

Dynamic Provisioning Modeling for Virtualized Multi-tier Applications in Cloud Data Center

Jing Bi^{1,3}, Zhiliang Zhu^{1,2}, Ruixiong Tian³, Qingbo Wang³

School of Information Science and Engineering¹, College of Software²,

Northeastern University, Shenyang 110004, P.R. China

IBM China Research Lab³, Beijing 100094, P.R. China

E-mail: neubijing@gmail.com, zzl@mail.neu.edu.cn, {tianruix, wangqbo}@cn.ibm.com

Abstract—Dynamic provisioning is a useful technique for handling the virtualized multi-tier applications in cloud environment. Understanding the performance of virtualized multi-tier applications is crucial for efficient cloud infrastructure management. In this paper, we present a novel dynamic provisioning technique for a cluster-based virtualized multi-tier application that employ a flexible hybrid queueing model to determine the number of virtual machines at each tier in a virtualized application. We present a cloud data center based on virtual machine to optimize resources provisioning. Using simulation experiments of three-tier application, we adopt an optimization model to minimize the total number of virtual machines while satisfying the customer average response time constraint and the request arrival rate constraint. Our experiments show that cloud data center resources can be allocated accurately with these techniques, and the extra cost can be effectively reduced.

Keywords—cloud computing; resource provisioning; virtualized application; performance modeling

I. INTRODUCTION

Virtualization technologies have facilitated the realization of cloud computing services [1]. Cloud computing [2, 3] includes three kinds of computing capacities as a service in different abstraction levels for different business purposes, such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Here, we consider only IaaS, which aims to provide computing resources or storage as a service to customers. One major player in cloud computing is Amazon's Elastic Compute Cloud (EC2), which comprises several data centers worldwide. Amazon EC2 lets customers deploy virtual machines (VMs) on-demand on Amazon's infrastructure and pay only for the computing, storage, and network resources they use. While a number of recent papers address virtualization of enterprise applications, such as resource virtualization [4, 5], on-demand resource provisioning management based on virtual machines [6, 7], and QoS management of virtual machine [8]. These works lead to improvements in the performance of virtualization and resource utilizations. Since IT infrastructure customer requirements for cloud infrastructure services are varied, infrastructure providers have to ensure that they can be flexible in their service delivery while keeping the infrastructure service customers efficiently increasing

ability of commodity hardware to run applications within VMs. VMs allow both the isolation of applications from the underlying hardware and other VMs, and the customization of the infrastructure resource to meet the requirements of the IT infrastructure customer.

However, the application of virtualization technologies shows its advantages for further challenges, such as the intelligent allocation of VM resources for managing computing resource demands of the infrastructure customers. In addition, enterprise IT infrastructure customers with virtualized applications require lesser resource cost, and thus save resource by distributing workload requests to virtualized multi-tier applications in cloud environment. This creates the need for establishing a computing atmosphere for dynamically provisioning cloud resources from multi-tier domains within and across enterprises. Furthermore, there are many open challenges involved in on-demand resources dynamic provisioning for cloud data centers, such as the CPU, memory, disk and network bandwidth, to be partitioned among the resident VMs, and optimal configuration for VMs.

Therefore, this paper considers the current trends in the environment of cloud computing and presents relationships between performance and resources provisioning of virtualized applications. In order to address the above challenge, this paper proposes a hybrid model for dynamic resource provisioning in VM-based cloud data center, which can be based on combination of an M/M/c model and multiple M/M/1 queueing models methods. Such model is important for the following reasons: (1) **Resource provisioning**, a number of cluster VMs are dedicated to virtualized multi-tier application and the model must determine how many VMs are allocated to virtualized multi-tier application to satisfy the requirement of given response time for customers, that is, the model can not only meet the needs of customers, but also cause little waste of resources. (2) **VM configuration**, which enables various configuration parameters of the VM to be determined for a certain performance goal. Thus we can determine the most effective resource utilization by VM optimal configuration.

The main contributions of the paper include: (1) We develop a hybrid model composed of an M/M/c model and multiple M/M/1 models to provision computing resources for virtualized application; (2) Based on the proposed hybrid model, an optimization model to minimize the total

number of virtual machines for computing resources is developed, and the proposed optimization model is verified with the three-tier virtualized application of handling dynamic workloads through simulation.

The rest of this paper is organized as follows. In Section 2, we describe the infrastructure management of a cloud environment for virtualized multi-tier applications. Section 3 presents the analytic models used to solve the defined problem. Section 4 demonstrates the results of prototype experiments. In Section 5, we review some related work in the area of dynamic and scalable resources provisioning. Concluding remarks and discussion about future work are given in Section 6.

II. THE DYNAMIC VIRTUAL MACHINES IN CLOUD DATA CENTER

In order to dynamically provisioning resources for virtualized multi-tier application execution environments (VAEEs) of different customers, the most common approaches are based on self-managing techniques [9], such as Monitor, Analyze, Plan, and Execute (MAPE) control loops architecture is needed. The goal is to meet the virtualized application requirements while adapting IT architecture to workload variations. Usually, each request requires the execution of virtualized application allocated on the VM of each physical tier. A cloud data center enables multiple virtualized applications may be increased when workload increases and reduced when workload reduces. This dynamic resource provision allows flexible response time in a VAEE where peak workload is much greater than the normal steady state. Figure 1 provides a high-level dynamic resource provision architecture for cloud data center, which shows relationships between computational resources pool and self-management community.

Computational Resources Pool contains physical resources and virtualized resources. Plenty of VMs hold several VAEEs sharing the capacity of physical resources and can isolate multiple applications from the underlying hardware. VMs of each tier of a virtualized application may correspond to a physical machine. Computational resources pool delegates self-management community for satisfying the requirement goal of the customer to automatically allocate sufficient resources to the each tier of virtualized application. **Self-management community** means mechanisms to automate the VMs of configuring and tuning the virtualized multi-tier application so as to maintain the response time requirements of the different customers. It generates result of run-time provisioning for cloud data center. It includes four components as follows:

- **Monitor:** collects the workload and the performance metric of all running VAEEs, such as the request arrival rate, the average service time, and the CPU utilization, etc.
- **Analyzer:** receives and analyzes the measurements from the monitor to estimate the future workload. It also receives the response times of different customers.

- **Resource Scheduler:** sets up performance analytic models for each tier of the VAEE, and uses its optimizer with the optimization model to determine resource provisioning according to these workload estimates and response time constrains of different customer such that the resource requirements of the overall VAEE is minimized.
- **Virtualized application Executor:** assigns the VM configuration, and then runs the VAEEs to satisfy the resource requirements of the different customers according to the optimized decision.

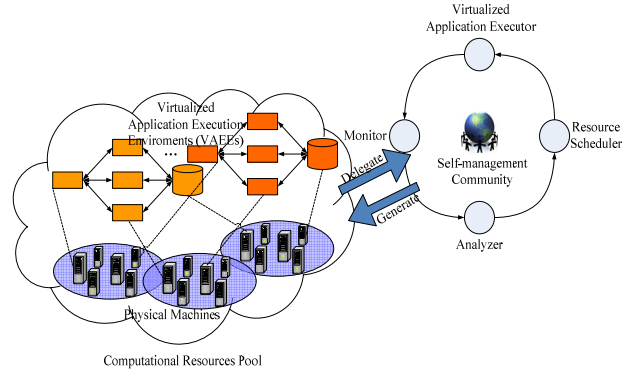


Figure. 1 The dynamic resource provisioning of cloud data center

This paper focuses on the design of resource scheduler for virtualized multi-tier applications. The goal is to minimize the using of resources under a workload while satisfying different customer for the constraints of average response time.

III. VIRTUALIZED MULTI-TIER APPLICATION QUEUEING MODEL

In this section, we present a hybrid queueing model for a virtualized multi-tier application, and then define a non-linear constrained optimization problem for dynamic resource provisioning. Moreover, the optimal model of relationships between performance and resources provisioning is used to maximize resources utilization according to the response time of customer requirement.

A. Analytic Performance Model

A virtualized multi-tier application in cloud computing environment is deployed on multiple virtual machines (VMs), and each tier provides certain functionality to its preceding tier. Here we consider an online e-commerce application that consists of n tiers, denoted by T_1, T_2, \dots, T_n . We assume that there are c parallel identical VMs in the each tier of VAEE, and the requests of all the arriving sessions enter into a common queue maintained by the on-demand scheduler (ODS), waiting for available resources in the first tier. The ODS schedules these requests of each session. Note that in our system, scheduler decisions are made only for the first tier of the virtualized application. Once scheduled, a request is decided processing at the VMs of the first tier within the virtualized application.

Dispatcher of other tiers is used for collecting requests processed in the pre-tier and distributing them to multiple parallel VMs queuing models of that tier to execute. Multiple VMs queuing models of other tiers are responsible for dynamic resource provisioning by the requests of that tier. Each tier is assumed to employ a perfect load-balancing element for a virtualized application that is responsible for processing requests at that tier, and each request is forwarded to its succeeding tier for further processing. Once the result is processed by the final tier T_n , the results are sent back by each tier in the reverse order until it reaches T_1 , which then sends the results to the customer. In more complex processing scenarios, each request at tier T_i , can trigger zero or multiple requests to tier T_{i+1} . For example, a static web page request is processed by the Web tier entirely and will not be forwarded to the following tiers. On the other hand, a keyword search at a Web tier may trigger multiple requests to the next tier. We also assume that database tier with a shared-everything architecture [10], which can be clustered and replicated on-demand. Through modeling all the tiers and their interactions, our multi-tier model allows us to integrate decisions for the first tier into scheduler and other tiers into each VM. Therefore, the amount of concurrency VMs may be determined by the number of concurrent requests that tier supports according to our model. In order to capture the virtualized multi-tier application for dynamic resources provisioning, we define that our model is a hybrid analytical model, which accords with real environment. This not only saves the network transmission time but also improves the processing efficiency of the request.

B. Open Queueing Model of Virtualized multi-tier application

The workload on the virtualized multi-tier application is typically session-based customer, where a customer session consists of a succession of requests. At a time, multiple concurrent customer requests interact with the virtualized multi-tier application. In order to capture the multiple concurrent requests of customer sessions, we model the serving system as an M/M/c queueing system to the first tier, and other tiers can be modeled as multiple M/M/1 queueing systems. The requests of each tier are serviced in a first-come-first-served (FCFS) order, as shown in figure 2.

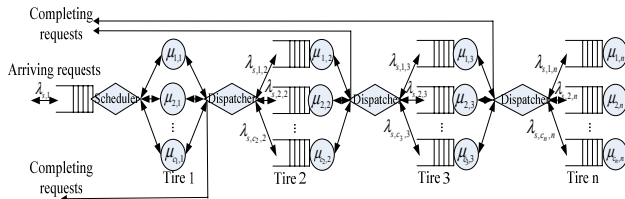


Figure. 2 Open queueing model for virtualized multi-tier application

Figure 2 shows the open virtualized multi-tier application queueing models. The arrival of requests at the each tier for a virtualized application is assumed to be described by a Poisson distribution [11, 12], i.e. inter-arrival time

between requests subjects to the negative exponential distribution. This assumption is validated by analyzing traces taken from an e-commerce multi-tier application website. Besides, the average service time of requests is also considered the negative exponential distribution.

The first step in solving our model is to determine the capacity of multiple VMs for each tier in terms of the request rate they can handle. Given the capacity of each tier VMs, the next step computes the number of VMs required at each tier to satisfy the requirement of customer response time. Here, let R be the desired end-to-end response time for a virtualized multi-tier application. Denote $T_r(1), T_r(2), \dots, T_r(n)$ as the per-tier end-to-end response times,

such that $\sum_{i=1}^n T_r(i) = R$, these values are obtained by given

analytical model. We let $\lambda_{s,i}$ be the aggregate request arrival rate to the tier i for a customer s , such that $\forall s \in [1, m], \forall i \in [1, n]$. Note that a session in our model corresponds to a customer. Assume that requests of forwarded tier will not arrival the following tiers, such that requests are reduced at the following tiers, the probability of relative request arrival rate can be denoted by

$\lambda_{s,i}/\lambda_{s,(i-1)} = \alpha_{i-1} \leq 1, \forall i \in [2, n]$, such that, $\lambda_{s,2}/\lambda_{s,1} = \alpha_1, \lambda_{s,3}/\lambda_{s,2} = \alpha_2, \dots, \lambda_{s,n}/\lambda_{s,(n-1)} = \alpha_{n-1}$. The parameters $\alpha_1, \alpha_2, \dots, \alpha_{n-1}$ are derived using online measurements.

Each request brings with a certain amount of work for the VM to do. We assume that the per-tier multiple VMs have equal processing capability, such that $\mu_{1,i} = \mu_{2,i} = \dots = \mu_{c_i,i}$.

The sum of the service rates of per-tier multiple VMs is $\sum_{j=1}^{c_i} \mu_{j,i}, \forall i \in [1, n]$, the service rate of per-tier single VM

is $\sum_{j=1}^{c_i} \mu_{j,i} / c_i, \forall i \in [1, n]$. The sum of the service rates of

virtualized multi-tier application is $\sum_{i=1}^n u_i(j)$, where $u_i(j)$

$$= \mu_{1,i} + \mu_{2,i} + \dots + \mu_{j,i}.$$

In our model systems, per-tier service times are assumed to be drawn from a known fixed distribution. Assume that the per-tier utilization of VM for virtualized multi-tier application is $\rho_i = \lambda_{s,i}/u_i(j) < 1, 0 < \rho_i < 1$, where ρ_i correspond to the utilization of the busiest resource (e.g. CPU, disk, or network) for the tier i .

Our model can also express useful system metrics like average request arrival rates and throughputs at multiple VMs, in terms of the distributions of their per-tier inter-arrival and service times. Therefore, our model enables us to capture the behavior of various tiers such as HTTP, J2EE, and database VMs. Here, the behavior of first tier can be modeled an M/M/c system, using Little's Law [13], which derives the end-to-end average respond time for the first tier, as follows:

$$T_r(i) = \frac{(\lambda_{s,i})^{c_i-1} \cdot (\rho_i + c_i - c_i \rho_i)}{\prod_{j=1}^{c_i} \mu_j(j)(1-\rho_i)^2} \cdot p_{0,i} + \frac{1}{\lambda_{s,i}} \cdot \sum_{k=0}^{c_i-1} (k \cdot p_{k,i}) \quad (1)$$

where $\rho_i = \lambda_{s,i}/\mu_i(j) < 1$ ($\forall i=1, \forall s \in [1,m]$) is the first tier utilization of VMs for virtualized multi-tier application. $p_{0,i}$ is system state probability that a request leaves i tier for the virtualized application just after completing service.

The models of other tiers divide the coming requests into multiple M/M/1 models by some rules, and send them to different VMs respectively for response. It's assumed that the requests of customer s arrive at VM j for the tier i with arrival rate $\lambda_{s,j,i}$ ($2 \leq i \leq n, 1 \leq j \leq c_i$), and the end-to-end average respond time for the other tiers can be derived as follows:

$$T_r(i) = \frac{1}{\mu_{1,i} - \lambda_{s,1,i}} = \frac{1}{\mu_{2,i} - \lambda_{s,2,i}} = \dots = \frac{1}{\mu_{c_i,i} - \lambda_{s,c_i,i}} \quad (2)$$

where these rules mean that the same response time can be ensured no matter which VM is allocated for the next request. $\sum_{j=1}^{c_i} \lambda_{s,j,i} = \lambda_{s,i}$, ($\forall i \in [2,n], \forall s \in [1,m]$) is request arrival rate for the tier i which is equally distributed to each VM for that tier. $\mu_{1,i} = \mu_{2,i} = \dots = \mu_{c_i,i} = \mu$ is equal capacity of each VM for tier i .

Observe that our model can handle virtualized multi-tier applications with an arbitrary number of tiers, since the complex task of modeling a virtualized multi-tier application is reduced to modeling each tier. Assume that VMs in each tier are homogeneous and load-balanced. Every VAAE has VMs number c_i , which is a function of the performance metrics for each tier of that virtualized multi-tier application, and thus

$$c_i = f_i(\lambda_{s,i}, \mu_{1,i}, \dots, \mu_{c_i,i})$$

The global function C_g is a self-optimization function of each virtualized multi-tier application. Thus, our model of resource optimization would then be to minimize total weighted VMs of the system, which can be formulated as follows:

$$\min\{C_g = f(\lambda_{s,1}, \mu_{1,1}, \dots, \mu_{c_1,1}; \dots; \lambda_{s,i}, \mu_{1,i}, \dots, \mu_{c_i,i}; \dots; \lambda_{s,n}, \mu_{1,n}, \dots, \mu_{c_n,n})\} \quad (3)$$

$$s.t. \quad \sum_{i=1}^n T_r(i) \leq R_{0,s} \quad (4)$$

$$\sum_{j=1}^{c_i} \mu_{j,i} > \lambda_{s,i} \quad \forall i \in [1,n], s \in [1,m] \quad (5)$$

Given the request rate, service rate, and end-to-end response time for a virtualized multi-tier application, our objective is to determine how many VMs to allocate such

that virtualized multi-tier application can service all incoming requests with a given response time $R_{0,s}$. The output of the model would be to minimize the total number of VMs for a virtualized multi-tier application, denoted by C_g , such that meet to handle a request rate $\lambda_{s,i}$. Note that the first constraint given by (4) requires that the average response time for each tier cannot be greater than a certain response time ($R_{0,s}$, such as 0.5 second). Response time of customer s requirement $R_{0,s}$ is specified by enterprise IT customer's contract. The second constraint describes a condition, $\sum_{j=1}^{c_i} \mu_{j,i} > \lambda_{s,i}$, in (5), necessary for average utili-

zation in the VMs, which can not occur the state of infinite queue. For satisfy with the constraints, and then we adjust the capacity of all tiers to these values, resulting in an immediate increase for effective capacity.

In order to compute the number of VMs, the model requires several input parameters. In practice, these parameters can be estimated through online monitoring virtualized application. Therefore, we analyze the general trends at each VM node for each tier in our cloud environment, such as the Apache Web server with VMs. The target application used in our experiments is that auction system commonly used a benchmark for multi-tier enterprise applications.

The main notations used throughout this paper are summarized in Table 1 for clarity.

TABLE I. SUMMARY OF NOTATIONS

| Symbol | Description |
|-----------------|---|
| n | Number of virtualized application tiers |
| m | Number of customers |
| c_i | Number of VMs for tier i ($1 \leq i \leq n$) |
| R | End-to-end response time for a virtualized multi-tier application (sec) |
| $R_{0,s}$ | Response time (sec) of customer s requirement for virtualized application |
| α_{i-1} | Relative probability of request arrival rate |
| $\lambda_{s,i}$ | Request arrival rate (req/s) of customer s for tier i ($1 \leq s \leq m$) |
| $\mu_{j,i}$ | Service rate (req/s) of server j for tier i ($1 \leq j \leq c_i$) |
| $T_r(i)$ | End-to-end response time for tier i (sec) |
| C_g | Minimized number of VMs for a virtualized multi-tier application |

IV. EXPERIMENTAL EVALUATION

In this section we present our experimental results on the efficiency of our autonomic resource provisioning technique for optimizing the number of VMs in the cloud environment. The results show that under fine-grained resource provisioning, the provider's achieved revenues can be maximized while the customer's operational cost is reduced as much as possible. The following experiments are for the validation of the model.

A. Experimental Setup

We establish a prototype system of cloud environment, such that each of the server nodes was run on two Intel Pentium 4 2.66GHz processors with 2GB RAM. Processing capacity of each server is equal in cloud data center. The on-demand scheduler was run on a machine with 4 Inter Xeon 3.00GHz processor with 3GB RAM. VAAE Host ran the open-source version of the Xen 3.0.3 to build the virtualization environment. All machines were conducted on a Linux kernel 2.6.16.29 cluster interconnected by a Gigabit Ethernet (GigE) switch. Each Linux was installed as a guest OS in each domain of Xen. Note that because Xen places device drivers for physical devices into a separate guest virtual machine called domain 0, all incoming and outgoing network communication passes through an extra node, and incurs additional latency. Moreover, since this node potentially shares the CPU with the other VMs, this latency depends on both the utilization of the node and the number of messages. Therefore, we explicitly model and measure parameters for this VM monitor delay.

We present profiling result on one open-source multi-tier application service based on Enterprise Java Beans (EJB) in our experimental study: the *RUBiS* online auction benchmark [14], running on VMs hosted on different servers. *RUBiS* implements the core functionality of an auction site similar to eBay, including 26 interactions that can be performed from a client’s Web browser. It follows the three-tier application. The front tier was based on the Apache 2.2 Web server. The middle tier was based on Java servlets that implement the application logic with an embedded Tomcat 5.0.28 as the servlets container. Finally, the database tier was based on MySQL 4.0. To isolate performance interference, we restrict the management domain (domain 0) to use one CPU and VMs to use the other CPU. Table 2 shows the values for various parameters in our simulation experiments.

TABLE II. WORKLOAD CHARACTERISTICS FOR RUBIS

| Parameter | Web Tier | App Tier | DB Tier |
|----------------|-----------|-----------|-----------|
| $T_r(i)$ | 0.08 sec | 0.4 sec | 0.32 sec |
| $\mu_{j,i}$ | 250 req/s | 150 req/s | 100 req/s |
| α_{r-1} | - | 0.8 | 0.8 |
| $R_{0,s}$ | 0.8 sec | | |

Because our tested application is CPU-intensive, the only resource type we currently consider is CPU capacity and we assume that all resources are identical. We do not show the memory and disk I/O profiling results for brevity. The memory and disk I/O consumption for the virtualized application is relatively insignificant and they never become the bottleneck resource in our test settings.

B. Effectiveness of Multi-Tier Model

In the following experiments, we evaluate our dynamic resource provisioning technique for virtualized multi-tier

appliances. We built a time-driven optimizer that models the system as a hybrid queue with two different queuing models, and is fed with workload traces from virtualized applications. For the self-optimization strategies, the optimizer is coupled to an optimization model solver, which is called at the each tier interval to calculate the number c_i of resource provisioning according to the end-to-end response time $T_r(i)$ for tier i , for the next interval.

During each interval, per-request response time as well as per-tier throughput and CPU utilization are collected and used to compute the minimized number of VMs for each tier of virtualized applications with different workloads. Moreover, our optimizer employs a fair admission mechanism, which accepts a request with probability $\lambda_{s,i}/\lambda_{s,(i-1)}$. Thus, the assumption of Poisson arrivals holds for the accepted requests in order to describe virtualized applications.

First, the *RUBiS* application is provided for each server with embedded VMs. Here, each tier employs its own provisioning technique. System parameter values are shown in Table 2. Our technique is aware of the demands at each tier and can take idiosyncrasies such as optimization model into account, as shown in Figure 3, where arrival rates vary from 0 to 2000 requests per second. The customer s given response time is 0.8 seconds for the *RUBiS* application, and service rate is 250, 150, and 100 requests per second for Web, App, and DB tiers, respectively. For App and DB tiers, the request probability is 0.8 and 0.8, respectively, for the hybrid analytic model. In Fig.3, the minimized VM number for each tier by the response time constraint is presented, which is computed with our model and optimal approach.

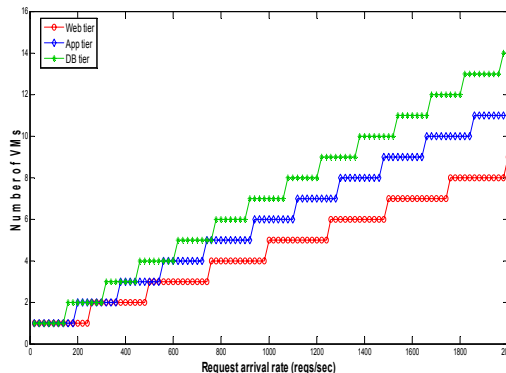


Figure. 3 Validation results on the number of VMs at various request arrival rates

Next, we repeat this experiment to predict the throughput of virtualized application, as shown in Figure 4. It shows validation results on the overall system throughput for *RUBiS*, the application throughput continues to increase with the increasing workload. We measure the rate of successfully completed requests at different requests rate. In our experiment, a request is counted successfully only if it responses within 0.8 seconds for the customer requirement. The request probability is 0.8 for App tier and

for DB tier. The result shows that the system throughput can be accurately predicted with our model.

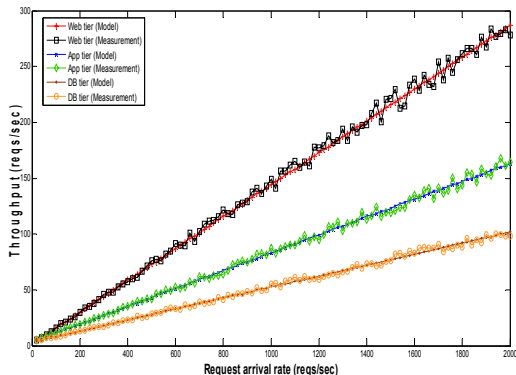


Figure. 4 Validation results on system throughput

Since the virtualized application throughput in our model is derived from resource usage at each tier, we further examine the accuracy of per-tier resource usage prediction using the same parameter values in Table 2. Figure 5 presents validation results on the CPU utilization at Web, App and DB tiers, respectively. They compare the predicted CPU utilization to the measured CPU utilization for the three tiers with the workload increasing. Web, App and DB tiers were running on their own virtual machine with average 53.82%, 62.25%, and 62.51% CPU utilization, respectively, which is close to the optimal solution 60%. CPU utilization of domain 0 was average 22% for each tier of virtualized application. Overall, these figures demonstrate that the model is reasonably accurate and can effectively use CPU resource.

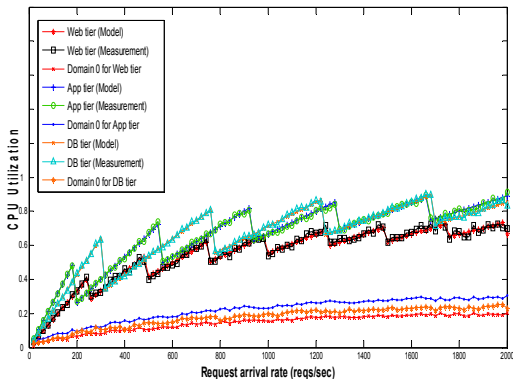


Figure. 5 Comparison of models vs. experimental results

Therefore, the minimized number of VMs as well as the maximized CPU resource utilization can be achieved with our method by dynamic resource provisioning technique, and then we can keep the high global utility.

V. RELATED WORK

Previous literature on issues related to managing resources in multi-tier applications of data centers. In this section we describe some prior work related to this paper as follows.

Some papers have considered the provisioning of resources at finer granularity of resources. Urgaonkar et al. [15] presented an analytical model for multi-tier Internet applications, which is general to capture various characteristics of an arbitrary number of heterogeneous tiers. Then, the model was applied to dynamic resource provisioning. Ardagna et al. [16] proposed a provisioning controller for multi-tier data center which maximize profits using a cost model, and developed a heuristic solution. The limitation is that they did not distinguish servers in different tiers, but allocated physical resources instead of virtual machines. At the same time, they adopted a closed queueing network performance model for the autonomic system. Overall, the above approaches are commonly based on the provisioning of identical servers as unit, while our work is different in that we adopt full virtual machines based on an open queueing network model, which supports fine-grained sharing of the physical infrastructure as well as guarantees the performance isolation of different virtualized application environments by deploying them on separate virtual machines.

Other research efforts have focused on the modeling of multi-tier application environments. Urgaonkar et al. [17] proposed a dynamic capacity provisioning model for multi-tier Internet applications, which determine how much of the resources to provisioning to each tier of the application, and a combination of predictive and reactive methods that determine when to provision these resources, both at large and small time scales. Chen et al. [18] proposed a closed-system model of multi-tier business applications, and based on mean value analysis (MVA) algorithm to predicate performance of multi-tier applications. Kamra et al. [19] presented a single queue model for all tiers and based on control-theoretic approach for admission control in multi-tier Web sites that both prevented overload and enforced absolute client response times, while still maintaining high throughput under load. Jung et al. [20] proposed a generating adaptation for multi-tier applications in virtualized consolidated server environments. It provides dynamic management method and optimizes offline resources to generate suitable configurations by evaluating a model consisting of multi-tier M/M/n queues. However, the primary difference is that we have established sophisticated models different from traditional analytic models which adopt inaccurate MVA or single queueing model. Here, we can conclude that a hybrid model, which is an M/M/c queueing model combined with multiple M/M/1 queueing models can be adopted in this paper, which can be achieved more accurate provisioning for virtualized multi-tier applications than other models.

Another area of related researches has focused on optimization problems arising in multi-tier applications. For example, Zhang et al. [21] presented a nonlinear integer optimization model for determining the number of machines at each tier in a multi-tier server network. Similar to ours, they profile the computing resource of data center in physical servers' environment while we profile the computing resource of cloud environment in virtual

machines environment. As a result, our approach can support a more fine grained resource provisioning and management for a virtualized application in cloud environment. Additionally, their approach uses a simply open queueing network model at each server, which is less accurate than hybrid queueing models we used. Cunha et al. [22] presented a new self-adaptive capacity management framework for multi-tier virtualized environments. It executes periodically and reassigns resources by evaluating a model consisting of multi-tier M/M/1 queues and solves an optimization problem. Instead, in our work we consider that the domain 0 of Xen potentially shares the CPU with the other VMs, and this latency depends on both the utilization of the node and the number of messages. Moreover, we propose an optimal method for VMs, which can compute effective utilization for CPU of each tier of virtualized application. Therefore, our work has presented models close to reality.

VI. CONCLUSION AND FUTURE WORK

In this paper, it is argued that dynamic provisioning of virtualized multi-tier applications raises new challenges not addressed by prior work on provisioning technique for cloud environment. We presented an optimal autonomic virtual machine provisioning architecture for cloud data center. We proposed a novel dynamic provisioning technique, which was a hybrid model for a virtualized multi-tier application in cloud data center. A constrained non-linear optimization model is employed to minimize the total number of VMs for the requirement of customer. Hence the efficiency and flexibility for resource provisioning were improved in cloud environment. We evaluated and contrasted the performance of three tier virtualized applications through simulation experiments. Results have shown that under fine-grained resource provisioning, computing resources are optimized utilization. Moreover, our technique is also demonstrated that by optimizing provisioning the overall performance could be further enhanced while maintaining average response time targets.

Our work can be improved in a number of ways. First, we further integrate load prediction method technique to fit our workload characteristics. Second, we will focus on expanding the utility analytic model to fit cloud environments with heterogeneous servers produced by different manufacturers. Third, we adopt Service Level Agreement (SLA) based negotiation of prioritized applications to determine the costs and penalties by the achieved performance level. If the entire request cannot be satisfied, some virtualized applications will be affected by their increased execution time, increased waiting time, or increased rejection rate.

ACKNOWLEDGMENT

This work was supported in part by the IBM Ph.D. Fellowship, and the National Natural Science Foundation of China under Grant 60872040.

REFERENCES

- [1] D. Reed, I. Pratt, and P. Menage, et al, "Xenoservers: Accountable execution of untrusted programs", The Seventh Workshop on Hot Topics in Operating Systems, Rio Rico, Arizona, 1999.
- [2] M. Armbrust, A. Fox, and R. Griffith, et al, "Above the clouds: A Berkeley view of cloud computing", Technical Report No. UCB/EECS-2009-28, University of California Berkley, USA, Feb. 10, 2009.
- [3] R. Buyya, C.S. Yeo, and S. Venugopal, et al, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Future generation computer systems, Elsevier science, Amsterdam, the Netherlands, 2009, 25(6), pp. 599-616.
- [4] D. Gupta, S. Lee, and M. Vrable, et al, "Difference engine: harnessing memory redundancy in virtual machines", The 8th USENIX Symposium on Operating Systems Design and Implementation, 2008, pp. 309-322.
- [5] P. Barham, B. Dragovic, and K. Fraser, et al, "Xen and the art of virtualization", Proceedings of the 19th ACM Symposium on Operating Systems Principles, Bolton Landing, NY, USA, 2003, pp. 164-177.
- [6] Y. Song, Y. Li, and H. Wang, et al, "A service-oriented priority-based resource scheduling scheme for virtualized utility computing", Proceedings of the 9th IEEE International Symposium on Cluster Computing and the Grid, 2009, pp. 148-155.
- [7] J. Zhang, M. Yousif, and R. Carpenter, et al, "Application resource demand phase analysis and prediction in support of dynamic resource provisioning", Proceedings of the 4th International Conference on Autonomic Computing, 2007.
- [8] X.Y. Wang, Z.H. Du, and Y.N. Chen, et al, "Virtualization based autonomic resource management for multi-tier Web applications in shared data center", The Journal of Systems and Software, 2008, 81(9), pp. 1591-1608.
- [9] S.R. White, J.E. Hanson, and I. Whalley, et al, "An architectural approach to autonomic computing", Proceedings of the International Conference on Autonomic Computing, 2004.
- [10] Oracle9i. 2005. <http://www.oracle.com/technology/products/oracle9i>.
- [11] D.A. Menascé, M.N. Bennani, "Autonomic virtualized environments", Proceedings of IEEE International Conference on Autonomic and Autonomous Systems, 2006, pp. 28-37.
- [12] R.P. Doyle, J.S. Chase, and O.M. Asad, et al, "Model-based resource provisioning in a web service utility", Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems, 2003.
- [13] J. McKenna, "A Generalization of Little's Law to moments of queue lengths and waiting times in closed, product form queueing networks", Journal of Applied Probability, 1989, 26, pp. 121-133.
- [14] E. Cecchet, J. Marguerite, and W. Zwaenepoel, "Performance and scalability of EJB applications", Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, 2002, pp. 246-261.
- [15] B. Urgaonkar, G. Pacifici, and P. Shenoy, et al, "An analytical model for multi-tier Internet services and its applications", Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, 2005, pp. 291-302.
- [16] D. Ardagna, M. Trubian, and L. Zhang, "SLA based profit optimization in multi-tier systems", Proceedings of the 4th IEEE International Symposium on Network Computing and Applications, 2005, pp. 263-266.
- [17] B. Urgaonkar, P. Shenoy, and A. Chandra, et al, "Agile dynamic provisioning of multi-tier Internet application", ACM Transactions on Autonomous and Adaptive Systems, 2008, 3(1), pp. 1-39.
- [18] Y. Chen, S. Iyer, and X. Liu, et al, "SLA decomposition: Translating service level objectives to system level thresholds", Proceedings of the 4th International Conference on Autonomic Computing, 2007.

- [19] A. Kamra, V. Misra, and E. Nahum, "Yaksha: A self-tuning controller for managing the performance of 3-tiered web sites", Proceedings of International Workshop on Quality of Service, 2004, pp. 47-58.
- [20] G. Jung, K.R. Joshi, and M.A. Hiltunen, et al, "Generating adaptation policies for multi-tier applications in consolidated server environments", Proceedings of the 5th International Conference on Autonomic Computing, 2008, pp. 23-32.
- [21] A. Zhang, P. Santos, and D. Beyer, et al, "Optimal server resource allocation using an open queueing network model of response time", HP Labs Technical Report, HPL-2002-301.
- [22] I. Cunha, J. Almeida, and V. Almeida, et al, "Self-adaptive capacity management for multi-tier virtualized environments", Proceedings of the 10th International Symposium on Integrated Network Management, 2007, pp. 129-138.