SLA Based Resource Provisioning-A Method for SaaS Applications in Cloud Computing

Vivek.V.P, Santhosh.D, J.Udaya Kumar

***Abstract*— The challenges like licensing, distribution configuration, and operation of enterprise applications associated with the traditional IT infrastructure, software sales and deployment model are solved using cloud technique. Migrating from a traditional model to the cloud model provides ongoing revenue for software as a service (SaaS) provides and also reduces the maintenance complexity and cost for enterprise customers. When more customers delegate their tasks to cloud providers, service level agreement (SLA) between consumers and providers emerge as a key aspect. Due to the dynamic nature of the cloud, the quality of service (QOS) should be continuously monitored to enforce SLAs.**

The proposed Methodology include the customers driven SLA-based resource provisioning algorithms to minimize resource and penalty cost and improve customer satisfaction level by minimizing SLA violations. The provisioning algorithms take into account customer profiles and providers quality parameters (e.g. response time) to handle dynamic customer requests and user infrastructure level heterogeneity for enterprise systems. The customer side parameters (proportion of upgrade requests) and infrastructure parameters (service initiative time) to compare algorithms

***Index Terms*— cloud computing, resource provisioning, SLA, Resource scheduling, quality of service**

I. INTRODUCTION

Cloud computing has become the new trend for delivery of application, platforms, and computing resources (processing power/bandwidth/storage) to customers in a "pay as you go model". Cloud computing has 3 categories software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS). When complexity of the applications has given the administration difficulty becomes evident. Then the enterprises choose to outsource some the applications to third party SaaS providers enabled by cloud computing. The SaaS model has been increasingly adopted for distributing many enterprise software systems, such as banking and e-commerce business software due its flexibility, scalability and cost effectiveness. The enterprises establish a service level agreement (SLA) with the SaaS providers which ensures the quality of service (QOS) requirements are met. If any party violates the SLA, the defaulter need to pay penalty according to the clauses defined in the SLA.

The need to ensure software response time, enterprise software providers in the industry allocate dedicated VMs for each customer. However, this will lead to the wastage of hardware resources due to the underutilized resources at non-peak load. When the provider violates the Predefined response time in the SLA, the customer satisfaction level (CSL) is impacted badly and the SLA violation causes penalty. The comparison between how much faster the actual response time with the minimum response time documented in the SLA is defined by service quality improvement (SQI).

To maximize the CSL, algorithms are designs which will reduce the SLA violation by request reservation and request re-scheduling. The proposed methodology includes customer driven heuristic algorithms to minimize the total cost by resource provisioning. The algorithm also takes into account the customer profiles such as the credit level and the multiple Key Performance Indicators (KPI) criteria which will improve the SaaS application's performance quality rating. The KPIs are considered for performing quality rating, one from providers' perspective: cost and two from customer perspective: service response time and SLA violations.

II. RELATED WORK

Experiments on market based resource allocation was started in 1980's, Market based resource allocation methods are mostly designed for fixed number of resources. The SAAS providers are mainly aimed on two objectives, First is to minimize cost and profit maximization through resource allocation and second, maximizing customer satisfaction level (CSL).

The key area in which major attention need to be given on is USER driven SLA-based economic-oriented resource provision with dynamic number of resources. The usage pattern and usage prediction are also to be taken care off. Web usage mining is an application of data mining techniques.WUM is used to extract usage pattern from web check stream. Web usage mining was grown rapidly in the past few years and in the current WUM area, data has been classified as content, structure usage and user profile. The first three data categories are completely dependent on web sites but not the e-commerce transaction. Currently usage prediction algorithms like history based, sequence based and Markov-based algorithms are used for content, structure and usage data categories.

In the process of calculating credit level rather than focusing on designing strategies, the user profile using history based method are used. In the area resource allocation and SLA

Manuscript received April 12, 2015.

Vivek.V.P Is Pursuing Final Year B.Tech In Cse At Perunthalaivar Kamarajar Engineering And Technology, Karaikal.His Research Interest Incudes Cloud Computing And Networking

Santhosh.D Is Pursuing Final Year B.Tech In Cse At Perunthalaivar Kamarajar Engineering And Technology, Karaikal.His Research Interest Incudes Cloud Computing And Networking

J.Udaya Kumar Working As Professor At Department Of Computer Science And Engineering Perunthalaivar Kamarajar Engineering And Technology,Karaikal

management in both Grid and cloud computing are detailed as follows.

A. Grid computing:

There are different scheduling strategies in computational grids and rather than focusing on scientific tasks which run for short term more attention is given to the transaction based application which run for very long term. Since the main focus is on the cost and SLA violations, the evaluation metrics are different which focused on response time and utilization.

Market based resource allocation algorithm for grid computing have certain similarities with Methodologies. Firstly, the consideration of state based and pre-emptive strategies. The state based strategy indicates all the resource allocation based on the current service/system rate and pre-emptive strategy allows tasks assign to a resource which can be migrated to other resources. Secondly, In the Market based resource allocation, the customer requests with multiple QOS parameters using dynamic and flexible resources Instead of QOS parameters using fixed number of resources.

The QOS guided task scheduling algorithm on Grid is presented in which the bandwidth is considered as one of the major QOS parameter. The strategy is based on minimizing cost by considering QOS parameter on both customer and provider side without aiming on the earliest competition work. To minimize the resource consumption for serving request and executing them within a deadline data intensive transaction based application, which run for long term Instead of complete intensive Independent application, in which relatively short term are used.

SLA-based dynamic scheduling algorithm of distributed resources for streaming is presented. After evaluating various SLA-based scheduling heuristics on parallel computing resources with two evaluation metrics: resource (number of CPU nodes) Utilization and income the main attention is given to scheduling enterprise application on VMs in cloud computing environment.

B. Cloud computing:

In cloud computing virtualization is a core technology, the VM placement has become crucial in the resource management and scheduling while the virtualization at the storage level and operating system entering the mainstream. The prediction system is used to enable the scheduling policies to discard the service of requests if the available resource capability is not able to complete the request before its deadline. The prediction system contributed on minimizing the resource consumption for serving requests and executing them before its deadline.

After evaluating various SLA-based scheduling heuristics on parallel computing resources using resource (number of CPU nodes) Utilization and income as evaluation Metrics derived an SLA-based dynamic scheduling algorithm of distributed resources for streaming. Various algorithms for assignment of VMs are investigated. Similarly, the resource provisioning and VM placement is presented. A dynamic consolidation mechanism for homogeneous resources is designed. These related publications did not consider uncertainty of future demand or monetary cost. To maximize profit and to minimize the total cost for the SaaS providers, a dynamic heuristic based VM placement methodology that did not focus on customer-driven scenario is used.

QOS parameters providers are mainly considered on the resource provider’s side but not in the user’s side. To gain profit and improve reputation, the profit driven service request scheduling for workflow is investigated. In contrast, focused on

a) SLA driven QOS parameters on both user and provider sides, and
b) Solving the challenge on dynamic changing customer reques

An allocation algorithm which minimizes the number of VM migrations during resource reallocation is presented. After applying stochastic programming approach in multiple phases in cloud computing, for optimization the resource provisioning cost is minimized by considering the uncertainty. The genetic algorithms presented in the virtualized environments is a resource allocation algorithm enterprise application; however, the genetic algorithm require long execution time and create a preplanning schedule which increases the probability of SLA-violation in the cloud computing environments, where customers need to be served immediately. The previous work has been updated by two extended algorithm, which takes in account of QOS parameter namely credit level. In order to optimize total cost and SLA violations two strategies, Resource provisioning and request migration are used.

III. System Model

The SaaS model for serving the customer request in the cloud is shown in the figure. The SaaS provider uses a three layered model, called the application layer, the platform layer of infrastructure layer, to complete the customer requests the secured application services, such as the Customer Relationship Management (CRM) or Enterprise Relationship Packages (ERP) application provided by the SaaS provider to the customers are managed by the application development, deployment and it is also responsible for mapping and scheduling policies for translating customer side QOS requirements to infrastructure level parameters.

To measure the SaaS providers’ QOS the mapping policy considers customer profiles and KPI criteria. The infrastructure layer performs the virtualization VM management service and controls the actual initiation and termination of VMs resources, which leased from the IaaS providers. The minimization of these VMs will deliver savings for the providers such as Amazon EC2 or own private virtualized clusters.

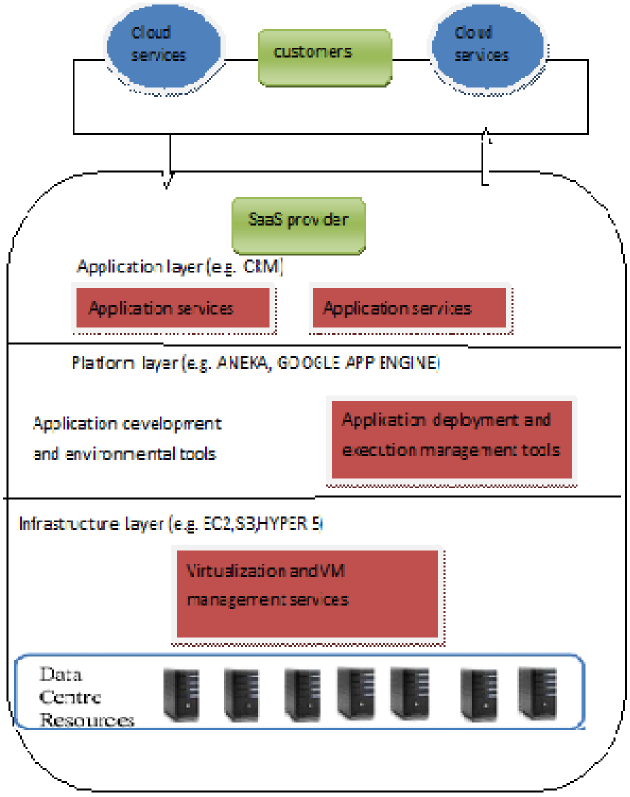
Cloud services
customers
Cloud services
SaaS provider
Application layer (e.g. CRM)
Application services
Application services
Platform layer (e.g. ANEKA, GOOGLE APP ENGINE)
Application development and environmental tools
Application deployment and execution management tools
Infrastructure layer (e.g. EC2,S3,HYPER 5)
Virtualization and VM management services
Data Centre Resources

Fig1:a system model for SaaS layer structure

A. *Actors:*

The actors involved in this system model are described below along with their objectives, activities and constraints.

B. *SaaS providers:*

SaaS providers provide Web-based enterprise software as a service to customers. The mean objective of SaaS providers is to minimize cost and SLA violations which are achieved by the customer driven SLA-based resource provisioning algorithms for web based enterprise application.

A SaaS service provider ‘A’ offers CRM or ERP software packages with three product editions such as standard, professional enterprise etc and each product edition with fixed price. In this service model a company B submits its ‘First Time Rent’ request with a product edition (standard) and additional number of accounts, the provider provides the log-in information to the customer after allocating the resources.

When the company B needs to update the product edition or upgrade service by adding additional user accounts, In this case sometimes a new VM is created and content will be transferred from the previous to the nee VM. In this scenario the provider has to handle these on-demand requests in line with the SLA which contain provider’s pre-defined parameters and the customer specified QOS parameters such as

- Product edition (PE): it is defined as the software product package that is offered to the customer by the providers. For example, SaaS offers standard, professional and enterprise product editions.

- Request type (reqT): the customer request type may be first time rent or a service package request. A service upgrade includes two types ‘add account’ and ‘upgrade product’.

- Contract length (cl): how long the customer is going to use software service.

- Number of accounts (Nacc): the actual number of user accounts the customer need to create, and restricted by the type of product edition.

- Number of records (NR): The average numbers of records the customer can create for each account during a transaction and may be this this impact the data transfer time during service upgrade.

- Response time (Tresp): it represents the time taken by the provider to process a request. An SLA violation occurs when the actual response time is longer than it was agreed in the SLA.

- Penalty conditions: when the SLA is violated the provider need to pay a penalty, which is based on the delay in the response time to the customer. There will be different penalty for each request type and the penalty rate is the monetary cost incurred to the providers for unit time delay in serving the customer request.

The VM images are used by the infrastructure layer to create instances on their physical infrastructure according to mapping decisions. The following infrastructure property include

1. VM types (L): the type of VM image that can be initiated, they may be small, large or medium.

2. Service initiation time (SIT): it describes how long it takes to initialize the service, which includes the VM initiation time and installation time.

3. Service processing time (SPt): it is defined as the time taken by the SaaS providers to process an operation.

4. VM price (VM price): how much it costs to use a VM for the customer request per hour

5. Data transfer time (DTT): how long it takes to transfer one GB of record from one VM to the other which will be depend on the network bandwidth.

C. *Customers:*

When customer registers on the SaaS providers portal, their profile information is gathered by giving forms for registration. This may include company name (comp name), size (comp size) and future interest expression (future interest)

D. *Mathematical Models :*

a) *Customer Profile Model:*

Credit level (credit level): it is used to measure the credibility of a customer, which depends on the value of the company and credit level factor.

Credit level = comp Type value*CF

Comp type is the company type value which is categorized based on the range of company size. The credit level factor (CF) is the ratio of customers actual upgrade value and historical update request.

CF= (actual Upgrade Value) / (future Interest Value)

For example company "B" expresses a future interest to add 2 user accounts, then the future interest value is 2 and they add only one account then credit level is ½=0.5. if no future interest then the credit level is 0.

b) *Cost model:*

Let C be the number of requests and represent the customer id. At given time t, a customer submits service request to the SaaS provider, so the total cost will be

Cost= VM cost + penalty cost.

IV. Resource provisioning algorithm

The main objective for SaaS providers is to minimize cost and SLA violation which is trying to achieve by resource provisioning strategies. The methodology for resource provisioning is in the best fit algorithm in which the profit maximization and minimizing the cost by sharing the minimum available space VMs. The algorithm is designed to minimize the number of VMs by utilizing the same which is already initiated one for serving other user requests as well.

The algorithm will be avoiding the SLA violations of existing request, by not allocating new request to the initiated VM is the new can cause an SLA violation to the existing customers. The best sit algorithm minimizes the number of initiated VMs to minimize cost but there is a probability of penalty cost. For example, when a new customer is requested to add more accounts on the VM which has been fully occupied by other requests, initially a new VM will be more expensive than penalty delay.

The solution has been given by two algorithms.

A. *Algorithm 1: BF reserve Resource*

To optimize the cost caused by adding new accounts, the BF reserve resource algorithm provides more resources than requested based on customer credit level. When a request credit level is greater than provider's expected value, additional resources will be granted.

```
Let 'pe' be the product type and 'Nacc' be the number of accounts required by request c
Let L be type of VM which can serve c after applying mapping strategy.
For each VM 'i' of type 'l' from 'L' to 'Large'
{
    Let vmList = GetVMlist (l, pe,Nacc)//get list of VMs of type l which can serve request 'C'
    If (vmList is empty)
        Continue;
    Else;
    {
        Allocate capacity of VMmin with minimum available space in vmList to request 'C'
        CreditLevel = getCreditLevel (profile information)//get the credit level for request 'C'
        If (CreditLevel>=Threshold)
            Update the available capacity of VMmin to (VMmin's available capacity - ac(future interest))
        Else
            Update the available capacity of VMminto(VMmin's available capacity - Nacc)
        Break;
    }
}
If (request c is still not served)
{
    Initiate a new Vm of type L and deploy the product type p on VM
    Allocate capacity of the new VM to request c
    Update the available capacity of the new VM to (available capacity - Nacc)
}

Upgrade(C)
If (upgrade type is 'add account')
{
    Get VMil which is processing the previous request from the same customer c
    If (VMil has enough space to serve request c and guarantee SLA objectives of existing requests)
    {
        Process request c using VMil
    }
    Else
    {
        Let ac be the number of account that are already by the customer.
        Let new ac be the number of more accounts requested by the customer
        Using similar process as of the function First Time Rent(c) search a new VMil which can serve request with (Nacc + new ac) accounts
        Transfer data from VMil to new VMil
        Release the space in old VMil
    }
}
If (upgrade type is 'upgrade service')
```

```
{
    Get the VMil which processed the previous request from the same customer c
    Using similar process as of the function First Time Rent (C) search a new VMil which can serve the request
    Transfer data from VMil to new VMil
    Release the space in old VMil
}
```

B. Algorithm 2: BF Reschedule Request

To prevent the penalties caused by upgrading the product edition and it also further reduces the penalty by rescheduling accepted request, which leads to a reduction of SLA violations and total cost. This algorithm is designed in a way that all VMs are deployed with the software package which will reduce the resource discovery and content migration time for rescheduling accepted request.

```
Let ‘pe’ be the product type and ‘Nacc’ be the number of accounts required by request c
Let vmList= GetVMlist (l, pe,Nacc) //get list of VMs of type l which can serve request ‘C’
IF (vmLlist  is empty)
{
    Allocate capacity of VMmin with minimum available space in vmList to request ‘C’
    CreditLevel = getCreditLevel (profile information)//get the credit level for request ‘C’
    If (CreditLevel>=Threshold)
        Update the available capacity of VMmin to (VMmin’s available capacity - ac(future interest))
}
Else
{
    Update the available capacity of VMminto(VMmin’s available capacity - Nacc)
    Initiate a new Vm of type L and deploy the product type p on VM
    Allocate capacity of the new VM to request c
    Update the available capacity of the new VM to (available capacity - Nacc)
}
Upgrade(C)
{
    If (upgrade type is ‘add account’)
    {
        Get VMil which is processing the previous request from the same customer c
        If (VMil has enough space to serve request c and guarantee SLA objectives of existing requests)
        {
            Process request c using VMil
        }
        Else
        {
            Let ac be the number of account that are already by the customer.
            Let new ac be the number of more accounts requested by the customer
            Using similar process as of the function First Time Rent(c) search a new VMil which can serve request with (Nacc  + new ac) accounts
            Transfer data from VMil to new VMil
            Release the space in old VMil
        }
    }
    If (upgrade type is ‘upgrade service’)
    {
        Get the VMil which processed the previous request from the same customer c
        If (the available space of VMil is less than request c required in VMil)
        {
            If (migrating c generate minimum penalty cost // after migrating all request,available space in VMil is still than request c required)
            {
                 Find or initiate the vm where new and previous requests generate minimum penalty cost
                Migrate  c and assign c to the VMs found or initiated in last step
                Transfer all the data to this VM
            }
            Else
            {
                Find or initiated the VM where migrating other requests generate minimum penalty cost
                Migrate  c and assign c to the VMs found or initiated in last step
                Transfer all the data to this VM
            }
            Release the space in old VMil
        }
        Else
        {
            Allocate c to VMil
        }
```

V. CONCLUSION

In cloud computing, primarily three types of on demand services are available to the customers they are software as a service(SaaS),platform as a service(PaaS) and infrastructure as a service(IaaS). The methodology used here focused on the explicit aim of cost minimization while maximizing CSL by minimizing the number of SLA violations. To achieve the goal, the customer profiles and KPI criteria are used while mapping and scheduling mechanisms to deal with the dynamic demands and resource level heterogeneity.

The two customer driven algorithms which will consider the various qos parameters from both customer's and provider's perspectives using resource reservation and request rescheduling strategies respectively and in addition, it also used to find out how many resources should be reserved to further optimize the solution. There is a scope in exploring

1. The SLA negotiation process in cloud computing to improve resource provisioning for multi-tier applications, customer satisfaction level.

2. Considering other pricing strategies such as spot pricing for minimizing the cost for service providers.
3. Modeling the inaccuracy of customer information and its impact by exploring credit level calculation based on the usage pattern an usage prediction technologies.

REFERECES

[1] C.S. Yeo and R. Buyya, ''Service Level Agreement Based Allocation of CLUSTER RESOURCES: HANDLING PENALTY to Enhance Utility,'' in Proc. 7th IEEE Int'l Conf. Cluster, Bostan, MA, USA, 2005, pp. 1-10.

[2] Y.C. Lee, C.Wang, A.Y. Zomaya, and B.B. Zhou, ''Profit-Driven Service Request Scheduling in Clouds,'' in Proc. 10th Int'l Symp. CCGrid, Melbourne, Australia, 2010, pp. 15-24.

[3] O.F. Rana,M. Warnier, T.B. Quillinan, F. Brazier, andD. Cojocarasu, ''Managing Violations in Service Level Agreements,'' in Proc. 5th Int'l Workshop GenCon, Gran Canaris, Spain, 2008, pp. 1-10.

[4] D.E. Irwin, L.E. Grit, and J.S. Chase, ''Balancing Risk and Reward in a Market-Based Task Service,'' in Proc. 13th Int'l Symp. HPDC, Honolulu, HI, USA, 2004, pp. 160-169.

[5] Y. Yemini, ''Selfish Optimization in Computer Networks Processing,'' in Proc. 20th IEEECDC, San Diego, CA,USA, 1981, pp. 374-379.

[6] I. Popovici and J. Wiles, ''Proitable Services in an Uncertain World,'' in Proc. 18th SC, Seattle, WA, USA, 2005, p. 36.

[7] 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,'' Fut. Gener. Comput. Syst., vol. 25, no. 6, pp. 599-616, June 2009.

[8] D. Parkhill, The Challenge of the Computer Utility. Reading,MA,USA: Addison-Wesley, 1966.

[9] M.A. Vouk, ''Cloud Computing-Issues, Research and Implementation,'' in Proc. 30th Int'l Conf. ITI, Dubrovnik, Croatia, 2008, pp. 31-40.

[10] J. Broberg, S. Venugopal, and R. Buyya, ''Market-Oriented Grids and Utility Computing: The State-of-the-Art and Future Directions,'' J. Grid Comput., vol. 3, no. 6, pp. 255-276, Sept. 2008.

[11] R.N.Calheiros,R. Ranjan,A.Beloglazov,C.A.F.DeRose, and R. Buyya, ''CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms,'' Softw., Pract. Exp. (SPE), vol. 41, no. 1, pp. 23-50, Jan. 2011.

[12] G. Reig, J. Alonso, and J. Guitart, ''Prediction of Job Resource Requirements for Deadline Schedulers to Manage High-Level SLAs on the Cloud,'' in Proc. 9th IEEE Int'l Symp. NCA, Cambridge, MA, USA, 2010, pp. 162-167.

[13] S.K. Garg, R. Buyya, and H.J. Siegel, ''Time and Cost Trade-Off Management for Scheduling Parallel Applications on UtilityGrids,'' Fut. Gener. Comput. Syst., vol. 26, no. 8, pp. 1344-1355, Oct. 2010.

[14] C.Vecchiola,X.C. Chu,M.Mattess, andR.Buyya, ''AnekaVIntegration of private and public clouds,'' in Cloud Computing Principles and Paradigms. Hoboken, NJ, USA: Wiley, 2011, pp. 251-274.

[15] SalesForce.com, Referenced on Dec 6 2010. [Online]. Available: http://www.salesforce.com

[16] Computer Associates Pty Ltd, Referenced on Dec 6 2010. [Online]. Available: http://www.ca.com

[17] Compiere ERP on Cloud, Referenced on Dec 6 2010. [Online]. Available: http://www.compiere.com/

[18] E.F. Yang, Y. Zhang, L. Wu, Y.L. Liu, and S.J. Liu, ''A Hybrid Approach to Placement of Tenants for Service-Based Multi- Tenant SaaS Application,'' in Proc. 6th IEEE Asia-Pac. Serv. Comput. Conf., Jeju Island, Korea, 2011, pp. 124-130.

[19] T. Gad, Why Traditional Enterprise Software Sales Fail, Referenced on March 6 2010. [Online]. Available: http://www.sandhill. com/opinion/editorial_print.php?id=307

[20] Y. Fu and A. Vahdat, SLA Based Distributed Resource Allocation for Streaming Hosting Systems, Referenced on 6th Dec. 2010. [Online]. Available: http://issg.cs.duke.edu

BIOGRAPHICAL NOTES

Vivek.V.P Is Pursuing Final Year B.Tech In Cse At Perunthalaivar Kamarajar Engineering And Technology, Karaikal.His Research Interest Incudes Cloud Computing And Networking

Santhosh.D Is Pursuing Final Year B.Tech In Cse At Perunthalaivar Kamarajar Engineering And Technology, Karaikal.His Research Interest Incudes Cloud Computing And Networking

J.Udaya Kumar Working As Professor At Department Of Computer Science And Engineering Perunthalaivar Kamarajar Engineering And Technology,Karaikal