Cost Minimization for Service Providers with Impatient Consumers in Cloud Computing

Shuang Wang,Xiaoping Li
School of Computer Science and Engineering
Southeast University
Nanjing 211189, China
Email: wangshuang,xpli@seu.edu.cn

Rubén Ruiz
Grupo de Sistemas de Optimización Aplicada,
Instituto Tecnológico de Informática,
Universitat Politècnica de València,
Camino de Vera s/n, 46022, València Spain
Email: rruiz@eio.upv.es

Yun Wang
School of Computer Science and Engineering
Southeast University
Nanjing 211189, China
Email: ywang_cse@seu.edu.cn

***Abstract*—In this paper, we consider the cost minimization problem for scheduling stochastic service requests to heterogenous servers in cloud computing. Service requests are impatient with different maximizing waiting time. Using queuing theory, a queuing system model is constructed. An algorithm framework is proposed to minimize the cost. The actual expected waiting time of service requests is analyzed. The rejection probability of the system is obtained. Comparing the rejection probability of the system to a given system availability, suitable servers are selected to minimize the cost. Based on the proposed framework, algorithms with different components are compared. Experimental results show that the algorithm with the mixed server selection strategy outperforms the others on efficiency.**

***Index Terms*—Cost minimization; impatient consumers; maximizing waiting time; cloud computing.**

I. Introduction

Cloud computing is a novel computing model enabling service providers to rent resources (CPU, storage, memory) from cloud providers. Cloud providers encapsulate resources as services which mainly include infrastructure as a service (IaaS), platform as a service (PaaS), software as a service (SaaS) [1]. Consumers are impatient if they tend to join the queue only when a short wait is expected and tend to remain in line if the wait has been sufficiently small [2]. Service consumers are impatient with given maximizing waiting time and send requests stochastically to service providers [3], [4]. Service requests have to be processed before the given maximizing waiting time which satisfies quality of service (QoS). Service providers rent servers from cloud provider and construct multiple server systems to process service requests sent by service consumers. Different service consumers have different patience. In this paper, service consumers with the same maximizing waiting time are classified into together. If they wait too long, they will leave the system which increase the rejection probability. Cloud providers provide service by three instance types: reserved instances, on-demand instances and spot instances. There are different instances for servers in cloud centers which corresponds to different prices and price schemes. Therefore, it is urgent for service providers to select suitable servers for impatient service consumers to minimize the cost.

In this paper, we consider the problem of scheduling independent service requests with different maximizing waiting time to heterogeneous servers to minimize the cost for service providers. Requests arrive at the system stochastically and dynamically. The heterogeneity of the servers means that servers have different service rates. Different service rates result in different configurations and prices. For guaranteeing the QoS, the system availability is defined as the probability of an adequate level of service [5]. Service providers rents resources from cloud providers by on-demand instances because service providers may not obtain spot instances in terms of limit bidding price. In addition, the utilization of reserved servers is low since service requests arrive stochastically and dynamically.

Stochastically arriving requests of impatient consumers, heterogeneous servers and different prices lead to a very complex problem. Numbers of heterogeneous servers affects the cost and the response time of service requests. If numbers of heterogeneous servers are too bigger, it increases the cost for on-demand instances. On the other hand, if the numbers of heterogeneous servers are smaller, it increases the rejection probability which cannot satisfy the system availability. In this paper, we consider different classes of service requests. For decreasing the rejection probability, the earliest start-due time first (ESF) similar as earliest deadline first for multiple servers [6] is adopted. The actual expected waiting time of the sorted requests is calculated in terms of the start-due time. According to the actual expected waiting time, the rejected requests are determined. For a given system availability, less on-demand servers are required by assigning the urgent service requests to servers. It decreases the cost. To minimize the cost, the main challenges for the problem under study lie in: (i) Due to the property of maximizing waiting time for service requests, it is important to analyze the actual expected waiting time for service requests. (ii) It is crucial to determine the number of servers because different configurations result in different prices. For minimizing the cost, the contributions of this paper are summarized as follows. (i) A rejection probability determination algorithm is proposed to evaluate the system rejection probability. (ii) A cost minimization framework is proposed to select suitable servers to minimize the cost.

978-1-7281-0350-1/19/$31.00 ©2019 IEEE

The remainder of the paper is organized as follows. The related work is described in Section II. Section III details the model and problem formulation. Optimization methods are proposed in Section IV. Experimental results are given in Section V, followed by conclusions and future research.

II. Related Work

Service requests with constraints for impatient consumers have attracted a considerable research attention recently [3], [7]–[10]. A multiple priority preemptive $M/G/1/./EDF$ model for requests was proposed in [10]. It processed requests with the earliest deadline first. An $M/M/m+D$ queuing model is constructed with service requests with a constant maximizing waiting time D for profit maximization [3]. The maximizing waiting time or deadlines are constant [7]–[10]. It is common for different classes of service consumers to have different impatience in real scenarios. However, only few paper consider service requests for different classes of impatient consumers in cloud computing.

There are many existing papers considering about cost minimization in [4], [7]–[9], [11], [12]. A cost model is developed to address performances/cost tradeoff with homogeneous servers [4]. Genetic algorithm is applied for users' requests to select suitable on-demand virtual machines that will minimize the cost of execution [11]. Dynamic and reactive resource provisioning and load distribution algorithms were proposed which heuristically optimize the overall cost and the response delays [12]. An integer programming model is constructed to determine the right amount of resources for multiple periodical workflow applications for cost minimization [9]. Cost minimization for computational applications on hybrid cloud infrastructures is studied [7]. The purpose is to construct an application scheduling model for determining the optimum dynamic vehicular cloud resources to minimize the cost [8]. The stochastic impatience for different consumers is not considered in [4], [7]–[9], [11]–[13].

To the best of our knowledge, cost minimization to rent suitable heterogeneous servers from cloud providers for service providers with impatient consumers have never been considered simultaneously. All the conditions mentioned above are considered simultaneously in this paper which brings the research results closer to real life scenarios.

III. System Model and Problem Description

A. System Model

The system model is illustrated in Figure 1. Requests are scheduled by the scheduler between the queue and servers. The scheduler controls the processing order of the requests. Different requests have different maximizing waiting time. In this paper, requests are classified into K classes with different maximizing waiting time. Every request in the system has maximizing waiting time. All requests in the same class have the same maximizing waiting time. According to the maximizing waiting time, the start due time of requests is obtained. Using the ESF rule, bigger start due time implies

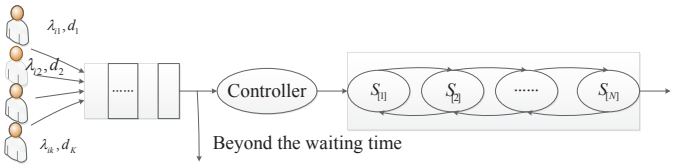

Fig. 1: System Model.

lower priority. The requests with higher priorities are processed earlier than those with lower priorities. When requests arrive in the system, they are added to the queue if the actual expected waiting time is less than the maximizing waiting time. The scheduler dispatches requests in the queue to a idle server according to the priorities.

Cloud providers offer S types of servers. The service rates for servers are independent and exponentially distributed random variables with rates $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_S$. Service providers rent on-demand servers from cloud providers. The price per hour for on-demand servers are $p_1, p_2, \ldots, p_S$ corresponds to service rates μ_1, μ_2, …, μ_S. The total number of rental servers is N. N is dynamic with the change of requests' arrival rates. When the number of requests in the system is less than N, the scheduler just allocates the requests to the servers. New coming request is executed immediately. If the waiting time of new arriving request is bigger than its start due time, it is rejected. Otherwise, the coming requests are pushed to the queue if the number of requests in the system is between N and the system capacity. The request with the highest priority will be processed firstly in the queue. The system availability defined as the probability of an adequate level of service [5] is ξ.

In real scenarios, the requests arrive in the system changing dynamically. For simplicity, we assume that new request arrives at the system following the Poisson process with dynamic arrival rates λ_i. λ_i is made up of small streams $\lambda_{i1}, \lambda_{i2}, \ldots, \lambda_{iK} (i = 0, \ldots, N+R-1)$ when there are i requests in the system. The maximizing waiting time corresponds to $d_1, d_2, \ldots, d_K$. For ESF rule, priorities are determined by maximizing waiting time $d_1 \leq d_2 \leq \ldots \leq d_K$ and arrival time of the currently arriving requests. With time going, the priority of the K^{th} class requests increases. Denote R as the queue capacity. According to above assumptions, it forms Markov process. The system model can be constructed as an $M[K]/M[S]/N/N+R$ queueing model with different classes of requests. In this paper, if the number of requests in the system is less than N, no request would be rejected because all requests are within the start due time. The waiting requests are served by the priority order in the queue. Those waiting requests with the actual expected waiting time exceeding start due time are rejected. It results in the actual arrival rate changing (less than the arrival rate). The priority definition manner exerts influence on the number of requests in the queue. It leads to different actual arrival rates. Denote the rejection probability as P_R. When some requests are rejected, the actual arrival rate changes because the waiting time is

beyond the maximizing waiting time.

B. Performance Analysis

For obtaining the rejection probability, the actual expected waiting time of requests in queue is analyzed. Comparing the actual expected waiting time of different class requests to the maximizing waiting time, the rejection rates are calculated. According to the rejection rates, the actual arrival rates are computed. In addition, the balance equations are obtained according to the steady state probabilities. The infinitesimal generator matrix is obtained according to which the rejection probability is obtained.

1) Waiting Time Determination: In this paper, we assume that dynamic arriving requests of every class is Poisson distributed. Denote $S(x)$ as a sign function where $x \leq 0, S(x) = 0$ and $x > 0, S(x) = 1$. Denote $W_{(i+1)k}$ as the expected waiting time of new request of the k^{th} class (which is called the tagged request in this paper) when there are i requests in the system using ESF rule. Denote $\mu_{[i]}(i = 1, \ldots, N)$ as the the selected servers among the S server types. If there are $i(i < N)$ requests in the system, new request of the k^{th} class is processed immediately, i.e. $W_{(i+1)k} = 0$. All servers are turned on if the queue is not empty. $W_{(N+1)k}$ denotes the residual service time which is the expected delay time to finish the current request when a request arrives. It is calculated according to [14]

$$W_{(N+1)k} = \sum_{j=1}^{K} \frac{2\lambda_{ij}}{(\sum_{p=1}^{N} \mu_{[p]})^2} \tag{1}$$

The difference between maximizing waiting time of two classes of arrival streams is represented as $D_{j,k} = d_j - d_k \ \ (j > k)$. Considering the newly arriving request to the k^{th} class as the tagged request when there are i requests in the system. $N_{j,k}^{i}$ and $M_{j,k}^{i}$ denote the expected numbers of requests of the j^{th} class arriving at the system before and after the tagged request, respectively, while they are served prior to the tagged request. When there are i requests in the system, the waiting time of new request of the k^{th} $(k = 1, 2, \ldots, K)$ class is as follows.

$$W_{(i+1)k} = S(i-N)W_{Nk} + S(i-N-1)\sum_{j=1}^{K} \frac{(N_{j,k}^{i} + M_{j,k}^{i})}{\sum_{p=1}^{N} \mu_{[p]}} \tag{2}$$

Since priorities are determined by start due time, the request with the earliest start due time has the highest priority for processing. There are three kinds of requests from different classes being served before the tagged request.

(i) Requests from the higher priority classes arriving earlier than the tagged request are served prior to the tagged request. The number of the j^{th} $(j = 1, \ldots, k)$ class requests is obtained as follows.

$$N_{j,k}^{i} = \sum_{l=N}^{i} \lambda_{lj} W_{lj} \tag{3}$$

(ii) Requests from the lower priority classes arriving prior to the tagged request are served prior to the tagged request because of the earlier start due time. The number of the j^{th} $(j = i+1, \ldots, K)$ class requests is calculated as follows.

$$N_{j,k}^{i} = \sum_{l=N}^{i} \lambda_{lj} \max(0, W_{lj} - D_{j,k}) \tag{4}$$

(iii) Requests from the higher j^{th} classes arrive at the system after the tagged request. They are served prior to the tagged request because of the earlier start due time. The number of requests denoted as i' are determined by $R, \sum_{p=1}^{N} \mu_{[p]}$ and $D_{k,j}$. If $D_{k,j} \geq \frac{1}{\sum_{p=1}^{N} \mu_{[p]}}$, i' is calculated as $\min(\lfloor D_{k,j} N \mu \rfloor, R) + i$. Otherwise, there is no classes j^{th} request arriving in the system. The number of the j^{th} $(j = 1, \ldots, i-1)$ class request is computed as follows.

$$M_{j,k}^{i} = S(i'-i)\sum_{l=i}^{i'} \frac{\lambda_{lj}}{\lambda_l} \tag{5}$$

Based on Equation (2), the actual expected waiting time of new request of the k^{th} $(k = 1, 2, \ldots, K)$ class is calculated as follows when there are i requests in the system.

$$\begin{aligned} W_{(i+1)k} =& S(i-N)W_{Nk} + \\ & S(i-N-1)\sum_{j=1}^{k}\sum_{l=N}^{i} \frac{\lambda_{lj}}{\sum_{p=1}^{N} \mu_{[p]}} W_{lj} + \\ & S(i-N)\sum_{j=k+1}^{K}\sum_{l=N}^{i} \frac{\lambda_{lj}}{\sum_{p=1}^{N} \mu_{[p]}} \max(0, W_{lj} - D_{j,k}) \\ & + S(i-N)S(i'-i)\sum_{j=1}^{k-1}\sum_{l=i}^{i'} \frac{\lambda_{lj}}{\lambda_l \sum_{p=1}^{N} \mu_{[p]}} \end{aligned} \tag{6}$$

2) Performance Evaluation: Denote matrix $\mathbf{\Lambda}$ as the dynamic arrival rates of requests of different classes for different states in the system. Denote $\lambda_i^a (i = 0, 1, \ldots, N+R-1)$ as the actual arrival rates. Let $\vec{P} = (P_0, P_1, \ldots, P_{N+R})$ be the probability vector of the state space. The steady state equations are obtained as follows in terms of [14].

$$\begin{cases} \lambda_0^a P_0 = \mu_{[1]} P_1 \\ \lambda_0^a P_0 + (\mu_{[1]} + \mu_{[2]}) P_2 = (\lambda_1^a + \mu_{[1]}) P_1 \\ \ldots \\ \lambda_{N-2}^a P_{N-2} + \sum_{p=1}^{N} \mu_{[p]} P_N = (\lambda_{N-1}^a + \sum_{p=1}^{N-1} \mu_{[p]}) P_{N-1} \\ \lambda_{N-1}^a P_{N-1} + \sum_{p=1}^{N} \mu_{[p]} P_{N+1} = (\lambda_N^a + \sum_{p=1}^{N} \mu_{[p]}) P_N \\ \ldots \\ \lambda_{N+R-1}^a P_{N+R-1} = \sum_{p=1}^{N} \mu_{[p]} P_{N+R} \end{cases} \tag{7}$$

The infinitesimal generator matrix is described as follows which corresponds to steady state equations.

$$\mathbf{Q} = \begin{pmatrix} -\lambda_0^a & \lambda_0^a & & & \\ \mu_{[1]} & -\mu_{[1]} - \lambda_0^a & \lambda_0^a & & \\ & & \ddots & & \\ & & & \sum_{p=1}^{N} \mu_{[p]} & -\sum_{p=1}^{N} \mu_{[p]} \end{pmatrix} \quad (8)$$

The balance vector equation is shown as follows.

$$\begin{cases} \vec{P}\mathbf{Q} = \mathbf{0} \\ \sum_{j=0}^{j=N+R} P_j = 1 \end{cases} \quad (9)$$

According to Equation (9), the steady state probabilities are obtained for the system states. The performance metrics of the cloud center, such as the expected actual arrival rate E_λ and the rejection probability P_R could be determined respectively as follows.

$$\begin{cases} E_\lambda = \sum_{j=0}^{N+R-1} P_j \lambda_j^a \\ P_R = 1 - \frac{E_\lambda}{\sum_{j=0}^{N+R-1} P_j \lambda} + P_{N+R} \end{cases} \quad (10)$$

C. Problem Description

Service consumers are impatient with given maximizing waiting time. For a given system availability ξ, service requests are processed by on-demand servers. To minimize the rental cost, service providers select appropriate on-demand servers. Denote $\vec{n}$ as a $S-$dimension vector where $\vec{n}_i$ represents the number of selected i^{th} type servers. N is equal to $\sum_{i=1}^{S} \vec{n}_i$. The cost minimization problem for service providers with impatient servers is described as follows.

$$\begin{aligned} &\min \sum_{i=1}^{S} \vec{n}_i p_i \\ &s.t. \\ &\sum_{i=1}^{S} \vec{n}_i u_i \geq E_\lambda, \\ &p_i > 0, \\ &P_R \leq 1 - \xi, \\ &\vec{n}_i \geq 0. \end{aligned} \quad (11)$$

IV. Proposed Methods

According to the constructed model, the actual expected waiting time is analyzed. By comparing the actual waiting time with maximizing waiting time, the rejection probability P_R is obtained. Comparing P_R to the system availability ξ, servers are selected to minimize the cost. The cost is minimized by the proposed algorithm framework in Algorithm 1 which includes two components: Determine the rejection probability. Select servers to minimize the cost in terms of system availability ξ. Due to the complexity of the considered problem, there are no similar scenarios in existing papers. When selecting servers, three different rules are proposed.

(i) Cheapest servers are always selected.
(ii) Fastest servers are chosen.
(iii) The minimal price per service rate is chosen firstly. The number of servers is calculated in terms of the system availability. Severs are replaced by other suitable servers or not to minimize the cost with the constraint of the system availability.

The system always choose the fastest servers from selected servers firstly to minimize the rejection probability.

Algorithm 1: Cost Minimization (CM)

1 Determine the rejection probability;
2 Select servers to minimize the cost using RULE;

A. Rejection Probability Determination

According to the analysis in subsubsection III-B1, the actual expected waiting time is analyzed. When there are $N+1$ service requests in the system, the residence service time of service requests is calculated by Equation (1). The actual expected waiting time is calculated in terms of Equation (6). The actual arrival rates for the queue could be obtained by comparing actual expected waiting time to maximizing waiting time. The arrival rates $\lambda_{ij}(i = 0, 1, \ldots, N+R-1; j = 1, \ldots, K)$ of different classes are generated randomly and stored in matrix $\mathbf{\Lambda}$. The arrival rates $\lambda_i(i = 0, 1, \ldots, N+R-1)$ is computed in terms of $\mathbf{\Lambda}$ and stored in $\vec{\Lambda}'$. The actual arrival rates $\lambda_i^a(i = 0, 1, \ldots, N+R-1)$ are obtained comparing the actual expected waiting time to the maximizing waiting time. The steady state probabilities of states are computed. According to the steady state probabilities and the system states, the rejection probability is obtained. The rejection probability determination procedure is described in Algorithm 2. The time complexity of the Algorithm 2 is $O(KR^2)$.

B. Server Selection

During server selection, there are three rules. Denote CM_C, CM_F, CM_M as different rules corresponding to the cheapest, fastest and mixed servers respectively. $\vec{C}^p$ is indicated as the unit price per service rate of servers in different types. According to the given system availability ξ, the range of the rejection probability P_R is computed. Numbers of servers using different rules are calculated in terms of the system availability ξ and the rejection probability P_R. When using the CM_M rule, servers of the minimal unit price per service rate are chosen. Servers are replaced by other servers which satisfy the system availability ξ meanwhile minimizing the cost. $\vec{n}^2$ is represented as the new server selection when the server is replaced by cheaper servers. $\vec{n}^3$ is indicated as the new server selection when servers are substituted for faster servers. C_2, C_3 are denoted as the new cost after servers are replaced. Comparing C to C_2, C_3, the cost for rental servers is improved. The server selection procedure is described in Algorithm 3. The time complexity of Algorithm 3 is $O(\max(S, N_1, N_2 S))$.

Algorithm 2: Rejection Probability Determination Algorithm

Input: $K, N, R, \mathbf{\Lambda}, \vec{\Lambda}', \vec{n}, \mu_1, \ldots, \mu_S, d_1, d_2, \ldots, d_K$

$[\mathbf{W}]_{(N+R)\times K} \leftarrow \mathbf{0}$;
for $i = N$ **to** $N + R - 1$ **do**
  for $j = 1$ **to** K **do**
    Calculate $\mathbf{W}_{(N+1)j}$ using Equation (1);
  for $k = 1$ **to** K **do**
    $s_1 \leftarrow \mathbf{W}_{(N+1)j}, s_2 \leftarrow 0, s_3 \leftarrow 0$;
    for $j = 1$ **to** k **do**
      for $l = N + 1$ **to** i **do**
        $s_1 \leftarrow s_1 + \frac{\mathbf{\Lambda}_{lj}\mathbf{W}_{lj}}{\sum_{i_1=1}^{N}\mu_{[i_1]}}$;
    for $j = k + 1$ **to** K **do**
      for $l = N + 1$ **to** i **do**
        $s_2 \leftarrow s_2 + \frac{\mathbf{\Lambda}_{lj}\max(0,\mathbf{W}_{lj}-d_j+d_k)}{\sum_{i_1=1}^{N}\mu_{[i_1]}}$;
    for $j = 1$ **to** $k - 1$ **do**
      $i' \leftarrow \min(D_{kj}\sum_{i_1=1}^{N}\mu_{[i_1]}, R)$;
      for $l = i + 1$ **to** i' **do**
        $s_3 \leftarrow s_3 + \frac{\mathbf{\Lambda}_{lj}}{\vec{\Lambda}'_l\sum_{i_1=1}^{N}\mu_{[i_1]}}$;
    Calculate $\mathbf{W}_{(i+1)k}$ using Equation (6);
for $i = N + 1$ **to** $N + R$ **do**
  for $j = 1$ **to** K **do**
    if $W_{ij} > d_j$ **then**
      $\lambda_i^a \leftarrow \lambda_i^a - \lambda_{ij}$;
Calculate the steady state probability using Equation (9);
Compute P_R using Equation (10);
return P_R;

Algorithm 3: Server Selection Algorithm

Input: RULE, S, C

$s \leftarrow S$,$C^p_{min} \leftarrow \vec{C}^p_1$,$P_R \leftarrow 1$;
if *RULE*=CM_C **then**
  $j \leftarrow 1$;
if *RULE*=CM_F **then**
  $j \leftarrow S$;
if *RULE*=CM_M **then**
  for $i = 1$ **to** S **do**
    if $C^p_{min} > \vec{C}^p_i$ **then**
      $C^p_{min} \leftarrow \vec{C}^p_i$, $j \leftarrow i$;
$N_1 \leftarrow \lfloor \frac{\lambda}{\mu_j} \rfloor$, $C \leftarrow N_1 p_j$;
while $P_R > 1 - \xi$ **do**
  $N_1 \leftarrow N_1 + 1$,$\vec{n}_j \leftarrow N_1$,$C \leftarrow C + p_j$;
  Calculate P_R using Algorithm 2;
if *RULE*=CM_M **then**
  $\vec{n}^2_j \leftarrow N_1$, $C_2 \leftarrow C$,$\vec{n}_j \leftarrow N_1 - 1$,$s \leftarrow j - 1$,$N_2 \leftarrow 0$, $C \leftarrow C - p_j$;
  Calculate P_R using Algorithm 2;
  while $s > 1$ **do**
    for $i = 1$ **to** s **do**
      if $C^p_{min} > \vec{C}^p_i$ **then**
        $C^p_{min} \leftarrow \vec{C}^p_i$, $q \leftarrow i$;
    while $P_R > 1 - \xi$ **do**
      $N_2 \leftarrow N_2 + 1$,$\vec{n}_j \leftarrow N_2$,$C \leftarrow C + p_q$;
      Calculate P_R using Algorithm 2;
    $\vec{n}_q \leftarrow N_2 - 1$,$s \leftarrow q - 1$;
  for $i = j + 1$ **to** S **do**
    $N_2 \leftarrow \lceil \frac{\mu_i}{\mu_j} \rceil$,$C_3 \leftarrow C - N_2 p_j + p_i$;
    $\vec{n}^3_i \leftarrow 1$,$\vec{n}^3_j \leftarrow \vec{n}_j - N_2$;
    Calculate P_R using Algorithm 2;
    if $P_R \leq 1 - \xi$ & $C < C_3$ **then**
      $\vec{n} \leftarrow \vec{n}^3$,$C \leftarrow C_3$;
  if $C < C_2$ **then**
    $\vec{n} \leftarrow \vec{n}^2$,$C \leftarrow C_2$;
return $\vec{n}, C$;

V. Experiment Analysis

There are several variants for parameters of the proposed framework. We calibrate them first. All tested algorithms are coded in Matlab R2018a and run on an Intel Core i5-3470 CPU @3.20GHz with 8 GBytes of RAM.

Similar to the problems on profit maximization on cloud computing [3], the parameters of the proposal are given to analyze the system. In this paper, parameters are calibrated on testing instances. The involved instance parameters are the class of requests $K \in$\{5,10,15,20\}, the range of arrival rates $\lambda \in$\{(0,1], (1,2],(2,3],(3,4]\}, the range of queue capacity $R \in \{10, 20, 30, 40\}$, the range of the system availability $\xi \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$. For analyzing the problem easily Similar to [3], we assume that $S = 10, \mu_1 = 0.2, \mu_2 = 0.3, \mu_3 = 0.7, \mu_4 = 1.1, \mu_5 = 1.3, \mu_6 = 1.7, \mu_7 = 1.9, \mu_8 = 2.3, \mu_9 = 2.9, \mu_{10} = 3.1, p_i = i(i = 1, \ldots, 10), d_i = \frac{\lambda}{\mu_1} i(i = 1, \ldots, K)$. Five instances are randomly generated for each group of arrival rates λ. Therefore, there are 1600 instances for the above 320 combinations.

Experimental results are analyzed by the multi-factor analysis of variance (ANOVA) technique. The three main hypotheses (normality, homoscedasticity, and independence of the residuals) are checked from the residuals of the experiments. All three hypotheses are acceptable from the analysis. The p-values are less than 0.05 which means that all studied factors have a significant effect on the response variables at the 95% confidence level with the ANOVA.

The means plot of the four studied factors on the cost C with 95% Tukey Honestly Significant Difference (HSD) intervals are shown in Figure 2. From Figure 2, it can be observed that the cost C increase with the increase of K because it increases the arrival rates of service requests. The cost C increases with

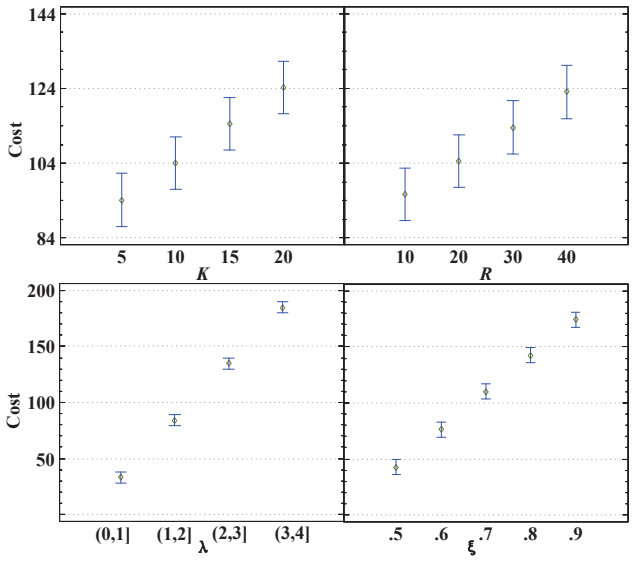

Fig. 2: Means plot of the four studied system parameters with 95% confidence level Tukey HSD intervals.

TABLE I: Algorithm comparison.

	CM_M	CM_F	CM_C	$Random$
C	109.164	118.813	131.456	131.056
CPU time	0.0016	0.0014	0.0013	0.0018

the increase of the queue capacity R. It lies in the reason that the maximizing waiting time of service requests are constants. Therefore, more rental servers are needed to process requests in the queue. The arrival rates λ has a big impact on C because more service requests requires more on-demand servers. The system availability ξ has great influence on the cost C because higher ξ means lower rejection probability which require more servers to reduce the rejection probability.

$Random$ algorithm which choose servers randomly is compared with the proposed algorithms. 6400 random instances are conducted with results shown in Table I. Table I illustrates the CM_M Algorithm performs better than CM_F Algorithm while the cost of CM_F is less than CM_C. $Random$ algorithm is a little better than the CM_C algorithm. However, CM_C requires the least CPU time.

VI. Conclusions and Future Work

In this paper, Markov process is adopted to construct the queueing model to analyze the actual expected waiting time of service requests in different classes. The rejection probability of the system is calculated in terms of the actual expected waiting time and the maximizing waiting time of service requests. A CM algorithm framework is proposed to select suitable servers to minimize the cost for service providers. Comparing different algorithms, the cost is minimized.

The problem on cost minimization problem for heterogeneous cloud centers is worth studying because it is more general and practical in real cloud environments. It is important to distinguish different service consumers with maximizing waiting time which can increase the QoS of the system. In future, the maximizing waiting time is relaxed to be stochastic rather than constant.

Acknowledgment

This work is supported by the National Key Research and Development Program of China (No. 2017YFB1400801), the National Natural Science Foundation of China (Nos. 61572127, 61872077, 61832004) and Collaborative Innovation Center of Wireless Communications Technology. Rubén Ruiz is partly supported by the Spanish Ministry of Economy and Competitiveness, under the project "SCHEYARD-Optimization of scheduling problems in container yards" (No. DPI2015-65895-R) partly financed with FEDER funds.

References

[1] P. D. Kaur and I. Chana, "A resource elasticity framework for QoS-aware execution of cloud applications," *Future Generation Computer Systems*, vol. 37, no. 7, pp. 14–25, 2014.

[2] D. Gross, J. F. Shortie, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory, Fourth Edition*, 2008.

[3] J. Mei, K. Li, A. Ouyang, and K. Li, "A profit maximization scheme with guaranteed quality of service in cloud computing," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3064–3078, 2015.

[4] Y. J. Chiang, Y. C. Ouyang, and C. H. Hsu, "Performance and cost-effectiveness analyses for cloud services based on rejected and impatient users," *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 446–455, 2016.

[5] G. Bolch, S. Greiner, H. D. Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation With Computer Science Applications, Second Edition*. Wiley-Interscience, 2006.

[6] J. Singh and S. P. Singh, "Schedulability test for soft real-time systems under multiprocessor environment by using an earliest deadline first scheduling algorithm," *Computer Science*, vol. 46, no. 7, pp. 29–38, 2012.

[7] M. Malawski, K. Figiela, and J. Nabrzyski, "Cost minimization for computational applications on hybrid cloud infrastructures," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1786–1794, 2013.

[8] L. Aminizadeh and S. Yousefi, "Cost minimization scheduling for deadline constrained applications on vehicular cloud infrastructure," in *International Econference on Computer and Knowledge Engineering*. Mashhad, Iran: IEEE, October 2014, pp. 358–363.

[9] L. Chen, X. Li, and R. Ruiz, "Resource renting for periodical cloud workflow applications," *IEEE Transactions on Services Computing*, vol. PP, 2017.

[10] V. G. Abhaya, Z. Tari, P. Zeephongsekul, and A. Y. Zomaya, "Performance analysis of EDF scheduling in a multi-priority preemptive M/G/1 queue," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2149–2158, 2014.

[11] S. Kansal, H. Kumar, S. Kaushal, and A. K. Sangaiah, "Genetic algorithm-based cost minimization pricing model for on-demand IaaS cloud service," *Journal of Supercomputing*, no. 2, pp. 1–26, 2018.

[12] N. Grozev and R. Buyya, "Multi-cloud provisioning and load distribution for three-tier applications," *Acm Transactions on Autonomous and Adaptive Systems*, vol. 9, no. 3, pp. 1–21, 2014.

[13] J. Cao, H. Kai, K. Li, and A. Y. Zomaya, "Optimal multiserver configuration for profit maximization in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1087–1096, 2013.

[14] Kleinrock and Leonard, *Queueing systems*. Wiley, 1975.