Characterizing Web Application Performance for Maximizing Service Providers' Profits in Clouds

Xi Chen, Haopeng Chen, Qing Zheng, Wenting Wang, Guodong Liu
REINS Group, School of Software
Shanghai Jiao Tong University
Shanghai, P.R.China
{chenxi_sjtu, chen-hp, mark, wwtvanessa, mealsd}@sjtu.edu.cn

***Abstract*—A number of challenges in implementing cloud technique related to further improving Web application performance and decreasing the cost. In order to achieve high profits, cloud-based web application providers must carefully balance cloud resources and dynamic workloads. However, this task is usually difficulty because of the complex nature of most web application. In this paper, we presented a predictive performance model to analyze such applications and to determine when and how much resource to allocate to each tier of an application. In addition, we proposed a new profit model to describe revenues specified by the Service Level Agreement (SLA) and costs generated by leased resources. Furthermore, we employed profit driven model to guide our resource management algorithms to maximize the profits earned to the service providers. We also designed and implemented a simulation experiment on CloudSim that adopts our proposed methodology. Experimental results indicated that our model faithfully captures the performance and resources are allocated properly in response to the changing workload, thus the goal of maximizing the profit has been achieved.**

Keywords-cloud computing; resource management; multi-tier web application performance modeling

I. INTRODUCTION

As a new computational model, cloud computing [1] [2] is a large-scale distributed computing paradigm that is driven by economies of scaling, in which a pool of virtualized, dynamically-scalable, and managed computing resource, storage, platforms, and services are delivered as "pay-per-use" manner to external customers over the Internet. Many IT giants such as IBM, Sun, Amazon, Google, and Microsoft offer computational and storage resource rental services to consumers, in which application, data, and IT resources are provided as service over the Internet rather than products locally installed on their own machines. Therefore, service providers overcome server's limitation: non-scalability and poor high-availability. Furthermore, service providers can also optimize global resources with minimum infrastructure and maximum resource utilization, in order to maximize their profits.

The advantage and success of cloud computing are promoting many service providers and enterprises to deliver their existing applications to cloud. According to a survey of 1,780 data center managers in 26 countries worldwide proposed by Symantec [3], increasing applications and more demanding SLA are combing to make data center continue to become more complex and harder to manage. Over 82% of respondents indicated that reducing the cost of managing data centers and Web applications is one of the most critical objectives for the coming year. Over 36% of respondents indicated that the complexity and the large number of applications are huge problems that they faced.

Consider a simple motivation scenario in which a mid-sized company runs an application service, e.g. a typical ecommerce application consists of three tiers---a front-end Web tier that is responsible for HTTP processing; a middle tier Java enterprise server that implements core application functionality, and there may be also enterprise workflow in this tier; and a back-end database that stores product catalogs and user order. Rather than purchasing its own infrastructure to run its services, the company lease resources from a cloud hosting provider to reduce cost and operating fee. The service providers and their customers often negotiate utility based Service Level Agreements (SLAs) to determine cost and penalties based on achieved performance level (see Figure 1) -- a service provider, with a controller and a monitor, running web services on cloud resources.

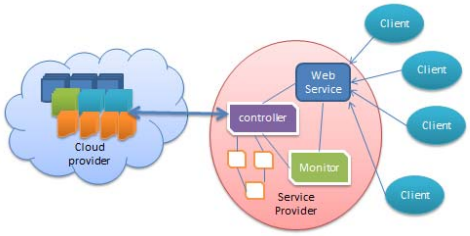


Figure 1. Web application run on Cloud resource

Despite the significant benefit, migrating enterprise Web application to cloud computing is a major problem [4]. On one hand, little is known regarding to the performance of each tier of the application. Estimating the application response time allows dynamically allocating the necessary resource and then increasing overall performance. On the other hand, the resources of Cloud application need to scale up and down quickly according to workload in order to save money, which acquiring and releasing resources at fine grain and in a short period of time.

This paper addressed the problem of: 1) modeling the performance of multi-tier Web application in clouds; 2)

978-1-4577-1637-9/11/$26.00 ©2011 IEEE

scaling up and down quickly according to profit-driven resource management policy.

Performance modeling: The challenge stems from the fact that an enterprise Web application consists of a large number of components, and multi-tier architecture is widely used for constructing Internet application and service. Compared with existed multi-tier application modeling method [5, 6, 7], our model is more suitable for cloud environment for the following reasons: (i) *generality*, which enables multiple services run on a virtual machine (VM) to support complex application environment, (ii) *modeling on specific resources,* e.g. CPU, which enables application bottlenecks to be identified for the basic needs, (iii) *support heterogeneous environment*, which enables our model to support multiple VM environment, trend of future cloud, to support fine-grained resource control in order to look at the role of enterprise workflow in this context .

Profit-driven controlling: Many existing approaches have been proposed on exploring profit-driven resource location to handle this problem, such as FirstPrice [11], the proportional-share [12], while most of them are applicable to scheduling batch jobs with a fixed number of resources. Our approach is practical for real Web service provider on clouds and can be applied to a wide range of applications. Present solutions [15] [18] to resource management have mainly focused on global optimization with respect to application performance. Regarding to migrating applications to cloud, one common disadvantage of these approaches is that it is much harder to scaling down resources that scaling up according to the performance of an application, due to ignoring the profit factor. Furthermore, it is in the service provider's interest to minimize the cost of using the resources offered by the cloud infrastructure vendor [10] and maximize the revenue (specifically, service profit) generated through serving consumers' application. Our proposed mechanism solves the problem how many virtual machines (VMs) a service provider should request from cloud (e.g. Amazon) in response to the observed applications' performance and profit fluctuations.

The rest of this paper is organized as follows. Section II presents an overview of the proposed system. Section III presents our performance model and provisioning policy. We present our experiment results in Section IV, related work in Section V, and conclusions in Section VI.

II. ARCHITECTURE

In this section, the architecture of the proposed system is presented. Specifically, it is a web application hosting provider with an unlimited resource rent from Cloud service provider.

Our management system architecture, illustrated in Figure 2, is composed of the following components:

- *Client Component (C)* is a logical component that measure workloads intensity of system, e.g. arrival rate. Ideal implication of resource management is that full spectrum of workload for which we predict web application performance may be supportive for resource scheduling.

-*Transaction Handler (TH)* is a front-end server that receives client requests to back-end servers, and then provider details of transaction type, number of transaction, service type and numbers of service to registry center.

- *Service Handler (SH)* is in charge of sending service to corresponding VMs.

- *Registry Center (RC)* is responsible for collecting detail information of the whole system, e.g. transaction, service and servers.

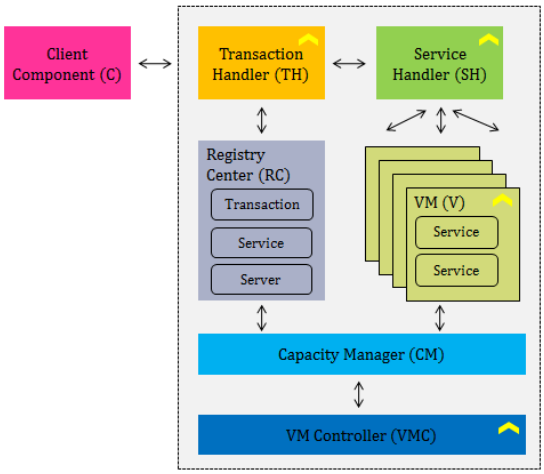


Figure 2. System Architecture

- *Capacity Manager (CM)* is responsible for adapting the whole architecture to the needs of web applications, which means some key actions like reconfiguration the number of VMs and communicate with registry center to turn ON or OFF those VMs.

- *VMs Controller (VMC)* is a component that allows service providers to rent an unlimited amount of external resources in order to run any number of web services on top of them.

III. PROFIT-DRIVEN RESOURCE MANAGEEMENT

We have designed a profit-driven resource management in order to address the problem of dynamically managing the cloud resources in cost-effective way, i.e. maximizing their utilization and increasing their profit. The main operational aspects of our system involve three phrases: First, the volume of demand in applications fluctuates on several times scales. Therefore, performance model must be effectively adjustable to workloads in order to support next step scheduling. Second, extending single-tier model to multi-tier one is non-trivial, since each tier has different features. Our provisioning mechanism can detect which tier is responsible for this increase and decrease of the number of VMs. Third, dispatching decision will be made according to the service provider's interest to maximize revenue generated through providing consumer's application.

3.1 Problem statement

For the purpose of maximizing the profit, we define the following variables:

Revenue & Cost: Two key factors in our approach are the cloud revenue *(R)* and cloud cost *(C)*. The cloud revenue is estimated using the utility function, which is based on Service Level Agreements (SLAs) to determine the cost and

penalties by achieved performance level. [11] Monotonic non-increasing utility functions are quite realistic since the better the achieved performance by end users, the higher are the revenues gained per request by the Service Provider. Modern workloads are session-based, therefore service types strongly influence system resource demands, e.g. "browser", add-to-shopping basket", "purchase". Let $S = \{S_1, S_2, \dots, S_i, \dots, S_n\}$ denotes different types of services. Let $N_{S_i} = \{ n_1, \dots, n_i, \dots, n_n\}$ denotes the number of S_i and $R_{s_i} = \{R_1, \dots, R_i, \dots, R_n\}$ denotes the revenue that belongs to different services. In some cases, when the violation of SLA occurred and then SaaS providers have to pay for the penalties, but in some cases if the profit is relatively high, the capacity manager may not turn ON or OFF VMs since we concentrate on profit-driven policies. From the utility law, one can easily obtain Eq. (1) for the total revenue:

$$R = \sum_{s_i \in S} n_i * R_{s_i} \quad (1)$$

The cloud cost is the cost of renting the cloud resources from external cloud providers. There are m classes of VM available among the set of $R_{s_i} = \{R_1, \dots, R_i, \dots, R_n\}$, in which $M_{r_i} = \{m_1, m_2, \dots, m_i, \dots, m_n\}$ denote the number of resource R of type i and $P_{r_j} = \{P_{r1}, \dots, P_{ri}, \dots, P_{rn}\}$ denotes the prices of different resources. The cost is calculated by multiplying the number of VMs running in the cloud provider during that period of time with its corresponding price (e.g. Price of VM per hour of a small instance in Amazon EC2 is $0.085 per hour).

$$C = \sum_{r_j \in r} m_j * P_{r_j} \quad (2)$$

In this paper, we only focus on traditional cloud providers VM offering type that is specified as a fixed price. We will discuss our scheduling policy to this problem under both fixed and variable pricing schemes, e.g. in spot market in the future.

Profit: Given the system model above we formulate the cost optimization problem be the total profit of the whole application is $\text{Profit} = R - C$ (3)

3.2 Performance model

Performance models are powerful tools when it comes to addressing resource location task. This model is useful as they provide insight into application's behavior, enabling service providers to allocate their computing resource with flexibility, based on workloads and business needs, thereby improving overall efficiency and application profits. This section provides a baseline performance model for Web application, associated by several enhancements to capture certain application idiosyncrasies. Modern Web applications are designed using multiple tiers. Each tier receives partially processed requests from the previous tier and feeds these requests into the next tier. For example, an online can be designed using three tiers: front-end Web server is responsible for HTTP processing, middle-tier Java application server implements the application logic, in which a request-reply may consist of several services at this tier, a backend database that stores catalogs and user orders.

A. Basic model

To capture each tier behavior under time scale we observe a number of different sessions over monitoring windows of fixed length of Δt. Session mix, system utilization and system performance are recorded at the end of each monitoring window.

Assuming that there are totally N service types processed by the system, we use the following notations:

- Δt is the length of the monitoring window;
- N_i is the number of services of type *i*;
- *W* is workload characteristics, we use λ_i to represents the main characteristic, which is the arrival rate of requests of type i service.
- $res_{s_{j,i}}$ is the average service time of services of j-th type at the i-tier of the system, where $1 \le i \le N$.

To work out how many servers to allocate to each tier and each application, we construct an analytical model of an internet application. All models have the same general high-level form:

$$P = F(w, \delta) \quad (4)$$

where P is the scalar summary of application performance, F specifies the functional form of the model, including the workloads w and system parameters δ. To avoid continued violation of the profits during VM scale-up (down), it would be better to predict performance violations rather than waiting until the requirement is violated.

B. Performance Modeling and Prediction

The capacity manager is required in the framework as for adjusting allocation of VMs for future demand. More specifically, we want to predict the demand curves in the time period $[t_i, t_i + \Delta t]$ for SaaS provider, where t_i denotes the current time and Δt denotes the prediction period. Workload tracing seems to be unpredictable and random, but it has structure to be exploited by mapping it to an algorithm. We assumed that the change of users' request is relatively periodically during one certain time, e.g. one day or one month, and it has specific amount of request during certain time, e.g. off-peak, sub-peak, and peak time during a single period. Even if service types or their resources demands change completely, it takes only a relatively short period to gather sufficient data to calibrate completely new performance models. Hence we use a time-series method to forecast the future workload for each k class of services of $\lambda_{i,k}$ for $t_i \le t \le t_i + \Delta t$ using past workload history. Future demand in general has been widely studied in both economic and computer system, similar problem also has been studied in [13].

A simple auto-regressive (AR) mode was adopted to predict the expected demand curve. In our experiments, we used linear AR model provided by MATLAB to obtain $\lambda_{i,k+1}$. Its value is estimated by using the historical values $\lambda_{i-1,k}, \dots, \lambda_{i,k-j}$ as:

$$\lambda_{i,k+1} = \sum_{j=1}^{k} \eta_j . \lambda_{i,k-j} + \epsilon_t \quad (5)$$

where η_1, η_2, …, η_j constitutes a set of parameters for historical values, and ϵ_t is white noise uncorrelated. All of the above parameters can be computed from historical data. AR model is appropriate for this problem for the following reasons: first, AR model is light weighted than machine learning based model, such as neural networks; second, AR model is capable of capturing short term trends, which is more suitable for predictive resource allocation, e.g. we observe a number of different sessions over monitoring windows of fixed length of interval 5mins in one day.

C. *Extended model*

We have implicitly assumed that an application's set of service types are fixed in an interval Δt and the relationship between services types and resource demands is stable. To address the response time of each tier, we first note that the term $r = \frac{1}{\mu - \lambda}$ represents the residence time of in *M/M/1* queuing model, where λ is the arrival rate and μ is the service capacity. It is often practically difficult to obtain the information of the client session in real applications and the systematic parameters required by closed models. Compared to closed models, our model provides more thorough empirical validation [12]. The resource pool is modeled by a queuing network composed of a set of multi-class single-VMs queue as shown in Fig. 3.

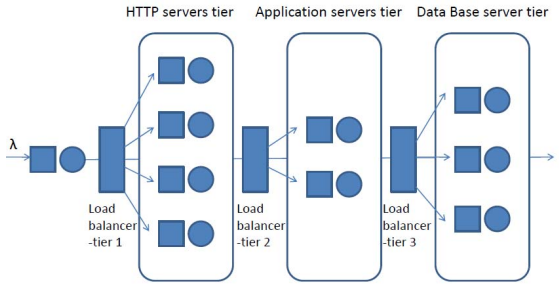


Figure 3. Performance model

The application environment under study is a resource pool consisting of m class of heterogeneous VM. We extend this residence time to Equation 4 by adding terms representing system characteristics as follow:

$$res_{S_{j,i}} = \frac{1}{U_j * C_i * \phi_{j,i}/\beta_i - \sum_j \lambda_{j,i}/n_i} \quad (6)$$

A user request is supported by multi-tier web applications. We apply our control schemes and solve the profit optimization for independent tiers. When confronted with different kinds of service, each VM distributes its own computing resource in accordance with the ratio of service arrival rate β_i. Let $C_i = \left(c_i^{cpu}, c_i^{ram}, c_i^{bandwidth} \right)^T$ specifies the average capacity of a single VM of single tier i to serve S_j. To be specific, the service rate for service S_j at single tier i, includes the CPU, RAM, and bandwidth. By this definition, the estimated CPU service rate of a service managed by tier i is given by:

$$c_i^{cpu} = \frac{capcity * cores(p)}{ls_i} \quad (7)$$

where ls_i is the total number of instructions that the service will need to execute on a processor, and $cores(p)$ is the number of CPUs in one VM. In reality, CPU capacity is often priority for performance modeling. First, assume that there are sufficient resources to serve the service requests in system, CPU would be confronted with bottleneck mainly affecting the whole performance. In addition, there is no close relation between the capacity of RAM or bandwidth and the type of services in reality. Therefore, c_i^{ram} and $c_i^{bandwidth}$ are defined by the configuration of each VM in real system.

Let $U_i = \left(u_i^{cpu}, u_i^{ram}, u_i^{network} \right)$ denotes the average utilization of specific resource at single tier i. β_i is the ratio of $\lambda_{j,i}$ of different services, and accordingly, we unite different service into one service coming with the arrival rate of $\sum_j \lambda_{j,i}/n_i$ and restore them through different service capacity. Let $\phi_{j,i}$ be the scheduling parameter for service S_j of tier i. To solve for $\phi_{j,i}$, one can choose regression method. An approximated $res'_{s_{j,i}}$ for the next interval Δt can be calculated through Equ. (6)

Finding the ideal regression method for this problem is over the scope of this paper. In our system, the object of regression method is to minimize the variance: $\sum_k (res'_{s_{j,i}} - res_{s_{j,i}})_k^2$ where k is the index of monitoring window over time. Then, this set of $\phi_{j,i}$ can be solved in order to estimate the parameter of different service in different tiers. After this step, the approximated response time of next monitoring window is computed from the predicted $\lambda_{i,k+1}$ and estimated $\phi_{j,i,k+1}$. In the next section, we explore profit-driven scheduling policy with the result of workload and performance of our modeling solution.

3.3 Profit-driven scheduling policy

The proposed policy takes into account not only the profit achievable from the current service, but also the profit from other services being processed in the system. Specifically, it can bring additional service profits when new services are assigned into the service instance yields. So scheduling event takes place either at the time that a new service request arrives or at the requests is completed [10].

Two key factors in our policy are the cloud revenue *(R)* and the cloud cost *(C)* as we discussed in section 3.1. In practice, physical resources of clouds are limited and a performance bottleneck will eventually developed. Therefore, regardless the nature of these bottlenecks, the response time reaches a minimum for a specific number of VMs. By meaning of the predicted response time estimated in the system, capacity manager represented in section 2 increases the services' profit via deploying VMs to the specific tier we choose of the application, or removing additional VMs, so how many VMs to increase or decrease is the essential concern in this policy.

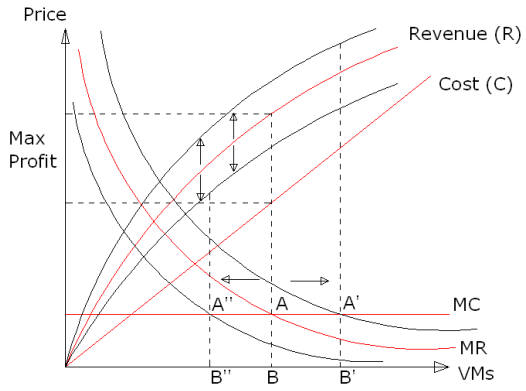


Figure 4. Profit-driven scheduling policy

Algorithm 1 NumberOf VMsNeeded
Input: R () : User revenue function
C ( ): cost of VMs
setUp is the initial number of VMs to be set up in system;
step =1 in order to start up the algorithm
VMNow[] and VMBefore [] denote the number of VM of each tier for present and last VM changing time;
resN [j][i] and resB [j][i] denote and response time of tier i of S_j for present and last VM changing time
flag is true for initialization
Output: Number of VMs needed

1: if (flag == true) then
2: VMNow[] = setup
3: VMBefore[] = setup – step
4: flag = false
5: else
6: VMIncrease = 1
7: dRev = $\sum_j$ Rev $\sum_i$ resN[j][i]-$\sum_j$ Rev $\sum_i$ resB[j][i]
8: dVM = $\sum_i$ VMNow[i]-$\sum_i$ VMBefore[i]
9: MR = dRev/dVM
10: MC = dC/dVM
11: While (|MR-MC|> ε)
12: tier = tierSelected()
13: VMBefore = VMNow
14: if MR >MC then
15: VMNow[tier]=VMNow[tier]+VMIncrease
16: else if MR< MC then
17: VMNow[tier]=VMNow[tier] - VMIncrease
18: end if
19: dRev = $\sum_j$ Rev $\sum_i$ resN[j][i]-$\sum_j$ Rev $\sum_i$ resB[j][i]
20: dVM= $\sum_i$ VMNow[i]-$\sum_i$ VMBefore[i]
21: MR = dRev/dVM
22: MC = dC/dVM
23: end while
24: end if
25: return $\sum_i$ VMNow[i]

Figure 5. Algorithm 1: numberOfVMs

In Figure 4, Marginal Revenue (MR) and the Marginal Cost (MC) are two critical factors we need to account for in this problem: $\mathrm{MR} = \frac{\mathrm{dRev}}{\mathrm{dVM}}$, $\mathrm{MC} = \frac{\mathrm{dC}}{\mathrm{dVM}}$

The maximization of profit is when MR=MC at point A. The total profit is $\mathrm{P} = \mathrm{R} - \mathrm{C}$.

Accordingly we should provide number of *B* VMs at this point. When many new requests arrive, total revenue increased because of increased number of services, each VM costs more than the original offered *(MR > MC)* and the maximization point move to A', until the number of VMs reaches B'.

Algorithm 1 computed the number of VMs, depending on the difference between *MR* and *MC*, we increased or decreased the number of VMs to be set in virtual environment by finding this point to maximize the profits. MR is estimated by the user's utility function and the response time. MC is estimated by SaaS providers' resource scheduling policy according to cloud administration [16]. Here, number of VM does not reach its maximization point B'or B'' immediately, since the revenue of services will change because of the alteration of performance in turn. So gradual approach was used in line 11 to 23, and MR and MC are re-estimated every time the number of VMs has been changed.

In Algorithm 2 shows how we decided on which tier to increase/decrease. To be specific, this method focuses on enabling multi-tier Web application to acquire maximum change of response time through choosing the largest ratio of response time and number of VMs among different tiers, as shown in line 5, while maximizing the overall resource utilization. We linearly changed the number of VMs by 1 in one round in Algorithm 1, but for large infrastructure, it is able to sustain a high number of VMs, thus the VM_increase should be set according to workloads and performance.

Algorithm 2 tierSelected
Input: VMNow[] and VMBefore [] denote the number of VM of each tier for present and last VM changing time;
resN [j][i] and resB [j][i] denote and response time of tier i of S_j for present and last VM changing time
Output: which tier is responsible for maximizing profits

1: maxI=0
2: for i from 1 to n
3: $a_i = \frac{\sum_j \mathrm{resN[j][i]} - \sum_j \mathrm{resB[j][i]}}{\mathrm{VMNow[i]} - \mathrm{VMBefore[i]}}$
4: if $a_i > max$
5: maxI = i
6: max=a_i
7: return (maxI)

Figure 6. Algorithim 2: tierSelected

In brief, we maximized the SaaS providers' profits by increasing or decreasing VMs to the most obvious tier step linearly by single step. Algorithms 1 provides the direction of remove or add VMs, and Algorithm 2 gives the answer of

which tier to change. As for implementation, the capacity manager *(CM)* is responsible for controlling the VM allocation for the next period using the theorem discussed in Section 3.2.

IV. SIMULATION AND EXPERIMENT

To evaluate the effectiveness of the proposed approach, we have designed and implemented a prototype of our framework using CloudSim [8], a toolkit for modeling and simulation of Cloud computing infrastructures and services. CloudSim supports modeling of on-demand virtualization enabled resource and application environment. We then extended the framework in order to enable dynamic profit driven scheduling policy in virtual datacenter as the original core framework does not provide this functionality. In addition, we have incorporated the ability to monitor SLA violations and process QoS-driven scheduling policy as a contrast. Finally, we simulated dynamic workloads that act as a typical load of 1998 world cup web site of a period [9].

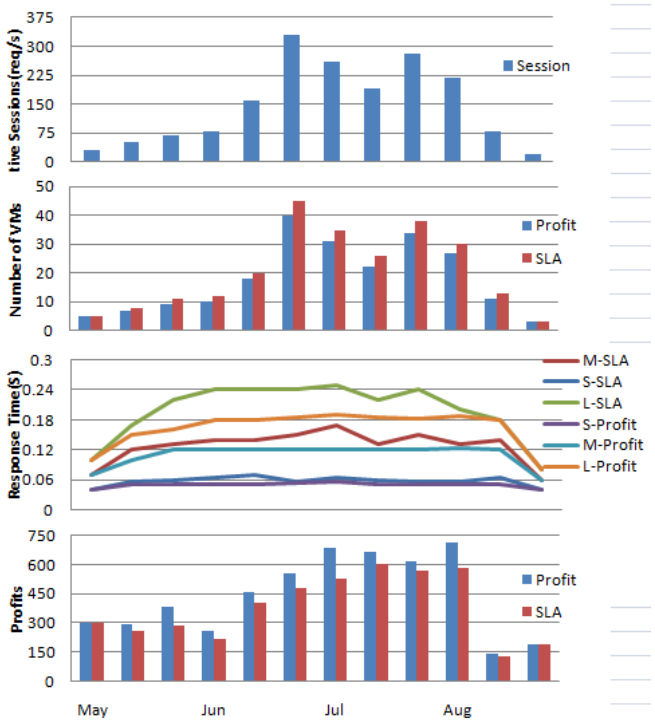
The simulation data consisted of 20 VMs on homogenous physical nodes on the first place. In the simulation, each node is modeled to have one CPU core with performance of 3000 MIPS, 8 Gb of RAM and I TB storage, and each VMs has 512 Mb of RAM, the same as the physical machine in our lab for the purpose of processing physical environments in the future. The execution of the simulation consisted of three phases: The first phase is to produce workloads that act as a typical workload of web sites, so we reproduce the load of 1998 world cup web site of one period in our system. The second one consisted of a set of services, the number of which are randomly produced as real situation and attributes of which are defined. Finally, our capacity manager as a control part in datacenter collects service information needed to execute profit-driven scheduling policy.

TABLE I. CLOUDSIM SIMULATION PARAMETER SETTINGS

Simulation Parameters	Value
Initial number of VMs	5
VM cost	0.3$
VM per Memory	0.05$
VM per Storage	0.1$
VM per Bandwidth	0.1$
Small service revenue function	$R_{S_1} = 2/\sqrt{t}$
Middle service revenue function	$R_{S_2} = 3/\sqrt{t}$
Large service revenue function	$R_{S_1} = 5/\sqrt{t}$

Table 1 shows all parameters of our simulation. There are three types of services, small, middle and large. The large service gained more profits than the middle and the small one. In our simulation we analyzed how profit-driven is capable of maximizing profits by processing QoS-driven scheduling as comparison. To be specific, we monitor the response time of each service and signal a VM increase if the average response time of one kind of service exceed 0.05s, 0.12s, and 0.18s respectively. Like many QoS-driven policy, we increased the number of VMs when the violation was occurred in system, but as we discussed before, it is hard to decrease the number of VMs dynamically in practice in this method, so we use static scheduling to implement.

We repeated these two experiments by using our profit-driven provisioning technique and QoS-driven. Our results are shown in Figure 8. As shown in Figure 8 (b), to ensure the QoS requirements, the datacenter often need more VMs than profit-driven policy. Figure 8 (c) shows the response time of different services in both QoS-driven and profit-driven provisioning policy. As shown in the figure, the performance of small services is almost the same. The larger the services are, the more loss of performance would generate, which is due to the fact that large services are more sensitive to the lack of resource. Figure 8(d) shows profit-driven are more profitable than QoS-driven policy by using less VMs, and we believe this result is also profitable for Cloud infrastructure provider.



Firgue 7. Two scheduler evaluation: QoS-driven scheduling policy and Profit-driven scheduling policy. From the top to bottom: input workload (a), number of VMs needed in two policies (b), response time of different services (c), profit obtained in two policies (d).

TABLE II. OVERALL COMPARATIVE RESULTS

Algorithm	Avg. response time (%)	Total Revenue	Total cost	Total profit
Profit	84.1%	15650.0	10396.6	5253.4(15.37%)
SLA	100%	17200.7	12663.3	4537.4

The entire results obtained from our simulation are summarized in Table 2. Our results demonstrated that Service provider and Cloud resource provider can both be profitable through profit-driven provisioning technique by less number of VMs and 15.37% profit increased. Also,

because that our provisioning technique is able to take profits imposed by revenue and cost as priority, it can also maintain a relatively acceptable response time targets by adding more VMs when profits do not reach the max at this workload.

V. RELATED WORK

Dynamic resource management for QoS evaluation and SLAs revenues in virtual infrastructure is a critical problem in Clouds. There is a wide-range of research around resource allocating policy proposed for either of the two purposes in virtual infrastructure environment. Among these techniques, queuing theory is a widely used methodology for modeling system behavior and providing allocating questions [19, 20, 21]. For example, [19] had proposed a HYDRA historical and layer queuing technique to meet the requirements, such as, predicting workloads through analyzing historical data. In [17] and [18], a control loop mechanism was proposed to make resource allocation decision with a collection of virtual machines. Nevertheless, these works did not address the problem of dealing with the differences between each tier of the application. B. Urgaonkar et al. [5] proposed an analytical model for open and closed multi-tier application. But many efforts on precisely estimating the system parameters, e.g. service time and visit ratio from each server's log made this model difficult to apply in complex environment. We solved this modeling problem in a predictable and regression way to modify system parameters in order to be self-adaptive when predicting the performance. Similar measurements in [7] abstracted the response time as we did in this paper, in which they also took utilization as a factor, but this approach did not consider the fact that multiple services for a system may result in different service capacity in the system, so we introduced service rate to scale the capacity of handling different services in system.

The idea of using profit-driven policy in resource sharing policies is an old one [4, 10, 11]. The problem of maximization of SLA revenue in shared data center environment had attracted far-ranging of attention since long before. D. Ardagna et al. [6] designed a set of dispatching and control policies for the dispatcher in service oriented environment to maximize the provider's profits associated with multiple classes of SLAs. However, the nature of Cluster did not allow such allocation methodology to be widely applied in Clouds. Similar methodology as our method was found in [16], K. Tsakalozos et al. proposed a mechanism that automatically adjusted to the ever-changing equilibrium point caused by dynamic workloads and thus ensures that resource were shared proportionally to money spend by users. J. O. Fito et al. [14] presented an elastic web hosting provider that made use of outstanding technique to properly react to the dynamic load received by the web applications and achieved the aforesaid revenue by the maximization of the provider by performing an SLA-aware resource management. Compared to our method, we solved this problem to SaaS providers and remove or add VMs to the tier which can generate the most obvious tier to ensure the efficiency of VM reengineering.

Many existing works of dynamic provisioning either focus on one specific type of application or they do not consider with profits but only independent performance, such as QoS requirements. H. N. Van et al. [22] aimed to optimize a global utility function which integrated both the degree of SLA fulfillment and the operating cost. However, this method seemed more appealing to cloud provider or end users since less trade-off was required on their part. In paper [15], their research focused on enabling clouds to offer multi-tier Web application owners maximum response time guarantees while maximizing resource utilization. And also their preliminary results showed that dynamic bottleneck detection and resolution for multi-tier Web application hosted on the cloud would help to offer SLAs that can offer response time guarantee.

VI. CONCLUSION AND FUTURE WORK

In this paper, we addressed the new challenges raised by profit-driven provisioning of multi-tier Web application in Clouds. We proposed a dynamic profit-driven scheduling technique for multi-tier application that employs (i) a performance model to analyze and predict such application and provide insight into application's behavior to determine how much resources to allocate to each tier of the applications, enabling service providers to allocate their computing resource with flexibility, and (ii) a predictive and reactive method that determines when to remove or add these resources, and (iii) a profit-driven provisioning technique to maximize profits of SaaS provider. This work showed promising application of Cloud computing as this paradigm is primarily driven by its cost effectiveness. As the scheduling of composite multi-service Web application, particularly in cloud system, has been intensively studied, our proposed technique allows multi-tier web application to overcome their performance limitations, non-scalability and poor availability by taking advantages of Cloud computing infrastructures.

As a future work we are examining the practicability and correctness of our performance model in real environment, e.g, Eucalyptus-based cloud environment, combining the provision policy and method under consideration. For example, we will focus on the improvement of our provisioning algorithm as scaling up and down the number of VM according to the workload of the system is a crucial factor to the operation of controller. Since we have predicted the incoming workload, an appropriate method to revise the number of change should be proposed. In addition, we will discuss our scheduling policy to this problem under variable pricing schemes, e.g. in a spot market. Through multiple purchasing policy integration, we will choose the most profitable resource for the sake of SaaS provider.

REFERENCES

[1] A. Weiss, "Computing in the Clouds," Networker, vol, 11(4), Dec. 2007, pp. 16-25.

[2] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. H. Katz etc, "Above the Clouds: A Berkeley View of Cloud Computing," Technical Report No. UCB/EECS-2009-28. Symantec. 2010 State of the Data Center Global Data.

[3] http://www.symantec.com/content/en/us/about/media/pdfs/Symantec_DataCenter10_Report_Global.pdf

[4] M. Hajjat, X. Sun, Y. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, M. Tawarmalani, "Cloudward Bound: Planning for Beneficial Migration of Enterprise Applications to the Cloud, " In Proceedings of the ACM SIGCOMM'10 2010 conference on SIGCOMM.

[5] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitizer, A. Tantawi, "An Analytical Model for Multi-tier Internet services and its applications," In Proceedings of ACM SIGMETRICS, 2005 .

[6] D. Ardagna, M. Trubian, L. Zhang, "SLA based profit optimization in multi-tier systems," In Proceedings of the 2005 Fourth IEEE International Symposium on Network Computing and Application.

[7] Q. Zhang, L. Cherkasova, E. Smirni, "A Regression-Based Analytic model for Dynamic Resource Provisioning of Multi-Tier Application," In Proceedings of the 4th International Conference on Autonomic Computing, IEEE, 2007.

[8] [8] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, and Rajkumar Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," Software: Practice and Experience (SPE), Volume 41, Number 1, Pages: 23-50, ISSN: 0038-0644, Wiley Press, New York, USA, January, 2011.

[9] M. Arlitt, T. Jin, "A Workload Characterization Study of the 1998 World Cup Web Site", Network, issue: 3, page 30-37, IEEE, 2000.

[10] Y. C. Lee, C. Wang, A. Y. Zomaya, B.B. Zhou, "Profit-driven Service Request Scheduling in Clouds," In Proceedings of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), 2010.

[11] L. Zhang, D. Ardagna, "SLA Based Profit Optimization in Autonomic Computing System," In Proceedings of ICSOC '04 ACM, 2004.

[12] B. Schroeder, A. Wierman, and M. Harchol-Balter, "Open versus closed: A cautionary tale," In NSDI, pages 239-252, May 2006.

[13] Q. Zhang, E. Gurses, R. Boutaba, "Dynamic Resource Allocation for Spot Market in Clouds,"In Proceedings of the 11the USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services, 2011..

[14] J.O. Fito, I. Goiri and J. Guitart, "SLA-driven Elastic Cloud Hosting Provider, " In Proceedings of 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, IEEE, 2010.

[15] W. Iqbal, M. N. Dailey, D. Carrera, " SLA-Driven Dynamic Resource Management for Multi-tier Web Applica tions in a Cloud," In Proceedings of the (CCGRID) 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp.832–837,2010.

[16] K. Tsakalozos, H. Kllapi, E. Sitaridi, M. Roussopoulos, D. Parapras, A. Delis, "Flexible Use of Cloud Resources through Profit Maximization and Price Discrimination," In Proceedings of IEEE 27th International Conference on Data Engineering (ICDE), 2011.

[17] P. Padala, K.G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, K. Salem, "Adaptive Control of Virtualized Resources in Utility Computing Environments," In Proceedings of the ACM SIGOPS/EuroSys European Conference on Computer System 2007, pp. 289-302.

[18] Q. Li, Q. Hao, L. Xiao, Z. Li, "Adaptive Management of Virtualized Resource in Cloud Computing Using Feedback Control," In Proceedings of 1st International Conference on Information Science and Engineering, 2009.

[19] D. A. Bacigalupo, J. van Hemert, A. Usmani, D. N. Dillenberger, G. B. Wills and S. A. Jarvis. Resource Management of Enterprise Cloud Systems Using Layer Queuing and Historical Performance Models. In Proceedings of IEEE International Symposium on Parallel Distributed Processing, 2010.

[20] S. Ferretti, V. Ghini, F. Panzieri, M. Pellegrini, and E. Turrini, " QoS-aware Clouds" In Proceedings of IEEE 3rd International Conference on Cloud Conputing, 2010.

[21] J.Li, J. Chinneck, M. Woodside, M. Litoiu, and G. Iszlai, "Performance Model Driven QoS Guarantees and Optimization in Clouds," In Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, 2009.

[22] H.N. Van, F. D. Tran, J.M. Menaud, "SLA-aware Virtual Resource Management for Cloud Infrastructures," In Proceedings of Ninth IEEE International Conference on Computer and Information Technology, 2009.