Provisioning Virtual Resources Adaptively in Elastic Compute Cloud Platforms*

Fan Zhang[1], Junwei Cao[2,3], Hong Cai[4], James J. Mulcahy[5], Cheng Wu[1,3]
[1]*National CIMS Engineering and Research Center, Department of Automation Tsinghua University, Beijing 100084, P. R. China*
[2]*Research Institute of Information Technology, Tsinghua University, Beijing 100084, P. R. China*
[3]*Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, P. R. China*
[4]*IBM China Software Development Lab*
[5]*Florida Atlantic University, Boca Raton, Florida, USA*
Corresponding email: jcao@tsinghua.edu.cn

ABSTRACT:
Provisioning Virtual machines on demand is significant in elastic compute cloud for reliable service delivery. The importance and major difficulty lies in satisfying the conflicting objectives of satisfying contracted service level agreement while lowering used resource costs. In this paper, we propose a mathematical multi-tier framework for adaptive virtual resource allocation problem. The framework captures the performance of the virtualized cloud platform gracefully. We first use simulations to derive virtual resource allocation policies, and later use real benchmarking applications, to verify the effectiveness of this framework. Experimental results show that the model can be simply and effectively used to satisfy the response time requirement as well as lowering the cost of using the virtual machine resources.

KEY WORDS:

Dynamic resource provision; multi-tier applications; virtual machines, and virtualized cloud platform.

Introduction

Research background

Cloud computing allocates Virtual Cluster (VC) resources in Infrastructure as a Service (IaaS), Platform as a Service: PaaS, and Software as a Service: SaaS (Turner, Budgen, Brereton, 2003) applications. A virtual cluster (Emeneker, et al, 2006) is an organized heterogeneous Virtual Machine (VM) hosting platform, which is used in cloud platforms to provide elastic and reliable services. Examples of virtual machine software include VMware, KVM and Xen. The VMs are allocated on demand, which satisfies user requirements for response time while lowering the cost of using the resources.

Dynamic resource provisioning (Li, Tirupati, 1995) which has been widely used in internet hosting platforms, has proven to be useful in handling multiple time-scale workloads. However, this kind of dynamic provisioning in previous work has been more commonly based upon physical resource allocation in large scale server farms, which is not flexible enough for the effective delivering of services.

Manuscript submitted to International Journal of Web Service Research on October 23, 2010. This work is performed when Fan Zhang is an intern student in IBM China

Development Lab supported by 2010-2011 IBM Ph.D. Fellowship.
Unlike other resources, VMs are flexibly deployed on physical machines, which can be automatically generated for different applications. For example, some scientific applications such as gene expression and transferring demand a lot of CPU resources, while some other real-time transaction applications and interactive online games are memory and network-intensive. Though traditional physical capacity provisioning has long been used, over-provisioning or under-provisioning has been a common difficulty for most resource vendors.

The advent of cloud computing, which takes full advantage of virtualization technology, potentially solves this difficulty. We propose a queuing and multi-tier network model, in which virtual clusters are deployed on each tier for service delivering. The flexibility of our proposed adaptive resource provisioning is threefold:

(1) The flexibility of allocating virtual resources: we can dynamically create and/or destroy a VM from a resource pool using a simple command with very low overhead. This advantage can be used to handle the problems in resource over-provisioning and/or under-provisioning.

(2) The flexibility of handling data or communication locality: we can easily deploy those VMs that have the most frequent communication or interaction in such a way that they are physically or logically close to each other. This advantage saves the communication cost noticeably.

(3) The flexibility of migrating services: virtual resources used to provide reliable services can use service migration to re-organize service hosting platforms in order to maximize the utilization of resources, which can dramatically improve operational costs.

Given the above advantages and benefits from the use of adaptive resource provisioning, we propose the design of a virtualized resource allocation framework using the cloud platform, which allocates VMs on demand in order to provide services, as well as minimizing the cost of using those virtual resources.

Motivation

One of the major difficulties of virtual resource allocation is how to provide proper policies at different times to satisfy different user demands. This difficulty lies in the unpredictable execution time, variable operational costs and frequently fluctuating workloads (Wu and Hwang, et al, 2009). These aspects require a proper model to capture the characteristics of complex systems.

Model-View-Control (MVC), an architectural design pattern widely used in internet hosting platforms, is also supported in Windows Azure and other cloud platforms. This design pattern is unique to the new proposed paradigm in that full virtualization is employed in order to achieve higher reliability and flexibility. Based on a typical MVC architecture, we model our application as a multi-tier queuing network to leverage virtual resource provisioning as shown in Fig. 1.

Each tier can be modeled as a set of VMs. The inputs of this model are the *cost* of using each VM at each tier, the *transferring probability* that a request should be forwarded to the next or previous tier, and most importantly, the *inter-arrival time* of the workloads. The output is the *optimal number* of VMs allocated for each tier needed to achieve a reasonable response time and minimized total resource costs. We analyze the relationship of these parameters and show its derivation based on queuing theory (Kleinrock, 1976).

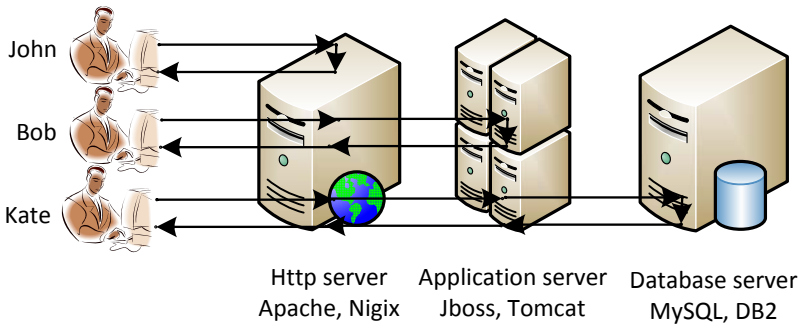

Fig 1. A typical three tiers of http, application and database server internet hosting platform. Different requests are used and they traverse different number of tiers. The requests of John traverse http server only. Typical requests of this type are static web page retrieving. The requests from Bob traverse http and application server. Typical example is invoking a simple Java servlet, such as inquiring current session number. The requests from Kate traverse all the servers. One example is retrieving the unread mails.

Contributions of this work

In this paper, we focus on the mathematical formation and optimization of the model, and show how the above goals can be satisfied in a simple yet applicable way. The major contributions of this work are described as follows:

(1) Derives response time through mathematic formulation: we establish a direct relationship between a workload and average response time could be caused by this workload through a queuing model. The relationship can be used to determine the number of VMs we need to satisfy user demands while minimizing the cost of using these VMs.

(2) Extends traditional single size VM scheduling problems: our model includes two kinds of VM instances (fat and thin) that have a different number of CPUs and memory capacities to provide different computing capabilities. This extension is much more applicable in cloud platforms such as Amazon EC2 and Eucalyptus.

(3) Carries out benchmarks using the Rice University Bidding System (RUBiS) on IBM X3950 servers with different sizes of VMs, to verify the correctness and efficiency of our proposed model. This model is also tested with simulations to make comparisons.

The rest of this paper is organized as follows: We introduce the related work in section 2. System architecture including our virtualized cloud platforms and queuing network model are introduced in section 3. The virtual resource allocation framework is introduced in section 4. We describe experimental studies using both simulation and real benchmarks to verify our proposed method in section 5. We conclude this work and propose the future research direction in section 6.

Related work

Modeling a single tier of servers is first proposed (Slothouber, 1996), where four queues are adopted in a model web server. Two are used for the web servers and the remaining two are used for modeling communication networks. Modeling CPU, memory and disk bandwidth is suggested

for performance prediction in web servers (Chase, and Doyle, 2001). Modeling replicated single tier application using a G/G/1 queuing network is also discussed (Urgaonkar, and Shenoy, 2004). Using an M/M/1 queuing model to calculate response time of web requests are proposed (Levy, Nagarajarao, et al, 2003). The throughput of web servers using a queuing network combined with Markov chain is suggested (Menasce, 2003). These works discuss a single tier server solution, and later a multi-tier platforms. We differentiate our work by using our model to model multi-tier internet hosting platforms.

There is existing prior work using queuing networks to model multiple tier internet hosting platforms. An M/G/1/PS model is proposed (Villela, Pradhan, and Rubenstein, 2007) to provide servers on application tiers. A G/G/N queuing system with N servers for an e-commerce application model is discussed (Ranjan, Rolia, and Fu, 2002). An M/GI/1/PS model for an e-commerce application is also suggested (Kamra, Misra, and Nahum, 2004) to use a single queue to model multi-tier systems. The similarity of these works is that they only model the bottleneck aspects of the servers, which cannot capture the global characteristics of an entire system. In our model, the framework covers all the parts of a multi-tier system, including capturing bottleneck issues with the use of load balancing techniques.

A queuing network based analytical model is proposed to model multi-tier internet hosting platforms (Urgaonkar, Pacifici, Shenoy, ans Spreitzer, 2007). An enhanced model of this work can be used to deal with load balancing, handling concurrency limit, and multiple session classes at each tier. The improvement of this work is suggested (Urgaonkar, Shenoy, et al, 2007), where a combination of predictive and reactive methods is used to determine when to provide virtual resources. They also use queuing network model to determine how much virtual resource to provide. Our work is different from these works in that it not only takes the cost of using virtual resources into consideration, but also includes a framework to schedule thin and fat VMs upon user demand. These different-sized VM provisioning techniques are commonly seen in Eucalyptus, Amazon EC2, and other cloud platforms.

Recently, cloud computing (Armbrust, Fox, et al, 2009) has become a popular research topic in which computational resources are virtualized based on user demand. Buyya's work (Buyya, Yeo, et al, 2008) has a similar motivation with ours such as costs minimizing, and they use CloudSim (Calheiros, Ranjan, et al 2010) for simulating data centers. We differentiate our work using adaptive virtual machines provisioning on virtualized cloud computing platforms. Similar of our previous works are carried out under distributed platform (Zhang, et al, 2010) using adaptive mapreduce framework. The model used in this work is more directly related to cloud services.

Delivering reliable web service applications using petri-net or related techniques are fully investigated (Tan, et al, 2009, 2010). Quality of Services under web service configuration are discussed (Xiong, et al, 2008, 2009). All these techniques are ensuring methods in web service researches. Our ensuring level is conducted under resource allocation level, unlike the application verification level.

We propose a quantitative framework to analyze the VM scheduling problem, in which virtual clusters are used in each tier of a queuing network to provide services on demand based on different workloads. Furthermore, we sole the above model using mathematical analysis from simulation and real experimental verification.

System architecture overview

In this section, we introduce our virtualized cloud platform using a queuing network. We then formulate this virtual resource scheduling problem as a constrained integer optimization problem.

Virtualized cloud platforms

In a virtualized cloud platform, various physical clusters are utilized to provide computational resources, such as CPU and memory allocations. A physical cluster contains many interconnecting physical nodes. VMs are deployed on these physical nodes based on their CPU or memory configuration. In Fig. 2, we show 3 physical clusters. Each physical cluster contains 3 physical nodes with 12 VMs deployed in each physical cluster.

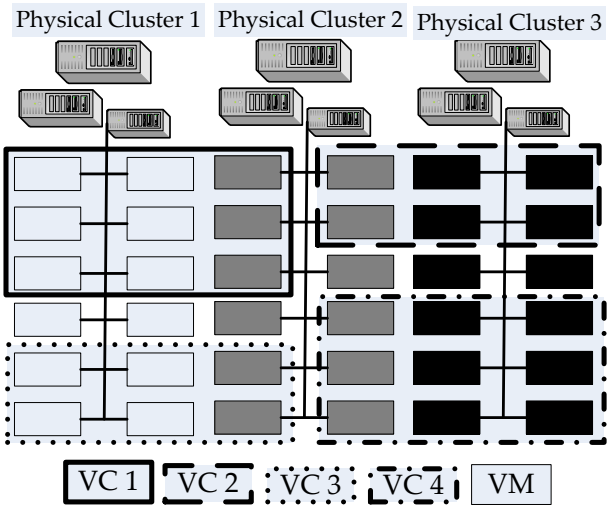

Fig 2. Four virtual clusters built over 3 physical clusters. Each physical cluster consists of a number of interconnected servers, represented by the three servers. Each physical cluster contains 12 virtual machines, represented by the rectangular boxes with 3 different shadings. The virtual machines are implemented on the servers (physical machines). Each virtual cluster can be formed with either physical machines or VMs hosted by multiple physical clusters. The boundaries of the virtual clusters are shown with 4 dot/dash-line boxes. The provisioning of VMs to a virtual cluster can be dynamically based upon user demands.

Virtual clusters are used to partition or reorganize the VMs based on the computational capacity of each VM, the requirement of different application scenario, or more specifically, different workloads. The purpose is to provide an automatic scaling or shrinking mechanism to improve utilization. This mechanism gives customers reliable response times and service availability. It also allows for flexible resource provisioning on demand, which can reduce the cost for using the resources. It benefits vendors in that the resources are allocated as needed which takes utilization into account, providing more effective services.

Based on the resource provisioning principles that many cloud service vendors use, different VMs are pre-deployed with different resources. For example, some VMs are deployed with 1 CPU core with 1 GB memory (thin VMs). Some are deployed with 2 CPU cores with 2 GB memory (fat VMs). This mechanism is shown in Fig. 3.

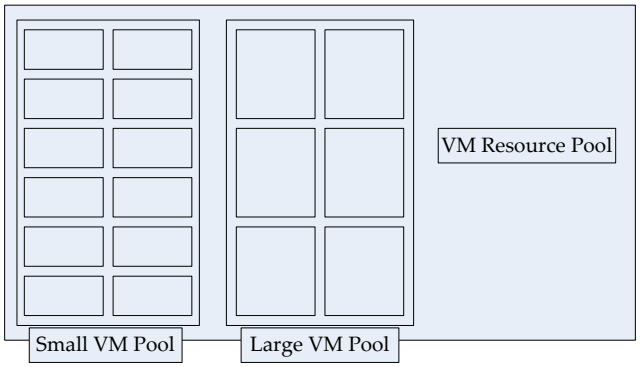

Fig. 3. VM resource pool with VMs of different computational resource. VMs in thin/fat VM pool are equipped with 1 or 2 CPU cores with 1 or 2 GB memories. Different number of VMs in thin/fat VM pool is scheduled into virtual cluster to provide services on demand.

Multi-tier queuing network model

We describe our application specific to multi-tier hosting platforms in Fig. 4. To simplify the illustration, we use a typical and widely used *J*-tier platform as an example. The model in our experiment is a queuing network with *J* queues. Each queue corresponds to one tier in order to handles the inter-arrival requests. There are *n* sessions concurrently generating requests. Each request that arrives at tier *j* ($j \in [1, J-1]$) should either proceed into the $j+1$ tier, with a probability P_j, or return to the $j-1$ tier, with a probability $1-P_j$. The requests that arrive at the J^{th} tier will be returned to the $(J-1)^{th}$ tier with a probability of 1. Thus $P_J = 0$.

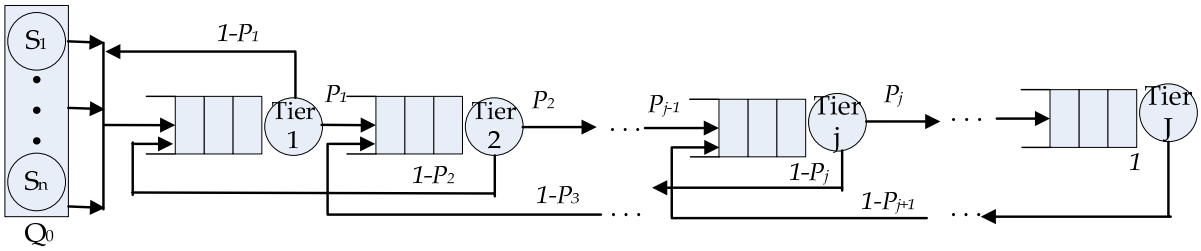

Fig. 4. A multi-tier application hosting platform with *n* sessions generating requests. Each tier is modeled as a queuing system. Any request arriving at the j^{th} tier either proceeds into the $(j+1)^{th}$ tier with a probability denoted by P_j, or returns back to the $(j-1)^{th}$ tier with probability $1-P_j$. All the requests arriving at the J^{th} tier (the last tier) should return to the $(J-1)^{th}$ tier.

Our scheduling target is to achieve an appropriate average response time for customers as well as reducing the costs of using those virtual resources. In our proposed model above, network latency is not included between two tiers since we have conducted numerous experiments within our virtual machine farm and we conclude that the network latency is too small compared with the time used for intensive transaction processing, especially in our followed experiments that all the VMs are constructed within one powerful server. On the other side, all the requests that awaiting for processing are buffered in the modeled queue. The detailed formulation of this problem is elaborated in the following section.

To benefit readers, we introduce the major symbols, concepts, definitions and explanations that will be used in our followed sections, in Table 1.

Table 1. Symbols and definitions

i or i-1	The current or previous stage
j or J	Number of current or total tiers
$N_j^L(i)$ or $N_j^S(i)$	Number of thin or fat VMs used of the j^{th} tier in the i^{th} stage
$\mu_j^S(i)$ or $\mu_j^L(i)$	Average service rate of thin or fat VM of the j^{th} tier in the i^{th} stage
$\rho_j^S(i)$ or $\rho_j^L(i)$	Average arriving-service ratio of thin or fat VM
AST_j^S or AST_j^L	Average staying time of each request in thin/fat VM of the j^{th} tier
AAR_j or $\lambda_j(i)$	Average arrival rate of the j^{th} tier
$\lambda'_j(i)$	Avg. arrival rate of each VM in the j^{th} tier
$AAR_{j-1,j}$ or $AAR_{j+1,j}$	Average arrival rate of the j^{th} tier from the $(j\text{-}1)^{th}$ or $(j\text{+}1)^{th}$ tier
$ADR_{j,j-1}$ or $ADR_{j,j+1}$	Average departure rate of the j^{th} tier to the $(j\text{-}1)^{th}$ or $(j\text{+}1)^{th}$ tier
ADR_j	Average departure rate of the j^{th} tier
AST_j	Average staying time of request in the j^{th} tier
C_j	Concurrency limit of VM in the j^{th} tier
$R(i)$	Requests that are generated during stage *i*
P_j	The probability a request generated from tier *j* proceeds into tier *j*+1.
Θ_S or Θ_L	Available number of thin or fat VM

The relationship of these symbols and their roles in this work are introduced in the followed section.

Virtualized resource scheduling

In this section, we firstly discuss our multi-tier queuing network model and mathematical derivation of the response time requirement. Then we introduce how to solve the model from several mathematical steps. At last, we introduce some parameters used in our model and discuss how to estimate their values.

Modeling of tier structure

Suppose in the i^{th} stage, the number of thin or fat VM of the j^{th} tier is denoted as follows in EQ. 2:

$$N_j^S(i) = N_j^S(i-1) + \Delta N_j^S(i-1) \quad (2.a)$$

$$N_j^L(i) = N_j^L(i-1) + \Delta N_j^L(i-1) \quad (2.b)$$

As shown in Fig. 4, in the current stage, the expected inter-arrival rate of requests denoted by $\lambda_j(i)$, and the inter-arrival rate for each queue is denoted by:

$$\lambda_j'(i) = \lambda_j(i) \Big/ \left[N_j^S(i) + N_j^L(i) \right] \quad (3)$$

$\lambda_j(i)$, which is also called AAR_j, is composed of two parts as shown in Fig. 4. The first part is the requests coming from the previous tier, namely the $(j\text{-}1)^{th}$ tier, which is denoted as $AAR_{j-1,j}$, the second part is the requests that coming from the following tier, namely the $(j+1)^{th}$ tier, denoted as $AAR_{j+1,j}$. We write this as follows:

$$\lambda_j(i) \text{ or } AAR_j = AAR_{j-1,j} + AAR_{j+1,j} \ (j \in [1, J\text{-}1]) \quad (4)$$

$AAR_{0,1}$ is the inter-departure rate of Q_0. In the J^{th} tier, we have

$$\lambda_J(i) \text{ or } AAR_J = ADR_{J-1,J} \quad (5)$$

Each VM can be viewed as an M/M/1/C/∞/FIFO queuing system (Kleinrock, 1976). C_j is the concurrency limit of each VM. Thus we can calculate the average staying time for each request in thin or fat VM of the j^{th} tier by:

$$AST_j^S = \frac{1}{\mu_j^S(i) - \lambda_j'(i)} - \frac{C_j \rho_S^{C_j+1}}{\lambda_j'(i)(1-\rho_S^{C_j})}, \rho_j^S(i) = \frac{\lambda_j'(i)}{\mu_j^S(i)} \quad (6.a)$$

$$AST_j^L = \frac{1}{\mu_j^L(i) - \lambda_j'(i)} - \frac{C_j \rho_L^{C_j+1}}{\lambda_j'(i)(1-\rho_L^{C_j})}, \rho_j^L(i) = \frac{\lambda_j'(i)}{\mu_j^L(i)} \quad (6.b)$$

Based on the above analysis, the average departure rate of the j^{th} tier is

$$ADR_j = N_j^S(i) \Big/ AST_j^S + N_j^L(i) \Big/ AST_j^L \quad (7)$$

Next, we can derive the average staying time of each request in the j^{th} tier as:

$$AST_j = 1/ADR_j \quad (8)$$

Based on our model in Fig. 5, the average arrival rate of the $(j+1)^{th}$ tier from the j^{th} tier is denoted by

$$AAR_{j,j+1} = ADR_j * P_j \quad (9)$$

Similarly, the arrival rate of the $(j-1)^{th}$ tier from the j^{th} tier is denoted by

$$AAR_{j,j-1} = ADR_j * (1 - P_j) \quad (10)$$

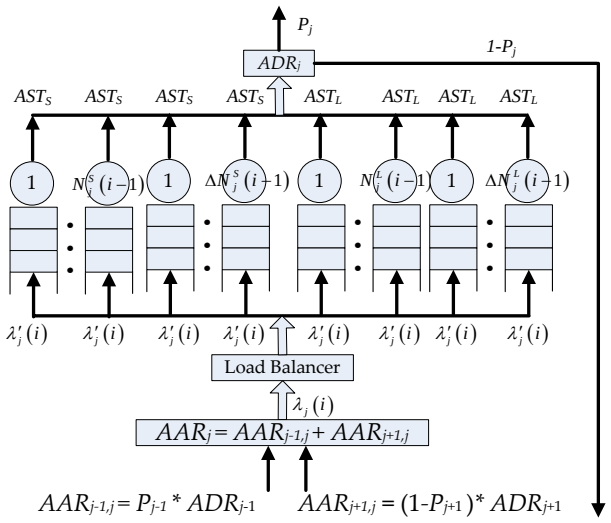

Fig. 5. The inner structure of the j^{th} tier at the i^{th} stage. Each VM is modeled as an individual queuing system. A load balancer is used to fairly distribute the incoming workloads to the VMs. Our target is to find an optimal number of fat and thin VMs in each tier at each stage in order to satisfy the response time demands in SLA.

Implementation procedure consideration

Given the number of fat and thin VMs of the j^{th} tier at the i^{th} stage as demonstrated above, we can calculate the expected average response time at this stage as follows.

(1) Analyze the J^{st} tier:

Based on EQ. (7), ADR_J can be represented as a function of $\lambda_J(i)$, which is initially unknown. We also can derive the average staying time of each request at the J^{st} tier as $AST_J = 1/ADR_J$ based on EQ. (8). In AST_J, we still have one unknown variable: $\lambda_J(i)$. We simplify EQ. (6) as:

$$ADR_J = f_1(\lambda_J(i))$$

(2) Analyze the $(J\text{-}1)^{st}$ tier:

Since $\lambda_{J-1}(i) = AAR_{J-2,J-1} + AAR_{J,J-1} = AAR_{J-2,J-1} + ADR_J$. We have two unknown variables: $AAR_{J-2,J-1}$ and $\lambda_J(i)$. Now we can calculate ADR_{J-1} based on (7), which also has the above same two unknown variables. Suppose this formula is denoted as:

$$ADR_{J-1} = g_2(AAR_{J-2,J-1}, ADR_J) = g_2(AAR_{J-2,J-1}, f_1(\lambda_J(i)))$$

Since $\lambda_J(i) = ADR_{J-1} * P_{J-1}$, we then replace $\lambda_J(i)$ with $ADR_{J-1} * P_{J-1}$ in ADR_{J-1}. In this way, ADR_{J-1} becomes a function of $AAR_{J-2,J-1}$, thus there is only one unknown variable at the end of this step. Suppose this formula is denoted as:

$$ADR_{J-1} = f_2(AAR_{J-2,J-1})$$

We can also derive $AST_{J-1} = 1/ADR_{J-1}$ as shown in (1), which also has one unknown variable.

(3) Analyze the j^{st} tier. j is valued sequentially from J-2 to 1:

We repeat the steps in (2) and calculate AST_j, which is a function of an unknown variable $AAR_{j-1,j}$. When we come to the 1st tier, we have $AAR_{0,1}$ as the average arrival rate of the request generating in clients, which is a known variable in our experiment. In this way, we can calculate the value of ADR_1 and AST_1. If $AAR_{0,1}$ is unknown beforehand, we can use a method such as Kalman filter or neural network to predict the workload of the first tier.

(4) Analyze the j^{st} tier: j is valued sequentially from 2 to J.

In the previous step, we obtain the value of ADR_1. We can calculate $AAR_{1,2}$ as $P_1 * ADR_1$. Since AST_2 is a function of $AAR_{1,2}$, we can calculate AST_2. Similarly, we can go through each tier to get AST_j. j is sequentially valued from 2 to J.

(5) Calculate the average response time:

Suppose there are $R(i)$ requests processed at the i^{th} stage. Based on Fig. 4, we use Num($R_j(i)$) to denote the number of requests that are processed sequentially from the 1^{st} tier to the j^{th}, and then return to the 1^{st} tier until reaching Q_0. Notice that these requests do not visit the (j+1) tier. We use TraverseTime($R_j(i)$) as the average traverse time for each of the above requests. The total average response time can be calculated as shown in EQ. (11).

Table 2 is used to show the relationship between j, Num($R_j(i)$) and TraverseTime($R_j(i)$), which will then be used to determine the average response time.

Table 2. Relationship of the number of tier and traverse time

j	Num($R_j(i)$)	TraverseTime($R_j(i)$)
1	$R(i)(1-P_1)$	$R(i)(1-P_1)*AST_1$
2	$R(i)P_1(1-P_2)$	$R(i)P_1(1-P_2)*(AST_2+2*AST_1)$
---	---	---
j	$R(i)P_1P_2$--- $P_{j-1}(1-P_j)$	$R(i)P_1P_2$--- $P_{j-1}(1-P_j)*(AST_j+2*AST_{j-1}+\ldots+2*AST_1)$
---	---	---
J–1	$R(i)P_1P_2$---$P_{J-2}(1-P_{J-1})$	$R(i)P_1P_2$--- $P_{J-2}(1-P_{J-1})*(AST_{J-1}+2*AST_{J-2}+\ldots+2*AST_1)$
J	$R(i)P_1P_2$---P_{J-2} P_{J-1}	$R(i)P_1P_2$--- $P_{J-1}(1-P_J)*(AST_J+2*AST_{J-1}+\ldots+2*AST_1)$

The average response time is calculated as:

$$\sum_{j=1}^{J} TraverseTime\left(R_j(i)\right) \Big/ R(i)$$
$$= \sum_{j=1}^{J}\left(\prod_{k=0}^{j-1} P_k\right) * \left(1-P_j\right) * \left(2 * \sum_{l=1}^{j} AST_l - AST_j\right) \qquad (11)$$

We define $P_0 = 1$, which means the requests generated from the sessions (Q_0) are bound to arrive at the 1st tier.

Suppose the cost of using a thin and fat VM is denoted by $Cost_S$ and $Cost_L$ respectively. Then the total cost of the i^{th} stage is:

$$Cost(i) = Cost_S * \sum_{j=1}^{J} N_j^S(i) + Cost_L * \sum_{j=1}^{J} N_j^L(i)$$

Our virtual resource allocation problem is converted to a constrained optimization problem as below:

$$\underset{N_j^S(i),N_j^L(i)}{Min}\left(Cost(i)\right)$$

$$\sum_{j=1}^{J}\left(\prod_{k=0}^{j-1}P_k\right)*\left(1-P_j\right)*\left(\prod_{l=1}^{j}\left(2*AST_l-AST_j\right)\right)<SLA$$

$$\sum_{j=1}^{J}N_j^S(i)\le\Theta_S$$

$$\sum_{j=1}^{J}N_j^L(i)\le\Theta_L$$

Θ is the resource pool as shown in Fig. 3. Θ_S and Θ_L denote the available number of thin and fat VMs respectively.

Parameter estimation

Based on our previous analysis, we must evaluate the parameters P_j ($j\in[1, J\text{-}1]$), μ_S, μ_L, $Cost_S$ and $Cost_L$ in order to evaluate the cost and average response time.

The transferring probability P_j can be estimated in an experiment by calculating the average ratio of the number of requests proceed into the P_{j+1} tier to the total number of requests that arrived at tier j in the previous stage. This value is easily monitored offline based on specific application scenario. There are other ways to get this value. For example, we could analyze the source code scripts in Java servlets to get the visiting frequency for each one and this value can also be determined. In our followed experimental studies, this value can be set in the client tools, which benefits our analysis.

$\mu_j^S(i)$ and $\mu_j^L(i)$ can be estimated beforehand. As previously introduced, 1 or 2 CPUs and 1 or 2 GB memories are allocated to thin or fat VMs, respectively. We also perform offline experiments to decide the average service rate based on a certain workload as a test. Networking latency is also included in this offline tests. Concurrency limits of thin and fat VMs are preset based on the servers we are using.

We estimate the cost of using the devices based on the price list found on Amazon EC2 website. The granularity of analytical time span in our experiment is 1 minute, thus we set the cost of using a thin and fat VM as \$0.0127 and \$0.0287 per minute, respectively. This figure is used by combining the price of their hourly costs and current storage costs.

Experimental studies

There are two parts in our experiments. Firstly, we use SimEvents (from Simulink) to develop event-based models of queuing networks to evaluate service times for thin and fat VMs in both web and application tiers. We use Matlab to calculate the response times based on our introduced formula to predict performance. The parameters, such as transferring probabilities, are derived from direct observations of RUBiS [25], an auction site similar to eBay, where transactions of bidding, buying, and selling are used to conduct real transactions. We then run RUBiS benchmark with same parameters to compare the results.

Apache 2.0.55 is used as web server to handle requests as a load balancer as shown in Fig. 5. We use tomcat 5.5 as an application server to deploy servlet applications and connect to a MySQL database server. Our applications are deployed on OpenSUSE 11.

The experiments are carried out on an IBM X3950, with a CPU of 16 cores and memory of 24 GBytes. We virtualized one thin and one fat VM for the web servers. We also virtualized four thin and two fat VMs for application servers. Since the database tier is difficult to cluster, we used one fat VM as database tier in order to process requests effectively.

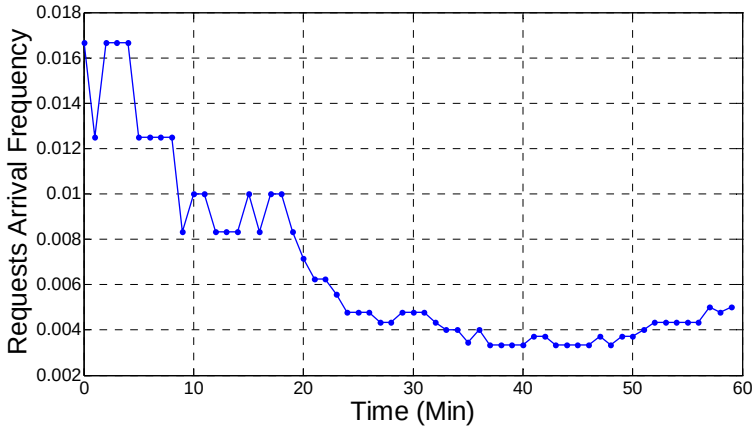

Fig. 6. The workloads generated for benchmarking. We use the inter-arrival time of requests over one hours.

Our workloads are generated based on a web trace from the 1998 Soccer World Cup site [26]. We traced average arrival times during each minute over sixty-minute duration as shown in Fig. 6. The units of the y-axis measures arrival frequency.

In Fig. 7, we demonstrate our experimental results using different numbers of VMs as the workloads increased. We can see that the increased number of VMs can capture the characteristics of such workloads.

We further illustrate our analytical results in Fig. 8. The blue dashed line is the predicted response time from our queuing network model proposed in section 4, and the black solid line denotes the measured response time using the RUBiS benchmark.

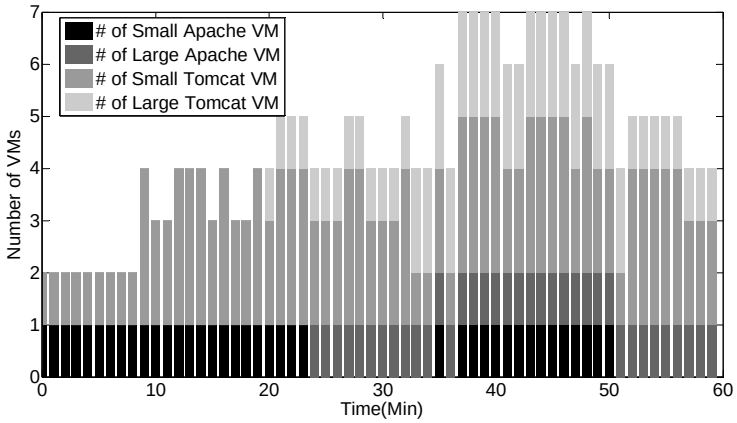

Fig. 7. The number of VMs used for web and application tier over the experiments.

The response time to satisfy the service level agreement (SLA) is set to be 10 seconds. We can see from the one-hour experiment that we can satisfy the requirement with a probability close to 98%. The only time this surpasses the SLA is in the 23rd minute, which is an acceptable variance.

Results show that our proposed model can be used to capture the response time, which can also be used to judge the number of VMs that should be used to properly meet the response time requirements.

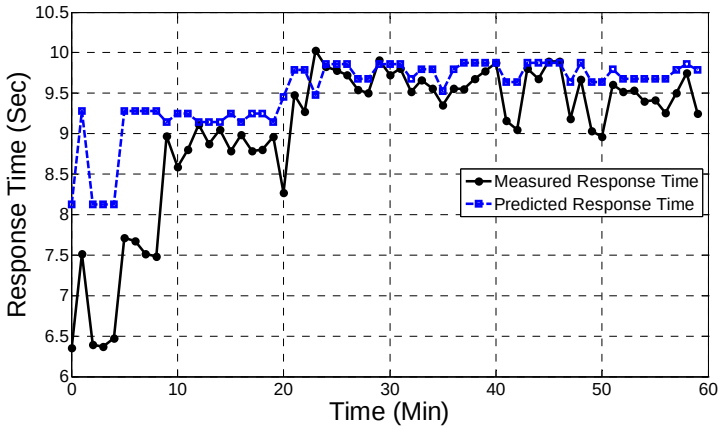

Fig. 8. Comparative results between predicted response times using the proposed queuing network model and measured response times using RUBiS in real transactions

We further compare our model with a commonly used experience based, utilization oriented method. It conducts VM scheduling as follows.

This method ensures that the average utilization rate of all the VMs of each tier is close to 75%. If the value surpasses 95%, a fat VM is initiated if such VM is available, else if the value is less than 95% and larger than 85%, a thin VM is added if one is available. Similarly, if the value is less than 45%, a fat VM is removed if one is running, otherwise if the value is less than 60% and larger than 45%, a thin VM is removed if possible. At least one thin VM should be running in each tier.

This method conducts initiation and removal of VMs at the beginning of each new stage while considering the utilization of the previous stage. This method is noticeably cost-aware. The comparative results of response times and costs are shown in Fig. 9 and Fig. 10.

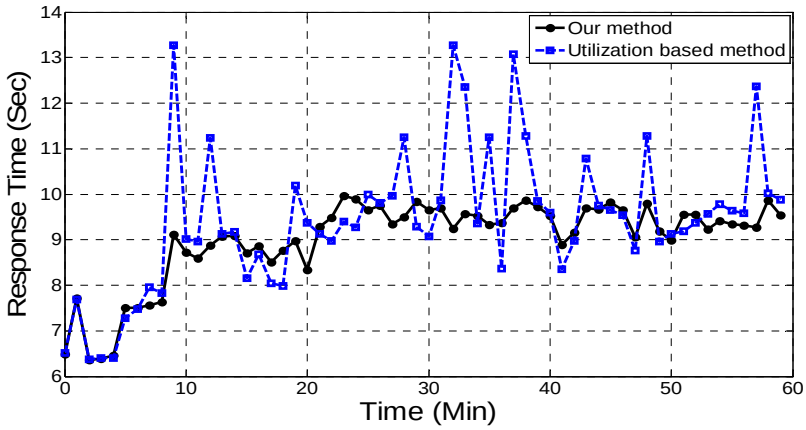

Fig. 9. Comparative results of response times between our method and utilization based method

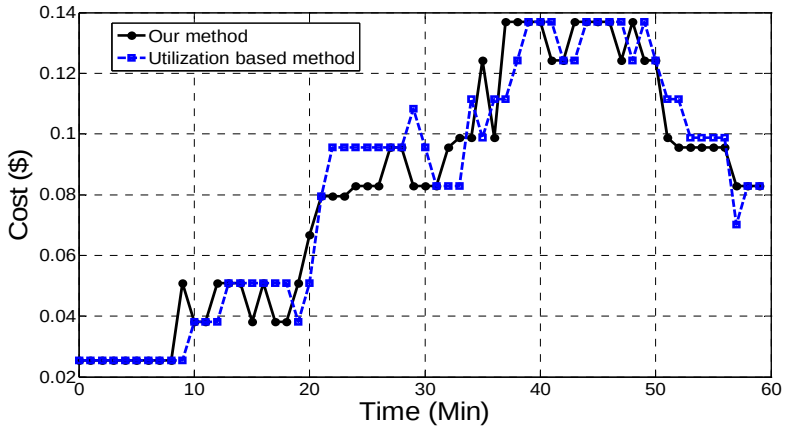

Fig. 10. Comparative results of costs between our method and utilization based method

The advantage of our proposed method is well illustrated in Fig 9 and 10. From the above comparative results, we can see that utilization-based method can also capture the workload variations. However, it cannot be agile enough to provide the proper number of VMs in order to meet the response time demands. From the perspective of cost, this method uses fewer resources than our proposed method, sometimes at the cost of violating the SLA.

Conclusions and future work

We first conclude our major contributions in this work and then suggest two possible directions to extend this work.

Conclusions of this work

In this paper, we have proposed a mathematical model for capturing the characteristics of a virtualized cloud platform using multiple virtual machine instances, and then converted this model into a constrained integer programming problem. The model can directly derive the relationship between the inter-arrival rate of requests and the average response time of requests. We first used a simulation to depict the described relationships, and then benchmarked our

methods using RUBiS, a real transaction web site using VMs on an IBM X3950. Experimental results show that the model can be used to appropriately satisfy the response time requirements as well as reducing the costs of using those virtual machines.

Our future work

For further research, we suggest extending the work in the following two directions:

(1) **Customized virtual resource provisioning**. We propose thin and fat VMs allocation based on a fixed CPU and memory settings. This can be further extended by using flexible resource provisioning principles to satisfy CPU-intensive or memory-intensive applications. To this end, we should reconsider the costs, including service deployment, starting up and shutting down of services, et cetera.

(2) **Building useful tools to serve for larger virtualized cloud platform**. The Matlab simulation used to analyze the optimal number of virtual resources in our experiment should be packaged into software toolkits in order to make it available for larger virtualized cloud platforms. Our experimental software can be tailored and prototyped to this end.

Acknowledgements

This work is supported by National Science Foundation of China (grant No. 60803017) and Ministry of Science and Technology of China under National 973 Basic Research Program (grants No. 2011CB302505 and No. 2011CB302805). This work is also supported by National Science Foundation of United States under Grant No. OISE-0730065 of Partnerships for International Research and Education (PIRE). Fan Zhang thanks IBM for 2010-2011 and 2011-2012 IBM Ph.D. Fellowship support.

REFERENCES

Armbrust M., Fox A., Griffith R., Joseph A. D., Katz R. H., Konwinski A., Lee G., Patterson D. A., Rabkin A., Stoica I., and Zaharia M. (2009). Above the clouds: A berkeley view of cloud computing. Technical Report No. UCB/EECS-2009-28

Arlitt M., and Jin T. (1999). Workload Characterization of the 1998 World Cup Web Site. Tech. Rep. HPL-1999-35R1, HP Labs.

Buyya R., Yeo C. S., and Venugopal S. (2008), Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities, Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC 2008, IEEE CS Press, Los Alamitos, CA, USA), Sept. 25-27, Dalian, China.

Chase J., and Doyle R. (2001). Balance of power: Energy management for server clusters. In Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII). Elmau, Germany.

Calheiros R. N., Ranjan R., Beloglazov A., Rose C., and Buyya R. (2010), CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience, ISSN: 0038-0644, Wiley Press, New York, USA.

Emeneker W., et al (2006), Dynamic Virtual Clustering with Xen and Moab, International Symposium on Parallel and Distributed Processing with Applications (ISPA), Springer-Verlag LNCS 4331, 440-451

Li S., and Tirupati D. (1995). Technology choice with stochastic demands and dynamic capacity allocation: A two-product analysis. Journal of Operations Management, 12(3-4) 239-258.

Kamra, Misra V., Nahum E.M. (2004), Yaksha: a self-tuning controller for managing the performance of 3-tiered Web sites, In Proceedings of the 12th International Workshop on Quality of Service(IWQoS), Passau, Germany.

Kleinrock L. (1976), Queueing Systems, Volume 2: Computer Applications. John Wiley and Sons, Inc..

Slothouber L. (1996). A model of web server performance. In Proceedings of the 5th International World Wide Web Conference (WWW). Paris, France.

Levy R., Nagarajarao J., Pacifici G., Spreitzer M., Tantawi A., and Youssef A. (2003). Performance management for cluster based web services. In IFIP/IEEE 8th International Symposium on Integrated Network Management. 247–261.

Menasce D. (2003), Web server software architectures. IEEE Internet Computing. 7(6), 78-81.

Ranjan S., Rolia J., FU H., and Knightly E. (2002). QoS-driven server migration for internet data centers. In Proceedings of the 10th International Workshop on Quality of Service(IWQoS), Miami, FL.

Tan W., Fan Y., Zhou M., A Petri Net-based Method for Compatibility Analysis and Composition of Web Services in Business Process Execution Language. IEEE Transactions on Automation Science and Engineering. 2009, vol. 6, issue 1: 94-106

Tan W., Fan Y., Zhou M., Tian Z., Data-driven Service Composition in Building SOA Solutions: A Petri Net Approach. IEEE Transactions on Automation Science and Engineering. 2010, 7(3): 686 - 694

Turner M., Budgen D., and P. Brereton (2003). Turning Software into a service. Computer. 36(10), 38-44

Urgaonkar B., Pacifici G., Shenoy P., Spreitzer M., and Tantawi A. (2007). Analytic Modeling of Multi-tier Internet Services and its Applications. ACM Transactions on the Web (TWEB 2007), 1(1), 1-35.

Urgaonkar B., and Shenoy P. (2004), Cataclysm: Handling extreme overloads in internet services. In Proceedings of the 23rd Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC). St. John's, Newfoundland, Canada.

Urgaonkar, Shenoy P., Chandra A., Goyal P., and Wood T. (2008). Agile Dynamic Provisioning of Multi-tier Internet Applications. ACM Transactions on Adaptive and Autonomous Systems (TAAS), 3(1).

Villela D., Pradhan P., and Rubenstein D. (2007). Provisioning servers in the application tier for e-commerce systems. ACM Transactions on Internet Technology (TOIT). 7(1), 57-66.

Wu Y., Hwang K., Yuan Y., and Zheng W. (2009), "Adaptive Workload Prediction of Grid Performance in Confidence Windows", IEEE Trans. on Parallel and Distributed Systems, to appear, (on-line published Aug. 14, 2009)

Xiong P., Fan Y. and Zhou M., "Web Service Configuration under Multiple Quality-of-Service Attribute," IEEE Transactions on Automation Science and Engineering, Vol. 6, Iss. 2, April 2009, pp.311-321

Xiong Pe., Fan Y. and Zhou M., "QoS-aware Web Service Configuration," IEEE Transactions on System, Man and Cybernetics, Part A, Vol. 38, Iss. 4, July 2008, pp.888-895

Zhang F., Cao J., Liu L., and Wu C., "Fast Autotuning Configurations of Parameters in Distributed Computing Systems Using Ordinal Optimization", Proc. 38th Int. Conf. on Parallel Processing Workshops, Vienna, Austria, 190-197, 2009.

Zhang F., Cao J., Song X., Cai H., and Wu C., "AMREF: An Adaptive MapReduce Framework for Real Time Applications", Proc. of 9th Int. Conf. on Grid and Cloud Computing (GCC'10), Nanjing, China, 2010.

ABOUT THE AUTHOR(S)

Fan Zhang received the B.S. in computer science from Hubei Univ. of Technology and M. S. in control science and engineering from Huazhong University of Science and technology. He is currently a Ph.D. student in Department of Automation, Tsinghua University, Beijing, China. His research interests include data center networks and grid/cloud computing. Contact him at: zhang-fan07@mails.tsinghua.edu.cn

Junwei Cao is currently Professor and Assistant Dean, Research Institute of Information Technology, Tsinghua University, China. He was a Research Scientist at MIT LIGO Laboratory and NEC Laboratories Europe. He received the PhD in Computer Science from University of Warwick, UK, in 2001. He is a Senior Member of the IEEE Computer Society and a Member of the ACM and CCF. Contact him via Email: jcao@tsinghua.edu.cn

Hong Cai is currently a senior software engineer of IBM China development lab and chair of TEC-GC (Technical Expert Council - Greater China). He received the B.S and Ph.D. from Tsinghua Univ., China. His research interests include service oriented architecture, dynamic resource provisioning, distributed and cloud computing, etc. Contact him via Email: caihong@cn.ibm.com

James J. Mulcahy is currently a Ph.D student in Computer Science at Florida Atlantic University in Boca Raton, Florida, USA. His research interests include grid/cloud applications, software engineering and re-engineering of legacy systems. He has more than 20 years of business experience in developing and evolving large supply chain and e-commerce software applications. James is a member of IEEE and the Association of Computing Machinery (ACM). James can be reached via email at jmulcah1@fau.edu

Cheng Wu is a Professor of Department of Automation, Tsinghua Univ. China, Director of National CIMS Engineering Research Center, and Member of Chinese Academy of Engineering. He received his B.S. and M.S. from Department of automation, Tsinghua Univ. in 1962 and 1966 respectively. His research interests include complex manufacturing system scheduling, grid/cloud applications, etc. Contact him via Email: wuc@tsinghua.edu.cn