International Journal of Computational Intelligence Systems
Vol. **12(2)**; 2019, *pp.* 485–497
DOI: https://doi.org/10.2991/ijcis.d.190322.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

An Adaptive Control Method for Resource Provisioning with Resource Utilization Constraints in Cloud Computing

Siqian Gong*, Beibei Yin, Zheng Zheng, Kai-yuan Cai

School of Automation Science and Electrical Engineering, Beihang University, Beijing, China

ARTICLE INFO

Article History
Received 10 Jan 2019
Accepted 22 Mar 2019

Keywords
Adaptive control
Neural network
Dynamic resource provisioning
Cloud computing
QoS
Resource efficiency

ABSTRACT

Cloud computing enables users to purchase virtual resources on demand; therefore, the service requests change over time. Dynamic resource provisioning for cloud computing has become a key challenge. To reduce the associated costs, resource utilization must be improved, and ensure the Quality of Service (QoS) to meet the service-level agreements (SLAs). However, it is difficult to improve the resource utilization level and maintain a high QoS with fluctuating workloads in shared cloud computing systems. In this paper, we propose an adaptive control method for resource provisioning in cloud computing systems to simultaneously improve the resource utilization, achieve a satisfactory QoS, and react to the dynamic workload. We proposed an approach integrating adaptive multi-input and multi-output (MIMO) control and radial basis function (RBF) neural network to react to the highly dynamic workloads, and precisely control the resource allocation to improve resource utilization based on the maximum allowable QoS requirements. Experiments based on real-world workloads show that the proposed approach jointly improves resource utilization, maintains a satisfactory QoS, and handles workload fluctuations in a coordinated manner.

1. INTRODUCTION

Cloud computing provides a pay-as-you-go mode in which users can purchase virtual resources as they desire [1]. As the service requests change over time, users may oversubscribe the resources for the possible peak workload; thus services may suffer from underutilization as over-provisioning for the peak workload [2]. Therefore, resource allocation in cloud computing should be a dynamic procedure that optimizes resource utilization, ensures Quality of Service (QoS), and react to the unpredictable workload fluctuations.

Recent studies have focused on dynamic resource allocation by maximizing resource utilization in cloud computing systems [3–5]. Maximizing resource utilization provides many benefits for both users and service providers; for example, it can greatly reduce users' costs [6]. However, maximizing resource utilization can cause overutilization, which may lead to poor QoS and even service-level agreement (SLA) violations. For example, we assume that a service requires a response time should be less than 0.6 s. Normally, the service response time is approximately 0.4 s at a resource utilization rate of 60%. If resource utilization rate is maximized, the utilization rate increases to approximately 90%. At this time, the response time increases to 0.9 s due to the insufficient available resources. Excessive resource utilization rate may lead to disputes for resources. As a result, the QoS becomes poor, and the requirements cannot be met. It is difficult to find a reasonable trade-off between maximizing resource utilization rate and improving QoS. Therefore, it is essential to maximize resource utilization and maintain a high QoS according to the dynamic workload. Existing solutions [7–10] have typically only focused on one objective, for example, maximizing resource utilization rate or ensuring a high QoS. However, these two objectives are conflict some times, and focusing on a single objective cannot address the reasonable trade-off between a high QoS and effective resource utilization, especially in dynamic workload situations. Therefore, a dynamic resource allocation approach with joint consideration of the QoS and resource utilization is required in a coordinated manner.

In this work, we use a multi-input multi-output (MIMO) control approach for dynamic resource allocation. MIMO control offers a control strategy that actuates multiple inputs and coordinates multiple interrelated outputs [11]. This method jointly considers the QoS requirements and resource utilization conditions in a coordinated manner by adjusting the resource allocation. Moreover, the feedback mechanism of MIMO control enables to react to the workload fluctuations to provide roughly the same QoS with the requirements maximum allowed QoS according to the dynamic workload at run time. In contrast to traditional MIMO control approaches [4, 12–14] that rely on empirical evaluations and manual adjustment, we design an adaptive module for the controller to adjust the parameters of the controller in real time. As a local approximation network, radial basis function (RBF) neural network can highly improve the convergence speed and avoids local minimum [15].

Particularly, when modelling the workload, we consider dynamic workloads in our approach. To react to the dynamic workload,

*Corresponding author. Email: gongsiqian@buaa.edu.cn

two scaling architectures, that is, horizontal scaling and vertical scaling, are used to dynamically adjust the resource allocation. Horizontal scaling methods [7, 16–19] perform dynamic resource allocation by changing the number of virtual machines (VMs) for services. In such methods, one VM is treated as a scaling unit. However, this coarse-grained approach may lead to underutilization as a result of excessive allocation. Vertical scaling methods [5, 20–26] dynamically allocate resources by changing the resource level, such as the CPU and memory allocation for each VM. This fine-grained approach easily meets the specific resource demands for services. In this paper, we adopt the vertical scaling architecture to adjust the CPU and memory resources for each VM.

To verify our approach, we conduct experiments on resource allocation in the cloud computing system using a real-world workload. We compare the proposed approach with a horizontal scaling approach that has been commonly cited and a similar vertical scaling approach based on control theory. The experimental results show that the proposed approach can adaptively improve resource utilization within a reasonable range, ensure QoS according to the requirements, and cope with dynamic workloads. The main contributions of this paper are as follows:

1. We design a resource management system adopting MIMO control that making dynamic resource allocation based on joint objectives of QoS requirements, resource utilization, and workload.
2. We propose an adaptive module using RBF neural network to adjust the controller's parameters online, so that the resource management system can cope with the unpredictable changes of the workload adaptively.
3. We compare a horizontal approach and a single-input and single-output (SISO) control approach with our adaptive MIMO approach to evaluate the efficiency of the proposed adaptive MIMO control approach using a real workload in a cloud computing system, and expand the experiment to a larger scale.
4. We analyze the experimental results find that the proposed adaptive MIMO control approach enables to react to the workload, ensure the QoS, and improve the resource utilization by making dynamic resource allocation at runtime, and the expended results verify the efficiency, stability, and scalability of our proposed approach.

The remainder of this paper is organized as follows: Section 2 discusses the specific novel contributions of our work by analyzing recent studies of QoS and resource management. The overview of the resource provisioning control system is shown in Section 3. The details of the adaptive resource provisioning control system is introduced in Section 4. Comparison experiments are presented in Section 5 and the experimental results are analyzed in Section 6. The conclusions are provided in Section 7.

2. RELATED WORKS

Various studies have addressed the problem of resource management in cloud computing systems. Some studies adopted horizontal scaling that making resource allocation by increasing or decreasing the number of VMs, and some studies used vertical scaling to manage resources by increasing or reducing the resource level such as CPU and memory of each VM. Control theory has been increasingly applied for resource management in cloud computing, and many studies have used various control algorithms and architectures for resource allocation. A comparison of related works is shown in Table 1. We categorize these studies into seven types according to the corresponding metrics, approaches, and issues.

We distinguish the previous works based on the seven categories shown in Table 1. The first category is the type of scaling: horizontal scaling or vertical scaling. The second category is related to the studies based on control theory. If the research involves a control theory approach, it is marked "yes"; otherwise, it is marked "no." The third category denotes whether the proposed approach manages resources based on multiple QoS requirements. If the research considers multiple QoS requirements, such as the response time, throughput, and resource utilization, it is marked "yes"; otherwise, it is marked "no". The fourth category reflects whether the method allocates heterogeneous resources. If heterogeneous resources, such as CPU and memory, are considered, the category value is marked "yes"; otherwise, it is marked "no." The fifth category indicates whether the studies consider a resource contention with cohosted services. If the contention is considered, the category value is marked "yes"; otherwise, it is marked "no." The sixth category is related to the mode used to cope with unpredictable changes and is a heuristic or an adaptive mode. The seventh category involves the technologies adopted to manage resource allocation.

Due to the elastic nature of cloud computing systems, dynamic resource management is a difficult problem for resource providers. Many studies [7, 16–19] resolved this problem with auto scaling methods. Most of the current providers [27, 30] such as Amazon and Rightscale use horizontal scaling method. This approach is feasible by changing the number of VMs and easy to implement. However, it takes time to start a new VM, and the process may lead to inefficient resource utilization because it is based on fixed resource units that cannot meet the exact user requirements. As shown in Table 1, most previous horizontal scaling studies did not involve heterogeneous resources management, as horizontal scaling adds or removes fixed resource units. In contrast to horizontal scaling method, vertical scaling method [5, 20–26] manages resources by increasing or reducing the sizes of VMs.

The elastic characteristics of cloud computing provide the opportunity to apply control theory and adaptively automate dynamic resource management. Many studies have designed feedback control loops to cope with unpredictable changes in the workload and disturbances to the system [8–10, 12, 22, 23, 28, 29]. As shown in Table 1, some of the control theory-based studies focused on a single metric that allocated resources according to a single objective, such as minimizing the response time [9, 31, 32]. Other studies allocated a single type of resource, such as CPU [12, 17, 18, 22] or memory [23, 24]. However, these control theory-based studies adopted single-input and single-output (SISO) control architectures which are too simple because focusing on only one type of resource may result in QoS violations, as different services corresponding to various resource requirements may also differ. For example, some services require intensive CPU resources and light memory resources, and some services are contrast. Focusing

Table 1 Comparison of related works.

Ref.	Scaling Type	Control Theoretic	Multiple QoS Requirements	Heterogeneous Resources	Resource Contention	Adaptivity	Technologies
[26, 27]	Horizontal	-	No	No	No	No	VM migration
[15]	Horizontal	-	Yes	Yes	No	Yes	VM migration
[16–18]	Horizontal	-	No	Yes	No	Yes	VM migration and replication
[19, 20]	Vertical	-	No	No	No	No	Aging-based approach/queue length based approach
[21, 33]	Vertical	SISO	No	Yes	No	No	Feedback control/access control
[8, 22, 23, 28]	Vertical	SISO	No	No	No	No	Feedback control
[24]	Vertical	-	No	Yes	Yes	Yes	ARIMA model
[7]	Horizontal	-	No	Yes	No	No	VM placement
[25]	Vertical	-	Yes	No	No	No	Time series
[9, 10]	Vertical	SISO	Yes	No	No	Yes	Feedback control
[29]	Vertical	SISO	No	No	No	Yes	LASSO method
[30]	Vertical	MIMO	No	Yes	No	Yes	Fuzzy control
[12, 13, 31]	Vertical	MIMO	No	No	Yes	Yes	Feedback control
[32]	Vertical	-	No	Yes	No	No	Machine learning
[3]	Vertical	MIMO	No	Yes	Yes	No	Robust control
[4]	Horizontal	SISO	No	No	No	No	VM migration
[5]	Vertical	SISO	No	Yes	No	No	Gray-box feedback control
[14]	Horizontal	MIMO	No	No	No	Yes	Pole placement
[33]	Horizontal	MIMO	Yes	No	No	No	Semi-Markov decision
[34]	Horizontal	MIMO	Yes	No	No	Yes	LPV control
[35, 36]	Horizontal	-	Yes	No	No	Yes	Machine learning
Ours	Vertical	MIMO	Yes	Yes	Yes	Yes	Adaptive PID control

SISO, single-input and single-output; MIMO, multi-input and multi-output; VM, virtual machine; PID, proportion integration differentiation; ARIMA, auto regressive integrated moving average; LASSO, least absolute shrinkage and selection operator; LPV, linear parameter varying.

on one type of resource may lead to insufficient CPU resources but excessive memory resources. In addition, some studies used SISO control to allocate resources by maximizing resource utilization [3–5]. However, maximizing resource utilization does not mean efficiently use of resources, as excessive resource utilization rate may lead to poor QoS. Therefore, it is needed to coordinate ensuring QoS and improving resource utilization to manage resource allocation.

These issues of resource management in cloud computing systems require more advanced MIMO control architectures. As shown in Table 1, recent studies have adopted MIMO control architectures that allocate heterogeneous resources [4, 12–14]. However, these studies allocated CPU and memory resources separately [25, 26, 33, 37], which ignored the interaction between CPU and memory. Unlike these MIMO control studies, we coordinate CPU and memory resources using a MIMO control architecture.

Moreover, because cloud computing is dynamic, nonlinear, and time-varying, uncertainties and disturbances influence the cloud computing system. It is necessary to adaptively resolve issues related to nonlinear uncertainties and disturbances and select the right settings for adjusting parameters at runtime. As shown in Table 1, some studies [13, 35] addressed this issue by modeling the behavior of the workload off-line. However, such approaches can be invalid when unpredictable workload variations occur. Some studies [34, 36] have adopted machine learning methods to cope with changes in the external environment in real time. However, the implementation of machine learning methods takes considerable amount of time. If the disturbances or requirements frequently change, the decision-making process cannot keep up with the changes in time. Some previous studies used [12, 14, 38] recursive least square (RLS) for system identification to adjust the parameters of the MIMO controller. However, this approach is extremely complex and may even fail to adjust the parameters of the controller.

Consequently, as shown in Table 1, the proposed approach in this paper using MIMO control can allocate heterogeneous resources according to multiple QoS requirements and improve resource utilization while ensure QoS. Moreover, the proposed approach adopts RBF neural network to handle external and internal disturbances in real time.

3. RESOURCE PROVISIONING SYSTEM OVERVIEW

An overview of the resource provisioning system is shown in Figure 1. As shown in Figure 1, services run on the VM. The service sensor monitors the current QoS and provides feedback to the control system. At the same time, the system sensor monitors the resource utilization, such as CPU and memory utilization, and feeds back them to the control system. Then, the control system predicts the resource allocation for the VM according to the QoS requirements and the feedback provided by the sensors. Then, the resource allocation is executed, and VM resources, such as CPU and memory resources, are reconfigured. The closed loop cyclically runs in real time. For clarity, various system definitions are given as follows:

- $rt(k)$: Current average service response time during one minute in the kth moment measured in seconds (s).
- $rt^*(k)$: Required average service response time during one minute in the kth moment, or the reference service response time, measured in seconds (s).
- $th(k)$: Current throughput of the service in the kth moment measured in responses per minute (responses/minute).
- $th^*(k)$: Required throughput of the service, or the reference throughput in the kth moment, measured in responses per minute (responses/minute).

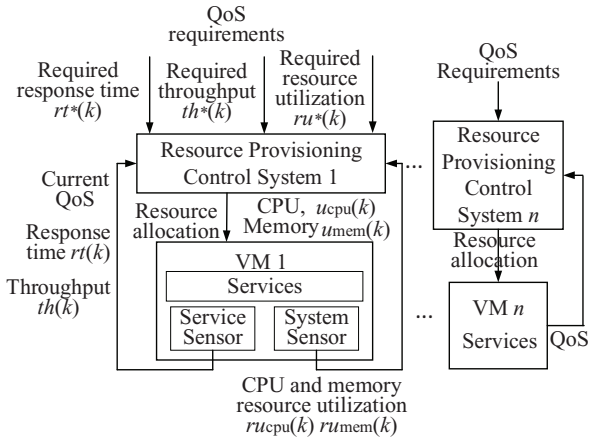

Figure 1 | Resource provisioning system overview.

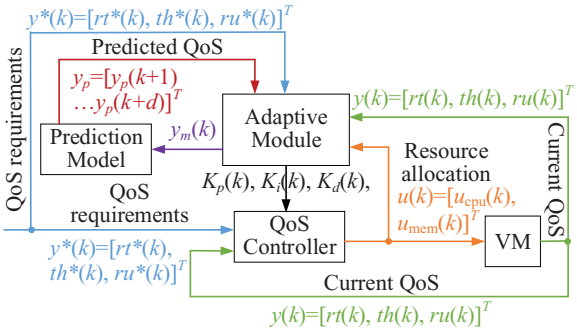

Figure 2 | Overview of adaptive resource provisioning control system.

- $ru_{cpu}(k)$: Current CPU resource utilization rate in the kth moment, or the percent utilization of the allocated resource (%).
- $ru_{mem}(k)$: Current memory resource utilization rate in the kth moment, or the percent utilization of the allocated resource (%).
- $ru_{cpu}{}^{*}(k)$: Required CPU resource utilization rate in the kth moment (%).
- $ru_{mem}{}^{*}(k)$: Required memory resource utilization rate in the kth moment (%).
- $u_{cpu}(k)$: CPU resource allocation in the kth moment measured in cores (cores).
- $u_{mem}(k)$: Memory resource allocation in the kth moment measured in gigabytes (GBs).

As shown in Figure 1, to simultaneously improve resource utilization and ensure QoS, the control system periodically adjusts the resource allocation such as CPU and memory according to the QoS requirements and the feedback current QoS provided by the sensors. The closed-loop method adopts the QoS model to adjust the resource allocation strategy for the VM by increasing or decreasing the number of CPU and reconfiguring the memory.

4. ADAPTIVE RESOURCE PROVISIONING CONTROL SYSTEM

The feedback mechanism of control theory provides ability for dealing with unpredictable changes, uncertainties, and disturbances in cloud computing systems. Although feedback control has been developed in cloud computing systems [8–10, 12, 22, 23, 28, 29], the previous control algorithms were designed according to the domain knowledge instead of being adaptive control model that considered the dynamic and nonlinear nature of cloud computing systems. In this paper, we propose an adaptive resource provisioning control system using adaptive control. An overview of the adaptive resource provisioning control system is shown in Figure 2.

As shown in Figure 2, the control system consists of three parts: the QoS controller, adaptive module, and prediction model. Due to the nonlinear and complex nature of cloud computing system, it is difficult to obtain the relationship between the QoS and resource allocation; therefore, the QoS model is difficult to build. The adaptive control does not require knowledge of the specific relationships among the inputs and outputs of the model, and the RBF neural network adaptively adjusts the K_p, K_i, and K_d parameters according to the QoS requirements. In addition, because the delay exists in the system, the controller requires some time to affect the QoS. Therefore, the prediction model is introduced to address the delay, and the RBF neural network is adjusted by eliminating the error between the predicted QoS and the actual QoS. The QoS controller based on proportion integration differentiation (PID) control, adaptive module based on RBF neural network, and prediction model will be introduced in the next section.

4.1. QoS Controller

PID control is a classic control algorithm widely used in industrial systems [39]. With the development of cloud computing, control theory has been increasingly applied to cloud computing systems. PID control has attracted considerable attention due to its simplicity, robustness, and reliability. Because cloud computing systems are nonlinear and time-varying environments, it is difficult to build accurate performance models. The relationship between QoS and resource allocation is described as in Equation 1:

$$y(k) = f\left(y(k-1)\cdots y\left(k-n_y\right), u(k-d)\cdots u(k-d-n_u)\right), \quad (1)$$

where d is the pure delay in the control system; $y(k)$ is the current QoS and $y(k) = [rt(k), th(k), ru(k)]^T$, where $ru(k) = [ru_{cpu}(k), ru_{mem}(k)]^T$; $u(k)$ is the current resource allocation and $u(k) = [u_{cpu}(k), u_{mem}(k)]^T$; $y(k-1)\cdots y(k-n_y)$ is the QoS in the $(k-1)$th$\cdots(k-n_y)$th moment; $u(k-d)\cdots u(k-d-n_u)$ is the resource allocation in the $(k-d)$th$\cdots(k-d-n_u)$th moment; and $f(*)$ is the nonlinear relationship between $y(k)$ and $u(k)$.

In this paper, we adopt incremental PID control, and the incremental resource allocation is described as follows:

$$\Delta u(k) = K_p(k-1)x_1(k) + K_i(k-1)x_2(k) + K_d(k-1)x_3(k) \quad (2)$$

where K_p, K_i, and K_d are the proportion, integration, and differentiation parameters of the PID controller, respectively. Additionally, $x_1(k)$, $x_2(k)$, and $x_3(k)$ can be expressed as follows:

$$\begin{cases} x_1(k) = e(k) - e(k-1) \\ x_2(k) = e(k) \\ x_3(k) = e(k) - 2e(k-1) + e(k-2) \\ e(k) = y^*(k) - y(k) \end{cases} \tag{3}$$

where $y^*(k) = [rt^*(k), th^*(k), ru^*(k)]^T$ represents the QoS requirements, or the reference QoS of the controller. Therefore, the resource allocation equation is derived as follows:

$$u(k) = u(k-1) + \Delta u(k) \tag{4}$$

The K_p, K_i, and K_d parameters are adjusted by the RBF neural network, and the performance index is as follows:

$$E(k) = \frac{1}{2}\left(y^*(k) - y(k)\right)^2 \tag{5}$$

We use the gradient descent method to adjust the PID parameters, and K_p, K_i, and K_d are adjusted as follows:

$$\begin{cases} K_p(k) = K_p(k-1) + \Delta K_p(k) \\ K_i(k) = K_i(k-1) + \Delta K_i(k) \\ K_d(k) = K_d(k-1) + \Delta K_d(k) \end{cases} \tag{6}$$

Moreover, $\Delta K_P(k)$, $\Delta K_i(k)$, and $\Delta K_d(k)$ and are calculated as follows:

$$\begin{aligned} \Delta K_p(k) &= -\eta_p \frac{\partial E(k)}{\partial K_p(k-1)} = \eta_p \frac{\partial E(k)}{\partial y(k)} \frac{\partial y(k)}{\partial u(k)} \frac{\partial u(k)}{\partial K_p(k-1)} \\ &= \eta_p e(k) \frac{\partial y(k)}{\partial u(k)} x_1(k) \end{aligned}$$

$$\begin{aligned} \Delta K_i(k) &= -\eta_i \frac{\partial E(k)}{\partial K_i(k-1)} = \eta_i \frac{\partial E(k)}{\partial y(k)} \frac{\partial y(k)}{\partial u(k)} \frac{\partial u(k)}{\partial K_i(k-1)} \\ &= \eta_i e(k) \frac{\partial y(k)}{\partial u(k)} x_2(k) \end{aligned} \tag{7}$$

$$\begin{aligned} \Delta K_d(k) &= -\eta_d \frac{\partial E(k)}{\partial K_d(k-1)} = \eta_d \frac{\partial E(k)}{\partial y(k)} \frac{\partial y(k)}{\partial u(k)} \frac{\partial u(k)}{\partial K_d(k-1)} \\ &= \eta_d e(k) \frac{\partial y(k)}{\partial u(k)} x_3(k) \end{aligned}$$

where η_p, η_i, and η_d are the learning rates, and $\frac{\partial y(k)}{\partial u(k)}$ cannot be directly obtained; therefore, this value is determined with the RBF neural network.

4.2. Adaptive Module

Since coupling and interactions exist among the MIMO control system components in the cloud computing environment, the control model and parameters are difficult to obtain. Traditional PID control depends on an accurate model, but the uncertainties of the cloud computing system often make the control effect of traditional PID control poor. RBF neural networks have been increasingly applied in computing systems due to their nonlinear and self-learning capabilities [40]. RBF neural network is a kind of effective feedforward network, which is local approximation that highly improves the convergence speed and avoids local minimum [15]. Therefore, we use it to adjust the K_p, K_i, and K_d parameters in cloud computing system.

The structure of the applied RBF neural network is shown in Figure 3. The network is a feedforward neural network with three layers: an input layer, a hidden layer, and an output layer. In Figure 3, $x_{\mathrm{RBF}1}(k) \cdots x_{\mathrm{RBF}n}(k)$ are the inputs of the network, $H_1(X(k)) \cdots H_m(X(k))$ are the RBFs of the hidden layer, $w_1 \cdots w_m$ are the neuron weights, and $y_m(k)$ is the output of the network.

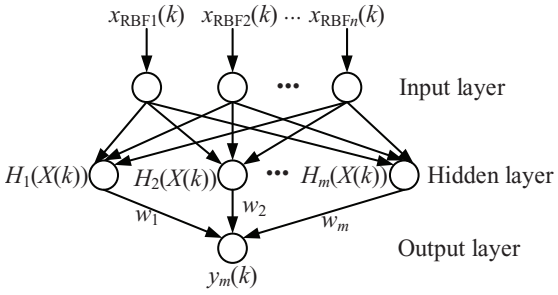

Figure 3 Applied radial basis function (RBF) neural network structure.

4.2.1. Input layer

To obtain the nonlinear relationship between the QoS and resource allocation in Equation 1, we build the RBF neural network model as follows:

$$y_m(k) = f_m\left(y(k-1) \cdots y\left(k-n_y\right), u(k-d) \cdots u(k-d-n_u)\right) \tag{8}$$

where $y_m(k) = [rt_m(k), th_m(k), ru_m(k)]^T$ is the estimated QoS and $f_m(*)$ is the estimated nonlinear relationship between $y(k)$ and $u(k)$. Then, the output of the input layer is $X = [x_{\mathrm{RBF}1}(k) \cdots x_{\mathrm{RBF}n}(k)]^T = [y(k-1) \cdots y(k-n_y), u(k-d) \cdots u(k-d-n_u)]^T$.

4.2.2. Hidden layer

We adopt a Gaussian function [41] as the inspirit function of the hidden layer shown in Equation 9.

$$H_j(X_{RBF}) = \exp\left(-\frac{\|X_{RBF} - C_j\|^2}{2b_j^2}\right), \quad j = 1, 2 \cdots m \tag{9}$$

where $c_j = [c_{j1}, c_{j2} \cdots c_{jn}]^T$ is the center vector of the jth node and b_j is the basis width, which is the width of basis function around the center.

4.2.3. Output layer

The output of the output layer is $y_m(k)$, which is estimated by the network, as shown in Equation 10.

$$y_m(k) = \sum_{j=1}^{m} w_j(k-1) H_j(X_{RBF}(k)) \tag{10}$$

where $w_j(k-1)$ is the weight of the jth node from the hidden neuron to the output neuron.

To obtain the estimated QoS $y_m(k)$, we train the network by narrowing the gap between $y_m(k)$ and $y(k)$. The performance index of the network is given Equation 11.

$$E_{RBF}(k) = \frac{1}{2}\left(y(k) - y_m(k)\right)^2 \tag{11}$$

For clarity, we let $e_{\mathrm{RBF}}(k) = y(k) - y_m(k)$ and use a gradient descent algorithm to calculate the parameters of the neural network. The learning of $w_j(k)$, $b_j(k)$, and $c_{ji}(k)$ is introduced as follows:

- Weight $w_j(k)$ learning algorithm

$$w_j(k) = w_j(k-1) + \Delta w_j(k) \tag{12}$$

where $\Delta w_j(k)$ is calculated as follows and η is the learning rate.

$$\begin{aligned}\Delta w_j(k) &= -\eta\frac{\partial E_{RBF}(k)}{\partial w_j(k-1)} \\ &= -\eta\frac{\partial E_{RBF}(k)}{\partial e_{RBF}(k)}\frac{\partial e_{RBF}(k)}{\partial y_m(k)}\frac{\partial y_m(k)}{\partial w_j(k-1)} \\ &= \eta e_{RBF}(k) H_j\left(X(k)\right)\end{aligned} \tag{13}$$

- Basis width $b_j(k)$ learning algorithm

$$b_j(k) = b_j(k-1) + \Delta b_j(k) \tag{14}$$

$$\Delta b_j(k) = -\eta\frac{\partial E_{RBF}(k)}{\partial b_{j-1}(k)} = -\eta\frac{\partial E_{RBF}(k)}{\partial H_j(X(k))}\frac{\partial H_j(X(k))}{\partial b_{j-1}(k)} \tag{15}$$

where $\frac{\partial E_{RBF}(k)}{\partial H_j(X(k))}$ and $\frac{\partial H_j(X(k))}{\partial b_j(k-1)}$ are calculated as follow:

$$\frac{\partial E_{RBF}(k)}{\partial H_j(X(k))} = \frac{\partial E_{RBF}(k)}{\partial y_m(k)}\frac{\partial y_m(k)}{\partial H_j(X(k))} = -e_{RBF}(k)\, w_j(k-1) \tag{16}$$

$$\frac{\partial H_j(X(k))}{\partial b_j(k-1)} = H_j(X(k))\frac{\|X(k) - c_j(k-1)\|^2}{b_j^3(k-1)} \tag{17}$$

- Center vector $\boldsymbol{c}_j$ learning algorithm

$$c_{ji}(k) = c_{ji}(k-1) + \Delta c_{ji}(k) \tag{18}$$

where $j = 1, 2\cdots m$, $I = 1, 2\cdots n$, $\Delta c_{\mathrm{ji}}(k)$ is calculated as follows.

$$\begin{aligned}\Delta c_{ji}(k) &= -\eta\frac{\partial E_{RBF}(k)}{\partial c_{ji}(k-1)} \\ &= -\eta\frac{\partial E_{RBF}(k)}{\partial H_j\left(X(k)\right)}\frac{\partial H_j\left(X(k)\right)}{\partial c_{ji}(k-1)} \\ &= \eta e_{RBF}(k) w_j(k-1) H_j\left(X(k)\right)\frac{x_i(k) - c_{ji}(k)}{b_j^2(k-1)}\end{aligned} \tag{19}$$

As shown in Eqs. 2–7, to derive the resource allocation $u(k)$, we must obtain $\frac{\partial y(k)}{\partial u(k)}$, and we use the output of the neural network $y_m(k)$ instead of $y(k)$ to calculate $u(k)$. Therefore, $\frac{\partial y(k)}{\partial u(k)}$ is calculated as follows:

$$\frac{\partial y(k)}{\partial u(k)} \approx \frac{\partial y_m(k)}{\partial u(k)} \tag{20}$$

Additionally, according to Equation 10, $\frac{\partial y_m(k)}{\partial u(k)}$ is calculated as follows:

$$\begin{aligned}\frac{\partial y_m(k)}{\partial u(k)} &= \sum_{j=1}^{m} w_j(k-1)\frac{\partial H_j\left(X(k)\right)}{\partial u(k)} \\ &= \sum_{j=1}^{m} w_j(k-1) H_j\left(X(k)\right)\frac{c_{j(n_y+1)}(k-1) - u(k)}{b_j^2(k-1)}\end{aligned} \tag{21}$$

We substitute this expression into Eqs. 7 and 2 to derive the resource allocation $u(k)$.

4.3. Prediction Model

Because the delay exists in the system, the current control requires some time to affect the QoS. Therefore, it is necessary to introduce predictive control, which predicts the future resource demands by narrowing the gap between the predictive QoS and the current actual QoS. We adopt the RBF neural network to make this prediction and adjust the parameters of the neural network, such as $w_j(k)$, $b_j(k)$, and $c_{ji}(k)$, by minimizing the error between the predicted QoS $y_p(k)$ and actual QoS $y(k)$. Analogously, we use the neural network to adjust the PID parameters by minimizing the error between the predictive QoS and the QoS requirements. Unlike the expression in Equation 5, we use the error $y^*(k+N) - y(k+N)$ instead of the error $y^*(k) - y(k)$. Therefore, the new performance index is as follows:

$$E_p = \frac{1}{2}\left(y^*(k+N) - y(k+N)\right)^2 \tag{22}$$

where N is the prediction length. Then, the PID parameters are adjusted with the gradient descent algorithm, as in Eqs. 6 and 7, and are shown as follows:

$$\begin{cases}\Delta K_p(k) = \eta_p\left(y^*(k+N) - y(k+N)\right)\dfrac{\partial y(k+N)}{\partial u(k)}x_1(k) \\ \Delta K_i(k) = \eta_i\left(y^*(k+N) - y(k+N)\right)\dfrac{\partial y(k+N)}{\partial u(k)}x_2(k) \\ \Delta K_d(k) = \eta_d\left(y^*(k+N) - y(k+N)\right)\dfrac{\partial y(k+N)}{\partial u(k)}x_3(k)\end{cases} \tag{23}$$

where $x_1(k)$, $x_2(k)$, and $x_3(k)$ are defined in Equation 3. Analogously, $\frac{\partial y(k+N)}{\partial u(k)}$ cannot be directly derived; thus, we adopt the RBF neural network to calculate this term.

First, we establish the model for the RBF neural network, as in Equation 8. Then, the parameters of the network, such as $w_j(k)$, $b_j(k)$, and $c_{ji}(k)$, are adjusted by the gradient algorithm in Eqs. 9–19, which makes the output of the neural network $y_m(k)$ approximately

equal to the actual QoS $y(k)$. Then, we derive the prediction model based on the output of the RBF neural network as follows:

$$\begin{aligned} y_m(k+1) &= f\big(y(k), y(k-1)\cdots y(k-n+1), \\ &\quad u(k-d+1)\cdots u(k-d-m+1)\big) \\ y_m(k+2) &= f\big(y(k+1), y(k)\cdots y(k-n+2), \\ &\quad u(k-d+2)\cdots u(k-d-m+2)\big) \\ &\vdots \\ y_m(k+N) &= f\big(y(k+N-1), y(k+N-2)\cdots y(k+N-n), \\ &\quad u(k)\cdots u(k-m)\big) \end{aligned} \tag{24}$$

where $y_m(k+i)$, $I = 1, 2\cdots N$ is the prediction model. As shown in Equation 24, these multistep prediction models are established based on $y_m(k)$. However, if model mismatch occurs, error accumulation can also occur. Therefore, online correction is needed to improve the accuracy of the prediction. The corrected predictive output is as follows:

$$y_p(k+N) = y_m(k+N) + \big[y(k) - y_m(k)\big] \tag{25}$$

Similar to Equation 20, to derive $\frac{\partial y(k+N)}{\partial u(k)}$, we use the predictive output of the neural network $y_p(k+N)$ instead of $y(k+N)$ as follows:

$$\frac{\partial y(k+N)}{\partial u(k)} \approx \frac{\partial y_p(k+N)}{\partial u(k)} \tag{26}$$

Using this equation, we can obtain the resource allocation $u(k)$.

The multistep predictive control method considers the delay in the system, and the PID control overcomes the nonlinear uncertainties. Therefore, we use the adaptive prediction PID control based on RBF neural network to simultaneously resolve the delay and nonlinear uncertainty issues. The method can predict and correct the QoS online, which improves the control accuracy. In addition, the adaptive PID control only adjusts K_p, K_i, and K_d based on unknown QoS and resource allocation requirements, which simplifies the design of the controller.

5. EXPERIMENTAL SETUPS

The experiments are conducted on two physical machines (PMs) and several VMs. An overview of the experimental environment is shown in Figure 4. As shown in Figure 4, the experimental environment consists of two sides: the service side and the controller side.

5.1. Initial Setups

On the service side, the VMs are implemented on PM equipped with 36 CPU cores and 72 GB of memory. As shown in Figure 4, the resource allocator executes the resource allocation provided by the controllers, which is developed by Python 3.4. This approach requires a VM equipped with 4 CPU cores and 8 GB of memory. Services are distributed on VMs, and the initial configuration of each VM includes 1 CPU core and 2 GB of memory. Each VM has a system sensor and QoS sensor developed by Python 3.4. The system sensor monitors the CPU and memory utilization of the VM and sends that information to the resource allocation controller. The QoS sensor monitors the QoS, such as the response time and throughput, and sends feedback to the controller.

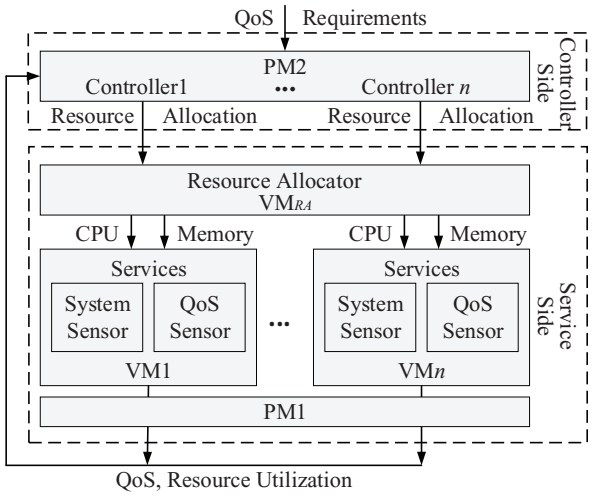

Figure 4 Overview of the experimental environment.

Table 2 Experiment initial setups.

Module	PM/VM	OS	Programming Language	CPU (Core)	MEM (GB)
Resource allocator	PM1/VM $_{RA}$	CentOS 7	Python 3.4	4	8
Resource allocation controller	PM2	CentOS 7	Python 3.4	4	8
Services	PM1/VM$_{1\cdots n}$	CentOS 7	OpenStack Icehouse API	2 (initial)	4 (initial)

PM, physical machine, VM, virtual machine.

On the controller side, the controllers run on a PM equipped with 4 CPU cores and 8 GB of memory. Each controller is designed for each VM according to the feedback provided by the sensors and QoS requirements. Then, the controllers determine the resource allocation for each VM and send this information to the resource allocator. The initial experimental setups are shown in Table 2.

In this experiment, we use the real-world workload of the FIFA 1998 soccer world cup website [42] on June 14. The workload is shown in Figure 5.

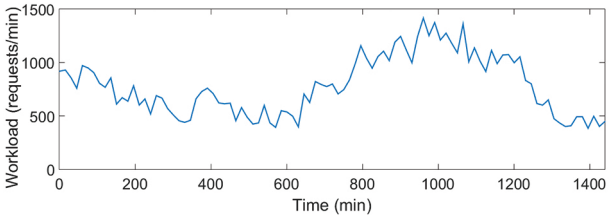

Figure 5 Workload.

5.2. Simulations

To evaluate the efficiency of the controller, we perform some simulations with MATLAB 2015b. The workload traces are the same in the experiments. For the MIMO controller, using one of the outputs as an example, we set the expected output to 1. We compare the PID control algorithm based on RBF neural network algorithm with the proposed predictive PID control method based on RBF neural network algorithm. The simulation results are shown in Figure 6.

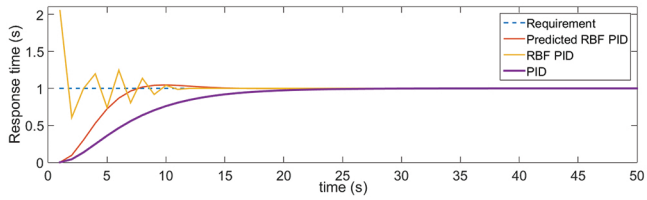

Figure 6 | Adaptive control system simulation results.

As shown in Figure 6, all three control algorithms meet the control requirements. In addition, the PID algorithm needs 25 seconds to converge to the expected output value, the PID method based on RBF neural network needs 10 seconds, and the proposed predictive PID method based on RBF approach needs 15 seconds. However, although the PID based on RBF neural network algorithm converges fastest, it is approximately as fast as the proposed predictive PID RBF algorithm. In addition, the proposed method is more stable than the PID RBF algorithm. Therefore, the proposed predictive PID method based on RBF neural network control algorithm is valid and performs better than the other algorithms.

6. EXPERIMENTAL RESULTS AND ANALYSIS

To evaluate the efficiency of the proposed MIMO control based on RBF neural network approach, and according to the comparison in Section 2 (Related Works), we compare a horizontal scaling approach [7], a vertical scaling approach based on control theory [5], and the proposed approach. Hereafter, we call the horizontal approach "Horizontal" for short, the vertical scaling approach based on control theory "SISO" for short, and the proposed approach "MIMO" for short.

In this experiment, we analyze five aspects of the results, namely (1) the response time, (2) the throughput, (3) the CPU resource utilization rate, and (4) the memory resource utilization rate to evaluate the advantages and disadvantages of each approach.

6.1. Adaptive Control System Evaluation

6.1.1. Response time

Because the response time is one of the most important indicators of QoS, we analyze the response time to evaluate the efficiency for managing QoS of the proposed approach. The service requirements demand a response time of less than 0.8 seconds. The response time results of the three approaches are shown in Figure 7.

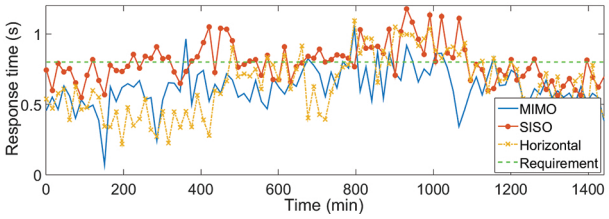

Figure 7 | Response time results.

As shown in Figure 7, the response time of the MIMO approach is approximately 0.8 seconds. Because the MIMO controller theoretically guarantees high control accuracy and system stability, it provides the optimal QoS based on the requirements and keeps the system stable when the workload fluctuates.

For the SISO approach, the response time is longer than that of the MIMO and Horizontal approaches, especially during the peak workload, when the response time is greater than 0.8 seconds. The SISO approach focuses on maximizing resource utilization rate, and resource over-utilization leads to a poor QoS as a result of resource contention.

The response time of the Horizontal approach fluctuates because it uses one VM as a scaling unit. This approach is a coarse-grained that cannot meet the specific service requirements. For example, during the peak workload stage, the response time is greater than 0.8 seconds, which indicates requirement violations. Although sufficient CPU resources are allocated in this approach, the memory resources are inadequate, which leads to a poor QoS during the peak workload stage.

6.1.2. CPU resource utilization rate

In this experiment, we choose 70% as the desired CPU resource utilization rate [43]. CPU resource utilization rate is the percentage of utilized CPU to allocated CPU. We analyze the CPU resource utilization rate to evaluate the efficiency of the proposed controller and whether it enables the efficient use of resources. The CPU resource utilization rate results for the three approaches are shown in Figure 8.

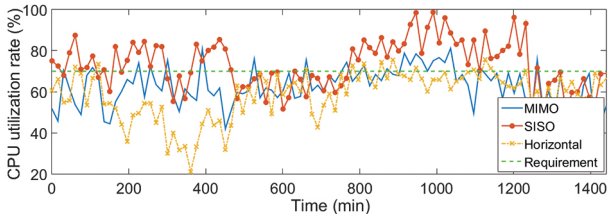

Figure 8 | CPU resource utilization rate results.

As shown in Figure 8, the CPU resource utilization rate of the MIMO approach varies from approximately 50% to 70% at all times, which highlights the stability of the system. In addition, as shown by the response time results of the MIMO approach in Figure 7, the response time is approximately 0.6 seconds to 0.8 s during the experiment, and this range meets the service requirements because the controller determines the resource allocation according to the

workload and service requirements. The feedback mechanism guarantees system stability, and this is fine-grained approach provides a rough response time relative to the required maximum allowable response time.

The CPU resource utilization rate of the SISO approach is higher than that of the MIMO and Horizontal approaches, reaching levels greater than 90% during the peak workload. Because the SISO approach adopts a single-input and single-output architecture, it focuses on maximizing resource utilization rate. However, as shown in Figure 7, the response time of the SISO approach is greater than 1 second, especially during the peak workload period. This result reflects a poor QoS and requirement violations. In this case, the resources are over-utilized, which leads to resource contention. Therefore, although the resource utilization rate of the SISO approach performs better than other approaches, it cannot ensure a satisfactory QoS.

For Horizontal approach, the CPU resource utilization is lower than other approaches some time, and it is nearly the same with others some time. This is because the Horizontal approach uses one VM (2 CPU cores and 4 GB of memory) as a scaling unit, which may result in excessive resource scaling. In addition, the Horizontal approach adjusts the resource allocation by changing the number of VMs, which is coarse-grained approach that cannot meet the specific service requirements. Therefore, although the response time is better than that of other approaches, the resource utilization rate indicates that it does not efficiently use resources.

6.1.3. Memory resource utilization rate

In this experiment, we select 70% as the desired memory resource utilization rate. Memory resource utilization rate is the percentage of the utilized memory to allocated memory. We analyze the memory resource utilization rate not only to evaluate whether the approach improves the level of resource utilization but also to validate whether it meets the specific service requirements. The memory resource utilization rate results for the three approaches are shown in Figure 9.

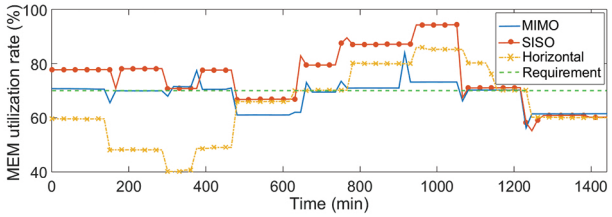

Figure 9 Memory resource utilization rate results.

As shown in Figure 9, the memory resource utilization rate of the MIMO approach ranges from 60% to 70% during the entire experiment, which indicates high system stability. Because the MIMO controller considers multiple metrics and coordinates CPU and memory resources according to multiple requirements, such as the response time, CPU resource utilization rate, memory resource utilization rate, and workload. For example, some services require massive memory resources but few CPU resources, and vice versa. Poor QoS may be caused by a lack of memory resources, yet the CPU resources may be excessive. Therefore, the proposed approach enables the coordination of CPU and memory resources according to the specific service requirements to ensure a satisfactory QoS.

The memory resource utilization rate of the SISO approach is obviously higher than that of the other approaches, especially during the peak workload period, with values greater than 90%. The SISO approach aims to maximize the resource utilization rate and ignores the coordinated allocation of CPU and memory resources. In some cases, this approach allocates the same amount of CPU resources as other approaches, but the response time is still slower than those of other approaches because the lack of memory resources when the resource utilization rate is greater than 90%. Therefore, although the resource utilization rate is better than that of the MIMO and Horizontal approaches, the memory resources are overutilized during the peak workload stage.

For the Horizontal approach, the memory resource utilization rate is lower than that of the other approaches, but during the peak workload period, memory resource utilization rate is greater than 90%, which indicates resource overutilization. However, the CPU resource utilization rate is approximately 60%, as shown in Figure 8, which indicates that the CPU resources are adequate. Horizontal scaling method uses one VM as a scaling unit, and this coarse-grained approach ignores the coordination of CPU and memory resources, which leads to CPU over-allocation and memory under-allocation. Therefore, as a result of the coarse-grained scaling, the response time of the Horizontal approach shown in Figure 7 is worse than that of the MIMO approach during the peak workload stage.

6.1.4. Throughput

The throughput results are shown in Figure 10. Because the throughput is one of the most important factors related to the QoS, we analyze the throughput results of the compared approaches to evaluate whether they can cope with fluctuations in the workload.

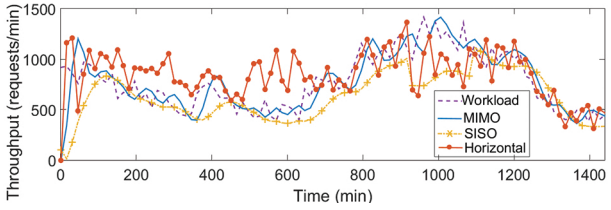

Figure 10 Throughput results.

As shown in Figure 10, the throughput results of the MIMO approach change with the workload fluctuations, and this approach provides the closet throughput to the workload compared with the other approaches, reflecting the adaptability of the control system. Because the feedback mechanism of the controller can precisely adjust the resource allocation, an accurate number of requests per minute is provided relative to the workload. In addition, the throughput generally keeps up with the workload because the RBF neural network adaptively adjusts the resource allocation in real time. Only focusing on the response time cannot ensure that QoS requirements are met. For example, in some cases, the average service response time may be only 0.2 seconds, but during that time,

some requests may not receive a response. The requests with no service response fails the QoS requirements. Therefore, focusing on single metric of the QoS such as response time cannot ensure the QoS. The MIMO controller provides a MIMO architecture to ensure that multiple factors, such as the response time and throughput of the QoS, are considered.

Because the SISO approach focuses on maximizing resource utilization rate, the throughput is less than that for the MIMO approach. As shown in Figure 10, due to the feedback loop of the SISO controller, the approach reacts to fluctuations in the workload. However, as a result of maximizing the resource utilization rate, resource over-utilization leads to poor throughput, especially during the peak workload stage, when there are insufficient resources to provide the required throughput.

For the horizontal approach, the throughput performs better than that for the MIMO and SISO approaches during the experiment. However, as shown in Figure 10, this approach provides excessive numbers of requests per minute relative to the workload, which indicates a waste of resources. Horizontal scaling method provides more resources than often needed because it uses one VM as a scaling unit. For example, during the beginning hours of the experiment, the services require 4 CPU cores and 14 GB of memory. However, the horizontal scaling method uses one VM (2 CPU cores and 8 GB of memory) as a scaling unit which provides 4 CPU cores and 16 GB of memory. Thus, the excessive 2 GB of memory is underutilized. Therefore, although the Horizontal approach yields a better QoS than the other approaches, it inefficiently uses resources.

6.2. Scalability Evaluation

Due to the shared nature of cloud computing, interference exists among VMs that share the same resource pool. Neglecting this interference may lead to the burst of the QoS. Therefore, we upscale the experiment to verify whether the proposed control system can cope with such types of interference and evaluate the efficiency, stability, and scalability of the control system.

We expand the experiment from one VM to three VMs, and we use three real-world workloads from the FIFA 1998 soccer world cup website [42] (shown in Figure 11). We also analyze the (1) response time, (2) CPU resource utilization, (3) memory resource utilization, and (4) throughput to discuss the efficiency and scalability of the adaptive MIMO control system.

6.2.1. Response time

The response time of the expanded three VMs are shown in Figure 12. The service requirements require the response time need to be less than 0.8 second, 0.6 second, and 0.5 second.

As shown in Figure 12, all the response time of the three VMs meet the requirements, which are under 0.8 second, 0.6 second, and 0.5 second, respectively. And the response time is closed to the requirements, which indicates that the control system still enables to provide the roughly the same QoS with the service requirements. Besides, the response time bursts not violent, which finds out the stability of the expanded system.

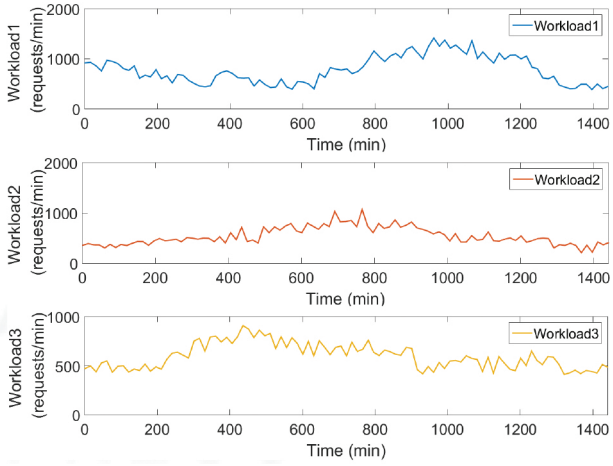

Figure 11 | Workloads.

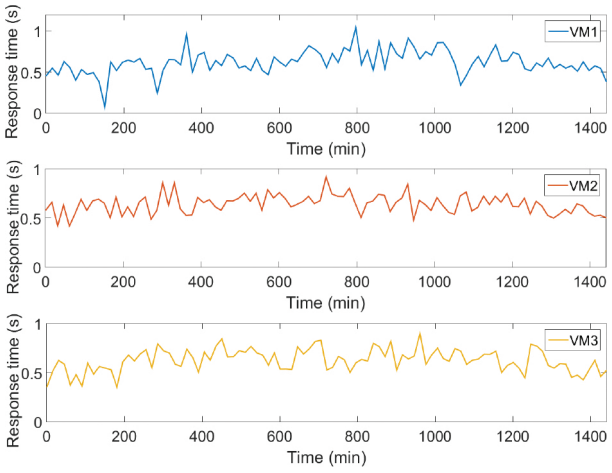

Figure 12 | Response time results of scaled virtual machines (VMs).

6.2.2. CPU resource utilization rate

The resource utilization results of the expanded three VMs are shown in Figure 13. We also choose 70% as the desired CPU resource utilization level of the three VMs.

As shown in Figure 13, the CPU resource utilization results of the expanded VMs keeps around 70% during the experiment, which indicates the stability of the system. However, it bursts and some of the resource utilization is over 80% during the peak workload. This because the VMs share one PM, where there exists interferences among the VMs. Ignoring the interferences may lead to poor QoS even requirements violation. Although the CPU resource utilization bursts, it still roughly meet the desired resource utilization level, thus the control system is still efficient and stable for the expanded system.

6.2.3. Memory resource utilization rate

Similar to the CPU resource utilization, the memory resource utilization results of the three VMs are shown in Figure 14. We also choose 70% as the desired memory resource utilization level of the expanded VMs.

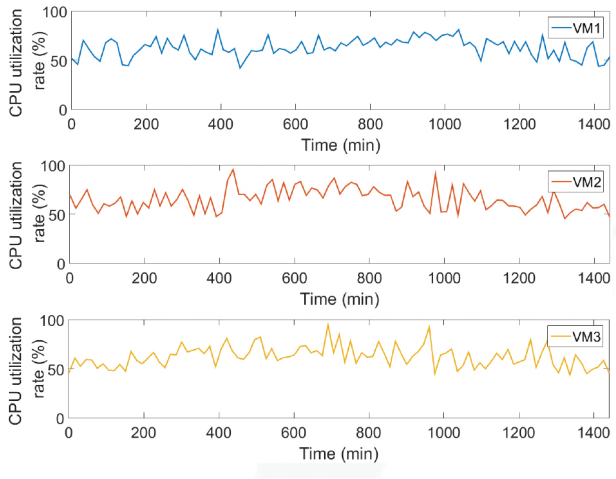

Figure 13 | CPU resource utilization rate results of scaled VMs.

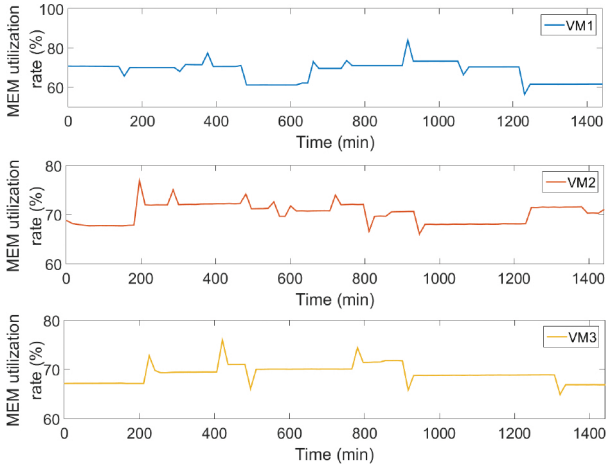

Figure 14 | Memory resource utilization rate results of scaled VMs.

The memory resource utilization of the three VMs is about 70% during the experiment. In contrast to the CPU resource utilization, the memory resource utilization does not burst during the fluctuations of the workload. Because the controller allocates adequate memory resources to the VMs. The memory resource utilization also demonstrates the stability of the control system.

6.2.4. Throughput

The throughput results of the expanded VMs are shown in Figure 15. As the interferences exist in the cloud computing system, we analyze throughput to evaluate whether it still enables to provide the precise number of requests relative to the workloads.

As shown in Figure 15, the throughput results of the VMs vary with the dynamic workload during the experiment, which indicates that the number of requests per minute is roughly the same as that of the workload. However, as a result of the interference, the throughput fluctuates when the workload changes. For example, the throughput of VM1 fluctuates when the workloads of VM2 and VM3 change.

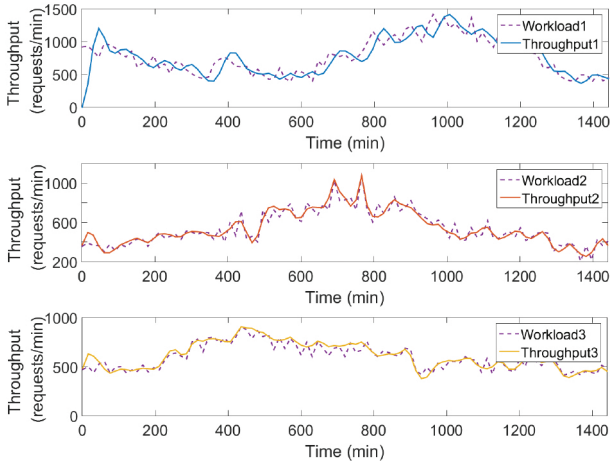

Figure 15 | Throughput results of scaled virtual machines (VMs).

Despite it fluctuates, the throughput is similar to the workload, and the maximum allowable requirement is met. Because the RBF neural network copes with the nonlinear disturbances and adjusts the parameters of the controller in real time. Therefore, the throughput results verify that the control system can adaptively cope with interference among VMs.

All of above results, it can be seen from the response time results and throughput results that the proposed adaptive MIMO control approach enables to react to the dynamic workload and ensure the QoS. The CPU and memory resource utilization rate results indicate the proposed approach enables to keep the resource utilization rate within a reasonable range.

7. CONCLUSIONS

In this paper, we propose an adaptive control system for resource allocation in cloud computing systems. The adaptive controller adopts a MIMO control architecture and uses PID control based on RBF neural network to adaptively perform resource allocation in real time.

The PID control allows the control system to provide an accurate QoS relative to the requirements. The RBF neural network ensures that the system rapidly reacts to the dynamic workload. In this method, the CPU and memory resource allocation are coordinated according to the dynamic workload and service requirements.

The experimental results highlight the efficiency, stability, and adaptability of the control system, which can improve resource utilization rate to approximately 70%–80%, yield a satisfactory QoS, react to dynamic workload changes, and cope with interferences in cloud computing systems.

ACKNOWLEDGMENTS

This work was partially supported by the Natural Science Foundation of China (Grant No. 61772055, 61872169).

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, A view of cloud computing, Commun. ACM. 53(4) (2010), 50–58.

[2] Q. Zhang, L. Cheng, R. Boutaba, Cloud computing: state-of-the-art and research challenges, J. Internet Serv. Appl. 1(1) (2010), 7–18.

[3] R.P. Pothukuchi, A. Ansari, P. Voulgaris, J. Torrellas, Using multiple input, multiple output formal control to maximize resource efficiency in architectures, in Proceeding ACM/IEEE International Symposium on Computer Architecture, Seoul, 2016, pp. 658–670.

[4] A. Wolke, M. Bichler, T. Setzer, Planning vs. dynamic control: resource allocation in corporate clouds, IEEE Trans. Cloud Comput. 4(3) (2016), 322–335.

[5] L. Baresi, S. Guinea, A. Leva, G. Quattrocchi, A discrete-time feedback controller for containerized cloud applications, in Proceeding ACM International Symposium on Foundations of Software Engineering, Seattle, 2016, pp. 217–228.

[6] R. Buyya, C.S. Yeo, S. Venugopal, Market-oriented cloud computing: vision, hype, and reality for delivering IT services as computing utilities, in Proceeding IEEE International Conference on High Performance Computing and Communications, Dalian, 2008, pp. 5–13.

[7] L.P. Fabio, B. Benjamin, B. Leonardo, Z. Saul, A. Augusto, Virtual machine placement for elastic infrastructures in overbooked cloud computing datacenters under uncertainty, Future Gener. Comput. Syst. 79(3) (2018), 830–848.

[8] D. Serrano, S. Bouchenak, Y. Kouki, T. Ledoux, J. Lejeune, J. Sopena, L. Arantes, P. Sens, Towards QoS-oriented SLA guarantees for online cloud services, in Proceeding IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, Delft, 2013, pp. 50–57.

[9] J. Tai, D. Liu, Z. Yang, X. Zhu, J. Lo, N. Mi, Improving flash resource utilization at minimal management cost in virtualized flash-based storage systems, IEEE Trans. Cloud Comput. 5(3) (2017), 537–549.

[10] L. Nikolaos, D. Dimitrios, D. Spyros, P. Symeon, A hierarchical control framework of load balancing and resource allocation of cloud computing services, Comput. Electr. Eng. 67 (2018), 235–251.

[11] A. Hameed, A. Khoshkbarforoushha, R. RanjanEmail, P.P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q.M. Malluhi, N. Tziritas, A. Vishnu, S.U. Khan, A. Zomaya, A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems, Computing. 98(7) (2016), 751–774.

[12] X. Shi, M. Shang, W. Tian, A. Khushnood, S. Wang, T. Wu, Autonomic performance management of cloud server based on adaptive control method, in Proceeding IEEE International Conference on Networking, Sensing and Control, Zhuhai, 2018.

[13] P. Lama, X. Zhou, Coordinated power and performance guarantee with fuzzy MIMO control in virtualized server clusters, IEEE Trans. Comput. 64(1) (2015), 97–111.

[14] P.S. Saikrishna, R. Pasumarthy, Multi-objective switching controller for cloud computing systems, Control Eng. Pract. 57 (2016), 72–83.

[15] F. Yang, M. Paindavoine, Implementation of an RBF neural network on embedded systems: real-time face tracking and identity verification, IEEE Trans. Neural Netw. 14(5) (2003), 1162–1175.

[16] X. Zhen, W. Song, Q. Chen, Dynamic resource allocation using virtual machines for cloud computing environment, IEEE Trans. Parall. Distr. Syst. 24(6) (2013), 1107–1117.

[17] A. Paya, D.C. Marinescu, Energy-aware load balancing and application scaling for the cloud ecosystem, IEEE Trans. Cloud Comput. 5(1) (2017), 15–27.

[18] H.N. Van, F.D. Tran, J.M. Menaud, Performance and power management for cloud infrastructures, in Proceeding IEEE International Conference on Cloud Computing, Miami, 2010, pp. 329–336.

[19] J. Yang, C. Liu, Y. Shang, Z. Mao, J. Chen, Workload predicting-based automatic scaling in service clouds, in Proceeding IEEE International Conference on Cloud Computing, Santa Clara, 2013, pp. 810–815.

[20] R.R. Righi, V.F. Rodrigues, C.A. Costa, G. Galante, L.C.E. Bona, T. Ferreto, AutoElastic: automatic resource elasticity for high performance applications in the cloud, IEEE Trans. Cloud Comput. 4(1) (2016), 6–19.

[21] E.B. Lakew, C. Klein, F. Hernandez-Rodriguez, E. Elmroth, Towards faster response time models for vertical elasticity, in Proceeding IEEE/ACM International Conference on Utility and Cloud Computing, London, 2015, pp. 560–565.

[22] S. Spinner, S. Kounev, X. Zhu, L. Lu, M. Uysal, A. Holler, R. Griffith, Runtime vertical scaling of virtualized applications via online model estimation, in Proceeding IEEE International Conference on Self-Adaptive and Self-Organizing Systems, London, 2015, pp. 157–166.

[23] S. Farokhi, P. Jamshidi, E.B. Lakew, I. Brandic, E. Elmroth, A hybrid cloud controller for vertical memory elasticity: a control-theoretic approach, Future Gener. Comput. Syst. 65 (2016), 57–72.

[24] S. Farokhi, P. Jamshidi, D. Lucanin, I. Brandic, Performance-based vertical memory elasticity, in Proceeding IEEE International Conference on Autonomic Computing, Grenoble, 2015, pp. 151–152.

[25] R.N. Calheiros, E. Masoumi, R. Ranjan, R. Buyya, Workload prediction using ARIMA model and its impact on cloud applications' QoS, IEEE Trans. Cloud Comput. 3(4) (2015), 449–458.

[26] N. Karvonen, L.L. Jimenez, M.G. Simon, J. Nilsson, B. Kikhia, J. Hallberg, Classifier optimized for resource-constrained pervasive systems and energy-efficiency, Int. J. Comput. Intell. Syst. 10 (2017), 1272–1279.

[27] Rightscale Web Site, RightScale Cloud Management Platform, 2010. http://www.rightscale.com.

[28] S. Farokhi, E.B. Lakew, C. Klein, I. Brandic, E. Elmroth, Coordinating CPU and memory elasticity controllers to meet service response time constraints, in Proceeding IEEE International Conference on Cloud and Autonomic Computing, Boston, 2015, pp. 70–80.

[29] S. Farokhi, P. Jamshidi, I. Brandic, E. Elmroth, Self-adaptation challenges for cloud-based applications: a control theoretic perspective, in Proceeding International Workshop on Feedback Computing, Seattle, 2015.

[30] Amazon, Amazon auto scaling service, 2010. http://aws.amazon.com/autoscaling/.

[31] X. Wang, J. Zhu, Z. Zheng, W. Song, Y. Shen, M.R. Lyu, A spatial-temporal QoS prediction approach for time-aware web service recommendation, ACM Trans. Web. 10(1) (2016), 7: pp. 1–25.

[32] Y. Wang, X. Wang, M. Chen, X. Zhu, PARTIC: power-aware response time control for virtualized web servers, IEEE Trans. Parall. Distr. Syst. 22(2) (2011), 323–336.
[33] J. Luo, H. Wang, X. Gong, T. Li, A novel role-based access control model in cloud environments, Int. J. Comput. Intell. Syst. 9 (2016), 1–9.
[34] A. Abdelaziz, M. Elhoseny, A.S. Salama, A.M. Riad, A machine learning model for improving healthcare services on cloud computing environment, Measurement. 119 (2018), 117–128.
[35] Y. Liu, M.J. Lee, Y. Zheng, Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system, IEEE Trans. Mobile Comput. 15(10) (2016), 2398–2410.
[36] H. Morshedlou, M.R. Meybodi, Decreasing impact of SLA violations: a proactive resource allocation approach for cloud computing environments, IEEE Trans. Cloud Comput. 2(2) (2014), 156–167.
[37] R. Leena Sri, N. Balaji, Speculation based decision support system for efficient resource provisioning in cloud data center, Int. J. Comput. Intell. Syst. 10 (2017), 363–374.
[38] P.S. Saikrishna, R. Pasumarthy, N.P. Bhatt, Identification and multivariable gain-scheduling control for cloud computing systems, IEEE Trans. Control Syst. Technol. 25(3) (2017), 792–807.
[39] K.H. Ang, G. Chong, Y. Li, PID control system analysis, design, and technology, IEEE Trans. Control Syst. Technol. 13(4) (2005), 559–576.
[40] D. Chen, Research on traffic flow prediction in the big data environment based on the improved RBF neural network, IEEE Trans. Ind. Info. 13(4) (2017), 2000–2008.
[41] D. Wang, J. Huang, Neural network-based adaptive dynamic surface control for a class of uncertain nonlinear systems in strict-feedback form, IEEE Trans. Neural Netw. 16(1) (2005), 195–202.
[42] 1998 World Cup Web Site, 1998 World Cup Data Trace, 2012. http://ita.ee.lbl.gov/html/contrib/WorldCup.html.
[43] Z. Wang, X. Zhu, S. Singhal, Utilization and SLO-based control for dynamic sizing of resource partitions, in Proceeding International Workshop on Distributed Systems: Operations and Management, Barcelona, 2005, pp. 133–144.