010) Profit-driven service request scheduling in clouds, In: Proceedings of the International Symposium on Clusterand Grid Computing (CCGrid 2010), Melbourne, Australia
3. Microsoft Azure, http://azure.microsoft.com/, retrieved on 10 September, 2014.
4. Popovici I, Wiles J (2005) Profitable services in an uncertain world, In: Proceedings of the 18th Conference on Supercomputing (SC 2005), Seattle, WA
5. Saleforce, http://www.salesforce.com/au/, retrieved on 10 September, 2014.
6. Jaideep D and Varma M (2010) "Learning based Opportunistic admission control algorithms for map reduce as a service," In Proceedings of the 3rd India Software Engineering Conference (ISEC 2010), Mysore, India
7. Bichler M, Setzer T, (2007) Admission control for media on demand services. Service oriented computing and application, In: Proceedings of IEEE International Conference on Service Oriented Computing and Applications (SOCA 2007), Newport Beach, California, USA
8. Broberg J, Venugopal S, Buyya R (2008) Market-oriented grids and utility computing: The state-of-the-art and future directions. Journal of Grid Computing 6:255–276
9. Netto M, Buyya R (2009) Offer-based scheduling of deadline-constrained bag-of-tasks applications for utility computing systems, In: Proceedings of the 18th International Heterogeneity in Computing Workshop (HCW 2009), in conjunction with the 23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS 2009), Roma, Italy
10. Varia J, Architecting applications for the Amazon Cloud, In: R. Buyya, J. Broberg, A. Goscinski (Eds.), Cloud Computing: Principles and Paradigms, Wiley Press, New York, USA, ISBN: 978-0470887998, 2010, http://aws.amazon.com
11. Mary NAB, "Profit Maximization for Service Providers using Hybrid Pricing in Cloud Computing," International Journal of Computer Applications Technology and Research, vol. 2, pp. 218-223.
12. ClO, http://www.cio.com.au, retrieved on 10 September. 2010.
13. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R (2011) CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and Experience (ISSN 0038-0644) 1 (41). Wiley Press, New York, USA, pp 23–50
14. Wu L, Kumar Garg S, Buyya R (2012) SLA-based admission control for a Software-as-a-Service provider in Cloud computing environments. J Comput Syst Sci 78:1280–1299
15. Wu L, Garg SK, Buyya R (2011) "SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments," in Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on
16. Liu Z, Squillante MS, and Wolf JL (2001) "On maximizing service-level-agreement profits," In Proceedings of the 3rd ACM conference on Electronic Commerce, pp. 213-223
17. Reig G, Alonso J, Guitart J. Deadline constrained prediction of job resource requirements to manage high-level SLAs for SaaS cloud providers, Tech. Rep. UPC-DAC-RR, Dept. d'Arquitectura de Computadors, University Politécnica de Catalunya, Barcelona, Spain, 2010.
18. Mohana RS, Thangaraj P (2013) "Machine learning approchaches in improving Service Level Agreement-based admission control for software as a service provider in cloud". J Comput Sci 9(10):1283–1294
19. Choi Y, Lim Y (2014) "Design of Cost Function for VM Allocation in Cloud Computing". International Journal of Energy, Information and Communications 5(4):17–26
20. N. Ani Brown Mary, "Profit maximization for saas using sla based spot pricing cloud computing", International Journal of Emerging Technology and Advanced Engineering, International Conference on Information Systems and Computing (ICISC-2013), INDIA, Volume 3, Special Issue 1, January 2013.