Learning-Based Resource Allocation in Cloud Data Center Using Advantage Actor-Critic

Zheyi Chen, Jia Hu, Geyong Min

Department of Computer Science, College of Engineering, Mathematics and Physical Sciences

University of Exeter

Exeter, United Kingdom

{zc300, j.hu, g.min}@exeter.ac.uk

Abstract—Due to the ever-changing system states and various user demands, resource allocation in cloud data center is faced with great challenges in dynamics and complexity. Although there are solutions that focus on this problem, they cannot effectively respond to the dynamic changes of system states and user demands since they depend on the prior knowledge of the system. Therefore, it is still an open challenge to realize automatic and adaptive resource allocation in order to satisfy diverse system requirements in cloud data center. To cope with this challenge, we propose an advantage actor-critic based reinforcement learning (RL) framework for resource allocation in cloud data center. First, the actor parameterizes the policy (allocating resources) and chooses continuous actions (scheduling jobs) based on the scores (evaluating actions) from the critic. Next, the policy is updated by gradient ascent and the variance of policy gradient can be significantly reduced with the advantage function. Simulations using Google cluster-usage traces show the effectiveness of the proposed method in cloud resource allocation. Moreover, the proposed method outperforms classic resource allocation algorithms in terms of job latency and achieves faster convergence speed than the traditional policy gradient method.

Index Terms—Resource allocation, reinforcement learning, cloud computing, actor-critic

I. INTRODUCTION

Cloud computing has rapidly developed as one of the most prevailing computing paradigms in recent years. In cloud computing, resource allocation can be regarded as a process for allocating computing, storage and networking resources in order to meet the collective performance objective from both users and cloud service providers (CSPs). Therefore, it is crucial to develop effective and efficient resource allocation solutions for cloud data center, which are highly challenging due to the dynamic nature of system states and requirements in cloud computing, described as follows.

- The complexity of cloud data center: There are various types of infrastructures in cloud data center (e.g., physical servers, virtual machines (VMs) and containers). They own different central processing units (CPUs), memories and disks, which bring huge challenges to cloud resource allocation.
- The diversity of demands from users: Jobs coming from heterogeneous users have different characteristics, such as requests for various types of resources (e.g., VM, CPU and memory) and requests for different job du-

rations (e.g., minutes, hours and days). Such diversity of demands escalates the difficulty in cloud resource allocation.

Moreover, due to the time-variance of system states and workloads, resource allocation in cloud data center must be on-line and adaptive. However, most of the traditional solutions depend on the prior knowledge of the system, which results in low efficiency and cannot fit in the real-world environment with complex and dynamic system states and user demands. As it is impossible to build an accurate system model under the ever-changing cloud environment, reinforcement learning (RL) [1] has emerged as an automatic decision-making method to solve the complicated yet crucial problem of resource allocation in cloud data center.

Although the existing RL-based methods can partially solve the resource allocation problem, they discretize continuous values (e.g., job latency and energy consumption) by using the value-based RL algorithms (e.g., Q-learning [1] and DQN [2]). Therefore, not only the integrity of continuous state and action spaces is destroyed but also noises are introduced, which makes it impossible to learn the exact optimal policy. By contrast, the policy-based RL (e.g., policy gradient [1]) can more effectively address the above problem of continuous spaces. However, the traditional policy methods are updated by round and generate high variance while estimating the gradient, which seriously reduces the training efficiency. Therefore, the actor-critic based RL algorithm [1] was proposed to make up for the defects of both value-based and policy-based RL methods. However, the traditional actor-critic is based on action-value critic that uses linear value function to approximate action-value function, which results in high variance and inaccurate policy gradient. To tackle this issue, an advantage actor-critic was designed in [3]. The actor selects actions based on the stochastic probability and the critic evaluates actions, and then the actor changes the probability for selecting actions based on the scores from the critic, which means that the critic guides the actor to the 'right direction'. Meanwhile, the variance of policy gradient can be significantly reduced with the help of advantage function. As the optimization goal (job latency) and the state (resource usage) have continuous space as well as for fast decision-making in cloud computing, we adopt an advantage actor-critic based RL method to explore

the optimal policy with low job latency and high learning efficiency for dynamic resource allocation in cloud data center. The main contributions are summarized as follows.

- An on-line adaptive resource allocation strategy based on RL is proposed to minimize the job latency in cloud data center with dynamic states and heterogeneous user demands.
- An advantage actor-critic method is proposed to achieve the optimal policy of resource allocation by interacting with the cloud environment, which is able to handle the problem of continuous state and action spaces with high learning efficiency.
- Simulations using Google cluster datasets validate the effectiveness of the proposed method on achieving lower job latency and faster convergence compared to the classic resource allocation algorithms and the traditional policy gradient method.

The rest of this paper is organized as follows. Section II introduces the related work. Section III describes the system model and the resource allocation problem. In Section IV, the proposed actor-critic based RL method is discussed in detail. In section V, we evaluate the proposed method with real-world datasets. The last section concludes our work.

II. RELATED WORK

Resource allocation problems are omnipresent in cloud computing and there are many contributions for promoting resource utilization by reasonable resource allocation and cost control. The job scheduling was formulated as a non-linear mixed integer programming problem and a relaxation with an equivalent linear programming problem was presented in [4]. Based on an imperfect information stackelberg game (CSAM-IISG), Wei *et al.* solved the problem of cloud resource allocation through using a hidden Markov model [5]. A α-approximation based method was proposed in [6] to initiate a random combinatory auction for dynamic resource provisioning in cloud computing. The authors offered a skewness-avoidance method for heterogeneous resource allocation (SAMR) in [7] to meet the diversified requirements on different types of resources. In general, most of the traditional solutions for resource allocation in cloud computing focus on game theory or heuristic algorithms. Meanwhile, these methods mainly depend on the manual configuration and the prior knowledge or experience, which results in low efficiency and cannot adapt to the real-time requirements of cloud computing due to the time-varying characteristic and the uncertainty.

Reinforcement learning (RL) [1] emphasizes how to take actions based on the environment to maximize the expected benefits. In the past few years, RL has been applied to solve the problem of cloud resource allocation. Kontarinis *et al.* adopted a Q-learning based approach to implement the adaptive resource control from the perspective of users in cloud environment [8]. A hierarchical framework was designed in [9] for adaptive resource allocation by using DQN algorithm, which reduced the power consumption in cloud system. A deep Q-learning based system for resource provisioning and

task scheduling was designed in [10] to minimize the energy cost of cloud service providers. Different from the above work, Mao *et al.* leveraged policy gradient to handle the resource management problem [11]. However, current related work mostly focuses on the valued-based RL (e.g., Q-learning [1] and deep Q-networks [2]), which not only discretizes the continuous values (e.g., job latency) in cloud environment but also introduces noises. Therefore, these methods cannot learn the accurate optimal policy under the problem of continuous state and action spaces. Different from the existing work, we first propose an actor-critic based RL method to achieve the optimal strategy for cloud resource allocation, which takes the advantages of both valued-based and policy-based RL methods for higher adaptiveness and learning efficiency.

III. SYSTEM MODEL AND PROBLEM FORMULATION

We design a unified model of resource allocation to minimize job latency while considering both user demands and dynamic status of cloud data center. Therefore, we regard a cloud data center with a set of servers $V = \{v_1, v_2, ..., v_n\}$ and each server provides multiple types of resource $R = \{r_1, r_2, ..., r_n\}$ (e.g., CPU and memory), as shown in Fig. 1. The scheduler is responsible for assigning jobs from the job sequence to servers according to resource requests of different jobs and current resource usage of cloud data center. More specifically, each job consists of a specific job duration (i.e., the duration of job under ideal conditions) and the request for different types of resources. Meanwhile, each server records the usage of different resources at each timestep. The major notations involved in the proposed model are listed in Table I.
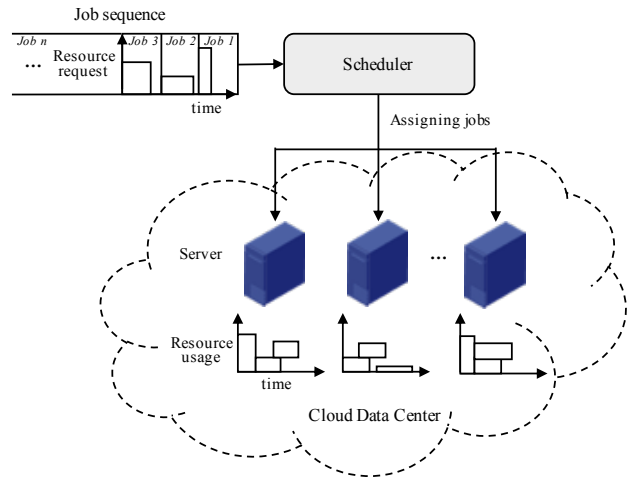

Fig. 1. The model of resource allocation in cloud data center.

To achieve as low job latency as possible by legitimately assigning jobs based on the current resource usage, we consider a job sequence as a set of jobs $J = \{job_1, job_2, ..., job_n\}$,and measure the actual completion time of each job j during resource allocation, $T_{finish}^j - T_{enter}^j$, which includes the job waiting time. As numerical discrepancy among different characteristics may cause excessive time consuming and tortuous process of gradient descent, the normalization is used to improve the training speed and convergence. Therefore, we

define L as the normalized mean job latency, which normalizes the job latency and then takes their average.

$$L = \frac{\sum_{j \in J} \left(\left(T^j_{finish} - T^j_{enter} \right) / D_j \right)}{Number\ of\ jobs\ in\ J}, \qquad (1)$$

where D_j is the duration of job j.

To reduce the normalized job latency L, we adopt RL-based methods to address resource allocation and job scheduling in cloud data center. Firstly, we regard the scheduler as an RL agent and cloud data center as the environment. At each timestep, the RL agent chooses an action by interacting with the environment. Then, we define the state space, action space and reward function for the proposed RL model as follows.

State space: The state of cloud data center s_t consists of two parts at timestep t, which are the resource usage of cloud data center s^V_t and resource requests of job s^j_t. More specifically, we define s^V_t as the usage for different types of resources $U^R_t = [u^{r_1}_t, u^{r_2}_t, ..., u^{r_n}_t]$, and define s^j_t as the union of occupancy requests of jobs for different types of resources $O^{j|R|}_t = [o^{j|r_1|}_t, o^{j|r_2|}_t, ..., o^{j|r_n|}_t]$ and job duration D_j. Thus, the state of cloud data center s_t can be denoted as follows:

$$\begin{aligned} s_t &= [s^V_t, s^j_t] = [U^R_t, O^{j|R|}_t, D_j] \\ &= [u^{r_1}_t, u^{r_2}_t, ..., u^{r_n}_t, o^{j|r_1|}_t, o^{j|r_2|}_t, ..., o^{j|r_n|}_t, D_j]. \end{aligned} \qquad (2)$$

Action space: The action a_t adopted by scheduler at timestep t is to select jobs from job sequence according to the current resource usage of cloud data center and resource requests from different jobs. And the scheduler then assigns jobs to servers $V = \{v_1, v_2, ..., v_n\}$ in cloud data center. Thus, we define action a_t as follows:

$$A = \{a^{j|V'|}_t | V' \in \{\emptyset, v_1, v_2, ..., v_n\}\}, \qquad (3)$$

where $\emptyset$ means that the scheduler does not assign jobs at timestep t and jobs still need to wait in job sequence.

Reward function: The reward function r_t is used to guide RL agent to learn the optimized policy with higher rewards for job scheduling at each timestep t, whose objective is to minimize the normalized job latency L and meet the resource requests from jobs simultaneously. We consider reward function consisting of three aspects of penalty, which are $-T^j_{wait}/D_j$, $-T^j_{work}/D_j$ and $-T^j_{dismiss}/D_j$, as described in Table I. Therefore, the reward function r_t is defined as follows:

$$r_t = -\sum_{j \in J} \left(\frac{w_1 \cdot T^{j|t|}_{wait}}{D_j} + \frac{w_2 \cdot T^{j|t|}_{work}}{D_j} + \frac{w_3 \cdot T^{j|t|}_{dismiss}}{D_j} \right), \qquad (4)$$

where w_1, w_2 and w_3 are used to weight the penalty.

During the learning process of optimizing policy for resource allocation, the RL agent (scheduler) chooses an action a_t (resource allocation and job scheduling) under current state s_t (resource usage and requests) of environment (cloud data center), and then it receives reward r_t (penalty of job latency) and goes to the next state s_{t+1}. This process can be formulated as a Markov Decision Process (MDP):

$$s_0, a_0 \to r_0, s_1, a_1 \to ... \to r_t, s_{t+1}. \qquad (5)$$

Due to the uncertainty of states in cloud data center, we formulate the problem of resource allocation and job scheduling with model-free RL. Meanwhile, as the above problem is a discrete-time based MDP with continuous spaces, we propose an actor-critic based RL method for better adaptiveness and achieving the accurate optimal policy more efficiently.

TABLE I
MAJOR NOTATIONS OF MODEL

Notation	Description
V	Set of servers in cloud data center
R	Types of resources in cloud data center
J	Set of jobs in the job sequence
T^j_{finish}	The timestep when job j is completed
T^j_{enter}	The timestep when job j enters job sequence
D_j	The duration of job j under ideal conditions
L	Normalized job latency
s_t	State of cloud data center at timestep t
U^R_t	Usage of resource R at timestep t
$O^{j|R|}_t$	Occupancy request of job j for resource R at timestep t
a_t	Action adopted by scheduler at timestep t
r_t	Reward received at timestep t
$T^{j|t|}_{wait}$	Waiting time of job j in job sequence at timestep t
$T^{j|t|}_{work}$	Execution time of job j at timestep t
$T^{j|t|}_{dismiss}$	Waiting time until job j is dismissed at timestep t

IV. ACTOR-CRITIC BASED RL ALGORITHM WITH ADVANTAGE FUNCTION

Actor-critic is a hybrid of RL algorithms, which incorporates the value-based (e.g., Q-learning) and the policy-based (e.g., policy gradient) RL algorithms. On the one hand, the value-based methods use Temporal-Difference (TD) learning to evaluate TD error generated during the learning process. Usually, they first use function approximators to determine the value function and then use ϵ-greedy method to balance exploration and exploitation, which allows the RL agent both exploring new actions and utilizing existing experiences to choose the optimal action. On the other hand, the policy-based methods parameterize the policy and directly output actions during the learning process without storing the value function, so it can choose actions under the continuous state and action spaces. As RL methods, the optimization goal of actor-critic is getting as much reward as possible. Thus, an objective function is to measure the learning quality. For continuous state and action spaces, we use the following formula to accumulate the instant reward obtained from actions adopted at each state based on a probability distribution.

$$J(\theta) = \sum_{s \in S} d^{\pi_\theta}(s) \sum_{a \in A} \pi_\theta(s, a) R_{s,a}, \qquad (6)$$

where $d^{\pi_\theta}(s)$ is a stationary distribution of Markov chain (as shown in Eq. 5) under the current policy π_θ.

When the objective function is determined, the next step is to optimize the parameters of policy so that the value of objective function can be maximized. Policy gradient algorithm can

make a move of $J(\theta)$ along the direction of gradient ascend to the local maximum and achieve the optimal parameter θ when maximum value is obtained:

$$\Delta(\theta) = \alpha\nabla_\theta J(\theta) = \alpha(\partial J(\theta)/\partial\theta_1, ..., \partial J(\theta)/\partial\theta_n)^T, \quad (7)$$

where α is learning rate and $\nabla_\theta J(\theta)$ is policy gradient.

Here, we combine the objective function $J(\theta)$ with policy gradient. We firstly consider one-step MDP, which starts at state $s \sim d(s)$ and finishes after one timestep with instant reward $r = R_{s,a}$. Then, we can get the objective function and the corresponding gradient as follows:

$$J(\theta) = E_{\pi_\theta}[r] = \sum_{s\in S} d(s) \sum_{a\in A} \pi_\theta(s,a)R_{s,a}, \qquad (8)$$

$$\nabla_\theta J(\theta) = \sum_{s\in S} d(s) \sum_{a\in A} \pi_\theta(s,a)\nabla_\theta log\pi_\theta(s,a)R_{s,a}$$
$$= E_{\pi_\theta}[\nabla_\theta log\pi_\theta(s,a)r]. \qquad (9)$$

It can be noticed that the gradient of objective function is equal to the product expectation of logarithmic gradient of policy function π_θ and instant reward r. When it comes to multi-step MDPs, we replace the instant reward r with long-term value $Q^\pi(s,a)$ and get the following theorem.

Theorem 1. *Policy Gradient Theorem* [1]: For any differentiable policy $\pi_\theta(s,a)$ and any policy objective functions, the policy gradient is

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta log\pi_\theta(s,a)Q^{\pi_\theta}(s,a)]. \qquad (10)$$

Based on Eq. 10, Monte Carlo (MC) and Temporal-Difference (TD) learning can be applied into practical problems. We firstly consider MC learning, which uses return v_t as an unbiased sample of $Q_\theta^\pi(s_t,a_t)$ to update parameter θ. Although MC learning is unbiased, the noise is relatively large, which means that variance of MC is high. If the value of state can be estimated relatively accurately and be used to guide the update of policy, better learning results can be achieved, which is also the basic idea of actor-critic algorithm. Fig. 2 illustrates the framework of actor-critic. Similar to the policy-based RL methods, actor-critic can handle the problem of continuous state and action spaces, and variance can also be effectively reduced by adding critic into this framework.
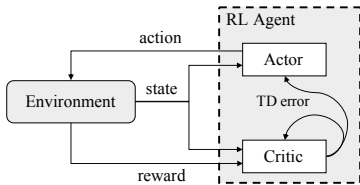

Fig. 2. The framework of actor-critic algorithm.

More specifically, critic updates parameters w by estimating action-value function $Q_w(s,a) \approx Q^{\pi_\theta}(s,a)$, and actor guides the update of policy parameter θ based on the value evaluated by critic. Thus, it follows an approximate policy gradient.

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta log\pi_\theta(s,a)Q_w(s,a)]. \qquad (11)$$

The original actor-critic is based on action-value critic that uses linear value function to approximate action-value function

$Q_w(s,a) = \phi(s,a)^T w$, which results in high variance. To avoid this problem and achieve the exact policy gradient, the value function is selected based on the following theorem.

Theorem 2. *Compatible Function Approximation* [1]: The policy gradient in Eq. 11 would be exact if the following two conditions are satisfied.

1) Value function approximator is compatible to policy.

$$\nabla_w Q_w(s,a) = \nabla_\theta log\pi_\theta(s,a). \qquad (12)$$

2) Optimal parameter w minimizes the mean-squared error.

$$\varepsilon = E_{\pi_\theta}[(Q^{\pi_\theta}(s,a) - Q_w(s,a))^2]. \qquad (13)$$

Algorithm 1: Advantage Actor-Critic RL Algorithm

---

1 Initialize actor network $V^{\pi_\theta}(s)$ and critic network $Q^{\pi_\theta}(s,a)$ with weights and biases
2 Initialize actors and critics learning rate γ_a and γ_c, and TD error discount factor β
3 **for** *each training epoch* $n = 1, 2, ..., N$ **do**
4   Receive initial state s_1, where $s_1 = env.observe()$
5   **for** *each episode* $t = 1, 2, ..., T$ **do**
6   Select action a_t according to s_t, where $a_t = actor.choose\_action(s_t)$
7   Execute action a_t, receive reward r_t and next state s_{t+1}, where $r_t, s_{t+1} = env.step(a_t)$
8   Calculate TD error in critic, where $\delta^{\pi_\theta} = r + \beta V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)$
9   Calculate policy gradient in actor using advantage function, where $\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta log\pi_\theta(s_t,a)\delta^{\pi_\theta}]$
10   update state $s_t = s_{t+1}$
11   **end**
12 **end**

Based on Theorem 2, we improve the original actor-critic algorithm by using a baseline to reduce variance. The basic idea is to subtract a baseline function $B(s)$ from the policy gradient. $B(s)$ is only related to state but has nothing to do with action, so it does not change the gradient. The characteristic of $B(s)$ is reducing variance while not changing the expectation of action values. When $B(s)$ matches this characteristic, the following inference is established:

$$E_{\pi_\theta}[\nabla_\theta log\pi_\theta(s,a)B(s)] = \sum_{s\in S} d^{\pi_\theta}(s) \sum_{a\in A} \nabla_\theta\pi_\theta(s,a)B(s)$$
$$= \sum_{s\in S} d^{\pi_\theta}B(s)\nabla_\theta \sum_{a\in A} \pi_\theta(s,a)$$
$$= 0. \qquad (14)$$

In principle, any functions that are not related to action can be used as $B(s)$. But a good $B(s)$ is a state-value function based on the current state, which is $B(s) = V^{\pi_\theta}(s)$. Therefore, the policy gradient can be rewritten as follows:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta log\pi_\theta(s,a)A^{\pi_\theta}(s,a)], \qquad (15)$$

where $A^{\pi_\theta} = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$ is called the advantage function and $V^{\pi_\theta}(s)$ is the state-value function.

The variance of policy gradient thus can be significantly reduced by the advantage function. Besides, both $V^{\pi_\theta}(s)$ and $Q^{\pi_\theta}(s,a)$ can be updated by TD learning, and TD error can be calculated based on the state-value function $V^{\pi_\theta}(s)$.

$$\delta^{\pi_\theta} = r + \beta V^{\pi_\theta}(s') - V^{\pi_\theta}(s). \qquad (16)$$

Finally, the exact policy gradient is achieved as follows:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta log\pi_\theta(s,a)\delta^{\pi_\theta}]. \qquad (17)$$

The key steps of advantage actor-critic RL algorithm are as shown in Algorithm 1. For each episode, an action is generated based on the current state and instant reward, the critic then estimates $V^{\pi_\theta}(s)$ and $Q^{\pi_\theta}(s,a)$ with Theorem 2 and calculates TD error δ^{π_θ}. Based on the evaluation from the critic, the actor calculates policy gradient by using the advantage function.

V. EXPERIMENTS

In this section, we first describe the simulation settings and datasets. Then, we evaluate the performance of the proposed model in terms of job latency. Finally, we compare the proposed method with other classic methods.

A. Settings and Datasets

We implement the proposed model for resource allocation in cloud data center based on TensorFlow 1.4.0 and use the real-world datasets from Google cluster-usage traces [12], which contain the resource usage data of Google cloud data center in May 2011. More specifically, we firstly randomly extract 1,000 servers from the Google datasets over 29 days, where each server consists of around 100,000 job traces. And each job consist of different job arrival time, arrival rates of new job, job durations and resource occupancy. Based on the job traces we selected, we initialize some settings of parameter in the proposed model of cloud data center, which include the number of servers $V = 1000$ and types of resources $R = 3$. Next, we calculate the average values of the above metrics of jobs and regard them as the default maximum thresholds of arrival rate of new job $C_t = 50\%$, job duration $D_j = 10$ and resource occupancy $O_t = 30\%$. Based on the the above settings, the experiments are conducted to evaluate the performance of the proposed method through the simulation of real-world cloud data center. To validate the advantages of our proposed method, we evaluate the performance of the policy gradient based RL method [11] and the other classic algorithms, including First Come First Served (FCFS), Shortest Job First (SJF) and Round Robin (RR), to conduct comparative experiments.

B. Experimental Results

The aim of our proposed method is to minimize the job latency while considering both changeable system states and different resource requests from different jobs, which is specially formulated as the normalized job latency L. Therefore, we study the performance of the proposed model in terms of job latency for dynamic resource allocation and job scheduling by changing the threshold values of parameter settings, which includes the maximum thresholds of job duration D_j, resource

occupancy of job at each timestep O_t, types of resource R and arrival rate of job at each timestep C_t.

Fig. 3 shows the distribution of the normalized job latency with different job durations D_j and resource occupancy of job O_t after algorithm converges. We set the maximum job duration from $D_j = 5$ to $D_j = 15$ and the maximum resource occupancy of job from $O_t = 5\%$ to $O_t = 50\%$. Generally, the normalized job latency rises with the increase of D_j and O_t. More specifically, there is almost no latency happening when the proposed method deals with jobs with small resource demands, as the normalized job latency is almost equal to 1. When resource demands of jobs go larger, larger number of resource is consumed with longer job duration, which results in the growth of the normalized job latency. Even so, our proposed method can also achieve satisfied results of low job latency, especially for small and medium-sized jobs (i.e., $D_j = 5$ and $D_j = 10$). Besides, the normalized job latency is evaluated with different types of resources R and different arrival rates of job C_t based on the above setting with large resource demands from jobs.
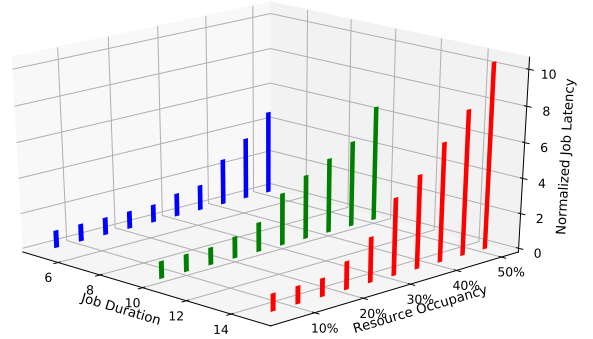

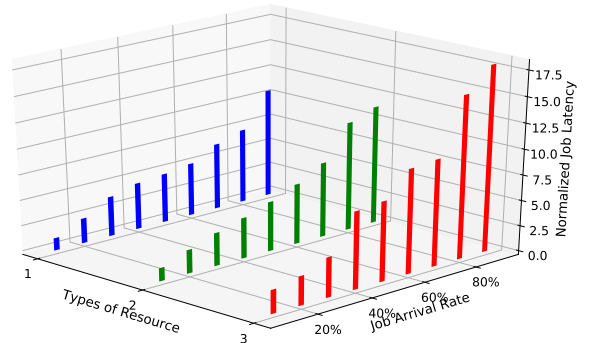Fig. 3. Latency under different job durations and resource occupancy.

Fig. 4. Latency under different types of resource and job arrival rates.

As shown in Fig. 4, we set the types of resource R as 1 to 3, and adjust job arrival rate from $C_t = 10\%$ to $C_t = 90\%$. With the increase of R and C_t, cloud data center becomes more complex with more types of resource and larger amount of workloads, which results in higher normalized job latency. Even in such complex condition, our proposed method can also succeed in handling the dynamic resource allocation problem adaptively and achieve excellent performance in terms of relatively low latency. As we can see from Fig. 4, even job arrival rate O_t rises to 50%, the proposed method can still

control the normalized job latency effectively with different settings of resource types.

As shown in Fig. 5, all heuristic methods (i.e., FCFS, SJF and RR) do not have learning process as RL-based methods, so they can only make decision for resource allocation simply depends on the current system state and cannot make any improvements for better adaptiveness to the dynamic environment in cloud data center. When it comes to the proposed actor-critic based method, although the normalized job latency is not much different from the existing methods under initial low-workload condition, it outperforms heuristic methods along with the increase of workloads. It can be seen that the proposed method can substantially reduce job latency compared to the existing methods by continuous learning and interacting with the environment. For example, when the newly-increased workloads rise to 45%, the proposed method can reduce around 13% job latency compared to RR(SJF), which performs best in the existing methods. Thus, the proposed method presents strong adaptiveness to the complex and dynamic environment.
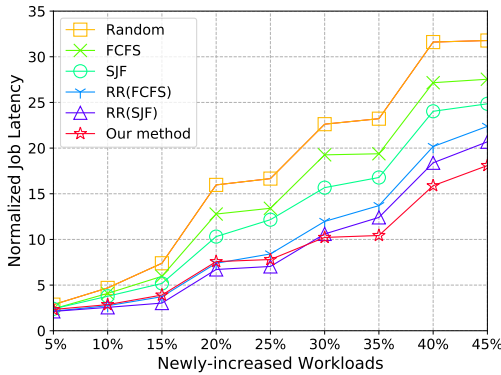

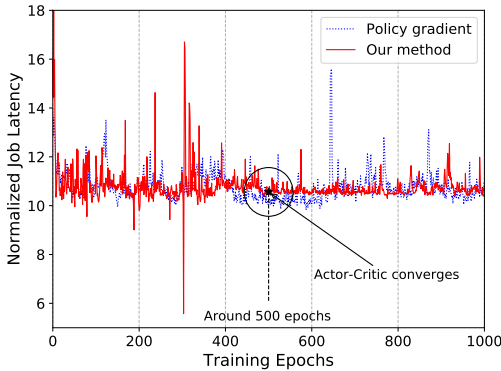Fig. 5. Performance comparision with heuristic methods.

Fig. 6. Performance comparision with the policy gradient method.

Fig. 6 shows the comparison of convergence speed between actor-critic based method and policy gradient method in terms of the normalized job latency. We can notice that the learning curve tends to converge after around 500 training epochs by using the proposed method, while policy gradient is divergent over the whole training epochs. The experimental result shows that the proposed method can achieve higher learning efficiency compared to the policy gradient, as it can effectively

avoid large variance and achieve faster convergence through using the advantage function while estimating the gradient.

VI. CONCLUSION

Dynamic resource allocation in cloud data center has received considerable research attention. However, some key challenges haven't been addressed, including the adaptability to the time-varying feature of system states and user demands. In this paper, we first formulate the resource allocation in cloud data center as a model-free RL problem with continuous state and action spaces. Next, we propose an advantage actor-critic based RL method to dynamically allocate resources and schedule jobs for minimizing the job latency while satisfying user demands. Simulations using workload datasets from Google data center demonstrate that the proposed method yields low job latency and high efficiency. Specifically, the job latency of our method is the lowest with the growth of workloads compared to FCFS, SJF and RR, which verifies the strong adaptability of our method. Our method also outperforms the policy gradient method in terms of learning efficiency. The proposed actor-critic based method is of great value for improving resource utilization and reducing job latency in cloud computing.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, and S. Petersen, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[3] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, 2016.

[4] L. Shi, Z. Zhang, and T. Robertazzi, "Energy-aware scheduling of embarrassingly parallel jobs and resource allocation in cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 6, pp. 1607–1620, 2017.

[5] W. Wei, X. Fan, H. Song, X. Fan, and J. Yang, "Imperfect information dynamic stackelberg game based resource allocation using hidden markov for cloud computing," *IEEE Transactions on Services Computing*, vol. 11, no. 1, pp. 78–89, 2018.

[6] L. Zhang, Z. Li, and C. Wu, "Dynamic resource provisioning in cloud computing: A randomized auction approach," in *INFOCOM, 2014 Proceedings IEEE*, pp. 433–441, IEEE, 2014.

[7] L. Wei, C. H. Foh, B. He, and J. Cai, "Towards efficient resource allocation for heterogeneous workloads in iaas clouds," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 264–275, 2018.

[8] A. Kontarinis, V. Kantere, and N. Koziris, "Cloud resource allocation from the user perspective: A bare-bones reinforcement learning approach," in *International Conference on Web Information Systems Engineering*, pp. 457–469, Springer, 2016.

[9] N. Liu, Z. Li, J. Xu, Z. Xu, S. Lin, and Q. Qiu, "A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, pp. 372–382, IEEE, 2017.

[10] M. Cheng, J. Li, and S. Nazarian, "Drl-cloud: deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," in *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*, pp. 129–134, IEEE Press, 2018.

[11] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pp. 50–56, ACM, 2016.

[12] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format + schema," *Google Inc., White Paper*, pp. 1–14, 2011.