Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 215016, 11 pages
http://dx.doi.org/10.1155/2014/215016

Research Article

Coarse-Grain QoS-Aware Dynamic Instance Provisioning for Interactive Workload in the Cloud

Jianxiong Wan, Limin Liu, Jie Lv, and Zhiwei Xu

School of Information Engineering, Inner Mongolia University of Technology, Hohhot 010080, China

Correspondence should be addressed to Jianxiong Wan; jxwan@csnet1.cs.tsinghua.edu.cn

Received 22 November 2013; Revised 22 January 2014; Accepted 23 January 2014; Published 25 March 2014

Academic Editor: Huiping Li

Cloud computing paradigm renders the Internet service providers (ISPs) with a new approach to deliver their service with less cost. ISPs can rent virtual machines from the Infrastructure-as-a-Service (IaaS) provided by the cloud rather than purchasing them. In addition, commercial cloud providers (CPs) offer diverse VM instance rental services in various time granularities, which provide another opportunity for ISPs to reduce cost. We investigate a Coarse-grain QoS-aware Dynamic Instance Provisioning (CDIP) problem for interactive workload in the cloud from the perspective of ISPs. We formulate the CDIP problem as an optimization problem where the objective is to minimize the VM instance rental cost and the constraint is the percentile delay bound. Since the Internet traffic shows a strong self-similar property, it is hard to get an analytical form of the percentile delay constraint. To address this issue, we purpose a lookup table structure together with a learning algorithm to estimate the performance of the instance provisioning policy. This approach is further extended with two function approximations to enhance the scalability of the learning algorithm. We also present an efficient dynamic instance provisioning algorithm, which takes full advantage of the rental service diversity, to determine the instance rental policy. Extensive simulations are conducted to validate the effectiveness of the proposed algorithms.

1. Introduction

Before the advent of cloud computing, Internet service providers (ISPs) used to reserve mass amount of resources in order to deal with the peak workload; otherwise the service response time may increase to an intolerable degree while facing the flash crowd and greatly degrade the user experience. However, this approach is energy-ineffective since peak resource utilization is often three times larger than the average utilization for a typical ISP. Things get even worse in systems that provide interactive service where the average utilization is only around 10% of the total capacity provisioned for the peak load [1]. The cloud computing technology provides a novel service paradigm called Infrastructure-as-a-Service (IaaS) to reduce the hardware cost and maintenance cost. In the IaaS, the ISPs only need to rent resource (e.g., virtual servers and network bandwidths) from the cloud providers (CPs) instead of purchasing a vast number of physical servers themselves. The IaaS service enables a more flexible and effective approach for resource provisioning. For example, users in the Amazon EC2 system can rent resource for a small period of time to cope with the flash traffic.

This paper studies a Coarse-grain Dynamic Virtual Machine (VM) Instance Provisioning (CDIP) problem for interactive workload subjected to a percentile delay constraint in the cloud from the perspective of ISPs. More specifically, this problem is related to the dynamic VM rental policy for the ISPs to minimize the resource rental cost while satisfying QoS constraints. A fine-grain (in the orders of seconds or minutes) resource provisioning policy may be more effective in increasing resource utilization and reducing cost, but it is more complex and hard to implement. For example, the startup phase of a VM instance in EC2 which "typically takes less than 10 minutes [2] (observed on November 2nd, 2013)" is not sufficient to support the fine-grain control policy. Further, the fine-grain policy can induce fluctuation and undermine the system stability. CPs like Amazon EC2 nowadays do provide a coarse-grain IaaS service instead of the fine-grain one. For example, the EC2 system offers IaaS service at 2 time scales. At a higher level,

Table 1: The pricing structure for Amazon EC2.

Notation	C_L	C_S
Cost ($/hr)	0.448	0.680

there is a VM rental service for 1 or 3 years (denoted as *Reserved Instance Service*, RIS); at a lower level, VM instances can also be acquired on an hourly bases (denoted as *Marginal Instance Service*, MIS) to absorb the instant flash traffic. Generally speaking, the cost for using MIS instances is much higher than using RIS instances (refer to Table 1 for a detailed pricing structure in Amazon's EC2 platform). How to properly use these two services is one of the most important problems faced by ISPs to minimize cost.

Beside the VM instance rental cost, ISPs also care about the Quality-of-Service (QoS) issue for their end users. For interactive workload, traditional QoS is expressed by the mean queueing delay which is easy to analyze using classic queueing theory. However, the self-similar nature revealed in the Internet traffic [3] failed queueing-based analysis. In addition, the fact that interactive workload can tolerate some QoS violations drives researchers to propose an alternative form of QoS specification

$$\Pr\left(d \geq D_{\text{th}}\right) \leq x, \tag{1}$$

where d is the system response delay, D_{th} and x are the desired threshold value determined by Service Level Agreement (SLA). Unfortunately, there is no analytical form of (1) for the self-similar traffic.

In this paper, we formulate the CDIP problem as an optimization problem where the QoS constraints cannot be precisely determined. We develop efficient algorithms to solve the CDIP problem and conduct numerical analysis to evaluate the proposed algorithms. Our contributions are that

(i) we design a resource prediction algorithm to estimate the performance of resource provisioning policy in the self-similar traffic,

(ii) we extend the resource prediction algorithm with function approximations to enhance the scalability of the algorithm,

(iii) we present a VM instance provisioning algorithm for ISPs to determine the optimal number of RIS and MIS VM instance, which minimizes the VM instance rental cost.

This paper proceeds as follows. Section 2 discusses the related works; Section 3 shows the opportunity for reducing rental cost using hybrid RIS/MIS; Section 4 presents a general optimization framework for the CDIP problem as well as the solution algorithms; Section 5 extends the algorithms with function approximations to address the scalability issue; Section 6 evaluates the proposed algorithms in various settings, followed with conclusions in Section 7.

2. Related Works

To make resource provisioning in the cloud computing environment, the first issue that must be addressed is to predict the future resource demand accurately. There are many researches dedicated to this area. Chen et al. [4] used a multiplicative Seasonal Autoregressive Moving Average (SARMA) approach to predict the mean and standard deviation of interarrival times and used a simple decomposed model as well as Winter's smoothing method to predict the mean and standard deviation of file size. Gmach et al. [5] developed a pattern prediction method for cyclic workload through a workload periodogram function and an autocorrelation function. Caron and Desprez [6] used pattern matching to forecast the resource demand in the cloud. Niu et al. [7] proposed a channel interleaving scheme which can predict demand for new videos that lack historical demand data.

There are a number of works to lower the operational cost for the cloud providers (CPs). Ahmad and Vijaykumar [8] proposed a PowerTrade method to lower the total energy consumption of active servers, standby servers, and cooling facilities. They also developed a SurgeGuard method to maintain an extra number of servers at two time granularities to absorb flash crowd. Meisner et al. [1] developed a PowerNap mechanism which includes a sleep-active state scheduling component and a network interface card (NIC) supported by Wake-on-LAN functionality. The system is put into the sleep mode when there are no workloads. The NIC can wake the system up within 1 ms as long as there are packet arrivals from the networks. Leverich and Kozyrakis [9] integrated Hadoop system with an energy controller which recasts the data layout and task distribution to enable significant portions of a cluster to be shut down. Our work, on the other hand, studies how to reduce the cost from the perspective of Internet service providers (ISPs).

There are some recent researches close to our works. In [10], the author formulated the resource leasing problem as an Integer Programming Problem (IPP) and developed CoH, a family of heuristic policy to solve the problem. However, [10] treated batch jobs only and had little SLA considerations. Reference [11] also studied the instance provisioning problem and purposed a dynamic instance purchasing scheme based on the Central Limit Theorem to minimize the cost. The SLA constraint they considered is the overload probability which is not suitable for delay-sensitive interactive workload. The works [12, 13] make resource provisioning decision based on the Autoregressive Integrated Moving Average (ARIMA) prediction method; they still did not consider delay constraint. In contrast, [14] explicitly incorporated the delay into the objective function of the optimization problem. However, the delay was derived based on Markovian queueing theory which is not the case in today's Internet dominated by self-similar traffic.

3. Problem Statement

The structure of a data center in a cloud computing system is shown in Figure 1. Inside the data center, there are a number of physical servers. A physical server hosts one or more Virtual Machine (VM) according to its resource capacity. Note that we only present the VM instead of the physical server in the figure. An ISP rents VMs from the cloud

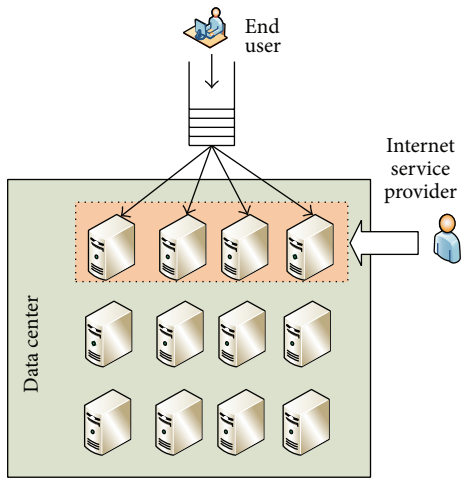

Figure 1: A data center in the cloud computing system.

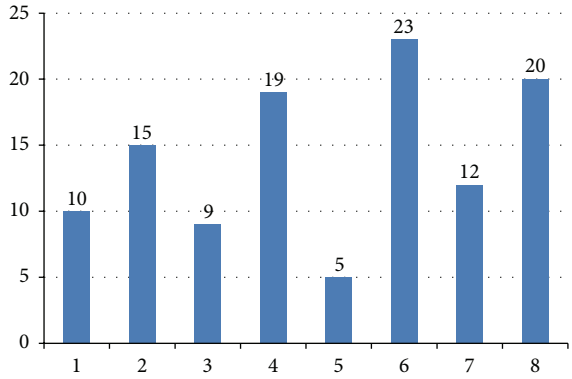

Figure 2: An example of VM instance demand in different hours of a day.

provider serve to its end users. To reduce the request response time, the data center often employs a shared queue structure.

The arrival rate of end user varies over time, which induces a time-changing VM instance demand. Figure 2 presents an example which divides a day into 8 phases (3 hr/phase) and the y-axis shows the VM instance demand to ensure the QoS requirement in each phase. The marginal rental cost in Amazon EC2 is given in Table 1. From Figure 2, we can see that there is a big gap between the maximum and the minimum instance demand. If the ISP only uses RIS instance, he must acquire 23 instances in order to satisfy the peak workload appeared in the 6th phase, which wastes a lot of resource and rises the daily instance rental cost to 247.96\$ (the rental cost for using only RIS instance can be computed as $23 \times 0.448 \times 24 = 247.96$\$ (the product of the number of instance, the marginal cost, and total 24 hours)). In contrast, if the ISP only adopts MIS instance, he will obtain the highest resource utilization, and there is an opportunity to reduce the daily rental cost to 230.52\$ (from Figure 2, the total number of MIS instances is $10 + 15 + 9 + 19 + 5 + 23 + 12 + 20 = 113$. Since a phase contains 3 hours, the rental cost for using only MIS instance can be computed as $113 \times 3 \times 0.680 = 230.52$\$).

If the ISP uses a hybrid approach which includes both RIS and MIS, on the other hand, the daily instance rental cost can be remarkably reduced. To see that, consider a resource provisioning policy which rents 10 RIS VM instances and acquires extra MIS instances if RIS instances are insufficient. The number of MIS instance can be formally written as $[K_i - 10]^+$ where K_i denotes the number of VM instance demand in phase i. The daily rental cost for this hybrid approach is 187.08\$, (the rental cost for RIS instance is $10 \times 0.448 \times 24 = 107.52$\$. The total number of MIS instances is $5 + 0 + 9 + 13 + 2 + 10 = 39$; therefore the rental cost for MIS instance is $39 \times 0.680 \times 3 = 79.56$\$. Thus, the total cost is $107.52 + 79.56 = 187.08$\$.), which saves 24.3% and 18.8% compared with using purely RIS and MIS instance, respectively.

The above analysis suggests 2 assumptions. First, the QoS performance in terms of percentile delay can be precisely predicted; second, the number of RIS and MIS instances can be determined to minimize the VM instance rental cost. The following sections explain these two assumptions in detail.

4. A General Optimization Framework for the CDIP Problem

The notations used in this paper are shown in Notations section. The CDIP problem can be formulated as

$$\min_{k_i, i \in \{0,\ldots,N\}} k_0 \times N \times C_L + C_S \times \sum_{i=1}^{N} k_i \tag{2}$$

subject to

$$\Pr\left(d_i \geq D\right) \leq x, \quad \forall i \in \{1, \ldots, N\}, \tag{3}$$

where k_0 is the number of RIS instance and $k_i, i > 0$, is the number of MIS instance in phase i.

Note that, in the CDIP problem, the distribution of d_i is determined by the characteristics of exogenous interactive workload arrivals and the number of active VM instance k_i. As stated in Section 1, this problem is hard to solve, since we can hardly derive an explicit form of constraint (3). In this section, we will show how to approximately characterize constraint (3) and obtain the optimal solution.

4.1. A Learning Algorithm to Characterize the Percentile QoS Constraint in Self-Similar Traffic. Algorithm 1 learns the performance of various instance provisioning policies in the form of percentile delay via the stochastic gradient method. The algorithm first creates a data structure called VP_table (Violation Probability Table), in which each item VP_table$[i][k]$ estimates the delay violation probability given the number of instance being k in phase i. The algorithm runs for several iterations to obtain unbiased delay violation probability samples p$[i][k]$ for each phase i. These samples, which can be generated via real system running or simulation, are further smoothed into VP_table$[i][k]$. Therefore, VP_table$[i][k]$ is an unbiased estimation of delay violation probability with k VM instances in phase i. Variables η, i, and k are iteration counter, decision point counter, and instance number counter, respectively. Algorithm 1 has the following property.

Proposition 1. *Algorithm 1 converges to the unbiased estimation of percentile QoS performance of using k VM instances in phase i.*

Proof. The right-hand side of line 11 in Algorithm 1 can be rewritten as

$$E[i][k] = \frac{(\eta - 1) \times \text{VP\_table}[i][k] + \text{p}[i][k]}{\eta}. \quad (4)$$

Since $\text{p}[i][k]$ is an unbiased sample of percentile QoS performance metric, $E[i][k]$ is the mean value of all samples up to iteration η. As long as the end user request arrival process and service process are stationary stochastic processes in phase i with k VM instances, $E[i][k]$ must be an unbiased estimation of percentile QoS performance as $\eta \to \infty$. □

In practice, it is impossible to let $\eta \to \infty$. In fact, Algorithm 1 converges very fast in our numerical analysis (it converges within tens of iterations). Alternatively, we can also use the following equation as the stop criterion:

$$\left|\text{VP\_table}[i][k]_{\eta} - \text{VP\_table}[i][k]_{\eta-1}\right| \le T, \quad \forall i, k, \quad (5)$$

where T is a threshold value to get a desired precision.

4.2. The Instance Provisioning Algorithm. Based on the VP_table, we can obtain the minimum number of VM instances needed to meet the QoS constraints in phase i, that is, K_i. To find the number of RIS instances k_0 is equal to solve the following optimization problem

$$\min_{k_0} N \times C_L \times k_0 + \sum_{i=1}^{N} [K_i - k_0]^+ \times C_S \quad (6)$$

subject to

$$\Pr(d_i(k_i) \ge D_{\text{th}}) \le x, \quad (7)$$

where delay is considered as a function of the number of VM instances.

Problems (6)-(7) are an integer piece-wise function of k_0 where the optimal solution must appear in the boundary points. Algorithm 2 provides the solution method for problem (6). It can be divided into three parts as follows.

(i) The first part (lines 1–8) uses exhaustive search to obtain the minimum number of VM instance required to satisfy QoS constraints. The result is stored in vector $K_i, i \in \{1, \ldots, \text{N}\}$.

(ii) The second part (lines 9–17) solves problems (6)-(7), and the result is k_0, the optimal number of RIS instances, and the corresponding value of object function m.

(iii) The third part (lines 18–20) computes the number of MIS instances based on K_i and k_0.

The worst time complexity of Algorithm 2 is $\mathbb{O}(N \times (\text{MAX\_NUM} - \text{MIN\_NUM} + 2))$.

5. Extensions

Algorithms 1 and 2 can effectively predict the number of instances needed for satisfying QoS constraints and reducing total rental cost for the ISPs. However, the scalability of these two algorithms is questionable: in order to obtain a precise estimation of the violation probability in VP_table, we must visit all possible instance provisioning policies and get sufficient violation probability samples. This section starts from the point of simplifying VP_table by function approximation techniques to enhance the scalability of Algorithms 1 and 2.

The idea of function approximation is to use a function $x = f_i(K_i)$ to approximate the mapping between the number of instances and the violation probability in phase i. In this paper, we use two forms of approximation:

(i) a linear approximation given by

$$x = a_i + b_i K_i, \quad b_i < 0, \ x > 0, \quad (8)$$

(ii) a nonlinear approximation given by

$$x = a_i \times K_i^{b_i}, \quad b_i < 0. \quad (9)$$

Note that function $f_i(K_i)$ is related to a certain phase i; therefore the parameters a and b have a subscript i. We have further remarks for these function approximations as follows.

(1) Intuitively, the QoS violation probability decreases as there are more VM instances; that is, f_i is a decreasing function with respect to K_i; therefore b_i must be negative in the nonlinear case.

(2) The value of f_i will all be 0 when K_i exceeds a certain threshold, since no QoS violations occur if there is sufficient number of VM instances. When using linear approximation, we should filter out the case $f_i(K_i) = 0$; otherwise the estimation precision will be remarkably undermined for cases where $f_i(K_i) > 0$.

We use the least square approach to obtain parameters a_i and b_i in the approximate function f_i. Formally, the least square approach is given by

$$\min_{a_i, b_i} F(a_i, b_i) = \frac{1}{2} \sum_{j=1}^{n} \left(\widehat{x}_i^j - f_i\left(K_i^j\right)\right)^2, \quad (10)$$

where n is the amount of samples and $\widehat{x}_i^j$ is the jth unbiased sample for violation probability x.

For the linear approximation, the optimal solution should satisfy

$$\frac{\partial F}{\partial a_i} = -\sum_{j=1}^{n} \left(\widehat{x}_i^j - \left(a_i + b_i K_i^j\right)\right) = 0,$$
$$\frac{\partial F}{\partial b_i} = -\sum_{j=1}^{n} \left(\widehat{x}_i^j - \left(a_i + b_i K_i^j\right)\right) K_i^j = 0. \quad (11)$$

Input: ITE, N, and SLA specification D_{th}; {ITE is the number of iterations and N is the number of decision points in a day.}
Output: VP_table;
(1) Create VP_table and initialize each item in VP_table to 0;
(2) Create p[i][k] and *counter*; {p[i][k] is a sample of QoS violation ratio of using k VM instances in phase i, and *counter* logs the number of delay violations in a phase.}
(3) **for** $\eta = 1$ to ITE **do**
(4) **for** $i = 1$ to N **do**
(5) **for** k = MIN_NUM to MAX_NUM **do**
(6) Log response time d_i for each incoming request;
(7) **if** $d_i > D_{\text{th}}$ **then**
(8) counter + +;
(9) **end if**
(10) Calculate an unbiased sample of delay violation probability p[i][k] = counter/n_i^{η} , where n_i^{η} is the total number of requests arrived in phase i, iteration η;
(11) VP_table[i][k] $\leftarrow (\eta - 1)/\eta \times$ VP_table[i][k] + $(1/\eta)$ p[i][k] ;
(12) **end for** {Loop k;}
(13) **end for** {Loop i;}
(14) **end for** {Loop η;}

Algorithm 1: The learning Algorithm to characterize the Percentile QoS Constraint.

Input: VP_table;
Output: $k_i, i \in \{0, \ldots, N\}$; {$k_0$ is the number of RIS instance, and $k_i, i \neq 0$ is the number of MIS instance in phase i.}
(1) **for** $i = 1$ to N **do**
(2) **for** j = MIN_NUM to MAX_NUM **do**
(3) **if** VP_table[i][j] $\leq x$ and VP_table[i][$j + 1$] $\geq x$ **then**
(4) $K_i = j + 1$;
(5) break;
(6) **end if**
(7) **end for**
(8) **end for**
(9) $k_0 = 0$;
(10) $m = N \times C_L \times k_0 + \sum_{i=1}^{N} [K_i - k_0]^+ \times C_S$;
(11) **for** $i = 1$ to N **do**
(12) $j = K_i$;
(13) temp $= N \times C_L \times j + \sum_{i=1}^{N} [K_i - j]^+ \times C_S$;
(14) **if** temp $< m$ **then**
(15) $k_0 = j$;
(16) **end if**
(17) **end for**
(18) **for** $i = 1$ to N **do**
(19) $k_i = K_i - k_0$;
(20) **end for**

Algorithm 2: The instance provisioning algorithm.

Rearranging these two equations, we have

$$\begin{aligned} n a_i + b_i \sum_{j=1}^{n} K_i^j &= \sum_{j=1}^{n} \widehat{x}_i^j, \\ a_i \sum_{j=1}^{n} K_i^j + b_i \sum_{j=1}^{n} \left(K_i^j\right)^2 &= \sum_{j=1}^{n} K_i^j \widehat{x}_i^j. \end{aligned} \tag{12}$$

The above analysis suggests

$$\begin{aligned} b_i &= \frac{\sum_{j=1}^{n} K_i^j \times \sum_{j=1}^{n} \widehat{x}_i^j - n \sum_{j=1}^{n} K_i^j \widehat{x}_i^j}{\left(\sum_{i=1}^{n} K_i^j\right)^2 - n \sum_{j=1}^{n} \left(K_i^j\right)^2}, \\ a_i &= \frac{\sum_{j=1}^{n} \widehat{x}_i^j - b_i \sum_{j=1}^{n} K_i^j}{n}. \end{aligned} \tag{13}$$

(1) **for** $i = 1$ to L **do**
(2) Measure λ_i; {λ_i is the number of request arrivals in time window i.}
(3) $t_i = 1/\lambda_i$; {Estimate the average inter-arrival time in time window i.}
(4) $A \leftarrow A + \lambda_i$; {$A$ logs the accumulative total number of request in this time slot.}
(5) **end for**
(6) $t_a = L/A$; {Estimate the average inter-arrival times in the time slot.}
(7) $\sigma = \sqrt{\sum_{i=1}^{L} \lambda_i (t_i - t_a)^2}$; {Estimate the standard deviation of inter-arrival time.}
(8) $C_A = \sigma / t_a$;

Algorithm 3: Online estimation of C_A.

For the nonlinear approximation, let $\chi = \ln x$, $\kappa_i = \ln K_i$, $A_i = \ln a_i$, and $B_i = b_i$, and take "ln" in both sides of (9), which transforms the nonlinear approximation into a linear approximation

$$\chi = A_i + B_i \kappa_i. \tag{14}$$

Following the idea of the linear approximation, we can obtain the solution for the nonlinear approximation as

$$b_i = \frac{\sum_{j=1}^{n} \ln K_i^j \times \sum_{j=1}^{n} \ln \widehat{x}_i^j - n \sum_{j=1}^{n} \ln K_i^j \ln \widehat{x}_i^j}{\left(\sum_{i=1}^{n} \ln K_i^j\right)^2 - n \sum_{j=1}^{n} \left(\ln K_i^j\right)^2},$$
$$a_i = \exp\left\{ \frac{\sum_{j=1}^{n} \ln \widehat{x}_i^j - b_i \sum_{j=1}^{n} \ln K_i^j}{n} \right\}. \tag{15}$$

We integrate the function approximations into Algorithms 1 and 2 where VP_table is replaced by an array func_app[N]. Each item in func_app[N] contains 2 elements, that is, a and b. With function approximations, some revisions are needed for Algorithms 1 and 2, which are shown in Table 5.

6. Evaluations

6.1. Simulation Setup. Internet traffic shows a strong self-similar property [3, 15]. We use the Multiscale Markov-Modulated Poisson Processes (MMPP) model to generate a self-similar like traffic. This approach has been proved effective in previous researches [16–18] and was successfully applied in the literatures like [19–22]. We use the approach the same as in [22], that is, a three-dimension Markov on-off modulated Poisson process, to generate the interactive workload arrivals. Consider the following.

(i) The first dimension is the workload burst in the order of 1 second. We assume that the peak workload arrives at the middle of the day, that is, the 43200th second; therefore the arrival rate as a function of time can be given by

$$\lambda(t) = \begin{cases} \dfrac{5t}{216 + 1000}, & \text{if } 0 \le t \le 43200 \\ \dfrac{-5t}{216 + 3000}, & \text{if } 43200 < t \le 86400. \end{cases} \tag{16}$$

(ii) The second dimension of workload burst is 2000 requests per 5 second.

(iii) The last dimension of workload burst is 5000 requests per 10 second.

6.2. Estimation of the Response Time. In a production cloud system, it is impossible to log the response time for each incoming request to calculate the delay violation probability. A more practical way is to measure the mean response delay $\overline{d}$ in a small time slot and view $\overline{d}$ as the response delay for all requests arrived in this time slot. This approximation of response delay will be more accurate as the length of the time slot decreases. For example, in [23], the length of the time slot is set to 10 minutes. In our work, we set it to 10 seconds since we need to measure delay violation probability in a higher precision.

To estimate the mean response time in a time slot, we employ the Allen-Cunneen approximation formula [24, 25] for the $G/G/m$ queueing system:

$$R = \frac{1}{\mu} + \frac{P_m}{\mu(1 - \rho)} \times \frac{C_A^2 + C_S^2}{2m}, \tag{17}$$

where R is the average response time, μ is the average service rate, λ is the average arrival rate, $\rho = \lambda/\mu m$ is the average utilization of a server, m is the number of servers. P_m takes value from the following formula:

$$P_m = \begin{cases} \rho^{(m+1)/2}, & \text{if } \rho \le 0.7 \\ \dfrac{\rho^m + \rho}{2}, & \text{if } \rho > 0.7, \end{cases} \tag{18}$$

C_A and C_S are the coefficients of variation of request interarrival times and service times, respectively.

In this paper, we assume a Poisson service process with $\mu = 100$ requests per second; therefore $C_S = 1$. In order to online estimate C_A, we further divide a time slot into L time windows (see Figure 3). The algorithm to estimate C_A is shown in Algorithm 3.

6.3. Result Analysis

6.3.1. Cost of Various Instance Provisioning Policies. In this experiment, the length of a phase is set to 1 hr. From

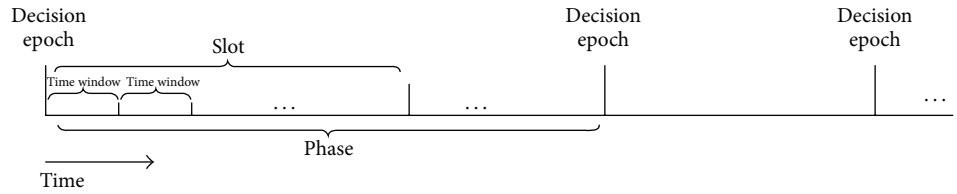

Figure 3: The time structure for simulation. A day is divided into several *phases*. The delay violation probability is evaluated in each phase. A phase is divided into several *slots*. The response delay for each request arrived within a slot is approximated by the mean response delay in this slot. To estimate the mean response delay using the Allen-Cunneen formula, a slot is further divided into several time windows to measure the coefficient of variation of inter-arrival time.

Algorithms 1 and 2, we can obtain that the optimal number of RIS instances is 29. Figure 4 shows the cost of three instance provisioning policies. Consider the following.

(i) In the RIS mode, the ISP should rent 37 instances in all hours of a day since the system must satisfy the peak workload demand. This policy yields 408.576$ per day.

(ii) In the MIS mode, the ISP makes instances provisioning decision in each hour according to the predicted demand; therefore the resource utilization is the highest. Unfortunately, the total daily cost (514.08$) is even higher than the one in the RIS mode.

(iii) In the hybrid mode, the optimal number of RIS instances is 29. Although, in some cases, this is a little waste of resource, the daily cost of this policy is the lowest (360.768$).

6.3.2. Effects of the Rental Granularity. The length of the phase (or interdecision time) in the Amazon EC2 is 1 hour. Here, we vary the length to 2 and 3 hours to study its impact on the daily cost. Figure 5 plots the optimal number of reserved instances in each hour. It goes "smoother" as the length of the interdecision time becomes longer. For example, the numbers of reserved instances for the three rental granularities in time interval $[10, 15]$ are $\{35, 36, 38, 38, 36, 35\}$, $\{35, 37, 37, 37, 37, 35\}$, and $\{37, 37, 37, 37, 37, 37\}$. The mean numbers in time interval $[4, 6]$ and $[7, 9]$ in the 3 hr granularity are 30 and 34, and the counterparts in the 1 hr granularity are 29.67 and 33. This implies that the instance provisioning policy could be more flexible as the interdecision time goes small.

Figure 6 presents the total cost for three rental granularities. It is obvious that the total cost is an increasing function of the length of the interdecision time. However, we can also see that this function is not linear; that is, the marginal cost is shrinking as the length of the interdecision time goes smaller. In production systems, a small interdecision time may induce additional system overhead; therefore there should be a tradeoff between the rental cost and system overhead.

Figure 7 describes the impacts of rental granularity to the delay violation probability. Using instance provisioning policies generated by Algorithms 1 and 2, the target SLA specification is satisfied in all three rental granularities. A

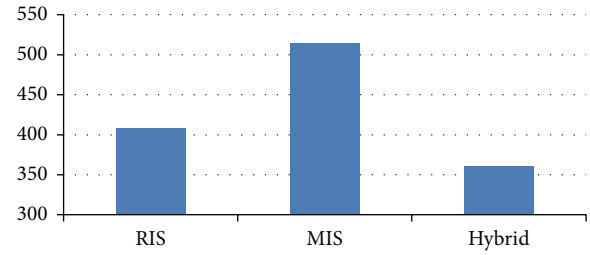

Figure 4: Cost of three instance provisioning policies.

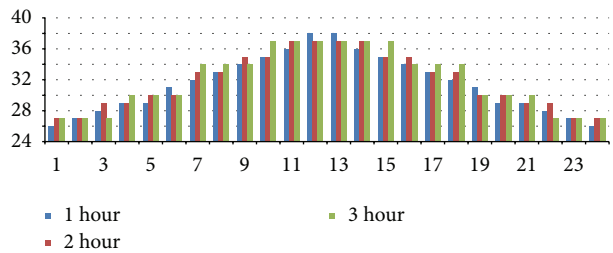

Figure 5: The optimal number of reserved VM instances for various rental granularities.

Table 2: Means and standard deviations for various rental granularities.

Rental granularity	1 hour	2hours	3hours
Mean	0.041215	0.041218	0.040743
standard deviation	0.002773	0.003848	0.006968

more detailed comparison is provided in Table 2. The means of delay violation probability in 1 hr granularity and 2 hr granularity are very close to each other, and the one in 3 hr granularity is relatively small, implying that more resources are reserved. On the other hand, the standard deviation of the delay violation probability decreases as the length of interdecision time goes smaller. Since a small standard deviation implies a more stable response delay, we propose to use 1 hr granularity rental policy in delay- and jitter-sensitive applications such as VoIP and video streaming.

6.3.3. Effects of Function Approximations. Here, we evaluate the effectiveness of two function approximation approaches with 1 hr rental granularity. We can obtain parameters a and

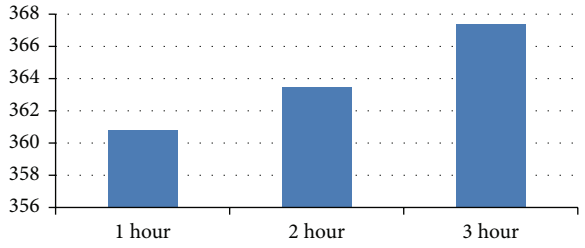

Figure 6: Cost for various rental granularities.

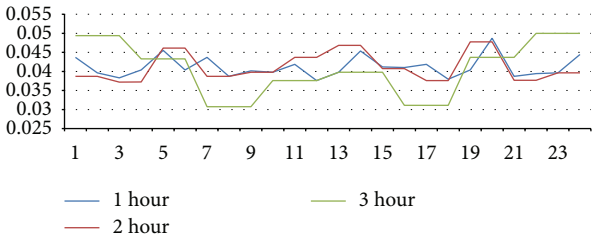

Figure 7: Impacts of rental granularity on the delay violation probability.

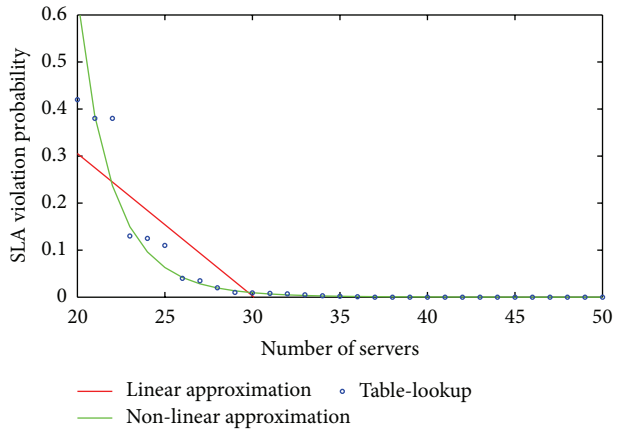

Figure 8: Function approximations for VP_table.

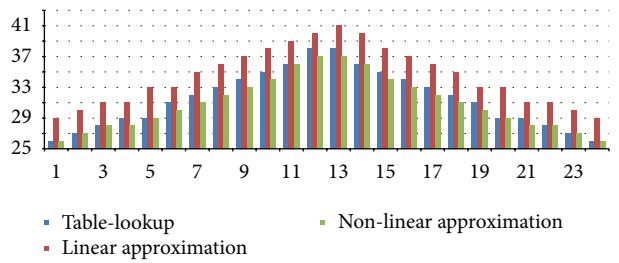

Figure 9: Impacts of function approximation on the instance demand estimation.

b using (13) and (15) for all phases, which are shown in Table 3. Specifically, the results in the first hour are plotted in Figure 8, where we can see that the nonlinear approximation is more accurate than the linear approximation. Figure 9 shows the estimation of VM instance demands. The linear approximation tends to overestimate the demand by 2–4, and the nonlinear approximation underestimates the demand by 0-1. Figure 10 shows the delay violation probability. By using VP_table structure, the delay violation probability is around 4%. The linear approximation approach reduces the delay violation probability to about 1% since it reserves more instances. By contrast, the delay violation probabilities in 13 phases (out of total 24 phases) exceed the target 5% objective. The delay violation probabilities even exceed 9% in the 10th and 16th phases.

The basic instance provisioning algorithm makes the best resource-SLA tradeoff but suffers from the scalability problem. The two function approximation approaches only need to estimate two parameters in each phase. They visit fewer instance provisioning policies and evade the lookup table structure (VP_table); thus the scalability of Algorithms 1 and 2 is enhanced. The effectiveness, however, lies in how well the function approximates the behavior of VP_table. A poor approximation may severely deviate from VP_table and generate a wrong instance provisioning policy which either damages the performance or increases the rental cost. Figures 11 and 12 present the number of RIS instances and total daily rental cost. We can see that the number of RIS instances in the VP_table approach is the same as in the one in the nonlinear approximation approach (29 VMs). The linear approximation approach, although achieves a lower delay violation probability, overestimates the VM instance demand too much (33 VMs).

In order to further evaluate the two function approximation approaches, define the instance deviation D_m and the violation probability deviation D_s as

$$D_m = \sum_{i=1}^{24} \left| K_i^f - K_i \right|,$$
$$D_s = \sum_{i=1}^{24} \left| s_i^f - s_i \right|, \tag{19}$$

where K_i^f and s_i^f denote the number of rented instances (including both RIS and MIS instances) and the violation probability using function approximations and K_i and s_i denote the same parameters but using the VP_table structure. Clearly, smaller D_m and D_s indicate a more accurate approximation. The results are shown in Table 4. The linear approximation achieves a lower violation probability at the expense of a much higher number of instances. In addition, nonlinear approximation has a lower violation probability deviation. Therefore, we purpose to use nonlinear approximation in Algorithms 1 and 2.

7. Conclusions

Dynamic instance provisioning is a key issue for Internet service providers in the cloud computing environment. In this paper, we investigate the coarse-grain (in the order of hours) QoS-aware dynamic instance provisioning problem

Table 3: Parameters for two function approximation approaches.

Decision epoch	Linear approximation		Nonlinear approximation	
	a	b	a	b
1	0.910558	−0.03026	$1.96E+13$	−10.3672
2	1.17268	−0.03832	$3.42E+13$	−10.4338
3	1.31243	−0.04182	$6.23E+13$	−10.5444
4	1.50093	−0.04697	$4.04E+13$	−10.3147
5	1.46158	−0.04394	$1.62E+13$	−9.97221
6	1.59491	−0.04727	$3.68E+12$	−9.43435
7	1.49158	−0.04142	$5.56E+11$	−8.76771
8	1.57272	−0.043	$2.35E+11$	−8.44603
9	1.63012	−0.04319	$1.16E+11$	−8.13732
10	1.6601	−0.04237	$5.02E+10$	−7.84783
11	1.75972	−0.04396	$1.81E+10$	−7.47832
12	1.81333	−0.04415	$6.70E+09$	−7.12764
13	1.81285	−0.04386	$8.02E+09$	−7.17651
14	1.75933	−0.0437	$1.76E+10$	−7.4644
15	1.7186	−0.04436	$4.01E+10$	−7.77709
16	1.63952	−0.04352	$6.46E+10$	−7.9815
17	1.58403	−0.04322	$1.83E+11$	−8.36223
18	1.55035	−0.04373	$1.20E+12$	−9.01639
19	1.54785	−0.0454	$3.54E+12$	−9.41871
20	1.38708	−0.04119	$5.03E+12$	−9.60535
21	1.50875	−0.04732	$8.56E+13$	−10.5575
22	1.31375	−0.04188	$2.37E+13$	−10.2367
23	1.16168	−0.03787	$3.70E+13$	−10.4633
24	0.866371	−0.02836	$1.52E+13$	−10.2863

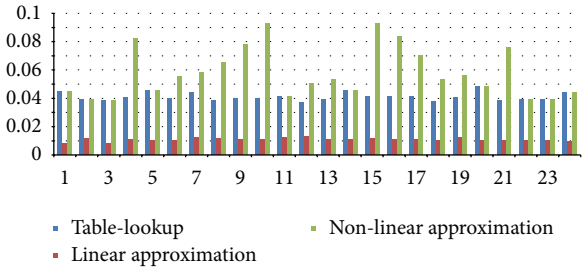

Figure 10: Impacts of function approximation on the delay violation probability.

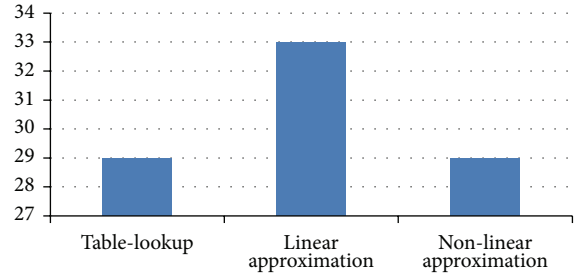

Figure 11: The optimal number of RIS instances.

for interactive workload. The optimization problem in our consideration (see (2)-(3)) is not a traditional optimization problem since the QoS constraint (3) has no analytical form for the self-similar Internet traffic; therefore it cannot be solved using classic methods. We use various approaches, for example, a lookup table and two function approximations to characterize constraint (3). The lookup table approach suffers from the scalability issue, because, in order to obtain a precise estimation of the violation probability in the table, we must visit all possible instance provisioning policies and get sufficient violation probability samples. In contrast, function approximations can predict the performance using a small set of samples. Function approximations (especially nonlinear approximation) address the scalability problem at the expense of a little sacrifice of prediction precision. We conduct extensive simulations to evaluate the effectiveness of the proposed dynamic instance provisioning policy.

Table 4: The instance deviation and the violation probability deviation for two function approximation approaches.

Metric	Approximations	
	Linear	Nonlinear
D_m	70	14
D_s	0.730222	0.407333

Table 5: Revisions for Algorithms 1 and 2 with function approximations.

Revisions for Algorithm 1	Revisions for Algorithm 2
(i) In line 5, there is no need to test each instance provisioning policy in [MIN_NUM, MAX_NUM]. Instead, we can use either approach listed as follows. (1) Define a positive integer variable *stepsize* which is greater than 1, and increase k by *stepsize* rather than 1. (2) Pick a small set of k randomly from [MIN_NUM, MAX_NUM]. (ii) Compute a_i and b_i using (13) and (15) after line 19. (iii) The output of the algorithm is func_app[N].	(i) The input of the algorithm is func_app[N]. (ii) The exhausted search approach (lines 2–7) is replaced by the inverse functions of (8) and (9) as follows. (1) For linear approximation case, $K_i = \left\lceil \frac{x - \text{func\_app}[i] \cdot \text{a}}{\text{func\_app}[i] \cdot \text{b}} \right\rceil$. (2) For nonlinear approximation case, $K_i = \left\lceil \sqrt[\text{func\_app}[i] \cdot b]{\frac{x}{\text{func\_app}[i] \cdot a}} \right\rceil$, where $[x]$ is the smallest integer no less than x.

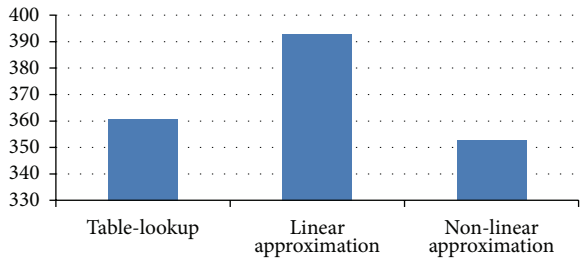

Figure 12: The daily rental cost.

Notations

N: The number of phases in a day to make instance provisioning decisions, that is, the number of decision points
K_i: The number of instances needed in phase i to meet the QoS requirement
C_L: The marginal rental cost for a RIS instance
C_S: The marginal rental cost for a MIS instance

d_i: The delay in phase i
D_{th}: The threshold delay set by the SLA
x: The threshold violation probability set by the SLA.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The work is funded in part by the National Natural Science Foundation of China (NSFC) under Grant no. 61363052, the Inner Mongolia Provincial Natural Science Foundation under Grant nos. 2010MS0913 and 2013MS0920, and the Science Research Project for Inner Mongolia College under Grant nos. NJZY14064 and NJZY13120.

References

[1] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: Eliminating server idle power," in *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '09)*, pp. 205–216, Washington, DC, USA, March 2009.

[2] "A. E. instance launch time," http://aws.amazon.com/ec2/faqs/#Howquicklywillsystemsberunning.

[3] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 1–15, 1994.

[4] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," in *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, Banff, Canada, 2005.

[5] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *roceedings of the IEEE 10th International Symposium on Workload Characterization (IISWC '07)*, Boston, Mass, USA, 2007.

[6] E. Caron and F. Desprez, "Forecasting for grid and cloud computing ondemand resources based on pattern matching," in *Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom '10)*, pp. 456–463, Indianapolis, Ind, USA, 2010.

[7] D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *Proceedings of the IEEE INFOCOM*, pp. 460–468, Orlando, Fla, USA, 2012.

[8] F. Ahmad and T. N. Vijaykumar, "Joint optimization of idle and cooling power in data centers while maintaining response time," in *Proceedings of the 15th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '10)*, pp. 243–256, March 2010.

[9] J. Leverich and C. Kozyrakis, "On the energy (in)efficiency of hadoop clusters," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 1, pp. 61–65.

[10] S. Shen, K. Deng, A. Iosup, and D. Epema, "Scheduling jobs in the cloud using on-demand and reserved instances," in *Euro-Par 2013 Parallel Processing*, vol. 8097 of *Lecture Notes in Computer Science*, pp. 242–254, Springer, Berlin, Germany, 2013.

[11] W. Ming, Y. Jian, and R. Yongyi, "Dynamic instance provisioning strategy in an iaas cloud," in *Proceedings of the 32nd*

Chinese Control Conference (CCC '13), pp. 6670–6675, Xi'an, China, 2013.

[12] D. Deng, Z. Lu, W. Fang, and J. Wu, "Cloudstreammedia: a cloud assistant global video on demand leasing scheme," in *Proceedings of the IEEE 10th International Conference on Services Computing (SCC '13)*, pp. 486–493, Washington, DC, USA, 2013.

[13] W. Fang, Z. Lu, J. Wu, and Z. Cao, "Rpps: a novel resource prediction and provisioning scheme in cloud data center," in *IEEE Ninth International Conference on Services Computing (SCC '12)*, pp. 609–616, Honolulu, Hawaii, USA, 2012.

[14] J. Jiang, J. Lu, G. Zhang, and G. Long, "Optimal cloud resource auto-scaling for web applications," in *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 58–65, Delft, The Netherlands, 2013.

[15] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: evidence and possible causes," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 835–846, 1997.

[16] T. Yoshihara, S. Kasahara, and Y. Takahashi, "Practical time-scale fitting of self-similar traffic with markov-modulated poisson process," *Telecommunication Systems*, vol. 17, no. 1-2, pp. 185–211, 2001.

[17] P. Salvador, R. Valadas, and A. Pacheco, "Multiscale Fitting Procedure Using Markov Modulated Poisson Processes," *Telecommunication Systems*, vol. 23, no. 1-2, pp. 123–148, 2003.

[18] A. T. Andersen and B. F. Nielsen, "A Markovian approach for modeling packet traffic with long-range dependence," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 5, pp. 719–732, 1998.

[19] M. Guazzone, C. Anglano, and M. Canonico, "Exploiting VM migration for the automated power and performance management of green cloud computing systems," in *Energy Efficient Data Centers*, vol. 7396 of *Lecture Notes in Computer Science*, pp. 81–92, Springer, Berlin, Germany, 2012.

[20] D. Bruneo, "A stochastic model to investigate data center performance and qos in iaas cloud computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 560–569, 2013.

[21] T. H. Szymanski, "Low latency energy efficient communications in global-scale cloud computing systems," in *Proceedings of the ACM Workshop on Energy Efficient High Performance Parallel and Distributed Computing (EEHDPC '13)*, pp. 13–22, 2013.

[22] J. Tai, J. Zhang, J. Li, W. Meleis, and N. Mi, "Ara: Adaptive resource allocation for cloud computing environments under bursty workloads," in *Proceedings of the IEEE 30th International Performance Computing and Communications Conference (IPCCC '11)*, pp. 1–8, 2011.

[23] Y. J. Hong, J. Xue, and M. Thottethodi, "Selective commitment and selective margin: techniques to minimize cost in an iaas cloud," in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS '12)*, pp. 99–109, 2012.

[24] A. O. Allen, *Probability, Statistics, and Queueing Theory with Computer Science Applications*, Academic Press, Boston, Mass, USA, 1990.

[25] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains*, Wiley-Interscience, New York, NY, USA, 1998.