Resource Allocation Avoiding SLA Violations in Cloud Framework for SaaS

Shantanu Sasane Abhilash Bari Kaustubh Memane Aniket Pathak Prof. A. A.Deshmukh

University of Pune University of Pune University of Pune University of Pune University of Pune
Smt. Kashibai Navale College of Engineering, Vadgaon Budruk, Pune, Maharashtra, India

Abstract

Cloud computing has been proved as a boon to distributed computing over a network, having ability to run a program on many connected computing at a same time. It is network based service provided by real server hardware, in fact served by virtual network. It is essential for using any service that makes uses of the Internet Network along with any non native hardware and software. Data center setup and maintenance is very expensive task thus many small scale businesses rely on hosting center to provide the cloud infrastructure to run their systems. In order to deliver hosted services fulfilling service level agreement (SLA), Software as a service provider companies have to satisfy minimum service level of customer that to in less cost. Optimal allocation is tedious task due to 1) heterogeneity in resource allocation 2) difficult to map customer request to infrastructure level parameter. 3) Managing dynamic change of customer. In this paper we introduce a framework called SLA-Based resource based allocation to reduce infrastructure cost and service level agreement violation offering control over all elements of the supplied by infrastructure provides.

General Terms

Software as a Service; Platform as a Service; Infrastructure as a Service; SLA violations.

Keywords

Cloud Computing; Resource Allocation; Service Level Agreement (SLA); Hosting center.

1. INTRODUCTION

E-commerce and digital services depend on data center and continue in increment of information technology. History of Software focuses mainly on shrink-wrapped software sales model. Customer need to purchase subscription- based license. It also included management of development and proposed and pays for non needed software or hardware cost. Although requirements are decreased, we need to pay for those unrequited services. Then there is introduction of cloud parameters which made available us utilizes like all other utilities you are charged based on what you consume. There was a transition from traditional software system to Software as a Service (SaaS).

Cloud service provides after their services in either (or both) of this two paradigms "Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) with the help of software as a service in application layer of cloud parameters. We provide a brief overview of these impediments.

Infrastructure as a Service: The provider is responsible for providing instances of machines as per the user specification. The resource available for each instance can be scaled on demand i.e. user can increase or decrease the number of CPUs, RAM, or storage through a web control panel or an Application Programming Interface (API).

Platform as a Service: This model locates above IaaS in the cloud framework and provides the user with an execution runtime, framework, operating system, database and web servers.

Software as a Service: SaaS provides any form of software or application as a service. A SaaS encourage a subscription model rather than a purchasing model. Customer simply subscribe to a number of users they need concurrently working on the software on the cloud.

A system model of SaaS layers for serving customers in cloud frameworks is shown in fig-1. A customer requests SaaS provider to satisfy requirements to utilize enterprise software services. It uses three layers, namely application layer, platform layers, and infrastructure layer. The application layer manages services offered to the customers by service provide. The platform layer includes scheduling and mapping policies for covering customer's quality of service (QoS). Parameters into infrastructure level parameter with allocation of virtual machine. The Infrastructure layer plays role of initiation and removal of VM's [1].

Service Level Agreement: A service Level Agreement (SLA) is a part of service contract where a service is defined formally. It is an agreement between customers and service from providers. It is legal binding map be formal or informal contract. IT records a common understanding about services, priorities, responsibilities, guarantees, and warrantees. The SLA may Specify the level of availability, serviceability, performance, operation, or other attributes of the service. [3]

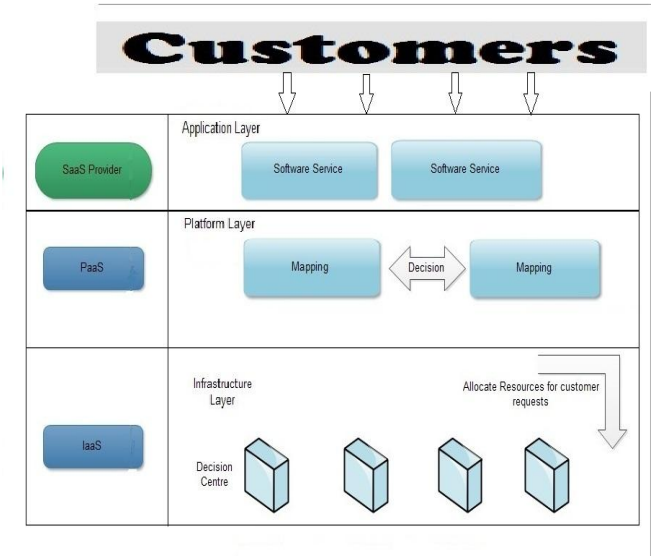

Fig 1: System Model of SaaS layer

2. Research Contribution

Previous works [1], [2], [3] have addressed the issue and SLA based Resource Allocation, minimizing cost of service provide and maximizing profit. We highlight the contribution of this paper in comparison to two closely related previous works, namely [1] and [2]: But still works related to SaaS provider with SLA violation avoidance are in their infancy. Dynamic allocation of resource, dynamic change and customer request, cost effective mapping and scheduling policies is the purpose of this paper.

The key contributions of this paper are as follows-

- Definition of SLA with respect QOS parameters.
- Mapping customer request to infrastructure level parameter.
- Ability to manage dynamic change of customers.
- It also focuses on handling heterogeneity of VM.
- Design a scheduling mechanism to maximize SaaS provider's profit by reduction in infrastructure cost.
- Managing to reduce incurred penalties for handling dynamic service demands.

The paper also focuses on arrival and proportion of upgrade request of customer and service initiation time and penalty rate according to SaaS provider.

3. Related Work

Research on resource allocation was started in 1980's. Most of these methods are non-pricing based [5]. Our works focus on profit maximization of SaaS providers.

I. Popovici et al. [6] focused on QoS parameters on the resource providers' side such as price and offered load, but user's side is not focused. We focus on QoS parameters from both the customers and SaaS provider's point of view. We also consider user driven scenarios.

Lee et al. [7] considered profit driven service, request scheduling for workflow. Here our work focuses on challenges of dynamic changes in customer request to gain profit.

Patel et al. [2] investigated data content, resource heterogeneity, generalized network flow-based resource allocation for hosting enters our work in addition handle resource heterogeneity and SLA violations.

Lenlin et al. [1] contributed in ensuring mapping customer request to infrastructure level parameter. Our work suggest WIN-WIN situation between customer and service provider using cloud mapping.

4. System Model

Our paper proposes that customer requires for enterprise services from a SaaS provider by agreeing to SLA parameters and mentioning QOS parameters. The Saas provider can use their provider [ref-SLACE]. The SaaS provider objective is to schedule job as per SLA given at the time of registration. It should be capable to change its state once SaaS changes its SLA. System should provide separate process to schedule which jobs should be placed into execution. System should be scalable, which means that its performance should not degrade with the addition of nodes and jobs. It should be configuration, and allow for various scheduling polices that can be modified to incorporate QOS parameters, system handles dynamic VM switching as per SLA and support dynamic load scheduling. The main task is of minimizing the number of SLA violation in a dynamic resource sharing. To satisfy customer's requests in order to enlarge market store and minimize cost, the following question have to be addressed.

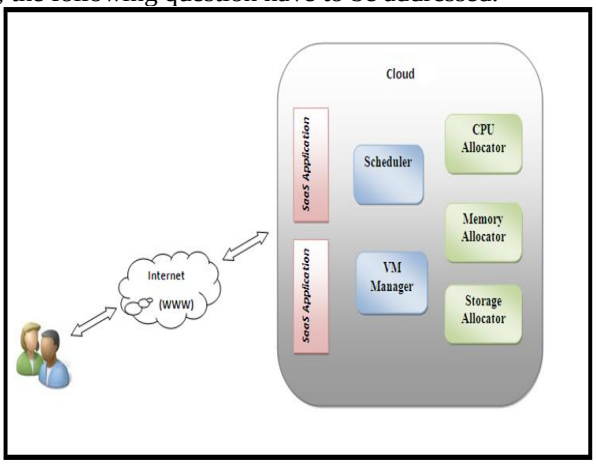

Fig 2: Cloud Provisioning and Deployment Model

We consider the customer's request for the enterprise software services from a SaaS provider by agreeing to the predefined SLA clauses and submitting their QOS parameters. Customer can dynamically change their requirement and usage of the hosted software services. The SaaS provider can use their own infrastructure or outsourced resources from public IaaS providers. A scheduling mechanism to maximize a SaaS provider's by reducing the infrastructure cost and minimizing SLA violation is designed. The scheduling mechanism determines where and which type of VM has to be initiated by incorporating the heterogeneity of VM's in terms of their price, dynamic service initiation time and data transfer time. It also manages to reduce penalties for handling dynamic service demands when customer is sharing resource.

The main purpose of the scheduler is to maximize use utility. The scheduler will aim to optimize resource utilization within user imposed constraints. Thus, user satisfaction is the primary concern as opposed to maximizing CPU utilization. The scheduler will allocate job based in the job parameters, which are job specification submitted by the user with the job.

4.1 Scheduling Strategies

Figure2 shows cloud provisioning and deployment model and use case scenario of combining three different layers, namely IaaS, PaaS, and SaaS [1]. The customer places their service deployment request to the service portal [SaaS Application]. Then this request is passing to CPU allocator. Here memory allocation and storage allocation taken into consideration. If the request is valid, it is then forwarded to the scheduler. The scheduler selects the appropriated VM through VM's manager in PaaS layer for deploying the requested service. Balancing of service provisioning among the running VM's is balanced by load balancer. The VM manager manages the VM's on the virtualization layer which interacts with physical resource. There is possibility of provisioning at the single layer alone. But our proposed scheduling consider at this layers are not trivial considering their different requirement and constraint. At IaaS layer, VM's are to be deployed with respect to agreed SLA's with the customer. Deploying application at SaaS layer is very challenging as each application leads to fulfill the SLA terms.

4.2 Properties defined in SLA

1. SLA Request Type (slaRequest): It defines the type of request stated by customer. Whether it is 'Initial Rent or Promote Services.' 'Initial Rent' refers to customer who is renting a new service. 'Promote Services' refers 'add account' and 'Up gradation of product'.
2. SLA Product Type (slaProduct): It is a type a software product that is offered to customer. For example, SaaS provider offers different types of product viz. basic, professional and enterprise. Each product type supports various types of accounts.
3. SLA Account Type (slaAccount): It refers to maximum number of accounts a customer can create. For example Group account, Team Account, Department Account, which allows customer to create up to n, 3n and 7n number of accounts respectively.
4. SLA Contract Length (slaLength): It is the actual number at account that a customer want to create.
5. SLA Account Count (slaAccNo): it is an actual number of accounts that a customer want to create.

4.3 Mapping Strategy

Table 1. Mapping Strategy

VM Type	Product Type	Account Type	Max Account #	Min Account #
Minor	Basic	Group	n	1
Moderate	Basic, Professional	Team	3n	n+1
Extended	Basic, Professional, Enterprise	Department	7n	3n+1

Mapping strategy defines the way of mapping of customer Quality of Services (QoS) requirement to the resource. Our paper proposes Infrastructure Layer focusing on the VM level, but not the host level. For example, if a service provider is restricted to create maximum n number of record, the mapping of product type and number of account is shown in Table 1.

4.4 Mathematical Model

Set Theory

Set Theory Analysis

Let 'S' be the "SLA-based Resource Allocation in cloud"

S= {…….}

Set S is divided into 6 modules

S= {S1, S2, S3, S4, S5, S6}

S1= GUI Handlers (GH)

S2= Configuration Manager (CM)

S3= VM Manager (VMM)

S4= SLA Manager (SLAM)

S5= Policy Manager (PM)

S6= Database Manager (DM)

Identify the inputs.
Inputs = {X1, X2, X3, ……..Xn}
X1= SLA
X2= Account addition
X3= Update SLA

Identify the output as O.
Outputs = {Y1, Y2, Y3, ……..Yn}
Y1= VM Allocation
Y2= Resource Allocation

Problem Description: Let S be a system which do SLA-based Resource Allocation in cloud; such that S = {S1, S2, S3, S4, S5, S6} where S1 represents GUI Handlers (GH) Module; S2 represents Configuration Manager (CM) Module; S3 represents VM Manager (VMM) Module; S4 represents SLA Manager (SLAM) Module; S5 represents Policy Manager (PM) Module; S6 represents Database Manager (DM) Module.

4.5 UML Design Observations

S holds list of modules in the system
Activities:

4.5.1 Activity I: SaaS Provider SLA Creation

Let S1 be a set of SaaS provider parameters for SLA creation.
S1= {user_id, sla_id, sla_values}
Where,
user_id: user's id
sla_id: defined SLA id
sla_values: SLA values

Table 2. SaaS Provider SLA Creation

Condition/Parameter	Operation/Function
If no change in SLA data	Discard value
Else.	f1:Proceed()

UML Design observations:
If user's SLA info is not updated or changes then please discard the value Else read the SLA value any send data to database.

4.5.2 Activity II: Allow number of accounts as per SLA defined

Lets S2 be a set of accounts policy details parameters:
S2= {user_id, sla_id, sla_type, account_count}
Where,
user_id: user's ID
sla_id: SLA ID
sla_type: Type of SLA like basic, enterprise, etc…
account_count: No of account allowed in SLA type

Table 3. Allow number of accounts as per SLA defined

Condition/Parameters	Operation/Function
account_count	f1:searchSLAPolicy();
If (user is allowed to add account)	f2:checkSLAPolicy() ;
Add Account	f3:CreateAccount();
Else, discard account add info	

UML Design observations:
Search in the user's SLA policy that the user's can create account or not. If the user's SLA policy allow then add account. Else don't add account and show error messages.

4.5.3 Activity III: Allocation of VM as per SLA

Let S3 be a set of user's VM allocation parameters:
S3= {user_id, sla_id, sla_type, vm}
Where,
user_id: user's ID
sla_id: SLA ID
sla_type: Type of SLA like basic, enterprise, etc…
vm = Virtual machine to be added on host

Table 4. Allocation of VM as per SLA

Condition/Parameters	Operation/Function
Vm_policy	f1:searchVMPolicy();
If (user is allowed to add VM on host)	f2:checkVMPolicy();
Allocated and add new VM as per available space on host	f3:AllocateVM();
Else, discard users VM request	F4: discard()

UML Design Observations:
Search in the user's SLA policy that the user's VM should be added. If the user's policy allow then add VM. Else don't add VM and throw error.

4.5.4 Activity I: Show Result graphs

Let S4 be a set of parameters required for graph generation.
S4= {user_id, sla_id, reports}
Where,
user_id: user's ID
sla_id: SLA ID

reports= Reports to be Displayed

UML Design Observations:
Here dynamically graphs will be shown to user.

5. Implementation Issues and Performance Evaluation

The proposed scheduling is implemented as a new scheduling policy in the CloudSim simulation tool for the purpose of evaluation. It is a framework for modeling and simulation of cloud computing infrastructure and services in repeatable and controllable environment free of cost and tunes the performance bottleneck before deploying on real clouds. At the provider side, simulation environments allow evaluation of different kinds of resources leasing schematic under varying load and pricing distribution, studies could did the provider in optimizing the resource access cost with focus on improving profit. It helps in finding and removal of errors before implementing in real time.

6. Conclusion and Future Work

Though there is revolution in cloud computing, there are still open research challenges in maintaining application's required quality of service and achieving resource efficiency. This paper focused on scheduling of resource utilization and customer requests for Software as Service providers with cost minimization and profit maximization. Simultaneously resource level heterogeneity and dynamic changes of customer requests is addressed. Paper also focused on mapping customer requirement to infrastructure level parameters.
In future work, we will investigate scheduling and application deployment in cloud considering increased in efficiency in allocation and utilization of resources. Furthermore, we would like to add more services and various strategies that will maximize the profit of service providers. Moreover, we will strictly look into penalty limitation considering system failures.

7. Acknowledgements

We thank anonymous reviewers and staff of Smt. Kashibai Navale College of Engineering, Pune who helped us to improve the quality of this paper.

8. References

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

[2] Kimish Patel and Murli Annavaram, "NFRA:Generalized Network Flow-Based Resource Allocation for Hosting Centers" in IEEE TRANSACTIONS ON COMPUTERS, VOL 62, NO. 9, SEPTEMBER 2013.

[3] Mojun Su, Shenglin, Yang, Hao Li and Joan Lu, "A Service Level Agreement for the Resource Transaction Risk Based on Cloud Bank Model" in Proceedings of International Conference on Cloud Computing and Service Computing, 2012.

[4] Vincent C. Emeakaroha, Ivona Brandic, Michael Maurer and Ivan Berskovic, "SLA-Aware Application Deployment and Resource Allocation in Clouds", in proceedings of 35th IEEE Annual Computer Software and Applications Conference Workshops, 2011.

[5] J. Broberg, S. Venugopal, and R Buyya, Market-oriented Grids and Utility Computing: The state-of-the-art and future directions, Journal of Grid Computing, 3(6), (pp.255-276).

[6] I. Popovici, and J. Wiles, "Proitable services in an uncertain world". In Proceeding of the18th Conference on Supercomputing (SC 2005), Seattle, WA.

[7] 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.