International Journal of Electrical and Computer Engineering (IJECE)
Vol.8, No.6, December 2018, pp. 4398~4411
ISSN: 2088-8708, DOI: 10.11591/ijece.v8i6.pp4398-4411

Profit Driven Decision Assist System to Select Efficient IaaS Providers

Mohan Murthy MK[1], Sanjay HA[2], Supreeth BM[3]
[1,2]Department of Information Science and Engineering, Nitte Meenakshi Institute of Technology, India
[3]Misys Financial Software (India) Pvt Ltd, India

Article Info

Article history:

Received Oct 8, 2017
Revised Jan 2, 2018
Accepted Jan 16, 2018

Keyword:

Cloud
Decision assist
IaaS providers
IaaS selection

ABSTRACT

IaaS providers provide infrastructure to the end users with various pricing schemes and models. They provide different types of virtual machines (small, medium, large, etc.). Since each IaaS provider uses their own pricing schemes and models, price varies from one provider to the other for the same requirements. To select a best IaaS provider, the end users need to consider various parameters such as SLA, pricing models/schemes, VM heterogeneity, etc. Since many parameters are involved, selecting an efficient IaaS provider is a challenging job for an end user. To address this issue, in this work we have designed, implemented and tested a decision-assist system which assists the end users to select efficient IaaS provider(s). Our decision-assist system consists of an analytical model to calculate the cost and decision strategies to assist the end user in selecting the efficient IaaS provider(s). The decision assist system considers various relevant parameters such as VM configuration, price, availability, etc. to decide the efficient IaaS provider(s). Rigorous experiments have been conducted by emulating various IaaS providers, and we have observed that our DAS successfully suggests the efficient IaaS provider/ providers by considering the input parameters given by the user.

Corresponding Author:

Mohan Murthy MK,
Department of Information Science and Engineering,
Nitte Meenakshi Institute of Technology,
Gollahalli, Yelahanaka, Bengaluru, Karnataka 560064, India.
Email: maakem@gmail.com

1. INTRODUCTION

Cloud computing is the distributed computing model which provides computing facilities and resources to the users in an on-demand pay-as-you-go model [1]. Cloud computing provides the facility to access shared resources and common infrastructure, offering services on demand over the network to perform operations that meet changing business needs [2]. Users are moving towards cloud because it offers several benefits such as elasticity, maintenance free, cost effectiveness, etc. It provides a higher QoS than a traditional software model with less initial investment. Based on the type of services provided in the cloud paradigm, three important service models are defined: Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Service (PaaS). In IaaS, infrastructure such as computing resources (Virtual Machines), storage space, network, etc. are given as services. VM selection is a complicated task in cloud computing environment because there are many alternative VMs with varying capacities [3]. Since IaaS providers use their own pricing schemes and models, price varies from one provider to the other for the same requirements. We have conducted a detailed survey [4] of different IaaS providers. The next few paragraphs brief our important findings of the survey.

IaaS providers provide infrastructure to the customers with various pricing options. For example, pay-as-you-go model; in this model, theuser will be paying the money for what he has used. A variation of

the pay-as-you-go model is also available in which if the user is interested in the long-term utilization of the resource, then initially a one-time subscription fee is collected from the user with reduced hourly usage charge. This pricing option is called as subscription based pricing. We have observed this model in Amazon EC2. The subscription-based pricing details of Amazon EC2 are given in Table 1.

Table 1. Subscription Fee Details in Amazon

Instance type	Initial Fee		Linux/Unix usage per hour	Windows usage per hour
	1-year term	3-years term		
Small	$227.5	$350	$0.03	$0.05
Large	$910	$1400	$0.12	$0.20
Extra Large	$1820	$2800	$0.24	$0.40

IaaS providers provide different types of virtual machines. For instance, Amazon EC2 provides small, large, extra-large types of VMs (Virtual Machines). The pricing details of these VM types are given in Table 2. Few IaaS providers offer adiscount on the total billed amount. For instance, the discount details of the IaaS Provider Cloud Sigma is given in Table 3. Some IaaS providers give the option to end users to configure the VMs while creating them. In this case, end users can configure the RAM, CPU, and Storage Space of the VM. This type of configurable VM option is observed in CloudSigma. In such cases, pricing will be at themore granular level.

Table 2. Pricing details of Amazon EC2 VMs

Instance type	Linux/Unix usage	Windows usage
Small	$0.085 per hour	$0.12 per hour
Large	$0.32 per hour	$0.48 per hour
Extra Large	$0.64 per hour	$0.96 per hour

Table 3. Discount details in Cloud Sigma

	Duration in months					
	1	3	6	12	24	36
% discount	0	3	10	25	35	45

IaaS providers like RackSpace, Amazon EC2, provide fixed VMs where the capacity of the VM is predefined, and the end user will not have any option to change it. Due to the vast diversity of the available cloud services, from the customer's point of view, it is very difficult to decide whose services they should use and what is the basis for their selection [5]. Selecting efficient IaaS providers is a tedious job for the end users since he/she must consider various parameters like SLA, pricing models/schemes, and different types of VM instances. A decision assist system which assists the user to select efficient IaaS providers makes the end user job easier.

In this work, we have designed, implemented and tested a decision-assist system (DAS). The DAS consists of ananalyticalmodel to compute the cost and decision strategies to assist the end user in selecting the efficient IaaS provider(s). The DAS has the user interface to capture the end user requirements. After capturing the requirements, using the analytical model and decision strategies the system will suggest efficient IaaS provider(s) based on the user requirements. We have considered the following parameters to develop analytical model and decision strategies

a. The requirements such as memory, CPU, storage, etc.
b. Tenure, which plays an important role in selecting the pricing scheme.
c. VM heterogeneity.
d. Different pricing schemes such as pay-as-you-go, subscription, etc.
e. QoS parameters such as server availability and VM initiation time.
f. Location of the data center.

The decisions of the DAS will be accurate only if the IaaS provider information is up-to-date. Any changes in the parameters (which are going to affect the decision of selecting efficient IaaS provider) at the IaaS providers should be reflected in real time at DAS. Otherwise, the DAS will use obsolete data to decide the efficient IaaS provider which results in aninaccurate decision. To address this issue, we have developed the webservice APIs which are exposed by the DAS. These APIs are used to receive the information and helps the DAS to be in synchronization with the information available at the IaaS providers. By agreeing to

provide the information about the VM instance types and the pricing details, an IaaS provider can attract a large number of customers if quality service is provided at the reasonable cost. Using the DAS, end users can set their priorities on different parameters (price, availability, etc.) to search the most suitable IaaS providers. The DAS also provides an option to relax the search criteria on different parameters. For example, if the users are looking for a VM with memory in a certain range (rather than a fixed number), they can use the option of memory variation which is provided by DAS. The options of setting priorities and variations on different parameters make the DAS flexible and user-friendly.

To test our DAS, we have emulated various IaaS providers, and different popular scenarios are tested. In the tested scenarios, we have found that the DAS provided most efficient IaaS provider/providers considering the different input parameters. Rest of the paper is organized as follows. Section 2, briefs about the related work, section 3 gives an overview of the decision assist system, section 4 and section 5 explains the analytical model and decision strategies respectively, section 6 gives the details of the experiments and results, followed by a conclusion.

2. RELATED WORK

The work by S.K. Garg et al [5] presents a framework (SMICloud) to rank cloud service providers based on performance metrics like sustainability, suitability, stability, etc. authors have designed Analytical Hierarchical Process (AHP) based ranking mechanism to compare different cloud services. The work by Michael Smit et al [6] presents a methodology and an implementation of a service-oriented application that provides relevant metadata information describing offered cloud services via a uniform RESTful web services. This work concentrates only in fetching the information using web services. Selecting the best IaaS providers according to the user requirement is not addressed. The work by Dhaval Limbani et al [7] proposes a service broker for the selection of data center based on thelatency of the user requests. The work considers thecost only when more than one datacenters have thelowest latency within a region. In this work, only the problem of selecting an effective datacenter is addressed.

The work by Stella Gatziu Grivas et al [8] proposes a cloud broker which has knowledge of the supported business processes, the existing service offerings from the marketplace, the current relations between the business processes and the cloud services. Cloud broker manages a repository of all providers and services which are relevant to the value chain of a company. In this work, different VM heterogeneity, pricing models, and schemes, VM initiation time are not considered moreover the work is in the proposal stage, implementation is not done. The work by Srijith K. Nair et al [9] describes the concepts of cloud bursting, cloud brokerage and discusses the security issues associated with the two models. The cloud brokerage model does not have the ability to give efficient cloud providers by considering user requirements since it is only servicing based on storage and computing use case scenarios.

The work by May Al-Roomi et al [10] focuses on comparing many employed and proposed pricing models techniques and highlights the pros and cons of each. The comparison is based on many aspects such as fairness, pricing approach, and utilization period. In this work, the comparison of the pricing models is made. They have not considered the pricing schemes, VM heterogeneity and QoS of multiple providers. The work by Hyun Jin Moon et al [11] analyzes the performance of resource scheduling policies. They have considered several models and scheduling policies,which are profit model, SLA model, and SLA-based scheduling. This work concentrates on optimization of cost. The work by Linlin Wu et al [12] has defined mapping strategy by interpreting customer requirements to infrastructure level parameters. It also designs and implements scheduling mechanisms to maximize SaaS provider's profit by reducing the infrastructure cost and minimizing SLA violations. This work concentrates on SaaS providers. Different VM heterogeneity, QoS parameters, pricing models and pricing schemes are not considered.

In work [13], authors address 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. They define the scheduling strategies to satisfy the objectives of service providers and consumers. This work does not address the issue of selecting an efficient IaaS provider when the end user wants to approach the IaaS providers for the service directly.

Plan For Cloud [14] is a free cloud cost calculator which gives cost reports for deployment options. It selects aserver based on RAM and CPU count and lists the resulting server of only a few providers. We have observed that cost report generation option is available only for 3 years duration. The changes at the IaaS providers will not reflect immediately at the PlanForCloud website. The QoS parameters like VM initiation time and availability are not considered.

In work [15], authors present a Cloud service selection framework that uses a recommender system (RS) which helps a user to select the services from different Cloud providers (CP). The RS recommends a service based on the network QoS and Virtual Machine (VM) platform factors of different CPs. The ranking

method proposed by authors only consider services' inside attributesand ignore the relations between context providers and consumers. Junping Dong et al propose services recommendation system [16] based on heterogeneous network analysis in cloud computing. Authors propose service recommendation system based on heterogeneous service network ranking and clustering. In this work, QoS parameters like availability and VM initiation time are not considered. In work [17] authors propose a hierarchical information model for integrating heterogeneous cloud information from different providers and a corresponding cloud information collecting mechanism. Also, authors propose a preference-aware solution evaluation model for evaluating and recommending solutions according to the preferences of application providers. In this work, authors use web page parsing and web APIs invocation to collect the information in real time. These operations are triggered when the user requests cloud service. Parsing webpage of the existing providers in realtime in the ocean of internet is virtuallyimpossible, and it is error-prone. As the authors rightly pointed out in the paper, only a few IaaS providers provide webservice APIs to provide cost and VM information.

The work does not consider QoS parameters, and the option of exposing the webservice APIs in the brokering system is not considered. Also, the work doesn't consider the different pricing schemes and pricing models offered by IaaS providers. M. Whaiduzzaman et al [18] talks about multi-criteria based cloud service selection. The authors have done a survey on different multicriteria methods which can be used to select the cloud services. The work doesn't talk about the VM heterogeneity, different pricing schemes, and models. Also, collecting information from IaaS providers is not addressed in this work. The work [19] is about exploiting performance heterogeneity by selecting a proper VM in an IaaS provider. In this work, multiple IaaS providers and different pricing schemes offered by the same provider are not considered.Inwork [20] authors propose a brokerage based architecture for efficient service selection. In this model, the cloud broker collects and indexes the service provider's properties. The index is used to identify the best-matched service when a request is received from the customer.

3. DECISION ASSIST SYSTEM (DAS)

Figure 1 shows the overview of our decision assist system. End users will interact with DAS using thin clients (browsers). Through the web interface, users can login and provide their requirements. The DAS has the following important components

a. Front controller: All the user requests are received by the front controller. It does the first level of screening. After pre-processing the user requests, it forwards the requests tothe decision maker. The front controller also receives the results from the decision maker and forwards it to the browser.
b. Decision maker: This module has the business logic (analytical model and decision strategies). It receives the pre-processed requests from the front controller. Using the analytical model, it computes the total cost for the user requirements. This computed cost is used bydecision strategies to decide the best IaaS provider. Decision strategies also consider the user input parameters to decide the best IaaS provider.
c. DB: Up-to-date information of the IaaS providers such as provider identity, location, pricing info, and VM details are stored in the DB. Synchronizer and Decision maker modules directly interact with DB.
d. Synchronizer: The synchronizer exposes the webservice APIs. These APIs make sure any update done at the provider site is automatically reflected in the system's database. Published web service APIs are utilized by the IaaS providers to update any change in the relevant information. Also, the DAS has a web interfaceoption, using which an admin can enter the IaaS provider details manually.

Following are web service APIs which we have developed for providers to update any change in the relevant information.

a. sendStaticVMDetails() -This API is used to send information of newly created static VM templates.
b. sendPriceDetails()-This API is used to send price details for existing static VM templates.
c. sendConfigurationVMDetails()-This API is used to send price details for configurable VM templates.
d. deleteVMDetails()-This API is used to send information of the deleted static VM templates.
e. sendDiscountDetails()-This API is used to send the discount details.
f. sendInitialfeeDetails()-This API is used to send the initial fee details.
g. sendAvailabilityDetails()-This API is used to send the availability details.
h. sendInitiationtimeDetails()-This API is used to send the VM initiation time details.

The DAS also provides a simple web interface to manually enter the details of the VM, pricing schemes, etc.

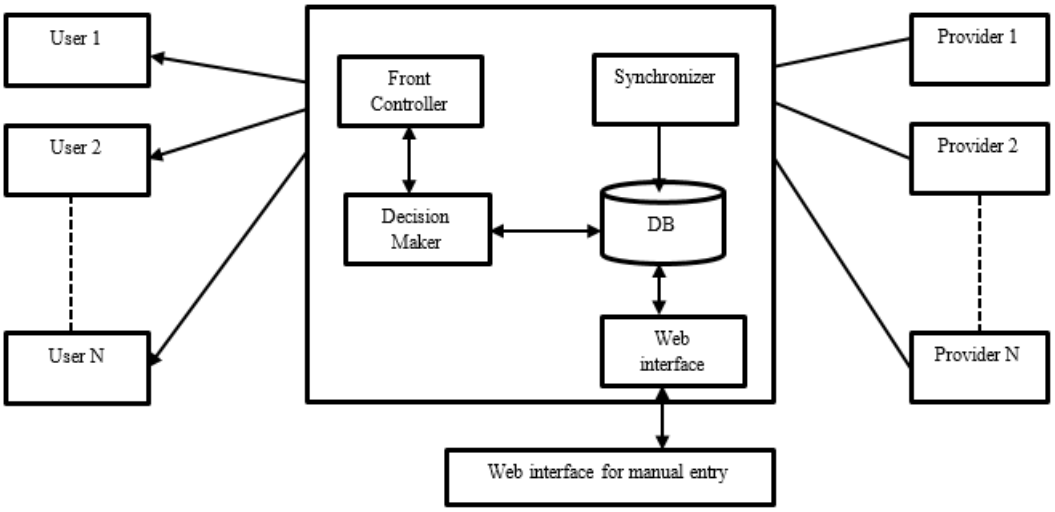

Figure 1. Decision assist system

4. ANALYTICAL MODEL

We have developed an analytical model to calculate the cost by considering user requirements, duration of the service required, VM heterogeneity, pricing schemes, and models. In configurable VMs the price depends on the price per unit of resource and number of units of resource required by the user. For instance, if the price per unit (1 GB) of RAM is $0.02, then for 2 GB of RAM total price would be $0.04. In general, the configurable VMs cost can be calculated by using below formula.

$$P = Cost_{cpu} + Cost_{memory} + Cost_{storage}$$

where; $Cost_{cpu} = CPU_N * CPUPRICE_U$
$Cost_{memory} = MEM_N * MEMPRICE_U$
$Cost_{storage} = STG_N * STGPRICE_U$

where; P=Price of VM
CPU_N=Number of units of CPU
$CPUPRICE_U$=Price per unit for CPU
MEM_N=Number of units of memory
$MEMPRICE_U$=Price per unit for memory
STG_N=Number of units of storage
$STGPRICE_U$=Price per unit for storage

For fixed (static) VMs the price can be calculated as

$$P = VM_{price}$$

where; VM_{price}=Price of VM

In fixed VMs, the price depends on the price of the VM. For e.g., in Amazon EC2, for standard on-demand small instance with windows OS, the price is $0.12 per hour. In some cases, if the user is interested in long term utilization of a resource, then initially a subscription charge will be collected from the user, later the hourly usage charge will be reduced. In few cases, we have observed that cloud providers offer adiscount on the total billed amount. The final price is given by the below analytical model.

$$FP = I + (P * T) - D$$

where; FP=Final price
I=Initial fee
P=Price of VM.
T=Duration (Tenure)

D=Discount offered

The analytical model considers the different pricing schemes and pricing models. It also covers the cases where initial fee and discount comes into the picture. VM heterogeneity is implicitly considered when calculating the price of the VM. The value of *P* is calculated differently based on the type of VM (fixed or configurable).

5. DECISION STRATEGIES

We have developed algorithms to select efficient IaaS providers based on the user requirements and SLA parameters. In the first step, requirements of the user are collected from the web interface. The requirements include RAM, CPU, storage space, OS, duration, location, and priorities for the cost and QoS parameters (availability & initiation time). The user can set the priorities as per his needs. For example, if the user is interested only in getting the lowest cost provider, without giving much importance to QoS parameters then he/she can set the cost priority to highest. If theuser is interested in QoS, then he can set the priorities accordingly. The main algorithm will take user inputs to decide the best IaaS providers.

Algorithm 1: Main Algorithm

1. **Inputs**: RAM, CPU, Storage, OS, Duration, Location, Cost Variation, Pc, Pa, Pi
2. Set VM_{main}=[ ], $VM_{sublist}$=[ ], $VM_{potential}$=[ ], VM_{avail}=0, VM_{init}=0, VM_{cost}=0, $IaaS_{providers}$=[ ]
3. VM_{main}=read from database as per the inputs RAM, CPU, Storage, OS, Duration, Location
4. **if** VM_{main}=∅
5. Indicate this to the user and exit.
6. **endif**
7. **for** i=1 to number of VMs present in VM_{main}
8. $VM_{main}[i].cost=I_i + (P_i*T_i)-D_i$
9. **endfor**
10. **if** $[(P_c> P_i)$ && $(P_c> P_a)]$
11. create $VM_{sublist}$ | $VM_{sublist} \subseteq VM_{main}$ and $\forall$ VM in $VM_{sublist}$, $VM_{cost}=MIN(VM_{cost})$
12. **if** | $VM_{sublist}$ |=1
13. $VM_{potential}[1]=VM_{sublist}[1]$
14. **else**
15. $VM_{potential}$=tieBreaker1(P_a, P_i,$VM_{sublist}$)
16. **endif**
17. **elseif** $[(P_i> P_c)$ && $(P_i> P_a)]$
18. VM_{main}=readFromDB($VM_{main}[i]$, 2)
19. create $VM_{sublist}$ | $VM_{sublist} \subseteq VM_{main}$ and $\forall$ VM in $VM_{sublist}$, $VM_{init}=MIN(VM_{init})$
20. **if** | $VM_{sublist}$ |=1
21. $VM_{potential}[1]=VM_{sublist}[1]$
22. **else**
23. $VM_{potential}$=tieBreaker2(P_c, P_a, $VM_{sublist}$)
24. **endif**
25. **elseif** $[(P_a > P_c)$ && $(P_a> P_i)]$
26. VM_{main}=readFromDB($VM_{main}[i]$, 1)
27. create $VM_{sublist}$ | VMsublist $\subseteq VM_{main}$ and $\forall$ VM in $VM_{sublist}$, $VM_{avail}=MAX(VM_{avail})$
28. **if** | $VM_{sublist}$ |=1
29. $VM_{potential}[1]=VM_{sublist}[1]$
30. **else**
31. $VM_{potential}$=tieBreaker3(P_c, P_i,$VM_{sublist}$)
32. **end if**
33. **else if** $[(P_c=P_a$&& $P_c> P_i)$ || $(P_c=P_i$ && $P_c > P_a)$ || $(P_c=P_i=P_a)]$
34. create $VM_{sublist}$ | $VM_{sublist} \subseteq VM_{main}$ and $\forall$ VM in $VM_{sublist}$, VM_{cost} MIN(VM_{cost}) $\pm$ cost variation
35. **if** | $VM_{sublist}$ |=1
36. $VM_{potential}[1]=VM_{sublist}[1]$
37. **else**
38. $VM_{potential}$=tieBreaker1(P_c, P_a, P_i,$VM_{sublist}$)
39. **endif**
40. **endif**
41. $IaaS_{providers}$ [ ]=getIaaSProviderDetailsFromDB($VM_{potential}$)

The purpose of the main algorithm is to provide the best IaaS provider/s based on the user inputs. First, it computes the total cost using the analytical model (step 8). Then, based on the priorities set by the user for cost, availability and VM initiation time it decides the best IaaS providers. The main algorithm is assisted by sub-algorithms to decide the best IaaS provider/s. Working principle of the sub-algorithms is same. Sub-algorithm 1 (tieBreaker1) is called from the main algorithm when the cost has the highest priority and if we have more than one VM with the same cost.

There is a possibility that user may be not sure about the priorities. He may set all the priorities same, or the priority of the cost is equal to the priority of any one of the other parameter (availability or initiation time) in these cases we have given an option of setting the Cost Variation. In such cases, the VMs with the minimum cost ± cost variation, are considered to decide the potential VMs (Steps 33 and 34).

Sub algorithm 1
Function $VM_{potential}$=tieBreaker1(Pa, Pi,$VM_{sublist}$)
1. **Inputs**: P_a, P_i, $VM_{sublist}$
2. **if** [(P_a> P_i) || (P_a=P_i)]
3. $VM_{sublist}$=readFromDB($VM_{sublist}$[i], 1)
4. create $VM_{secondlist}$ | $VM_{secondlist} \subseteq VM_{sublist}$ and $\forall$ VM in $VM_{secondlist}$, VM_{avail}=MAX(VM_{avail})
5. **if** | $VM_{secondlist}$ |=1
6. $VM_{potential}$[1]=$VM_{secondlist}$[1]
7. **else**
8. $VM_{sublist}$=readFromDB($VM_{sublist}$[i], 2)
9. create $VM_{thirdlist}$ | VMthirdlist $\subseteq$ $VM_{secondlist}$ and $\forall$ VM in $VM_{thirdlist}$, VM_{init}=MIN(VM_{init})
10. $VM_{potential}$=getPotentialVMs($VM_{thirdlist}$)
11. **end if**
12. **else if**(P_a< P_i)
13. $VM_{sublist}$=readFromDB($VM_{sublist}$[i], 2)
14. create $VM_{secondlist}$ | $VM_{secondlist} \subseteq VM_{sublist}$ and $\forall$ VM in $VM_{secondlist}$, VM_{init}=MIN(VM_{init})
15. **if** | $VM_{secondlist}$ |=1
16. $VM_{potential}$[1]=$VM_{secondlist}$[1]
17. **else**
18. $VM_{sublist}$=readFromDB(VMsublist[i], 1)
19. create $VM_{thirdlist}$ | $VM_{thirdlist} \subseteq VM_{secondlist}$ and $\forall$ VM in $VM_{thirdlist}$, VM_{avail}=MAX(VM_{avail})
20. $VM_{potential}$=getPotentialVMs($VM_{thirdlist}$)
21. **end if**
22. **end if**

Sub-algorithm 2 (tieBreaker2) is called from the main algorithm when the VM initiation time has the highest priority and if we have more than one VM with the same VM initiation time. Sub-algorithm 3 (tieBreaker3) is called from the main algorithm when the availability has the highest priority and if we have more than one VM with the same availability. The main algorithm along with the sub-algorithms 1, 2, and 3 are the core part of the decision strategies. Apart from these algorithms we have written utility functions which serve as helper functions to finalize the best IaaS providers.

Sub algorithm 2
Function $VM_{potential}$=tieBreaker2(P_c, P_a, $VM_{sublist}$)
1. **Inputs**: P_c, P_a, P_i, $VM_{sublist}$
2. **if** [(P_c> P_a) || (P_c=P_a)]
3. create $VM_{secondlist}$ | $VM_{secondlist} \subseteq VM_{sublist}$ and $\forall$ VM in $VM_{secondlist}$, VM_{cost}=MIN(VM_{cost})
4. **if** | $VM_{secondlist}$ |=1
5. $VM_{potential}$[1]=$VM_{secondlist}$[1]
6. **else**
7. $VM_{sublist}$=readFromDB($VM_{sublist}$[i], 1)
8. create $VM_{thirdlist}$ | $VM_{thirdlist} \subseteq$ $VM_{secondlist}$ and $\forall$VM in $VM_{thirdlist}$, VM_{avail}=MAX(VM_{avail})
9. $VM_{potential}$=getPotentialVMs($VM_{thirdlist}$)
10. **endif**
11. **elseif**(P_c< P_a)
12. $VM_{sublist}$=readFromDB($VM_{sublist}$[i], 1)

13. create $VM_{secondlist}$ | $VM_{secondlist} \subseteq VM_{sublist}$ and $\forall$ VM in $VM_{secondlist}$, $VM_{avail}=MAX(VM_{avail})$
14. **if** | $VM_{secondlist}$ |=1
15. $VM_{potential}[1]=VM_{secondlist}[1]$
16. **else**
17. create $VM_{thirdlist}$ | $VM_{thirdlist} \subseteq VM_{secondlist}$ and $\forall$ VM in $VM_{thirdlist}$, $VM_{cost}=MIN(VM_{cost})$
18. $VM_{potential}$=getPotentialVMs($VM_{thirdlist}$)
19. **end if**
20. **end if**

Sub algorithm 3
Function $VM_{potential}$=tieBreaker3(P_c, P_i,$VM_{sublist}$)
1. **Inputs**: P_c, P_a, P_i, $VM_{sublist}$
2. **if** [($P_c > P_i$) || ($P_c=P_i$)]
3. create $VM_{secondlist}$ | $VM_{secondlist} \subseteq VM_{sublist}$ and $\forall$ VM in $VM_{secondlist}$, $VM_{cost}=MIN(VM_{cost})$
4. **if** | $VM_{secondlist}$ |=1
5. $VM_{potential}[1]=VM_{secondlist}[1]$
6. **else**
7. $VM_{sublist}$=readFromDB($VM_{sublist}$[i], 2)
8. create $VM_{thirdlist}$ | $VM_{thirdlist} \subseteq VM_{secondlist}$ and $\forall$ VM in $VM_{thirdlist}$, $VM_{init}=MIN(VM_{init})$
9. $VM_{potential}$=getPotentialVMs($VM_{thirdlist}$)
10. **end if**
11. **else if**($P_c < P_i$)
12. $VM_{sublist}$=readFromDB($VM_{sublist}$[i], 2)
13. create $VM_{secondlist}$ | $VM_{secondlist} \subseteq VM_{sublist}$ and $\forall$ VM in $VM_{secondlist}$, $VM_{init}=MIN(VM_{init})$
14. **if** | $VM_{secondlist}$ |=1
15. $VM_{potential}[1]=VM_{secondlist}[1]$
16. **else**
17. create $VM_{thirdlist}$ | $VM_{thirdlist} \subseteq VM_{secondlist}$ and $\forall$ VM in $VM_{thirdlist}$, $VM_{cost}=MIN(VM_{cost})$
18. $VM_{potential}$=getPotentialVMs($VM_{thirdlist}$)
19. **end if**
20. **end if**

The utility function 1 (getPotentialVMs) stores VM details from one list to another. The utility function 2 (readFromDB) reads the availability or initiation time of a VM based on the value of the input 'key'. The utility function 3 (getIaaSProviderDetailsFromDB) gets the IaaS provider details from the database based on the VM key.

Utility function 1
Function $VM_{potential}$=getPotentialVMs($VM_{thirdlist}$)
1. Set $VM_{potential}$=[ ]
2. **for** i=1 to number of VMs present in $VM_{thirdlist}$
3. $VM_{potential}[i]=VM_{thirdlist}[i]$
4. **end for**

Utility function 2
Function VM_{list}=readFromDB(VM_{list}, key)
1. **if** key=1
2. **for** i=1 to number of VMs present in VM_{list}
3. $VM_{list}[i]$.availability=readAvailabilityFromDB($VM_{sublist}[i]$.id)
4. **end for**
5. **else if** key=2
6. **for** i=1 to number of VMs present in VM_{list}
7. $VM_{list}[i]$.inittime=readInitTimeFromDB($VM_{sublist}[i]$.id)
8. **end for**
9. **end if**

Utility function 3
Function IaaSproviders [ ]=getIaaSProviderDetailsFromD ($VM_{potential}$)

1. Set IaaSproviders [ ]=[ ]
2.**for** i=1 to number of VMs present in VMpotential
3. IaaSproviders[i]=readIaaSDetailsFromDB($VM_{potential}$[i].key)
4. **end for**

where; VM_{main}=First list of the VMs which matches the first level of search criteria (RAM, CPU, Storage, OS, Duration, Location)
$VM_{sublist}$=Optimal list of the VMs after applying another level of search criteria (priorities of cost, availability and VM initiation time).
$VM_{potential}$=Possible list of VMs which match user criteria.
$VM_{secondlist,}$ $VM_{thirdlist}$=List of the VMs after applying the tiebreakers based on the priorities set by the users on cost, availability and VM initiation time.
$IaaS_{providers}$=Final list of VMs which is shown to the user.
P_c=Priority of cost set by theuser.
P_a=Priority of availability set by theuser.
P_i=Priority of VM initiation time set by theuser.
VM_{avail}=Availability of the VM.
VM_{init}=Initiation time of the VM.
VM_{cost}=Cost of the VM.

6. RESULTS AND DISCUSSION

The decision assist system is implemented using Java and Java related technologies. The system is hosted on Tomcat server. We have used JDK 1.7.0_25 and Apache Tomcat 7.0.42. MySQL 5.5 is used as the database to store the IaaS provider and VM information. Apache Ant 1.9.2 has been used to develop build scripts. The user interface is designed using JSP. Front controller design pattern has been used between the end users and the DAS. The decision strategies are implemented using core java.

We have used publisher, subscriber design pattern between the IaaS providers and our DAS. Our system acts as publisher by publishing the APIs, and the IaaS providers are the subscribers. To develop webserviceAPIs, we have used Axis 2 framework. Desktop machines with intel core i3 processor, 4 GB RAM and 500 GB hard disk are used to emulate IaaS providers. All the machines are connected through LAN. We have emulated IaaS providers to test different scenarios. Initially, we have tested our Webservice APIs for their functionality. Following are some of the operations which are executed at the emulated IaaS providers.

a. Inserting a new VM.
b. Inserting a new pricing scheme for a VM.
c. Updating the discount details.
d. Updating the pricing details of a VM.
e. Updating the VM configuration details.
f. Deleting a VM.
g. Deleting a pricing scheme.

We have observed that as soon as the operations are completed at the emulated IaaS providers, the changes made are reflected in the DASs database.

The user can access our decision assist system via a web browser. The pages which are used to collect the user inputs are designed using JSP technology. Users can navigate from one page to another via hyperlinks. User can enter the values for the following parameters

a. Location: Location of the datacenter.
b. Duration: Start and End date during which user needs the VM.
c. Operating System: OS of the VM.
d. Memory (GB): RAM size in GB.
e. Memory variation (%): Sometimes fixed template VMs does not exactly match with the user requirement. Users can specify how much variation they can tolerate.
f. CPU: CPU speed in GHz.
g. CPU variation (%): The variation in CPU capacity users can tolerate.
h. Storage (GB): Storage space of VM in GB.
i. Storage variation (%): The variation in storage space users can tolerate.
j. Priorities: Priorities of the cost, availability and VM initiation time. 10 is the highest priority, and 1 is the lowest.

k. Cost Variation (%): User should set this percentage only when theuser customizes three priorities equal or when theuser gives priority of cost and one SLA parameter equal, and priority of another SLA parameter is less than the other two.

Users will submit their requirements by clicking on the submit button. Pressing the submit button triggers our decision strategies. Based on the input given by the user the corresponding decision strategy will be executed, and the results are published to the user. In the result page, the provider name is hyperlinked to the actual provider's VM selection and payment page, where theuser can select the VM and make the payment.

Based on our survey of different IaaS providers [4] we had mocked the data to test our DAS. Part of the data which is relevant to the documented test scenarios in the next subsection is tabulated in the Tables 4, 5, 6, 7. Each VM instance type is assigned a unique identifier (which is not shown here) in the database which helps in identifying the corresponding provider, pricing schemes/models, discount details, etc. Table 4 shows the data for the fixed VM template. We can observe different categories of VM (Small, Medium, etc.) with fixed configurations. All the prices are in USD.Table 5 shows the data for configurable VMs. Table 6 shows the discount details offered by the providers on the total billed amount. IaaS providers provide ahigher discount when the duration of tenure is long.Some IaaS providers offer the VMs at a discounted rate for the users who are interested in long term business if users are ready to pay some initial fee. Table 7 shows the initial fee details. Rigorous testing has been conducted by running several scenarios with different requirements and found that in each scenario our DAS suggests best IaaS provider/s based on the user input. Few tested scenarios and their results are tabulated in Table 8 and Table 9 respectively.

Table 4. Fixed VM Template Details

VM Type	CPU (GHz)	RAM	Storage	VM initiation time (minutes)	Availability	Location	OS	Price per month	Provider
Small	1.2	2	25	15	99	Chicago	Linux	500	Provider 1
Small	1.5	2	20	30	99.5	Bangalore	Windows	550	Provider 3
Medium	2.4	4	50	30	99	Chicago	Linux	750	Provider 2
Large	3.6	6	75	30	99.5	New York	windows	1200	Provider 4
Extra Large	4.8	8	100	60	99	New York	windows	1400	Provider 10

Table 5. Configurable VM Details

IaaS provider	RAM price /GB/month	CPU price /GHz/month	Storage price/GB/ Month	VM Initiation Time (minutes)	Availability in %	Location	OS
Provider 1	50	75	10	30	99	Sydney	Windows
Provider 2	40	70	12	45	99.5	London	Linux
Provider 3	40	85	15	45	99.9	Bangalore	Windows
Provider 4	75	90	10	15	99.9	Bangalore	Linux
Provider 5	50	65	20	15	99	Bangalore	Windows

Table 6. Discount Details on the Total Billed Amount

IaaS provider	VM #	VM Type	Discount on the total bill in %		
			For 3-6 months	For 7-12 months	For 1 year +
Provider 1	1	Small	10	20	30
Provider 1	Configurable		10	15	20
Provider 2	1	Medium	15	20	25
Provider 3	2	Small	5	10	15
Provider 4	1	Large	0	20	30
Provider 5	3	Small	0	10	20

Table 7. Details of the Reduced Price after Initial Fee

IaaS Provider	VM #	VM Type	Initial fee			Price after initial fee			Regular price/month
			For 3 -6 months	For 7-12 months	For 1 year and more	For 3-6 months	For 7-12 months	For 1 year +	
Provider 1	2	Small	100	150	250	425	400	375	450
Provider 6	1	Medium	200	300	400	825	815	800	850
Provider 7	1	Small	0	100	200	475	450	400	475
Provider 8	2	Medium	0	0	250	850	850	750	850

Table 7. Details of the Reduced Price after Initial Fee

IaaS Provider	VM #	VM Type	Initial fee			Price after initial fee			Regular price/month
			For 3 -6 months	For 7-12 months	For 1 year and more	For 3-6 months	For 7-12 months	For 1 year +	
Provider 9	1	Large	300	600	1000	1400	1250	1000	1500
Provider 10	2	Small	0	250	250	500	450	450	500

Table 8. Few Test Scenarios

#	Duration (Months)	OS	RAM (GB)	CPU (GHz)	Storage (GB)	Priorities			Variation in %			
						Cost	VM Initiation time	Availability	RAM (GB)	CPU (GHz)	Storage (GB)	Cost
1	1	W	2	1	5	9	7	3	-	-	-	-
2	1	L	2	1	5	9	7	3	-	-	-	-
3	1	-	2	1	5	9	7	3	-	-	-	-
4	1	-	2	1	5	5	9	3	-	-	-	-
5	1	-	2	1	5	5	9	8	-	-	-	-
6	1	-	2	1	5	5	6	9	-	-	-	-
7	1	-	2	1	5	5	5	5	-	-	-	10
8	3	-	4	2	40	9	3	5	20	20	50	-
9	3	-	4	2	40	9	3	5	-	-	-	-
10	3	-	4	2	40	6	9	5	-	-	-	-

W=Windows L=Linux

In scenario 1 cost has the highest priority and the operating system is windows, row #1 of table 8 gives the complete details about the search inputs. Row #1 of Table 9 shows the result provided by DAS. From the Table 5, we can observe that providers 1, 3 and 5 are having Windows VMs. Among these 3 providers, the cost is less in provider 1. In fixed template VM, we do not have the match for given configuration hence the Provider 1 is the best match as per the user requirements. In scenario 2 we changed the operating system to Linux, all other search criteria are same as scenario 1.

The result obtained is shown in row #2 of Table 9. From the Table 5, we can observe that providers 2 and 4 are having Linux VMs. Among these 2 providers, the cost is less in provider 2. Hence the Provider 2 is the best match as per the user requirements. In scenario 3 we do not have operating system preference, all other search criteria are same as scenario 1. The result obtained is shown in row #3 of Table 9. From the results of scenario 1 and scenario 2, we can conclude that for scenario 3 either provider 1 or provider 2 is the potential VM. The cost of the Provider 2 is less compare to Provider 1. Hence the Provider 2 is the best match as per the user requirements.

In scenario 4 we changed the priorities settings. The VM initiation time is having highest priority followed by cost and availability. All other search criteria are same as scenario 3. The result obtained is shown in row #4 of Table 9. From the Table 5, we can observe that providers 4 and 5 are having lowest VM initiation time which is 15 minutes. The costhas next highest priority, so among providers 4 and 5, the provider with the lowest cost is the potential VM. In this case, it is provider 5. In scenario 5, again we changed the priorities settings. This time availability is having higher priority than cost. All other settings are same as scenario 4.

The result obtained is shown in row #5 of Table 9. Since all other settings are same as scenario 4, initially we get provider 4 and 5 as the potential providers. While breaking the tie, our algorithm first considers availability since it has the highest priority. The availability of Provider 4 is more comparing to Provider 5 hence Provider 4 is the best match in case of the scenario 5. In scenario 6, once again we changed the priorities settings. This time availability is having higher priority, followed by VM initiation time and cost. All other settings are same as scenario 5. The result obtained is shown in row #6 of Table 9. From Table 5 we can observe that provider 3 and provider 4 are having highest availability which is 99.9. Since the VM initiation time is having higher priority than thecost, it is used for breaking the tie in the beginning. Among provider 3 and provider 4, provider 4 is having lowest VM initiation time (15 minutes). Hence provider 4is the best match in case of the scenario 6.

In scenario 7 we set all the priorities (cost, availability and VM initiation time) equal and we gave 10% cost variation. In this case, the provider with the lowest cost is the potential VM. Since theuser can tolerate 10% cost variation the providers with the cost not exceeding the 110% of the cost of the lowest cost provider are also potential VMs. The results are shown in row #7 and #8 of Table 9.

Table 9. Results Obtained for Different Test Scenarios

#	Scenario #	Provider Name	VM instance Type	Memory (GB)	CPU (GHz)	Storage (GB)	Availability	VM Initiation Time	Approx Cost/ month (USD)
1	1	Provider 1	C	2	1	5	99	30	225
2	2	Provider 2	C	2	1	5	99.5	45	210
3	3	Provider 2	C	2	1	5	99.5	45	210
4	4	Provider 5	C	2	1	5	99	15	265
5	5	Provider 4	C	2	1	5	99.9	15	290
6	6	Provider 4	C	2	1	5	99.9	15	290
7	7	Provider 1	C	2	1	5	99	30	225
8		Provider 2	C	2	1	5	99.5	45	210
9	8	Provider 2	M	4	2.4	50	99.5	45	637.5
10	9	Provider 1	C	4	2	40	99	30	675
11	10	Provider 4	C	4	2	40	99.9	15	880

C=Configurable M=Medium

In scenario 8 we changed the tenure duration to 3 months. RAM, CPU, and Storage variations are set to 20%, 20%, and 50% respectively. The complete search criteria are shown in row #8 of Table 8. Since theuser can tolerate variations in the VM configurations, fixed template VMs also falls within the search criteria. In this scenario, the cost has the highest priority, so the provider with the lowest cost is the potential VM which is Provider 2. Note that in this case we got a fixed template VM (medium) as the best match and the discount 15% is applicable for fixed 'medium' VM instance type in case of the Provider 2 for the tenure range 3-6 months. The result is shown in row #9 of Table 9. In scenario 9 we removed the RAM, CPU and Storage variations which mean users are not tolerant to any variations in their configurations. The complete search criteria are shown in row #9 of Table 8. We do not have the exact match in the fixed template VMs for the user search criteria, so our algorithms select the configurable VMs. Since the cost is having the highest priority the provider with the lowest cost is the potential VM which is Provider 1. In this case, a discount of 10% is applicable for Provider 1 for the tenure range 3-6 months. The result is shown in row #10 of Table 9.

In scenario 10 we changed the priorities. VM initiation time is having the highest priority followed by the cost and availability. Rest of the search criteria is same as scenario 9. Like the previous scenario, we do not have the exact match in the fixed template VMs for the user search criteria, so our algorithms select the configurable VMs. In this case, VM initiation time has the highest priority. In Table 5 we can observe that provider 4 and provider 5 are having the lowest VM initiation time which is 15 minutes. Since the cost has the next priority, it is used as tie breaker. Among the provider 4 and provider 5, Provider 4 is having the lowest cost hence the provider 4 is the best match in this scenario. The result is shown in row #11 of Table 9. Experiments and results obtained prove that our DAS suggests an efficient IaaS provider considering the different search criteria provided by the user. Spending a few minutes to search best IaaS provider using our DAS saves a lot of money for the end user.

7. DAS vs OTHER SOLUTIONS

None of the works mentioned in the related work section address the issue of real-time synchronization of information between the DAS and the IaaS providers. Our Decision Assist System is a comprehensive solution which assists the end users in selecting the efficient IaaS provider/providers by considering parameters like resource requirements, tenure, pricing models/schemes, QoS parameters, etc. It also makes sure that the information (which is required to make the decision of selecting efficient IaaS provider/providers) available at the DAS is in synchronization with that of IaaS providers. The DAS provides options to the end users to set the priorities for cost, availability and VM initiation time, using these options users can set the priorities according to their requirement.

Since all the price details and provider details are stored locally, the search will be much faster. The synchronization of the provider information (VM details, pricing details, etc.) happens in the real time; thismakes sure that user searches are not using obsolete data to identify the efficient IaaS providers. Spending couple of minutes on DAS for searching efficient IaaS providers saves a lot of money to the end users. The DAS provides a platform for the IaaS providers to attract alarge number of customers by giving the services in a competitive manner.

8. CONCLUSION

Price varies from one IaaS provider to the other for the same requirement, and IaaS providers are using various pricing models, pricing schemes. They give different types of VM instances. End users should

select the appropriate provider by considering the various parameters like pricing models/schemes VM heterogeneity etc. Since end users must consider many parameters while selecting an efficient IaaS provider, itwill become a difficult and complex job. In this work, we have designed, implemented and tested a decision-assist system which assists the end users in selecting efficient IaaS providers. Experimental results show that our DAS suggests the best IaaS providers by considering the cost, VM heterogeneity, pricing models/schemes, etc. In this work, we have considered VM initiation time and availability as QoS parameters.

Our future work is to enhance the decision strategies by considering other parameters such as response time, penalty,customer rating, etc. We are also planning to replace the relational database with a document oriented database (ex: MongoDB). Since the different IaaS providers' data will be in the different structure, it will be easy to store the information in a document-oriented database instead of a relational database. IaaS provider like Amazon EC2 provides their own APIs for querying the prices of AWS services [21]. Users can also subscribe to Amazon Simple Notification Service (Amazon SNS), users are notified when prices for the services change through notification service. Other IaaS providers are also expected to provide such services soon. We will be developing APIs to use these services provided by the IaaS providers to update the pricing details and other information in DAS in real time.

REFERENCES

[1] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility", *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] Gurudatt Kulkarni, Jayant Gambhir, Rajnikant Palwe, "Cloud Computing-Software as Service", *International Journal of Cloud Computing and Services Science (IJ-CLOSER)* Vol.1, No.1, pp. 11~16, March 2012

[3] Ergu D, Kou G, Peng Y, Shi Y, Shi Y. "The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment."*The Journal of Supercomputing*, 64: 835–848, 2013.

[4] Mohan Murthy MK, Ashwini JP, Sanjay HA, "Pricing Models and Pricing Schemes of IaaS Providers: A Comparison Study", in *ICACCI '12*, Chennai, India, August 03–05, 2012.

[5] S.K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services", *Future Generation Computer Systems*, vol. 29, no. 4, pp. 1012–1023, 2013.

[6] Michael Smit, Przemyslaw Pawluk, Bradley Simmons, Marin Litoiu, "A Web Service for Cloud Metadata" in *SERVICES*,2012.

[7] Dhaval Limbani, Bhavesh Oza, "A proposed Service Broker Policy for Data Center Selection in Cloud Environment with Implementation", in *IJCTA*, May-June 2012.

[8] Stella Gatziu Grivas, Tripathi Uttam Kumar, Holger Wache, "Cloud Broker: Bringing Intelligence into the Cloud An Event-Based-Approach", *in ICCC*, 2010.

[9] Srijith K. Nair1, Sakshi Porwal2, Theo Dimitrakos1, Ana Juan Ferrer3, Johan Tordsson4, Tabassum Sharif5, Craig Sheridan5, Muttukrishnan Rajarajan6 and Afnan Ullah Khan1, "Towards Secure Cloud Bursting, Brokerage, and Aggregation", in *ECOWS*, 2010.

[10] May Al- Roomi, Shaikha Al-Ebrahim, Sabika Buqrais and Imtiaz Ahmad "Computing Pricing Models: A Survey" *International Journal of Grid and Distributed Computing* Vol.6, No.5, 2013.

[11] Hyun Jin Moon, Yun Chi, Hakan Hacıgümüş, "*SLA-Aware Profit Optimization in Cloud Services via Resource Scheduling*", *In proceedings of 6th World Congress on Services*, IEEE, 2010.

[12] Linlin Wu, Saurabh Kumar Garg and Rajkumar Buyya, "SLA-based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments", *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2011

[13] Y.C. Lee, C. Wang, A.Y. Zomaya and B.B. Zhou, "Profit-driven Service Request Scheduling in Clouds".*In Proceedings of the International Symposium on Cluster and Grid Computing, (CCGrid 2010)*, Melbourne, Australia.

[14] Plan for Cloud, http://www.planforcloud.com/pages/features.html

[15] S.M. Han, M.M. Hassan, C.W. Yoon, and E.N. Huh, "Efficient service recommendation system for cloud computing market", *in Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human (ICIS ' 09)*, Seoul, Republic of Korea, November 2009.

[16] Junping Dong, Qingyu Xiong, Junhao Wen and Peng Li, "Services Recommendation System based on Heterogeneous Network Analysis in Cloud Computing", *Research Journal of Applied Sciences, Engineering and Technology*, volume 7, April 2014

[17] Zhipeng Gui, Chaowei Yang, Jizhe Xia, Qunying Huang, Kai Liu, Zhenlong Li, Manzhu Yu, Min Sun, Nanyin Zhou, and Baoxuan Jin, "A Service Brokering and Recommendation Mechanism for Better Selecting Cloud Services", *PLOS ONE journal*, August 29, 2014

[18] M. Whaiduzzaman, A. Gani, N.B. Anuar, M. Shiraz, M.N. Haque, and I.T. Haque, "Cloud service selection using multicriteria decision analysis", *The Scientific World Journal*, vol. 2014, Article ID 459375, 10 pages, 2014.

[19] Farley B, Juels A, Varadarajan V, Ristenpart T, Bowers KD, et al. More for your money: Exploiting performance heterogeneity in public clouds. In: Proc. of the Third ACM Symposium on Cloud Computing, ACM, 14p., 2012

[20] S. Sundareswaran, A. Squicciarini, and D. Lin, "A brokerage-based approach for cloud service selection", in *Proceedings of the IEEE 5th International Conference on Cloud Computing (CLOUD '12)*, pp. 558–565, Honolulu, Hawaii, USA, June 2012
[21] http://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/price-changes.html

BIOGRAPHIES OF AUTHORS

Mohan Murthy M K is a Ph.D. candidate in Information Science & Engg. at Nitte Meenakshi Institute of Technology, Bangalore. He received his professional degrees BE and MTech from Visvesvaraya Technological University, Karnatka. His research interests are in developing different techniques for effective resource utilization in Cloud computing and to develop strategies to maximize the profit to the vendors and end users in the cloud market.

Dr. Sanjay H A is professor and head of the department of Information Science & Engg. at Nitte Meenakshi Institute of Technology, Bangalore. He has a Ph.D. in Computer Science & Engg. from Indian Institute of Science, Bangalore. His research interests are in performance modeling of parallel applications, scheduling, and rescheduling of a parallel application in a Distributed/Grid computing environment. His present research focuses on is-sues related to Hybrid computing, Cloud computing, and data mining.

Supreeth B M is currently working as a Software Engineer in Misys Financial Software (India) Pvt Ltd, Bangalore. He received his professional degrees BE and MTech from Visvesvaraya Technological University, Karnatka. His research interests are in Cloud Computing and Big Data.