Revenue Maximization Using Adaptive Resource Provisioning in Cloud Computing Environments

Guofu Feng
School of Information Science,
Nanjing Audit University,
Nanjing, China
njufgf@gmail.com

Saurabh Garg, Rajkumar Buyya
CLOUDS Laboratory,
Dept. of Computing and Info. Systems
Melbourne, Australia
{sgarg, raj}@csse.unimelb.edu.au

Wenzhong Li
State Key Lab for Novel Software Technology,
Nanjing University,
Nanjing, China
lwz@nju.edu.cn

Abstract**—***Compared with the traditional computing models such as grid computing and cluster computing, a key advantage of Cloud computing is that it provides a practical business model for customers to use remote resources. However, it is challenging for Cloud providers to allocate the pooled computing resources dynamically among the differentiated customers so as to maximize their revenue. It is not an easy task to transform the customer-oriented service metrics into operating level metrics, and control the Cloud resources adaptively based on Service Level Agreement (SLA). This paper addresses the problem of maximizing the provider's revenue through SLA-based dynamic resource allocation as SLA plays a vital role in Cloud computing to bridge service providers and customers. We formalize the resource allocation problem using Queuing Theory and propose optimal solutions for the problem considering various Quality of Service (QoS) parameters such as pricing mechanisms, arrival rates, service rates and available resources. The experimental results, both with the synthetic dataset and with traced dataset, show that our algorithms outperform related work.*

Keywords-*Cloud Computing; Service Level Agreement; Resource Allocation*

I. INTRODUCTION

Cloud computing represents the delivery of computing as a service, whereby such resources as CPU, software, information, and devices are provided to end-users as a metered service over the Internet. Cloud computing provides a vastly efficient, flexible, and cost-effective way for IT to meet escalating business needs: IT as a Service.

This study considers two scenarios. First, some multinational corporations rent the IT infrastructure and services from a Cloud provider to build a virtual datacenter/cluster for their branches all over the world. Second, a Cloud datacenter provides transaction services of electronic commerce for some small companies.

Under both these scenarios, the Cloud providers aim to obtain the revenues by leasing the resources as services and the customers rent services to meet their application requirements. The customers only rent the required services and only pay for the consumed ones. The business model based on Service Level Agreements (SLA) plays a crucial role in Cloud paradigm and it is the key point to distinguish Cloud computing from conventional distributed computing paradigms such as grid computing and cluster computing [1].

Particularly, SLAs facilitate the transactions between customers and service providers by providing a platform for consumers to indicate their required service level or Quality of Service (QoS) [2]. SLA usually specifies a common understanding about responsibilities, guarantees, warranties, performance levels in terms of availability, response time, etc. A charging model, including the charging mechanism and penalties in case of non-compliance of SLA, is also specifically defined. Service providers usually charge customers according to the achieved performance level. For example, Amazon EC2 offers three types of compute services (i.e., on-demand, spot and reserved) at different prices based on their service/ performance levels.

SLA is also the fundamental basis for service providers to provision their Cloud resources. The service provider using a multi-tenant model assigns pooled compute resources in the form of a virtual machine to multiple consumers. The pooled physical resources can be assigned and reassigned to the different virtual machines dynamically based on consumers' requests and available resources.

We define ***service instance*** as the combination of a customer and his/her assigned virtual machine (a certain type of rental resource or service). A customer may belong to more than one service instances since a customer may request many types of services.

The service instances may have different attributes such as arrival rate, execution time, and pricing mechanism. Even for the same service instance, its request arrival rate can vary with different random distributions. The challenge is how much underline physical resources must be assigned to maintain the promised level of performance as described in SLAs. Since resource allocation strategies have an impact on the service performance, a fundamental problem faced by any Cloud service provider is how to maximize their revenues by allocating resources dynamically among the service instances and providing differentiated performance levels based on SLA and measurable performance indices.

Generally, we should allocate more resources for those instances with high arrival rate and high price in order to obtain high revenues. However, some instances may have high throughput rate (high arrival rate) and low price, and vice versa. Thus, it is nontrivial to allocate the resources

1550-5510/12 $26.00 © 2012 IEEE
DOI 10.1109/Grid.2012.16

properly and precisely according to the heterogeneous parameters.

Therefore, this paper focuses on how a Cloud data center can maximizes the SLA-based revenues by proper resource allocation and presents optimal allocation algorithms for different pricing schemes. The basic idea in this paper is to schedule the Cloud resources among different service instances adaptively based on the dynamically collected information.

Our main **contributions** in this paper include:

(1) Queuing Theory based mathematical formulation for the resource allocation problem. The formulation models the application's requirement using various parameters such as resource quantity, request arrival, service time and pricing model.

(2) Optimal SLA-based resource allocation algorithms among different Cloud service instances, by which Cloud providers can maximize their revenues. Our simulation shows that they outperform related work.

The remainder of this paper is organized as follows. Section II introduces two pricing models in terms of response time. Section III formulates the problem of resource allocation and provides the optimal solutions to these two problems. In Section IV, we have carried out several simulations to verify our solutions. Section V presents some related work. Finally, Section VI concludes our paper.

II. MODELS

Cloud computing as discussed by Buyya et al. [2], should incorporate autonomic resource management models that effectively self-manage changes in service requirements to satisfy both new service demands and service obligations according to the signed SLA. In this paper, we contribute towards this aim of Cloud computing and therefore, consider a similar scenario where a Cloud provider offers various services to customers at different SLA. Each service is hosted within a datacenter using certain amount of virtual infrastructure, which can grow and shrink on demand. The objective is to find how many servers should be assigned for each service instance in order to achieve maximum revenue for a given a charging/pricing model. Figure 1-1 illustrates the system model of a Cloud-computing environment.

A. *Mathematical Model*

We assume that the Cloud datacenter is composed of N homogenous servers. The servers are grouped into clusters dynamically and each server can only join one cluster simultaneously. Every service instance is mapped to a server cluster. Each cluster is virtualized as a single machine. The users do not need to know the specific details of the operation on virtual machines. A service provider signs long-term SLAs with m customers. Every service instance is allocated to $n_1, n_2, \ldots n_m$ servers to provide services. We assume that the capability of every virtual machine is proportional to the number of assigned servers. This assumption is reasonable especially for those computing tasks that can be divided into several pieces and dispatched to many servers to execute concurrently. For example, many dynamic web pages are composed of many parts that should be computed separately; or some tasks can be decomposed for parallel computing.

We assume that the requests from any service instance arrive at the system in a Poisson distribution with average arrival rate λ and the processing times by one server follow a negative exponential distribution with average service rate $1/\mu$ (μ is the number of processed requests per unit time). Then the service rate of a virtual machine with n servers is $1/n\mu$.

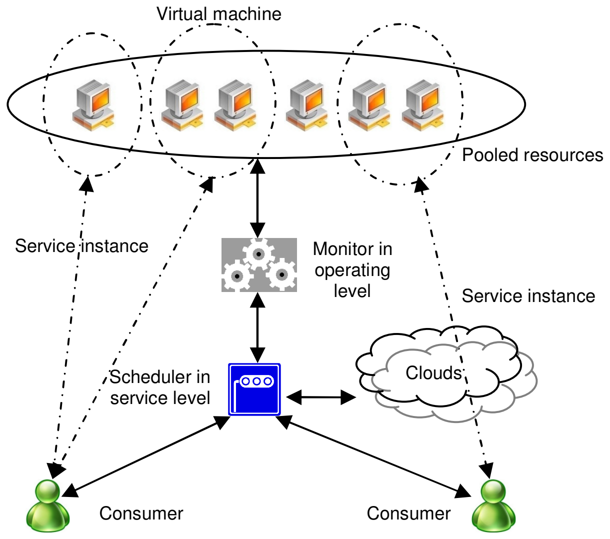


Figure 1-1 system model of Cloud

We also assume that it costs much for servers to shift their running environments. For example, it needs a long time to read the commercial data of a new customer into cache from the external memory. Hence, the requests cannot be moved easily from one virtual machine to another. Each service instance, a virtual machine associated with a user, can be modeled as a FIFO (First In First Out) M/M/1 queue. Here we define service intensity ρ as the ratio of arrival rate to service throughput of one server,

$$\rho = \lambda / \mu \qquad (1)$$

The notations used in this paper are:

- λ Arrival rate of service requests of each instance
- μ Service rate of service instance with one server
- ρ Service intensity
- m Number of service instances
- n Variable of assigned servers to an instance
- N Number of all the servers in resource pool
- b A constant of service price
- B Benchmark to evaluate the service performance
- r Variable of response time
- R User requirement on response time in SLA
- g Mean revenue from a service provision
- G Provider's revenue from Cloud provision
- q Slope of pricing curve in Mean Response Time (*MRT*)
- F Performance level of service in *MRT*

B. *Pricing Model in terms of Mean Response Time (MRT)*

The pricing mechanism is usually defined in SLA. It specifies how service requests are charged. For instance, requests can be charged based on submission time (peak/off-peak), pricing rates (fixed/changing) or availability of resources (supply/demand) [2]. However, these mechanisms are service provider-oriented. Here we propose two customer-oriented pricing mechanisms *MRT* and *IRT*, in which the customers are charged according to achieved service performance in terms of mean response time.

Mean response time is a commonly used metric to evaluate the service performance. Here we define response time as the interval from when a request arrives at the system to the instant at which the service is completed (ignoring the link latency). The response time can also be termed as the ***sojourn time*** of a request in ***M/M/1 queuing model***. It is necessary for service providers to divide the whole provisioning time into some slots. We calculate the mean response time of every time slot independently because arrival rate varies over time.

We first propose a pricing model called *MRT*. We use *F* to denote an offset factor of actual response time to benchmark. We define *F* as,

$$F = r / R \tag{2}$$

where *r* is the measured average response time during a time slot and *R* represents a benchmark of response time defined in SLA. Every service instance has different *R*, which is determined by customer's actual requirement. For example, the recommended response time for transactions in e-commerce is 2-4 seconds. This pricing model is also called service demand driven model [3].

Then we formulate the pricing mechanism as,

$$B = b(1 - F) \tag{3}$$

where *B* is the price of each service provision and *b* is the price constant. As displayed in Fig. 1 (a), the price *B* actually is a linear function of mean response time *r*. When mean response time is longer than the threshold *R*, service provider will be penalized. Figure 1 also shows us that b/R is the slope of pricing function,

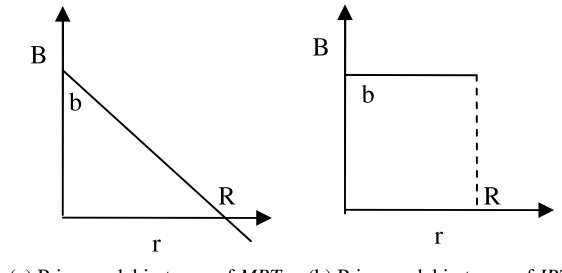
$$q = b / R \tag{4}$$

C. *Pricing Model in terms of Instant Response Time (IRT)*

MRT may work well when the measurements are evenly distributed over a narrow range. However, *MRT* is not meaningful as a performance metric when the response time varies quite a bit over a large range. Therefore, we propose another pricing model in terms of *IRT*. A request in *IRT* is charged according to the measured response time. That is,

$$B = \begin{cases} b, r \le R \\ 0, r > R \end{cases} \tag{5}$$

The pricing model *IRT* can be illustrated as Fig. 1(b). The billing under this model is determined by the number of service provisions with response time within required *R*.



(a) Price model in terms of *MRT* (b) Price model in terms of *IRT*

Figure 1. Price models

III. Mathematical Formulations for Resource Allocation Problem in Clouds

A. *Optimal Allocation Based on MRT*

According to the research conclusions on Queuing Theory of M/M/1 model, the average response time r_i of service instance *i* at the steady system state,

$$r_i = \frac{1}{n_i \mu_i - \lambda_i} \tag{6}$$

Then service performance level F_i is,

$$F_i = \frac{1}{(n_i \mu_i - \lambda_i) R_i} \tag{7}$$

According to (3), the mean revenue g_i brought by a service provision is,

$$g_i = b_i \left(1 - \frac{1}{(n_i \mu_i - \lambda_i) R_i} \right) \tag{8}$$

The overall revenues during a time slot from service instance *i* is,

$$G_i = \lambda_i g_i = \lambda_i b_i \left(1 - \frac{1}{(n_i \mu_i - \lambda_i) R_i} \right) \tag{9}$$

Then our optimization problem can be formulated as,

$$\begin{aligned} &Objective:\ Max \sum_{i=1}^{m} \lambda_i b_i \left(1 - \frac{1}{(n_i\mu_i - \lambda_i) R_i}\right) \\ &s.t. \sum_{i=1}^{m} n_i = N \end{aligned} \quad (10)$$

We resolve this problem using Lagrange Multiplier Method in the following. Constructing Lagrange composite function,

$$L(n_i) = \sum_{i=1}^{m} \lambda_i b_i \left(1 - \frac{1}{(n_i\mu_i - \lambda_i) R_i}\right) + \overline{\lambda}\left(N - \sum_{i=1}^{m} n_i\right) \quad (11)$$

where $\overline{\lambda}$ is a constant of Lagrange multiplier.

Letting $dL/dn_i = 0,\ i = 0,1,2...m$,

$$\frac{\lambda_i b_i}{R_i} \frac{\mu_i}{(n_i\mu_i - \lambda_i)^2} - \overline{\lambda} = 0 \quad (12)$$

$$n_i = \sqrt{\frac{1}{\overline{\lambda}}} \sqrt{q_i \rho_i} + \rho_i \quad (13)$$

Substituting (13) into the constraint of the optimization problem (10),

$$N = \sqrt{\frac{1}{\overline{\lambda}}} \sum_{j=1}^{m} \sqrt{q_j \rho_j} + \sum_{j=1}^{m} \rho_j \quad (14)$$

$$\sqrt{\frac{1}{\overline{\lambda}}} = \frac{N - \sum_{j=1}^{m} \rho_j}{\sum_{j=1}^{m} \sqrt{q_j \rho_j}} \quad (15)$$

Substituting (15) into (13), we yield the final answer, i.e., the number of servers used for each service instance,

$$n_i = \frac{N - \sum_{j=1}^{m} \rho_j}{\sum_{j=1}^{m} \sqrt{q_j \rho_j}} \sqrt{q_i \rho_i} + \rho_i \quad (16)$$

However, equation (16) is correct on the premise that (6) holds. Equation (6) is valid only when the request arrival rate of each service instance is less than service processing rate according to the conclusions on Queuing Theory of M/M/1 model. Otherwise, the length of request queue with FIFO will not converge and the mean response time always increases as time elapses. Therefore, our conclusion of (16) holds only if arrival rate is less than service processing rate,

$$\lambda_i < n_i \mu_i \quad (17)$$

$$n_i > \rho_i \quad (18)$$

Moreover, Figure 1 shows us that the service providers will be penalized once mean response time cannot met the service demand R_i. Therefore, our service allocation strategy guarantees that mean response time should be less than R_i,

$$r_i = \frac{1}{n_i\mu_i - \lambda_i} < R_i \quad (19)$$

$$n_i > \frac{1}{\mu_i R_i} + \rho_i \quad (20)$$

Equation (18) and (20) offer us the lower bound of assigned resources for each service instance. It is obvious that (18) holds once (20) is satisfied.

B. *Optimal Allocation Based on IRT*

According to the conclusions on Queuing Theory of M/M/1 model, the sojourn time probability distribution is,

$$\omega(t) = (\mu - \lambda) e^{(\lambda - \mu)t} \quad (21)$$

We assume that service instance *i* is allocated to n_i servers. Then the mean revenue brought by a service provision is,

$$\begin{aligned} g_i &= \int_0^{R_i} b_i \omega(t) dt = \int_0^{R_i} b_i (\mu - \lambda) e^{(\lambda_i - n_i\mu_i)t} dt \\ &= b_i \left(1 - e^{(\lambda_i - n_i\mu_i)R_i}\right) \end{aligned} \quad (22)$$

Then the overall mean revenue from service instance *i* during a time slot is,

$$G_i = \lambda_i g_i = \lambda_i b_i (1 - e^{(\lambda_i - n_i\mu_i)R_i}) \quad (23)$$

Thus, our optimization problem can be formulated as,

$$\begin{aligned} &Objective:\ Max \sum_{i=1}^{m} \lambda_i b_i \left(1 - e^{(\lambda_i - n_i\mu_i)R_i}\right) \\ &s.t. \sum_{i=1}^{m} n_i = N \end{aligned} \quad (24)$$

Constructing Lagrange composite function,

$$L(n_i) = \sum_{i=1}^{m} \lambda_i b_i \left(1 - e^{(\lambda_i - n_i\mu_i)R_i}\right) + \overline{\lambda}\left(N - \sum_{i=1}^{m} n_i\right) \quad (25)$$

where $\overline{\lambda}$ also is Lagrange multiplier.

Letting $dL/dn_i = 0, i = 0,1,2...m$,

$$\lambda_i \mu_i b_i R_i e^{(\lambda_i - n_i \mu_i) R_i} - \overline{\lambda} = 0 \qquad (26)$$

$$n_i = \frac{\ln(\lambda_i \mu_i b_i R_i)}{\mu_i R_i} - \frac{\ln \overline{\lambda}}{\mu_i R_i} + \rho_i \qquad (27)$$

Substituting (27) into (24),

$$N = \sum_{j=1}^{m} \frac{\ln(\lambda_j \mu_j b_j R_j)}{\mu_j R_j} - \ln \overline{\lambda} \sum_{j=1}^{m} \frac{1}{\mu_j R_j} + \sum_{j=1}^{m} \rho_j \qquad (28)$$

$$\ln \overline{\lambda} = \frac{\sum_{j=1}^{m} \frac{\ln(\lambda_j \mu_j b_j R_j)}{\mu_j R_j} + \sum_{j=1}^{m} \rho_j - N}{\sum_{j=1}^{m} \frac{1}{\mu_j R_j}} \qquad (29)$$

Substituting (29) into (27), we can obtain the result,

$$n_i = \rho_i + \frac{\ln(\lambda_i \mu_i b_i R_i)}{\mu_i R_i} - \frac{\sum_{j=1}^{m} \frac{\ln(\lambda_j \mu_j b_j R_j)}{\mu_j R_j} + \sum_{j=1}^{m} \rho_j - N}{\mu_i R_i \sum_{j=1}^{m} \frac{1}{\mu_j R_j}} \qquad (30)$$

Because of the same reason as previous subsection, the arrival rate should be larger than the service rate (processing speed) of the virtual machine composed of all the assigned servers. Namely, the allocated resource n_i should be larger than service intensity ρ.

IV. PERFORMANCE EVALUATION

In this section, we present our experimental results on the validity of our algorithms for optimizing the resource provisioning in the Cloud environment. In the following, we provide two types of experiments, where requests are modeled using synthetic dataset and traced dataset.

We develop a C-based simulator based on a time-driven model to conduct the experiments. Simulation clock increases at a constant rate of one millisecond. After each millisecond, we check and handle those events that happen at the current time slot. The events mainly include four types: arrival, departure, resource reallocation, and output the experiment results.

Since our main goal is to maximize the revenue of the Cloud provider, we use revenue from service provisioning as our main metric to evaluate the strategies. In the evaluation, we use a resource allocation algorithm proposed by Michele in his thesis [4], as our base algorithm to compare with, because this work is the most recent work that is similar to ours.

In the following, we use *MRT*, *IRT*, and *Heuristic* to denote our optimal algorithm based on mean response time, our optimal algorithm based on instant response time, and the heuristic allocation algorithm proposed by Michele [4] respectively. The related parameters and their default values are listed in Table I.

TABLE I. PARAMETERS AND THEIR DEFAULT VALUES

Parameter	Default Value	Parameter	Default Value
Arrival Rate λ	Random (20…30)	Service Rate μ	10
Intercept b	20/60	Intercept R	Random (2…32)/60
customers m	20	Revenue unit	$

A. *Simulations with Synthetic Data*

In this section, each simulation runs for one hour. We partition the time into slots. After each time slot, we calculate and output the revenue gain during this slot. Our results are derived from the last time slot.

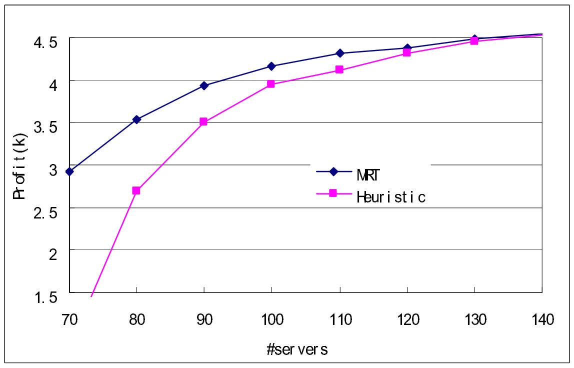


Figure 2. Revenue versus the number of servers

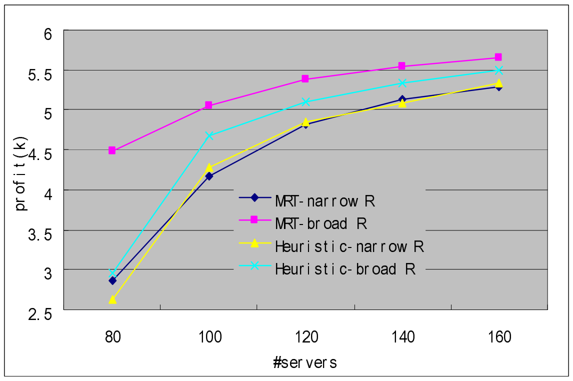


Figure 3. Revenue versus the number of serves under different *q*

Figure 2 is the comparison of revenue between *MRT* and *Heuristic* with different servers. Time slot in this simulation is set 30 minutes. Figure 2 shows that the revenue from *MRT* and *Heuristic* increases with the increase of the server number. To calculate dG/dn in (9),

$$G' = \frac{\lambda b \mu}{R\left(n\mu - \lambda\right)^2} \tag{31}$$

G' in (31) is always larger than zero, which implies that the revenue always increases with the increase of servers. However, G' decreases with the increase of n, which means that revenue increases more and more slowly with the increase of server number.

Figure 2 shows us that our resource allocation strategy of *MRT* always outperforms *Heuristic*. This is because *MRT* is the optimal strategy in theory. Moreover, the superiority of *MRT* is remarkable especially when the resources are relatively rare. Therefore, *MRT* is much valuable to improve the revenue through proper allocation when the resource is rare or the accepted requests are numerous.

Figure 2 also shows us that *Heuristic* performs well, mostly close to *MRT*. Equation (16) can explain this result. The optimal allocation can be divided into two steps. First, each service instance is allocated according to ρ. Second, the remaining resources are allocated according to ρ and q. Thus, the service intensity mostly dominates the optimal allocation. What's more, *Heuristic* actually achieves the same optimal allocation as *MRT* when q follows the same distribution as ρ. This may explain why *MRT* and *Heuristic* are mostly close. Figure 3 also can verify this viewpoint.

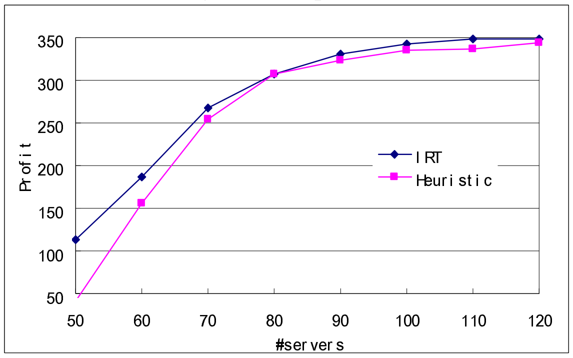


Figure 4. revenue versus the number of servers

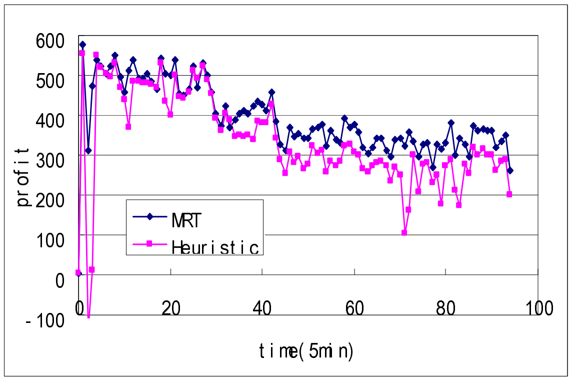


Figure 5. Evolution of revenue during 5 minutes over time with traces

Figure 3 is the revenue comparison under the simulations with different q between *Heuristic* and *MRT*. Each service instance in the simulations has the same ρ, b and different response time demands R. The revenue, both of *Heuristic* and *MRT*, increases with the increase of response time demand. However, *MRT* rises remarkably especially when the resources are rare. When all the b is fixed, q in simulations with broad R is more diverse. q has impact on the allocation in our optimal algorithm, but has no impact on *Heuristic* according to (16). That is why revenue of *MRT* increases more than *Heuristic* when q changes from a narrow distribution to a broad distribution.

Figure 4 is the comparison of revenue between *IRT* and *Heuristic*. Time slot in this simulation is set to 10 minutes. It shows us that, due to the same reason as *MRT*, both the revenue of *IRT* and *Heuristic* increase with the increase of server number. *IRT* always outperforms *Heuristic* because *IRT* is optimal in theory. Moreover, the superiority of *IRT* is remarkable especially when resources are relatively rare. Therefore, *IRT* is much valuable to improve the revenues through proper allocation when the resource is rare or the accepted requests are numerous.

B. *Simulations with Traced Data*

We use the traced data to simulate the requests and allocate the resources dynamically and adaptively according to probed parameters. Due to unavailability of public Cloud data, we used web application data, which are taken from [5]. All the data are records of HTTP requests to WWW servers. We intercept consecutive request records of 8 hours from the traces to simulate the arrival of a service instance. The detailed information is shown in Table II.

TABLE II. METADATA ON TRACED DATASET

#	source	Date	time	#records
1	EPA-HTTP	30 Aug. 1995	09:00-17:00	31385
2	EPA-HTTP	30 Aug. 1995	16:00-24:00	14714
3	SDSC-HTTP	22 Aug. 1995	09:00-17:00	15479
4	SDSC-HTTP	22 Aug. 1995	16:00-24:00	7178
5	NASA-HTTP	01 Jul. 1995	00:00-08:00	16481
6	NASA-HTTP	01 Jul. 1995	09:00-17:00	24021
7	NASA-HTTP	01 Jul. 1995	16:00-24:00	25476
8	NASA-HTTP	25 Jul. 1995	00:00-08:00	9360
9	NASA-HTTP	25 Jul. 1995	09:00-17:00	34965
10	NASA-HTTP	25 Jul. 1995	16:00-24:00	20652

We partition the time into slots, each with a length of 5 minutes. During the execution, we count the number of arrived requests at each time slot. Then we predict the average arrival rate of next time slot according to the records of previous and current slot. The predicting algorithm is formulated as,

$$\lambda_{next} = \lambda_{current} + 0.5(\lambda_{current} - \lambda_{previous}) \tag{32}$$

where $\lambda_{previous}$ and $\lambda_{current}$ are the real number of requests during the passed two slots. They can be counted easily.

The servers are partitioned into 10 groups, each with a FIFO (First In First Out) waiting queue. At the end of each time slot, the system re-allocates the resources to every service instance according to λ_{next} . However, we do not adjust the group size immediately after the calculation, but after a time slot. This is because the current queue length is a consequence of previous time slot. The specific parameters are listed in Table III.

TABLE III. PARAMETERS AND THEIR DEFAULT VALUES

Parameter	Default Value	Parameter	Default Value
Service Time Distribution	Negative Exponential	Intercept b	Random (10…20)/60
Service Rate μ	Random (10…15)	Intercept R	Random (10…30)/60 mins
customers m	10		

Figure 5 depicts the evolution of revenues every 5 minutes. Eighty servers are deployed in this simulation. The overall curve tendencies, both of *MRT* and *Heuristic*, mainly depend on arrival rate. It can be seen from Fig. 5 that *MRT* outperforms *Heuristic*. The average revenues every 5 minutes of *MRT* and *Heuristic* during the 8 hours are 392.76 and 334.07 respectively. The revenues of *MRT* is 17.57% higher than *Heuristic*.

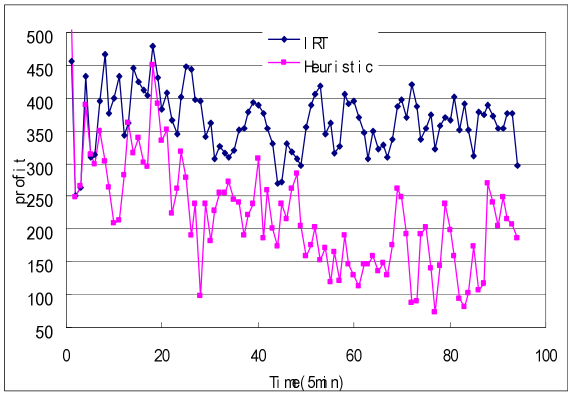


Figure 6. Evolution of revenues during 5 minutes over time

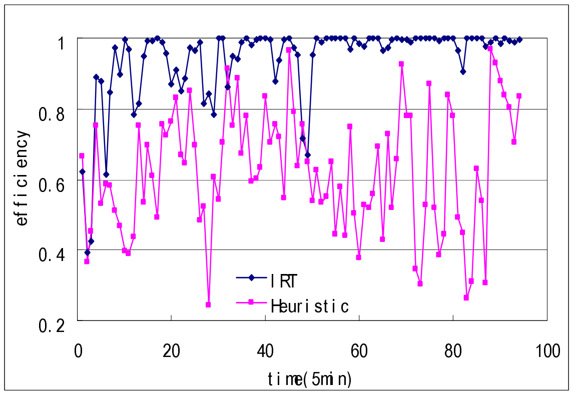


Figure 7. Evolution of efficiency during 5 minutes over time

Figure 6 and Figure 7 illustrate the performance comparison of *IRT* and *Heuristics*. There are 60 servers deployed in the simulations. Figure 6 is the revenues of both *IRT* and *Heuristic* over time. It shows us that *IRT* is better than *Heuristic*. The average revenues every 5 minutes of *IRT* and *Heuristic* during the 8 hours are 360.87 and 217.45 respectively. The revenues of *IRT* is 65.9% higher than *Heuristic*.

Figure 6 indicates that *IRT* is better in improving revenues than *Heuristic*. However, it does not reflect the real performance levels of these two strategies. For example, maybe both algorithms are at a very low level compared with other algorithms. Figure 7 is the *efficiency* evolution of *IRT* and *Heuristic* over time.

Here ***efficiency*** is defined as the ratio of actual revenues to the maximum revenues in theory, namely the whole revenues when all the requests are responded within their required response time demand. Efficiency reflects the real performance more objectively with the overall benchmark and standard.

Figure 7 shows us that *IRT* mostly closes to the revenues upper bound in theory, with and average efficiency of 93.89%. *Heuristic* has a low and fluctuant efficiency, with an average efficiency of 62.24%. The efficiency of *IRT* is much higher than *Heuristic* with 31.65%.

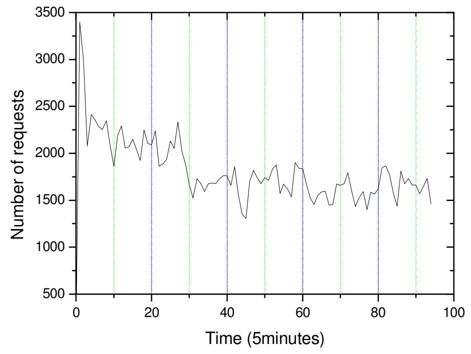


Figure 8. Evolution of total arrival requests over time

Simultaneously, both Figure 5 and Figure 6 show us that the revenues of *MRT* and *IRT* during the 47th time slot decreases sharply. We believe that it results from the extreme fluctuation of arrival rate. As displayed in Figure 8, the arrival rate decreases sharply from 42nd to the 45th time slot, while it rises quickly after the 46th time slot. Thereby, the prediction of arrival rate by Equation (32) is not correct, which misleads the resource allocation. The revenues of *MRT* and *IRT* also decrease sharply at the 26th time slot because of the same reason. Therefore, a more precise prediction of arrival rate has a positive impact on the validity of our resource allocation strategies during their practical applications.

V. RELATED WORK

The business model is the key characteristic to distinguish Cloud computing from previous typical computing paradigms [1]. As a bridge, SLA plays a vital role in facilitating the realization of an economic-based Cloud system. SLA provides mechanisms and tools that allow service providers and end users to express their requirements and constraints such as response time and price scheme. It is very natural but challenging for service providers to allocate the resources dynamically among the end-users based on the agreement, thereby to maximize the revenues.

There is an extensive literature on resource management techniques for commercial data centers. Utility is often adopted as a metric for resource allocation. Walsh et al. [6] discuss a distributed architecture with the aim of solving the resource allocation problem for dedicated data center architecture with dynamic virtual pool. The emphasis of these papers is on the use of utility functions as fundamental framework to optimize resource usage rather than the utility of data center. Utility functions provide criteria for trading off between multiple competing system objectives. Rajkumar et al. [7] propose a QoS-aware resource allocation model Q-RAM. The objective of Q-RAM is to maximize the utility derived from concurrent applications under the multi-dimensional QoS constraints. Ghosh et al. [8] and Hansen et al. [9] further extend Q-RAM, where the scalability and ability to make resource trade-off decisions are enhanced.

Many works illustrate how to meet the QoS and SLA requirements by the proper resource allocation. Menascé et al. [10, 11] propose an approach based on hill climbing techniques to guide the search for the best combination of configuration parameters of a multilayered architecture. It uses the existing resources best in a manner that the desired QoS levels are met to cope with short term fluctuations in the workload. Chandra et al. [12] present techniques for dynamic resource allocation in shared web servers. Using a combination of online measurement, prediction and adaptation, the techniques can dynamically determine the resource share of each application based on QoS and measured workload. Levy et al. [13] present an architecture and prototype implementation of a performance management system, where cluster utility is used to encapsulate business value in the face of service level agreements. The system dynamically allocates server resources, balances the load among multiple classes according to performance demand. Li et al. [14] take the minimization of resource consumption as the objective and propose a strategy for autonomic computing to meet SLA requirements in terms of response time and server utilization. However, the majority of previous work does not take the economic issues related to SLAs into account.

Zhang and Ardagna [15] propose a resource allocation controller for autonomic computing data center. The objective is to maximize the provider's revenues associated with multi-class Service Level Agreement. In the system, the revenue depends on discrete QoS levels and the revenue gained per request increases with the achieved performance level. Liu et al. [16] propose a theoretical model to maximize the revenues of a hosting platform subject to multi-class SLAs. Tim, et al. [17] presents a framework that links technical and economical aspects to the management of computational resources. It combines some technical methods such as dynamic pricing, different job priorities, and client classification into an economically enhanced resource management that increases revenue for the local resource sites. Villella et al. [18] study how a service provider should allocate the application tier of an Ecommerce application subject to QoS constraints. The paper models each server as an M/G/1/PS queue, and derives three simple methods that approximate the allocation that maximizes revenues. Anthony et al. [19] propose overbooking models to find an ideal overbooking limit that exceeds the maximum Grid capacity without incurring greater compensation cost. However, all the works adopt a flat-rate discrete price levels while we adopt a continuous price function in this paper. We also provide the formal precise answer to the problems.

There are several works sharing similar scenarios with our work. Zhu et al. [3] proposes an allocation strategy of server resources among customers to minimize the mean response time. Nevertheless, this work does not consider the economic model. In addition, the parameter of weight q in optimal solution lacks the specific practical meaning. The work [4] is very similar to ours. It provides two strategies for the resource allocation, *Heuristic* and *Greedy*. *Greedy* is optimal but it often costs an impractically long execution time while the improved algorithm does not always work well. *Heuristic* is simple but our work displays that its validity is affected by the environment parameters.

VI. Conclusions and Future Directions

Cloud computing has emerged as a new business model for delivering various IT services to customers in a "pay-as-you-go" manner. The business model of Cloud computing requires legal service level agreements to facilitate the collaboration between end-users and service providers. This paper addresses how to maximize providers' revenues based on the performance-aware pricing model in SLAs through the proper resource allocation among the customers.

This paper has formulated the optimization problem and given the optimal results by the Method of Lagrange Multiplier. Our experimental results have shown that the proposed algorithms in this paper always outperform the previous work and they are of higher significance especially when the Cloud environments face with computing resource shortage.

However, we considered the server group serving each customer as an M/M/1 model in this paper. It is more reasonable to model the server group as an M/M/c model when running environments are easily initiated. Moreover, pricing model is the foundation of our work. But pricing model can be very complex. Therefore, it will be interesting to see how to apply techniques proposed in this paper to such scenarios. We will further investigate these problems in the future work.

Acknowledgments

This work was supported by the National Basic Research Program of China (Grant No. 2009CB320705), the National Natural Science Foundation of China (Grant Nos. 60803111, 61073028), and the Jiangsu Natural Science Foundation (Grant Nos. BK2009396, BK2009100). The first author would like to thank Jiangsu Provincial Government and the CLOUD Laboratory for supporting and hosting his visit to The University of Melbourne, Australia.

REFERENCES

[1] Chunye Gong, Jie Liu, Qiang Zhang, Haitao Chen and Zhenghu Gong. The Characteristics of Cloud Computing. In Proc. of 39th International Conference on Parallel Processing Workshops, 2010, 275-279.

[2] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, Ivona Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 25 (2009), 599-616.

[3] Huican Zhu, Hong Tang, Tao Yang. Demand-driven Service Differentiation in Cluster-based Network Servers. In Proc. IEEE INFOCOM 2001, 679-688.

[4] Michele Mazzucco. Revenue Maximization Problems in Commercial Data Centers. PhD Thesis, University of Newcastle, May 7, 2009.

[5] The Internet Traffic Archive, http://ita.ee.lbl.gov/html/traces.html, accessed on August 23, 2011.

[6] William E. Walsh, Gerald Tesauro, Jeffrey O. Kephart, and Rajarshi Das. Utility Functions in Autonomic Systems. In Proc. of the International Conference on Autonomic Computing, 2004, 70–77.

[7] R. Rajkumar, C. Lee, J. Lehoczky, D. Siewiorek. A resource Allocation Model for QoS Management. In Proc. of the 18th IEEE Real-Time Systems Symposium. 1997, 298-307.

[8] S. Ghosh, R. Rajkumar, J. Hansen, and J. Lehoczky. Scalable Resource Allocation for Multi-processor QoS Optimization. In Proc. of 23rd International Conference on Distributed Computing Systems, 2003, 174-183.

[9] J.P. Hansen, Sourav Ghosh, Ragunathan Rajkumar, and J. Lehoczky. Resource Management of Highly Configurable Tasks. In Proc. 18th International Parallel and Distributed Processing Symposium, 2004, 116.

[10] Daniel A. Menascé, Daniel Barbará, and Ronald Dodge. Preserving QoS of E-Commerce Sites Through Self-Tuning: A Performance Model Approach. In Proc. of 3rd ACM conference on Electronic Commerce, 2001, 224-234.

[11] Mohammed N. Bennani and Daniel Menascé. Resource Allocation for Autonomic Data Centers Using Analytic Performance Models. In Proc. of the Second International Conference on Autonomic Computing, 2005, 229–240.

[12] Abhishek Chandra, Weibo Gong, and Prashant Shenoy. Dynamic Resource Allocation for Shared Data Centers Using Online Measurements. In Proc. of the 11th IEEE/ACM International Workshop on Quality of Service, 2003, 381-400.

[13] R. Levy, J. Nagarajarao, G. Pacifici, A. Spreitzer, A. Tantawi, and A. Youssef. Performance Management for Cluster Based Web Services. In Proc. of IEEE 8th International Symposium on Integrated Network Management, 2003, 247-261.

[14] Ying Li, Kewei Sun, Jie Qiu, and Ying Chen. Self-reconfiguration of Service-based Systems: A Case Study for Service Level Agreements and Resource Optimization. In Proc. of IEEE International Conference on Web Services, 2005, 266-273.

[15] Li Zhang and Danilo Ardagna. SLA Based Profit Optimization in Autonomic Computing Systems. In Proc. of the 2nd International Conference on Service Oriented Computing, 2004, 173–182.

[16] Zhen Liu, Mark S. Squillante, and Joel L. Wolf. On Maximizing Service-Level-Agreement Profits. In Proc. of the 3rd ACM Conference on Electronic Commerce, 2001, 213-223.

[17] Tim Püschel, Nikolay Borissov, Dirk Neumann, Mario Macías, Jordi Guitart and Jordi Torres. Extended Resource Management Using Client Classification and Economic Enhancements. In Proc. of eChallenges Conference, 2007, 65-72.

[18] Daniel Villela, Prashant Pradhan, and Dan Rubenstein. Provisioning Servers in the Application Tier for E-Commerce Systems. ACM Transactions on Internet Technology, 7(1), 2007, 57-66.

[19] Anthony Sulistio, Kyong Hoon Kim, and Rajkumar Buyya, Managing Cancellations and No-shows of Reservations with Overbooking to Increase Resource Revenue, In Proc. of IEEE International Symposium on Cluster Computing and the Grid , 2008, 267-276.