Resource Allocation for Cloud-Based Software Services Using Prediction-Enabled Feedback Control With Reinforcement Learning

Xing Chen, Fangning Zhu, Zheyi Chen, Geyong Min, Xianghan Zheng, and Chunming Rong, *Senior Member, IEEE*

Abstract—With time-varying workloads and service requests, cloud-based software services necessitate adaptive resource allocation for guaranteeing Quality-of-Service (QoS) and reducing resource costs. However, due to the ever-changing system states, resource allocation for cloud-based software services faces huge challenges in dynamics and complexity. The traditional approaches mostly rely on expert knowledge or numerous iterations, which might lead to weak adaptiveness and extra costs. Moreover, existing RL-based methods target the environment with the fixed workload, and thus they are unable to effectively fit in the real-world scenarios with variable workloads. To address these important challenges, we propose a Prediction-enabled feedback Control with Reinforcement learning based resource Allocation (PCRA) method. First, a novel Q-value prediction model is designed to predict the values of management operations (by Q-values) at different system states. The model uses multiple prediction learners for making accurate Q-value prediction by integrating the Q-learning algorithm. Next, the objective resource allocation plans can be found by using a new feedback-control based decision-making algorithm. Using the RUBiS benchmark, simulation results demonstrate that the PCRA chooses the management operations of resource allocation with 93.7 percent correctness. Moreover, the PCRA achieves optimal/near-optimal performance, and it outperforms the classic ML-based and rule-based methods by 5∼7% and 10∼13%, respectively.

Index Terms—Cloud-based software services, resource allocation, reinforcement learning, Q-value prediction, feedback control

1 Introduction

With the rapid development of cloud computing, more resources become available for the software that is deployed on cloud platforms [1]. Therefore, cloud-based software services necessitate automatic resource allocation to best meet their requirements with high Quality-of-Service (QoS) and low resource costs [2]. However, it would be highly challenging for software services to request on-demand resources, due to constantly-changing workloads in the cloud environment [3]. Moreover, irrational resource allocation may seriously degrade QoS and lead to high resource costs for cloud-based software services [4]. To address these challenges, adaptive and efficient resource allocation is necessary for cloud-based software services.

- *X. Chen, F. Zhu, and X. Zheng are with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China, and also with the Fujian Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou 350116, China.*
  E-mail: {chenxing, n180320043, xianghan.zheng}@fzu.edu.cn.
- *Z. Chen and G. Min are with the Department of Computer Science, College of Engineering, Mathematics and Physical Sciences, University of Exeter, EX4 4QF Exeter, United Kingdom. E-mail: {zc300, g.min}@exeter.ac.uk.*
- *C. Rong is with the Department of Electronic Engineering and Computer Science, University of Stavanger, 4036 Stavanger, Norway.*
  E-mail: chunming.rong@uis.no.

Manuscript received 22 Dec. 2019; revised 30 Mar. 2020; accepted 1 May 2020. Date of publication 4 May 2020; date of current version 7 June 2022.
(Corresponding author: Zheyi Chen and Geyong Min.)
Recommended for acceptance by S. Guo.
Digital Object Identifier no. 10.1109/TCC.2020.2992537

There are classic approaches (e.g., rule-based strategies [5], heuristics [6], and control theory [7]) that can solve the resource allocation problem to some extent. But most of them are unable to effectively meet the real-world requirements of software services in the cloud environment with dynamic workloads and service requests. For instance, a fixed number of cloud resources can be easily allocated to software services by using rule-based strategies for meeting the expected maximum workload. However, it is difficult to set a resource threshold that can adaptively fulfill the dynamic demands of software services. Thus, they might lead to serious resource wastes. Moreover, heuristics commonly use the expert knowledge or management rules for a specific cloud system. Thus, they limit the application scopes and result in high overheads of rule settings and management. Besides, the control-theory based methods require numerous iterations to find feasible resource allocation plans. Therefore, they may cause unnecessary costs due to the frequent switching of virtual machines (VMs). Besides, machine learning (ML) [8] or deep learning (DL) [9] based solutions commonly need a large amount of historical system data and massive training time to build accurate QoS or workload prediction models. However, there is commonly no sufficient training data in the real-world cloud environment. Thus, the accuracy of prediction models would be seriously affected, and it is difficult to support the effectiveness of resource allocation. By contrast, reinforcement learning (RL) [10] can automatically make decisions through interacting with the environment without the support of historical data. Thus, RL has been recently used to tackle the complicated problem of resource allocation with low complexity and high robustness. However, the existing RL-based methods target the environment with the static workload, and thus the decision-making model needs to be retrained when workloads change [11]. Therefore, these classic approaches are unable to efficiently adapt to the real-world scenarios of cloud-based software services with variable workloads and service requests.

To address these essential challenges, a Prediction-enabled feedback Control with Reinforcement learning based resource Allocation (PCRA) method is proposed, in order to effectively achieve adaptive and efficient resource allocation for cloud-based software services. Through integrating the Q-learning algorithm [10] and multiple ML-based prediction learners, the proposed Q-value prediction model enables feedback control to efficiently find objective resource allocation plans for software services in the complex and dynamic cloud environment. The main contributions of this paper are summarized as follows.

- A new framework of resource allocation is designed to coordinate cloud resources for software services and collect runtime information. Based on the proposed framework, cloud resources are well monitored, while the decisions for resource allocation can be accurately made and executed.
- A novel Prediction-enabled feedback Control with Reinforcement learning based resource Allocation method is proposed to effectively obtain adaptive and efficient resource allocation for cloud-based software services. First, the Q-learning algorithm is used to evaluate the values of management operations in terms of Q-values. Next, based on the Q-values calculated by the Q-learning, a Q-value prediction model is developed to offer accurate and adaptive prediction for the Q-values of management operations at different system states. Notably, the proposed Q-value prediction model integrates multiple ML-based prediction learners, and the learner with the highest prediction accuracy is selected. Finally, a new feedback-control based decision-making algorithm is designed to make decisions for management operations based on the predicted Q-values, and it can be used to find objective resource allocation plans for cloud-based software services.
- Extensive simulation experiments using the real-world RUBiS benchmark are conducted to validate the effectiveness of the proposed PCRA method in achieving adaptive and efficient resource allocation for cloud-based software services. The simulation results show that the PCRA method is able to choose management operations of resource allocation with 93.7 percent correctness. Moreover, the optimal/near-optimal performance (QoS and resource costs) is obtained by using the PCRA method, and it outperforms classic ML-based and rule-based approaches by 5∼7% and 10∼13% under the runtime environment with various workloads and service requests.

The rest of this paper is organized as follows. In Section 2, the related work is analyzed. Section 3 formulates the resource allocation problem for cloud-based software services. In Section 4, the PCRA method is discussed in detail. Section 5 shows performance evaluation and analysis with the RUBiS benchmark. Finally, we conclude this work in Section 6.

2 Related Work

Resource allocation in cloud computing has attracted much research attention, while many scholars have contributed to addressing this important problem. In this section, we review the related work for cloud resource allocation.

As traditional methods for resource allocation, the rule-based strategies or heuristics commonly use different settings of thresholds or rules for meeting various types of service demands. Zahid *et al.* [5] proposed a ruled-based language for service providers with adaptation strategies, in order to improve QoS compliance in high-performance computing (HPC) clouds. A system model with probabilistic thresholds was developed in [12] for completing the switching between different operating levels of service. Kim *et al.* [13] proposed a threshold-based dynamic controller to alleviate the issues of immediacy and overhead that occurred in software-defined data center networks. The Johnson's rule and the genetic algorithm were integrated in [14] for dealing with the problem of multiprocessor scheduling in cloud data centers. To reduce resource costs while meeting user demands, Khatua *et al.* [6] proposed a heuristic-based method for resource reservation in public clouds. A clustering-based heuristic approach for edge resource allocation was developed in [15], in order to minimize the average service response time of applications. Ficco *et al.* [16] designed a meta-heuristic method for cloud resource allocation, where the game theory was adopted to optimize the policy. A heuristic method was designed in [17] to satisfy the QoS requirements of online social networks while reducing overheads of cloud resources. However, only deterministic resource demands were considered without performing dynamic provisioning. Therefore, when the environment changed (e.g., the user locations), it might not meet QoS requirements anymore. Although it is feasible to meet a specific requirement of a cloud-based software service by allocating a fixed number of cloud resources based on the rule-based strategies, different rules must be set respectively for fulfilling the dynamic service demands (e.g., response time and throughout). Moreover, the strategy achieved by using heuristics might be only a local optimum. Thus, not only the application scopes of these methods are seriously limited but also high overheads of rule settings and management are generated.

Control theory is well-established to design and model feedback loops for making cloud services self-adaptive and obtain a balance between speed and stability during the decision-making process. Avgeris *et al.* [7] proposed a hierarchical resource allocation and admission control mechanism, in order to enable mobile users to choose the proper edge servers for executing application tasks with lower response time and computational costs. Based on the feedback-control mechanism, a dynamic model was designed in [18] for Big Data MapReduce systems, in order to reduce the costs of cluster reconfigurations. Haratian *et al.* [19] developed an adaptive resource management framework for meeting QoS requirements, where the fuzzy controller can make decisions for resource allocation in each iteration of the control cycle. Based on reactive control techniques, synchronous programming and discrete controller were integrated in [20] for the design of autonomic management systems. Baresi *et al*. [21] proposed a discrete-time feedback controller for Web applications to autoscale their resources at VM and container levels. Based on feedback control and queueing theory, an autonomic controller was developed in [22] to elastically provision VMs for meeting the performance objectives that are associated with a particular data stream. Saikrishna *et al*. [23] utilized the feedback-control theory for modeling and performance management of a Web server hosted on a private cloud. A hybrid controller was designed in [24] for cloud elasticity, where the queuing theory was used to decide the service capacity per control interval with the consideration of concurrent request rates. However, the solution might not guarantee the application performance and it did not consider the latency caused by the switch-on of VMs. In general, the traditional control-theoretic solutions require numerous feedback iterations to find feasible resource allocation plans. Therefore, they may result in unnecessary costs due to the frequent discontinuing of VMs.

Learning-based methods (e.g., ML [8] and DL [9]) empower cloud systems with the ability to learn the specific domain knowledge from the historical data of applications for better resource allocation. Ranjbari *et al*. [25] proposed a learning-automata based method for optimizing QoS, energy efficiency, and VM migrations in cloud data centers. An ML-based prediction model was designed in [26] to forecast the memory requirements of applications with specific service level agreements (SLAs). Chen *et al*. [27] proposed a deep Learning based Prediction Algorithm for cloud Workloads (L-PAW) based on historical workload data, in order to better support cloud resource allocation via advanced resource configurations. A skewness-avoidance multi-resource allocation (SAMR) approach was designed for infrastructure-as-a-service (IaaS) clouds in [28] by using a model-based method to approximate the proper number of active physical machines. Based on the QoS prediction model trained by ML, the genetic algorithm was used in [29] to find feasible resource allocation plans. The historical data was gathered in [30] to extract the similarity among different cloud environments by using an ML-based method, and thus the optimal/near-optimal resource allocation plan can thus be preserved beforehand. Commonly, ML or DL based approaches need a large amount of historical system data and massive training time to build accurate QoS or workload prediction models for better supporting resource allocation. However, there is usually no adequate training data in the real-world cloud environment. Thus, the accuracy of prediction models would be degraded, and the effectiveness of resource allocation might be seriously affected. Meanwhile, cloud environments are innumerably changeable with unpredictable situations, and thus it is infeasible to simply execute training on the historical data. By contrast, RL-based solutions can make the decisions of resource allocation by interacting with the environment without the support of historical data. Orhean *et al*. [11] adopted the Q-learning algorithm to schedule the heterogeneous nodes in distributed systems for minimizing the execution time. By using the deep Q-networks (DQN) algorithm, a hierarchical framework was developed in [31] for adaptive resource allocation, which can reduce the power consumption in cloud data centers. Alsarhan *et al*. [32] designed an SLA framework based on RL for deriving a VM hiring policy, which can fit in dynamic system changes and meet QoS requirements of different clients in the cloud environment. An advantage actor-critic RL based resource allocation method was proposed in [33] for reducing the latency in job scheduling. However, the existing RL-based solutions target the environment with the static workload, and thus the decision-making model needs to be retrained whenever workloads change. Thus, these classic learning-based methods are unable to efficiently fit in the real-world scenarios of cloud-based software services with variable workloads and service requests.

To address these open challenges, we first integrate Q-learning and multiple ML-based algorithms to accurately predict the Q-values of management operations under the cloud environment without much available historical data. Next, based on the predicted Q-values, a new feedback-control based framework is used to efficiently find the objective resource allocation plans for cloud-based software services without excessive feedback iterations.

3 Problem Formulation

In this section, the problem of resource allocation for cloud-based software services is formulated. In general, self-adaptive cloud platforms are expected to balance QoS and resource costs when allocating resources to cloud-based software services. Thus, an objective function is needed to measure the potential resource allocation plans [34]. To guarantee QoS while reducing resource costs (denoted by $Cost$), the objective function $\mathcal{F}_{obj}$ is defined as

$$\mathcal{F}_{obj} = w_q \cdot \frac{1}{QoS} + w_c \cdot Cost, \tag{1}$$

where w_q and w_c are used to weight $\frac{1}{QoS}$ and $Cost$, respectively. More specifically, these two weights are pre-defined by cloud engineers, which reflect their different preferences on QoS and resource costs. For instance, the higher value of w_q represents a more sensitive preference for QoS, and thus more VMs are needed to guarantee QoS under the same workload. Moreover, the higher value of w_c indicates a more sensitive preference for resource costs, and thus less VMs are required to reduce resource costs. In practice, the most common objective function is to balance QoS and resource costs, but it is challenging due to their complex relationship in cloud services. Therefore, more rational resource allocation plans should be with the smaller values of $\mathcal{F}_{obj}$.

On one hand, QoS follows the service level agreements with the performance indicators, such as response time (RT) and data throughput (DH) [35]. Therefore, QoS is defined as

$$QoS = SLAs(RT, DH, \ldots), \tag{2}$$

where RT is the time of responding to service requests, and DH represents the ability of a system to process the information on a time scale.

TABLE 1
Datesets of Runtime Environment and Optional Resource Allocation Plans With Performance Indexes

Runtime environment		Optional resource allocation plans with performance indexes			
$WL_{current}$	$VM_{current}$	$VM_{optional}$	QoS	$Cost$	$\mathcal{F}_{obj}$
$x_{1,0}, x_{1,1}, \ldots, x_{1,w}$	$x_{1,w+1}, x_{1,w+2}, \ldots, x_{1,w+r}$	$x_{1,w+r+1}, x_{1,w+r+2}, \cdots \; x_{1,w+r+r}$	$x_{1,w+2r+1}$	$x_{1,w+2r+2}$	$x_{1,w+2r+3}$
$x_{2,0}, x_{1,1}, \ldots, x_{2,w}$	$x_{2,w+1}, x_{2,w+2}, \ldots, x_{2,w+r}$	$x_{2,w+r+1}, x_{2,w+r+2}, \ldots, x_{2,w+r+r}$	$x_{2,w+2r+1}$	$x_{2,w+2r+2}$	$x_{2,w+2r+3}$
...	...	...	...	...	...
$x_{u,0}, x_{u,1}, \ldots, x_{u,w}$	$x_{u,w+1}, x_{u,w+2}, \ldots, x_{u,w+r}$	$x_{u,w+r+1}, x_{u,w+r+2}, \ldots, x_{u,w+r+r}$	$x_{u,w+2r+1}$	$x_{u,w+2r+2}$	$x_{u,w+2r+3}$

On the other hand, $Cost$ consists of two parts, including the leased costs (the costs of renting VMs), denoted by $Cost_L$, and the discontinued costs (the costs of switching VMs), denoted by $Cost_D$. Thus, $Cost$ is defined as

$$Cost = Cost_L + Cost_D. \tag{3}$$

To reduce $Cost$, VMs should be rented in an on-demand manner. Moreover, it is necessary to avoid the extra costs caused by the frequent switching of VMs, in order to improve the stability of cloud-based software services.

Based on the above definitions, the runtime environment of resource allocation for cloud-based software services is first formulated by a 2-tuple, denoted by $\langle WL_{current}, VM_{current} \rangle$. As shown in Table 1, u pieces of historical data were collected. For each piece of historical data, $WL_{current} = (x_{i,0}, x_{i,1}, \ldots, x_{i,w})$ represents the current workload, where $x_{i,0}$ $(1 \leq i \leq u)$ is the amount of the current workload, w is the number of different types of service requests in the collected historical datasets, and $x_{i,m}$ $(1 \leq m \leq w)$ indicates the proportions of different types of service requests under the current workload. $VM_{current} = (x_{i,w+1}, x_{i,w+2}, \ldots, x_{i,w+r})$ represents the current resource allocation plan, where r is the number of VM types for resource allocation and $x_{i,w+n}$ $(1 \leq n \leq r)$ indicates the number of VMs of the nth type.

For each runtime environment, there are multiple resource allocation plans that are available to be chosen. As shown in Table 1, $VM_{optional}$ represents an optional resource allocation plan, which is denoted by $(x_{i,w+r+1}, x_{i,w+r+2}, \ldots, x_{i,w+r+r})$, where r is the number of VM types and $x_{i,w+r+n}$ $(1 \leq n \leq r)$ is the number of VMs of the nth type. Meanwhile, the corresponding QoS, $Cost$ and $\mathcal{F}_{obj}$ of each optional resource allocation plan are denoted by $x_{i,w+2r+1}$, $x_{i,w+2r+2}$ and $x_{i,w+2r+3}$, respectively. Therefore, an objective resource allocation plan (denoted by $VM_{objective}$) for the current environment is with the smallest value of $\mathcal{F}_{obj}$, and it is obtained by balancing QoS and $Cost$.

However, it is challenging to find objective resource allocation plans for cloud-based software services under the complex runtime environment with ever-changing workloads and service requests. The irrational resource allocation not only degrades QoS but also results in excessive resource costs. Therefore, resource allocation should be processed via an adaptive manner, in order to meet the real-world requirements of cloud-based software services with high QoS and low resource costs.

4 Prediction-Enabled Feedback Control for Adaptive Resource Allocation With RL

This section presents the proposed Prediction-enabled feedback Control with Reinforcement learning based resource Allocation method. The PCRA can be used to effectively obtain the adaptive and efficient resource allocation plans for cloud-based software services. Fig. 1 illustrates the framework of the PCRA method, whose main process is described as follows.

Step 1. The historical datasets contain the runtime data at different system states, including the current workload, resource allocation plan with QoS, and the corresponding objective resource allocation plan, as shown in Table 2. Next, the Q-learning algorithm [10] is applied to evaluate the values of management operations (by Q-values) that are taken under different system states. More specifically, the system state consists of the current workload and resource allocation plan in the runtime environment, management operations are to add or remove VMs of different types, and the corresponding rewards can be received when objective resource allocation plans are found.

Step 2. The Q-values of management operations in the Q-value table (built in *Step 1*) are first preprocessed according to the management experience. Next, the Q-value prediction model can be attained through training ML-based algorithms based on the preprocessed Q-values. More specifically, three ML-based prediction algorithms (i.e., support vector machine (SVM), classification and regression tree (CART), and nonlinear regression (NLREG)) [8] are used as the potential learners to train the Q-value prediction model, and the model with the highest accuracy is selected. Thus, the Q-values of different management operations can be predicted accurately when inputting the current system state (the workload and resource allocation plan with QoS).

Step 3. At runtime, the Q-value prediction model (obtained in *Step 2*) is first used to predict the Q-values of different management operations based on the current workload, resource allocation plan, and the corresponding QoS. Next, the runtime decisions of choosing management operations are executed by comparing their corresponding Q-values. Finally, objective resource allocation plans can be gradually found through using a feedback-control based decision-making algorithm.

4.1 Q-Value Evaluation for Management Operations

Reinforcement learning [10] can make automatic decisions through interacting with the environment without prior knowledge. This advantage enables the application of RL in the problem of resource allocation under the complex cloud environment. Therefore, an RL algorithm (i.e., Q-learning) is applied to evaluate the values of different management operations (by Q-values) for exploring the most effective resource allocation for cloud-based software

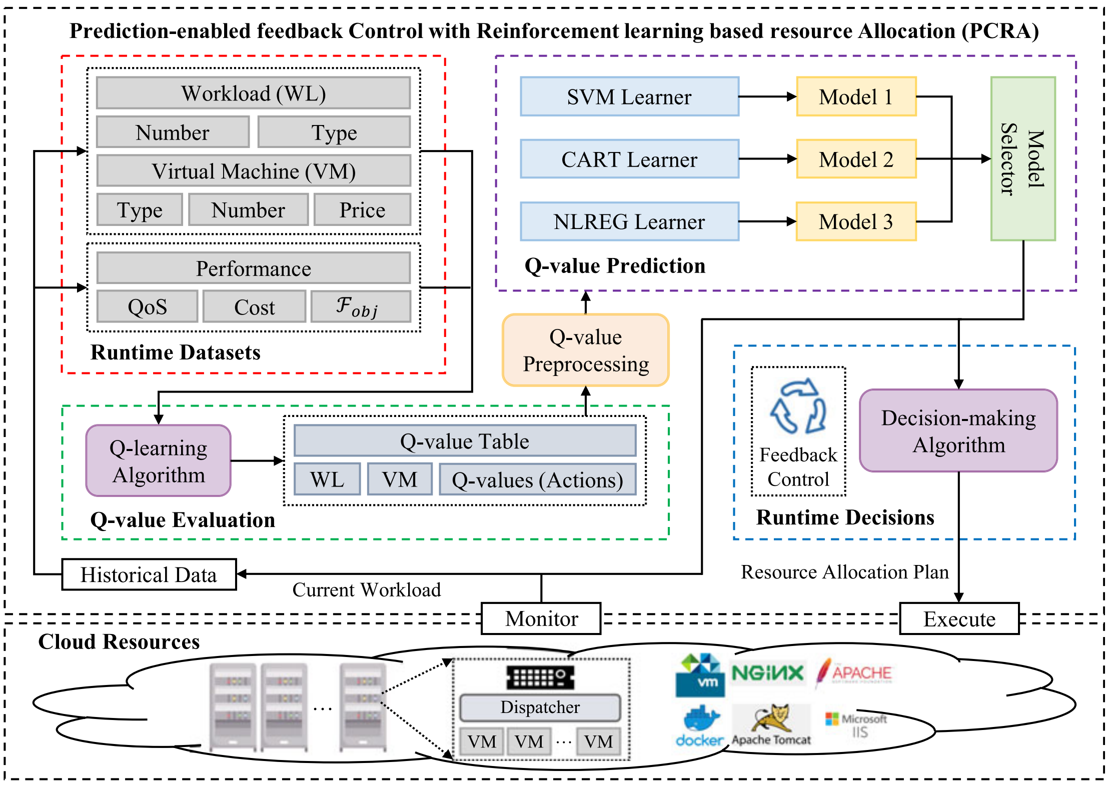

Fig. 1. The framework of the proposed PCRA method.

TABLE 2
Datasets of Runtime Data at Different System States

$WL_{current}$	$VM_{current}$	QoS	$VM_{objective}$
$x_{1,0}, x_{1,1}, \ldots, x_{1,w}$	$x_{1,w+1}, x_{1,w+2}, \ldots, x_{1,w+r}$	$x_{1,w+r+1}$	$x_{1,w+r+1+1}, x_{1,w+r+1+2}, \ldots\ x_{1,w+r+1+r}$
$x_{2,0}, x_{2,1}, \ldots, x_{2,w}$	$x_{2,w+1}, x_{2,w+2}, \ldots, x_{2,w+r}$	$x_{2,w+r+1}$	$x_{2,w+r+1+1}, x_{2,w+r+1+2}, \ldots\ x_{2,w+r+1+r}$
...	...	...	...
$x_{p,0}, x_{p,1}, \ldots, x_{p,w}$	$x_{p,w+1}, x_{p,w+2}, \ldots, x_{p,w+r}$	$x_{p,w+r+1}$	$x_{p,w+r+1+1}, x_{p,w+r+1+2}, \ldots, x_{p,w+r+1+r}$

services. Different resource allocation plans are with various values of the objective function. The objective resource allocation plan (i.e., the optimal one) should consider both QoS and resource costs. Based on the operating data of the current environment and software services, administrators can obtain the objective resource allocation plan (denoted by $VM_{objective}$) with the smallest value of the objective function. Therefore, for each runtime environment in Table 1, $VM_{objective}$ is searched and recorded in Table 2. More specifically, Table 2 records p pieces of operating data obtained from Table 1 with different system states, where $WL_{current} = (x_{i,0}, x_{i,1}, \ldots, x_{i,w})$ $(1 \leq i \leq p)$ is the current workload, $VM_{current} = (x_{i,w+1}, x_{i,w+2}, \ldots, x_{i,w+r})$ is the current resource allocation plan, the corresponding QoS is denoted by $x_{i,w+r+1}$, and the objective resource allocation plan under the current environment is denoted by $VM_{objective} = (x_{i,w+r+1+1}, x_{i,w+r+1+2}, \ldots, x_{i,w+r+1+r})$.

Through utilizing the historical system data, the RL agent can gradually find the management operation with the highest Q-value under different system states, where the Q-learning algorithm guides the learning process of Q-value evaluation. As the target of RL is to maximize the cumulative rewards, it is commonly modeled by using the Markov decision process (MDP). More specifically, an MDP can be defined as a 4-tuple, denoted by $\langle S, A, P, R\rangle$, where S is the state space, A is the action space, P is the state transition function, and R is the reward function.

Based on the problem formulation (described in Section 3), the corresponding state space S, action space A, and reward function R in the proposed resource allocation problem are defined as follows.

State Space. The state space S is defined as a 2-tuple, denoted by $S = \langle WL_{current}, VM_{current}\rangle$, as described in Table 2. Thus, S represents the current system state of the runtime environment, which consists of the current workload and resource allocation plan.

Action Space. The action space is defined as $A = \{add_1, rm_1, add_2, rm_2, \ldots, add_r, rm_r\}$, where an action (management operation) is to add or remove a VM of a certain type. For example, add_r is to add a VM of the rth type, and rm_r is to remove a VM of the rth type.

Reward Function. The reward function is used to guide the RL agent to find the objective resource allocation plan, which is defined as

$$R(VM_{current}, VM_{objective}, a) = \begin{cases} 10, & getNextState(VM_{current}, a) == VM_{objective} \\ -1, & getNextState(VM_{current}, a) \notin VM_{optional} \\ 0, & other\ cases \end{cases}, \tag{4}$$

where $VM_{objective}$ is defined as an objective resource allocation plan, and the action a ($a \in A$) represents a management operation. If $VM_{objective}$ is found by taking an action a under the current resource allocation plan $VM_{current}$, the RL agent will receive the reward of 10. If an unknown resource allocation plan (not in $VM_{optional}$) is generated by taking an action a under $VM_{current}$, the RL agent will receive the reward of -1. In other cases, no reward will be produced for taking any actions.

During the training process, the RL agent chooses an action a ($a \in A$) by using the ϵ-greedy algorithm [10] at the state s ($s \in S$), and then it receives the instant reward r and the state transition $P(s'|s, a)$ happens. Therefore, the Q-value, denoted by $Q(s, a)$, represents the value obtained by choosing the action a at the state s, which is updated by

$$Q(s, a) \leftarrow Q(s, a) + \alpha \cdot [r + \gamma \cdot max_{a'} Q(s', a') - Q(s, a)], \tag{5}$$

where α is the learning rate, γ is the discount factor, and $max_{a'} Q(s', a')$ is the maximum Q-value obtained by choosing the action a' at the next state s'.

Based on the above definitions, the Q-learning algorithm is used to evaluate the Q-values of different management operations for achieving the objective resource allocation plans. The key steps of the proposed algorithm are as shown in Algorithm 1. First of all, the Q-value table (denoted by Q_Table) is initialized with arbitrary Q-values (Line 4). Next, the Q-values of management operations in each optional resource allocation plan are evaluated by using the Q-learning algorithm, and the training process continues until the Q-values converge (Line 5). In each epoch, $VM_{current}$ is initialized with a random optional resource allocation plan in $VM_{optional}$ (Line 6). If $VM_{current}$ is not $VM_{objective}$ (Line 7), an action a is chosen from the action space A by using the ϵ-greedy algorithm according to the current Q-values (Line 8). Next, the reward r of the action a is calculated by using Eq. (4) (Line 9), and the corresponding resource allocation plan in the next step (denoted by VM_{next}) is generated (Line 10). Then, the Q-values in Q_Table are updated by using Eq. (5) (Line 11). Next, $VM_{current}$ is replaced by VM_{next} and the state transition happens (Line 12). Finally, Q_Table will be continuously updated until $VM_{objective}$ is found.

Thus, the Q-value table records the workloads, resource allocation plans, and the corresponding Q-values of each management operation (adding or removing VMs of different types) at different times.

4.2 Q-value Prediction Model

Although feasible management operations can be decided according to the Q-values evaluated by using the Q-learning algorithm, the decision-making model of resource allocation needs to be retrained when workloads change. This is because the traditional RL-based approaches target the cloud environment with the static workload. Therefore, they are unable to effectively fit in the real-world scenarios of cloud-based software services with variable workloads and service requests. To address this important problem, a Q-value prediction model is designed to predict the Q-values of management operations under different scenarios with various workloads and service requests. Therefore, the proposed Q-value prediction model can significantly improve the adaptiveness and efficiency of resource allocation in the runtime environment.

TABLE 3
An Example of Q-Value Table When $WL_{current} = (5000, 0.45, 0.55)$ and Management Operation $a = add_3$

No. of VMs		The 2nd type							
		...	2	3	4	5	6	7	8
	...	...	...	...	...	...	...	...	...
	2	...	5.12	6.40	8.00	10.0	10.0	8.00	6.40
	3	...	4.01	5.12	6.40	***0.00***	***0.00***	6.40	5.12
	4	...	4.01	5.12	6.40	***0.00***	***0.00***	6.40	5.12
The 3rd type	5	...	3.28	4.01	5.12	6.40	6.40	5.12	4.01
	6	...	2.62	3.28	4.01	5.12	5.12	4.01	3.28
	7	...	2.10	2.62	3.28	4.01	4.01	3.28	2.62
	8	...	***0.00***	***0.00***	***0.00***	***0.00***	***0.00***	***0.00***	***0.00***

Algorithm 1. Q-Learning for Evaluating the Q-Values of Management Operations

```
1: Input: the current workload $WL_{current}$, the current resource
   allocation plan $VM_{current}$, and the objective resource
   allocation plan $VM_{objective}$.
2: Output: the Q-value table $Q\_Table$ that contains the
   Q-values of different management operations.
3: Initialize the learning rate $\alpha$ and the discount factor $\gamma$.
4: Initialize $Q\_Table$ with arbitrary Q-values.
5: for each epoch do
6:    Initialize the current resource allocation plan:
      $VM_{current} = random(VM_{optional})$.
7:    while $VM_{current} \neq VM_{objective}$ do
8:       Choose action $a$ from the action space $A$ using the
         $\epsilon$-greedy according to Q-values:
         $a = choose\_action(VM_{current}, Q\_Table)$.
9:       Execute action $a$ and receive reward $r$:
         $r = R(VM_{current}, VM_{objective}, a)$.
10:      Step to the next resource allocation plan:
         $VM_{next} = getNextState(VM_{current}, a)$.
11:      Update the Q-values in $Q\_Table$:
         $Q(VM_{current}, VM_{objective}, a) \leftarrow Q(VM_{current}, VM_{objective}, a) +$
         $\alpha \cdot [r + \gamma \cdot max_{a'} Q(VM_{next}, a') - Q(VM_{current}, VM_{objective}, a)]$.
12:      Update the current resource allocation plan:
         $VM_{current} = VM_{next}$.
13:   end
14: end
```

However, it is difficult to train an effective Q-value prediction model by directly using the raw data of Q-values, and the reasons are described as follows.

- When the number of VMs of a certain type in a resource allocation plan is not within the assignable amount after taking management operations (i.e., $getNextState(VM_{current}, a) \notin VM_{optional}$), the corresponding Q-values mutate into 0. Table 3 indicates the impact of changing the number of VMs on Q-values when the current workload $WL_{current} = (5000, 0.45, 0.55)$ and management operation $a = add_3$, where the number of VMs of the 1st type is 0, the amount of the current workload is 5,000, and the proportions of two different types of tasks (service requests) under the current workload are 0.45 and 0.55, respectively. When executing the management operation (i.e., add_3), the number of VMs of the 3rd type will exceed its maximum assignable amount. Thus, the Q-values of the management operation will always stay at 0 regardless of the current number of VMs of the 2nd type. In this case, there is no gradual process of Q-value changes, and it would seriously influence the prediction accuracy of Q-values for different management operations.

- When approaching $VM_{objective}$, the Q-values of management operations commonly become larger. However, the Q-values mutate into 0 when $VM_{objective}$ is achieved (i.e, $VM_{current} == VM_{objective}$). This is because there is a singularity problem [36] existing in the process of solving the fitting function. This problem makes it difficult to effectively predict the Q-values of management operations when $VM_{current}$ is close to $VM_{objective}$. For example, as shown in Table 3, when the number of VMs of the 1st type is 0, and the number of VMs of the 2nd and the 3rd types are 5 and 2 (i.e., $VM_{current} = (0, 5, 2)$), high Q-values (i.e., 10) are obtained and $VM_{objective} = (0, 5, 3)$ is approaching. However, the Q-values will mutate into 0 if a VM of the 3rd type is added at this time.

To address the above problems, the Q-values of management operations are preprocessed as

$$Q\_value(VM_{current}, VM_{objective}, a) = \begin{cases} I, & getNextState(VM_{current}, a) \notin VM_{optional} \\ 0, & VM_{current} == VM_{objective} \\ \frac{1}{Q\_value}, & other\ cases. \end{cases} \tag{6}$$

If $getNextState(VM_{current}, a) \notin VM_{optional}$, the corresponding management operations will be regarded as illegal, and thus the Q-values of these management operations will be indexed by I. If $VM_{current} == VM_{objective}$ (i.e., $VM_{objective}$ is found), the Q-values will be set to 0. In other cases, the Q-values will be set to their reciprocals (i.e., $\frac{1}{Q\_value}$). Therefore, the closer to $VM_{objective}$, the smaller the Q-values of management operations are. Table 4 illustrates an example of preprocessing results for the Q-values in Table 3.

Based on the preprocessed Q-values of management operations, the Q-value prediction model is trained by respectively using the SVM, CART, and NLREG algorithms [8], and the model with the highest prediction accuracy is selected. For the problem of Q-value prediction, the correlation between input X and output Y is studied. As shown in Table 5, $X = (WL_{current}, VM_{current}, QoS)$, and Y denotes the Q-values of corresponding management operations. Especially, the illegal management operations (the Q-values are indexed by I) are eliminated by default. More specifically, the core ideas of the SVM, CART, and NLREG algorithms are described as follows.

TABLE 4
An Example of Preprocessing Results for the Q-Values in Table 3

No. of VMs		The 2nd type							
		...	2	3	4	5	6	7	8
	...	...	...	...	...	...	...	...	...
	2	...	***0.20***	***0.16***	***0.13***	***0.10***	***0.10***	***0.13***	***0.16***
	3	...	***0.24***	***0.20***	***0.16***	0.00	0.00	***0.16***	***0.20***
The 3rd type	4	...	***0.24***	***0.20***	***0.16***	0.00	0.00	***0.16***	***0.20***
	5	...	***0.31***	***0.24***	***0.20***	***0.16***	***0.16***	***0.20***	***0.24***
	6	...	***0.38***	***0.31***	***0.24***	***0.20***	***0.20***	***0.24***	***0.31***
	7	...	***0.48***	***0.38***	***0.31***	***0.24***	***0.24***	***0.31***	***0.38***
	8	...	***I***	***I***	***I***	***I***	***I***	***I***	***I***

SVM. The hyperplane equation is defined as

$$Y = u^T \cdot \varphi(X) + \upsilon, \tag{7}$$

where X is the input matrix and Y is the output matrix. Moreover, the problem of solving the parameters (u^T, υ) is transformed into a mapping problem of feature space, and it is processed via the Gaussian kernel function as

$$K(x_i, x_j) = exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right), \tag{8}$$

where $x_i, x_j \in X$, and $\sigma > 0$.

CART. The purity of datasets is first calculated by

$$Gini(D) = \sum_{k=1}^{|r|} \sum_{k' \neq k} p_k \cdot p'_k = 1 - \sum_{k=1}^{|r|} p_k^2, \tag{9}$$

where the datasets are denoted by $D = (X, Y)$ that contain r categories, and p_k is the proportion of the kth category.

Next, the Gini index function is defined as

$$Gini(D, att) = \sum_{\upsilon=1}^{r} \frac{|D^\upsilon|}{|D|} Gini(D^\upsilon), \tag{10}$$

where att is the attribute list of D. When the smallest value of $Gini(D, att)$ is achieved, the corresponding att can be regarded as the optimal partition.

NLREG. The regression equation is defined as

$$\begin{aligned} y_k = w_0 \cdot x_{k,0} + w_1 \cdot x_{k,1} + \ldots + w_m \cdot x_{k,m} \\ + w_{m+1} \cdot x_{k,m+1} + \ldots + w_{m+n} \cdot x_{k,m+n} + b, \end{aligned} \tag{11}$$

where $W = [w_0, w_1, \ldots, w_m, w_{m+1}, \ldots, w_{m+n}]$ is the weight matrix and b is the bias. Moreover, the least-squares method is used to establish the matrix about W and b. The target of the NLREG is to minimize the loss function, and it is evaluated by using the mean square error (MSE).

Furthermore, the mean absolute error (MAE) and R-squared (R^2) [8] are respectively used to measure the prediction accuracy of Q-values for the above three algorithms as

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - y'_i|, \tag{12}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N} (y_i - y'_i)^2}{\sum_{i=1}^{N} (y_i - \bar{y}_i)^2}, \tag{13}$$

TABLE 5
Training Datasets of Q-Value Prediction Model

Input X			Output Y				
$WL_{current}$	$VM_{current}$	QoS	add_1	rm_1	...	add_r	rm_r
$x_{1,0}, x_{1,1}, \ldots, x_{1,w}$	$x_{1,w+1}, x_{1,w+2}, \ldots, x_{1,w+r}$	$x_{1,w+r+1}$	$Q_{1,10}$	$Q_{1,11}$	...	$Q_{1,r0}$	$Q_{1,r1}$
$x_{2,0}, x_{2,1}, \ldots, x_{2,w}$	$x_{2,w+1}, x_{2,w+2}, \ldots, x_{2,w+r}$	$x_{2,w+r+1}$	$Q_{2,10}$	$Q_{2,11}$	...	$Q_{2,r0}$	$Q_{2,r1}$
...	...	...	...	...	...	...	...
$x_{p,0}, x_{p,1}, \ldots, x_{p,w}$	$x_{p,w+1}, x_{p,w+2}, \ldots, x_{p,w+r}$	$x_{p,w+r+1}$	$Q_{p,10}$	$Q_{p,11}$	...	$Q_{p,r0}$	$Q_{p,r1}$

where N is the size of the testing set, y_i is the actual Q-value, and y_i' is the predicted Q-value. For achieving high prediction accuracy, MAE and R^2 are expected to be close to 0 and 1, respectively.

Finally, one of the above three algorithms with the highest prediction accuracy will be selected and used to train the proposed Q-value prediction model. The model promises the accurate and adaptive prediction of the Q-values for different management operations under the runtime environment with various workloads and service requests.

4.3 Runtime Decisions for Management Operations

Through utilizing the predicted Q-values of different management operations from the proposed Q-value prediction model, the decision-making process of choosing management operations is executed at runtime. The main steps of this process are described in Algorithm 2. First of all, if the management operations are regarded as illegal, the corresponding Q-values will be indexed by I (Lines 9 and 10). Otherwise, the Q-values of management operations will be evaluated by using the Q-value prediction model (Lines 11~13). Next, if the Q-values of all legal management operations are less than or equal to the predefined threshold T (except the Q-values indexed by I), the objective resource allocation plan is considered to be found and it is unnecessary to take any more management operations (Lines 15 and 16). Otherwise, the management operations with the minimum Q-value will be executed (Lines 18~21), and the objective resource allocation plan will continually be sought.

Based on the proposed decision-making algorithm, the objective resource allocation plan can be gradually found via feedback control in the runtime environment. At each iteration, feasible management operations are chosen and executed. And the iterations of feedback control will continue until the output of the decision-making algorithm is null (the objective resource allocation plan is found).

5 Experiments

In this section, we first present the settings and datasets in our experiments (Section 5.1). Next, we implement the proposed PCRA method and conduct performance evaluation to explore the research questions (RQs) as follows.

- *RQ 1:* Whether the PCRA method is able to effectively realize adaptive resource allocation in different runtime environments (Section 5.2)?
- *RQ 2:* How does the PCRA method perform in Q-value prediction and decisions for management operations (Section 5.3)?
- *RQ 3:* How much does the PCRA method improve the performance of resource allocation compared with classic approaches (Section 5.4)?

For *RQ 1*, the experimental results show that the PCRA method achieves the performance comparable with the ideal resource allocation plans at different system states, and the performance gap between them is less than 3 percent. For *RQ 2*, the PCRA method can choose management operations of resource allocation with 93.7 percent correctness when using the SVM algorithm for training the Q-value prediction model. For *RQ 3*, the results show that the PCRA method outperforms the classic ML-based and rule-based resource allocation approaches by 5~7% and 10~13%, respectively.

Algorithm 2. Decision-Making Algorithm for Management Operations

```
1:  Input: the current workload $WL_{current}$, and the current
    resource allocation plan $VM_{current}$ with the corresponding
    value of $QoS$.
2:  Output: the action $a$ (management operation).
3:  Declare:
4:    $A$ – the action space that contains all possible
    management operations, where $a \in A$.
5:    $a\_List$ – the set of $a$ with the minimum Q-value.
6:    $Q\_value[a]$ – the Q-value of $a$.
7:    $getQvalue()$ – call the Q-value prediction model.
8:  for each $a \in A$ do
9:    if $getNextState(VM_{current}, a) \notin VM_{optional}$ then
10:     $Q\_value(a) = I$.
11:   else
12:     Evaluate the Q-value of $a$ by calling the Q-value
        prediction model:
        $Q\_value[a] = getQvalue(WL_{current}, VM_{current}, QoS, a)$.
13:   end
14: end
15: if (foreach $Q\_value[a] <= T \,||\, Q\_value[a] == I$) then
16:   return null.
17: else
18:   Record the actions with the minimum Q-value:
      $a\_List = A.getAction\_MinQvalue()$.
19:   Randomly choose an action in $a\_List$:
      $a = a\_List.getAction\_Random()$.
20:   return $a$.
21: end
```

5.1 Settings and Datasets

The experiments were conducted based on the CloudStack [37], where three different types of VMs (i.e., small, medium and large) are deployed. Table 6 lists the configurations of different types of VMs on the CloudStack, including CPUs,

TABLE 6
Configurations of Different Types of VMs on CloudStack

Property	Different types of VMs		
	Small	*Medium*	*Large*
CPU	1 core	1 core	1 core
Memory	1 GB	2 GB	4 GB
$Cost_L$	1.761 RMB	1.885 RMB	2.084 RMB
$Cost_D$	0.440 RMB	0.471 RMB	0.521 RMB

memories, $Cost_L$ and $Cost_D$. Moreover, the number of VMs of each type is denoted by vm_S, vm_M, and vm_L, respectively. Thus, the current resource allocation plan can be expressed as $VM_{current} = (vm_S, vm_M, vm_L)$. Based on the CloudStack, we run the RUBiS benchmark [38], a prototype of the auction site that is modeled according to *eBay.com*. The RUBiS benchmark can be used to simulate user behavior for different workload patterns and evaluate the performance scalability of application servers. More specifically, the number of users represents the amount of the current workload, and the user behaviors contain the service requests of browsing and auctioning (i.e., different types of tasks).

Next, the runtime datasets were generated and collected by running the RUBiS benchmark on the CloudStack for two months. The datasets contain the runtime data at different system states, including the current workload, current resource allocation plan with QoS, and objective resource allocation plan. Specifically, the amount of the current workload and the proportions of different types of tasks (the service requests of browsing and auctioning in user behaviors) are distributed in [3,000, 5,000] and [0, 1], respectively. In more detail, we randomly split the datasets (about 4,000 records) into two parts, including the training set (75 percent) and the testing set (25 percent). In the PRCA method, w_q and w_c are set to 320 and 10 for balancing the QoS and resource costs [39]. Based on the historical datasets, the Q-learning algorithm is implemented based on the TensorFlow 1.4.0 [40], where the episode, the learning rate α, and the discount factor γ are set to 100, 0.1 and 0.8, respectively. Next, the Q-values of different management operations are evaluated by using the Q-learning algorithm, and then the Q-values are preprocessed. Finally, three ML-based algorithms are used to train the Q-value prediction model, where the one with the highest prediction accuracy will be selected.

TABLE 7
Different Cases on RUBiS Benchmark With Various Initializations

No.	$WL_{current}$			$VM_{current}$ (Initialized)		
	Workload	*Browse*	*Auction*	vm_S	vm_M	vm_L
1	3,000	0.25	0.75	2	3	5
2	3,000	0.20	0.80	3	1	4
3	3,500	0.30	0.70	1	4	2
4	3,500	0.35	0.65	2	5	5
5	4,000	0.65	0.35	4	4	6
6	4,000	0.45	0.55	5	3	2
7	4,500	0.75	0.25	4	5	3
8	4,500	0.25	0.75	2	3	4
9	5,000	0.45	0.55	0	1	1
10	5,000	0.35	0.65	3	2	2

TABLE 8
Comparison Between PCRA Plans and Ideal Plans

No.	PCRA Plans			Ideal Plans		
	vm_S	vm_M	vm_L	vm_S	vm_M	vm_L
1	0	1	4	0	2	4
2	0	1	4	0	1	4
3	0	3	4	0	2	4
4	0	3	4	0	3	4
5	0	3	5	0	3	5
6	0	2	4	0	3	4
7	0	4	4	0	4	4
8	0	3	5	0	2	5
9	0	3	5	0	3	5
10	0	2	6	0	2	5

According to the above settings, ten different cases were simulated based on the RUBiS benchmark with various initializations of $WL_{current}$ (including the amount of the current workload and the proportions of different types of service requests) and the corresponding initialized resource allocation plans ($VM_{current}$), as shown in Table 7. Moreover, the runtime decision-making algorithm is used to find the objective resource allocation plan, where the threshold T is set to 0.1. Therefore, if the Q-values of all management operations are less than or equal to T, no further management operation will be performed.

5.2 RQ 1: Effectiveness Evaluation of the PCRA

First of all, we evaluate the effectiveness of the proposed PCRA method for adaptive resource allocation with different cases in Table 7. More specifically, we compare the resource allocation plans obtained by the PCRA method and the ideal ones in these ten different cases, where the ideal plan can best meet the requirements with high QoS and low resource costs in reality. However, it is infeasible to find the ideal plans in practice since it needs to exhaustively try all the possibilities and thus the complexity is unacceptably high. For example, there are three types of VMs with their allocation numbers ranging from 0 to 8, and thus we must execute 729 resource allocation plans for the given workload and allocated VMs. Meanwhile, we need to compare the values of the objective function for these 729 plans, in order to find the ideal one under the current case. In fact, we combine management experience and local verification to obtain ideal plans under different cases. As shown in Table 8, the PCRA method is able to attain similar resource allocation plans as the ideal ones, and the difference between them is tiny. For instance, in Cases 2, 4, 5, 7 and 9, they are the same. In other cases, there is only a small difference (one management operation for a VM) between our plans and the ideal ones. Furthermore, we compare the performance of resource allocation (in terms of $\mathcal{F}_{obj}$) between the PCRA method and the ideal plans under different cases. As shown in Fig. 2, the PCRA method can achieve comparable performance with the ideal plans in different cases, and the performance gap between them is always less than 3 percent. The results show that the PCRA method achieves optimal/near-optimal performance (QoS and resource costs) of resource allocation, and it can well meet the requirements of resource allocation

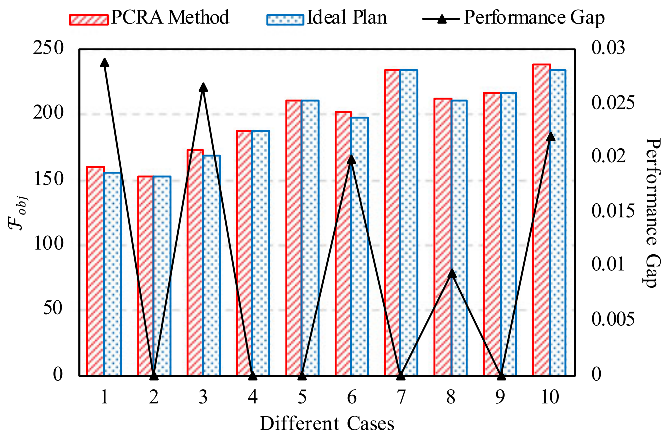

Fig. 2. Performance comparison between the PCRA and ideal plans.

TABLE 9
Resource Allocation Process of the PCRA in Case 9

$VM_{current}$		add_1	rm_1	add_2	rm_2	add_3	rm_3
0	0, 1, 1	0.491	0.465	***0.312***	0.440	0.316	0.448
1	0, 2, 1	0.387	0.378	0.272	0.390	***0.248***	0.372
2	0, 2, 2	0.306	0.348	0.213	0.317	***0.193***	0.311
3	0, 2, 3	0.240	0.255	0.168	0.246	***0.153***	0.242
4	0, 2, 4	0.182	0.197	0.131	0.188	***0.130***	0.187
5	0, 2, 5	0.146	0.154	***0.106***	0.154	0.135	0.148
6	0, 3, 5	0.098	0.098	0.098	0.098	0.097	0.099

for cloud-based software services with various workloads and service requests.

Next, we take Case 9 as an example to explain the resource allocation process of the PCRA method, where $WL_{current} = 5000$ and the proportions of different types of tasks (the service requests of browsing and auctioning) are 0.45 and 0.55, respectively. As shown in Table 9, when the initialized $VM_{current}$ is (0, 1, 1), the predicted Q-value of add_2 is the smallest among the Q-values of all management operations. Thus, the number of VMs of the 2nd type is added by one, and then $VM_{current}$ transfers to (0, 2, 1). At this step, add_3 is with the smallest predicted Q-value, and thus a VM of the 3rd type is added. Similarly, the management operation with the smallest predicted Q-value is executed at each step. Finally, when $VM_{current}$ transfers to (0, 3, 5), the predicted Q-values of all management operations are less than the predefined threshold T (i.e., 0.1). Thus, the objective resource allocation has been found efficiently, and no further management operation will be taken after this step.

5.3 RQ 2: Performance Evaluations of Q-Value Prediction Model and Decisions for Management Operations

In this subsection, performance evaluations of the proposed Q-value prediction model are first conducted with different training algorithms (i.e., SVM, NLREG, and CART). More specifically, MAE and R^2 (defined in Eqs. (12) and (13)) are used to measure the accuracy of the Q-value prediction model. Moreover, the action accuracy rate (AAR) is defined to measure the correctness of management operations in the decision-making process as

TABLE 10
Performance Comparison Among Different Q-Value Prediction Models

Different models	Training time	Performance indexes		
		MAE	R^2	AAR
SVM	2.34 s	0.2012	0.6569	93.7%
CART	1.99 s	0.2683	0.5539	90.5%
NLREG	3.03 s	0.2993	0.5128	88.4%

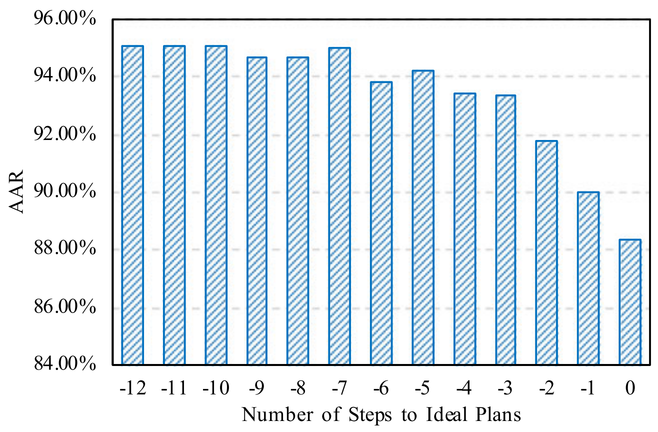

Fig. 3. Decision-making correctness with different steps to ideal plans.

$$AAR = \frac{A}{M} \cdot 100\%, \tag{14}$$

where M is the number of all management operations that are taken, and A is the number of correct ones. If the management operations have beneficial effects on finding the objective resource allocation plan (i.e., taking the management operations can make the current resource allocation plan approach to the objective plan), they are regarded as the correct management operations.

As shown in Table 10, all these three models with different algorithms can complete the training process in about 3 seconds. The results exhibit their excellent training efficiency. Moreover, the SVM-based model achieves the highest Q-value prediction accuracy among these three models in terms of MAE and R^2, while it can obtain higher AAR than the others with around 3∼5% performance improvement. Therefore, better management operations of resource allocation can be taken during the decision-making process by using the SVM-based model for Q-value prediction.

Next, the correctness of management operations (measured by AAR) in the decision-making process of the proposed PCRA method is studied with different numbers of steps to ideal resource allocation plans. As shown in Fig. 3, the AAR decreases when the current resource allocation plans are approaching ideal ones. For example, the AAR is about 95 percent when there are 7 or more steps from current resource allocation plans to the ideal ones. Even if the current resource allocation plans are close to the ideal ones, the AAR still keeps around 90 percent. Therefore, when there are many steps to the ideal resource allocation plans, the PCRA method can always make decisions for management operations with high correctness under the support of Q-value prediction model. Only when approaching the ideal plans,

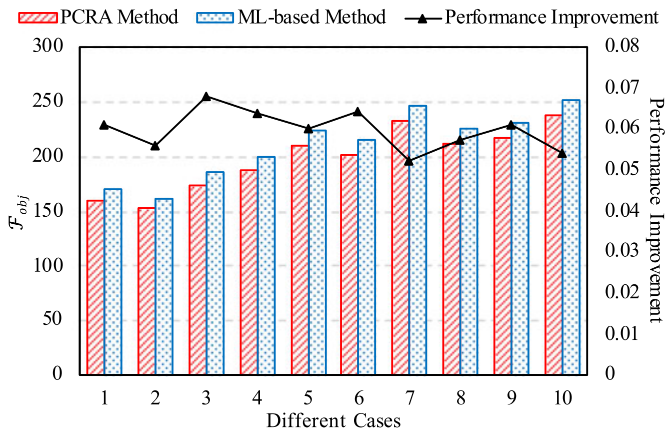

Fig. 4. Performance comparison with the ML-based method.

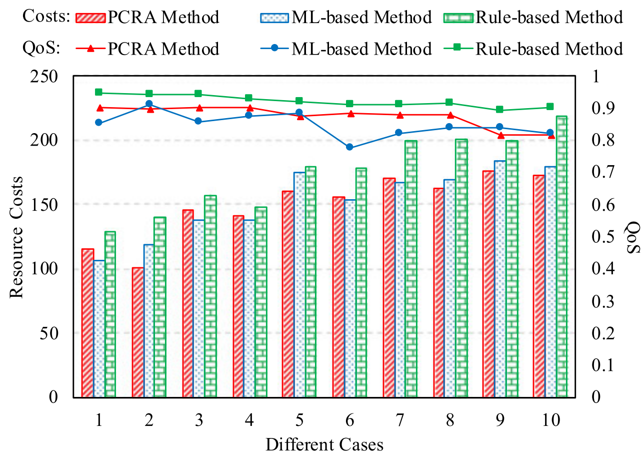

Fig. 6. Comparison of cost-effectiveness among different methods.

slight deviations occur in the decisions for management operations. Therefore, the resource allocation plans that have been obtained commonly can meet the requirements of system management.

5.4 RQ 3: Performance Improvement of the PCRA Versus Classic Approaches

Finally, we compare the performance of the proposed PCRA method with classic ML-based [29] and rule-based approaches, in order to verify the advantage of the PCRA method for resource allocation. On one hand, the classic ML-based method adopts the particle swarm optimization (PSO) algorithm to search for objective resource allocation plans based on the QoS prediction model. On the other hand, the rule-based method uses the response time as the judgment condition to take the corresponding management operations as follows.

- If RT $>$ 1.4 s, vm_L is increased by one.
- If 1.2 s $<$ RT $\leq$1.4 s, vm_M is increased by one.
- If 1.0 s $<$ RT $\leq$1.2 s, no change to VMs.
- If 0.8 s $<$ RT $\leq$1.0 s, vm_M is decreased by one.
- If RT $\leq$0.8 s, vm_L is decreased by one.

As shown in Figs. 4 and 5, for ten different cases (described in Table 7), the proposed PCRA method outperforms classic ML-based and rule-based approaches in terms of $\mathcal{F}_{obj}$ by 5$\sim$7% and 10$\sim$13%, respectively. The results show that the PCRA method is able to achieve a better trade-off between QoS and resource costs than the others. Especially, the same datasets are used to train the QoS prediction model in the ML-based approach. In only 77.2 percent of cases, the model error is less than 15 percent. This is because that the ML-based method needs a large amount of historical system data to build an accurate QoS prediction model. With the limited historical data, it might lead to inefficient resource allocation due to inaccurate QoS prediction. Moreover, the rule-based method involves the management rules that are set by experts. Thus, they are unable to effectively fit in the complex problem of resource allocation in the dynamic cloud environment. By contrast, the PCRA method is able to achieve better adaptiveness and higher efficiency for resource allocation through integrating RL and ML algorithms in the runtime environment with various workloads and service requests.

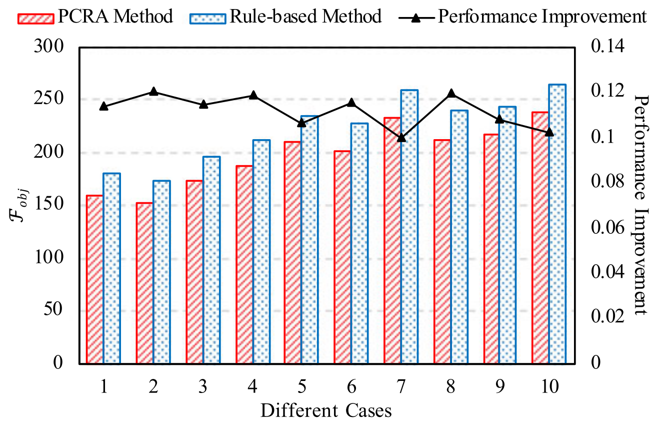

Fig. 5. Performance comparison with the rule-based method.

Furthermore, we evaluate the performance of the proposed PCRA method in terms of QoS and resource costs, respectively. As shown in Fig. 6, the rule-based method generates the best QoS in all cases, while the ML-based method produces the worst QoS in most cases (i.e., Cases 1, 3, 4, 6, 7 and 8). In general, the QoS obtained by the PCRA method lies between the other two methods. Although the rule-based method achieves the best QoS, the resource costs are also the highest cost in all cases. This is because that the rule-based method only considers the optimization of RT, but it ignores the corresponding resource costs. Meanwhile, it is hard to design the rules according to different objectives of real-world applications. By contrast, the PCRA method is able to better balance QoS and resource costs. In Cases 2, 5, 9 and 10, the PCRA method achieves lower resource costs than the other two methods, while the QoS is just slightly worse. Moreover, the PCRA method maintains relatively stable QoS values. For example, the QoS value is about 0.90 in Cases 1 to 4, about 0.88 in Cases 5 to 8, and about 0.82 in Cases 9 and 4. However, the QoS value obtained by the ML-based method is the most unstable. For instance, the QoS value is 0.91 in Case 2 and it changes to 0.78 in Case 6.

6 Conclusion

In this paper, a novel PCRA method is proposed to explore adaptive and efficient resource allocation for cloud-based software services with high QoS and low resource costs.

First, a Q-value prediction model is designed to predict the values of management operations by integrating the Q-learning algorithm and multiple ML-based learners. Next, a feedback-control based decision-making algorithm is developed for finding objective resource allocation plans. Based on the RUBiS benchmark, simulation results demonstrate the effectiveness of the proposed PCRA method in achieving optimal/near-optimal performance (QoS and resource costs) of resource allocation. More specifically, the PCRA method is able to take management operations of resource allocation with 93.7 percent correctness. Moreover, the PCRA method outperforms the classic ML-based and rule-based approaches by 5$\sim$7% and 10$\sim$13% under the runtime environment with various workloads and service requests. At present, the PCRA method is to allocate resources based on the current workload. In cloud environments, it is feasible to predict workloads for more efficient resource allocation. Therefore, our future work is to explore an adaptive strategy for resource allocation with the consideration of both the current workload and future changes of workloads.

Acknowledgments

This work was supported by the National Key R&D Program of China under Grant No. 2018YFB1004800, the Talent Program of Fujian Province for Distinguished Young Scholars in Higher Education, and the China Scholarship Council (No. 201706210072).

References

[1] C. Pahl, P. Jamshidi, and O. Zimmermann, “Architectural principles for cloud software,” *ACM Trans. Internet Technol.*, vol. 18, no. 2, 2018, Art. no. 17.

[2] M. Shojafar, N. Cordeschi, and E. Baccarelli, “Energy-efficient adaptive resource management for real-time vehicular cloud services,” *IEEE Trans. Cloud Comput.*, vol. 7, no. 1, pp. 196–209, Jan.–Mar. 2019.

[3] C. Xu, X. Ma, R. Shea, H. Wang, and J. Liu, “Enhancing performance and energy efficiency for hybrid workloads in virtualized cloud environment,” *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2018.2837040.

[4] S. Mireslami, L. Rakai, M. Wang, and B. H. Far, “Dynamic cloud resource allocation considering demand uncertainty,” *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2019.2897304.

[5] F. Zahid, A. Taherkordi, E. G. Gran, T. Skeie, and B. D. Johnsen, “A self-adaptive network for HPC clouds: Architecture, framework, and implementation,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 12, pp. 2658–2671, Dec. 2018.

[6] S. Khatua, P. K. Sur, R. K. Das, and N. Mukherjee, “Heuristic-based resource reservation strategies for public cloud,” *IEEE Trans. Cloud Comput.*, vol. 4, no. 4, pp. 392–401, Fourthquarter 2016.

[7] M. Avgeris, D. Dechouniotis, N. Athanasopoulos, and S. Papavassiliou, “Adaptive resource allocation for computation offloading: A control-theoretic approach,” *ACM Trans. Internet Technol.*, vol. 19, no. 2, 2019, Art. no. 23.

[8] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer, 2006.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[11] A. I. Orhean, F. Pop, and I. Raicu, “New scheduling approach using reinforcement learning for heterogeneous distributed systems,” *J. Parallel Distrib. Comput.*, vol. 117, pp. 292–302, 2018.

[12] A. Brandwajn, T. Begin, H. Castel-Taleb , and T. Atmaca, “A study of systems with multiple operating levels, probabilistic thresholds and hysteresis,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 4, pp. 748–757, Apr. 2018.

[13] W. Kim, J. Hong, and Y. Suh, “T-DCORAL: A threshold-based dynamic controller resource allocation for elastic control plane in software-defined data center networks,” *IEEE Commun. Lett.*, vol. 23, no. 2, pp. 198–201, Feb. 2019.

[14] Y. Xiong, S. Huang, M. Wu, J. She, and K. Jiang, “A johnson’s-rule-based genetic algorithm for two-stage-task scheduling problem in data-centers of cloud computing,” *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 597–610, Jul.–Sep. 2019.

[15] L. Zhao, J. Wang, J. Liu, and N. Kato, “Optimal edge resource allocation in IoT-based smart cities,” *IEEE Netw.*, vol. 33, no. 2, pp. 30–35, Mar./Apr. 2019.

[16] M. Ficco, C. Esposito, F. Palmieri, and A. Castiglione, “A coral-reefs and game theory-based approach for optimizing elastic cloud resource allocation,” *Future Gener. Comput. Syst.*, vol. 78, pp. 343–352, 2018.

[17] L. Jiao, J. Li, T. Xu, W. Du, and X. Fu, “Optimizing cost for online social networks on geo-distributed clouds,” *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 99–112, Feb. 2016.

[18] M. Berekmeri, D. Serrano, S. Bouchenak, N. Marchand, and B. Robu, “Feedback autonomic provisioning for guaranteeing performance in MapReduce systems,” *IEEE Trans. Cloud Comput.*, vol. 6, no. 4, pp. 1004–1016, Fourthquarter 2018.

[19] P. Haratian, F. Safi-Esfahani, L. Salimian, and A. Nabiollahi, “An adaptive and fuzzy resource management approach in cloud computing,” *IEEE Trans. Cloud Comput.*, vol. 7, no. 4, pp. 907–920, Fourthquarter 2019.

[20] N. Berthier, E. Rutten, N. De Palma, and S. M. Gueye, “Designing autonomic management systems by using reactive control techniques,” *IEEE Trans. Softw. Eng.*, vol. 42, no. 7, pp. 640–657, Jul. 2016.

[21] L. Baresi, S. Guinea, A. Leva, and G. Quattrocchi, “A discrete-time feedback controller for containerized cloud applications,” in *Proc. 24th ACM SIGSOFT Symp. Found. Softw. Eng.*, 2016, pp. 217–228.

[22] 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.

[23] 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.

[24] A. Ali-Eldin, J. Tordsson, and E. Elmroth, “An adaptive hybrid elasticity controller for cloud infrastructures,” in *Proc. 10th Netw. Operations Manage. Symp.*, 2012, pp. 204–212.

[25] M. Ranjbari and J. A. Torkestani, “A learning automata-based algorithm for energy and SLA efficient consolidation of virtual machines in cloud data centers,” *J. Parallel Distrib. Comput.*, vol. 113, pp. 55–62, 2018.

[26] L. Tsai, H. Franke, C. Li, and W. Liao, “Learning-based memory allocation optimization for delay-sensitive big data processing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 6, pp. 1332–1341, Jun. 2018.

[27] Z. Chen, J. Hu, G. Min, A. Zomaya, and T. El-Ghazawi, “Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 923–934, Apr. 2020.

[28] L. Wei, C. H. Foh, B. He, and J. Cai, “Towards efficient resource allocation for heterogeneous workloads in IaaS clouds,” *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 264–275, Firstquarter 2018.

[29] X. Chen, J. Lin, B. Lin, T. Xiang, Y. Zhang, and G. Huang, “Self-learning and self-adaptive resource allocation for cloud-based software services,” *Concurrency Comput. Pract. Experience*, vol. 31, no. 23, 2019, Art. no. e4463.

[30] J. Wang *et al.*, “A machine learning framework for resource allocation assisted by cloud computing,” *IEEE Netw.*, vol. 32, no. 2, pp. 144–151, Mar./Apr. 2018.

[31] N. Liu *et al.*, “A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning,” in *Proc. 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 372–382.

[32] A. Alsarhan, A. Itradat, A. Y. Al-Dubai, A. Y. Zomaya, and G. Min, “Adaptive resource allocation and provisioning in multi-service cloud environments,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 1, pp. 31–42, Jan. 2018.

[33] Z. Chen, J. Hu, and G. Min, “Learning-based resource allocation in cloud data center using advantage actor-critic,” in *Proc. 53rd Int. Conf. Commun.*, 2019, pp. 1–6.

[34] A. Soltanian, D. Naboulsi, R. Glitho, and H. Elbiaze, "Resource allocation mechanism for media handling services in cloud multimedia conferencing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1167–1181, May 2019.
[35] I. V. Paputungan, A. F. M. Hani, M. F. Hassan, and V. S. Asirvadam, "Real-time and proactive SLA renegotiation for a cloud-based system," *IEEE Syst. J.*, vol. 13, no. 1, pp. 400–411, Mar. 2019.
[36] G. C. Berresford and A. M. Rockett, *Brief Applied Calculus*. Boston, MA, USA: Cengage Learning, 2015.
[37] N. Sabharwal, *Apache CloudStack Cloud Computing*. Birmingham, U.K.: Packt Publishing, 2013.
[38] "RUBiS: Rice university bidding system benchmark," 2013. Accessed: Mar. 15, 2019. [Online]. Available: http://rubis.ow2.org/
[39] X. Chen, H. Wang, Y. Ma, X. Zheng, and L. Guo, "Self-adaptive resource allocation for cloud-based software services based on iterative QoS prediction model," *Future Gener. Comput. Syst.*, vol. 105, pp. 287–296, 2020.
[40] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Des. Implementation*, 2016, vol. 16, pp. 265–283.

Xing Chen received the BS and PhD degrees in computer software and theory from Peking University, Beijing, China, in 2008 and 2013, respectively. He is currently an associate professor and the deputy director of Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, China, and leads the Systems Research Group. He has authored or coauthored more than 30 journal and conference articles. His research interests include the software systems and engineering approaches for cloud and mobility. His current projects cover the topics from self-adaptive software, computation offloading, model-driven approach, and so on. He was recipient of the first Provincial Scientific and Technological Progress Award, in 2018.

Fangning Zhu received the BS degree in software engineering from Fuzhou University, Fujian, China, in 2018. She is currently working toward the MS degree in computer software and theory in the College of Mathematics and Computer Science, Fuzhou University, China. Her current research interests include system software, edge computing, and cloud computing.

Zheyi Chen received the BSc degree in computer science from Shanxi University, China, in 2014, and the MSc degree in computer science from Tsinghua University, China, in 2017. He is currently working toward the PhD degree in computer science at the University of Exeter, United Kingdom. His research interests include cloud computing, mobile edge computing, deep learning, and resource optimization.

Geyong Min received the BSc degree in computer science from the Huazhong University of Science and Technology, China, in 1995, and the PhD degree in computing science from the University of Glasgow, United Kingdom, in 2003. He is a professor of high performance computing and networking with the Department of Computer Science within the College of Engineering, Mathematics and Physical Sciences, University of Exeter, United Kingdom. His research interests include future Internet, computer networks, wireless communications, multimedia systems, information security, high-performance computing, ubiquitous computing, and modelling and performance engineering.

Xianghan Zheng received the MSc degree in distributed system and the PhD degree in information communication technology from the University of Agder, Norway, in 2007 and 2011, respectively. He is currently a professor with the College of Mathematics and Computer Sciences, Fuzhou University, China. His current research interests include new generation network with a special focus on cloud computing services and applications, and big data processing and security.

Chunming Rong (Senior Member, IEEE) is a professor and the head of the Center for IP-based Service Innovation (CIPSI) at the University of Stavanger (UiS), Norway. He is the chair of IEEE Cloud Computing and an executive member of Technical Consortium on High Performance Computing (TCHPC) and the chair of STC on Blockchain in IEEE Computer Society, and served as global co-chair of IEEE Blockchain, in 2018. He is also advisor of the StandICT.EU to support European scandalization activities in ICT. He is also a co-founder of two start-ups bitYoga and Dataunitor in Norway, both received EU Seal of Excellence Award, in 2018. He was adjunct senior scientist leading Big-Data Initiative at NORCE (2016–2019), the vice president of CSA Norway Chapter (2016–2017). His research work focuses on cloud computing, data analytics, cyber security and blockchain. He is honoured as member of the Norwegian Academy of Technological Sciences (NTVA) since 2011. He has extensive contact network and projects in both industry and academic. He is also the founder and steering chair of IEEE CloudCom conference and workshop series. He is co-editors-in-chief of the *Journal of Cloud Computing* (ISSN: 2192-113X) by Springer, has served as the steering chair (2016–2019), steering member and associate editor of the *IEEE Transactions on Cloud Computing (TCC)* since 2016. He has supervised 26 PhDs, nine PostDocs and more than 60 master projects. He has extensive experience in managing large-scale R&D projects, both in Norway and EU.