State Space Model and Queuing Network Based Cloud Resource Provisioning for Meshed Web Systems

Yamin Lei, Zhicheng Cai, *Member, IEEE*,
Xiaoping Li, *Senior Member, IEEE*, and Rajkumar Buyya, *Fellow, IEEE*

Abstract—Functions provided by Web applications are increasingly diverse which make their structures complicated and meshed. Cloud computing platforms provide elastic computing capacities for these meshed Web systems to guarantee Service Level Agreement (SLA). Though workloads of meshed Web systems usually change steadily and periodically in total, sometimes there are sudden fluctuations. In this paper, a hybrid State-space-model-and-Queuing-network based Feedback control method (SQF) is developed for auto-scaling Virtual Machines (VMs) allocated to each tier of meshed Web systems. For the case with workloads changing steadily, a State-space-model based static Feedback Control method (SFC) is proposed in SQF to stabilize request response times near the reference time. For unsteadily changing workloads, a Queuing-network based multi-tier collaborative Feedback Control method (QFC) is proposed for effectively eliminating bottlenecks. QFC builds a control system for each tier individually and uses the queuing network to measure the interaction relationships among different tiers. Experimental results show that QFC is able to improve the efficiency of eliminating bottlenecks (decreasing upper-limit SLA violation ratios by 31.99%∼56.52%) with similar or a little bit high VM rental costs compared to existing methods while SFC obtains more stable response times for requests with reasonable additional costs.

Index Terms—Bottleneck eliminating, cloud computing, feedback control, resource provisioning, state-space model

1 Introduction

Machine learning and data mining techniques are gradually applied to deal with various requests of Web systems to enrich the provided functions of services [1], which makes the structure of a Web system so complicated that it has a meshed topology. A meshed Web system usually contains multiple tiers, each of which provides a single atomic function. Each tier can be deployed on Virtual Machines (VMs) elastically rented from public Clouds to obtain dynamic computing power. VMs allocated to each tier might be insufficient or excessive as workloads change dynamically. When VMs of one tier are insufficient, request response times become longer than the allowed upper limit specified in Service Level Agreement (SLA) leading to a bottleneck tier. On the contrary, excess resources make response times much shorter than the upper limit incurring resource wastes. Therefore, it is crucial to design elastic VM provisioning methods for allocating an appropriate number of VMs to each tier of meshed Web systems to stabilize Mean Response Times (MRTs) with minimal VM rental costs ensuring good service experience for users. The main challenges of VM provisioning for meshed Web systems consist of complex meshed relationships, nonlinear VM performance models and dynamic workloads.

Because VM performance models are nonlinear, it is hard to stabilize the MRTs of each tier at a reference time smaller than the upper limit. For each single tier, queuing models have been widely used to describe the nonlinear relationships among the minimum number of VMs, the MRT and the request arrival rate. But it is still hard to keep MRTs closely following the reference time because of inevitable deviations between the real system and queuing models. Therefore, the feedback control has been used to amend these deviations. However, feedback controllers of existing algorithms [2], [3] are usually designed for each tier separately without considering the interactions among different tiers. Meanwhile, because VMs are provisioned in coarse granularity, a VM is added or released only if the change of workloads exceeds a threshold. A slight workload change may not cause adjustments to the number of allocated VMs based on queuing models, but could make MRTs fluctuate [2], [4]. Therefore,

- *Yamin Lei is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China, and also with Southeast University, Nanjing 211189, China. E-mail: yaminlei@126.com.*
- *Zhicheng Cai is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China. E-mail: caizhicheng@njust.edu.cn.*
- *Xiaoping Li is with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China. E-mail: xpli@seu.edu.cn.*
- *Rajkumar Buyya is with Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Melbourne 3010, Australia. E-mail: rbuyya@unimelb.edu.au.*

Manuscript received 24 Aug. 2021; revised 22 Mar. 2022; accepted 25 Apr. 2022. Date of publication 27 Apr. 2022; date of current version 26 July 2022.
This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1402500, in part by the National Natural Science Foundation of China under Grants 61972202, 61872186, 61872077, and 61832004, and in part by the Fundamental Research Funds for the Central Universities under Grant 30919011235.
(Corresponding author: Zhicheng Cai.)
Recommended for acceptance by D. Grosu.
Digital Object Identifier no. 10.1109/TPDS.2022.3170834

both single-tier based pure queuing-models and feedback control methods are not suitable for stabilizing MRTs of meshed Web systems.

Complex interactions among tiers are the main reason for bottleneck shifting. A surge in workloads usually leads to serious bottlenecks, and it is necessary to eliminate bottlenecks quickly to stabilize MRTs. However, eliminating one bottleneck tier may cause new bottleneck tiers (called bottleneck shifting) due to sudden passed requests to other tiers. Thus, it is a challenge to determine which tiers' VMs to scale up simultaneously to avoid bottleneck shifting. A Proportional Provisioning Method (PPM) was proposed to adjust allocated servers according to the ratio of each tier's arrival rate to the total arrival rate [5], [6]. But the ratios change as the proportions of request types change which decreases the performance of PPM. Therefore, a Jackson network based Reactive Bottleneck Elimination method (JRBE) [7] was developed to estimate the impacts of eliminating bottleneck tiers on other tiers, which was able to eliminate bottlenecks while avoiding bottleneck shifting. However, the inaccuracy of queuing models in the Jackson network degrades the performance of JRBE due to lack of reacting to real-time deviations from MRTs to the reference time.

In this paper, a hybrid State-space-model-and-Queuing-network based Feedback control method (SQF) is proposed for meshed Web systems to stabilize MRTs under an upper limit with minimum VM rental costs. The main contributions are: (1) When the system works near the reference mean response times, a State-space-model based static Feedback Control method (SFC) is developed in SQF to describe the linear relationships among all tiers and mitigate fluctuations of MRTs near reference points. The state-space based static feedback controller is more stable than multiple queuing models. (2) For serious bottlenecks in the meshed Web system, a Queuing-network based multi-tier collaborative Feedback Control method (QFC) is designed in SQF to improve the efficiency of eliminating bottlenecks. The feedback control result of each tier is applied to calculate impacts of bottleneck tiers to other tiers more accurately.

The rest of the paper is organized as follows. Section 2 reviews related works. The problems are described in Section 3. Section 4 introduces SQF and Section 5 shows experimental results. Conclusions and future work are depicted in Section 6.

2 Related Work

It has attracted much attention about how to provision resources for single-tier and multi-tier Web systems which are surveyed separately.

2.1 Resource Provisioning for Single-Tier Web Systems

For single-tier Web applications, resource provisioning techniques are mainly based on threshold [8], [9], time series analysis [10], [11], [12], queuing theory [2], [3], [13], [14], [15], [16], [17] and control theory [2], [3], [17], [18], [19], [20], [21]. Threshold-based methods usually compare system characteristics (e.g., CPU utilization) with the predefined thresholds to trigger corresponding resource scaling actions. However, whether thresholds are fixed [8] or adjusted automatically [9], the performance is affected by specific workload patterns. Considering the periodic fluctuations of workloads, Roy *et al.* [10] used a second order AutoRegressive Moving Average method (ARMA) to predict the incoming workloads. To compensate for the inaccuracy of using a single forecasting model, the genetic algorithm was used to combine predictive results of several time series analysis models [11]. Based on predicted workloads, a variety of queuing models such as M/M/N [2], [11], [13], [14], G/GI/N [15], GI/G/N [16] and G/G/N [17] have been used to describe the relations among the MRT, allocated amount of resources and the request arrival rate for Web applications. And deviations between queuing models and the real system performance can be reduced by combining with control theory. For example, some works improved traditional queuing models by adjusting the arrival-rate adjustment coefficient [2] or the processing rate [17], using the arrival-rate-related processing rate [3], or adding an additional value to the output of queuing models [21] through feedback controllers.

2.2 Resource Provisioning for Multi-Tier Web Systems

For multi-tier Web systems, the most intuitive way is to apply techniques designed for single-tier Web systems directly to auto-scale resources of each single tier independently [22], which ignores interactions among tiers degenerating the performance. Thus, tiered-but-joint based strategies are more suitable for multi-tier Web systems, such as queuing network [5], [6], [23], [24], [25], [26], [27], control theory [28], [29], [30], [31], [32] and machine learning [33], [34], [35], [36], [37], [38]. Urgaonkar *et al.* [5], [6], [27] and Cunha *et al.* [23] modeled the whole Web system as a queuing network, and auto-scaled resources of each tier by G/G/1 and M/M/1, respectively. And a hybrid queuing network consisting of M/M/c and M/M/1 was established by Bi *et al.* to provision VMs for each tier [24], [25]. Moreover, Jiang *et al.* [26] allocated the SLA to each tier dynamically based on M/M/N queuing models. As mentioned above, control theory can be applied to reduce the inaccuracy of queuing models. For example, a Multiple Input Multiple Output (MIMO) controller was developed for two-tier systems to adjust the CPU of VMs [28], [29]. However, no feedback control method has been proposed for queuing-network based systems. No prior knowledge of system architectures and workload patterns is required by machine-learning based methods. For example, Iqbal *et al.* [33] and Rao *et al.* [36] selected increment, decrement or no operation of resources for a two-tier Web system by reinforcement-learning. And a machine-learning based performance modeling method was proposed to predict the end-to-end tail latency of each containerized micro-service [38], which was similar to one tier of meshed Web systems. But the high accuracy required a long period of constant trial-and-errors. For eliminating appeared bottlenecks, Salah *et al.* [39], [40] tried to eliminate each bottleneck tier individually. However, these methods might cause bottleneck shifting [5], [6]. Thus, PPM [5], [6] and JRBE [7] were proposed to solve this problem.

2.3 Comparison to Existing Studies

A comparison between our approach and existing methods is showed in Table 1. Interactions among tiers of meshed

TABLE 1
A Comparison Between SQF and Existing Resource Provisioning Methods for Web Systems

Methods	Web Systems	Resource Provisioning	Bottleneck Eliminating
[8], [9]	Single-tier	Threshold	×
[10], [11], [12]	Single-tier	Time series analysis	×
[2], [3], [13], [14], [15], [16], [17]	Single-tier	Queuing theory	×
[2], [3], [17], [18], [19], [20], [21]	Single-tier	Control theory	×
[22]	Linear two-tier	Considering each tier separately	×
[5], [6]	Linear multi-tier	Queuing network	Scaling up all tiers proportionally
[23], [24], [25]	Linear multi-tier	Queuing network	×
[28], [29], [30], [31], [32]	Linear two-tier	Control theory	×
[33]	Linear two-tier	Reinforcement learning	Reinforcement learning
[34], [35], [36], [37]	Linear two-tier	Reinforcement learning	×
[39], [40]	Linear two-tier	Embedded Markov chain	Solving non-decreasing function
[38]	Meshed multi-tier	Machine learning technique	×
[26]	Meshed multi-tier	Multi-tier negotiating	Multi-tier negotiating
[7]	Meshed multi-tier	Jackson queuing network	Jackson queuing network
Our approach SQF	Meshed multi-tier	State-space model, queuing network and control theory	Queuing network and control theory

Web systems are not considered in most existing approaches. And for some methods applying queuing networks to describe interaction relations, the inaccuracy of queuing models is not considered. On the contrary, our approach SQF applies the state-space model to describe relationships among tiers accurately. And SQF combines the queuing network and the control theory to amend the inaccuracy of queuing networks.

3 Problem Description

Cloud meshed Web systems usually provide diverse services (businesses) such as picture searching, machine learning, business recommendation and so on. Fig. 1 shows a platform with networked structures and several collaborative tiers, which are commonly implemented in the form of micro-services collaborating through RESTful APIs. All user requests are received by the Router Tier first, and a sequence of tiers will be called in different orders determined by the business type of each request. The request generated by a previous tier to invoke another subsequent tier is called a sub-request of the original request. For example, for a request of the business type searching for similar pictures, the Router Tier generates a sub-request to the Searching Engine Tier first to accept a submitted picture. Next, the Searching Engine Tier generates a sub-request to the Machine Learning Tier to analyse the submitted picture. Finally, the Machine Learning Tier generates a sub-request to the Database Tier to obtain similar pictures. All tiers are state-less, and sub-requests can be generated only after ancestor sub-requests generating them have been processed completely (asynchronous calls). A meshed Web System usually provides several different business types with different tier accessing orders (accessing paths). Using a divide and conquer strategy, the total SLA of each type of business is determined by setting a fixed sub-SLA for each tier considering the trade-off between cost and user experience. Because VMs are usually charged by small intervals (e.g., seconds or minutes), provisioning resources for each tier individually (each VM only hosts one tier) is helpful for renting (releasing) VMs flexibly from (to) public Clouds based on the real-time requests of each tier. Meanwhile, because VMs of each tier usually process the same kind of tasks, multiple homogeneous VMs are elastically rented for each tier. Different tiers need the same or different types of VMs, and one VM can be held by only one tier at one time. However, VMs released by one tier could be reused directly by other tiers which need the same type of VMs.

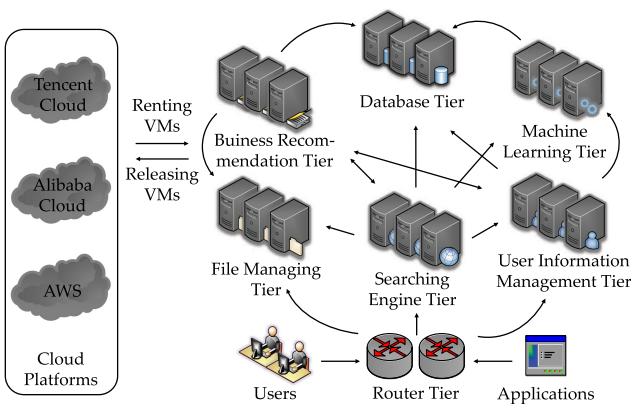


Fig. 1. An example of meshed Web systems with multiple tiers.

The objective of this paper is to design a VM auto-scaling strategy which is able to minimize the VM rental cost of each tier L_i while satisfying two constraints defined in the sub-SLA of L_i. For each tier L_i, the first constraint is that the ratio $\bar{V}_i$ of the real MRTs greater than an upper limit R_i^{sla} should be lower than a threshold V^{sla} (e.g., 10%), which is called upper-limit SLA. And the second constraint is that the average deviation $\bar{D}_i$ from the real MRTs to the reference response time R_i^r (smaller than R_i^{sla}) should be lower than D^{sla} (e.g., 5%), which is called deviation SLA. The main function of deviation SLA is to make the MRTs stable which is selectable based on the requirements of service providers. T_c is the length of the control interval between two resource control actions and N_c is the number of total control intervals considered. $N_i(k)$ is the number of VMs rented for L_i during the kth control interval, and p_i is the price per minute for the VM of L_i. Then the optimization problem of each tier L_i ($i \in \{1, 2, \ldots, N^l\}$) is modeled as

$$\begin{aligned} \min \quad & \sum_{k=1}^{N_c} p_i \times T_c \times N_i(k) \\ s.t. \quad & \bar{V}_i < V^{sla} \quad and \quad \bar{D}_i < D^{sla} \end{aligned} \tag{1}$$

TABLE 2
A Summary of Frequently Used Notations in SQF

Notations	Descriptions
T_c	Length of a time interval between two resource control actions (minutes)
N^l	Number of tiers in the meshed Web system
L_i	ith tier of the meshed Web system
$\lambda_i^r(k)$	Real-time sub-request arrival rate of L_i in the kth control interval
$\mu_i(k)$	Total processing rate of all VMs allocated for L_i in the kth control interval
$r_i(k), y_i(k)$	Mean Response Time (MRT) and Mean Waiting Time (MWT) of L_i in the kth interval, respectively
$W_i^c(R_i^r), W_i^q$	Reference MWT (MRT) of feedback control and MWT of queuing model for L_i, respectively
$e_i(k)$	Output error of L_i in terms of MWT in interval k
$N_i(k)$	Allocated number of VMs in interval k for L_i

Frequently used notations are shown in Table 2.

4 Proposed VM Provisioning Method

Widely used queuing models only describe the performance of one tier without considering the interactions among multiple tiers, and likely lead to response time fluctuations. Fortunately, when the Web system works near the reference MRTs, linear models can be used to describe the relations among multiple tiers more accurately. Therefore, a State-space-model based static Feedback Control method (SFC) is developed to stabilize MRTs of each tier around reference times. However, when the system deviates greatly from the reference points, queuing models are still the methods which describe nonlinear performance models more accurately than linear models. Although the feedback control is an effective method to amend deviations between queuing models and the real system, existing queuing-model based feedback control methods (such as Unequal-interval based loosely coupled Control Method (UCM) [2]) are all tailored for single tiers. A feasible effective method is to apply the queuing network to model the relations among multi-tiers based on queuing models. In this paper, UCM and the queuing network are combined to construct a Queuing-network based multi-tier collaborative Feedback Control method (QFC) for unstable conditions. The hybrid of SFC and QFC is called State-space-model-and-Queuing-network based Feedback control method (SQF). Algorithm 1 is the formal description of the proposed SQF. In every interval T_c, SQF collects the average end-to-end response time $\bar{R}$ (the sum of all sub-requests' response times in each business type). If $\bar{R}$ is smaller than a predefined reference time $\hat{R}$, fluctuations of workloads are stable, SFC is applied to keep MRTs as stable as possible. Otherwise, there are bottlenecks, QFC is applied to eliminate bottlenecks avoiding shifting.

4.1 State-Space-Model Based Feedback Control

In SFC, a discrete linear state-space model for the whole meshed Web system with N^l tiers is first built. Based on the state-space model, the integral control is applied to establish a static feedback control system to stabilize MRTs of each tier quickly near the corresponding reference time.

Algorithm 1. Hybrid State-Space-Model-and-Queuing-Network Based Feedback Control (SQF)

Input: T_c, $\hat{R}$
Initialize renting plan $\Upsilon \leftarrow \phi$
while True **do**
  for every control interval T_c **do**
    Calculate average end-to-end response time $\bar{R}$;
    if $\bar{R} \leq \hat{R}$ **then**
      $\Upsilon \leftarrow$ Call SFC;
    else
      $\Upsilon \leftarrow$ Call QFC;
    end if
    Adjust VMs of each tier according to Υ;
  end for
end while

4.1.1 Discrete Linear State-Space Model

A linearized state-space model Ψ with Multiple Input Multiple Output (MIMO) is used to describe the interactive relationships among tiers which is as follows:

$$\Psi : \begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) \\ \boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{cases}, \tag{2}$$

where $\boldsymbol{x}(k) = (\lambda_1^r(k), \ldots, \lambda_i^r(k), \ldots, \lambda_{N^l}^r(k))^{\mathrm{T}}$ is the vector of arrival rates, $\boldsymbol{u}(k) = (\mu_1(k), \ldots, \mu_i(k), \ldots, \mu_{N^l}(k))^{\mathrm{T}}$ is the vector of total processing rates of VMs, and $\boldsymbol{y}(k) = (y_1(k), \ldots, y_i(k), \ \ldots, y_{N^l}(k))^{\mathrm{T}}$ is the vector of mean waiting times (MWTs) of all tiers. Although the objective is to stabilize MRTs, Ψ is built based on MWTs which does not contain processing times because MWTs are able to represent the degree of blocking more accurately. $\boldsymbol{A}$ is a $(N^l - 1) \times (N^l - 1)$ matrix describing the linear relations among arrival rates of different tiers, $\boldsymbol{C}$ and $\boldsymbol{D}$ are both a $(N^l - 1) \times (N^l - 1)$ diagonal matrix modeling the relations among MWTs, arrival rates and processing rates of each tier.

4.1.2 State-Space Feedback Controller

For state-space models, the dynamic feedback controller is used widely which is able to obtain a zero steady-error [41]. In order to use the dynamic feedback controller, the state vector should be extended to be $\boldsymbol{m}(k) = [\boldsymbol{x}(k), \boldsymbol{e_I}(k)]^{\mathrm{T}}$ by adding the integral output error

$$\boldsymbol{e_I}(k+1) = \boldsymbol{e_I}(k) + \boldsymbol{e}(k) = \boldsymbol{e_I}(k) + \boldsymbol{r} - \boldsymbol{y}(k), \tag{3}$$

where $\boldsymbol{r} = (W_1^c, \ldots, W_i^c, \ldots, W_{N^l}^c)^{\mathrm{T}}$ is the vector of reference MWTs and $\boldsymbol{e}(k) = (e_1(k), \ldots, e_i(k), \ldots, e_{N^l}(k))$ is the vector of control output errors. In the steady state, $\boldsymbol{e_I}(k+1) = \boldsymbol{e_I}(k)$. Therefore, $\boldsymbol{e}(k) = \boldsymbol{r} - \boldsymbol{y}(k) = 0$ means the steady state output error is zero. Then, the augmented state-space model is obtained based on (2) and (3) as follows.

$$\boldsymbol{m}(k+1) = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ -\boldsymbol{C} & \boldsymbol{I} \end{bmatrix} \boldsymbol{m}(k) + \begin{bmatrix} \boldsymbol{0} \\ -\boldsymbol{D} \end{bmatrix} \boldsymbol{u}(k) + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{r}. \tag{4}$$

The control input $\boldsymbol{u}(k)$ of the dynamic feedback controller is proportional to the state vector $\boldsymbol{x}(k)$ and the integral output error $\boldsymbol{e_I}(k)$ as shown in Fig. 2.

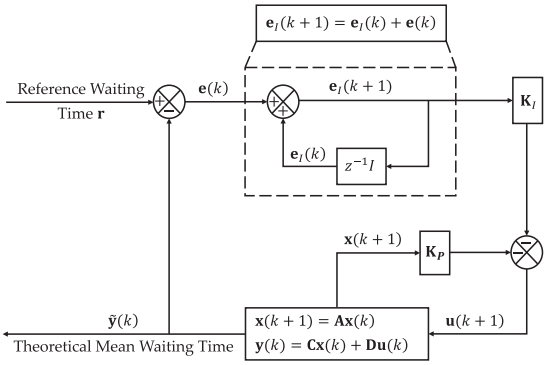


Fig. 2. The overall framework of SFC for meshed Web systems.

$$\boldsymbol{u}(k) = -[\,\boldsymbol{K}_P \quad \boldsymbol{K}_I\,] \times \begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{e_I}(k) \end{bmatrix}. \tag{5}$$

To select appropriate values for control gains $\boldsymbol{K}_P, \boldsymbol{K}_I$, Linear Quadratic Regulator (LQR) [41] is applied to minimize the objective function

$$J = \frac{1}{2}\sum_{k=0}^{\infty}[\boldsymbol{m}^{\mathrm{T}}(k)\boldsymbol{Q}\boldsymbol{m}(k) + \boldsymbol{u}^{\mathrm{T}}(k)\boldsymbol{R}\boldsymbol{u}(k)]. \tag{6}$$

Where $\boldsymbol{Q}$ and $\boldsymbol{R}$ are used to keep a balance between the control accuracy $\boldsymbol{m}(k)$ and control effort $\boldsymbol{u}(k)$. Smaller control errors usually need larger control efforts. LQR aims to minimize control errors with appropriate control efforts. $\boldsymbol{Q}$ determines which states in $\boldsymbol{m}(k)$ converge to the corresponding reference points more quickly than others. But $\boldsymbol{x}(k)$ is not determined by $\boldsymbol{u}(k)$ in this paper. Therefore, $\boldsymbol{Q}$ is mainly used to control $\boldsymbol{e_I}(k)$. $\boldsymbol{R}$ determines the impacts of different control inputs on the objective function.

In most existing methods, control error $\boldsymbol{e}(k)$ is the difference between the reference MWT and the real MWT. However, the real MWT deviates from its theoretical value greatly sometimes making the control input fluctuates greatly. The main reason is that a fixed linear state-space model is still not very accurate although it is able to describe the relations among multiple tiers in some degree. Therefore, in this paper, the real MWT is replaced by its theoretical value to calculate the output error to minimize the fluctuations. The vector of theoretical MWTs $\tilde{\boldsymbol{y}}(k) = (\tilde{y}_1(k), \ldots, \tilde{y}_i(k), \ldots, \tilde{y}_{N^l}(k))$ can be obtained by substituting (5) into (2) as follows:

$$\begin{aligned} \tilde{\boldsymbol{y}}(k) &= [\,\boldsymbol{C} \quad \boldsymbol{0}\,]\begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{e_I}(k) \end{bmatrix} + \boldsymbol{D}\boldsymbol{u}(k) \\ &= [\,\boldsymbol{C} \quad \boldsymbol{0}\,]\begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{e_I}(k) \end{bmatrix} - \boldsymbol{D}[\,\boldsymbol{K}_P \quad \boldsymbol{K}_I\,]\begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{e_I}(k) \end{bmatrix} \\ &= [\,\boldsymbol{C} - \boldsymbol{D}\boldsymbol{K}_P \quad -\boldsymbol{D}\boldsymbol{K}_I\,]\begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{e_I}(k) \end{bmatrix}. \end{aligned} \tag{7}$$

After using this theoretical MWT, the dynamic feedback control degrades into a kind of static control which changes the control input only based on the arrival rates and theoretical output errors [41]. In other words, it only reacts to arrival rate changes ignoring real MWT changes. Meanwhile, in order to avoid large changes of control inputs, a four-thresholds based control error computing method is applied to obtain $\boldsymbol{e}(k)$. For each tier L_i, when the theoretical MWT $\tilde{y}_i(k)$ is within the first interval $[W_i^c \times \rho_l^i, W_i^c \times \rho_u^i]$, $e_i(k)$ equals to zero. If $\tilde{y}_i(k)$ exceeds the first interval but is still within the second interval $[W_i^c \times \eta_l^i, W_i^c \times \eta_u^i]$, $e_i(k)$ is defined to be the difference between $\tilde{y}_i(k)$ and the lower or upper threshold of the first interval. And if $\tilde{y}_i(k)$ exceeds the second interval, $e_i(k)$ equals to the difference between the lower or upper thresholds of two intervals.

$$e_i(k) = \begin{cases} W_i^c \times (\rho_l^i - \eta_l^i) & 0 < \tilde{y}_i(k) \le W_i^c \times \eta_l^i \\ W_i^c \times \rho_l^i - \tilde{y}_i(k) & W_i^c \times \eta_l^i < \tilde{y}_i(k) < W_i^c \times \rho_l^i \\ 0 & W_i^c \times \rho_l^i \le \tilde{y}_i(k) \le W_i^c \times \rho_u^i \\ W_i^c \times \rho_u^i - \tilde{y}_i(k) & W_i^c \times \rho_u^i < \tilde{y}_i(k) < W_i^c \times \eta_u^i \\ W_i^c \times (\rho_u^i - \eta_u^i) & W_i^c \times \eta_u^i \le \tilde{y}_i(k) \end{cases}. \tag{8}$$

4.1.3 Formal Description of Proposed SFC

Algorithm 2 is the formal description of SFC. For each interval T_c, after the current arrival rate $\boldsymbol{x}(k)$ is detected, (7) is applied to calculate the theoretical MWT $\tilde{\boldsymbol{y}}(k)$, based on which the output error $\boldsymbol{e}(k)$ is obtained by (8). Then, the integral output error $\boldsymbol{e_I}(k+1)$ and $\boldsymbol{x}(k+1)$ can be calculated by applying (3) and (2), respectively. Finally, (5) is used to obtain the input vector $\boldsymbol{u}(k+1)$, based on which the VM number $N_i(k+1)$ of each tier is set to $\lceil \mu_i(k+1)/\bar{\mu}_i \rceil$ given the service rate per VM $\bar{\mu}_i$ in each tier. Meanwhile, each change of $N_i(k+1)$ cannot be larger than n compared with the existing VM number $N_i^r(k)$, and $N_i(k+1)$ should not be smaller than a minimum value N_i^{min}.

Algorithm 2. State-Space-Model Based Static Feedback Control (SFC)

Input: $\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{K}_P, \boldsymbol{K}_I, \boldsymbol{x}(k), \boldsymbol{e_I}(k), \bar{\mu}_i, n, N_i^r(k), N_i^{min}$
Output: Υ

Initialize $\Upsilon \leftarrow \phi$;
for every control interval T_c **do**
  $\tilde{\boldsymbol{y}}(k) \leftarrow (\boldsymbol{C} - \boldsymbol{D}\boldsymbol{K}_P) \times \boldsymbol{x}(k) - \boldsymbol{D}\boldsymbol{K}_I \times \boldsymbol{e_I}(k)$;
  Obtain $\boldsymbol{e}(k)$ based on (8) and $\tilde{\boldsymbol{y}}(k)$;
  $\boldsymbol{e_I}(k+1) \leftarrow \boldsymbol{e_I}(k) + \boldsymbol{e}(k)$;
  $\boldsymbol{x}(k+1) \leftarrow \boldsymbol{A}\boldsymbol{x}(k)$;
  $\boldsymbol{u}(k+1) \leftarrow -\boldsymbol{K}_P \times \boldsymbol{x}(k+1) - \boldsymbol{K}_I \times \boldsymbol{e_I}(k+1)$;
  for each tier $L_i, i \in \{1, 2, \ldots, N^l\}$ **do**
    Initialize $N \leftarrow \lceil \mu_i(k+1)/\bar{\mu}_i \rceil$;
    if $N_i^r(k) < N$ **then**
      $N_i(k+1) \leftarrow min(N_i^r(k) + n, N)$;
    else if $N_i^r(k) > N$ **then**
      $N_i(k+1) \leftarrow max(N_i^r(k) - n, N, N_i^{min})$;
    else
      $N_i(k+1) \leftarrow N_i^r(k)$;
    end if
    $\Upsilon \leftarrow \Upsilon \cup N_i(k+1)$;
  end for
end for
return Υ

4.2 Queuing-Network Based Feedback Control

In QFC, an improved version of UCM [2] is applied to control each tier independently and control results are combined by queuing networks to avoid bottleneck shifting. UCM is an effective method to amend the inaccuracy of queuing models

by adjusting a coefficient of arrival rate dynamically for a single tier. Therefore, in QFC, the original UCM is modified (the modified version is named D-UCM) by applying a direct arrival-rate coefficient adjusting strategy to cope with fast changes of workloads. At first, D-UCM is used to obtain the initial adjusted arrival rate (control output) of each tier based on real-time output errors. Next, the impacts among tiers are calculated based on the request transition probability and the control output of each tier to obtain a final arrival rate for each tier. Finally, the final arrival rate is used to calculate the required VM number of each tier based on the M/M/N model separately.

4.2.1 M/M/N Queuing Model With Adjustable Arrival Rate

In UCM, a M/M/N queuing model was applied as the feedforward control to obtain the required number of VMs of one tier which had an adjustable arrival rate. For a single tier L_i, let $\bar{\mu}_i$ be the processing rate of each VM, λ_i^r be the actual sub-request arrival rate and N_i be the number of allocated VMs. According to the M/M/N queuing model [2], [7], the expectation of the MWT of sub-requests is as follows:

$$W_s(N_i, \lambda_i^r, \bar{\mu}_i) = \frac{(\frac{\lambda_i^r}{\bar{\mu}_i})^{N_i} \frac{\lambda_i^r}{N_i \times \bar{\mu}_i}}{N_i!(1 - \frac{\lambda_i^r}{N_i \times \bar{\mu}_i})^2 \lambda_i^r} P_0, \tag{9}$$

where P_0 is the probability of no requests in the system. Given λ_i^r, $\bar{\mu}_i$ and the reference MWT of the queuing model W_i^q, the minimum number of VMs can be obtained based on (9) [7]. However, there are inevitable deviations between the original queuing models and the real system. Therefore, UCM used a coefficient to fix the arrival rate of each tier according to real-time output errors

$$\lambda_i^c(k+1) = \lambda_i^r(k) \times \varphi_i(k), \tag{10}$$

where $\varphi_i(k)$ was the adjustment coefficient in each tier.

4.2.2 Direct-Updating Based UCM for Single Tier

For each tier L_i, in order to obtain the adjustment coefficient $\varphi_i(k)$, UCM modeled the single-tier Web system as a first-order linear control system

$$y_i(k+1) = y_i(k) + u_i(k), \tag{11}$$

where $u_i(k)$ was the control input, i.e., the expected adjustment of the MWT $y_i(k)$. $u_i(k)$ can be obtained by using a proportional controller as follows:

$$u_i(k) = K_p^i \times e_i(k), \tag{12}$$

where K_p^i was the control gain. Let W_i^c be the reference MWT of the feedback control, then $e_i(k) = W_i^c - y_i(k)$ was the output error. And when $y_i(k)$ was within the range $[W_i^c \times \rho_l^i,\ W_i^c \times \rho_u^i]$, $e_i(k)$ was set to zero. The target to change the MWT by $u_i(k)$ to fix output errors was implemented by adjusting $\varphi_i(k)$. Let $\mu_i(k)$ be the total processing rate of VMs. According to the inverse M/M/1 model [2], in order to change the MWT from $y_i(k)$ to $y_i(k+1)$ approximately, the arrival rate should be adjusted by the ratio

$$\omega_i(y_i(k), y_i(k+1)) = \frac{y_i(k)(1 + \mu_i(k)(y_i(k+1)))}{y_i(k+1)(1 + \mu_i(k) y_i(k))}. \tag{13}$$

Then, the arrival rate adjustment coefficient $\varphi_i(k)$ was update by

$$\varphi_i(k) = \varphi_i(k-1) \times \omega_i(y_i(k), y_i(k+1)), \tag{14}$$

where ω_i was limited within the range $[\omega_l^i, \omega_u^i]$ to avoid large fluctuations.

However, UCM's performance decreases when workloads change quickly. The main reason is that the arrival rate adjustment coefficient $\varphi_i(k)$ of UCM is updated in the form of an integral, i.e., it reflects the control errors of all past steps. Meanwhile, $\varphi_i(k)$ is updated in small steps by setting threshold limitations. When workloads change greatly, different scales of $\varphi_i(k)$ are required. However, the current updating strategy makes the change of $\varphi_i(k)$ slow which is not suitable for large workload fluctuations. In this paper, D-UCM is proposed to improve UCM by setting

$$\varphi_i(k) = \omega_i(y_i(k), y_i(k+1)), \tag{15}$$

limited in the range of $[\omega_l^i, \omega_u^i]$ and this limitation is released when workloads increase greatly and MWTs are far from the reference time W_i^c. In D-UCM, only when $y_i(k)/W_i^c$ is smaller than a predefined constant θ_i, the limitation exists. The benefit of such direct updating is able to react to sudden workload changes quickly compared with the integral based adjusting method. But D-UCM might lead to fluctuations when one control step cannot narrow the output error into an acceptable level. For example, when the MWT is greater than W_i^c, a larger coefficient is set to increase allocated resources in the first step. However, because the output error becomes small, the coefficient of the second control step might be smaller than that of the first step leading to resource releasing. Therefore, in order to solve this problem, $K_p^i = 1$ is set to try the best to correct the output error in one step, and the feedback is not invoked when the output error is smaller than a threshold [2].

Algorithm 3. Direct-Arrival-Rate-Adjusting Based UCM (D-UCM)

Input: $\lambda_i^r(k)$, $y_i(k)$, W_i^c, K_p^i, θ_i, ω_l^i, ω_u^i
Output: $\lambda_i^c(k+1)$

```
for every control interval T_c do
  Obtain e_i(k) based on W_i^c and y_i(k);
  u_i(k) ← K_p^i × e_i(k);
  Calculate ω_i(y_i(k), y_i(k+1)) based on (11) and (13);
  φ_i(k) ← ω_i(y_i(k), y_i(k+1));
  if y_i(k)/W_i^c < θ_i then
    Trim φ_i(k) using [ω_l^i, ω_u^i];
  end if
  λ_i^c(k+1) ← λ_i^r(k) × φ_i(k);
end for
return λ_i^c(k+1)
```

Algorithm 3 is the formal description of D-UCM. For each interval T_c, the control error $e_i(k)$ is obtained based on the actual MWT $y_i(k)$ and the reference time W_i^c. Based on

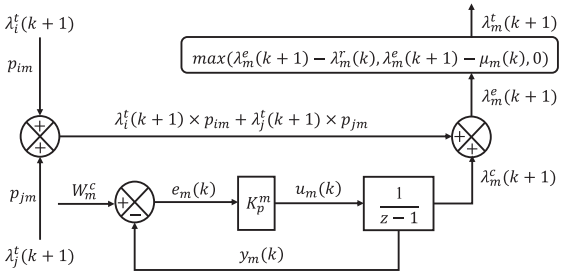


Fig. 3. A part of the feedback control system in QFC.

$e_i(k)$ and selected K_p^i, the expected adjustment $u_i(k)$ of $y_i(k)$ can be obtained. Then (11) and (13) are applied to obtain the adjustment ratio $\omega_i(y_i(k), y_i(k+1))$, based on which the arrival rate $\lambda_i^r(k)$ is adjusted by (15) and (10).

4.2.3 Collaborative Control Among Multiple Tiers

For meshed Web systems, there are interactions among tiers. Thus, for each tier L_i, the feedback control output $\lambda_i^c(k+1)$ obtained by D-UCM cannot be taken as the final arrival rate $\lambda_i^e(k+1)$. As shown in Fig. 3, $\lambda_i^e(k+1)$ is the sum of $\lambda_i^c(k+1)$ and the additional passing rate $\lambda_j^t(k+1)$ from other tiers:

$$\lambda_i^e(k+1) = \lambda_i^c(k+1) + \sum_{j=1}^{N^l} \lambda_j^t(k+1) \times p_{j,i}, \tag{16}$$

where $p_{j,i}$ is the probability of requests transferring from L_j to L_i, which can be estimated on-line based on historical requests, and $\lambda_j^t(k+1)$ is the number of additional sub-requests passing through L_j per second. Then, $\lambda_i^t(k+1)$ can be obtained based on $\lambda_i^e(k+1)$. Let $\lambda_i^r(k)$ be the current arrival rate and $\mu_i(k)$ be the total processing rate of all VMs. When L_i is stable ($\lambda_i^r(k) \leq \mu_i(k)$), $\lambda_i^t(k+1)$ is equal to $\lambda_i^e(k+1) - \lambda_i^r(k)$. Otherwise, $\lambda_i^t(k+1)$ equals $\lambda_i^e(k+1) - \mu_i(k)$. In other words, $\lambda_i^t(k+1)$ is always equal to $\lambda_i^e(k+1)$ minus the minimum value between $\lambda_i^r(k)$ and $\mu_i(k)$. Meanwhile, $\lambda_i^t(k+1)$ should not be smaller than zero. Mathematically

$$\lambda_i^t(k+1) = \max\{\lambda_i^e(k+1) - \lambda_i^r(k), \lambda_i^e(k+1) - \mu_i(k), 0\}. \tag{17}$$

$\lambda_i^e(k+1)$ cannot be obtained directly by solving linear equations similar with the traditional Jackson queuing network [42] because of the maximum operation in (17). Fortunately, when the meshed network is a Directed Acyclic Graph (DAG), $\lambda_i^e(k+1)$ of each tier can be calculated one by one based on the topological order.

4.2.4 Formal Description of Proposed QFC

Algorithm 4 is the formal description of QFC. For each interval T_c, the arrival rate $\lambda_i^c(k+1)$ is obtained by D-UCM. Then, (16) is used to calculate the final arrival rate $\lambda_i^e(k+1)$, based on which (17) is applied to determine the additional passing arrival rate $\lambda_i^t(k+1)$. Finally, given the reference MWT W_i^q of the M/M/N model, the renting plan including the estimated VM number $N_i(k+1)$ of each tier can be obtained by an exhausted search method as in [7].

Algorithm 4. Queuing-Network Based Multi-Tier Collaborative Feedback Control (QFC)

Input: $\lambda_i^r(k)$, $y_i(k)$, W_i^c, K_p^i, $\mu_i(k)$, $\bar{\mu}_i$, W_i^q
Output: Υ

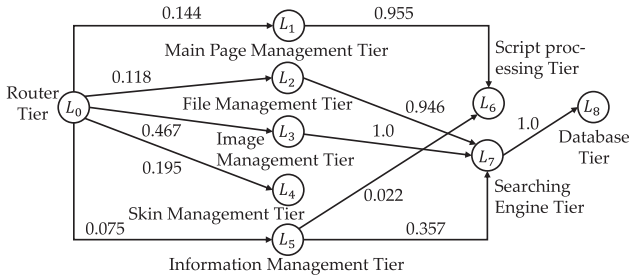
```
Initialize Υ ← φ;
for every interval T_c do
  for every tier L_i, i ∈ {1, 2, ..N^l} do
    λ_i^c(k+1) ← Call D-UCM(λ_i^r(k), y_i(k), W_i^c, K_p^i);
    Calculate λ_i^e(k+1) based on (16);
    Calculate λ_i^t(k+1) based on (17) given λ_i^e(k+1), λ_i^r(k) and μ_i(k);
    Obtain N_i(k+1) based on the M/M/N queuing model given λ_i^e(k+1), μ̄_i and W_i^q;
    Υ ← Υ ∪ N_i(k+1);
  end for
end for
return Υ
```

5 Performance Evaluation

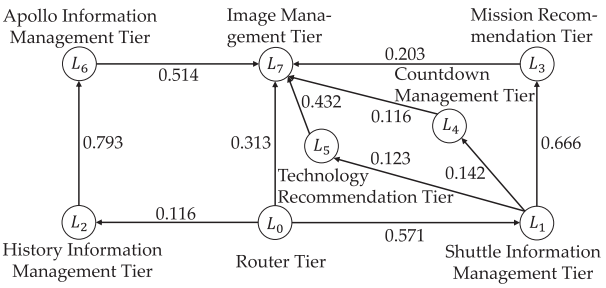
In this section, our approaches are compared with existing PPM [5], [6], JRBE [7] and PPE [26], which are all state-of-art algorithms for multi-tier Web systems. Experiments are performed on a simulation platform established based on the widely used CloudSim [43]. For simplification, we assume that all tiers rent on-demand VMs of Amazon EC2's "c4.2xL" charged by $0.007 per minute. Two Web applications: Wikipedia and NASA websites are simulated. The realistic Wikipedia access logs from 19th to 25th September in 2009 [44] are used to reconstruct the Wikipedia application. Each Uniform Resource Locator (URL) in Wikipedia traces is considered as an independent request, and each folder name in the URL is considered to be a tier needed to be visited by this request. In other words, several sub-requests accessing different tiers will be generated according to the order of folders in the URL one by one. Totally, there are nine tiers in the reconstructed meshed Wikipedia Web system as shown in Fig. 4a (called System I). Meanwhile, NASA website [45] with eight tiers as shown in Fig. 4b (called System II) is reconstructed and simulated in CloudSim based on its realistic access logs from 1st to 15th August in 1995 [45] similarly. After the systems are reconstructed, requests consisting of multiple sub-requests are generated based on the timestamp and the URL of their access logs. The mean size of requests s_i, the processing rate $\bar{\mu}_i$ per VM and the upper limit R_i^{sla} of response time in sub-SLA are shown in Table 3 for each tier L_i. $V^{sla} = 10\%$ and $D^{sla} = 5\%$ are set consistently for two systems. Meanwhile, in order to simulate the sudden surges of requests, the number of sub-requests arriving at L_3 in System I is increased to $2 \times \lambda_3^r$ and $3 \times \lambda_3^r$ in two sequential thirty-minute intervals separately, then decreased in the opposite order in the next two sequential thirty-minute intervals until returned to the stable state. And sub-requests in L_1 of System II are changed in the same way. Fig. 5 depicts dynamic changes of requests in the two systems.

5.1 Parameter Tuning

Although small control interval T_c is beneficial to react to workload changes timely, T_c should be greater than the VM setup time (about 2 minutes), and the control action takes



(a) System I.



(b) System II.

Fig. 4. The meshed Web systems reconstructed based on realistic access logs and the example of transition probabilities among tiers.

TABLE 3
Parameters of Meshed Web Systems

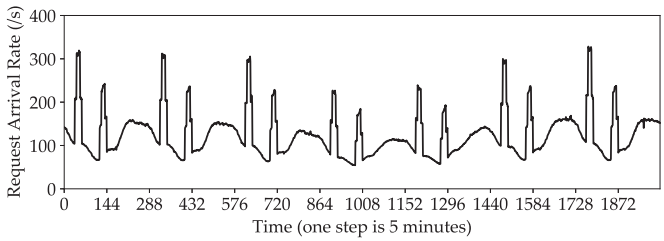
Parameters	System I		System II		
	L_1 to L_5	L_6 to L_8	L_1 to L_3	L_4 & L_5	L_6 & L_7
s_i(MIPS)	400	4000	500	5000	500
$\bar{\mu}_i$(requests/s)	38.75	3.875	31	3.1	31
R_i^{sla} (second)	0.04	0.40	0.06	0.60	0.06

several minutes to take effect. Therefore, T_c is set to be 5 minutes. When System I and II work near the reference points, the end-to-end MRTs of requests are usually smaller than $0.42\ s$ and $0.15\ s$ according to experiments, respectively. When systems deviate from their reference points, the end-to-end MRTs are greater than $0.42\ s$ and $0.15\ s$. Thus, $\hat{R}$ in System I and II are set to be $0.42\ s$ and $0.15\ s$, respectively.

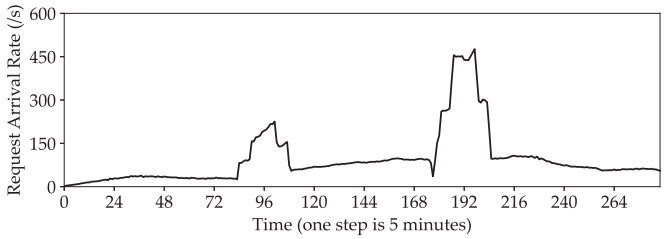
Parameters of SFC are shown in Table 4. For each tier L_i, an appropriate reference MWT W_i^c (reference MRT $R_i^r \approx W_i^c + 1/\bar{\mu}_i$) should be selected fulfilling the upper-limit SLA constraint. Greater W_i^c means the higher probability of violating SLA while smaller W_i^c means the higher resource renting cost which is usually selected by experiments [2]. ρ_l^i, ρ_u^i, η_l^i and η_u^i are set based on the granularity of MWT's changes in experimental results when one VM is added or released. Based on historical data, $\boldsymbol{A}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ are obtained by the least-square-regression based system identification method [41], [46]. And only data near the reference points is selected to fit the state-space model. Two control gains $\boldsymbol{K}_P$ and $\boldsymbol{K}_I$ for the state-space model are obtained by the LQR [41] based on (4) and (6) by setting $\boldsymbol{Q}$ and $\boldsymbol{R}$, which are tuned according to the overshoot and the settle time obtained by simulations.

Parameters of QFC are shown in Table 5. W_i^c of QFC are set to the same as those of SFC. For each tier L_i, the reference MWT W_i^q used by queuing models in QFC is set to be greater than W_i^c for feedback controllers, which is more likely to make MWTs fluctuate around W_i^c. Experiments are also carried out to evaluate the effectiveness of using θ_i as a switch of enabling the trim operation on $\varphi_i(k)$ based on $[\omega_l^i, \omega_u^i]$ and selecting appropriate values for θ_i. For example, Fig. 6 depicts the result of applying θ_3 on L_3 of System I which shows that using the θ_i-based trim operation is able to draw systems from unstable states back quickly.

In order to compare the performance of UCM and D-UCM, QFC using UCM and D-UCM are tested separately. Fig. 7 shows their MRTs of L_3 in System I, which denotes that D-UCM is more stable. The main reason is that the changing speed of arrival rate adjustment coefficient of UCM cannot catch up with the speed of workload changes and the coefficient of D-UCM can be changed quickly as needed as shown in Fig. 8.



(a) Sub-requests arriving at L_3 of System I in a week.



(b) Sub-requests arriving at L_1 of System II in one day.

Fig. 5. Dynamic fluctuations of requests in the meshed Web systems.

TABLE 4
Parameters of SFC

Parameters		$W_i^c(R_i^r)$(s)	ρ_l^i	ρ_u^i	η_l^i	η_u^i
System I	L_1 **to** L_5	0.002 (0.03)	0.85	1.15	0.65	1.35
	L_6 **to** L_8	0.012 (0.26)	0.85	1.15	0.75	1.25
System II	L_1 **to** L_3	0.004 (0.04)	0.90	1.10	0.65	1.35
	L_4 **&** L_5	0.02 (0.40)	0.90	1.10	0.65	1.35
	L_6 **&** L_7	0.004 (0.04)	0.90	1.10	0.80	1.20

TABLE 5
Parameters of QFC

Parameters	System I		System II		
	L_1 to L_5	L_6 to L_8	L_1 to L_3	L_4 & L_5	L_6 & L_7
W_i^c(second)	0.002	0.012	0.004	0.02	0.004
W_i^q(second)	0.004	0.017	0.008	0.027	0.008
ρ_l^i	0.90	0.90	0.90	0.90	0.90
ρ_u^i	1.10	1.10	1.10	1.10	1.10
ω_l^i	0.95	0.95	0.70	0.70	0.70
ω_u^i	1.05	1.05	1.30	1.30	1.30
θ_i	2000	2000	2000	2000	2000
K_p^i	1	1	1	1	1

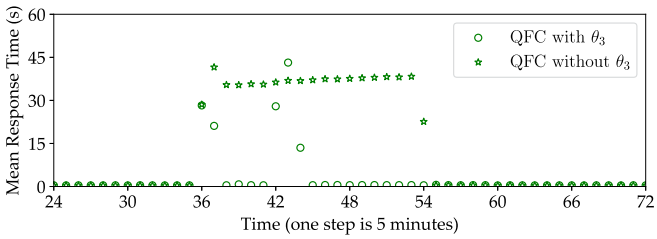


Fig. 6. Mean response times of QFC using θ_3 and not.

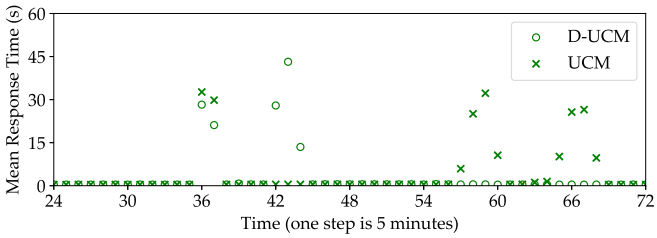


Fig. 7. MRTs of QFC with D-UCM or UCM in L_3 of System I.

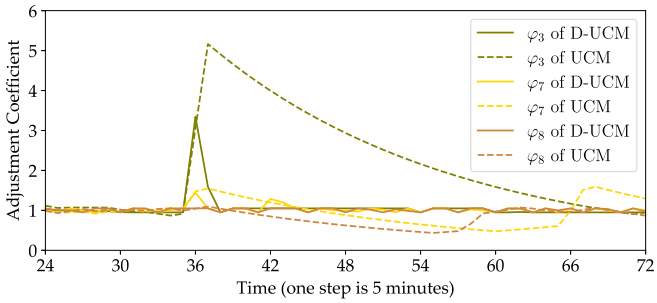


Fig. 8. Adjustment coefficients of QFC with D-UCM or UCM in different tiers of System I.

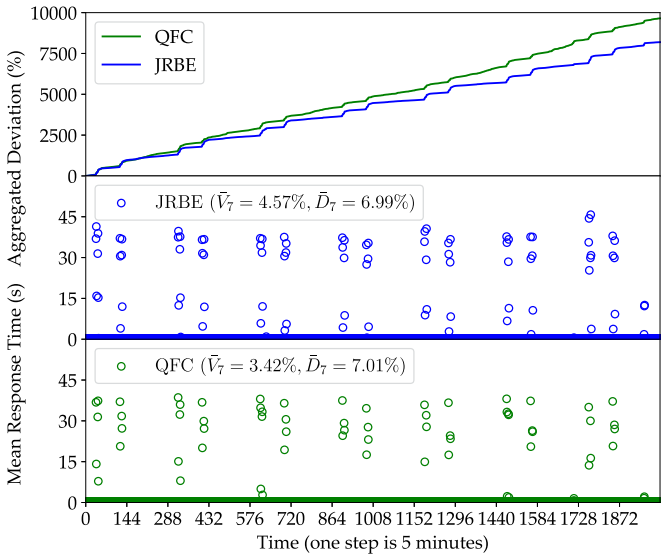


Fig. 9. MRTs and ADs of JRBE and QFC in L_7 of System I.

TABLE 6
Average Upper-Limit Violation Ratios and Deviations of Different Tiers in System I

Tier	QFC		JRBE		SQF	
	$\bar{D}_i(\%)$	$\bar{V}_i(\%)$	$\bar{D}_i(\%)$	$\bar{V}_i(\%)$	$\bar{D}_i(\%)$	$\bar{V}_i(\%)$
L_1	6.52	0	6.23	0	2.96	0
L_2	6.81	0	6.50	0	3.20	0
L_3	10.54	2.78	11.04	2.83	5.57	2.48
L_4	6.45	0	6.31	0	4.90	0
L_5	4.54	0	4.66	0	3.18	0
L_6	4.58	0.05	4.79	0.05	2.29	0
L_7	7.01	3.42	6.99	4.57	4.05	3.42
L_8	8.47	2.53	8.16	5.46	4.41	2.18
Sum	54.92	8.78	48.37	12.91	30.56	8.08

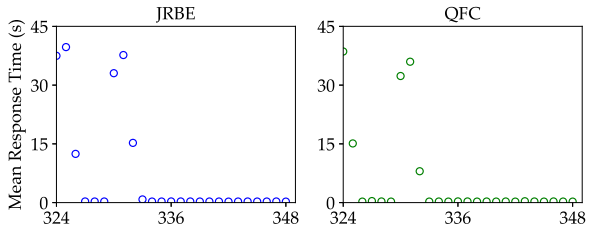
5.2 Experimental Results

Experiments are performed to evaluate the performance of adding the feedback control to queuing models, and combing the state-space based feedback control. And, our approach is also compared with other existing algorithms.

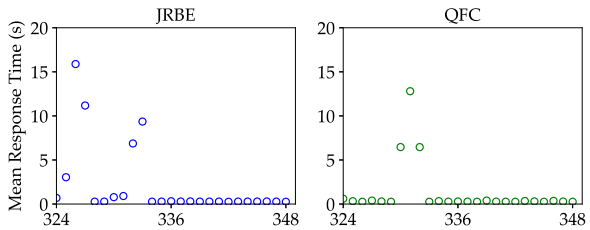
5.2.1 *Effectiveness of Feedback Control*

Existing algorithms JRBE [7] and PPM [5], [6] are methods based on pure queuing models, which are first compared with our queuing network and feedback control based hybrid method QFC to evaluate the effectiveness of applying the feedback control. For System I, since the arrival rate of L_3 is changed deliberately, L_3 becomes an original bottleneck tier periodically. L_7 and then L_8 are affected greatly by L_3 because there are high transition probabilities from L_3 to L_7 and then L_8 as shown in Fig. 4a. Fig. 9 displays MRTs of L_7, in which there are many overlapping blue circles violating the upper limit for JRBE (4.57%) while there are fewer green circles violating such an upper limit for QFC (3.42%). And the growth trend of QFC's Aggregated Deviations (ADs) over time is slightly quicker than that of JRBE. Table 6 shows similar trends for other tiers. The main reason is that JRBE calculates the number of required VMs only based on queuing models which have unavoidable deviations from real systems, while QFC uses the feedback control to correct deviations in time. And Fig. 10 shows enlarged images of MRTs belonging to a small interval $[324, 348]$ in L_7 and L_8 for illustrating this clearly, and also manifests the QFC's better ability of avoiding bottleneck shifting. It is because the feedback control is able to adjust the estimated additional passing rates of bottleneck tiers based on real-time output errors, and these additional passed rates will be transferred to other tiers by the queuing network. In other words, control actions of bottleneck tiers are transferred to other tiers too, which is helpful for adjusting the resource of affected tiers more quickly and avoiding bottleneck shifting. The reason of obtaining similar $\bar{D}_i$ is that MRTs exceeding R_i^{sla} have been filtered when calculating $\bar{D}_i$ to avoid the interference of greater MRTs. QFC and JRBE have similar ability of stabilizing MRTs when MRTs greater than the upper limit are ignored.

For System II, the change of L_1's arrival rate leads to the change of arrival rates of L_3, L_4, L_5 then L_7 as shown in Fig. 4b, which makes them become bottleneck tiers. Table 7 shows that $\bar{D}_i$ and $\bar{V}_i$ of QFC are both lower than those of PPM in all tiers. Taking L_4 as an example, QFC obtains much lower $\bar{V}_4$ and $\bar{D}_4$ than PPM as depicted in Fig. 11. The main reason is that PPM increases VMs for each tier L_i in



(a) L_7 (one step is 5 minutes).



(b) L_8 (one step is 5 minutes).

Fig. 10. MRTs of JRBE and QFC in L_7, L_8 of System I.

TABLE 7
Average Upper-Limit Violation Ratios and Deviations of Different Tiers in System II

Tier		L_1	L_2	L_3	L_4	L_5	L_6	L_7	Sum
QFC	$\bar{D}_i(\%)$	12.65	8.72	12.74	13.41	15.06	8.20	18.68	89.46
	$\bar{V}_i(\%)$	10.80	1.05	6.62	6.62	8.71	0.70	17.77	52.27
PPM	$\bar{D}_i(\%)$	15.61	13.91	15.87	22.25	25.13	11.18	19.45	123.40
	$\bar{V}_i(\%)$	16.38	5.57	17.07	23.00	34.15	5.23	18.82	120.22

proportion to the ratios of the request arrival rate in L_i to the total arrival rate in L_0. Due to the ratios are computed on-line considering all different kinds of requests together, which will change when their proportions change leading to inaccurate resource consumption estimations. On the contrary, QFC is able to estimate the impacts of bottleneck tiers on other tiers more accurately based on the transition probability of requests among tiers. Moreover, QFC is able to amend queuing model deviations by feedback control.

5.2.2 Effectiveness of State-Space-Model Based Control

SQF (QFC+SFC) is compared with pure QFC to illustrate the effectiveness of adding SFC in terms of stabilizing response times. Experimental results in Table 6 show that $\bar{D}_i$ and $\bar{V}_i$ of SQF are both lower than those of QFC for all tiers. Meanwhile, Fig. 12 shows that for L_6 of System I, MRTs of SQF are more stable at $R_6^r = 0.26\ s$ and the ADs grow more slowly, which indicate that SQF has a better ability of stabilizing MRTs. The main reason is that SFC in SQF adjusts VMs based on the state-space model when the system is relatively stable. On the contrary, QFC uses queuing models to estimate VM numbers for all intervals including stable times. For queuing models, the increase of the arrival rate will not lead to the increase of VMs until a threshold is reached, which takes some time for QFC to draw the system back to the reference points incurring MRTs' fluctuation.

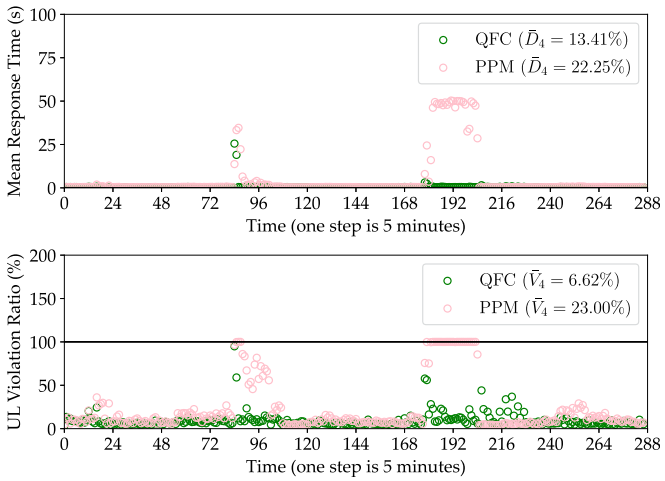


Fig. 11. MRTs and upper-limit (UL) violation ratios of PPM and QFC in L_4 of System II.

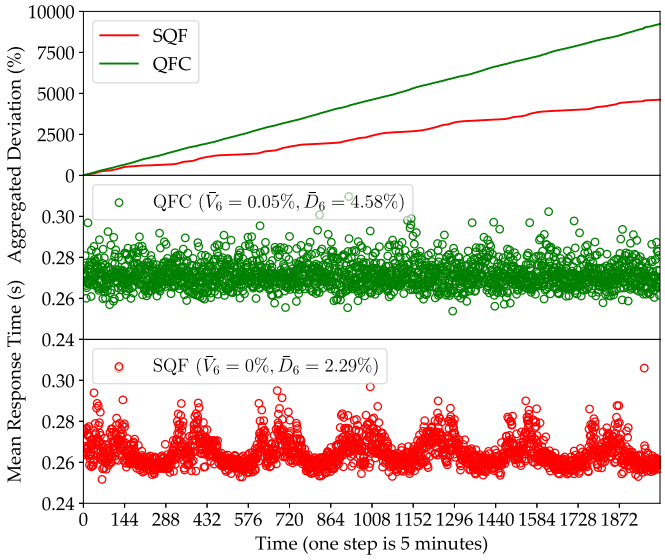


Fig. 12. MRTs and ADs of QFC and SQF in L_6 of System I.

5.2.3 Comparison of End-to-End Response Times

PPE [26] is an existing method which tries to make the end-to-end response times (the sum of sub-requests' response times of all tiers accessed by a business type) smaller than the total business-upper-limit (set to be the sum of all accessed tiers' upper limits) by adjusting the sub-SLA of each tier dynamically without setting a fixed sub-SLA to each tier. SQF is also compared with PPE to evaluate their ability in terms of stabilizing end-to-end response times. Experimental results show that the dynamic SLA distribution of PPE does not work well for meshed Web systems, but SQF can keep end-to-end response times more stable. Take the business type accessing L_1, L_3, L_7 of System II for example, Fig. 13 shows that end-to-end response times' deviations and business-upper-limit violation ratios of PPE are apparently higher than those of SQF. And for the business type (L_2, L_6) of System II, the end-to-end response times' deviation and business-upper-limit violation ratios of PPE are 28.81% and 16.77%, which are both much higher than 12.40% and 4.20% of SQF, respectively. The main reason is that PPE

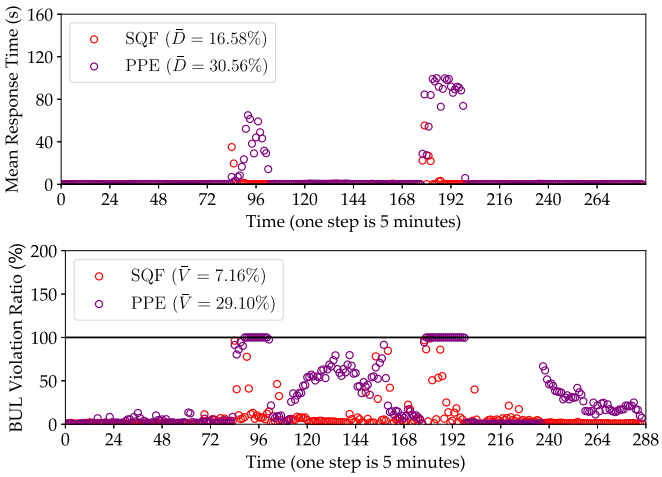


Fig. 13. End-to-end response times' deviations and business-upper-limit (BUL) violation ratios of PPE and SQF for the businesses type (L_1, L_3, L_7) of System II.

models each tier as a M/M/N queue, based on which the performance promise (decreased response time by adding one VM and increased response time by releasing one VM) of each tier is obtained. Because queuing models have deviations, the adjusted number of VMs obtained by multi-tier negotiating is inaccurate. More importantly, PPE does not consider complex interactions among tiers. On the contrary, SQF applies the state-space model to describe the relationships among tiers accurately.

5.2.4 Comparison of Rental Costs

Table 8 displays the VM rental cost C_i of each tier and the total cost $\mathcal{C}$ of compared algorithms. Compared with JRBE, our approach QFC consumes similar costs, but has lower upper-limit violation ratios, although both QFC and JRBE are able to fulfill the V^{sla} constraint for all tiers. However, both of them violate the deviation SLA if the optional D^{sla} constraint is considered by service providers. PPM's costs are lower than those of QFC, but has the average upper-limit violation ratio of $120.22\%/7 = 17.17\%$ much higher than $52.27\%/7 = 7.47\%$ of QFC. The business-upper-limit violation ratio 29.10% of PPE is also much higher than 7.16% of SQF as shown in Fig. 13. The reason is that VMs of PPM and PPE cannot be scaled up or down timely according to workload changes as the pink and purple crosses shown in Fig. 14, respectively, especially for the interval $[197, 228]$ as depicted in the black box. In a word, although the total cost

TABLE 8
VM Rental Costs of Different Tiers

Cost	System I		System II		System I		System II	
	QFC	JRBE	QFC	PPM	SQF	QFC	SQF	PPE
$C_1(\$)$	186.26	187.27	56.95	50.08	282.94	186.26	60.53	54.32
$C_2(\$)$	181.26	181.55	28.95	26.94	242.65	181.26	30.79	26.47
$C_3(\$)$	394.64	387.58	45.35	35.81	476.93	394.64	44.25	39.65
$C_4(\$)$	216.80	216.30	99.76	71.15	230.84	216.80	95.74	68.37
$C_5(\$)$	140.77	140.77	78.65	56.89	151.89	140.77	70.06	61.96
$C_6(\$)$	864.92	853.06	27.40	25.10	958.53	864.92	30.30	23.95
$C_7(\$)$	3401.35	3306.76	68.67	60.64	3551.23	3401.35	84.36	58.39
$C_8(\$)$	3455.02	3306.86	-	-	3607.81	3455.02	-	-
$\mathcal{C}(\$)$	8841.02	8580.15	405.73	326.61	9502.82	8841.02	416.03	333.11

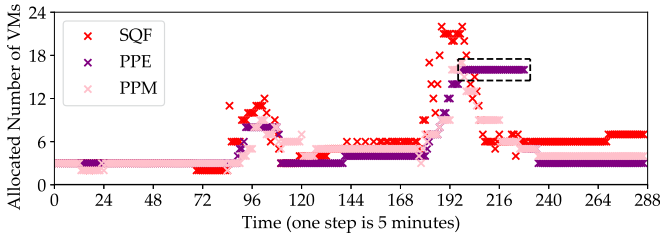


Fig. 14. VM adjustments of PPE, PPM and SQF in L_1 of System II.

of PPM and PPE is lower, both V^{sla} and D^{sla} constraints are greatly violated by them. On the contrary, our approach SQF is able to adjust VMs appropriately as shown by red crosses in Fig. 14. And compared with QFC, SQF applies the state-space model to keep MRTs more stable (fulfilling V^{sla} and D^{sla} constraints simultaneously for most cases) which requires only a little higher VM cost.

6 Conclusions and Future Work

In this paper, a VM provisioning method SQF is proposed to stabilize request response times of Cloud meshed Web systems near the reference times smaller than upper limits with minimum VM renting costs, which takes advantages of the state-space-model and the queuing-network based feedback control. Experimental results show that the state-space-model based feedback control can mitigate fluctuations of MRTs effectively when the system works near reference points with a reasonable high cost. And when the system deviates from reference points, the queuing-network based feedback control is able to improve the efficiency of eliminating bottlenecks compared with existing methods consuming similar costs. Developing nonlinear-state-space-model to improve the performance of the VM provisioning method is the promising future work.

References

[1] G. Huang, C. Luo, K. Wu, Y. Ma, Y. Zhang, and X. Liu, "Software-defined infrastructure for decentralized data lifecycle governance: Principled design and open challenges," in *Proc. 39th IEEE Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 1674–1683.

[2] Z. Cai, D. Liu, Y. Lu, and R. Buyya, "Unequal-interval based loosely coupled control method for auto-caling heterogeneous cloud resources for web applications," *Concurrency Comput. Pract. Exp.*, vol. 32, no. 23, pp. 1–16, 2020.

[3] Z. Cai and R. Buyya, "Inverse queuing model based feedback control for elastic container provisioning of web systems in kubernetes," *IEEE Trans. Comput.*, vol. 71, no. 2, pp. 337–348, Feb. 2022.

[4] D. Liu, Z. Cai, and Y. Lu, "Spot price prediction based dynamic resource scheduling for web applications," in *Proc. 7th Int. Conf. Adv. Cloud Big Data*,. 2019, pp. 78–83.

[5] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal, "Dynamic provisioning of multi-tier Internet applications," in *Proc. 2nd IEEE Int. Conf. Auton. Comput.*, 2005, pp. 217–228.

[6] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood, "Agile dynamic provisioning of multi-tier internet applications," *ACM Trans. Auton. Adaptive Syst.*, vol. 3, no. 1, pp. 1–39, 2008.

[7] Y. Lei, Z. Cai, H. Wu, and R. Buyya, "Cloud resource provisioning and bottleneck eliminating for meshed web systems," in *Proc. 13th IEEE Int. Conf. Cloud Comput.*, 2020, pp. 512–516.

[8] M. Maurer, I. Brandic, and R. Sakellariou, "Enacting SLAs in clouds using rules," in *Proc. 17th Int. Conf. Parallel Process.*, 2011, pp. 455–466.

[9] M. Maurer, I. Brandic, and R. Sakellariou, "Self-adaptive and resource-efficient SLA enactment for cloud computing infrastructures," in *Proc. 5th IEEE Int. Conf. Cloud Comput.*, 2012, pp. 368–375.

[10] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Proc. 4th IEEE Int. Conf. Cloud Comput.*, 2011, pp. 500–507.
[11] V. R. Messias, J. C. Estrella, R. Ehlers, M. J. Santana, R. C. Santana, and S. Reiff-Marganiec, "Combining time series prediction models using genetic algorithm to autoscaling web applications hosted in the cloud infrastructure," *Neural Comput. Appl.*, vol. 27, no. 8, pp. 2383–2406, 2016.
[12] D. Liu, Z. Cai, and X. Li, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Proc. 15th IEEE Int. Symp. Parallel Distrib. Process. Appl. 16th IEEE Int. Conf. Ubiquitous Comput. Commun.*, 2017, pp. 996–1003.
[13] J. Jiang, J. Lu, G. Zhang, and L. Guodong, "Optimal cloud resource auto-scaling for web applications," in *Proc. 3rd IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, 2013, pp. 58–65.
[14] S. Pal and P. K. Pattnaik, "A simulation-based approach to optimize the execution time and minimization of average waiting time using queuing model in cloud computing environment," *Int. J. Elect. Comput. Eng.*, vol. 6, no. 2, pp. 743–750, 2016.
[15] H. Nguyen, T. N. Minh, and N. Thoai, "Tool-driven strategies for resource provisioning of single-tier web applications in clouds," in *Proc. 5th IEEE Int. Conf. Ubiquitous Future Netw.*, 2013, pp. 795–799.
[16] F. Ahmad and T. N. Vijaykumar, "Joint optimization of idle and cooling power in data centers while maintaining response time," in *Proc. 15th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, pp. 243–256.
[17] A. Ali-Eldin, J. Tordsson, and E. Elmroth, "An adaptive hybrid elasticity controller for cloud infrastructures," in *Proc. 13th IEEE Netw. Oper. Manage. Symp.*, 2012, pp. 204–212.
[18] L. Baresi, S. Guinea, A. Leva, and G. Quattrocchi, "A discrete-time feedback controller for containerized cloud applications," in *Proc. 24th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 2016, pp. 217–228.
[19] A. Ashraf, B. Byholm, J. Lehtinen, and I. Porres, "Feedback control algorithms to deploy and scale multiple web applications per virtual machine," in *Proc. 38th IEEE Softw. Eng. Adv. Appl.*, 2012, pp. 431–438.
[20] A. Ashraf, B. Byholm, and I. Porres, "CRAMP: Cost-efficient resource allocation for multiple web applications with proactive scaling," in *Proc. 4th IEEE Int. Conf. Cloud Comput. Technol. Sci.*, 2012, pp. 581–586.
[21] C. Xu, B. Liu, and J. Wei, "Model predictive feedback control for QoS assurance in webservers," *Computer*, vol. 41, no. 3, pp. 66–72, 2008.
[22] R. Singh, U. Sharma, E. Cecchet, and P. Shenoy, "Autonomic mix-aware provisioning for non-stationary data center workloads," in *Proc. 7th Int. Conf. Auton. Comput.*, 2010, pp. 21–30,.
[23] I. Cunha, J. Almeida, V. Almeida, and M. Santos, "Self-adaptive capacity management for multi-tier virtualized environments," in *Proc. 10th IFIP/IEEE Int. Symp. Integr. Netw. Serv. Manage.*, 2007, pp. 129–138.
[24] J. Bi, Z. Zhu, R. Tian, and Q. Wang, "Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center," in *Proc. 3rd IEEE Int. Conf. Cloud Comput.*, 2010, pp. 370–377.
[25] J. Bi, H. Yuan, M. Tie, and W. Tan, "SLA-based optimisation of virtualised resource for multi-tier web applications in cloud data centres," *Enterprise Inf. Syst.*, vol. 9, no. 7–8, pp. 743–767, 2015.
[26] D. Jiang, G. Pierre, and C. H. Chi, "Autonomous resource provisioning for multi-service web applications," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 471–480.
[27] B. Urgaonkar, G. Pacificiy, P. Shenoy, M. Spreitzery, and A. Tantawi, "An analytical model for multi-tier Internet services and its applications," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 291–302, 2005.
[28] P. Padala, K. G. Shin, X. Zhu, M. Uysal, and K. Salem, "Adaptive control of virtualized resources in utility computing environments," *ACM SIGOPS Oper. Syst. Rev.*, vol. 41, no. 3, pp. 289–302, 2007.
[29] Y. Wang, X. Wang, M. Chen, and X. Zhu, "Power-efficient response time guarantees for virtualized enterprise servers," in *Proc. 29th IEEE Real-Time Syst. Symp.*, 2008, pp. 303–312.
[30] A. Ashraf, "Cost-efficient virtual machine provisioning for multi-tier web applications and video transcoding," in *Proc. 13th IEEE/ACM Int. Symp. Cluster*, 2013, pp. 66–69.
[31] A. Ashraf, B. Byholm, and I. Porres, "Prediction-based VM provisioning and admission control for multi-tier web applications," *J. Cloud Comput.*, vol. 5, no. 1, pp. 1–21, 2016.
[32] A. Kamra, V. Misra, and E. M. Nahum, "Yaksha: A self-tuning controller for managing the performance of 3-tiered web sites," in *Proc. 12th IEEE Int. Workshop Qual. Serv.*, 2004, pp. 47–56.
[33] W. Iqbal, M. N. Dailey, and D. Carrera, "Unsupervised learning of dynamic resource provisioning policies for cloud-hosted multitier web applications," *IEEE Syst. J.*, vol. 10, no. 4, pp. 1435–1446, Dec. 2016.
[34] L. Yazdanov and C. Fetzer, "VScaler: Autonomic virtual machine scaling," in *Proc. 6th IEEE Int. Conf. Cloud Comput.*, 2013, pp. 212–219.
[35] L. Yazdanov and C. Fetzer, "Lightweight automatic resource scaling for multi-tier web applications," in *Proc. 7th IEEE Int. Conf. Cloud Comput.*, 2014, pp. 466–473.
[36] J. Rao, X. Bu, C. Z. Xu, and K. Wang, "A distributed self-learning approach for elastic provisioning of virtualized cloud resources," in *Proc. 19th IEEE Int. Symp. Model., Anal. Simul. Comput. Telecommun. Syst.*, 2011, pp. 45–54.
[37] X. Bu, J. Rao, and C. Z. Xu, "A reinforcement learning approach to online web systems auto-configuration," in *Proc. 29th IEEE Int. Conf. Distrib. Comput. Syst.*, 2009, pp. 2–11.
[38] J. Rahman and P. Lama, "Predicting the end-to-end tail latency of containerized microservices in the cloud," in *Proc. 7th IEEE Int. Conf. Cloud Eng.*, 2019, pp. 200–210.
[39] K. Salah and R. Boutaba, "Estimating service response time for elastic cloud applications," in *Proc. 1st IEEE Int. Conf. Cloud Netw.*, 2013, pp. 12–16.
[40] K. Salah, K. Elbadawi, and R. Boutaba, "An analytical model for estimating cloud resources of elastic services," *J. Netw. Syst. Manage.*, vol. 24, no. 2, pp. 285–308, 2016.
[41] J. L. Hellerstein, Y. Diao, and S. Parekh, *Feedback Control of Computing Systems*. Hoboken, NJ, USA: Wiley, 2004.
[42] J. R. Jackson, "Networks of waiting lines," *Operations Res.*, vol. 5, no. 4, pp. 518–521, 1957.
[43] T. Goyal, A. Singh, and A. Agrawal, "Cloudsim: Simulator for cloud computing infrastructure and modeling," *Procedia Eng.*, vol. 38, pp. 3566–3572, 2012.
[44] G. Urdaneta, G. Pierre, and M. van Steen, "Wikipedia workload analysis for decentralized hosting," *Elsevier Comput. Netw.*, vol. 53, no. 11, pp. 1830–1845, 2009.
[45] M. F. Arlitt and C. L. Williamson, "Web server workload characterization: The search for invariants," in *Proc. 19th ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst.*, 1996, pp. 126–137.
[46] N. R. Draper and H. Smith, *Appl. Regression Anal.* Hoboken, NJ, USA: Wiley, 1981.

Yamin Lei received the MSc degree from the School of Computer Science and Engineering, Nanjing University of Science and Technology, China, in 2021. She is currently working toward the PhD degree with Southeast University, Nanjing, China. Her main research interests include resource scheduling of meshed web systems in cloud computing.

Zhicheng Cai (Member, IEEE) received the PhD degree in applied computer science from Southeast University, Nanjing, China, in 2015. He is currently an associate professor with the Nanjing University of Science and Technology, China. His research interests include resource scheduling in cloud, fog and edge computing. He is the author of more than 20 publications in journals such as *IEEE Transactions on Computers*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Automation Science and Engineering* and *Future Generation Computer Systems* and at conferences such as ICSOC, ICPADS, ISPA, ICA3PP, CLOUD, HPCC, SMC, CBD, and CASE.

Xiaoping Li (Senior Member, IEEE) received the PhD degree in applied computer science from the Harbin Institute of Technology, Harbin, China, in 2002. He joined Southeast University, Nanjing, China, in 2005, and is currently a full professor with the School of Computer Science and Engineering. He is the author or co-author more than 100 academic papers, some of which have been published in *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers* and *IEEE Transactions on Services Computing*, etc.

Rajkumar Buyya (Fellow, IEEE) is a redmond barry distinguished professor and director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne, Australia. He has authored more than 625 publications and seven text books. He is one of the highly cited authors in computer science and software engineering worldwide (h-index=153, g-index=324, more than 121,200 citations). Microsoft Academic Search Index ranked him as #1 author in the world (2005-2016) for both field rating and citations evaluations in the area of distributed and parallel computing. He is recognized as a "Web of Science Highly Cited Researcher" during 2016-2021 by Thomson Reuters.