Integrating Theoretical Modeling and Experimental Measurement for Soft Resource Allocation in Multi-Tier Web Systems

Yuliang Shi[1], Jiwei Huang[2], Xudong Zhao[1], Lei Liu[1], Shijun Liu[1], Lizhen Cui[1*]
[1]School of Computer Science and Technology, Shandong University, Jinan 250101, China
[2]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
Email: shiyuliang@sdu.edu.cn, huangjw@bupt.edu.cn, sdu_zxd@163.com, {l.liu, lsj, clz}@sdu.edu.cn

***Abstract*—Soft resources, which are system software components that use hardware or synchronize the use of hardware, are playing a critical role in the performance of multi-tier web systems, and thus it is quite important to tune the soft resource allocation for using the limited hardware resources to obtain maximum effectiveness. In this paper, we integrate both theoretical and experimental studies to the soft resource allocation problem. Specifically, we apply the queueing network model for formulating multi-tier web systems, and conduct experimental measurements based on the RUBiS benchmark system to obtain precise model parameters. Quantitative analysis is carried out, based on which an optimization model as well as an algorithm are put forward for soft resource allocation. The efficacy of our approach is validated by both theoretical analyses and experimental results.**

Keywords- soft resource allocation; performance; queueing network; multi-tier web system

I. INTRODUCTION

With the rapid development of internet technologies, various web systems have emerged, and web-based services have been widely applied in several aspects of human daily life. Since the number of web systems is growing exponentially with time, the performance of the web services and systems meets challenges. On one hand, the performance can be improved by hardware upgrading, bringing more system capacity and thus more revenue. On the other hand, however, it is quite costly to upgrade hardware, and hence an alternative approach is to make full use of hardware resources by efficient resource allocation. Recently, it has become a hot topic in both academia and industry.

Soft resources are system software components that use hardware or synchronize the use of hardware [1]. It is quite important for the performance especially in multi-tier web systems to allocate proper amount of resources to different layers in order to maximize the utilization of hardware resources. A representative example of soft resource is the number of threads in each layer of a multi-tier web system, which has been experimentally proved to be a critical factor affecting the parallelism of web services [2]. Given a fixed hardware configuration, soft resource allocation is one of the most important issues of performance optimization, and thus has become an urgently required research problem.

In this paper, we make an attempt at optimizing soft resource allocation (i.e. the number of threads in each server) by integrating both theoretical and experimental studies in multi-tier web systems. A web system is theoretically formulated by queueing network models, and mathematical analyses of performance indicators and resources utilization are given. Massive experiments based on the RUIBiS system are conducted to obtain model parameters, and thus the performance in different soft resource allocations can be precisely calculated. Based on such quantitative analyses, an optimization algorithm is put forward for soft resource allocation, whose efficacy is further validated by real-life experimental results.

The primary contribution of this paper is three-fold as follows.

- We theoretically formulate a multi-tier web system by queueing models. According to the characteristics of the system, servers in each tier can be modeled by an M/M/n queueing, and the whole system forms a Jackson queueing network. Detailed methodology of performance analysis is given.
- We experimentally study the relationship between model parameters and measurement results. Using a benchmark based on the RUBiS system, we present detailed procedures of obtaining model parameters from experimental data.
- We propose an optimization algorithm for soft resource allocation in multi-tier web systems. Besides model-based analysis, we also validate the effectiveness of our approach by experimental results.

The remainder of the paper is organized as follows. Section II introduces the background and related work. Queueing models of single server and multi-server are described respectively in Section III. Section IV introduces the experimental measurements for obtaining model parameters. Section V proposes an optimization algorithm of soft resource allocation and Section VI gives the experimental results of the efficacy of our approach. Finally, Section VII concludes the paper.

*Lizhen Cui is the corresponding author. Email: clz@sdu.edu.cn.

978-1-5090-2675-3/16 $31.00 © 2016 IEEE
DOI 10.1109/ICWS.2016.73

II. BACKGROUND AND RELATED WORK

A. Background

In order to efficiently use the hardware resources and achieve better system performance, many web systems are designed to employ multi-tier architecture. Each tier plays a certain role and cooperates with other tiers to ensure the system can efficiently and correctly complete the tasks or operations. According to the system workload demand and the difference of server computing capacity, through using cluster technologies, a tier may have multiple servers with the same functionalities to achieve load balancing. In general, three-tier architecture is most commonly used in multi-tier web systems, and the three tiers are web server tier, application server tier and database server tier, respectively.

In the first web server tier, the web servers mainly handle HTTP protocol. There are four main functions: (1) accepting the requests from clients; (2) processing static requests, such as static web pages; (3) distributing dynamic requests to application server tier; (4) receiving the request results from the application server tier and returning them to clients. Apache is one of the representative web servers.

In the second application server tier, the application servers mainly serve business logic through various protocols. There are three main functions: (1) accepting the dynamic requests from the web server tier; (2) handling the concrete business logic of the requests, such as deleting users; (3) receiving the processing results from database server tier and returning them to the web server tier. One typical example of the application servers is Tomcat server.

In the third database server tier, the database servers mainly manage and handle the data of the system. There are two main functions: (1) executing queries of the requests, such as adding users to the database; (2) storing various data of the system, for example, login account and password, user information and so on. MySQL and Oracle are the two common database servers.

Soft resources are software components, which can be configured by the software parameters and then have a great impact on the hardware utilization and overall system performance. According to the functions of the three tiers, it is clearly that the whole request processing in the three-tier web systems. When the clients send requests to the backend servers, the web server tier firstly accepts the requests and distributes them to the application server tier, then executes queries on the database server tier, finally, returns the results to the clients. In this process, soft resources are critical factors that make the interaction between each tier.

In this paper, we only consider one of the key and typical soft resources, which is the number of threads in each server. The reason is that every server needs to create concurrent threads to accept and handle various requests in the whole request process. Too few threads can limit large amount of concurrency, degrade the server processing capacity and then result in the insufficient usage of hardware resources, that is, it causes the soft resource bottleneck [1]. Adding threads can relieve this bottleneck. However, too many threads grab resources as running and consume resources even when idle, which may also cause performance degradation. Hence, it is imperative to tune soft resource allocation for efficient hardware utilization and better system performance.

B. Related Work

Multi-tier architecture, especially three-tier architecture, has been widely applied in the web systems for improving the flexibility and scalability. At the same time, because of the dependency between each tier, this architecture also makes it difficult to improve the overall performance.

Many previous works have focused on the study of single tier server, for example, web server tier [3], [4], to improve the system performance through various ways. Jin et al. [3] proposed an autonomous network control method to dynamically change the network configuration for enhancing the web server performance and then improving the whole system performance. However, because of the interaction between each tier, the most critical and complex issue is to optimize the resource allocation among multiple tiers. Effectively tuning the resources of multiple tiers can improve the system performance better than simply optimizing single server.

With the popularity of virtualization technology and cloud computing, more and more service providers tend to deploy the multi-tier web systems in the cloud environments. There are two main ways to meet the system demand. The first is to dynamically change the amount of the virtual machines [5], [6]. In [5], the authors proposed an auto-scaling scheme to predict the optimal number of virtual machines, and then the system can be dynamically scaled based on the optimal VMs demand. The second is to change the size of virtual machines based on the required resources, especially CPU [7]-[10]. Zhu et al. [7] proposed a dynamic resource provisioning algorithm to adjust the resources of a virtual instance. Bi et al. [11] presented a fundamental framework for dynamic fine-grained resource allocation in a virtualized cloud data center and a dynamic hybrid metaheuristic algorithm is proposed for optimization. However, the performance improvements by both the two ways are at the hardware level. In this paper, we improve the performance through tuning soft resource allocation, which is at the software level.

Previous works [1], [13] have studied the impact of soft resource allocation. Wang et al. [1] defined the soft resources and analyzed the similarities and differences between soft resources and hardware resources. Both under-allocation and over-allocation of soft resources may result in performance degradation. In [12], the authors analyzed the impact of soft resource allocation in consolidated n-tier applications, and found that limiting the number of runnable active threads can mitigate the performance interference among consolidated applications.

Many previous works have studied the performance of web services and systems based on the analytical approaches. A stochastic-queueing-network-based approach is proposed for performance analysis of cloud systems and a confidence-interval analysis is conducted [14]. In [15], the authors modeled the multi-tier Internet applications using the closed queueing model. Xiong et al. [10] applied the M/G/1/PS queue and designed an adaptive controller to control the

mean round trip time for improving the system performance. Xia et al. [16] presented quality evaluation of Infrastructure-as-a-service clouds by a stochastic model-based approach.

In this paper, we theoretically model multi-tier web systems using queueing models. Based on the relationship analysis between the model parameters and measurement results, we also propose an optimization algorithm for soft resource allocation in multi-tier web systems.

III. THEORETICAL MODEL

In this section, single server model and multi-server model are formulated to describe the request processing in one server and the whole multi-tier web systems respectively by queueing theory. And then we give the analysis of the two models.

A. Single Server Model

In web systems, it has been shown that web browsing arrivals above the session level conform to Poisson distribution [17]. Due to the property of independent Poisson processes, the request arrival of one server also follows Poisson distribution. There are many threads can be created and used to handle requests in one server. Assuming that the threads are inherently homogenous with the same average service rates, it is reasonable to use M/M/n queueing model to formulate a single server in each tier of the multi-tier web systems.

The M/M/n queueing model of a single server can be formulated as Fig.1, where λ is the request arrival rate of the server, T is the arrival time of one request and μ is the average service rate of each thread.

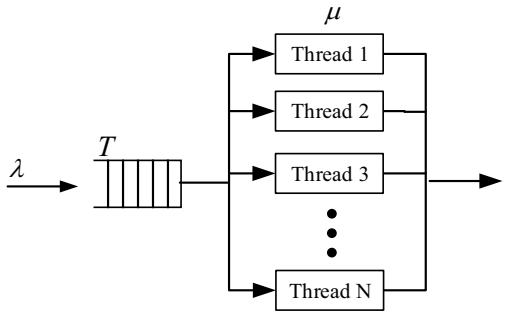

Fig. 1 M/M/n queueing model of single server.

Considering the dynamic activities of the queueing system, i.e., task arrivals and service procedures, the M/M/n queueing model can be formulated by a continuous-time Markov chain (CTMC). A state of the CTMC are defined by the number of tasks in the system at certain time period, and the transitions among them are illuminated by Fig. 2, where the variables indicating the transition rates.

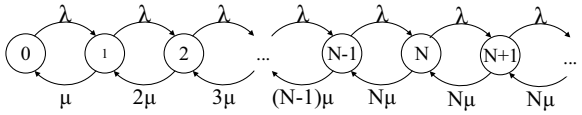

Fig. 2 CTMC model for the M/M/n queueing system.

As for the steady state of the server, the probability in each state expressed as $\eta = [\eta_0, \eta_1, ..., \eta_n]$ can be calculated by the following equations.

$$\eta_{\mathrm{k}} = \eta_0 \prod_{i=0}^{k-1} \frac{\lambda}{(i+1)\mu} = \eta_0 \left(\frac{\lambda}{\mu}\right)^k \frac{1}{k!}, k < N \quad (1)$$

$$\eta_{\mathrm{k}} = \eta_0 \prod_{i=0}^{N-1} \frac{\lambda}{(i+1)\mu} \prod_{j=N}^{k-1} \frac{\lambda}{N\mu} = \eta_0 \left(\frac{\lambda}{\mu}\right)^k \frac{1}{N! N^{k-N}}, k \geq N \quad (2)$$

$$\sum_{k=0}^{\infty} \eta_k = 1 \quad (3)$$

ρ is the average utilization of the server and can be defined as (4), it also can be used to calculate the thread's average service rate, N represents the number of threads in the server.

$$\rho = \frac{\lambda}{\mathrm{N}\mu} \quad (4)$$

The average service time T_s in the server is shown as (5).

$$\mathrm{T_s} = \frac{1}{\mu} \quad (5)$$

The average number of requests q in the server can be expressed by (6).

$$q = \sum_{k \geq 0} k\eta_k = N\rho + \rho \frac{(N\rho)^N}{N!} \frac{\eta_0}{(1-\rho)^2} \quad (6)$$

The average response time of a request T_q, i.e., the total amount of time that the request stays at the system including its waiting time and service time, can be calculated using the Little's law.

$$T_q = \frac{q}{\lambda} \quad (7)$$

B. Multi-Server Model

We consider a multi-tier web system with t tiers denoted by *Tier* 1, $\cdots$, *Tier* t, there might be multiple servers with the same functionalities in each tier. According to the single server model presented in the previous subsection, the web system with multiple tiers and servers can be formulated by a Jackson queueing network. Fig. 3 shows the queueing network model of multi-server in the web systems.

Tier a accepts the requests from the front *Tier* $a-1$, especially, *Tier* 1 is the first tier of the model and accepts the requests from the clients. The total requests arrival rate and the number of servers in *Tier a* is defined as λ_a and S_a, respectively. We assume that optimal load balance in every tier, which means that the request arrival rate of each server is the same. Hence, the request arrival rate of server b in *Tier a* can be calculated by (8).

$$\lambda_{ab} = \frac{\lambda_a}{S_a} \quad a = 1,...,t; \ \ b = 1,...,S_a \quad (8)$$

The state space of the whole network can be defined as $S_w = \{(m_1, m_2, ..., m_t) \mid m_a \geq 0\}$, where m_a is the number of

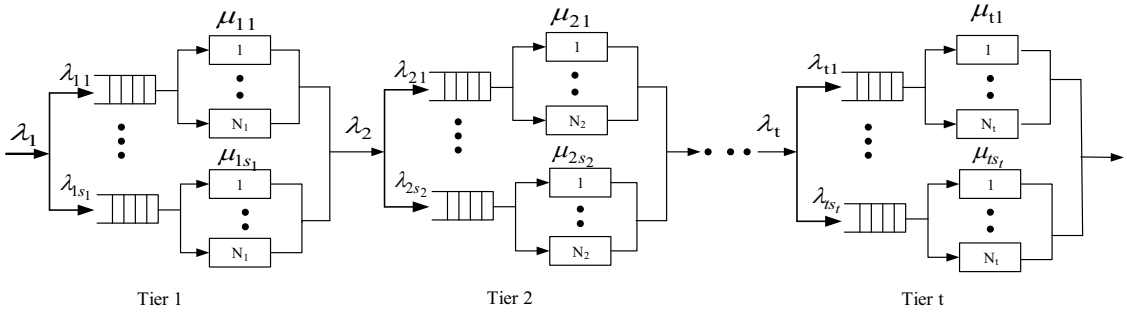

Fig. 3 Queueing network model of multi-server

steady servers in *Tier a*. In *Tier a*, the state space can be defined as $S_a = \{(m_{a1}, m_{a2}, ..., m_{as_a}) \mid m_{ab} \geq 0\}$, where m_{ab} is the queue length of server b in *Tier a*.

Assuming that the random variables of queue length vector and steady servers number vector in *Tier a* are $(M_{a1}, M_{a2}, ..., M_{aS_a})$ and $(M_1, M_2, ..., M_t)$, respectively. The steady state probability of the whole network expressed as $\eta(1,2,...,t) = [\eta_1, \eta_2, ..., \eta_t]$ can be obtained by (9) and (10). Noted that the steady state probability of each tier expressed as $\eta_a = [\eta_{a1}, \eta_{a2}, ..., \eta_{as_a}]$ is calculated by (9), where η_{ab} is the steady state probability of server b in *Tier a* and can be obtained according to (1), (2) and (3).

$$\eta_a = P(M_{a1} = m_{a1}, ..., M_{aS_a} = m_{as_a}) = \prod_{b=1}^{S_a} \eta_{ab} \quad (9)$$

$$\eta(1,...,t) = P(M_1 = m_1, ..., M_t = m_t) = \prod_{a=1}^{t} \eta_a \quad (10)$$

The total average number of requests in the network Q can be calculated by the following expression (11), where q_a equals to the average total average number of requests in any server of *Tier a* because of the homogenous of servers in each tier.

$$Q = \sum_{a=1}^{t} q_a \quad (11)$$

Therefore, the average response time of the whole request processing can be calculated by (12) with the Little's law.

$$\overline{\mathrm{RT}} = \frac{Q}{\lambda_1} \quad (12)$$

IV. Experimental Measurement

In this section, we present an experimental approach for obtaining the parameters used in the queueing models presented above. Detailed experimental setup is presented, and the measurements and calculations are introduced.

A. Experiment Setup

The RUBiS system [18] is an auction site prototype, which is usually used to study and evaluate the system performance as a common framework. We use this system as the testing cluster in our experimental study. The system supports 26 different types of request pages, such as Browse, PutBid and Sell. Requests are randomly generated using a Markovian state transition probability matrix, and metrics such as response time, utilization, amount of packets, etc. are captured. Each experiment is divided into three phases, which are a 1-minute up ramp, a 10-minute runtime and a 1-minute down ramp. In order to conduct precise steady-state analysis in our experiments, we mainly analyze the measurements of runtime phase.

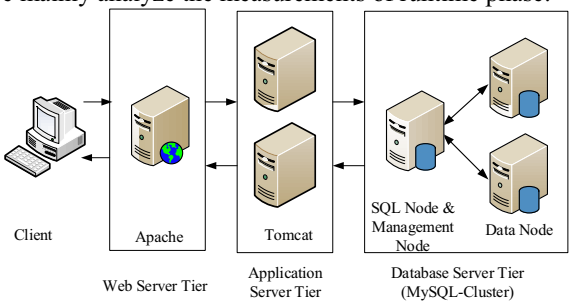

Fig. 4 RUBiS system network topology.

Fig. 4 outlines the basic network topology of the RUBiS benchmark system. The RUBiS system is implemented as three-tier architecture, the first web server tier uses one Apache 2.4.3 server, and then two Tomcat 8.0.9 servers are used in the application server tier. Through mod_jk module, Apache server and two Tomcat servers can be integrated to obtain the effect of load balance, in our experiments, it is optimal load balance. Finally, the database server tier is based on the MySQL-Cluster 7.3.6, the SQL node and the management node are on the same server, and the cluster has two data nodes.

All servers are on the physical nodes and the operating system is Centos 7.0, every machine has the same hardware configuration. The processor is Intel E7500 3GHz 64-bit and memory size is 4GB, using one gigabit Ethernet to connect two servers in LAN network, in addition, the disk size of each machine is 50GB.

We select one of the most important software parameters from each tier as the soft resource we study. All the three parameters can configure the maximum number of concurrent requests that the server can handle, which are the MaxClients in Apache server, the maxThreads in Tomcat server and the max_connections in SQL node of MySQL-Cluster. Note that we use prefork

MPM as the multi-processing module in Apache server, so only the MaxClients can control the maximum number of concurrent requests that the server can process.

B. Model Parameters

Based on the RUBiS system topology in Fig. 4 and the multi-server model in Fig. 3, we can theoretically model the RUBiS system using queueing network.

In the model, there are three tiers, i.e. *Tier* 1, 2 and 3, which represent web server tier, application server tier and database server tier, respectively. We put the MySQL-Cluster as a whole to consider, so the MySQL-cluster is formulated as one tier in queueing model.

We modify the source codes of the client to obtain the time spots of each request and its corresponding response. Comparing to the response time of web requests, we find that the latency of data transmission among the servers is relatively small. Therefore, assuming that T_i is the time spot of request i being submitted, we can obtain the average arrival rate of the web requests at Apache server tier with the following expression.

$$\lambda_1 = \frac{n-1}{\sum_{i=1}^{n-1}(T_{i+1} - T_i)} \tag{13}$$

Through using SysStat package in Linux [19], many useful performance measurements of each machine, such as CPU and memory utilization, can be monitored at one second granularity during the period of running system. The most useful measurements in our experiments are the average CPU utilization and total number of packets received/transmitted per second of each server.

The request processing in multi-tier web systems mainly contains two scenarios expressed in [15]. The simplest case is that each request is processed once at each tier and then distributed to next tier for further processing. Another complex but general case is that each request at one tier can trigger multiple requests to next tier, and finally all results are merged and sent back to clients. In the real web systems, both the two cases may exit in the request processing.

Considering that the request can trigger multiple requests to next tier and some static requests are handled in Apache server, the request arrival rates at *Tier* 2 and *Tier* 3 are measured from network layer. In detail, we use the proportion of the average total received packets rate in each tier, combined with the request arrival rate in *Tier* 1, to calculate the arrival rate of *Tier* 2 and *Tier* 3. The request arrival rate of two Tomcat servers in *Tier* 2 are the same because of the optimal load balance.

The packet transmission process in our system is shown as the Fig. 5.

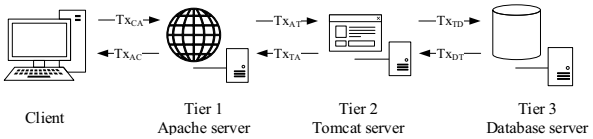

Fig. 5 Packet transmission process.

We use the Rx_a and Tx_a to express the total average received/transmitted packets rate in *Tier a* ($a = 1,2,3$), especially, the client uses Rx_c and Tx_c, this value can be obtained based on the monitoring logs. The client can only interact with the Apache, so the Tx_{CA}, Tx_{AT} and Tx_{TD} can be easily calculated according to the following equation set.

$$\begin{cases} Tx_{AC} = Rx_c \\ Tx_{CA} = Tx_c \\ Tx_{CA} + Tx_{TA} = Rx_1 \\ Tx_{AC} + Tx_{AT} = Tx_1 \\ Tx_{TA} + Tx_{TD} = Tx_2 \end{cases} \tag{14}$$

Then the request arrival rate of *Tier* 2 and *Tier* 3 can be calculated by (15) and (16)

$$\lambda_1 : \lambda_2 : \lambda_3 = Tx_{CA} : Tx_{AT} : Tx_{TD} \tag{15}$$

$$\lambda_{21} = \lambda_{22} = \frac{\lambda_2}{2} \tag{16}$$

Soft resources we select can limit the maximum of the concurrent threads in each server, so the threads created by the server may not achieve this maximum number when the workload is low, it is necessary to count the number of threads during the runtime. As to the Apache server, because of the character of prefork module, we use the way of monitoring the running processes in a duration. As to the Tomcat server, we count the number of the running threads in real time through viewing the server status page. As to MySQL-Cluster server, we can use its own ways to view the active threads list online and count the total threads time in real time.

After counting the number of threads in real time, in order to reduce the error, we set the average value as the parameter N in each tier. Especially, the size of N_2 can be set to the average value of two Tomcat servers.

Finally, the average service rate of one thread in each server of one tier $\mu_a (a = 1,2,3)$ can be calculated by the formula (4), where ρ is the CPU utilization of the server.

Because there exists system overhead as running in the real systems, such as threads switching, we assume that the average system overhead is $\Delta\mu$ as running threads, so the relationship between μ_c and μ_a can be expressed as the following expressions, where μ_c is the constant average thread's service rate we assume in the model, and μ_a is the real average thread's service rate. The μ_c and $\Delta\mu$ are the two key values to optimize the soft resource allocation in the following sections.

$$\mu_a = \mu_c - N\Delta\mu \tag{17}$$

V. Optimization

Soft resource allocation is a critical factor for the overall performance of a multi-tier web system. With all the theoretical analyses and experimental results, we propose

an optimization model and design an algorithm in this section for obtaining optimal configuration of soft resources.

A. Optimization Model

Our objective is to efficiently use hardware resources for providing performance-guaranteed web services or applications by tuning soft resource allocation. More specifically, we formulate our optimization problem by Eqns. (18)-(21). Given a specific workload, we want to make the hardware resources utilization as low as possible, while the performance should be guaranteed in an acceptable bound. Meanwhile, the queueing models should be stable, i.e., the utilizations should be less than 1.

$$\text{minimize} \quad \rho_{tot} = \sum_{a=1}^{t} \sum_{b=1}^{s_a} \rho_{ab} \tag{18}$$

Subject to

$$\overline{RT} \le t_0 \tag{19}$$

$$N_a > 0, \qquad \forall a \in \{1,...,t\} \tag{20}$$

$$0 < \rho_{ab} = \frac{\lambda_{ab}}{N_a \mu_{ab}} < 1, \ \forall a \in \{1,...,t\}, \forall b \in \{1,...,s_a\}. \tag{21}$$

In this paper, we use the CPU utilization to represent the overall hardware resource utilization, which has been widely acknowledged in several related literatures [10], [20]. The optimization objective is to minimize the average CPU utilizations of all servers in system, which is denoted by (18). In our optimization model, there are mainly three constraints. Firstly, the response time should be bounded denoted by (19), where $\overline{RT}$ is the average response time and t_0 is the maximum response time we set to guarantee the system performance. Secondly, the number of threads should be positive numbers, which is expressed by (20). Finally, the queueing network model is able to achieve the steady state, shown as (21), where ρ_{ab} represents the average CPU utilization of any servers in system.

B. Optimization Algorithm

According to the optimization model, an optimization algorithm is proposed and analyzed in this subsection.

We apply the simulated annealing [11], which is a probabilistic algorithm for approximating global optimum of a given evaluation function in a large search space, to optimize the soft resource allocation in multi-tier web systems. The pseudocode of the optimization procedure is shown as Algorithm 1.

The algorithm mainly consists of the following two steps:

Initialization: This step is to initialize the parameters of the algorithm, which should be subject to some limited conditions such as (20) and (21) to reduce the solutionspace. Selecting a random positive integer vector as the initial soft resource allocation and initializing the workload to WL. Then the total average CPU utilization of the whole network ρ_{tot} and the average response time $\overline{RT}$ can be calculated and initialized to ρ_{tot0} and $\overline{RT_0}$, respectively.

Algorithm 1 Optimization algorithm for soft resource allocation.

Step 1: Initialization

1: Select a positive integer vector $SR = SR_0 = (N_1,...,N_t)$;

2: $workload = WL$;

3: Let $\rho_{tot} = \rho_{tot0} = \sum_{a=1}^{t} \sum_{b=1}^{s_a} \rho_{ab} = \sum_{a=1}^{t} \sum_{b=1}^{s_a} \frac{\lambda_{ab}}{N_a \mu_{ab}}$, $\overline{RT} = \overline{RT_0}$;

Step 2: Iterative optimization

4: **for** $k = 1$ **to** $k_{\max}$ **do**

5: $T \leftarrow temperature(k / k_{\max})$

6: Pick a random neighbor, $SR_k \leftarrow neighbour(SR)$;

7: **if** $\overline{RT_k} \le t_0$ **then**

8: **if** $\Delta\rho = \rho_{totk} - \rho_{tot} < 0$ **then**

9: $\rho_{tot} \leftarrow \rho_{totk}$; $\overline{RT} \leftarrow \overline{RT_k}$; $SR \leftarrow SR_k$;

10: **else**

11: $P(\rho_{tot}, \rho_{totk}, T) = e^{\frac{-\Delta\rho}{T}} = e^{\frac{-1*(\rho_{totk} - \rho_{tot})}{T}}$

12: **if** $P(\rho_{tot}, \rho_{totk}, T) \ge random(0,1)$ **then**

13: $\rho_{tot} \leftarrow \rho_{totk}$; $\overline{RT} \leftarrow \overline{RT_k}$; $SR \leftarrow SR_k$;

14: **end if**

15: **end if**

16: **else**

17: **if** $\overline{RT_k} \le \overline{RT}$ **then**

18: $\rho_{tot} \leftarrow \rho_{totk}$; $\overline{RT} \leftarrow \overline{RT_k}$; $SR \leftarrow SR_k$;

19: **end if**

20: **end if**

21: **end for**

22: **return** $SR, \rho_{tot}, \overline{RT}$.

Iterative Optimization: This step is to find the optimal soft resource allocation through using iterative approaches.

In the process of each iteration, the new candidate soft resource allocation is the neighbor of the current allocation and randomly generated, then mainly using simulated annealing criterion to decide whether to accept this new allocation. In detail, firstly calculating the increment between the two allocations according to the objective function, If the increment is negative, accepting this allocation as the new current allocation, otherwise, probabilistically deciding whether to accept or not.

In order to guarantee the performance that the response time is in an acceptable bound, so we firstly estimate whether the new SR_k satisfy the performance requirements or not, i.e., the response time $\overline{RT_k}$ should be lower than t_0.

If the new SR_k satisfies the response time requirement, then the transition decision from the current state to the

new stated is based on the simulated annealing criterion, which ensures the algorithm can search the global optimal solution rather than local optimum.

If the response time $\overline{RT_k}$ is higher than the t_0, it will be directly compared with the $\overline{RT}$ of current allocation, this judgement is to guarantee the system response time can be optimized and approached to the performance objective.

VI. Experimental Results

In this section, we validate the effectiveness of our algorithm proposed in this paper by the experimental results.

The workload in our experiments is set to 1500 concurrent users, based on the relationship analysis between the model parameters and experimental measurements data in Section IV, we can obtain the two key factors μ_c and $\Delta\mu$, which is further used in our algorithm.

Fig. 6 shows the variations of ρ_{tot} and $\overline{RT}$ with the increase of the iteration in our algorithm. ρ_{tot} means the total average CPU utilization of severs in three tiers, so the value of ρ_{tot} may be higher than 1. From the curve ρ_{tot}, the trend gradually levels off, the final result approaches the minimum and it is reasonable to consider the final soft resource allocation as the optimal allocation at the given workload 1500 concurrent users. This trend also greatly illustrates the process of searching the global optimal solution by using simulated annealing algorithm, in which the worse solution may also be accepted at higher temperature (in the first few iterations) to extend the solution space and establish the foundation for global optimum.

Fig. 7 plots the variations of soft resource allocation, i.e., N_1, N_2 and N_3, during the iterative process. All the three soft resources are randomly generated in the process, combined with the total CPU utilization and response time trend in Fig. 6, it shows that carefully tuning the three soft resources allocation rather than simply increasing or decreasing all the three soft resources can obtain efficient resource utilization.

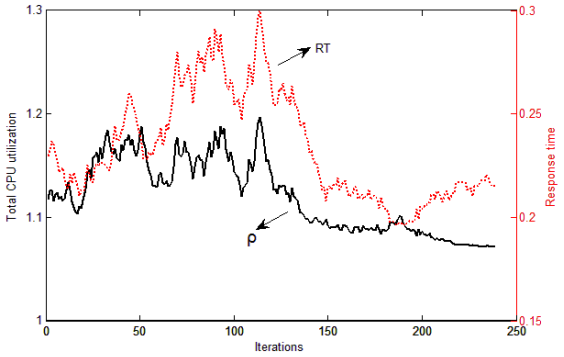

Fig. 6 ρ_{tot} and $\overline{RT}$ variations in the iteration process.

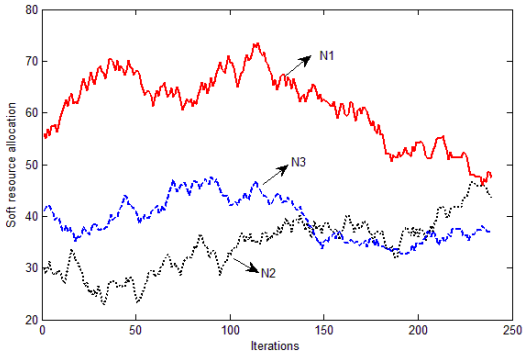

Fig. 7 N_1, N_2 and N_3 variations in the iteration process

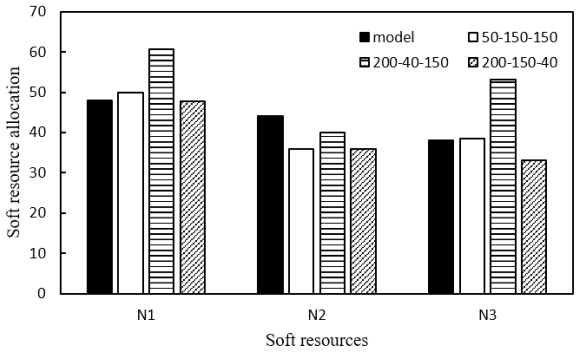

Fig. 8 Soft resource allocation comparison of optimal model result and experimental results.

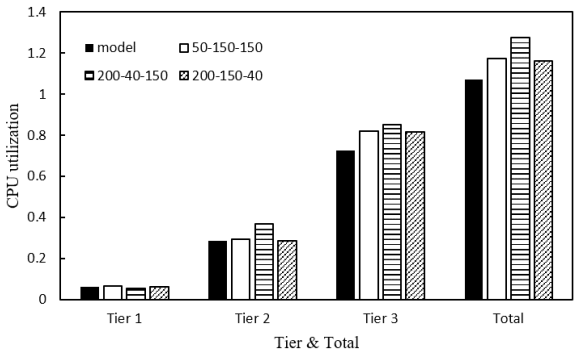

Fig. 9 CPU utilization comparison of optimal model result and experimental results.

The optimal model result for the given workload 1500 concurrent users is obtained by using our algorithm, which is 48, 44, and 38 respectively. We select other three representative experimental data to compare with the model result, which is shown as Fig. 8. Each soft resource allocation we select has one limited tier's soft resource but close to the model result we obtain. Note that the N_1, N_2 and N_3 of the three selected soft resource allocation represent the real running threads in the server. It can be

found that the model result is close to the real running threads, which means the soft resource allocation obtained from the algorithm can meet the threads demand. Fig. 9 shows the CPU utilization comparison from each tier to total system. The CPU utilization of optimal model result is calculated by the algorithm. Through the comparison, it is clearly that the CPU utilization of the model result is almost the lowest whether single tier or the total system, especially the total CPU utilization, which is also our optimization objective. As we see from Fig. 6, the performance is also guaranteed in that soft resource allocation, i.e., the response time is between 0.2 seconds and 0.25 seconds. In one words, our experimental results show that our approach can work well for optimizing the soft resource allocation.

VII. CONCLUSION

In this paper, based on the characteristics of the multi-tier web systems, we firstly theoretically formulate single-server in each tier by M/M/n queueing model, and then a Jackson queueing network is applied to model the whole system. Detailed mathematical analysis for model parameters and performance indicators is carried out. Based on the RUBiS system which has been deployed in reality, the relationship between the model parameters and measurement data is studied, and the procedures of obtaining model parameters from experimental data are described. Finally, at the foundation of the simulated annealing algorithm, an optimization algorithm for soft resource allocation in multi-tier web systems is proposed, whose effectiveness is validated by experimental results.

ACKNOWLEDGMENT

The research work was supported by the National Natural Science Foundation of China (No.61572295, 61272241, 61502043), the Innovation Method Fund of China (No.2015IM010200), the TaiShan Industrial Experts Programme of Shandong Province, the Natural Science Foundation of Shandong Province of China (No.ZR2014FM031, ZR2013FQ014), the Shandong Province Science and Technology Major Special Project (No.2015ZDXX0201B03, 2015ZDXX0201A04, 2015ZDJQ01002), the Shandong Province Key Research and Development Plan (No.2015GGX101015, 2015GGX101007), the Fundamental Research Funds of Shandong University (No.2015JC031), Beijing Natural Science Foundation (No.4162042), BeiJing Talents Fund (No.2015000020124G082).

REFERENCES

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

[2] H. Jamjoom, C.-T. Chou, and K. Shin, “The impact of concurrency gains on the analysis and control of multi-threaded internet services,” in INFOCOM ’04, 2004.

[3] Y. Jin, and T. Masahiko. "Web server performance enhancement by suppressing network traffic for high performance client." *Network Operations and Management Symposium (APNOMS), 2015 17th Asia-Pacific*. IEEE, 2015.

[4] V. Beltran, J. Torres, and E. Ayguadé. "Improving web server performance through main memory compression," in *14th IEEE International Conference on Parallel and Distributed Systems, (ICPADS'08)*. IEEE, 2008.

[5] J. Jiang, J. Lu, G. Zhang, and G. Long, “Optimal cloud resource autoscaling for web applications,” in *13th IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Computing*, pp. 58–65, 2013.

[6] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal, “Dynamic Provisioning of Multi-Tier Internet Applications,” in *IEEE 2nd Int. Conf. on Autonomic Computing*, pp. 217–228, 2005.

[7] Q. Zhu and G. Agrawal, “Resource Provisioning with Budget Constraints for Adaptive Applications in Cloud Environments,” in *Proc. 19th ACM Int. Symp. on High Performance Distributed Computing*, pp. 304–307, 2010.

[8] Y. Zhang, A. Bestavros, M. Guirguis, I. Matta, and R. West, “Friendly Virtual Machines: Leveraging a Feedback-Control Model for Application Adaptation,” in *Proc. 1st ACM/USENIX Int. Conf. on Virtual Execution Environments*, pp. 2–12, ACM, 2005.

[9] P. Lama, Y. Guo, and X. Zhou, “Autonomic performance and power control for co-located web applications on virtualized servers,” in *IEEE/ACM 21st Int. Symp. on Quality of Service*, pp. 1–10, 2013.

[10] 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 *31st Int. Conf. on Distributed Computing Systems*, pp. 571–580, IEEE, 2011.

[11] J. Bi, H. Yuan, W. Tan, M. Zhou, Y. Fan, J. Zhang & J. Li. “Application-Aware Dynamic Fine-Grained Resource Provisioning in a Virtualized Cloud Data Center,” in *IEEE Transactions on Automation Science and Engineering*, pp. 1-13, 2016.

[12] J. Li, Q. Wang, D. Jayasinghe, S. Malkowski, P. Xiong, C. Pu, Y. Kanemasa, and M. Kawaba, "Profit-based experimental analysis of iaas cloud performance: Impact of software resource allocation," in *2012 IEEE Ninth International Conference on Services Computing (SCC)*, pp. 344-351.

[13] J. Li, Q. Wang , C. A. Lai, J. Park, D. Yokoyama, & C. Pu. “The Impact of Software Resource Allocation on Consolidated n-Tier Applications” in *2014 IEEE 7th International Conference on Cloud Computing (CLOUD),* pp. 320-327.

[14] Y. Xia, M. Zhou, X. Luo, S. Pang & Q. Zhu. “Stochastic Modeling and Performance Analysis of Migration-Enabled and Error-Prone Clouds,” in *IEEE Transactions on Industrial Informatics*, 11(2), pp. 495-504, April 2015.

[15] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, “An Analytical Model for Multi-Tier Internet Services and Its Applications,” in *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, pp. 291–302, 2005.

[16] Y. Xia, M. Zhou, X. Luo, Q. Zhu, J. Li & Y. Huang. “Stochastic modeling and quality evaluation of infrastructure-as-a-service clouds,” in *IEEE Transactions on Automation Science and Engineering*, 12(1), pp. 162-170, Jan. 2015.

[17] E. Chlebus and J. Brazier, “Nonstationary poisson modeling of web browsing session arrivals,” in *Information Processing Letters*, vol. 102, no. 5, pp. 187–190, 2007.

[18] RUBiS: Rice University Bidding System http://rubis.ow2.org/.

[19] Sysstat Package in Linux: http://sebastien.godard.pagesperso-orange.fr/download.html.

[20] Y. Diao, J. Hellerstein, S. Parekh, H. Shaikh, and M. Surendra, “Controlling Quality of Service in Multi-Tier Web Applications,” in *26th IEEE Int. Conference on Distributed Computing Systems*, pp. 25–25, 2006.