www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 4 Issue 8 Aug 2015, Page No. 13822-13827

Profit Maximization Of SAAS By Reusing The Available VM Space In Cloud Computing

Richa[1], Hari Singh[2]

Computer Science and Engineering, N.C. College of Engineering,Israna
Panipat, India
[1]richavasuja@gmail.com
[2]harirawat@rediffmail.com

***Abstract*- Cloud computing has recently emerged as one of the buzzwords in the IT industry. SaaS layer is the upper layer of the cloud-computing model. It offers consistent access to software applications to the clients over the Internet without direct investment in infrastructure and software. It is a service provided to client in terms of applications running on the cloud-computing infrastructure hosted by the service providers. The main work of this layer is to deliver the software application services over the internet with in the real time when users demand for it in pay per use manner. The main concern of this paper is on how to minimize the infrastructure cost and maximize the SaaS provider profit for that two algorithms are proposed. The proposed algorithms are designed in such a way that works on the mean value which efficiently uses the available VM Space to handle the client request instead of creating a new VM to deliver the services. And by that the SAAS provider has not to pay the extra penalty cost and hence they get a profit to serve the request of clients with their satisfaction and without violation of SLA.**

***Keywords*- Cloud Computing, Software as a Service, Virtual Machines, Service Level Agreement.**

I INTRODUCTION

"Cloud Computing" one of the most imperative and new field in the current world. It has come out to be a smart and practical way of altering the whole computing world. It is a new idea for delivery of applications and resources to the end users as pay per use manner over the internet [8]. Due to its advantages cloud has touched each and every important field and proved beneficial and so it has been more and more adopted in areas such as banking, e-commerce [6][7] and many more areas like education, business, health etc. It is required to have an efficient use of resources by different application areas [3] [4]. It enables on-demand provisioning of computational resources, in the form of virtual machines (VMs) which can be deployed in a cloud provider's datacenter. The most important features of CC include lower cost, incremental scalability, consistency and fault tolerance, service-oriented, utility-based, virtualization and SLA [1] [2].
SAAS is the most basic form of CC where the applications requested by the end users run. In order to serve the request of the end user sometimes the SAAS providers have to pay the extra infrastructure cost which can be the penalty cost due to SLA violation. Therefore, SaaS providers are looking into solutions that minimize the on the whole infrastructure cost without adversely disturbing the customer's satisfaction level. Hence, the concern of this paper is on exploring policies and strategies to minimize the required infrastructure to meet customer demand in the context of SaaS providers offering hosted software services.

The Figure 1 depicts a SaaS model for serving clients in Cloud, how request comes to the SaaS provider and how the SaaS provider will fulfill the clients request. A client sends a request for utilizing the various application services offered by a SaaS provider, who uses three layers, namely the application or SaaS layer, the platform or PaaS layer and the infrastructure or IaaS layer, to satisfy the customer's request. The application or SaaS layer manages all the application services that are offered to customers by the SaaS provider. The platform or the PaaS layer includes the mapping, decision making and scheduling policies for translating the client's Quality of Service (QoS) necessities to infrastructure level parameters and allocating Virtual Machines (VMs) to serve their requests. The infrastructure layer controls the actual initiation and removal of VMs. The VMs can be leased from IaaS providers such as Amazon EC2 [5] or private virtualized clusters owned by the SaaS provider. In both cases, the minimization of the number of VMs will deliver savings. The savings or the profit are greater when SaaS providers use the third party IaaS providers since no capital expenditure is required for the maintenance of the applications or the software's.
So in order to accomplish the SaaS provider's intention to maximize profit and customer satisfaction levels, this paper proposes cost effective mapping and scheduling policies which minimize the cost by optimizing the resource allocation within a VM.

DOI: 10.18535/ijecs/v4i8.37

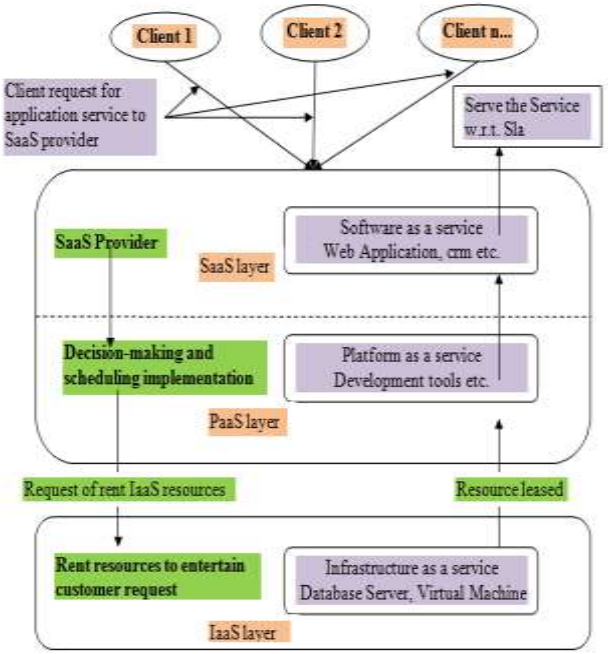

Figure:1 SaaS Model

II RELATED WORK

Linlin Wu et.al [9] author proposed architecture SLA-based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments, which is based on SLA. SaaS providers want to minimize infrastructure cost and SLA violations. In this paper author proposed the two algorithms which are responsive according to the specific condition and provide efficient resources allocation with minimum use of third party resources and maximize the profit of SaaS provider. In this ProfminVmMaxAvaiSpace and ProfminVmMinAvaiSpace are the algorithms which are used to provide the efficient resources allocation with less infrastructure cost. According to algorithm first a SaaS provider can maximize the profit by minimizing the resource cost, which depends on the number and type of initiated VMs.

Raj Kumar Buyya et.al [11] proposed solutions are able to maximize the number of accepted users through the efficient placement of request on VMs leased from multiple IaaS providers. We take into account various customer's QoS requirements and infrastructure heterogeneity. The key contributions of this paper are twofold author proposed a system and mathematical models for SaaS providers to satisfy customers. And proposed three innovative admission control and scheduling algorithms for profit maximization by minimizing cost and maximizing customer satisfaction level. Author proposed three algorithms which can maximize the profit by reducing the infrastructure cost, which depends on the number and type of initiated VMs in IaaS providers' data centre. Author mainly used three algorithms, which are ProfminVM, ProfRS, and ProfPD.

Hilda Lawrance et.al [10] Cloud computing is a very popular area in current world which is rising very fast and the futures of the field seems really broad and strong. In order to provide quality of service in the cloud environment is a highly challenging task. The cloud clients should obtain reliable services from the provider based on their desire. In the particular cloud computing service, the resource allocation process is based on quality of service and cost of resource. The provider should allocate the resources in a proper way to render good services to the clients. This paper elucidates an elegant survey made on the different resource allocation method environment.

III SYSTEM IMPLEMENTATION

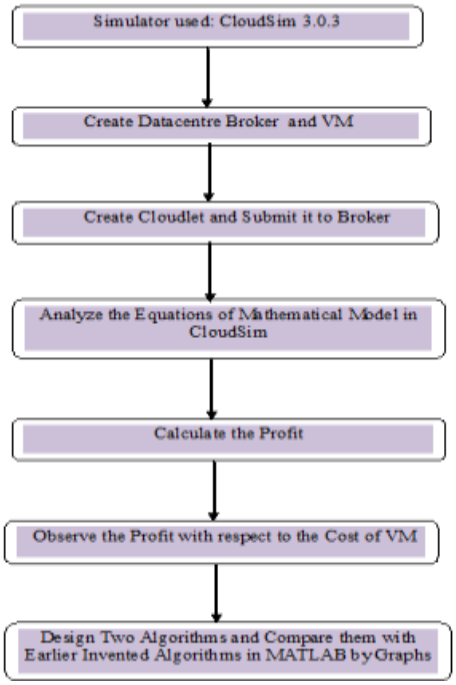

Figure:2 Design Flow

1. MATHEMATICAL MODEL

In the mathematical model, the mathematical equations are used which can help the SaaS provider to increase their profit. The profit ($prof^{cl}_{ki}$) shows the profit gained by SaaS provider to serve the client request cl using specific vm (VM_{ki}). **T**he request length is in which the user will define the length of the service which they want from the service provider. Then the total profit gained by the SaaS provider corresponding to specific number of customers request defined in equation 1 below.

DOI: 10.18535/ijecs/v4i8.37

$$\sum_{cl=1}^{CL} \text{prof}_{ki}^{cl} = \sum_{cl=1}^{CL} \text{PriceSrv}^{cl} * \sum_{cl=1}^{CL} \text{ReqLen} - \sum_{cl=1}^{CL} \text{Cost}_{ki}^{cl}$$

Where, ∀i ∈ I, k ∈ K, cl ∈CL

(1)

For a client request cl, the final service price PriceSer^{cl} is subjected to the product type and account type. Let Cost^{cl}_{ki} indicate the cost for serving request cl with priceVm_{ki} and it depends on the VM cost (PrcieVm_{ki}) and penalty cost ($\text{PenaltyCost}_{ki}^{cl}$.).

$$\text{Cost}_{ki}^{cl} = \text{priceVm}_{ki} + \text{PenaltyCost}_{ki}^{cl}$$

Where, ∀i ∈ I ,k ∈ K, cl ∈CL

(2)

The cost of VM depends on the type of VMk that is used by the SaaS provider to handle the client request. In addition, the price of VMi with type k (PriceVM_{ki}), the service initiation Time (Intime_{ki}) and service request length (or contract length ReqLen) of customer request cl. Equation (3) defines the VM Cost.

$$\text{VmCost}_{ki}^{cl} = \text{priceVm}_{ki} * (\text{Intime}_{ki} + \text{ReqLen}^{cl})$$

Where, ∀i ∈ I ,k ∈ K, cl ∈CL

(3)

The price of vm (pricVm) depends on the processing cost (procsVm_{ki}), data transfer cost ($\text{DataTrsfrCost}_{ki}$). Equation (4) defines the PriceVm_{ki} below.

$$\text{priceVm}_{ki} = \text{procsVm}_{ki} + \text{DataTrsfrCost}^{cl}$$

Where, ∀i ∈ I ,k ∈ K, cl ∈CL

(4)

In Eq. (5), penalty delay cost (PDC_{ki}) is how much the service provider has to give discount to users for SLA violation. It is dependent on the penalty rate (β) and penalty delay time (PDT_{ki}) period. In equation (5) we define the penalty delay function.

$$PDC_{k\,i} = \beta * PDT_{ki}$$

Where, ∀i ∈ I ,k ∈ K, cl ∈CL

(5)

To serve the new request to the client, SaaS provider either can allocate a new VM or schedule the request on an already initiated VM. If service provider schedules the new request on an already initiated VMi , the new request has to wait until VM I becomes available. The time for which the new request has to wait until it start processing on VM i is $\sum_{r=1}^{R} \text{procT}_{ki}^{r}$, where R is the number of request yet to be processed before the new request. Thus, PDT_{ik} is given by:

$$\text{PDT}_{ik} = \sum_{r=1}^{R} \text{procT}_{ki}^{r} + \text{procT}_{ki} - \text{DL}, \quad \text{If new VM is not initiated}$$

$$\text{PDT}_{ik} = \text{procT}_{ki} + \text{Intime}_{ki} + \text{DTT}_{ik} - \text{DL}, \quad \text{If new VM is initiated} \qquad (6)$$

DL is the maximum time user would like to wait for the results. DTT_{ik} is the data transfer time which is the summation of time taken to upload the input (inDT_{ki}) and download the output data (outDT_{ki}) from the VM_{ki} on IaaS provider. The data transfer time is given by:

$$\text{DTT}_{ik} = \text{inDT}_{ki} + \text{outDT}_{ki}$$

Where, ∀i ∈ I ,k ∈ K, cl ∈CL

(7)

Thus, the response time (RT_{ki}) for the new request to be processed on VM_{ki} of IaaS provider is calculated in equation (8) and consists of VM initiation time (iniT newi jl), request's service processing time (procT_{ki}), data transfer time (DTT_{ki}), and penalty delay time (PDT_{ki}).

$$\text{RT}_{ki} = \sum_{r=1}^{R} \text{procT}_{ki}^{r} + \text{procT}_{ki}, \quad \text{If new VM is not initiated}$$

$$\text{RT}_{ik} = \text{procT}_{ki} + \text{Intime}_{ki} + \text{DTT}_{ik}, \quad \text{If new VM is initiated} \qquad (8)$$

If the SAAS provider is able to entertain the user request in the same VM then the cost of VM will remain constant but the profit made is more as shown in the figure.

Figure:3 Use of Same VM

If new VM is to be used for new user request then the cost of VM will be very high and the profit made by the SAAS provider will be very less as he has to pay the penalty cost in this case as shown in the figure:

DOI: 10.18535/ijecs/v4i8.37

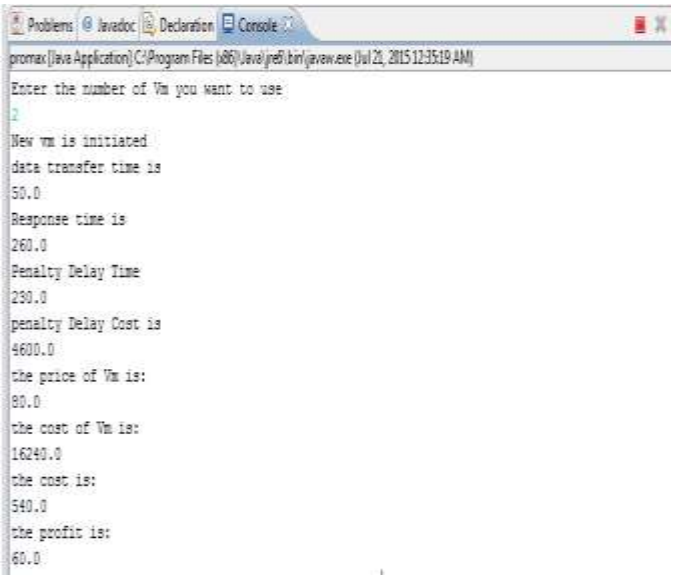

Figure:4 Less profit and high cost

2. *PROPOSED ALGORITHMS*

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' data centre. Therefore, two algorithms are proposed and designed in such a way to minimize the number of VMs by maximizing the utilization of already initiated VMs.

ALGORITHM 1 : MEANVMAVAILSPACE:

Input request (*r*) with QoSparameters ,VM*k*

Output Boolean

Functions request (*r*) with QoS parameters , VM*k*

First Time Rent (*r*){

1 If (there is initiated VM*k* with type that matches to the VM type requested by *r*) {

2 If (VM*k* deployed the same product type as required by *r*) {

3 For each initiated VM*k* with type *i* (VM*ki*){

4 If (VM*k* has enough space to place *r***){**

5 put VM*k* into *vmList*

6 }

7 }

8 Sort(*vmList***)** according to the available space

9 Calculate Mean Available Space = Sum of available VM capacities/No. of available capacities

10 If (Mean Available Space is in nearest float number then round it off)

11 Schedule to process *r* on ***Mean Availilable Space***

12 This VM is called **MeanVMAvailSpace**

13 }

14 Else {

15 Initiate new VM with type *i* and deploy the product type as request *r* required

16 }

17 }

18 Else While (*i*+*j*<=*I*) loop {

19 If (there is initiated VM with next type *i*+*j*, where type *i*+*j* matches to the VM type required by request *r*) {

20 Repeat from Step 2 to 16

21 j++

22 }

23 }

24 }

Upgrade(*r*) {

1 If (upgrade type is 'add account'**) {**

2 get Id *k* and type *i* of VM, which processed the previous request from same company as *r*

3 If (VM*k* has enough space to place *r*){

4 Schedule to process *r* on VM*k*.

5 }

6 Else {

7 Repeat step 1 to 24 of **First Time Rent(*r*)**

8 Transfer data from old VM to new VM

9 Release space in old VM

10 }

11 }

12 If (upgrade type is **'upgrade service'){**

13 Repeat step 7 to 9 of **Upgrade(r)**

14 }

15}

ALGORITHM 2 : HIGHMEANVMAVAILSPACE

Input request (*r*) with QoSparameters ,VM*k*

Output Boolean

Functions request (*r*) with QoS parameters , VM*k*

First Time Rent (*r*){

1 If (there is initiated VM*k* with type *i* matches to the VM type requested by *req*) {

2 If (VM*k* deployed the same product type as required by r) {

3 For each initiated VM*k* with type *i* (VM*ki*){

4 If (VM*k* has enough space to place *r***){**

5 put VM*k* into *vmList*

6 }

7 }

8 Sort(*vmList***)** according to the available space

9 First calculate Mean Available Space = Sum of available VM capacities/ No.of available capacity

10 If (Mean available space is in nearest float number then round it off)

11 Schedule to process *r* on ***VM available capacity higher than the calculated Mean Available Space***

12 This VM is called the **HighMeanVMAvailSpace**

13 }

14 Else {

15 Initiate new VM with type *i* and deploy the product type as request *c* required

16 }

17 }

18 Else While (*i*+*j*<=*l*) loop {

19 If (there is initiated VM with next type *i*+*j*, where type *i*+*j* matches to the VM type required by request *r*){

20 Repeat from Step 2 to 16

21 j++

22 }

23 }

24 }

Upgrade(*r*) {

1 If (upgrade type is 'add account'**) {**

2 Get Id *k* and type *i* of VM, which processed the previous request from same company as *r*

3 If (VM*k* has enough space to place *r*){

4 Schedule to process *r* on VM*k*.

5 }

6 Else {

7 Repeat step 1 to 24 of **First Time Rent(*r*)**

8 Transfer data from old VM to new VM

9 Release space in old VM

10 }

11 }

12 If (upgrade type is **'upgrade service'){**

13 Repeat step 7 to 9 of **Upgrade(r)**

14 }

15}

TO evaluate the proposed algorithm we consider 250 client request for the software service. In this, we consider only two-parameter that are impact of arrival rate and service up gradation on proposed algorithm.

To observe the impact of arrival rate we vary the arrival request factor and rest factor remain constant. Five experiments are being done using different value to observe the effect on the proposed algorithm and compare it in MATLAB.

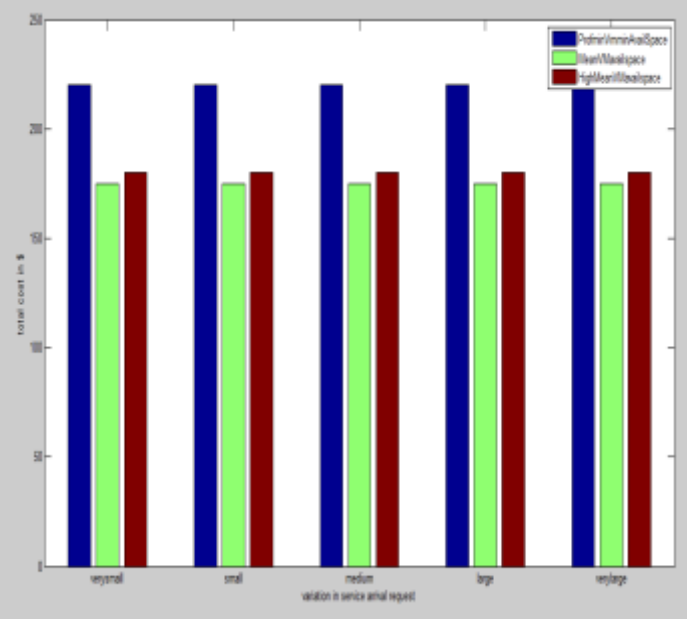

Figure:5 Total Cost in Variation to Request Arrival

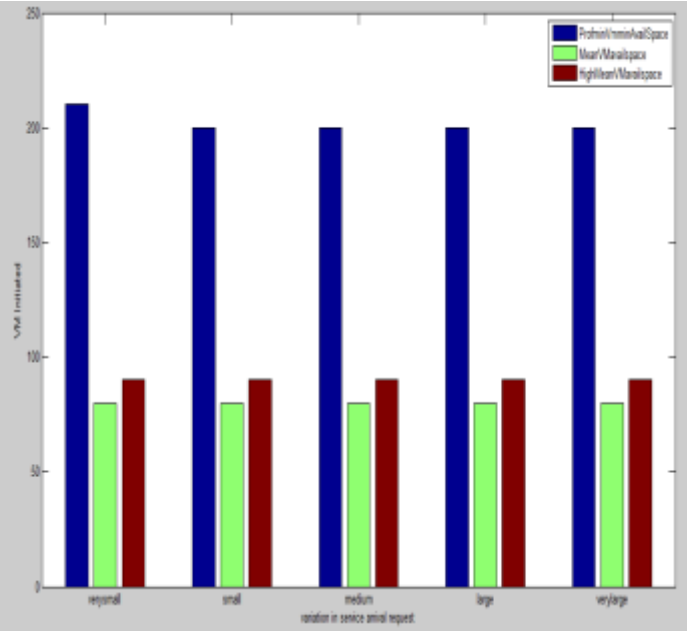

Figure:6 VM initiated in Variation to Request Arrival

IV CONCLUSION

The proposed system will minimize the extra infrastructure cost or the penalty cost which the SaaS provider has to pay by initiating the new VM every time to entertain users request. The proposed algorithms will maximize the profit and also considers the Customer satisfaction level. It will give the expected level of customer satisfaction and hence will minimize the penalty cost which is the result of SLA violation. It will handle the dynamic change in customers request and will work properly.

REFERENCES

[1] G. Boss, P. Malladi, D. Quan, L. Legregni and H. Hall, Cloud Computing (White Paper), IBM, october2007,http://download.boulder.ibm.com/ibmdl/pub/software /dw/wes/hipods/Cloud_computing_wp_final_8Oct.pdf , accessed on May. 19, 2009.

[2] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-Oriented Cloud and Atmospheric Computing: Hype, Reality, and Vision", Proc. of 10th IEEE International Conference on High Performance Computing and Communications (HPCC-08), 5-13, Dalian, China, September, 2008.

[3] I. Gandotra,P.Abrol,"Cloud computing :A new paradigm for Education",ICAET10, 2010, pp 92.

[4] I. Gandotra, P Abrol,"cloud computing a new Era of Ecommerce", book chapter in Strategic Service management, Published by Excel Books, pp 228, 2010

[5] Amazon Elastic Compute Cloud (EC2), http://www.amazon.com/ec2/ [22 Jan 2013]

[6] 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", 25(6), (pp. 599-616), Elsevier Science, Amsterdam, The Netherlands.

[7] M. A. Vouk, "Cloud Computing-Issues, Research and Implementation". In Proceedings of 30th International Conference on Information Technology Interfaces (ITI 2008), Dubrovnik, Croatia.

[8] I. Foster, Y. Zhao, I. Raicu and S. Lu, "Cloud Computing and Grid Computing 360 Degree Compared", Grid Computing Environments Workshop, Austin, 2008.

[9] SLA-based Resource Allocation for Software as a Service Provider (SaaS) in cloud Computing environments" in 2011 11th IEEE/ACM international symposium on cluster, Cloud and grid, computing.

[10] Hilda Lawrance et.al," Survey on Resource Allocation Methods in Cloud Computing" in International Journal of Engineering Sciences and Information Technology, March 2013.

[11] Linlin Wu ∗, Saurabh Kumar Garg, Rajkumar Buyya, "SLA-based admission control for a Software-as-a-Service provider in Cloud computing environments" Journal of Computer and System Sciences 78 (2012) 1280–1299,23December,2011.