Received December 9, 2018, accepted January 11, 2019, date of publication January 22, 2019, date of current version February 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2894188

Adaptive Multivariable Control for Multiple Resource Allocation of Service-Based Systems in Cloud Computing

SIQIAN GONG, BEIBEI YIN, ZHENG ZHENG, (Senior Member, IEEE), AND KAI-YUAN CAI
School of Automation Science and Electrical Engineering, Beihang University, Beijing 100083, China

Corresponding authors: Siqian Gong (gongsiqian@buaa.edu.cn) and Beibei Yin (yinbeibei@buaa.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61772055 and Grant 61872169.

ABSTRACT Service-based systems resource allocation in cloud computing is a key method of meeting service requests because service request workloads and resource demands change over time. When coping with dynamic fluctuating service requests and resource demands, adaptive resource allocation to ensure the quality of service (QoS) with the lowest resource consumption becomes challenging. In cloud computing, services share the same resource pool and compete for critical resources, such as CPU and memory resources. Because services need arbitrary resource combinations, focusing on a single resource may lead to excessive or deficient resource allocations or even service request failures. Due to the shared nature of cloud computing, QoS may be impacted by interference with co-hosted services. In this paper, we propose an adaptive control approach for resource allocation that adaptively reacts to dynamic request workloads and resource demands. The multivariable control is adopted to allocate multiple resources for multiple services according to the dynamic fluctuating requests and considers the interference between co-hosted services, thereby ensuring QoS even if the resource pool is insufficient. The comparative experiments show that the proposed approach can meet service requests and can improve resource utilization regardless of whether the resource pool is sufficient.

INDEX TERMS Adaptive resource allocation, cloud computing, multivariable control, QoS.

I. INTRODUCTION

The emergence of cloud computing has had a large impact on the Information Technology (IT) industry [1], and many enterprises are competing to provide more powerful, reliable and cost-efficient cloud services including large companies such as Google, Amazon, Microsoft and Alibaba. In addition, IT enterprises are striving to reshape their business services to gain more benefits from cloud computing [2]. In a cloud computing environment, service providers manage cloud resources according to an on-demand pricing scheme, and the service providers must ensure their own profits while providing good QoS and maximum user satisfaction. Therefore, resource allocation plays an important role in cloud computing [3] and affects the QoS, the performance of the whole system and SLA (Service Level Agreement), which indicates the level of user satisfaction [4].

User requirements may change over time, and at different times, many or a few users may try to concurrently access the server. For example, Alibaba has exceeded its single-day record as sales cross 25 billion dollars, and the amount of users accessing the servers is 43 million per second at peak times [5]. The provider must supply sufficient resources to ensure QoS; otherwise, unexpected loads may cause poor QoS that violates SLA. Amazon reported a loss of 245 million dollars because of an increase in response time of 100 ms [6]. To avoid such situations and ensure QoS, an effective dynamic resource allocation scheme is required.

Reactive approaches [7]–[15] solve this challenge mainly by increasing or decreasing resources according to predefined thresholds. However, the metrics and parameters must be specified, and upper and lower thresholds are difficult to obtain. Proactive approaches [16]–[24] predict future workload variations before their occurrence to react to the dynamic resource allocation. For example, time-series [17], [18], reinforcement learning [16], [19], [20], and queuing theory [21], [22] cope with the challenge by predicting and adapting to fluctuating workloads [25]. However, these existing approaches have shortcomings. Time-series relies too

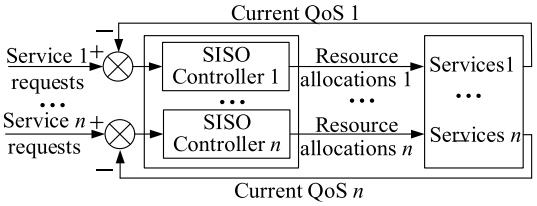

FIGURE 1. Decomposed MIMO control system structure.

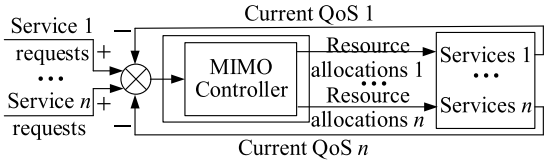

FIGURE 2. Coordinated MIMO control system structure.

heavily on historical data, reinforcement learning consumes considerable time for learning, and queuing theory must rebuild the model when the workload changes. To overcome these limitations, feedback mechanisms are increasingly applied as they enable complex computing systems to cope with fluctuating workloads or operating conditions [26].

Control theory provides principled feedback loops to manage unpredictable changes, uncertainties and disturbances in systems. In [27], single-input and single-output (SISO) control is used to allocate a single type of resource to services. However, processing service requests requires different types of resource combinations (e.g., CPU and memory). For example, if two users request different services with different resource demands but have similar SLAs, the provider may allocate insufficient memory and excessive CPU resources to the services that need considerable memory but limited CPU resources, thereby causing an SLA violation. However, most studies [28]–[33] have not fully addressed such characteristics, and users are usually only considered based on their SLAs. To cope with the multiple resource combination issue, multivariable control [34] has been applied for resource allocation problems.

Multivariable control provides a multi-input and multi-output mechanism to generate combinatorial resource allocation. As shown in Fig. 1, several studies [35]–[39] have allocated multiple resources to types of services by decomposing the multi-input and multi-output (MIMO) control system to multiple simple, independent SISO control systems. However, the QoS may be impacted by interactions and competition with co-hosted services, and ignoring these interferences may lead to SLA violations. The interactions cannot be precisely decoupled by decomposing the MIMO system to multiple SISO systems.

To resolve such problems, we use the coordinated MIMO control system [40] shown in Fig. 2. This system centralizes all inputs and outputs, and each output is impacted by all inputs. The system enables each type of resource allocation to be constrained by other co-hosted services, and thus coordinated resource allocation occurs according to the workloads of the services. This system considers the interactions between services and ensures QoS by coordinating the resource allocation of services.

In this paper, we propose an adaptive multivariable control resource allocation approach for SBS in cloud computing that can allocate multiple types of resources to multiple virtual machines (VMs), thus improving resource utilization and ensuring QoS. The main contributions of this paper are as follows:

(1) We propose an adaptive multivariable control strategy for resource allocation in cloud computing systems. Our proposed approach provides a powerful mechanism to manage unpredictable changes and uncertainties, performs combinatorial resource allocation according to the various characteristics of different services, and considers interference between co-hosted services.

(2) We compare the reactive approach, proactive approach, SISO control approach and decomposed MIMO control approach with our coordinated MIMO control approach under sufficient and insufficient resource pool situations. The comparative experimental results show that our approach allows reduced operating costs and increases resource utilization while simultaneously ensuring QoS, especially under the insufficient resource pool situation.

(3) The proposed control theoretic approach facilitates management of the challenges of resource allocation in cloud computing systems, saves operating costs and increases resource utilization while simultaneously ensuring QoS by adaptively reacting to the open environment full of interference and addressing unpredictable resource demands in real time.

The rest of the paper is organized as follows: Section II discusses novel dynamic resource allocation. An overview of our approach is described in Section III. We introduce the details of our resource allocation controller in Section IV. The experiments are shown in Section V. We compare our approach with related research efforts and analyze the results of the comparison in Section VI. Future work is introduced in Section VII, and conclusions are provided in Section VIII.

II. RELATED WORKS

Many studies have sought approaches for resource allocation in dynamic environments such as clouds. These approaches are divided into two categories: reactive approaches and proactive approaches. Reactive approaches increase and decrease resources based on whether predefined thresholds are reached. Proactive approaches anticipate the amount of resources to react to future workload fluctuations. In the following section, we discuss previous research in resource allocation. A comparison of related efforts is provided in Table 1. We classify these efforts in terms of approaches, underlying models, metrics and issues related to resource allocation.

TABLE 1. Comparison of related works.

Ref.	Approach type	Control theoretic	Control structure	Prediction	Multiple resource allocation	Resource combination	Interference consideration	Solution
7, 8	Reactive	-	-	✓	✓	-	-	VM migration
9	Reactive	-	-	-	-	✓	-	Threshold-based migration
10, 11	Reactive	-	-	-	-	✓	-	VM assignment
12	Reactive	-	-	-	-	✓	-	Bidding language
13	Reactive	-	-	-	-	-	✓	VM instances allocation
14	Reactive	-	-	-	-	-	-	NP-hard optimization
15	Reactive	-	-	-	-	-	-	Linear programming
16	Proactive	-	-	✓	-	✓	-	Learning automation
17, 18	Proactive	-	-	✓	-	-	-	Time series
19, 20	Proactive	-	-	-	✓	-	✓	Reinforcement learning
21, 22	Proactive	-	-	✓	-	-	-	Queuing theory
23, 24	Proactive	-	-	-	-	✓	-	Access control
28, 31, 32	Proactive	✓	SISO	-	-	-	-	Feedback control, Optimal control, PID control
30	Proactive	✓	SISO	✓	-	-	-	Lyapunov technique
27, 33	Proactive	✓	SISO	-	✓	-	-	Fuzzy control
29	Proactive	✓	SISO	✓	✓	-	-	Feedback control
35	Proactive	✓	MIMO	✓	-	-	-	Feedback control
36, 37, 38	Proactive	✓	MIMO	-	✓	-	-	LPV control, Stochastic model
39	Proactive	✓	MIMO	✓	✓	✓	-	Fuzzy control
Ours	Proactive	✓	MIMO	✓	✓	✓	✓	GPC control

We design eight dimensions for the classification as shown in Table 1. The first dimension is related to the approach types: reactive and proactive. The second dimension concerns whether the proposed approach in a reference is based on control theory. The third dimension is related to the structure of the controller employed if the proposed approach is based on control theory. If a controller is single input and single output, then the reference is marked as "SISO"; otherwise, if a controller is multi-input and multi-output, then the reference is marked as "MIMO". The fourth dimension concerns whether an approach enables prediction of the requested workload or future resource demand; if it enables prediction of future demands, it is marked as "✓", otherwise, it is marked as "−". The fifth dimension concerns whether an approach allows multiple types of resource allocation. If it only allocates a single type of resource, then it is marked as "−"; otherwise, it is marked as "✓". The sixth dimension concerns whether an approach allows combinatorial resource allocation, as processing service requests requires different types of resource combinations. If an approach considers this issue, then it is marked as "✓"; otherwise, it is marked as "−". The seventh dimension concerns resource interference, which indicates whether the proposed approach considers the interference between services. Because services in cloud computing share the public resource pool, QoS can be impacted by possible interactions and competition with co-hosted services. Ignoring this interference may lead to failed service requests. To avoid SLA violation, inferences should be considered. If interference is considered, then the approach is marked as "✓"; otherwise, it is marked as "−". The eighth dimension involves the approaches and technologies used to achieve resource allocation. For example, some studies use VM migration to minimize the number of physical machines (PMs), and some studies increase or decrease the computing resources of VMs. Finally, each prior effort is classified based on the above eight dimensions. The classification results are shown in Table 1.

As shown in Table 1, due to the dynamic characteristics of resource allocation in cloud computing, several studies [7]–[15] tackle this challenge through reactive approaches, which increase or decrease resources based on performance metrics and predefined thresholds. However, the metrics and parameters must be specified, and the upper and lower thresholds are difficult to obtain. In addition, as shown in Table 1, most of the reactive approaches do not have prediction mechanisms. Therefore, they may not able to react in time to unpredictable bursts in workloads.

Proactive approaches [16]–[24] anticipate future workload fluctuations using time-series analysis [17], [18], reinforcement learning [16], [19], [20], queuing theory [21], [22], access control [23], [24] and control theory techniques. They predict the behavior of the system and deter-mine the reconfiguration actions accordingly. However, each proactive approach has limitations. For example, the time-series approach in [17] and [18] is highly dependent on historical data; some approaches using machine learning [16], [19], [20] require considerable time for learning [48]; the approaches using queuing theory in [21] and [22] require recalculation of the model when the workload changes; and some approaches adopting access control [23], [24] must

process a large amount of data. Therefore, such approaches usually require considerable time and cannot sufficiently react when fluctuations occur frequently.

Many researchers [27]–[39] use control theoretic approaches for automatic resource allocation. Control theory provides an adaptive means of designing feedback loops to manage fluctuations in external and internal environments. Some studies [27]–[33] have used SISO control systems for resource allocation. However, processing different services requires different types of resources, such as CPU and memory resources. For example, two users who request different services with different resources demands but have similar SLAs can have the same priority for a provider, which may lead the provider to allocate insufficient memory and excessive CPU resources to the services that require considerable memory but limited CPU resources, thereby causing low QoS or even violating SLA. These SISO approaches perform single-resource allocation to achieve a single goal, and ignoring the resource combination may lead to different types of resource allocation mismatches.

To manage the mismatch problem mentioned above, several studies [35]–[39] have designed multivariable controllers to allocate multiple resources, such as CPU and memory resources, to meet QoS requirements. These decomposed multivariable control approaches decompose one MIMO control system into multiple independent and simple SISO control systems to allocate multiple types of resources to multiple services. However, due to the shared nature of cloud computing, QoS may be impacted by interactions and competition with co-hosted services; therefore, inaccurate decomposition of interference may cause low QoS and even SLA violation.

From Table 1, we find that coordinated MIMO control systems combine prediction, multiple resource allocation and interference consideration into one controller; therefore, we adopt coordinated multivariable control to achieve resource allocation that resolves the challenges mentioned above.

III. APPROACH OVERVIEW

An overview of the proposed approach is shown in Fig. 3. In this paper, we focus on the most important resources that affect QoS: CPU and memory resource allocations. We deploy multiple services on multiple VMs (VM1, VM2 and VM3) that share the same resource pool. The services on the VMs have different demands of multiple resources. For example, the services deployed on VM1 are CPU-intensive and consume more CPU resources than memory, the services deployed on VM2 are memory-intensive and consume more memory than CPU resources, and the services on VM3 are normal services that consume nearly the same amount of CPU and memory resources. To clearly express this scheme, we provide definitions below.

- $W(k) = [w_1(k), w_2(k), w_3(k)]^T$: $W(k)$ is the current workload at moment k and contains the numbers of requests per second of each VM. $w_1(k)$ (requests/s) is the number of requests per second of VM1, $w_2(k)$ (requests/s) is the number of requests per second of VM2, and $w_3(k)$ (requests/s) is the number of requests per second of VM3.
- $Q(k) = [q_1(k), q_2(k), q_3(k)]^T$: $Q(k)$ is the current QoS at moment k. $q_1(k)$ is the number of service responses per second of VM1, $q_2(k)$ is the number of service responses per second of VM2, and $q_3(k)$ is the number of service responses per second of VM3.
- $R(k) = [CPU_1(k), MEM_1(k), CPU_2(k), MEM_2(k), CPU_3(k), MEM_3(k)]^T$: $R(k)$ is the current resource allocation at moment k and contains the CPU and memory resources of each VM. $CPU_1(k)$ and $MEM_1(k)$ are the CPU and memory resource allocations of VM1, respectively. Analogously, $CPU_2(k)$ and $MEM_2(k)$ and $CPU_3(k)$ and $MEM_3(k)$ are the CPU and memory resource allocations of VM2 and VM3, respectively.

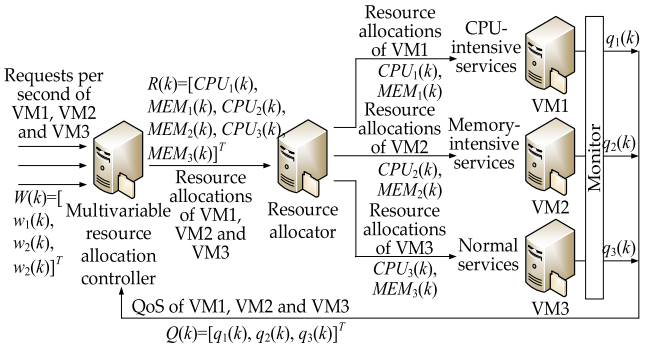

FIGURE 3. Overview of resource allocation.

As shown in Fig. 3, the requests of each VM $W(k)$ simultaneously arrive at the resource allocation controller. The controller calculates the CPU and memory resource allocation $R(k)$ to each VM. Then, the resource allocator executes the calculated multiple resource allocations $R(k)$ to the VMs. In addition, the current QoS $Q(k)$ is observed by the monitor and fed back to the resource allocation controller. According to this feedback $Q(k)$ and workload $W(k)$, the controller calculates the resource allocations $R(k)$ for the next interval.

IV. RESOURCE ALLOCATION CONTROLLER

The cloud computing environment is dynamic and complex. To achieve economic benefits and ensure QoS, a controller with an adaptive model and a simple algorithm that enables prediction of future demands are required. Unlike traditional controls in [29]–[32], GPC (Generalized Prediction Control) [41] is a kind of control algorithm combined with feedback correction that recedes horizon optimization and prediction to meet the real-time control requirements of cloud computing. In addition, since the parameters of the control model are unknown and setting an unreasonable parameter may cause oscillations, an adaptive mechanism is proposed to estimate the parameters in real time.

An overview of the resource allocation controller is shown in Fig. 4. We adopt a coordinated MIMO control structure for the controller consisting of four parts: a transition process,

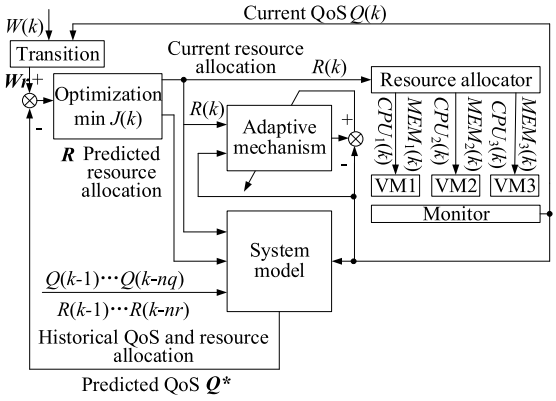

FIGURE 4. Multivariable resource allocation controller.

optimization, the system model and the adaptive mechanism. The transition process facilitates the transition from the actual workload to the expected workload. The optimization calculates and predicts the CPU and memory resource allocations for the three VMs by minimizing resource consumption. The system model then predicts the future QoS based on the historical and current resource allocations and QoS. Concurrently, the adaptive mechanism corrects the parameters of the system model according to the current resource allocation and QoS. All these components consist of a closed control loop to calculate the resource allocation online.

In Fig. 4, $\boldsymbol{W_r}$ is the expected workload containing the expected workload of each VM; $\boldsymbol{R}$ is the predicted resource allocation containing the predicted CPU and memory resource allocations of these VMs; $\boldsymbol{Q_o}$ is the observed QoS based on moment k; and $\boldsymbol{Q}^*$ is the predicted QoS based on moment k. The specific expressions are shown in (1).

$$\begin{cases} \boldsymbol{W_r} = [W_r(k+1)\cdots W_r(k+N)]^T \\ W_r(k+j) = [w_{r1}(k+j), w_{r2}(k+j), w_{r3}(k+j)]^T \\ \qquad j = 1, 2\cdots N \\ \boldsymbol{R} = [R(k+d)\cdots R(k+N-1)]^T \\ R(k+j) = [CPU_1(k+j), MEM_1(k+j), CPU_2(k+j), \\ \qquad MEM_2(k+j), CPU_3(k+j), MEM_3(k+j)]^T \\ \qquad j = 1, 2\cdots N-1 \end{cases} \tag{1}$$

where N is the prediction length; $w_{r1}(k+j)$, $w_{r2}(k+j)$, and $w_{r3}(k+j)$ are the expected workloads of the three VMs, respectively, which indicate the number of requests per second for each VM; $CPU_1(k+j)$, $MEM_1(k+j)$, $CPU_2(k+j)$, $MEM_2(k+j)$, $CPU_3(k+j)$, and $MEM_3(k+j)$ are the predicted CPU and memory resource allocations to the three VMs, respectively; and $q_1(k+j|k)$, $q_2(k+j|k)$, and $q_3(k+j|k)$ are the predicted QoS of the three VMs based on k, respectively.

A. SYSTEM MODEL

Considering the non-linear and time-varying characteristics of cloud computing, the specific relationship between QoS and resource allocation is difficult to capture. We describe the QoS and resource allocation in (2).

$$Q(k) = f\left(Q(k-1)\cdots Q(k-n_q), R(k-d)\cdots R(k-d-n_r)\right) \tag{2}$$

where $f(*)$ describes the relationship between QoS and resource allocation and d is the latency of controller. As latency exists in real-time systems, especially for cloud computing systems, ignoring the latency may lead to model mismatch of even an invalid control system, thereby affecting QoS [42]. Therefore, it is necessary to consider latency in resource management. We consider the latency in the real-time system by compensating for the controller with a latency d, and the setting of d is discussed in the next section.

In the non-linear complex cloud computing environment, we describe the relationship between QoS and resource allocation by using an auto-regressive integrated moving average (ARIMA) model [36], and the adaptive mechanism estimates the time-varying parameters in every control interval. In ARMA form, the system model is described in (3).

$$Q(k) = \sum_{i=1}^{n_q} A_i Q(k-1) + \sum_{j=0}^{n_r} B_j R(k-d-j) \tag{3}$$

where $A_i \in R^{3\times3}$, $B_j \in R^{3\times6}$ are the parameters of the system model.

B. TRANSITION PROCESS

The goal of the controller is to derive the resource allocation by minimizing resource consumption and the gap between QoS and the workload. Therefore, to transition from the expected workloads to actual workloads smoothly, we use the transition process in (4) [7].

$$W_r(k+j) = \alpha W_r(k+j-1) + (I-\alpha)W(k) \tag{4}$$

where $j = 1, 2\ldots N$ and $\alpha \in R^{3\times3} = diag\{\alpha_1, \alpha_2, \alpha_3\}$ is the transition coefficient, which determines the transition rate from the expected workloads to the actual workloads.

C. OPTIMIZATION

The receding horizon optimization is updated online rather than using a fixed global optimization. We predict QoS $Q^*(k+j|k)$, $(j = 1, 2\ldots N)$ based on moment k, and the resource allocation $R(k)$ is calculated according to the prediction. Therefore, the optimal resource allocation is updated by the predictions and regulations online.

1) PREDICTION

Due to the existence of network transmission delay and controller execution delay, it is difficult to precisely determine the latency in the system [43]. Prediction control is used to cope with the latency by predicting the future QoS [44]. It predicts QoS in the future by minimizing the error between the observed QoS $Q_o(k+j)$ and the predicted QoS $Q^*(k+j)$, and the performance index is shown in (5). We adopt recursive multi-steps prediction to obtain the observed QoS $Q_o(k+j)$

based on the current QoS $Q(k)$. Therefore, the prediction and optimization are cycled online to compensate for the latency in real time.

$$J_{\min} = E\left\{\left[Q^*(k+j|k) - Q_o(k+j)\right]^2\right\} \tag{5}$$

To compensate for the latency, we define the observed prediction vector $\boldsymbol{Q_o}$ and prediction vector $\boldsymbol{Q^*}$, the details of which are as follows:

$$\begin{cases} \boldsymbol{Q_o} = [Q_o(k+d), Q_o(k+d+1) \cdots Q_o(k+N)]^T \\ \boldsymbol{Q^*} = \left[Q^*(k+d), Q^*(k+d+1) \cdots Q^*(k+N)\right]^T \end{cases} \tag{6}$$

where $Q_o(k+d) \ldots Q_o(k+N)$ are iterated based on $Q(k)$ as follows:

$$\begin{cases} Q_o(k+1) = f(Q(k) \cdots Q(k-n_q+1), \\ \qquad R(k-d+1) \cdots R(k-d-n_r+1)) \\ \vdots \\ Q_o(k+d) = f(Q_o(k+d-1) \cdots Q_o(k-n_q+d), \\ \qquad R(k) \cdots R(k-n_r)) \\ \vdots \\ Q_o(k+N) = f\left(Q_o(k+N-1) \cdots Q(k-n_q+N), \right. \\ \qquad \left. R(k-d+N) \cdots R(k-d-n_r+N)\right) \end{cases} \tag{7}$$

where the iteration is based on (3). Therefore, we can derive the observed prediction vector $\boldsymbol{Q_o} = [Q_o(k+d) \ldots \mathrm{Qo}Q_o(k+N)]^T$. Then, the prediction performance index expressed in matrix form is as follows:

$$J_{\min} = E\left\{(\boldsymbol{Q^*} - \boldsymbol{Q_o})^T (\boldsymbol{Q^*} - \boldsymbol{Q_o})\right\} \tag{8}$$

According to (3) and (8), we can derive the prediction $\boldsymbol{Q^*}$ by minimizing the deviation between prediction $\boldsymbol{Q^*}$ and observation $\boldsymbol{Q_o}$. $\boldsymbol{Q^*}$ is calculated in (9).

$$\boldsymbol{Q^*} = \boldsymbol{Q_o} + \boldsymbol{C}\,\boldsymbol{\Delta R} \tag{9}$$

where $\boldsymbol{C}$ is the control coefficient matrix. The specifics of $\boldsymbol{C}$ are shown in (10).

$$\boldsymbol{C} = \begin{bmatrix} C_1 & 0 & \cdots & 0 \\ C_2 & C_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_{N-d+1} & C_{N-d} & \cdots & C_1 \end{bmatrix} \tag{10}$$

where the element C_j of $\boldsymbol{C}$ is calculated by the iteration formulae (7) and (3). The calculation is described in (11).

$$\begin{cases} C_1 = B_0 \\ C_j = B_{j-1} + \sum_{i=1}^{j_1} A_i C_{j-i}, j_1 = \min\left\{j-1, n_q\right\}, \\ \quad j-1 > n_r, \quad B_{j-1} = 0 \end{cases} \tag{11}$$

2) OPTIMAL CONTROL

The optimal resource allocation is determined by minimizing the resource allocation consumption and the gap between the expected workloads $\boldsymbol{W_r}$ and the predicted QoS $\boldsymbol{Q^*}$. The performance index is described in (12).

$$J_{\min} = E\left\{\sum_{j=1}^{N}\left[Q^*(k+j) - W_r(k+j)\right]^2 + \sum_{j=1}^{N}\left[\Delta R(k+j-1)\right]^2\right\} \tag{12}$$

where $R(k) = [\Delta CPU_1(k), \Delta MEM_1(k), \Delta CPU_2(k), \Delta MEM_2(k), \Delta CPU_3(k), \Delta MEM_3(k)]^T$ is the incremental control. Formula (10) indicates that the resource allocation is optimized in every control interval and is not a fixed global optimized allocation determined offline. According to (1), the performance index expressed in matrix form is shown in (13).

$$J_{\min} = E\left\{(\boldsymbol{Q^*} - \boldsymbol{W_r})^T (\boldsymbol{Q^*} - \boldsymbol{W_r}) + \boldsymbol{\Delta R}^T \boldsymbol{\Delta R}\right\} \tag{13}$$

where $\Delta \boldsymbol{R}$ is the increment resource allocation, and its specific expression is shown in (14).

$$\begin{cases} \boldsymbol{\Delta R} = [\Delta R(k), \Delta R(k+1) \cdots \Delta R(k+N-1)]^T \\ \Delta R(k+j) = \left[\Delta CPU_1(k+j), \right. \\ \qquad \Delta MEM_1(k+j), \quad \Delta CPU_2(k+j), \\ \left. \Delta MEM_2(k+j), \quad \Delta CPU_3(k+j), \quad \Delta MEM_3(k+j)\right]^T \end{cases} \tag{14}$$

To derive the minimized J_{min} in (13), we let $\partial J / \partial \Delta \boldsymbol{R} = 0$. The minimizing process is shown in (15), which can be derived according to (13) and (9).

$$\begin{aligned} \frac{\partial J}{\partial \boldsymbol{\Delta R}} &= (\boldsymbol{Q^*} - \boldsymbol{W_r})^T \frac{\partial (\boldsymbol{Q^*} - \boldsymbol{W_r})}{\partial \boldsymbol{\Delta R}} + \frac{\partial (\boldsymbol{Q^*} - \boldsymbol{W_r})^T}{\partial \boldsymbol{\Delta R}} \\ &\quad \times (\boldsymbol{Q^*} - \boldsymbol{W_r}) + \boldsymbol{\Delta R}^T + \boldsymbol{\Delta R} \\ &= (\boldsymbol{Q_o} + \boldsymbol{C}\,\boldsymbol{\Delta R} - \boldsymbol{W_r})^T \frac{\partial (\boldsymbol{Q_o} + \boldsymbol{C}\,\boldsymbol{\Delta R} - \boldsymbol{W_r})}{\partial \boldsymbol{\Delta R}} \\ &\quad + \frac{\partial (\boldsymbol{Q_o} + \boldsymbol{C}\,\boldsymbol{\Delta R} - \boldsymbol{W_r})^T}{\partial \boldsymbol{\Delta R}} (\boldsymbol{Q_o} \\ &\quad + \boldsymbol{C}\,\boldsymbol{\Delta R} - \boldsymbol{W_r}) + \boldsymbol{\Delta Q}^T + \boldsymbol{\Delta Q} \\ &= \left[(\boldsymbol{Q_o} + \boldsymbol{C}\,\boldsymbol{\Delta R} - \boldsymbol{W_r})^T \boldsymbol{C}\boldsymbol{C}\,\boldsymbol{\Delta R}^T\right] \\ &\quad + \left[\boldsymbol{C}^T (\boldsymbol{Y}\boldsymbol{Q_o} + \boldsymbol{C}\,\boldsymbol{\Delta R} - \boldsymbol{W_r}) + \boldsymbol{\Delta R}\right] \\ &= 0 \end{aligned} \tag{15}$$

Then, the increment control $\Delta \boldsymbol{R}$ can be derived, and the expression is shown in (16).

$$\boldsymbol{\Delta R} = (\boldsymbol{C}^T \boldsymbol{C} + \boldsymbol{I})^{-1} \boldsymbol{C}^T (\boldsymbol{W_r} - \boldsymbol{Q_o}) \tag{16}$$

where $\boldsymbol{C}$ is set according to (10) and (11), and $R(k)$ is the first element of $\Delta \boldsymbol{R}$. Then, we can derive the resource allocation $R(k)$ in (17).

$$R(k) = R(k-1) + \Delta R(k)$$

$$= R(k-1) + [\underbrace{1\cdots1}_{6}, 0\cdots0]\boldsymbol{\Delta R}$$
$$= R(k-1) + [\underbrace{1\cdots1}_{6}, 0\cdots0]$$
$$\times(\boldsymbol{C}^T\boldsymbol{C} + \boldsymbol{I})^{-1}\boldsymbol{C}^T(\boldsymbol{W_r} - \boldsymbol{Q_o}) \quad (17)$$

where $\boldsymbol{W_r}$ and $\boldsymbol{Q_o}$ are derived by (4) and (3), respectively, and $R(k-1)$ is observed by the monitor.

From the above functions, we derive the k moment CPU and memory resource allocations $R(k)$ for the three VMs. Then, the monitor feeds back QoS $Q(k)$ to the controller for the next control cycle.

D. ADAPTIVE MECHANISM

Because the parameters in (3) are unknown, an adaptive mechanism is needed to estimate the parameters in real time. We use FFRLS (Forgetting Factor Recursive Least Square) [45] to estimate the parameters of the controller. The goal of the adaptive mechanism is to minimize the gap between the estimated and actual QoS. As the scale of the data increases, the capacity of correction becomes worse. When changes in the system occur, the adaptive mechanism cannot react to the changes, leading to failure of real-time parameter estimation. To overcome this deficiency, the forgetting factor is adopted to forget the historical data, which introduces a weighting coefficient to the data. The coefficient of the latest data is 1, and older data correspond to a coefficient closer to 0; thus, the historical data are forgotten by this forgetting factor. The performance index of the adaptive mechanism is shown in (18).

$$J_{\min} = \sum_{j=1}^{k} \lambda^{k-j}\left[Q(j) - \hat{Q}(j)\right]^2 \quad (18)$$

where $\hat{Q}(j)$ is the estimated QoS of service responses per second and λ is the forgetting factor. The coefficient weight of the historical data at the $(k-1)$, $(k-2)\ldots1$ moment is $1, 2\ldots k-1$. Therefore, the current data are maximally weighted, and the historical data are exponentially forgotten. For clarification, we let $\hat{Q}(k) = \boldsymbol{\varphi}^T(k)\hat{\boldsymbol{\theta}}(k)$, and the specific expressions of $\varphi^T(k)$ and $\hat{\boldsymbol{\theta}}(k)$ are shown in (19).

$$\begin{cases} \hat{Q}(k) = \boldsymbol{\varphi}^T(k)\hat{\boldsymbol{\theta}}(k) \\ \boldsymbol{\varphi}(k) = [Q(k-1)\cdots Q(k-n_q), \\ \qquad R(k-d)\cdots R(k-d-n_r)]^T \\ \hat{\boldsymbol{\theta}}(k) = \left[\hat{A}_1(k)\cdots\hat{A}_{nq}(k), \hat{B}_0(k)\cdots\hat{B}_{d+n_r}(k)\right]^T \end{cases} \quad (19)$$

where $\varphi(k)$ is the initial configuration and $\hat{\boldsymbol{\theta}}(k)$ is the estimated parameters. In (18), the forgetting factor λ will weaken the impact of historical data with data updates. Then, we can derive the estimated parameters of the controller by minimizing the performance index in (18), and the estimated parameters $\hat{\boldsymbol{\theta}}(k)$ is shown in (20).

$$\begin{cases} \boldsymbol{\theta}(k) = \boldsymbol{\theta}(k-1) + K(k)\left[Q(k) - \boldsymbol{\varphi}^T(k)\boldsymbol{\theta}(k-1)\right] \\ K(k) = \dfrac{P(k-1)\boldsymbol{\varphi}(k)}{\lambda + \boldsymbol{\varphi}^T(k)P(k-1)\boldsymbol{\varphi}(k)} \\ P(k) = \dfrac{1}{\lambda}\left[\boldsymbol{I} - K(k)\boldsymbol{\varphi}^T(k)\right]P(k-1) \end{cases} \quad (20)$$

For all of the above processes, the resource allocation is summarized in Table 2.

TABLE 2. Resource allocation control algorithm.

Resource allocation control algorithm
1. Input initial $\boldsymbol{\varphi}$, N, λ, d and α
2. Monitor current QoS $Q(k)$=[$q_1(k)$, $q_2(k)$, $q_3(k)$]^T.
3. Adaptive mechanism estimates parameters of the controller online.
4. Calculate control matrix $\boldsymbol{C}$ using (10) and (11).
5. Structure $\boldsymbol{Q_o}$ and $\boldsymbol{W_r}$ according to (3) and (4).
6. Calculate current resource allocation according to (12).
7. Return to 1 (k→k+1).

V. EXPERIMENTAL SETUPS

We build two types of experimental situations to verify our resource allocation approach: insufficient and sufficient resource pools. We select four approaches from the 30 reviewed papers according to the approach type shown in Table 1: the reactive approach in [15], the proactive approach in [19], the SISO control approach in [29] and the decomposed MIMO control approach in [39]. In [15], the reactive approach allocates the system resources to the VMs using linear programming; hereafter, this approach will be abbreviated as "LP". In [19], the proactive approach proposes a distributed learning mechanism using reinforcement learning that facilitates adaptive VM resource allocation; hereafter, it will be abbreviated as "RL". In [29], the SISO control approach designs multiple SISO controllers for each VM, which allocates CPU or memory resources according to the feedback information; hereafter, we call this approach "SISO" for short. In [39], the decomposed MIMO control approach also adopts MIMO control to allocate multiple types of resources to the VMs, but it decomposes the MIMO controller to multiple simple SISO controllers; thus, we call this decomposed MIMO control approach "de-MIMO" hereafter for short. Different from the de-MIMO control in [39], we use only one coordinated MIMO control to allocate multiple types of resources to multiple VMs, and hereafter we call our approach "co-MIMO" for short.

In this experiment, we allocate the most important resources, CPU and memory resources, to the VMs. To extend our approach to a large-scale cloud, we classify the services into three types, CPU-intensive services, memory-intensive services and normal services [46], separately for the three VMs. Services for which the ratio of CPU and memory demands is greater than 1:2 are termed CPU-intensive services; services for which the ratio of CPU and memory demands is less than 1:4 are termed memory-intensive ser-

vices, and services for which the ratio of CPU and memory demands is between 1:4 and 1:2 are termed normal services.

In this experiment, we use "Route planning", "Image analysis", and "Regional location" as the three services to evaluate the proposed controller. "Route planning" is considered a CPU-intensive service, "Image analysis" is considered a memory-intensive service, and "Regional location" is considered a normal service. Therefore, VM1 is run with the service "Route planning", VM2 is run with "Image analysis", and VM3 is run with "Regional location". The QoS requirements indicate that the response time of these three types of services must be less than 0.8 s.

The resource allocator and controller run on two separate physical machines, and both are developed by Python 3.4. In our experiments, CPU and memory resource allocation is required for each VM. OpenStack [47] is an open-source platform with well-defined APIs, and therefore we provide an interface for each VM based on OpenStack Icehouse.

A. INITIAL PARAMETER SETTINGS

To determine the initial parameters mentioned in Table 2, we will discuss the impact of the parameters on our coordinated MIMO performance by analyzing the simulation results.

- φ: φ is the initial CPU and memory resource allocation of the VMs; the initial configurations of the three VMs under insufficient and sufficient situations are shown in Table 3 and Table 4, respectively.
- N: N is the prediction length; the more the controller predicts, the more stable the system is. However, too long of a prediction length may lead to enormous calculation. Therefore, we set $N = 6$ as its initial value, which is a relatively short prediction length, on the premise of system stability.
- λ: $\lambda \in (0, 1]$ is the forgetting factor, which will weaken the impact of historical data as the current data are updated. As shown in (16), the current data are maximally weighted, and the historical data are exponentially forgotten. In this paper, according to general studies [45], we choose the initial λ as 0.9.
- d: d is the latency of the system, and each control interval is 2 minutes. The latency is expressed as d control intervals. We monitored the system and determined that our system requires approximately 3-4 minutes to execute a resource adjustment, which indicates that the actual latency is approximately 3-4 minutes. To ensure the stability of the control system, the latency of the controller should be nearly the same as the actual latency. As one control interval is 2 minutes, we set the latency of the controller as $d = 2$.
- $\alpha : \alpha \in R^{3\times3} = diag\{\alpha_1, \alpha_2, \alpha_3\}$ is the transition coefficient, which determines the transition rate from the reference workload to the actual workload. We simulate three representative values to evaluate its impact on the controller, and the simulation results are shown in Fig. 5.

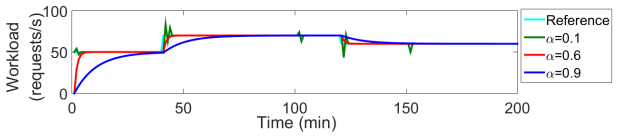

FIGURE 5. Setting values of α simulation.

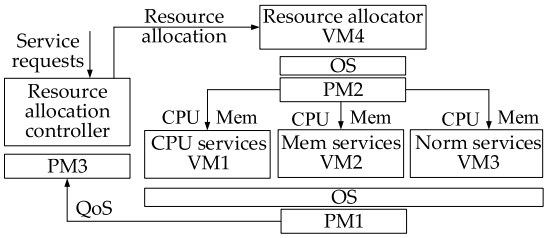

FIGURE 6. Insufficient resources situation experimental environment.

TABLE 3. Insufficient situation experiment configuration.

PM	OS	CPU (core)	Mem (G)	VM initial configuration				
				VM	OS	CPU	Mem	Function
1	CentOS 7	16	32	1	CentOS 7	3	4	Route planning
				2	CentOS 7	1	8	Image analysis
				3	CentOS 7	1	3	Regional location
2	CentOS 7	4	8	4	CentOS 7	2	6	Resource allocator
3	CentOS7	2	8	Resource allocation controller				

As shown in Fig. 5, the fastest transition rate is $\alpha = 0.1$. However, although the transition rate of $\alpha = 0.1$ is the fastest, the transition process is not stable because the real system cannot adapt to frequent dramatic changes made by controller. Under the setting of $\alpha = 0.9$, it takes too much time to transit from the reference value to the actual value, which indicates that the system cannot react to the workload changes in time. For $\alpha = 0.6$, although transit is slower than for $\alpha = 0.1$, the transition process is more stable than for $\alpha = 0.1$ and is considerably faster than for $\alpha = 0.9$. Compared with $\alpha = 0.1$ and $\alpha = 0.9$, the transition process is fast and stable, and thus $\alpha = 0.6$ is more acceptable than other values.

B. INSUFFICIENT RESOURCES SITUATION

To evaluate the efficiency of our approach, a critical situation is implemented in which the resource pool is insufficient. An overview of the experimental environment under the insufficient resource condition is shown in Fig. 6. These three VMs are implemented on one PM, indicating that the resource pool that they share is finite. Because the resource pool is insufficient, we restrict the total CPU use to no more than 16 cores and memory consumption to no more than 32 G. The configurations of the physical machines and the initial configuration of the VMs are described in Table 3.

C. SUFFICIENT RESOURCES SITUATION

When the resource pool is sufficient, the three VMs are deployed on three physical machines. An overview of the

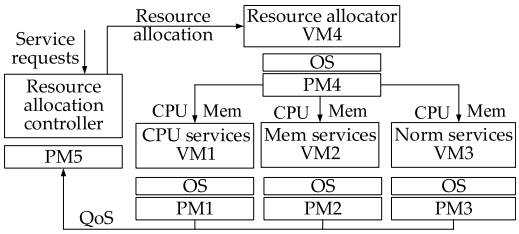

FIGURE 7. Sufficient resources situation experimental environment.

TABLE 4. Sufficient situation experiment configuration.

PM	OS	CPU (core)	Mem (G)	VM initial configuration				
				VM	OS	CPU	Mem	Function
1	CentOS 7	16	32	1	CentOS 7	3	4	Route planning
2	CentOS 7	16	32	2	CentOS 7	1	8	Image analysis
3	CentOS 7	16	32	3	CentOS 7	1	3	Regional location
4	CentOS 7	4	8	4	CentOS 7	2	6	Resource allocator
5	CentOS 7	2	8	Resource allocation controller				

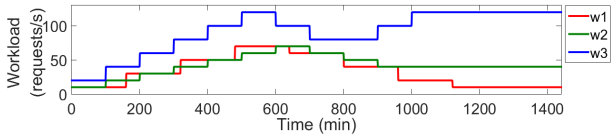

FIGURE 8. Workloads of the three VMs.

sufficient resources situation is shown in Fig. 7. The three types of services run on three VMs, but unlike the insufficient resources situation, these three VMs run on three PMs separately. The experimental configuration of the sufficient resources situation is shown in Table 4.

D. WORKLOADS

In this experiment, we use the workload generator LoadRunner 11 to generate three types of workloads for the three VMs. After classifying the services above, the requests will be assigned to the relative VM according to the details of the requirements. The three workloads of the three VMs $w_1(k)$, $w_2(k)$ and $w_3(k)$ are shown in Fig. 8. They represent the number of requests per second of the three VMs, and the experiment lasts 24 hours.

VI. EXPERIMENTAL RESULTS

We analyze the experimental results from seven aspects by comparing them with those of existing approaches under sufficient and insufficient resource pool situations: (1) CPU resource allocation; (2) memory resource allocation; (3) CPU resource utilization; (4) memory resource utilization; (5) service response time; (6) service request satisfaction rate and (7) adjustment time. For each aspect, the analysis is performed under two situations: sufficient and insufficient resource pools.

Resource utilization is the percentage of consumed resources of all allocated resources and is also used as an evaluation index reflecting the efficiency of our approach. Higher resource utilization is not better because excessive resource utilization (more than 90%) may lead to failed service requirements, while low resource utilization (less than 30%) may lead to resource waste.

The request satisfaction rate is the percentage of the number of services meeting the requests of the total number of services. Because the adjustment time can reflect the complexity of each approach, we analyze this rate to evaluate whether an approach enables a quick response to an unpredictable burst in workload.

A. INSUFFICIENT RESOURCES SITUATION RESULTS

1) CPU RESOURCE ALLOCATION

For the insufficient resources situation, the CPU allocations are shown in Fig. 9. The LP and RL approaches allocate considerably more CPU resources to VM1 than the control theoretic approaches (SISO, de-MIMO and co-MIMO). However, the RL and LP approaches allocate less CPU resources to VM2 and VM3 than the control theoretic approaches during the peak of the workload, as shown in Fig. 9-2 and Fig. 9-3, because the RL and LP approaches determine resource allocation by only considering whether the QoS satisfies the requirements. As a result, they allocate excess CPU resources to VM1, resulting in a lack of resources for VM2 and VM3, especially during the peak workload.

As shown in Fig. 9-1 and Fig. 9-3, the SISO approach allocates nearly the same amount of CPU resources as the MIMO approaches (de-MIMO and co-MIMO) to VM1 and VM3 but obviously less than the MIMO approaches during the peak workload. However, it always allocates more CPU resources to VM2 than the MIMO approaches, indicating that VM2 is CPU over-allocated. Since the SISO approach only focuses on a single metric and the VM2 deploys memory-intensive services that consume massive memory resources but few CPU resources, the SISO approach allocates not only extensive memory resources but also massive CPU resources to VM2. As a result, insufficient CPU resources are allocated to VM3, and VM3 is allocated obviously less CPU resources than by the other approaches, especially during the peak workload.

The de-MIMO approach allocates nearly the same or even greater CPU resources to the three VMs but less CPU resources than the co-MIMO approach during the peak workload because it easily decomposes one MIMO control system to several SISO control systems. However, interactions and competition occur between services when the resource pool is insufficient, and the interactions and competition cannot be easily decomposed. Therefore, inaccurate decomposition leads to CPU under-allocation during the peak workload.

2) MEMORY RESOURCE ALLOCATION

Memory resource allocation in the insufficient resource pool situation is shown in Fig. 10. Similar to CPU resource

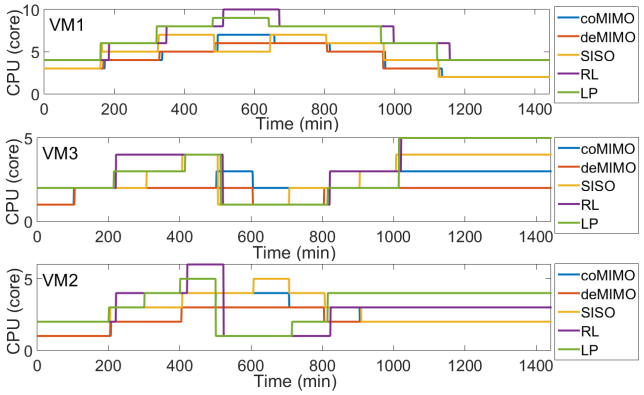

FIGURE 9. CPU resource allocation under the insufficient situation.

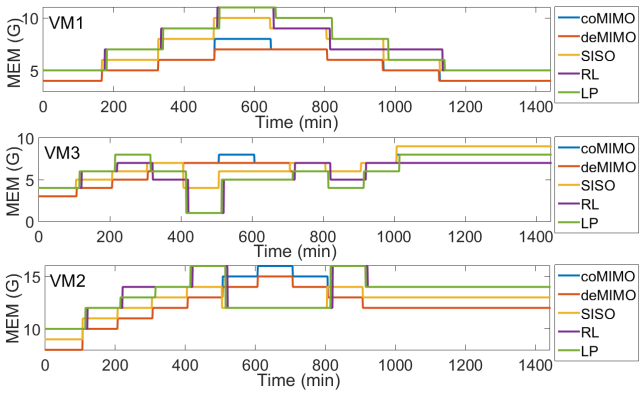

FIGURE 10. Memory resource allocation under the insufficient situation.

allocation, the RL and LP approaches allocate considerably more memory resources than the control theoretic approaches. For VM2 and VM3, the memory resource allocation is obviously less than that with the control theoretic approaches during the peak workload because no feedback is available with the RL and LP approaches, which only focus on whether the QoS meets the requirements. Therefore, over-allocation to VM1 leads to a lack of memory for VM2 and VM3.

Since the SISO approach focuses on a single type of resource allocation, it also allocates massive memory resources to VM1. However, VM2 deploys memory-intensive services requiring massive memory resources, and insufficient memory resources are available for VM2 and VM3, especially VM2. Therefore, as shown in Fig. 10-2 and Fig. 10-3, the SISO approach allocates obviously less memory resources than the MIMO approaches to VM2 and VM3 during the peak workload.

Similar to CPU resource allocation, the de-MIMO approach allocates nearly the same amount of memory resources as the co-MIMO approach to the three VMs but less than the co-MIMO approach during the peak workload because the de-MIMO approach ignores the interactions and competition between services, especially when the resource pool is insufficient. When the interactions and competition

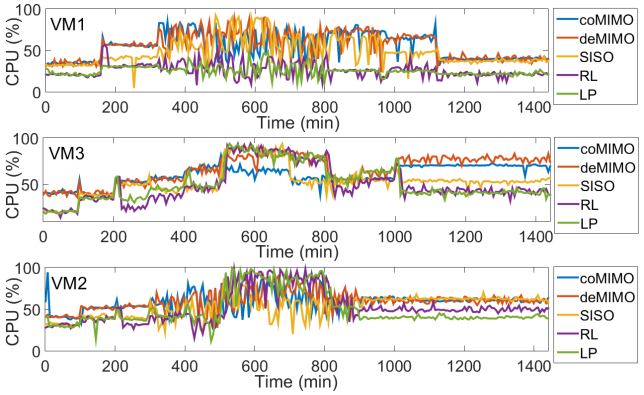

FIGURE 11. CPU resource utilization under the insufficient situation.

persist, inaccurate decomposition may lead to unreasonable resource allocation.

3) CPU RESOURCE UTILIZATION

The CPU resource utilization of the three VMs under the insufficient resource pool situation is shown in Fig. 11. As the RL and LP approaches allocate massive CPU resources to VM1, the CPU resource utilization of the RL and LP approaches for VM1 is obviously lower than that of the control theoretic approaches, indicating that VM1 is CPU over-allocated. Therefore, insufficient CPU resources are available for VM2 and VM3, and the CPU resource utilization of the RL and LP approaches is greater than 90% during the peak workload, which indicates that some services may violate the QoS requirements.

The CPU resource utilization of the SISO approach for VM1 is lower than that for the MIMO approaches but obviously higher than that for the MIMO approaches during the peak workload. However, the utilization of VM2 is only approximately 20%-40%, as shown in Fig. 11-2, and is always obviously lower than that for the MIMO approaches because the SISO approach is manipulated by a single-control strategy that allocates not only massive memory but also massive CPU resources to VM2, which leads to VM2 CPU over-allocation but VM1 CPU under-allocation during the peak workload. This unreasonable resource allocation also leads to a lack of CPU resources for VM3, and the CPU resource utilization of VM3 is greater than 90% during the peak workload.

For the de-MIMO approach, consistent with CPU resource allocation, the CPU resource utilization of the three VMs is nearly the same as that for the co-MIMO approach but higher than that for the co-MIMO approach during the peak workload. In addition, we find that some CPU resource utilization is greater than 90% during the peak workload, indicating poor QoS that cannot satisfy the requirements because the de-MIMO approach does not consider the interactions and competition between services.

4) MEMORY RESOURCE UTILIZATION

The memory resource utilization of the three VMs is shown in Fig. 12. Similar to CPU resource utilization, the memory

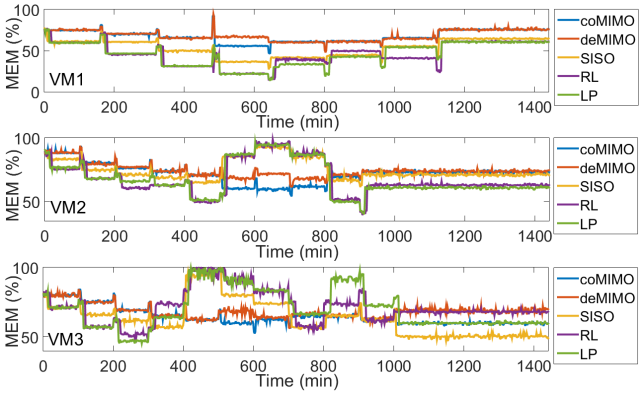

FIGURE 12. Memory resource utilization under the insufficient situation.

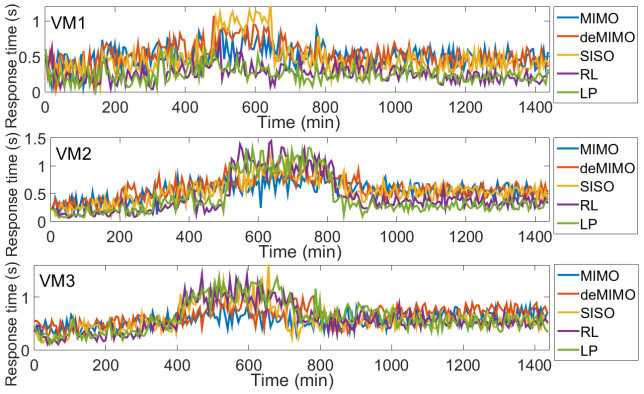

FIGURE 13. Service response time under the insufficient situation.

resource utilization of the RL and LP approaches for VM1 is obviously lower than that for the control approaches. For VM2 and VM3, memory resource utilization is greater than 90% during the peak workload and less than 30% when the peak passes due to the over-allocation of VM1 by the RL and LP approaches, with insufficient memory resources for VM2 and VM3.

The SISO approach results in different CPU resource utilization. The memory utilization of VM1 shown in Fig. 12-1 is only approximately 70%, which is lower than that for the MIMO approaches all the time, but the CPU resource utilization of VM1 (shown in Fig. 11-1) is greater than 90% during the peak workload. However, as shown in Fig. 12-2, the memory resource utilization of VM2 is greater than 90%, but the CPU resource utilization of VM2 (shown in Fig. 11-2) is only approximately 40% during the peak workload. Thus, VM1 is memory over-allocated while CPU is under-allocated, and VM2 is CPU over-allocated while memory is under-allocated. As mentioned above, focusing on a single metric leads to unreasonable resource allocation. As a result, both CPU and memory resources are under-allocated to VM3 during the peak workload; therefore, both the CPU utilization and memory resource utilization of VM3 are greater than 90% during that time.

The memory resource utilization of the de-MIMO approach is similar to the CPU resource utilization and is nearly the same as that with the co-MIMO approach but greater than 80% during the peak workload. This occurs because when resources are limited, competition and interactions between services occur, and ignoring the competition and interactions leads to greater than 80% resource utilization of some services.

5) AVERAGE SERVICE RESPONSE TIME

The average response time during one minute is shown in Fig. 13. As more resources are allocated, QoS increases, reflecting a lower response time. The RL and LP approaches allocate high amounts of CPU and memory resources to VM1, and the response time is therefore obviously lower than that for the control theoretic approaches. However, for VM2 and VM3, the response time is obviously higher than that for the control theoretic approaches during the peak workload because insufficient resources are available for VM2 and VM3.

For the SISO approach, the response time of VM1 is obviously higher than that for the MIMO approaches when the peak workload arrives. Because VM1 is memory over-allocated but CPU resources are under-allocated, the lack of CPU resources for VM1 leads to lower QoS and requirement violation. For VM2, the response time is also higher than that for the MIMO approaches during the peak workload. The poor QoS, which is different from VM1, is due to a lack of memory resources for VM2 as VM2 is CPU over-allocated but memory is under-allocated. Therefore, any type of resource under-allocation may lead to poor QoS. Due to this unreasonable resource allocation, both CPU and memory resources are insufficient for VM3, the response time is considerably higher than that for the MIMO approaches during the peak workload, and the poor QoS indicates that many services cannot satisfy the QoS requirements.

For the de-MIMO approach, the response time of the three VMs is nearly the same as that for the co-MIMO approach but higher than that for the co-MIMO approach during the peak workload because interactions and competitions occur when resources are insufficient. Ignoring these interferences leads to lower QoS than in the co-MIMO approach.

6) SERVICE REQUEST SATISFACTION RATES

The service request satisfaction rate is the percentage of the total number of services that meet the requests. The request satisfaction rates of the five approaches under the insufficient resources pool situation are shown in Fig. 14. For VM1, the satisfaction rates of the RL and LP approaches are 100%, and the MIMO approaches shows a rate greater than 90%. However, the satisfaction rate of the SISO approach is only 88.96% because of the unreasonable resource allocation of this approach, which results in VM1 CPU under-allocation but memory over-allocation. This CPU under-allocation leads to the low satisfaction rate of the SISO approach.

For VM2 and VM3, the satisfaction rate of the co-MIMO approach is greater than 90%, but the rate for the de-MIMO

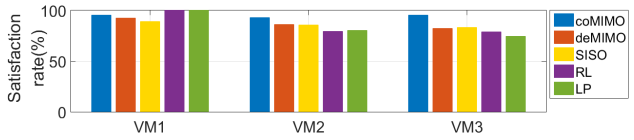

FIGURE 14. Satisfaction rate under the insufficient situation.

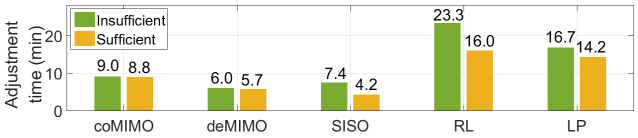

FIGURE 15. Adjustment time.

approach is less than 90%. As mentioned above, the reason for the low satisfaction rate of the de-MIMO approach is that interference between services is ignored. The satisfaction rate of the SISO approach for VM2 is only 85.76%. Unlike VM1, the low satisfaction rate of the SISO approach for VM2 is due to a lack of memory resources rather than CPU resources. As a result of memory over-allocation for VM1 and CPU over-allocation for VM2, both CPU and memory resources are insufficient for VM3. Therefore, the satisfaction rate of the SISO approach for VM3 is only 83.19%. However, the satisfaction rates of the RL and LP approaches are less than 80%, indicating that many services fail to meet the requests. The reason for such poor satisfaction rates is that the RL and LP approaches allocate extra resources to VM1, resulting in a serious lack of resources for VM2 and VM3.

7) ADJUSTMENT TIME

The adjustment time results of the five approaches are shown in Fig. 15. Every time the request workloads change, the RL approach applies adjustments obviously later than the other approaches. As shown in Fig. 15, the adjustment time of these five approaches decrease in the order RL, LP, co-MIMO, de-MIMO and SISO, indicating that an easier algorithm corresponds to less adjustment time needed. The RL approach requires considerably more time than the other approaches because reinforcement learning requires extensive time and cannot sufficiently react when fluctuations occur frequently. The LP approach cannot predict future resource demands, and when the workload changes, it cannot promptly adjust resource allocation; therefore, the adjustment time is higher than that for the control theoretic approaches.

B. SUFFICIENT RESOURCES SITUATION RESULTS

1) CPU RESOURCE ALLOCATION

The CPU resource allocations under the sufficient situation are shown in Fig. 16. The RL and LP approaches allocate considerably more CPU resources to the VMs than the control theoretic approaches under the sufficient situation. This result occurs because the RL and LP approaches determine resource allocation only by analyzing whether the QoS can satisfy the requirements or whether the predefined threshold is reached, whereas the control theoretic approaches calculate

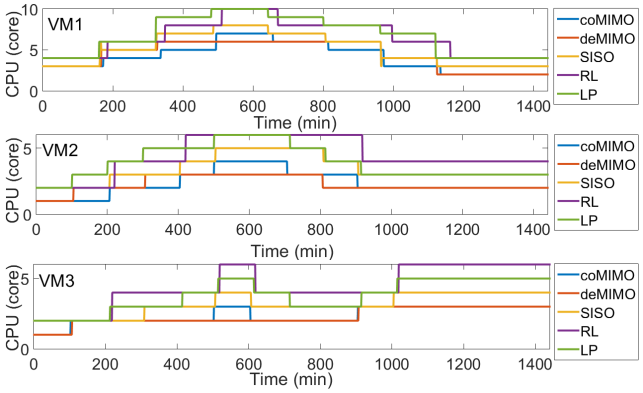

FIGURE 16. CPU resource allocation under the sufficient situation.

resource allocation by narrowing the gap between the provided QoS or resource and the users' requirements.

The SISO approach allocates nearly the same or even more CPU resources to VM1 and VM3 than the MIMO approaches but obviously more CPU resources to VM2 than the MIMO approaches, which indicates that it not only allocates massive memory but also massive CPU resources to VM2. CPU resource allocation is obviously greater than that for the MIMO approaches because the SISO approach is used to allocate a single type of resource to VMs, which is manipulated and constrained by a single control strategy.

The de-MIMO approach allocates nearly the same or even more CPU resources to the VMs than the co-MIMO approach. However, when the peak workload arrives, it allocates less than the co-MIMO approach because, as resource utilization increases, interactions and competition between services occur. The de-MIMO approach only decomposes one MIMO control system to multiple easy SISO control systems, and the interactions and competition cannot be easily decomposed. Therefore, inaccurate decomposition leads to less CPU resources allocated by the de-MIMO approach than that by the co-MIMO approach during the peak workload.

2) MEMORY RESOURCE ALLOCATION

The memory resource allocations of the three VMs are shown in Fig. 17. Similar to CPU resource allocation, the RL and LP approaches allocate obviously more memory resources to the three VMs than the control theoretic approaches since they do not try to minimize resource allocation on the basis of satisfying the QoS requirements.

Unlike CPU resource allocation, the SISO approach allocates obviously more memory resources to VM1 than the MIMO approaches and nearly the same amount of memory resources to VM2 and VM3 as the MIMO approaches. This result occurs because the SISO approach only focuses on a single metric and thus allocates both massive CPU and memory resources to VM1.

Similar to CPU resource allocation, the de-MIMO approach also allocates nearly the same or even more memory to the three VMs than the co-MIMO approach but less during the peak workload because the de-MIMO approach ignores

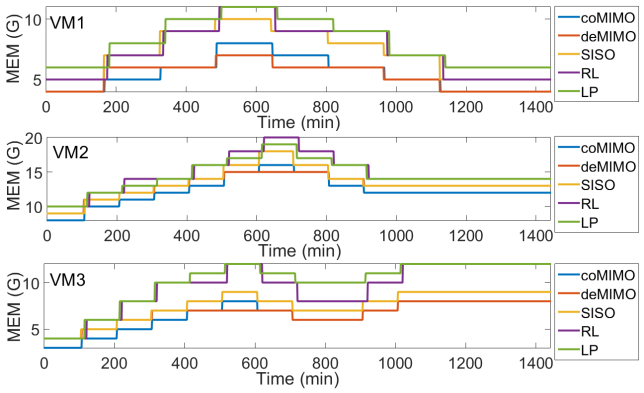

FIGURE 17. Memory resource allocation under the sufficient situation.

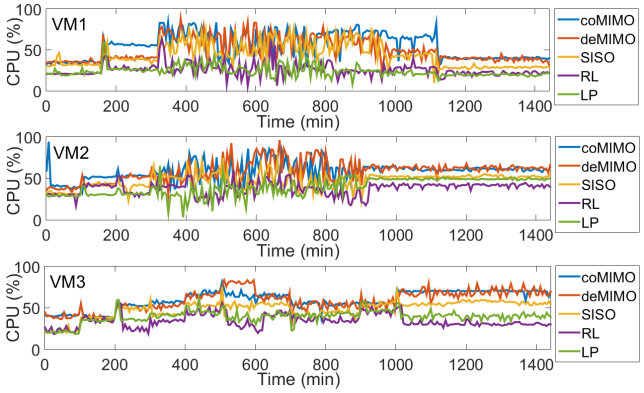

FIGURE 18. CPU resource utilization under the sufficient situation.

interactions and competition between services when resource utilization increases.

3) CPU RESOURCE UTILIZATION

The CPU resource utilization of the three VMs under the sufficient situation is shown in Fig. 18. Consistent with CPU resource allocation to the VMs, the CPU resource utilization of the RL and LP approaches is the lowest for the three VMs compared with that of the other approaches, with some services even under 30%, which indicates CPU resource waste.

For the SISO approach, the resource utilization of VM1 and VM3 is nearly the same as that for the MIMO approaches, which is approximately 60%-80%. However, for VM2, it is only approximately 20%-50%, indicating that VM2 is CPU over-allocated, which occurs because the SISO approach focuses on a single type of resource allocation.

The utilization of the de-MIMO approach for the three VMs is nearly the same or even less than that of the co-MIMO approach but more than that of the co-MIMO approach during the peak workload because the de-MIMO approach ignores interactions and competition between services when resource utilization increases.

4) MEMORY RESOURCE UTILIZATION

The memory resource utilization of the VMs is shown in Fig. 19. Greater resource allocation corresponds to lower resource utilization. The lowest memory resource utilization

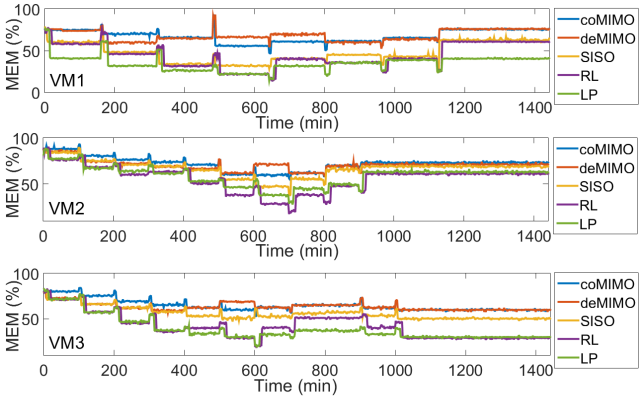

FIGURE 19. Memory resource utilization under the sufficient situation.

is observed with the RL and LP approaches because these approaches have no feedback and only focus on whether requirements are met.

Unlike CPU resource utilization, the memory resource utilization of the SISO approach for VM2 and VM3 is nearly the same as that with the MIMO approaches. However, for VM1, the memory resource utilization is only approximately 30%-50%. The SISO approach allocates both massive CPU and memory resources to VM1, which leads to memory over-allocation and obviously lower memory resource utilization than that for the MIMO approaches.

For the de-MIMO approach, the memory resource utilization of the VMs is also higher than that for the co-MIMO approach during the peak workload since interactions and competition occur as resource utilization increases; however the de-MIMO approach does not consider the interactions and competition.

5) AVERAGE SERVICE RESPONSE TIME

As shown in Fig. 20, the average response time increases as utilization increases and vice versa. As the RL and LP approaches allocate massive CPU and memory resources to the three VMs, these approaches show the lowest response time among the five approaches, which indicates the best QoS.

Although the SISO approach allocates nearly the same amount of CPU resources to VM1 as the MIMO approaches, memory is over-allocated, and the response time of VM1 is therefore lower than that for the MIMO approaches. Similarly, VM2 is allocated nearly the same amount of memory resources as with the MIMO approaches, but CPU is over-allocated, and the response time of VM2 is therefore lower than that with the MIMO approaches. For VM3, the SISO approach allocates nearly the same amount of CPU and memory resources as the MIMO approaches, and therefore the response time is nearly the same.

In response to resource allocation, the response time of the de-MIMO approach for the VMs is nearly the same or even less than that for the co-MIMO approach. However, contrary to expectations, the response time is higher than that

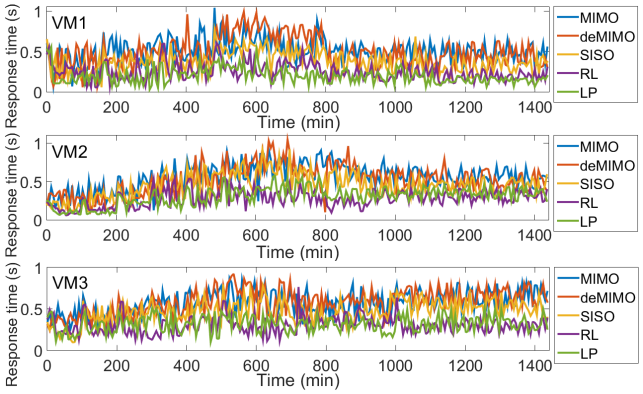

FIGURE 20. Service response time under the sufficient situation.

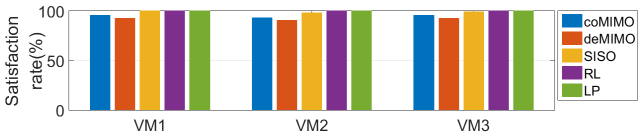

FIGURE 21. Satisfaction rate under the sufficient situation.

for the co-MIMO approach during the peak workload because interactions and competition between services are ignored.

6) SERVICE REQUEST SATISFACTION RATES

The satisfaction rates of the five approaches for the three VMs under the sufficient situation are shown in Fig. 21. We find that the satisfaction rates of all approaches for the three VMs are greater than 90%, indicating that nearly all services meet the QoS requirement. Although the co-MIMO approach has a lower satisfaction rate and a higher response time than the SISO, RL and LP approaches, its satisfaction rate is still greater than 90%, which indicates that the co-MIMO approach still meets the SLA of users.

7) ADJUSTMENT TIME

The adjustment times of the five approaches under the sufficient situation are shown in Fig. 15. We find that the adjustment time is lower under the sufficient resource situation than the insufficient situation. Because the resource allocation algorithm itself consumes substantial resources, it requires more time when the resource pool is insufficient. In addition, the adjustment times of the RL, LP and SISO approaches under the insufficient resource pool situation are obviously greater than those under the sufficient resource pool situation. However, for the MIMO approaches, the adjustment times are nearly the same under these two situations because the unreasonable resource allocation of the RL, LP and SISO approaches causes a lack of resources, resulting in greater time consumption.

VII. FUTURE WORK

In future work, we will evaluate the efficiency of our proposed approach using some real-world workloads [49], [50] in real time. In addition, we will also extend the scale of the experiment to evaluate the stability and scalability of our resource allocation control system.

VIII. CONCLUSIONS

In this paper, we propose an adaptive multivariable control strategy for resource allocation in cloud computing systems. Our proposed approach provides a powerful mechanism to manage unpredictable changes and uncertainties, executes combinatorial resource allocation according to the various characteristics of different services and considers interference.

Our adaptive resource allocation approach has been evaluated under situations of sufficient and insufficient resources. The experimental results show that the proposed approach enables management of the challenges of resource allocation in cloud computing systems, saves operating costs and increases resource utilization. Simultaneously, this approach ensures QoS, adaptively reacts to the open environment full of interference and manages unpredictable resource demands in real time.

REFERENCES

[1] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Trans. Services Comput.*, vol. 5, no. 2, pp. 164–177, Apr./Jun. 2012.
[2] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.
[3] A. Yousafzai *et al.*, "Cloud resource allocation schemes: Review, taxonomy, and opportunities," *Knowl. Inf. Syst.*, vol. 50, no. 2, pp. 347–381, May 2016.
[4] Z. Zheng, K. S. Trivedi, K. Qiu, and R. Xia, "Semi-Markov models of composite Web services for their performance, reliability and bottlenecks," *IEEE Trans. Service Comput.*, vol. 10, no. 3, pp. 448–460, May/Jun. 2017.
[5] N. Singh and B. M. Keating, "Hyper-localizing e-Commerce strategy: An emerging market perspective," in *Emerging Markets from a Multidisciplinary Perspective*. Cham, Switzerland: Springer, 2018, pp. 89–94.
[6] G. Linden. (2006). Make data useful. Amazon Findory. [Online]. Available: http://www.gduchamp.com/media/StanfordDataMining.2006-11-28.pdf
[7] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107–1117, Jun. 2013.
[8] G. Zhang, X. Zhu, W. Bao, H. Yan, and D. Tan, "Local storage-based consolidation with resource demand prediction and live migration in clouds," *IEEE Access*, vol. 6, pp. 26854–26865, 2018.
[9] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generat. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012.
[10] A. G. Kumbhare, Y. Simmhan, M. Frincu, and V. K. Prasanna, "Reactive resource provisioning heuristics for dynamic dataflows on cloud infrastructure," *IEEE Trans. Cloud Comput.*, vol. 23, no. 2, pp. 105–118, Apr./Jun. 2015.
[11] L. Wang *et al.*, "GreenDCN: A general framework for achieving energy efficiency in data center networks," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 1, pp. 4–15, Jan. 2014.
[12] H. Zhang, H. Jiang, B. Li, F. Liu, A. V. Vasilakos, and J. Liu, "A framework for truthful online auctions in cloud computing with heterogeneous user demands," *IEEE Trans. Comput.*, vol. 65, no. 3, pp. 805–818, Mar. 2016.
[13] L. Mashayekhy, M. M. Nejad, D. Grosu, and A. V. Vasilakos, "An online mechanism for resource allocation and pricing in clouds," *IEEE Trans. Comput.*, vol. 65, no. 4, pp. 1172–1184, Apr. 2016.
[14] M. R. Rahimi, N. Venkatasubramanian, S. Mehrotra, and A. V. Vasilakos, "MAPCloud: Mobile applications on an elastic and scalable 2-tier cloud architecture," in *Proc. IEEE UCC*, Nov. 2012, pp. 83–90.
[15] S. S. Yau and H. G. An, "Adaptive resource allocation for service-based systems," in *Proc. ACM Asia–Pacific Symp. Internetware*, Oct. 2009, p. 3.
[16] H. Morshedlou and M. R. Meybodi, "Decreasing impact of SLA violations: A proactive resource allocation approachfor cloud computing environments," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 156–167, Apr./Jun. 2014.

[17] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "CloudScale: Elastic resource scaling for multi-tenant cloud systems," in *Proc. ACM Symp. Cloud Comput.*, Oct. 2011, p. 5.
[18] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449–458, Oct. 2015.
[19] J. Rao, X. Bu, C.-Z. Xu, and K. Wang, "A distributed self-learning approach for elastic provisioning of virtualized cloud resources," in *Proc. IEEE Int. Conf. Modeling, Anal. Simulation Comput. Telecommun. Syst.*, Jul. 2011, pp. 46–54.
[20] K. Gai, M. Qiu, and H. Zhao, "Cost-aware multimedia data allocation for heterogeneous memory using genetic algorithm in cloud computing," *IEEE Trans. Cloud Comput.*, to be published.
[21] R. Tolosana-Calasanz, J. Diaz-Montes, O. F. Rana, and M. Parashar, "Feedback-control & queueing theory-based resource management for streaming applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1061–1075, Apr. 2017.
[22] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog-computing-based radio access networks: Issues and challenges," *IEEE Netw.*, vol. 30, no. 4, pp. 46–53, Jul. 2016.
[23] Z. Yan, X. Li, M. Wang, and A. V. Vasilakos, "Flexible data access control based on trust and reputation in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 5, no. 3, pp. 485–498, Jul./Sep. 2017.
[24] M. Ali *et al.*, "SeDaSC: Secure data sharing in clouds," *IEEE Syst. J.*, vol. 11, no. 2, pp. 395–404, Jun. 2017.
[25] Z. Zheng, K. S. Trivedi, N. Wang, and K. Qiu, "Markov regenerative models of webservers for their user-perceived availability and bottlenecks," *IEEE Trans. Depend. Sec. Comput.*, to be published.
[26] G. Ning *et al.*, "Optimization of two-granularity software rejuvenation policy based on the Markov regenerative process," *IEEE Trans. Rel.*, vol. 65, no. 4, pp. 1630–1646, Dec. 2016.
[27] J. Rao, Y. Wei, J. Gong, and C.-Z. Xu, "QoS guarantees and service differentiation for dynamic cloud applications," *IEEE Trans. Netw. Service Manag.*, vol. 10, no. 1, pp. 43–55, Mar. 2013.
[28] S. Farokhi, P. Jamshidi, I. Brandic, and E. Elmroth, "Self-adaptation challenges for cloud-based applications: A control theoretic perspective," in *Proc. 10th Int. Workshop Feedback Comput.*, Apr. 2015.
[29] L. Lu *et al.*, "Application-driven dynamic vertical scaling of virtual machines in resource pools," in *Proc. IEEE Symp. Netw. Oper. Manage.*, May 2014, pp. 1–9.
[30] M. A. Islam, S. Ren, G. Quan, M. Z. Shakir, and A. V. Vasilakos, "Water-constrained geographic load balancing in data centers," *IEEE Trans. Cloud Comput.*, vol. 5, no. 2, pp. 208–220, Jun. 2017.
[31] Z. Song, Y. Sun, J. Wan, and P. Liang, "Data quality management for service-oriented manufacturing cyber-physical systems," *Comput. Elect. Eng.*, vol. 64, pp. 34–44, Nov. 2017.
[32] S. Gong, B. Yin, W. Zhu, and K. Cai, "An adaptive control strategy for resource allocation of service-based systems in cloud environment," in *Proc. IEEE Int. Conf. Softw. Qual. Rel. Secur.-Companion*, vol. 53, no. 4, Aug. 2015, pp. 32–39.
[33] L. Baresi, S. Guinea, A. Leva, and G. Quattrocchi, "A discrete-time feedback controller for containerized cloud applications," in *Proc. ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, Nov. 2016, pp. 217–228.
[34] I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. Hoboken, NJ, USA: Wiley, 2007, pp. 55–64.
[35] S. Gong, B. Yin, W. Zhu, and K.-Y. Cai, "Adaptive resource allocation of multiple servers for service-based systems in cloud computing," in *Proc. IEEE Int. Conf. Comput. Softw. Appl.*, Jul. 2017, pp. 603–608.
[36] P. S. Saikrishna, R. Pasumarthy, and N. P. Bhatt, "Identification and multivariable gain-scheduling control for cloud computing systems," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 3, pp. 792–807, May 2017.
[37] P. Padala *et al.*, "Automated control of multiple virtualized resources," in *Proc. ACM Eur. Conf. Comput. Syst.*, Apr. 2009, pp. 13–26,
[38] Y. Xia, M. Zhou, X. Luo, Q. Zhu, J. Li, and Y. Huang, "Stochastic modeling and quality evaluation of infrastructure-as-a-service clouds," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 1, pp. 162–170, Jan. 2015.
[39] S. Farokhi, E. B. Lakew, C. Klein, I. Brandic, and E. Elmroth, "Coordinating CPU and memory elasticity controllers to meet service response time constraints," in *Proc. IEEE Int. Conf. Cloud Autonomic Comput.*, Sep. 2015, pp. 69–80.
[40] S. Y. Zhang and L. Q. Gao, *Modern Control Theory*. Beijing, China: Tsinghua Univ. Press, 2007, pp. 15–18.
[41] E. F. Camacho and C. B. Alba, *Model Predictive Control*. London, U.K.: Springer, 2013, pp. 47–79.
[42] W. Venters and E. Whitley, "A critical review of cloud computing: Researching desires and realities," *J. Inf. Technol.*, vol. 27, no. 3, pp. 179–197, Sep. 2012.
[43] S. Kim, Y. He, S. Hwang, S. Elnikety, and S. Choi, "Delayed-dynamic-selective (DDS) prediction for reducing extreme tail latency in Web search," in *Proc. ACM Int. Conf. Web Search Data Mining*, Feb. 2015, pp. 7–16.
[44] W. Li, Y. Xia, M. Zhou, X. Sun, and Q. Zhu, "Fluctuation-aware and predictive workflow scheduling in cost-effective infrastructure-as-a-service clouds," *IEEE Access*, vol. 6, pp. 61488–61502, 2018.
[45] A.-G. Wu, Y.-Y. Qian, and W.-J. Wu, "Bias compensation-based recursive least-squares estimation with forgetting factors for output error moving average systems," *IET Signal Process.*, vol. 8, no. 5, pp. 483–494, Jul. 2014.
[46] Alibaba Cloud. (2018). *Elastic Computing*. [Online]. Available: https://www.aliyun.com/product/ecs?spm=5176.8789780.765261.205.euNuAm
[47] F. Wuhib, R. Stadler, and H. Lindgren, "Dynamic resource allocation with management objectives—Implementation for an OpenStack cloud," in *Proc. IEEE Int. Conf. Netw. Service Manage.*, Oct. 2012, pp. 309–315.
[48] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *J. Grid Comput.*, vol. 12, no. 4, pp. 559–592, Dec. 2014.
[49] A. Hameed *et al.*, "A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems," *Computing*, vol. 98, no. 7, pp. 751–774, Jul. 2016.
[50] F. Farahnakian *et al.*, "Using ant colony system to consolidate VMs for green cloud computing," *IEEE Trans. Services Comput.*, vol. 8, no. 2, pp. 187–198, Mar. 2015.

SIQIAN GONG received the M.S. degree from Zhejiang University, China, in 2013. She is currently pursuing the Ph.D. degree with the School of Automation Science and Electrical Engineering, Beihang University, China. Her research interests include cloud computing and software cybernetics.

BEIBEI YIN received the Ph.D. degree from Beihang University, China, in 2010. She was a Research Scholar with the Department of Electrical and Computer Engineering, Duke University, in 2015. She is currently a Lecturer with Beihang University. Her research interests include software testing, software reliability, and software cybernetics.

ZHENG ZHENG received the Ph.D. degree from the Chinese Academy of Science, China, in 2006. He was a Research Scholar with the Department of Electrical and Computer Engineering, Duke University, in 2014. He is currently a Professor with Beihang University, China. His research interests include software dependability modeling and software fault localization. He is a Senior Member of the IEEE.

KAI-YUAN CAI received the B.S., M.S., and Ph.D. degrees from Beihang University, China, in 1984, 1987, and 1991, respectively, where he has been a Full Professor, since 1995. He has been a Cheung Kong Scholar (Chair Professor) jointly appointed by the Ministry of Education of China and the Li Kashing Foundation of Hong Kong, since 1999. His research interests include software testing, software reliability, reliable control, and software cybernetics.