A Reinforcement Learning based Approach to Identify Resource Bottlenecks for Multiple Services Interactions in Cloud Computing Environments

Lingxiao Xu[1], Minxian Xu[2](✉), Richard Semmes[3], Hui Li[3], Hong Mu[3], Shuangquan Gui[3], Wenhong Tian[1](✉), Kui Wu[4], and Rajkumar Buyya[1,5]

[1] School of Software and Information Engineering, University of Electronic Science and Technology of China, China
[2] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
[3] Siemens Industry Software (Chengdu) Co., Ltd , China
[4] Department of Computer Science of University of Victoria, Canada
[5] CLOUDS Lab, School of Computing and Information Systems, University of Melbourne, Australia

Abstract. Cloud service providers are provisioning resources including a variety of virtual machine instances to support customers that migrate their services to the cloud. From the customers' perspective, selecting the appropriate amount of resources is tightly coupled with performance and cost. By identifying the potential resource bottlenecks in the early stage of the service deployment process, resource planning can be significantly optimized. However, due to the unpredictable workloads and heterogeneous resources, it is difficult to identify resource bottlenecks that can degrade system performance. To support system non-functional requirements (NFR) in a better manner, we propose a reinforcement learning based approach to support the NFR management of system concerning the multiple services interactions scenario by identifying the potential resource bottleneck and optimizing the demanded resources. The proposed approach can predict the resource bottleneck for multiple services interactions, e.g. bottleneck in CPU or overloads in specific service, and provide guidance for resource planning. We modeled and simulated the proposed approach using an extended version of the CloudSim toolkit. Comprehensive evaluations with realistic use case from Siemens Digital Industries Software's MindSphere Solution on AliCloud show that our proposed approach can achieve high accuracy in terms of performance metrics, such as response time, queries per second (QPS), and resource usage.

Keywords: Cloud Computing · Reinforcement Learning · Service Interactions · Non-functional Requirement · Resource Bottleneck

Minxian Xu and Wenhong Tian are corresponding authors

1 Introduction

The rapid development of cloud computing has made it be regarded as the fifth utility, like electricity, gas, and water [4]. Rather than assigning all tasks to a single local computer or a traditional computer cluster, cloud computing enables users to utilize computing or storage resources remotely, which provisions transparent and on-demand resources. In essence, the cloud is a networked computer paradigm based on virtualization techniques to improve resource usage. The pay-as-you-go model provided by cloud computing also helps the service providers and customers to start their business with minimal costs and eliminates the efforts to maintain the data centers [10].

Among all the benefits provided by cloud computing, some features are particularly attractive for customers. To be more specific, the first one is flexibility, which allows customers to acquire or release resources dynamically to fit their demands. The second is cost reduction, which means cloud computing service can convert capital expenditures to operational expenditures [23]. By utilizing the cloud, expensive infrastructures like servers and professionals will not be the main concern of customers anymore. And the third is the device and location independence, which supports customers to access computing resources anywhere and anytime, instead of using the specific interfaces to access the local machines and avoiding unavailability due to machine maintenance. Benefiting from these advantages, many customers have migrated their business to the cloud, for instance, Alibaba has announced a statement that their services should be ”All in Cloud” [6]. Currently, the prominent IT companies such as Amazon, Google, Microsoft, and Alibaba, have established their own cloud data centers and become cloud providers.

Though cloud computing has significant advantages, based on the practice of utilizing resources of cloud computing, resource bottlenecks can still happen occasionally, like in CPU, memory, and bandwidth [24]. The reason is that when customers set up their services, they need to estimate how much of the resources will be used for their services and reserve specific amount of resources. However, due to the fluctuations of loads, overloads can occasionally happen and lead to bottlenecks. When bottlenecks exist, the system performance, such as non-functional requirement (NFR) can be significantly influenced [20], for instance, CPU bottleneck can cause insufficient computing power and reduced number of parallel processes, memory bottleneck can lead to insufficient tasks or data waiting to be processed [27], and bandwidth bottleneck can trigger requests failed to be responded.

As bottlenecks can greatly affect the performance and user experience of cloud services [11], it is important to identify the potential bottlenecks. However, different cloud services often provision various resource capacity, such as VM instances with different CPU, memory, and bandwidth, thus they may have bottlenecks under diverse load conditions. Besides, the bottlenecks of individual services are more difficult to be predicted based on divergent resource demands, running status and internal logic. If service providers can identify the potential bottleneck of each service and modify configurations, the NFR can be satisfied.

However, identifying the bottlenecks at the early stage is not easy. It's not feasible that arranging testers to perform a large number of stress tests to evaluate the bottlenecks when changing resource configurations. Apart from it, purchasing much more resources than required is not cost-effective and evaluation results may not be reproducible due to uncontrolled factors, such as network traffics.

To deal with the above challenges, using a simulation toolkit to simulate the real environment is a promising way. In this paper, we use CloudSim [5], which is a well-known cloud simulation toolkit that enables seamless modeling, simulation and experimentation of cloud computing. With CloudSim, we can model the resource provisioning of various cloud infrastructure configurations and generate reproducible results. Large-scale simulations can also be easily conducted. Our motivation is to predict the potential resource and service bottlenecks of a sample use case that is based on Siemens Digital Industries' MindSphere solution [16],which is a cloudbased Internet-of-Things (IoT) open operating system. We aim to generate various metrics to help plan hardware resources for the use case. We also make efforts to ensure our approach as generic as possible in order to extend it other cloud platforms with ease. To achieve these goals, we also have some challenges to address, including how to build a model for realistic scenarios, how to ensure that the system output results are consistent with the real test results, and how to make it a generic approach.

In this paper, we propose a data-driven framework to support the non-functional requirement, e.g. system performance, for multiple services interactions to identify the potential bottlenecks of service in the early stage. A policy gradient approach based on reinforcement learning is also proposed to predict the potential system bottleneck based on data collected from real test cases and guide companies to optimize the resource configuration. Additionally, the approach can also help to predict the response time and QPS when the system has reached the bottleneck.

Our key **contributions** are as follows:

- Presented a framework to identify the potential bottleneck for multiple services interacted in the cloud to optimize resource planning for companies.
- Proposed a policy gradient approach to model resource utilization and predict the system behaviors under different loads.
- Utilized CloudSim components to setup specific underlying IaaS infrastructure model, specific MindSphere service model, and realized performance test simulation.

The rest of the paper is organized as: we start by discussing the related work in Section 2, where we highlight the differences between our work and existing work. In Section 3, we introduce our proposed framework, named IRBS, for bottleneck identification. Then we discusses the modeling of multiple services interactions scenario and introduce our proposed reinforcement learning based approach for bottleneck prediction in Section 4. Afterward, the evaluation results based on the sample scenario are demonstrated in Section 5. Finally, conclusions and future work are given in Section 6.

https://siemens.mindsphere.io/en

2 Related work

To utilize cloud service more efficiently and reduce the cost, it is important to model cloud services and scheduling process and optimize resources. Some work has been done to model services and optimize resources in Clouds. Neeraja et al. developed a data-driven system named PARIS, which can predict workload performance, resulting in costs and workloads across multiple cloud providers [26]. To decrease the task execution failure, Lattif et al. proposed the DCLCA (dynamic clustering league championship algorithm) scheduling technique for fault tolerance awareness to address cloud task execution which would reflect on the currently available resources and reduce the untimely failure of autonomous tasks [13]. Sekaran et al. presented a new meta-heuristic algorithm, named the dominant firefly algorithm, which can optimize load balancing of tasks among the multiple virtual machines in the Cloud server, thereby improving the response efficiency of Cloud servers that concomitantly enhances the accuracy of m-learning systems [18]. Cheng et al. introduced DRL-Cloud, a novel Deep Reinforcement Learning (DRL)-based RP and TS system, to minimize energy cost for large-scale CSPs with a very large number of servers that receive enormous numbers of user requests per day [7]. Nayak et al. used AHP (Analytic Hierarchy Process) as a decision-maker in the backfilling algorithm to choose the possible best lease from the given best-effort queue to schedule the deadline sensitive lease [15]. Priya et al. constructed a Fuzzy-based Multidimensional Resource Scheduling model to obtain resource scheduling efficiency in cloud infrastructure and increased utilization of Virtual Machines through effective and fair load balancing are then achieved by dynamically selecting a request from a class using Multi-dimensional Queuing Load Optimization algorithm [17]. These work can optimize resource usage, however, their objectives are not identifying the potential bottleneck of resources and services in the system.

In order to generate reproducible results and simulate a large-scale cloud environment, CloudSim is frequently used. For instance, Wickremasinghe et al. developed CloudAnalyst, which is a CloudSim based tool for simulating large-scale Cloud applications to study the behavior of such applications under various deployment configurations [21]. Shi et al. used CloudSim to develop efficient energy-saving methods to reduce the huge energy consumption in the cloud datacenter [19]. Jung et al. proposed a simulation tool that supports the MapReduce model, implemented on CloudSim [12]. Belalem et al. investigated two approaches that aim at returning a better availability of data centers without deteriorating the performances for the answers of the users [2]. Alla et al. presented Task Scheduling optimization using a novel approach based on Dynamic dispatch Queues (TSDQ) and hybrid meta-heuristic algorithms, which is based on CloudSim and showed a great advantage in terms of waiting time, queue length, makespan, cost, resource utilization, degree of imbalance, and load balancing [1]. Sharma et al. demonstrated a modified particle swarm optimization (MPSO) task scheduling algorithm in order to optimize execution time, transmission time, makespan, transmission cost, and load balancing of virtual machines and got the best cost as compared to original PSO on the CloudSim [3].

However, these works do not model service internal logic. Besides, they do not provide the model for the intended sample scenario and use cases which will be elaborated in Section 4.

Table 1: Comparison of related work

Approach	Resource Optimization	Service Modelling	Bottleneck Identification	Multiple Services Interactions Modelling	Evaluations with CloudSim Extension	Sample Scenario
Neeraja et al. [26]	✓	✓				
Latif et al. [13]	✓	✓				
Sekaran et al. [18]	✓	✓				
Cheng et al. [7]	✓	✓				
Nayak et al. [15]	✓	✓				
Priya et al. [17]	✓	✓				
Wickremasinghe et al. [21]	✓				✓	
Shi et al [19]	✓				✓	
Jung et al. [12]	✓				✓	
Alla et al. [1]	✓				✓	
Sharma et al. [3]	✓				✓	
Our Approach	✓	✓	✓	✓	✓	✓

We compare our approach with other related work from multiple perspectives in Table 1. Our work mainly contributes to the current research area by providing an approach for identifying the resource and service bottlenecks for cloud system, it also models the multiple services interactions for the sample scenario. The proposed approach can be further applied to other related scenarios and service providers.

3 IRBS Framework

To make our proposed approach to be reusable, correct and universal, we propose a framework to identify resource and service bottlenecks for multiple services interactions in Cloud, named IRBS (Identifying Resource Bottlenecks Solution). IRBS can evaluate different kinds of cloud services to achieve their availability and scalability. IRBS aims to predict the bottlenecks of the cloud service and identify the metrics when the service has reached the bottleneck. From the companies' perspective, selecting the appropriate amount of resources is tightly coupled with performance and cost, and bottlenecks are the important metric to improve the rationality of resource allocation. We provide a high-level summary below and then discuss the details in the subsequent sections.

We aim to make generic design, thus IRBS should be applicable to cloud services of different specifications by changing configurations. We also target to fit the data to achieve high accuracy. Furthermore, when a new service is integrated into the system, our framework should also adapt it in a good manner.

The framework of IRBS is shown in Fig. 1, it contains five components: Input Controller, Resource Usage Model, Data Fitting Model, Queue Processing Model, Multiple Services Interactions Modelling.

The *Input Controller Component* handles the input workloads. The workloads can be generated by this component, then the concurrent loads can be

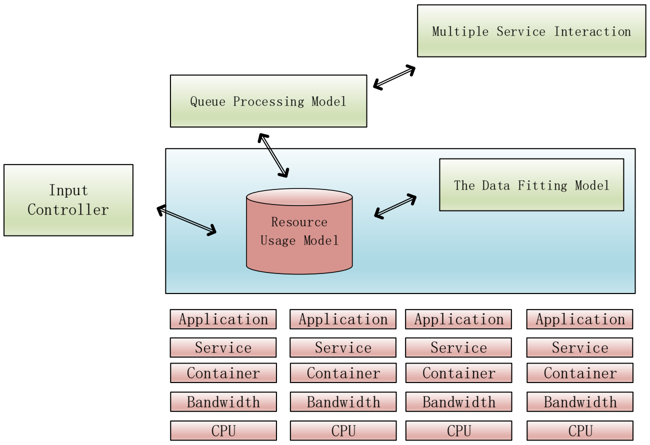


Fig. 1: Framework of IRBS

processed by other components in the framework. The characteristic of loads can be defined in the input property files, including the number of loads, duration, ramp-up time etc., which will be loaded firstly. In our framework, we consider the online scheduling policy to fit the realistic scenario, and the loads are generated as per time slot, e.g. 1 second.

The *Resource Usage Model Component* is the key component of IRBS. It produces the resource pool containing the CPU pool and bandwidth pool, then simulates the real scenario to perform the distribution of loads. Based on this, results including real-time CPU usage, the real-time bandwidth usage, the initial response time can be obtained and modelled. More details will be provided in Section 4.

The *Data Fitting Model Component* supports to calculate the evaluation results. In the real scenario, the hardware, temperature, voltage, and other factors can have significant impacts on cloud service performance. As for the software, the scheduling policy and the adopted cloud service can also influence the NFR. This component provides the approach to train the prediction model based on the actual data.

Queue Processing Model controls two queues by monitoring the data in Resource Usage Model, which named the deferral queue and processing queue. It is responsible for managing the workloads in the system and collaborating with the components in the system to schedule the resources. The deferral queue stores the loads that are deferred when the resource pool is full, and the processing queue stores the loads under processing. The component notifies the Input Controller about the total loads in the system and then adjusts the generated loads dynamically based on resource usage.

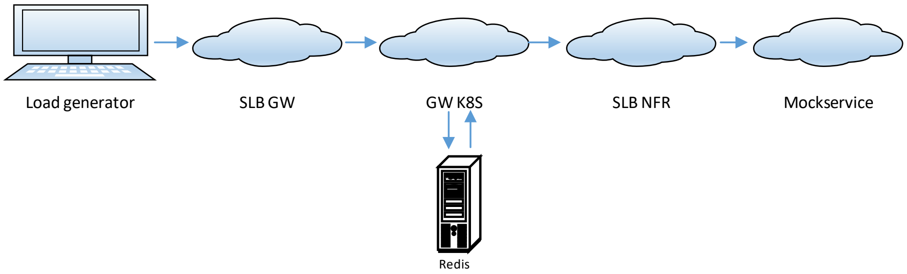


Fig. 2: The sample scenario for Simulation

The *Multiple Services Interactions Component* manages the interactions and collaborations between services. It can also represent the flow of workloads. Considering the collaboration of different services that if one service reaches its bottleneck, the other service will be affected, so IRBS designs the multiple service interactions to deal with the situation. By querying the Queue Processing Model, the component can obtain the previous service queue status and notify the next service, then achieve better resource optimization effects.

4 Problem Modelling and Proposed Solution

In this section, we will formulate our problem modeling for identifying the system bottlenecks by predicting the system resource usage. We then present our proposed approach based on policy gradient in reinforcement learning to solve the problem.

4.1 System Model

To support the understanding of our system model, the sample scenario is shown in Fig. 2, which mainly consists of 6 services collaborating together to support the scenario. The load generator is the service that generates loads for the system. SLB GW is used as the task scheduler (GW), which is responsible for maintaining the load balancing and performing task distribution. GW K8S is the service under test which is also the core component simulated in the whole system model, which will also refer to the simulation of the interatction between GW K8S and its backing database(Redis). SLB NFR is used to monitor various metrics in the entire system, and Mockservice is a black box component that generates as fixed processing delay component, e.g. 300 ms. Considering the different functionalities of these services, the services consume different amount of resources, and there are interactions between the services. Our objective is to identify the bottlenecks in the system resulted from CPU usage and bandwidth usage. A more general system model is introduced as follows.

Assuming the whole observation time period is T, and at time slot t, $t \in \{1, 2, \ldots, T\}$, the number of load requests is L_t. We consider that there are

M services collaborating together to support the sample scenario by providing services for the customers. These services can be deployed on public cloud, e.g. Alicloud, which provisions N ECS (Elastic Computing Service) VM nodes and I pods (containers). The pods are deployed on ECS. For ECS node E_n, $n \in \{1, 2, \ldots N\}$, it can have E_n^c cores. And for pod P_i, $i \in \{1, 2, \ldots, I\}$, it can have P_i^c cores. The cores represent the capacity to process the load requests coming into the system. The CPU utilization of each ECS and pod at time slot t can be represented as $E_n^u(t)$ and $P_i^u(t)$ respectively. For service S_m, $m \in \{1, 2 \ldots, M\}$, considering I_m pods are provided for it, thus the CPU utilization of service $S_m^u(t)$ can be represented:

$$S_m^u(t) = \frac{1}{I_m} \sum_{i=1}^{I_m} P_i^u(t) \tag{1}$$

CPU utilization is one of the key resource bottleneck that we would like to investigate.

Besides CPU utilization, we also consider the bandwidth resource bottleneck. Based on the obtained data, the bandwidth between different services can also experience bottlenecks due to the communications between services. For example, in Fig. 2 the requests are processed and distributed via the SLB GW service and then the requests are processed by GW K8S, where the bottleneck can exist due to the high volume of requests.

Considering services S_m and S_k, $m, k \in \{1, 2 \ldots, M\}$, are collaborating with each other. The bidirectional network traffics at time slot t can be represented as $B_{S_m,S_k}(t)$ and $B_{S_k,S_m}(t)$, where $B_{S_m,S_k}(t)$ represents the communication is from service S_m to S_k, and vice verse.

We use m^* to denote the id of the service with the highest CPU utilization, which can be calculated as:

$$m^* = \arg\max_{m,t} \{S_m^u(t)\} \tag{2}$$

Compared with the predefined CPU utilization threshold T^c, when $S_{m^*}^u(t) \geq T^c$, it means the bottleneck exists in service m^*. Similarly, the communication with the highest bandwidth utilization between services can be represented as $B_{S_{m^*},S_{k^*}}(t)$, where the m^* and k^* can be calculated as:

$$\{m^*, k^*\} = \arg\max_{m,k,t} \{B_{S_m,S_k}(t)\} \tag{3}$$

Assuming that the bandwidth threshold from service m^* to k^* is T_{m^*,k^*}^b, the bandwidth bottleneck exists when $B_{S_{m^*},S_{k^*}}(t) \geq T_{m^*,k^*}^b$.

Some sample data for the Gateway service are shown in Table 2, which includes the information on load, pod utilization, throughput, average response time and bandwidth. Based on these data, we can define a given set of J input and output data as $D = \{(\boldsymbol{x_i}, y_1), (\boldsymbol{x_2}, y_2), \ldots, (\boldsymbol{x_J}, y_J)\}$, and $X = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_J}\}$ and $Y = \{y_1, y_2, \ldots, y_J\}$ are the set of input and output data.

In our reinforcement learning based approach, we would like to predict some key metrics that can represent the system bottlenecks, e.g. CPU utilization and

Table 2: Performance metrics of Realistic Scenario

Case	**Load**	**Pod CPU utilization**	**Throughput**	**Average response Time**	**Bandwidth**
1	1,500 vu	38.5 cores	10,000 calls/s	50 ms	300 Mbps
2	2,000 vu	46 cores	11,300 calls/s	75 ms	342 Mbps
3	3,000 vu	47 cores	11,600 calls/s	158 ms	352 Mbps

bandwidth. The predicted metrics can be calculated by $\hat{y}_i = f(\boldsymbol{x_i})$, where $f(x)$ is the trained model by policy gradient approach. In our proposed approach, we aim to reduce the error between the predicted data and actual data, which can be evaluated by metrics such as Root Mean Square Error, Mean Relative Error, and R^2.

4.2 Policy Gradient Approach

Reinforcement Learning [14] is a data-driven approach for adaptively applying optimized control policies based on real-time feedback, which models the stochastic process under the framework of Markov Decision Process (MDP) [25]. Policy gradient [9] is one of the most common types of reinforcement learning algorithms. In the policy gradient approach, the optimal actions with model parameters can be learned directly. The control actions are selected as the ones that can maximize the system rewards. In this section, we exploit the policy gradient to seek the optimal policy.

In our problem, the approach will decide the amount of resources that should be allocated to each request load at each time slot. The key entities in our approach are defined as follows.

States: The system state $S(t)$ includes the status of (1) amount of loads $L(t)$, (2) CPU capacity for processing loads $W_c(t)$ (3) bandwidth capacity for processing loads $W_b(t)$, (4) CPU resource usage $U_c(t)$, and (5) bandwidth resource usage $U_b(t)$. The state space of $L(t)$ at time t is given as $L(t) < \overline{L} \in \mathbb{Z}^+$, where $\overline{L}$ is the maximum number of loads that the system can accommodate and $\mathbb{Z}^+$ is a set of non-negative integers. $W_c(t) \in \mathbb{R}^+$ and $W_b(t) \in \mathbb{R}^+$ represent the capacity to process loads based on CPU (per core) and bandwidth (per Mbps) respectively, where $\mathbb{R}^+$ is a set of non-negative real numbers. The CPU resource usage $U_c(t) \in [0, 1]$ and bandwidth resource usage $U_b(t) \in [0, 1]$ are obtained from system running status. Therefore, the system state at time t, $S(t)$, can be denoted by:

$$S(t) \triangleq [L(t), W_c(t), W_b(t), U_c(t), U_b(t)] \in \mathbb{S}, \tag{4}$$

where $\mathbb{S}$ stands for all possible states.

Actions: At the beginning of each time slot, the approach determines the actions to increase or decrease $W_c(t)$ and $W_b(t)$ to decide the CPU and bandwidth capacity to process loads. The next state $S(t+1)$ depends on the current state $S(t)$ by taking action $A(t) \triangleq [A_c(t), A_b(t)] \in \mathbb{A}$, where $A_c(t) \in (0, +\infty)$ and $A_b(t) \in (0, +\infty)$ are the corresponding actions for CPU capacity and bandwidth capacity, and $\mathbb{A}$ stands for all possible actions. Thus, the CPU and bandwidth capacity for processing loads at $t+1$ can be determined according to the following equations:

$$W_c(t+1) = A_c(t) \cdot W_c(t) \tag{5}$$

$$W_b(t+1) = A_b(t) \cdot W_b(t) \tag{6}$$

Rewards: At each unit time, the process is in a state $S(t)$, and we choose a possible action $A(t)$. The process randomly moves to the next state $S(t+1)$ at the next time slot, and gives the corresponding reward is $r(S(t), A(t))$. Our model concerns the Euclidean Distance. After adopting the action $A(t)$, the $U_c(t)$ and $U_b(t)$ that are received from the system, and the errors between the actual values and predicted values are accumulated. We set the cumulative errors of two consecutive actions, e.g.

$$r(t) = D(t-1) - D(t), \tag{7}$$

where $D(t-1)$ and $D(t)$ are the total cumulative errors in the current and previous time slots. The total cumulative errors at time slot t, is the summation of the cumulative errors of all actions from $t = 0$ to the current time slot. The positive reward values imply that actions are taken to decrease the total cumulative errors and the negative rewards imply an increase. With regard to the reward values, the actions can be taken to change the current state to other certain states in future time slots.

Objective Function: The approach chooses the actions based on policy π, which is defined as the mapping from the input state to a probability distribution over actions $\mathbb{A}$. We use parameters θ as policy parameters, and the policy distribution $\pi(A(t)|S(t);\theta)$ is learned by performing gradient descent on the policy parameters. In this research, we aim to maximize the reduction of cumulative errors, which represent the accuracy of resource utilization and bottleneck identification. The objective to maximize the reward values under the probability distribution $\pi(A(t)|S(t);\theta)$ can be denoted as:

$$J(\theta) = E_{\pi_\theta}[\sum_{t=0}^{T} \gamma^t r(t)] = E_{\pi_\theta}[R]. \tag{8}$$

Where γ is the discount factor, and R is defined as the sum of future discounted rewards.

In policy gradient approach, the gradient of the objective function in Equation (8) can be given by:

$$\bigtriangledown_\theta J = \sum_{t=0}^{T} E_{\pi_\theta}[\bigtriangledown_\theta log(A(t)|S(t);\theta)R_t]. \tag{9}$$

Algorithm 1 Policy gradient based reinforcement learning approach

```
Input: Reachable system states $S(t) \in \mathbb{S}$, observation time period $T$, possible actions
    $A(t) \in \mathbb{A}$, initialized parameters $\theta$ and $\theta_v$, the maximum step $K$, discount factor
    $\gamma$, and learning rate $\alpha$
Output: $\theta$, $\theta_v$
 1: for $t$ from 1 to $T$ do
 2:   for $S(t) \in \mathbb{S}$ do
 3:     perform action $A(t)$ according to policy $\pi(A(t)|S(t);\theta)$
 4:     obtain reward $R(t)$ and next state $S(t+1)$
 5:     if $S(t)$ is terminal state or $T - t = K$ then
 6:       $R = 0$
 7:       break;
 8:     else
 9:       $R = V(S(t);\theta_v)$
10:     end if
11:     for $t^{'}$ from $t$ to $t + K$ do
12:       $R \leftarrow R(t^{'} + \gamma R)$
13:       $\theta \leftarrow \theta + \alpha \triangledown_{\theta} log(A(t^{'})|S(t^{'});\theta)(R - V(S(t^{'});\theta_v))$
14:       $\theta_v \leftarrow \theta_v + \frac{\partial(R - V(S(t^{'});\theta_v))^2}{\partial\theta_v}$
15:     end for
16:   end for
17: end for
```

The equation can update the policy parameters θ in the direction $\bigtriangledown_{\theta} log(A(t)|S(t);\theta)$ in order to increase the probability of action $A(t)$ at state $S(t)$ if the action can increase the cumulative reward values, and vice verse.

By utilizing actor-critic approach [9], the parameters update can be done by selecting actions to achieve the terminal state or taking K steps. Therefore, the parameters θ can be updated through the following equation:

$$\theta = \theta + \alpha \sum_{t} \bigtriangledown_{\theta} log(A(t)|S(t);\theta) V(S(t), A(t); \theta, \theta_v), \tag{10}$$

where α is the learning rate, $V(S(t), A(t); \theta, \theta_v)$ is the estimate of advantage function [14] comparing the state value function $V^{\pi_{\theta_v}}(S(t))$ of current state and the state after K steps.

The pseudocode of our proposed algorithm is shown in Algorithm 1. The algorithm is based on policy gradient to maximize the reward values, which controls by the actions. With the inputs of reachable states, possible actions, and predefined system parameters, the algorithm iterates over time slots from periods 1 to T. Line 3 explores the possible actions under policies and the expected reward of a state is calculated in line 4. The future discounted rewards calculation is performed by lines 5 to 10. And lines 12-14 update the parameters.

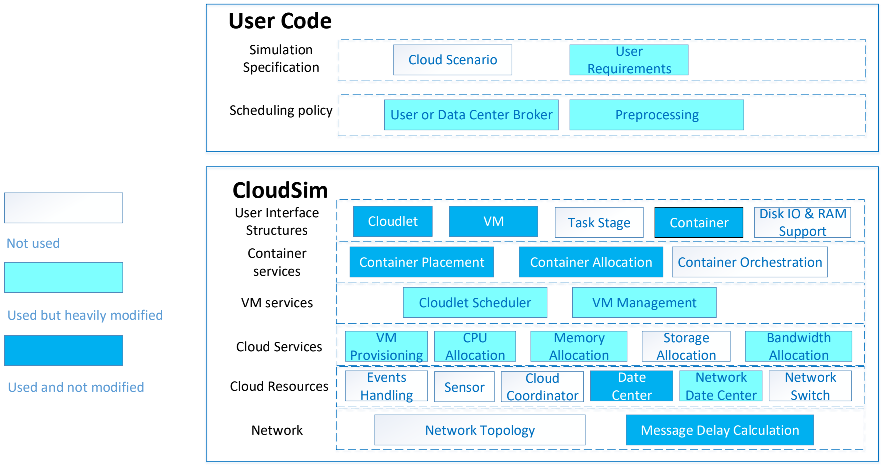


Fig. 3: Extension of CloudSim Components of this Scenario

5 Performance Evaluations

In this section, we introduce our experimental environment, including the extension of CloudSim, environment settings, two scenarios under different loads, convergence analysis, performance evaluations, bottleneck prediction, and scalability analysis.

5.1 Extension of CloudSim for this scenario

By following the simulation methodology of CloudSim 4.0, we have re-implemented some existing modules such as cloudlet, container, containerVM, and host. We add the queueing method and apply the resource scheduling approach to design our scheduling policy. The major modified components are shown in Fig. 3.

Considering the scenario, there is a mechanism to support admission control based on system feedbacks, which CloudSim does not have. And the scheduling logic of CloudSim is offline, in which it submits all the cloudlets firstly, and processes them later. This policy does not support the feedback mechanism. Additionally, we make improvements based on our objectives to make the scheduling more suitable for our scenario, and finally realizing our target scenario. Some components are kept as they are, e.g. CloudSim has virtual machine migration and container migration in resource scheduling, which are not suitable for our scenario.

5.2 Environment Settings

In our simulations, the time accuracy can be adjusted to make it more fine-grained. However, if the accuracy is improved, the simulation takes longer run-

Table 3: Infrastructure configuration

Case	Load (per sec.)	ECS number	Pod number	ECS cores	Pod cores	Bandwidth	Other service response time
1	1,500	9	18	8 cores	3 cores	350 Mbps	about 100 ms
2	2,000	9	18	8 cores	3 cores	350 Mbps	about 100 ms
3	3,000	9	18	8 cores	3 cores	350 Mbps	about 100 ms
4	3,020	20	40	8 cores	3 cores	650 Mbps	about 100 ms
5	4,000	16	32	8 cores	3 cores	800 Mbps	about 100 ms
6	4,000	20	40	8 cores	3 cores	610 Mbps	about 100 ms

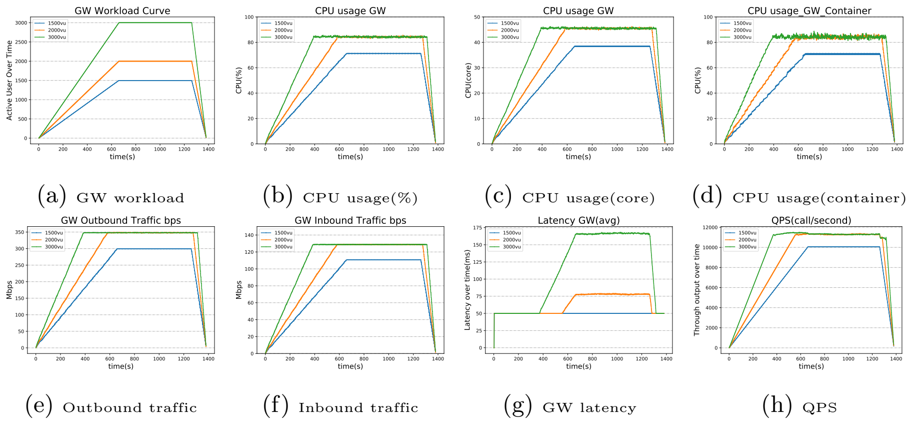
ning time and consumes more memory resources. To balance the trade-offs between accuracy and running costs, we simulate and collect the CPU usage, bandwidth usage every second. And then, we draw the results based on the obtained metrics. We use the policy gradient algorithm introduced in Section 4.2 to train the resource usage parameters in our model, and the simulations are conducted for the situation under this configuration.

Based on the results, we can know when the bottleneck will exist under specific loads. Then we can adjust the configuration files according to the corresponding loads to optimize resource planning. In this way, the number of the loads when CPU resources or the bandwidth resources reach the bottleneck can be identified under this configuration. Additionally, we can use IRBS to predict some scenarios without evaluations under real tests, for example in what CPU configuration or bandwidth configuration the QPS will reach 20,000 calls/s, and optimize resource planning without changing hardware specifications, only via changing the cores or the maximum bandwidth.

In the scenario, GW K8S is the core component to be evaluated, as the whole data is confidential, only the key data is shown in Table 2. In the configurations, there are 9 ECS and 18 pods in total for the resource provisioning. Each ECS is equipped with 8 cores, each pod has 3 cores, and the bandwidth is configured 350 Mbps. The detailed configurations are shown in Table 3.

To evaluate the performance of our approach to verify system performance under different conditions, we firstly conduct experiments for underutilized scenario and overutilized scenario, which represent the system is running without bottlenecks and with bottlenecks. The discount factor γ in Algorithm 1 is configured as 0.99, and the learning rate α is set as 0.1. The following metrics are adopted to evaluate the performance, which are also key metrics that company would like to investigate:

- GW workload: number of input loads per second for GW service
- CPU usage: CPU usage base on percentage and number of cores at each time slot
- Outbound traffic: bandwidth usage in the outbound direction
- Inbound traffic: bandwidth usage in the inbound direction
- GW latency: response time of GW
- QPS: Queries per second that system can accommodate



(a) GW workload (b) CPU usage(%) (c) CPU usage(core) (d) CPU usage(container)

(e) Outbound traffic (f) Inbound traffic (g) GW latency (h) QPS

Fig. 4: The metrics under different load conditions

5.3 Underutilized Scenario

The underutilized scenario demonstrates the scenario when both the CPU and the inbound and outbound traffic have not reached the bottleneck, and all loads can respond in time. The results are demonstrated in Fig. 4. In this scenario, the load is about 1500 vu (virtualized users) per second at peak, and the loads grow gradually until 660s and then keep stable until about 1200s, and finally dropping to 0. During this period, the maximum CPU resource usage is about 70%, the used cores are about 38.5, the output bandwidth is about 300 Mbps, the inbound traffic is about 110 Mbps, the latency is about 50 ms, and the QPS is about 10,000. Compared with the actual data, the results conform to the system behavior, which illustrates that our approach can predict resource usage and system performance under the underutilized scenario.

5.4 Overutilized Scenario

Considering the overutilized bandwidth as another example when the peak loads are 2000 and 3000 vu per second, which is also shown in Fig. 4. The increase and decrease patterns of loads are the same as the one with 1500 vu per second, which reaches the peak at time 660s and starts to drop at time 1200s. Based on the results, we can notice that when the outbound traffic reaches the peak and the saturation happens, the loads which arrive later cannot be processed by the service. Although CPU is not overutilized, the CPU utilization still grows up to be about 90% and the QPS also reaches 11,600 calls/s, which means bandwidth is the bottleneck rather than CPU in this case.

Given that resource competition will occur when resource usage approaches the bottleneck value, resulting in performance degradation, the average response time tends to rise in advance. Meanwhile, it is observed that when the bottleneck point is reached, the QPS remains at its peak, but the load is still rising. The

Table 4: Regression Models

Case	**Method**	**type**	**Regression fomula**
1	Single parameter regression	cpu	$\hat{U_c}(t)$=0.0324*$L(t)$+0.5691
2	Single parameter regression	bandwidth	$\hat{U_b}(t)$=0.1392*$L(t)$-1.0351
3	Double parameters regression	cpu	$\hat{U_c}(t)$=0.01129*$L(t)$+0.0626*t
4	Double parameters regression	bandwidth	$\hat{U_b}(t)$=0.0512*$L(t)$+0.2326*t
5	Triple parameters regression	cpu	$\hat{U_c}(t)$=0.00009*$L(t)$+0.2185*$U_b(t)$+0.0118*t
6	Triple parameters regression	bandwidth	$\hat{U_b}(t)$=-0.00005*$L(t)$+4.5441*$U_c(t)$-0.052*t

Table 5: Performance comparison with baselines

Approach	**RMSE (underutilized)**	**RMSE (overutilized)**	**RMSE**	**MAPE**	**R2**
Single parameter regression	55.832	54.561	55.200	35.6%	-0.243
Double parameters regression	79.782	14.638	57.356	30.3%	-0.342
Triple parameters regression	8.822	10.132	9.499	4.0%	0.963
PSO	8.171	8.603	8.390	6.7%	0.971
IRBS	**6.306**	**4.466**	**5.464**	**3.5%**	**0.988**

results obtained by the simulation system are nearly the same as the actual test results. According to the results, it can be observed that when the load is close to 1,800 vu, it is the bottleneck in this configuration. Therefore, we can conclude that our approach supports identifying the system bottleneck.

5.5 Evaluation Metrics and Results

To show the accuracy of our proposed approach, we used a linear regression algorithm [8] and PSO (particle swarm optimization) algorithm [22] to compare the performance of our proposed algorithm. We adopt these two algorithms as baselines because there is no algorithm focusing on the same scenario we are going to apply and these two algorithms have been validated to be efficient to model resource usage in cloud systems. In the linear regression algorithm, we concern about the different number of parameters (single parameter, double parameter, and triple parameters) for training. The corresponding models based on linear regression are shown in the Table 4. For instance, the predicted CPU utilization $\hat{U_c}(t)$ at time slot t with single parameter (loads number $L(t)$) regression can be represented as $0.0324 * L(t) + 0.5691$. PSO algorithm can be used as an automatic parameter training algorithm, in which specific parameters are trained and finally integrated into the system for evaluations. We evaluate the following metrics including Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and R^2, which have been widely used to evaluate the prediction accuracy:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2} \tag{11}$$

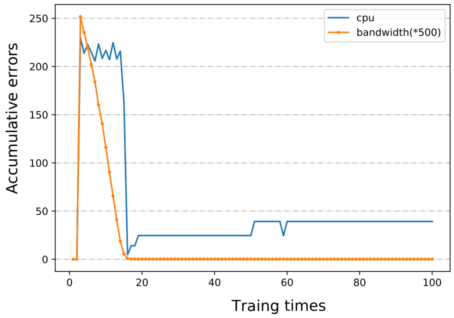


Fig. 5: Convergence process of cumulative errors

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{\hat{y_i} - y_i}{y_i} \right| \tag{12}$$

$$R^2 = 1 - \frac{\sum (y_i - \hat{y_i})^2}{\sum (y_i - \bar{y})^2} \tag{13}$$

where n is the size of the dataset, y_i is the actual value, $\bar{y}$ is the mean value of actual data, and $\hat{y_i}$ is the predicted value based on our approach. The performance evaluation results are shown in the Table 5.

It can be noted that our approach has the highest value of RMSE for the underutilized scenario, the overutilized scenario, and the combined scenario. Our approach can also obtain the smallest MAPE value and the best R^2 value, which all illustrate that our approach can achieve the highest accuracy than other algorithms.

5.6 Convergence Analysis

To show the convergence of our proposed approach, we also demonstrate the convergence process of cumulative errors as shown in Fig. 5. The CPU capacity and bandwidth capacity for processing loads are illustrated respectively. We use the algorithm introduced in Section 4.2 to realize the parameter updates. Based on the results, we can notice that the initial value and the final convergence value are significantly different, the cumulative errors have been reduced to be less than 100 within 100 training times. The experiments illustrate that our approach can achieve convergence efficiently.

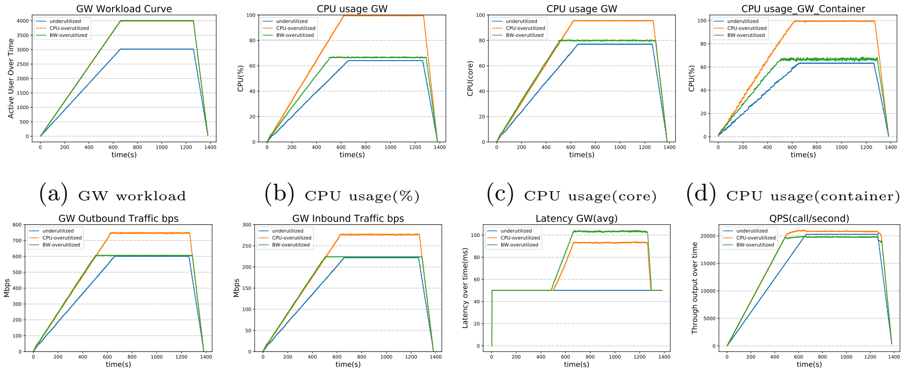
5.7 Bottleneck Prediction

Apart from identifying the bottleneck in specific configurations, IRBS can also predict the satisfaction of various metrics based on the parameters of the current configurations. Taking QPS as an example, Fig. 6 shows various situations when

QPS reaches 20,000 calls/s, and the configurations of each situation are the same as the configurations in Table 3.

The first case is that the CPU and bandwidth are not overutilized. In this case, we can notice that when the number of loads reaches 3020 vu, the QPS is close to 20,000 calls/s. At this moment, the peak response time of GW service is 50 ms, and other services are also at the system condition that is not overutilized, the total response time of the other parts is about 100 ms, and the total response time is about 150 ms. Thus, we can conclude that the prediction results are accurate.

The second and third cases are about the condition when the CPU or bandwidth reaches the overutilization and becomes the bottleneck. In these cases, the CPU or bandwidth utilization is about 100% as overutilized. Under this situation, the peak response time of the GW service is about 100 ms. And the total response time is about 200 ms, which shows that the results are also accurate. Then the users can use these results to optimize their infrastructure configuration, e.g. adding more CPU or bandwidth resources to avoid the bottleneck.



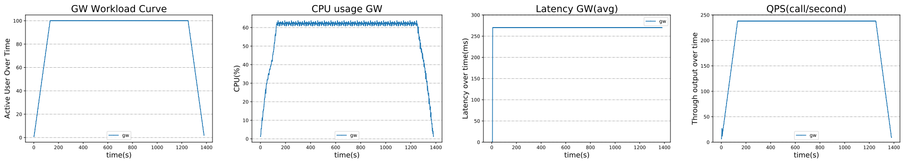
(a) GW workload (b) CPU usage(%) (c) CPU usage(core) (d) CPU usage(container)

(e) Outbound traffic (f) Inbound traffic (g) GW latency (h) QPS

Fig. 6: The metrics of different loads when QPS is 20,000 calls/s.

5.8 Scalability

Finally, we aim to show that this approach is not only suitable for the discussed scenario but can also be extended to other scenarios. Let us consider the scenario that there are only three services, namely load generation, GW, and Mockservice. Due to changes in the hardware resources to support services, we can re-adjust the parameters, select different parts by adjusting the VM and container capacity, and control the execution order by defining them in the property files. Some sample results are shown in Fig. 7. The only efforts needed to be made are changing the property files, including the load number, type, and the order of interacted services.



(a) GW Workload (b) CPU usage (c) GW latency (d) QPS

Fig. 7: Extended scenario

6 Conclusions and future work

In this paper, we present a resource bottleneck identification approach, called IRBS, which can predict resource and service bottlenecks in multiple services interactions scenarios. We also proposed a policy gradient algorithm based on reinforcement learning to support the prediction of resource usage under different conditions. Based on the dataset provided, which is deployed on AliCloud, our simulated results based on the extension of CloudSim show that our approach can effectively identify the potential bottlenecks and achieve high accuracy for adopted metrics. The dominant advantage of the proposed approach is easy to operate and it provides a feasible solution for the customers to plan infrastructure resources. It can also ensure high prediction accuracy while maintaining reusability and scalability.

As future work, we would like to investigate other datasets or a broad range of conditions to evaluate our proposed approach. We would also like to apply our approach to other cloud platforms, e.g. Amazon and Azure.

Acknowledgments

This research is partially supported by Key-Area Research and Development Program of Guangdong Province (NO. 2020B010164003), the National Natural Science Foundation of China, with Grant ID 61672136 and 61828202, SIAT Innovation Program for Excellent Young Researchers. We thank teams in Siemens Industry Software Co., Ltd, China, for their discussion and comments on this work.

References

1. Alla, H.B., Alla, S.B., Touhafi, A., Ezzati, A.: A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment. Cluster Computing 21(4), 1797–1820 (2018)
2. Belalem, G., Tayeb, F.Z., Zaoui, W.: Approaches to improve the resources management in the simulator cloudsim. In: International Conference on Information Computing and Applications. pp. 189–196. Springer (2010)

3. Ben Alla, H., Ben Alla, S., Ezzati, A., Mouhsen, A.: A novel architecture with dynamic queues based on fuzzy logic and particle swarm optimization algorithm for task scheduling in cloud computing. In: El-Azouzi, R., Menasche, D.S., Sabir, E., De Pellegrini, F., Benjillali, M. (eds.) Advances in Ubiquitous Networking 2. pp. 205–217. Springer Singapore, Singapore (2017)
4. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems 25(6), 599–616 (2009)
5. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and experience 41(1), 23–50 (2011)
6. Chen, W., Ye, K., Wang, Y., Xu, G., Xu, C.: How does the workload look like in production cloud? analysis and clustering of workloads on alibaba cluster trace. In: 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS). pp. 102–109 (2018)
7. Cheng, M., Li, J., Nazarian, S.: Drl-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers. In: in Proceedings of the 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC). pp. 129–134. IEEE (2018)
8. Fan, Jianqing: Local linear regression smoothers and their minimax efficiencies. Annals of Statistics 21(1), 196–216 (1993)
9. Funika, W., Koperek, P.: Evaluating the use of policy gradient optimization approach for automatic cloud resource provisioning. In: Wyrzykowski, R., Deelman, E., Dongarra, J., Karczewski, K. (eds.) Parallel Processing and Applied Mathematics. pp. 467–478. Springer International Publishing, Cham (2020)
10. Gao, H., Huang, W., Zou, Q., Yang, X.: A dynamic planning framework for qos-based mobile service composition under cloud-edge hybrid environments. In: Wang, X., Gao, H., Iqbal, M., Min, G. (eds.) Collaborative Computing: Networking, Applications and Worksharing. pp. 58–70. Springer International Publishing, Cham (2019)
11. Ibidunmoye, O., Hernández-Rodriguez, F., Elmroth, E.: Performance anomaly detection and bottleneck identification. ACM Computing Surveys (CSUR) 48(1), 1–35 (2015)
12. Jung, J., Kim, H.: Mr-cloudsim: Designing and implementing mapreduce computing model on cloudsim. In: Proceedings of the 2012 International Conference on ICT Convergence (ICTC). pp. 504–509. IEEE (2012)
13. Latiff, M.S.A., Madni, S.H.H., Abdullahi, M., et al.: Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm. Neural Computing and Applications 29(1), 279–293 (2018)
14. Mousavi, S.S., Schukat, M., Howley, E.: Traffic light control using deep policy-gradient and value-function-based reinforcement learning. IET Intelligent Transport Systems 11(7), 417–423 (Sep 2017)
15. Nayak, S.C., Tripathy, C.: Deadline sensitive lease scheduling in cloud computing environment using ahp. Journal of King Saud University-Computer and Information Sciences 30(2), 152–163 (2018)
16. Petrik, D., Herzwurm, G.: iiot ecosystem development through boundary resources: a siemens mindsphere case study. In: Proceedings of the 2nd ACM SIGSOFT International Workshop on Software-Intensive Business: Start-ups, Platforms, and Ecosystems. pp. 1–6 (2019)

17. Priya, V., Kumar, C.S., Kannan, R.: Resource scheduling algorithm with load balancing for cloud service provisioning. Applied Soft Computing 76, 416–424 (2019)
18. Sekaran, K., Khan, M.S., Patan, R., Gandomi, A.H., Krishna, P.V., Kallam, S.: Improving the response time of m-learning and cloud computing environments using a dominant firefly approach. IEEE Access 7, 30203–30212 (2019)
19. Shi, Y., Jiang, X., Ye, K.: An energy-efficient scheme for cloud resource provisioning based on cloudsim. In: Proceedings of the 2011 IEEE International Conference on Cluster Computing. pp. 595–599. IEEE (2011)
20. Wang, Z., Wen, Y., Zhang, Y., Chen, J., Cao, B.: A resource usage prediction-based energy-aware scheduling algorithm for instance-intensive cloud workflows. In: Gao, H., Wang, X., Yin, Y., Iqbal, M. (eds.) Collaborative Computing: Networking, Applications and Worksharing. pp. 626–642. Springer International Publishing, Cham (2019)
21. Wickremasinghe, B., Calheiros, R.N., Buyya, R.: Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In: Proceedings of the 2010 24th IEEE international conference on advanced information networking and applications. pp. 446–452. IEEE (2010)
22. Wu, D., Jiang, N., Du, W., Tang, K., Cao, X.: Particle swarm optimization with moving particles on scale-free networks. IEEE Transactions on Network Science and Engineering 7(1), 497–506 (2020)
23. Xu, M., Buyya, R.: Brownout approach for adaptive management of resources and applications in cloud computing systems: a taxonomy and future directions. ACM Computing Surveys (CSUR) 52(1), 1–27 (2019)
24. Xu, M., Tian, W., Buyya, R.: A survey on load balancing algorithms for virtual machines placement in cloud computing. Concurrency and Computation: Practice and Experience 29(12), e4123 (2017)
25. Xu, M., Toosi, A.N., Bahrani, B., Razzaghi, R., Singh, M.: Optimized renewable energy use in green cloud data centers. In: Service-Oriented Computing, pp. 314–330. Springer International Publishing (2019)
26. Yadwadkar, N.J., Hariharan, B., Gonzalez, J.E., Smith, B., Katz, R.H.: Selecting the best vm across multiple public clouds: A data-driven performance modeling approach. In: Proceedings of the 2017 Symposium on Cloud Computing. pp. 452–465 (2017)
27. Yazdanbakhsh, A., Thwaites, B., Esmaeilzadeh, H., Pekhimenko, G., Mutlu, O., Mowry, T.C.: Mitigating the memory bottleneck with approximate load value prediction. IEEE Design & Test 33(1), 32–42 (2016)