Real-Time Soft Resource Allocation in Multi-Tier Web Service Systems

Xudong Zhao*, Jiwei Huang‡, Lei Liu†, Yuliang Shi †, Shijun Liu †, Calton Pu§, and Lizhen Cui* †
*School of Software Engineering, Shandong University, Jinan, China
†School of Computer Science and Technology, Shandong University, Jinan, China
‡State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing, China
§College of Computing, Georgia Institute of Technology, Atlanta, USA
Email: sdu_zxd@163.com, huangjw@bupt.edu.cn, {l.liu, shiyuliang, lsj, clz}@sdu.edu.cn, calton.pu@cc.gatech.edu

***Abstract*—Soft resource allocation is an important factor of system configuration which plays a critical role in guaranteeing the performance of multi-tier web service systems. There is a tradeoff between real-time performance and resource consumption, and thus the real-time adjustment of soft resource allocation in response to dynamic workload is quite challenging. In this paper, we propose a real-time soft resource allocation method that integrates both model-based analysis and real-time optimization. Specifically, a multi-tier web service system is firstly formulated by a queueing network model, and theoretical analyses are provided. Then, an optimization approach for real-time soft resource allocation is designed by applying sliding window techniques, in order to cope with dynamic workloads and performance demands. Based on the RUBiS benchmark system, model parameters are obtained by measurements and the efficacy of our approach is finally validated.**

***Keywords*-soft resource allocation; queueing network model; sliding window, real-time; web service system;**

I. Introduction

Web service is an emerging technique that enables interoperable machine-to-machine interactions over the Internet. With the growing popularity of web services that have been widely applied in human lives, the quality of service (QoS) has become an important requirement, especially for some critical applications such as traffic control, and military applications. How to guarantee and optimize the QoS of web services is a hot topic in both academia and industry.

Recently, most of the Internet applications like eBay [1] that host web services are implemented by multi-tier architectures. For example, a web service system implemented by Java commonly consists of three tiers. The top one comprises Apache HTTP servers handling user requests via interactive websites, the second tier consists of Tomcat servers that perform complex business logics, while the database servers in the bottom tier handle all the data queries and operations. The design of the system architectures and the optimization of system parameters are critical to the QoS of the applications being hosted by the systems.

For quite a long time, the optimization of QoS was studied from hardware layer, by allocating appropriate hardware resources to different services or systems in order to achieve high efficiency while guaranteeing the SLA. Such efforts can also be generalized into the virtualized environments such as cloud computing, in which the management of virtualized (hardware) resources can be optimized and has been proved to be effective for improving the QoS of web services running on the virtual machines [2]. In recent years, however, some other scholars have found that soft resource is another important factor affecting the performance, and proper soft resource allocation is able to fully utilize the hardware capabilities and thus improve the QoS with nothing cost by hardware upgrade or reconfiguration [3].

Meanwhile, dynamic workload is another critical challenge for QoS optimization. It has been shown by [4] that the workload of web systems varies dramatically with different time periods of a day, and thus the systems should react dynamically to guarantee the real-time performance. To this end, since it is impossible to adjust hardware configuration continuously in response to real-time workload, dynamic soft resource allocation should be well studied accordingly.

In most of the existing researches, soft resource allocation is studied from a static aspect. A seminal work presented by Wang et al. [3] studied such problem experimentally and finally found the optimal strategy from numerous offline experiments with different parameter settings. In our previous work [5], we integrated both theoretical analysis and real-life experiments, and applied the simulated annealing algorithm to obtain the optimal soft resource allocation with a specific workload. Nevertheless, How to dynamically allocate soft resources in real time still remains unexplored.

In this paper, we make an attempt at filling this gap. A real-time soft resource allocation method, which integrates both model-based analysis and experimental study, is proposed in response to dynamic workloads and performance demands in multi-tier web service systems. Queueing network model is applied for theoretically formulating the multi-tier web service system and then an optimization approach for real-time allocation is put forward. Detailed analyses of performance indicators are also given. Based on the RUBiS benchmark system, model parameters are obtained by conducting massive experiments and the efficacy of our method is validated by experimental evaluation.

The contributions of this paper are as follows:

978-1-5386-0752-7/17 $31.00 © 2017 IEEE
DOI 10.1109/ICWS.2017.57

- A multi-tier web service system with limited processing capacity is formulated by a queueing network model. Each server of different tiers is formulated as an M/M/N/K queueing.
- An optimization approach on obtaining model parameters for real-time soft resource allocation is applied in order to address the challenge of dynamic workloads.

The remainder of the paper is organized as follows. Section II gives the related work of studying performance with different aspects. Section III introduces the analytical model of multi-tier web service systems and the optimization approach for real-time soft resource allocation. Based on the RUBiS benchmark system, the accuracy and effectiveness of our model and approach are validated by experimental evaluations in Section IV. Section V concludes the paper.

II. Related Work

Due to the wide application of web service systems, the performance of these systems has been widely concerned and extensively studied. Multi-tier architecture has also been widely used in the these systems. Three-tier and four-tier architecture are the two most commonly used architectures. In the three-tier architecture, the system includes web server tier, application server tier and database server tier, while in the four-tier architecture, it has an addition tier of clustering middleware between application server tier and database server tier. Understanding the behaviour and improving the performance of multi-tier web service systems have always been the focus of many studies.

Due to the application of virtualization and cloud technologies, many previous works [6]–[13] have focused on the study of system performance through changing the number of virtual machines or the amount of different resources. Babu et al. [6] studied the problem of resource utilization in cloud environments and proposed a genetic algorithm to solve the heuristic of resource allocating problem. In [8], by considering various cost on the cloud, the authors presented a resource provisioning policy to find the most cost optimal setup of virtual instances for fulfilling the incoming workload. Liu et al. [13] proposed an efficient mechanism to find the optimal number of VMS for improving system performance in the federated cloud environment.

Not only hardware resources can affect the system performance, soft resources will also influence the usage of hardware resources, and thus have a significant impact on the performance of servers and systems. Wang et al. [3] have studied the impact of soft resources on n-tier applications, and proposed an experiment based approach for optimizing soft resource allocation.

In order to better study the behavior of the system and analyze the performance, many previous works [14]–[17] have established theoretical model to facilitate the research. Urgaonkar et al. [14] applied closed queueing model to model a multi-tier application for capture the behaviour of different tiers and application idiosyncrasies, and then used MVA algorithm to evaluate the system performance.

For studying QoS, Ma et al. [18] presented a accurate prediction algorithm for unknown web service QoS values. Win et al. [19] proposed a self-adaptive web service discovery mechanism by using semantically ontology techniques for satisfying QoS requirements. Wang et al. [20] presented a multi-user web service selection framework to select feasible services for meeting the QoS demands of different users.

In this paper, we formulate a multi-tier web service system by using a queueing network model. Then an optimization approach for real-time soft resource allocation is presented to cope with dynamic workloads and performance demands.

III. Analytical Model and Optimization Approach of Multi-tier Web Service Systems

In this section, a multi-tier web service system is formulated as a queueing network model. Single server is formulated as a queueing system to describe the request processing with limited soft resource in one server, and queueing network model is used to formulate the whole multi-tier web service system. Sliding window based optimization approach is proposed to allocate soft resource in real time for addressing the challenge of dynamic workloads and performance demands. Detailed analyses are also given in the following description.

A. *Queueing System of Single Server*

Chlebus et al. [21] has shown that web browsing request arrivals above the session level can be modeled as Poisson distribution in web service systems. Consequently, the request arrival of one server also can be regarded as Poisson distribution and formulated as inputs of queueing system. The soft resources of threads or processes are created to handled the arrived requests. Because of the limited process capacity of the server, some requests will be dropped and not be handled. Assuming the threads or processes in a server are essentially homogenous with the same average service rates, therefore, M/M/N/K queueing, in which K represents the queueing length of the queueing system, can be used to formulate the request processing in one server.

Fig. 1 shows the formulated single sever through using M/M/N/K queueing system. As shown in the Figure, λ represents the total request arrival rate of the server, N is the number of threads or processes, i.e., the allocation of soft resources, μ is the average service rate of each thread or process, the gray grids represent the requests in the system. Q indicates a finite queue length for waiting, i.e., the maximum queueing length for incoming requests when all threads are busy. As the queue is full, any subsequent arrivals will be rejected and dropped, P^{drop} denotes the probability of a request is dropped.

Fig. 2 demonstrates the service process of the M/M/N/K queueing system. The state diagram is a continuous-time Markov chain, the state represents the number of requests

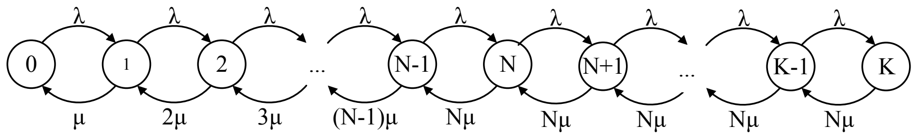

Figure 2. State diagram for the M/M/N/K queueing system of single server

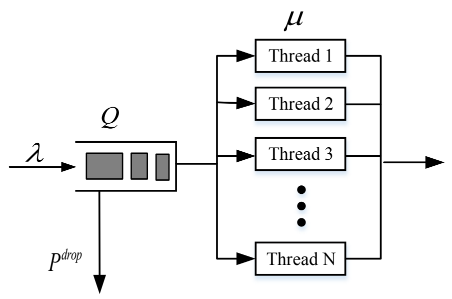

Figure 1. M/M/N/K queueing system of single server

in the queueing system and the maximum state is K, which is the maximum capacity of the system. The arrival rate is λ, the service rate is $N\mu$ when the number of requests in the server is more than N.

When the server is in steady state, the probability in each state can be expressed as $P = [P_0, P_1, ..., P_K]$, which can be calculated as follows:

$$P_n = \frac{1}{n!}(\frac{\lambda}{\mu})^n P_0, 0 \le n \le N \tag{1}$$

$$P_n = \frac{1}{N!N^{n-N}}(\frac{\lambda}{\mu})^n P_0, N+1 \le n \le K \tag{2}$$

$$\sum_{n=0}^{K} P_n = 1 \tag{3}$$

The ρ is defined as the service intensity, which can be calculated by Eq. (4).

$$\rho = \frac{\lambda}{N\mu} \tag{4}$$

Therefore, the probability in state 0 is:

$$P_0 = \begin{cases} [\sum\limits_{n=0}^{N-1} \frac{(N\rho)^n}{n!} + \frac{(N\rho)^N}{N!}\frac{1-\rho^{K-N+1}}{1-\rho}]^{-1}, \rho \ne 1 \\ [\sum\limits_{n=0}^{N-1} \frac{N^n}{n!} + \frac{N^N}{N!}(K-N+1)]^{-1}, \rho = 1 \end{cases} \tag{5}$$

The probability P^{drop} is equal to the probability in state K, that is:

$$P^{drop} = P_K \tag{6}$$

$\widetilde{\lambda}$ represents the effective arrival rate that the incoming requests can be handled by the server, this parameter satisfies the following equation.

$$\widetilde{\lambda} = (1 - P^{drop})\lambda \tag{7}$$

The average queue length L of the queueing system, i.e., the average number of requests in the server, can be obtained by Eq. (8).

$$L = \begin{cases} N\rho(1-P_K) + \sum\limits_{n=1}^{K-N} nP_{n+N}, \rho \ne 1 \\ N\rho(1-P_K) + \sum\limits_{n=1}^{K-N} n\frac{N^N}{N!}P_0, \rho = 1 \end{cases} \tag{8}$$

By using the Little's law, the average response time of a request T can be calculated as follows:

$$T = \frac{L}{\widetilde{\lambda}} \tag{9}$$

The actual mean server utilization, $\widetilde{\rho}$, can be obtained by:

$$\widetilde{\rho} = \frac{\widetilde{\lambda}}{N\mu} \tag{10}$$

B. Queueing Network Model of Multi-tier Web Systems

A multi-tier web service system can be formulated as a queueing network, which is shown as Fig. 3. As shown in the figure, each server in every tier can be formulated as a M/M/N/K queueing system according to the section III-A. In tier i, there are multiple servers with the same functions and S_i is the number of servers in tier i. It should be noted that the soft resource allocation of different servers may not be the same even though in the same tier.

The total arrival rate of the server j in tier i, λ_{ij}, can be calculated from Eq. (11).

$$\lambda_{ij} = \omega_{ij}\lambda_i \tag{11}$$

where λ_i represents the total arrival rate of tier i and ω_{ij} is the balance factor to determine the ratio of requests that are allocated to the server j in tier i.

The drop probability of the server j in tier i is defined as P_{ij}^{drop}, which can be calculated from Eqns. (1)-(6).

$\widetilde{\lambda}_{ij}$ represents the effective arrival rate of requests, which is related to the drop probability of that server P_{ij}^{drop}.

$$\widetilde{\lambda}_{ij} = (1 - P_{ij}^{drop})\lambda_{ij} \tag{12}$$

Base on the Little's law, the average response time of requests processed in the server j of tier i can be calculated as follows:

$$T_{ij} = \frac{L_{ij}}{\widetilde{\lambda}_{ij}} \tag{13}$$

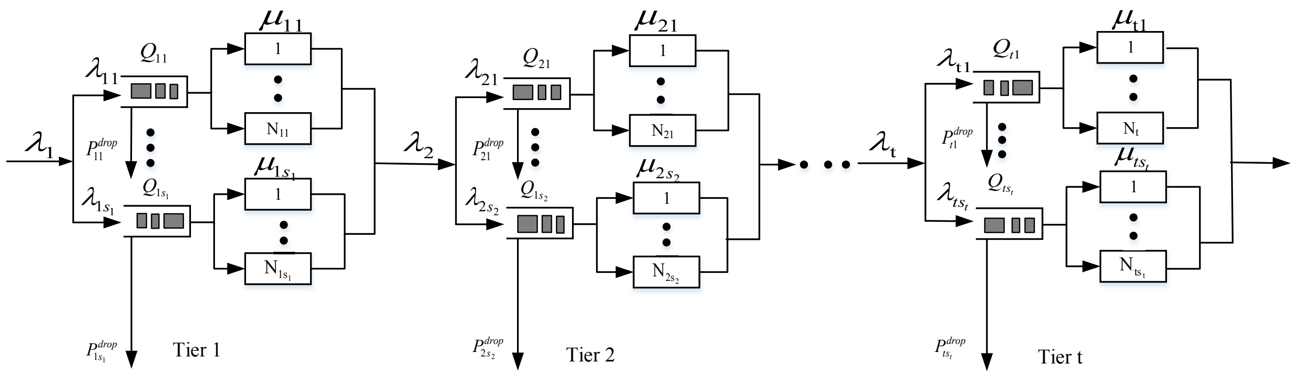

Figure 3. Queueing network model of a multi-tier web service system

where L_{ij} is the average queue length of the server j in tier i, which can be calculated according to Eq. (8)

The average response time of requests processed in tier i can be calculated by Eq. (14).

$$\overline{RT_i} = \sum_{j=1}^{S_i} \omega_{ij} T_{ij} \tag{14}$$

Finally, the average response time $\overline{RT}$ of the queueing system is the total of the average response time of all tiers, which can be calculated as follows:

$$\overline{RT} = \sum_{i=1}^{t} \overline{RT_i} \tag{15}$$

C. Optimization Approach

For addressing the dynamic workloads and meeting the performance demands, an optimization approach is designed for real-time soft resource allocation by applying sliding window. Due to the average arrival rate and service rate are used in the M/M/N/K queueing system for single server and queueing network model for multi-tier web service system, it is necessary to use sliding window techniques to obtain these two parameters. And then the performance indicators are recalculated according to the Section III-A and III-B.

Fig. 4 shows an example of sliding window technique to obtain the average arrival rate and service rate. As shown in the Figure, assuming that one grid implies a unit time, before 4th unit time, the workload of the system is WL_1, and after 4th unit time, the workload of the system is WL_2. The gray grids represent the sliding window and the number of gray grids represents the length of the sliding window, e.g., the length of sliding window in the figure is three unit time. Each time the multi-tier web service system runs for one unit time, the sliding window moves forward one unit time and recalculated the average arrival rate and service rate. Finally, the response time of the whole system and the drop probability of each server can be calculated and obtained.

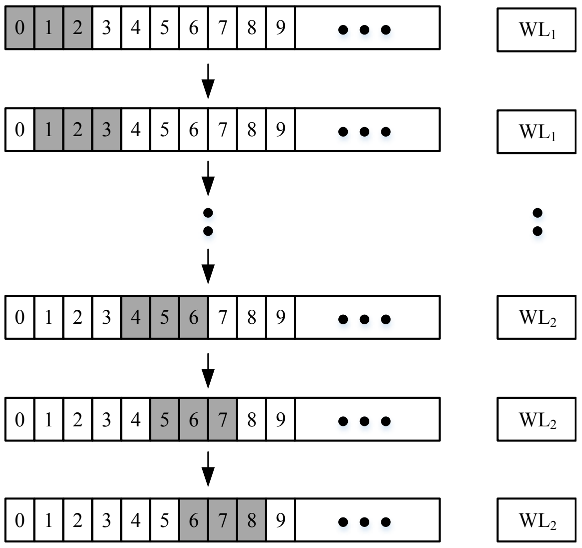

Figure 4. An example of sliding window approach

This paper aims at meeting performance demands in response to dynamic workloads by tuning soft resource allocation in real time. The performance demands include two aspects. On one hand, to enhance the user experience, the performance of the whole multi-tier web service system should be guaranteed within an acceptable bound, that is, the response time $\overline{RT}$ should less than a threshold Δt, which can be described as Eq. (16). On the other hand, the drop probability of each server should also be within a maximum range to ensure that more requests can be processed. This restriction can be expressed as Eq. (17).

$$\overline{RT} \le \Delta t \tag{16}$$

$$P_{ij}^{drop} \le \Delta P, \forall i \in \{1, ..., t\}, \forall j \in \{1, ..., S_i\} \tag{17}$$

According to the performance requirements described in Eq. (16) and Eq. (17), an optimization algorithm for real-time soft resource allocation is proposed and analyzed.

The pseudo-code of the optimization algorithm is shown as Algorithm 1.

Before introducing the procedure of algorithm, we firstly describe two simple heuristic optimization strategies.

1) Arithmetic Increment: Assuming that the current soft resource allocation of server j in tier i is N_{ij}, the strategy of arithmetic increment can be described as follows:

$$\begin{aligned} \beta_\alpha^+ &= N_{ij} \oplus \alpha = N_{ij} + \alpha * \Delta N \\ \beta_\alpha^- &= N_{ij} \ominus \alpha = N_{ij} - \alpha * \Delta N \end{aligned} \tag{18}$$

where β_α^+ and β_α^- represent the allocation are increased and decreased respectively, α represents the number of allocation changes and ΔN indicates the increment of allocation for each change.

2) Geometric Increment: Assuming that the current soft resource allocation of server j in tier i is N_{ij}, the strategy of geometric increment can be described as follows:

$$\begin{aligned} \beta_\alpha^+ &= N_{ij} \oplus \alpha = N_{ij} * (\Delta N)^\alpha \\ \beta_\alpha^- &= N_{ij} \ominus \alpha = N_{ij} * (\Delta N)^{-\alpha} \end{aligned} \tag{19}$$

where β_α^+ and β_α^- also represent the allocation are increased and decreased respectively, α represents the number of allocation changes and ΔN indicates the multiple of allocation for each change.

The response time of the whole system and the drop probability of each server in the sliding window area can be calculated and obtained from queueing models in the current workload and soft resource allocation at any time.

The algorithm mainly consists of two cases: short response time and long response time.

1) Short Response Time: In that case, the response time is shorter than the set maximum response time Δt.

On one hand, although the response time is guaranteed, some servers may also be set lower allocation that results in high drop probability. Therefore, soft resource allocation of these servers should be increased. α_{max} in the algorithm means the maximum number of changes, the value is related to the server capacity, the default value is infinity until the server utilization is full or achieves the utilization limits we set (e.g., 90% utilization), and then we will update this value. When α achieves maximum value and the drop probability is still higher than ΔP, we should use other methods, such as upgrading servers or distributing less requests to this server, to solve this problem. However, in our paper, we mainly consider the situation that the drop probability can meet requirements by changing soft resource allocation.

On the other hand, some servers under maximum drop probability may be allocated high soft resources. Hence, we should decrease soft resource allocation of these servers to reduce the resource consumption within the guaranteed performance bound. Specially, we firstly sort the servers in ascending order according to the drop probability, the process is to guarantee the server with higher allocation can be first adjusted. Then the soft resource allocation will be

Algorithm 1 Optimization algorithm for real-time soft resource allocation

```
1:  For any time k, the current workload is $WL_k$ and soft resource allocation is $SR_k = (N_{11}^k, ..., N_{tS_t}^k)$;
2:  Calculating response time $\overline{RT_k}$ and drop probability of each server $P_{ij}^{drop}, \forall i \in \{1,...,t\}, \forall j \in \{1,...,S_i\}$ in the sliding window;

3:  /*Short response time case*/
4:  if $\overline{RT_k} \le \Delta t$ then
5:      for each $P_{ij}^{drop} > \Delta P$ do
6:          $\alpha = 1$, $\beta_\alpha^+ = N_{ij} \oplus \alpha$;
7:          if $P_{ij}^{drop} > \Delta P$ && $\alpha \le \alpha_{max}$ then
8:              $\alpha = \alpha + 1$;
9:          else
10:             $N_{ij} \leftarrow \beta_\alpha^+$;
11:         end if
12:     end for
13:     /*Start from the server with higher allocation*/
14:     for each $P_{ij}^{drop} \le \Delta P$ in ascending order do
15:         $\alpha = 1$, $\beta_\alpha^- = N_{ij} \ominus \alpha$;
16:         if $\widetilde{\overline{RT_k}} \le \Delta t$ && $\widetilde{P_{ij}^{drop}} \le \Delta P$ then
17:             $\alpha = \alpha + 1$;
18:         else if $\widetilde{\overline{RT_k}} \le \Delta t$ && $\widetilde{P_{ij}^{drop}} > \Delta P$ then
19:             $N_{ij} \leftarrow \beta_{\alpha-1}^-$; continue;
20:         else
21:             $N_{ij} \leftarrow \beta_{\alpha-1}^-$; break;
22:         end if
23:     end for
24: end if

25: /*Long response time case*/
26: if $\overline{RT_k} > \Delta t$ then
27:     /*Start from the critical bottleneck server*/
28:     for each $P_{ij}^{drop} > \Delta P$ in descending order do
29:         $\alpha = 1$, $\beta_\alpha^+ = N_{ij} \oplus \alpha$;
30:         if $P_{ij}^{drop} > \Delta P$ && $\alpha \le \alpha_{max}$ then
31:             $\alpha = \alpha + 1$;
32:         else if $P_{ij}^{drop} \le \Delta P$ && $\widetilde{\overline{RT_k}} > \Delta t$ then
33:             $N_{ij} \leftarrow \beta_\alpha^+$; continue;
34:         else
35:             $N_{ij} \leftarrow \beta_\alpha^+$; break;
36:         end if
37:     end for
38:     for each $P_{ij}^{drop} \le \Delta P$ do
39:         $\alpha = 1$, $\beta_\alpha^- = N_{ij} \ominus \alpha$;
40:         if $P_{ij}^{drop} > \Delta P$ then
41:             $N_{ij} \leftarrow \beta_{\alpha-1}^-$;
42:         end if
43:     end for
44: end if
45: Return new soft resource allocation $\widetilde{SR_k}$.
```

adjusted one by one until the response time is longer than the threshold Δt or all the servers are adjusted.

2) Long Response Time: In that case, the response time is so long that cannot meet the performance requirements.

At that time, we should first adjust soft resource allocation to meet the performance demands described by Eq. (16) and Eq. (17). The servers are sorted in descending order and the server with maximum drop probability will be first handled, because this server is the critical bottleneck server that results in long response time. Then the soft resource allocation of these servers will be adjusted one by one until the performance demand are achieved.

Even in this case, there still may exist some servers with lower drop probability and high soft resource allocation, therefore, soft resource allocation of these servers should also be decreased to reduce the resource consumption.

Finally, the new soft resource allocation of each server in current workload can be obtained and the performance objective also can be guaranteed.

IV. Experimental Measurements

In this section, we present the experimental measurements for obtaining model parameters and validating the effectiveness of our approach. The environment is firstly described and then we introduce the method of obtaining parameters from measurement data, finally, the efficacy of optimization approach is validated by performance evaluations.

A. Environment

In the experimental measurements, the RUBiS benchmark system [22] is used as our testing environment. The RUBiS benchmark is based on the property of the eBay auction web service system and also can be used to study the performance of multi-tier web service system.

The RUBiS benchmark system can be implement as three-tier or four-tier architecture. In our experimental environment, the RUBiS system is implemented as four-tier architecture. Fig. 5 outlines the network topology of the four-tier RUBiS system in our experiments. The four-tier servers are Apache server, Tomcat server, C-JDBC server and MySQL server, respectively. All servers are on the Linux servers with the same hardware configurations. The tier with two servers plays a role of load balancing with the balance factor of 0.5, which means that the same amount of requests are processed by the servers in the same tier.

The RUBiS benchmark can simulate different request processes of auction system, such as register, browse, bid and so on. There are two workload types, which are browsing-only and read/write interaction mixes. Considering the actual characteristics of workload, we use read/write interaction mixes workload in our experiments. The system is session-based, each user starts a session and sends a series of requests to backend servers to serve. After receiving results from the servers, the next request depends on a Markov state transition table, which contains a probability matrix of accessing to different states to determine the next request.

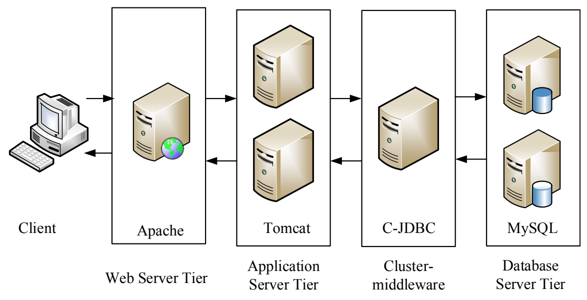

Figure 5. RUBiS system network topology

Soft resource of each tier we select is the software configuration that can control the maximum number of threads or processes, which will influence the maximum concurrent requests that the server can accept to process. Specially, the MaxClients is selected in Apache server due to the use of prefork MPM, the maxTheads, maxNbOfConnections and max_connections are selected in Tomcat server, C-JDBC and MySQL server, respectively.

B. Parameter Settings

Based on the queueing model described in Section III, in order to calculate the drop probability of each server and the response time of the whole multi-tier web service system, there are four model parameters we need to obtain, including soft resource allocation of each server N, queueing length of each server Q and average arrival rate of each server λ and average service rate of each thread or process μ.

Soft resource allocation and queueing length of each server can be directly obtained or calculated from configurations. Specially, soft resource allocation of server j in tier i is the value of N_{ij}. It should be noted that the maximum length for pending requests, i.e., the parameter Q described in Section III, can also be configured. The parameter Q of Apache server, Tomcat server, C-JDBC server and MySQL server can be configured by ListenBacklog, backlog, maxNbOfThreads and back_log, respectively. The parameter K_{ij} of server j in tier i is the sum of soft resource allocation N_{ij} and the maximum length for pending requests Q_{ij}, which can be described as follows.

$$K_{ij} = N_{ij} + Q_{ij}, i \in \{1, ..., t\}, j \in \{1, ..., S_i\} \qquad (20)$$

The average arrival rate of each server can be calculated from server logs. In the servers log, requests arrived in the servers and the time of requests that are processed can be recorded. So we can calculate the two parameters. Assuming that there are R_{ij} requests arriving in the sever j of tier i in the sliding window time SWT. Therefore, the average arrival rate λ_{ij} can be calculated as follows.

$$\lambda_{ij} = \frac{R_{ij}}{SWT} \qquad (21)$$

The average server utilization $\widetilde{\rho_{ij}}$ can be obtained from logs generated by SysStat, which is a performance monitoring tool in Linux and can record the real-time hardware resource utilization. The average CPU utilization is used to represent $\widetilde{\rho}$ of the server. Then the average service rate of each process can be calculated by Eq. (10).

C. Performance Evaluations

Based on the RUBiS system described in Section IV, the effectiveness of our model and approach for real-time soft resource allocation is validated by experimental evaluations.

The workload used in the experiments is shown as Fig. 6. There are 15 minutes in total. The workload changes every 5 minutes, and the three workloads are 3000, 1000, 2000, respectively. The sliding window length is 2 minutes, which is shown as gray grids. In the sliding window, the average workload from (a) to (e) in the figure is 3000, 2000, 1000, 1500, 2000, respectively. The experiments are relatively comprehensive, because the change of workload reflects the dynamic characteristics, i.e., increase, decrease or invariance.

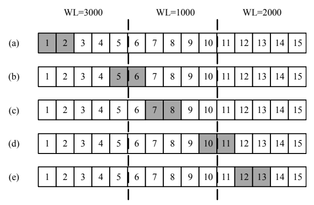

Figure 6. Workload

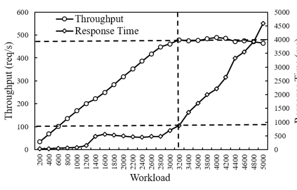

Figure 7. System performance

In the evaluations, soft resource allocation is set to the default configuration at the beginning, i.e., the soft resource allocation from Apache server to MySql server is 200, 150, 150, 150, respectively. It is noted that servers with the same tier use the same soft resource allocation in our experiments, because the servers with the same hardware configuration and the the balance factor is 0.5 in the same tier.

Fig. 7 represents the trend of throughput and response time as the workload increases at the default soft resource allocation. In the figure, the response time is relatively flat rather than steep when the workload is from 1400 to 2800 concurrent users, it is the buffer effect of soft resource in front tiers, which makes the whole system achieve better performance [3]. The throughput tends to be stable and the response time grows sharply when a certain workload (i.e., 3200 concurrent users) is reached. Due to the response time is about 1s at the bottleneck workload, so we set the threshold of response time is 1s. As to the drop probability of every server, the threshold is set to 10% in our experiments and it can be adjusted according to the actual demands.

In our experiments, the two heuristic optimization strategies are used to validate our method, meanwhile, the ΔN in the arithmetic increment and geometric increment is 50 and 2, respectively. Fig. 8 and 10 are the real-time soft resource allocation with the change of workload by using the two strategies. Fig. 9 and 11 are the comparison of response time by using the two strategies. The plot of current configuration means that the response time by using the previous soft resource allocation without changing by the two strategies. The time t in the Fig. 6 is expressed as the time interval $[t-1, t]$ in that four figures.

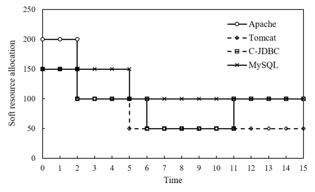

Figure 8. Arithmetic Increment

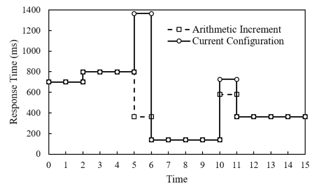

Figure 9. Response Time

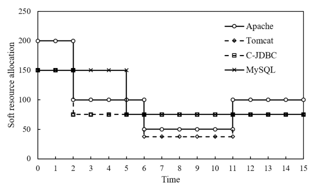

Figure 10. Geometric Increment

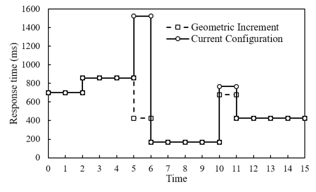

Figure 11. Response Time

As shown in the Fig. 8 and 10, before time 2, the soft resource allocation is not changed any more due to the sliding window is 2 minutes. After time 2, as the sliding window moves, the model parameters change with the change of workload. And then, the two strategies update the soft resource allocation with the guaranteed performance demands. Through comparing the soft resource allocation of the two figures, it can be found that soft resource allocation obtained from the two strategies are approximative. From Fig. 9 and 11, the response time is guaranteed through allocating soft resource in real time, e.g., at time 5, the response time is more than 1s in the current configuration, while the response time is reduced to within 1s after allocating new soft resource allocation whether the strategy is arithmetic increment or geometric increment. After using the new soft resource allocation, the response time of the two curves are the same as the workload is stable due to the same soft resource allocation, e.g., time interval $[2, 5]$. That is, experimental results show that the optimization approach can update soft resource allocation in real time according to the dynamic workloads and the approach is effective.

V. Conclusion

In this paper, we have proposed a real-time soft resource allocation approach for addressing dynamic workloads in the multi-tier web service systems. Each server of different tiers is firstly theoretically formulated by M/M/N/K queueing system considering the limited capacity of servers, and the whole system is formulated by queueing network model. Then, by applying sliding window techniques, an optimization approach for real-time soft allocation is presented to cope with dynamic workloads and meet performance demands. Model parameters and performance indicators are analyzed in detail. Based on the four-tier RUBiS benchmark system, the effectiveness of the optimization approach is validated by experimental measurements.

Acknowledgment

This work is partially supported by NSFC (No.61572295, 61502043),the Innovation Method Fund of China (No.2015IM010200), SDNSFC (No.ZR2014FM031), the TaiShan Industrial Experts Programme of Shandong Province, the Shandong Province Science and Technology Major Special Project (No.2015ZDXX0201B03, 2015ZDXX0201A04, 2015ZDJQ01002), the Shandong Province Key Research and Development Plan (No.2015GGX101015, 2015GGX101007), Beijing Natural Science Foundation (No.4162042), BeiJing Talents Fund (No.2015000020124G082). Corresponding authors are Lei Liu and Yuliang Shi.

References

[1] "ebay web site," http://www.ebay.com.

[2] S. Nanda, T. J. Hacker, and Y. Lu, "Predictive model for dynamically provisioning resources in multi-tier web applications," in *2016 IEEE International Conference on Cloud Computing Technology and Science*, 2016, pp. 326–335.

[3] Q. Wang, S. Malkowski, D. Jayasinghe, P. Xiong, C. Pu, Y. Kanemasa, M. Kawaba, and L. Harada, "The impact of soft resource allocation on n-tier application scalability," in *25th IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, 2011, pp. 1034–1045.

[4] R. Peña-Ortiz, J. A. Gil, J. Sahuquillo, and A. Pont, "Analyzing web server performance under dynamic user workloads," *Computer Communications*, vol. 36, no. 4, pp. 386–395, 2013.

[5] Y. Shi, J. Huang, X. Zhao, L. Liu, S. Liu, and L. Cui, "Integrating theoretical modeling and experimental measurement for soft resource allocation in multi-tier web systems," in *Conference on Web Services, ICWS 2016*, 2016, pp. 522–529.

[6] K. D. Babu, D. G. Kumar, and S. Veluru, "Optimal allocation of virtual resources using genetic algorithm in cloud environments," in *12th ACM International Conference on Computing Frontiers*, 2015, pp. 55:1–55:6.

[7] T. Subramanian and N. Savarimuthu, "Application based brokering algorithm for optimal resource provisioning in multiple heterogeneous clouds," *Vietnam J. Computer Science*, vol. 3, no. 1, pp. 57–70, 2016.

[8] S. N. Srirama and A. Ostovar, "Optimal resource provisioning for scaling enterprise applications on the cloud," in *IEEE 6th International Conference on Cloud Computing Technology and Science*, 2014, pp. 262–271.

[9] C. A. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervello-Pastor, and Á. Monje, "On the optimal allocation of virtual resources in cloud computing networks," *IEEE Trans. Computers*, vol. 62, no. 6, pp. 1060–1071, 2013.

[10] J. Jiang, J. Lu, G. Zhang, and G. Long, "Optimal cloud resource auto-scaling for web applications," in *13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, 2013, pp. 58–65.

[11] Y. Diao, J. L. Hellerstein, S. S. Parekh, H. Shaikh, and M. Surendra, "Controlling quality of service in multi-tier web applications," in *26th IEEE International Conference on Distributed Computing Systems*, 2006, p. 25.

[12] B. Urgaonkar, P. J. Shenoy, A. Chandra, and P. Goyal, "Dynamic provisioning of multi-tier internet applications," in *Second International Conference on Autonomic Computing (ICAC)*, 2005, pp. 217–228.

[13] C. Liu, K. Huang, Y. Lee, and K. Lai, "Efficient resource allocation mechanism for federated clouds," *IJGHPC*, vol. 7, no. 4, pp. 74–87, 2015.

[14] B. Urgaonkar, G. Pacifici, P. J. Shenoy, M. Spreitzer, and A. N. Tantawi, "An analytical model for multi-tier internet services and its applications," in *Proceedings of the International Conference on Measurements and Modeling of Computer Systems (SIGMETRICS)*, 2005, pp. 291–302.

[15] Y. Xia, M. Zhou, X. Luo, S. Pang, and Q. Zhu, "Stochastic modeling and performance analysis of migration-enabled and error-prone clouds," *IEEE Trans. Industrial Informatics*, vol. 11, no. 2, pp. 495–504, 2015.

[16] Y. Xia, M. Zhou, X. Luo, Q. Zhu, J. Li, and Y. Huang, "Stochastic modeling and quality evaluation of infrastructure-as-a-service clouds," *IEEE Trans. Automation Science and Engineering*, vol. 12, no. 1, pp. 162–170, 2015.

[17] P. Xiong, Z. Wang, S. Malkowski, Q. Wang, D. Jayasinghe, and C. Pu, "Economical and robust provisioning of n-tier cloud workloads: A multi-level control approach," in *2011 International Conference on Distributed Computing Systems*, 2011, pp. 571–580.

[18] Y. Ma, S. Wang, P. C. K. Hung, C. Hsu, Q. Sun, and F. Yang, "A highly accurate prediction algorithm for unknown web service qos values," *IEEE Trans. Services Computing*, vol. 9, no. 4, pp. 511–523, 2016.

[19] N. N. H. Win, J. Bao, G. Cui, and S. Rehman, "Self-adaptive qos-aware web service discovery using ontology approach," *IJGHPC*, vol. 7, no. 3, pp. 65–84, 2015.

[20] S. Wang, C. Hsu, Z. Liang, Q. Sun, and F. Yang, "Multi-user web service selection based on multi-qos prediction," *Information Systems Frontiers*, vol. 16, no. 1, pp. 143–152, 2014.

[21] E. Chlebus and J. Brazier, "Nonstationary poisson modeling of web browsing session arrivals," *Inf. Process. Lett.*, vol. 102, no. 5, pp. 187–190, 2007.

[22] "Rubis: Rice university bidding system," http://rubis.ow2.org.