SLAs-aware Online Task Scheduling based on Deep Reinforcement Learning Method in Cloud Environment

Longyu Ran*†, Xiaoyu Shi*, Mingsheng Shang*
*Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China
†University of Chinese Academy of Sciences, Beijing 100049, China
Corresponding author: Xiaoyu Shi, email: xiaoyushi@cigit.ac.cn

***Abstract*—As increasingly traditional applications migrate to the cloud, load balancing without sacrificing the performances of cloud-based applications is a highly challenging problem in cloud computing. Existing solutions focusing on online task scheduling have the inefficient issue, due to the highly dynamic nature of cloud workloads and the virtual machines (VMs) with heterogeneous hardware configurations. To address this issue, dependent on the strong perception and decision-making ability of deep reinforcement learning (DRL) in automatic control problems, we propose a deep reinforcement learning based task scheduling method, which can dynamic assign submitted tasks to different VMs. In details, we formulate the task scheduling as a dynamical optimal problem with constraints, and then adopt the deep deterministic policy gradients (DDPG) network to find the optimal task assignment solution with meeting the Service Level Agreements (SLAs) requirements. It makes the optimal scheduling decision only dependent on learning directly from its experience without any prior knowledge. The experimental results using the actual Alibaba cloud workload tracings show that compared with other existing solutions, our proposed task scheduling method can effectively reduce the average response time of tasks with guaranteeing the load balancing among VMs, when facing of dynamical workloads.**

***Keywords*—cloud computing, deep reinforcement learning, load balancing, service level agreements, task scheduling**

I. Introduction

With the repaid development of computer technologies and Internet economies, cloud computing becomes one of the most promising and valuable research directions following big data and artificial intelligent recently [1], [2]. It considers the hardware infrastructures, platform and software as multiply resources, and then provides those kind of services to customers through Internet. For example, Infrastructure as a Service (IaaS) is one of the basis form of cloud computing. One the one hand, the IaaS vendors (e.g., Amazon EC2, Microsoft Azure, and Alibaba Cloud) provide elastic hardware resources such as CPU, main memory, hard disk and network bandwidth to users via virtualization technologies [3]. On the other hand, different users can rent the hardware resources from IaaS vendors in the form of virtual machine(VM) on demand, and charge based on the actual usage. Hence, users can build their infrastructure easily and quickly with a lower cost. As a result, increasingly applications have migrated to the cloud for fast development and cost saving. Obviously, clouding computing with the feature of pay-as-you-go model brought subversive changes to the traditional business computing service model [2].

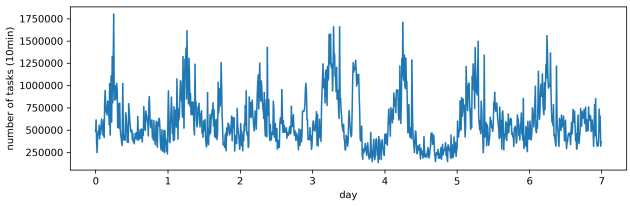

Fig. 1. Workload in Alibaba-Cluster-trace-v2018 [4]. It shows the number of tasks submitted by users every 10 minutes for 7 days.

However, there are multi-faceted challenges for effectively resource management in cloud environment. First, cloud workloads show highly dynamic variation and unexpected burst regarding of submitted task rate. Fig.1 shows Alibaba's 7-day cluster tracking data. It demonstrates that the workloads of the cloud cluster fluctuate significantly at different time of a day, with an average difference between peak and idle periods is about 1,801,547 tasks. Therefore, it requires to dispatch these large batches of tasks to the appropriate cloud nodes automatically for IaaS vendor. For application provider, it need to guarantee SLAs. Furthermore, the designed task scheduler needs to be robust to dynamic variations in submitted tasks. Second, the performance interference often occurs in cloud environment (e.g., multiple applications running on co-located VMs), since the contention of shared limited resources (i.e., processor, main memory, I/O disk and bandwidth). As a result, guaranteeing the Quality of Service (QoS) of each cloud-based applications is the priority target to cloud application providers [5]. Last but not least, IaaS vendor providers typically support users with multiple types of VM instances, such as CPU-intensive instance with high CPU and low memory, or Memory-intensive instance (i.e., high memory and low CPU). Hence, users can rent different types of VM instances with varied prices according to their actual needs. How to achieve the load balancing among the heterogeneous VM cluster is an

important issue to guarantee the SLAs in cloud environment [6].

To cope with the above challenges, a well-designed cloud task scheduling method is an important and popular tool. Generally, the function of task scheduling is to assign massive tasks submitted by users to the most suitable VM instance and meet the QoS requirement and SLAs as far as possible. Several solutions have been proposed. Some of these methods focus on scheduling problems for offline batch tasks, however they are not suitable for the highly dynamical workload case. For the online task scheduling method, existing solutions using heuristic algorithms or machine learning methods can only assign a single task at one specific time, which is inefficient to deal with the massive and dynamical workload case. If the scheduler accepts multiple tasks at the same time, it can only process them one by one, without considering the relationship between these tasks. In addition, most methods do not consider the high dynamics of tasks in a real cloud environment.

To do this, we propose a task scheduling framework based on DRL in this paper. Reinforcement Learning (RL) is an important branch of machine learning [7]. DRL can effectively solve the problem of state space and action space explosion problems in reinforcement learning (RL) by integrating the deep learning model. In here, we formulate the online task scheduling as a constrained dynamical optimization problem, then adopt the DDPG network to find the optimal task assignment solution. It can learn directly from its experience without any prior knowledge, making the appropriate scheduling decision to VMs for continuous online task requests. In detail, we take the information of VMs and the tasks received by the scheduler as the status of the agent. Through the defined states, the agent assigns each task to the appropriate VM adaptively, the response time of tasks is rewarded as the action of the agent. We use CloudSim Plus [8] to evaluate our proposed method in real workload scenarios. Compared with four benchmark methods, our proposed task scheduling method performs better in terms of task's average response time and VM's CPU utilization standard deviation (std).

The rest of this paper is organized as follows. In section II, we describe our proposed system architecture and propose the cloud task scheduling problem. Section III introduces the related theories of reinforcement learning and expounds the implementation of our proposed intelligent job scheduling algorithm. After that, we evaluate the proposed algorithm and discuss the experimental results in Section IV. We discusses related work in section V. Finally, some conclusions were drawn in Section VI and some suggestions are make for future work.

II. System Architecture

A common cloud computing scenario is considered in our task scheduling system. It includes three stakeholders, namely IaaS vendors, application providers and end users. Among them, IaaS vendors offer different kinds of VM instances on their public could platforms. Application providers construct their resource pool by renting VM instances from IaaS vendors. The applications are deployed on these rent VM instances and provide a variety of services to end user through Internet. Meanwhile, they continually receive various types of task submitted form end users. For end users, they enjoy the services from application providers, and have a right to claim the QoS requirements (e.g., response time or throughput). In such case, in order to guarantee the profits of application providers, an efficient task scheduler is a useful tool to meet diverse end users requirements in terms of QoS.

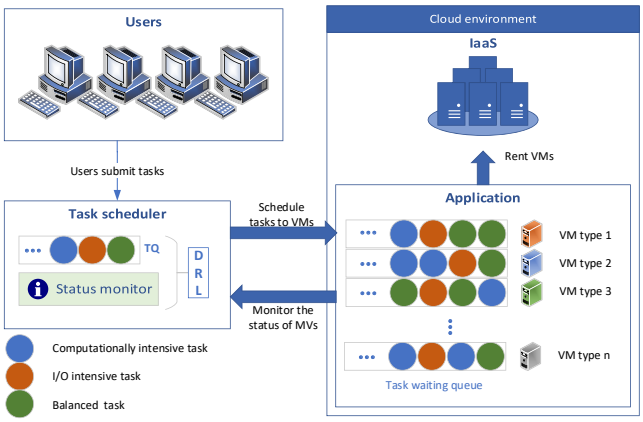

Fig. 2. Task scheduling framework.

The system architecture is shown in Fig.2. On the one hand, in order to depoly the application on cloud, the application provider rents several VM instances with varied configurations from IaaS vendor. Each VM maintains a task wait queue for received tasks. In here, we assume that each VM executes all assigned tasks in the first come first served (FCFS) way, rather than a non-preemptive way. On the other hand, the tasks submitted by end users are assigned to different VMs through the task scheduling module.

In our architecture, users allow to submit multiple types of tasks at the same time. These tasks can be different types of tasks, such as computation-intensive task, I/O-intensive task and other type task. Please noted that the time, quantity, and type of tasks submit by users are all unpredictable.

The task scheduling module consists of a task queue (TQ), a status monitor (SM) and a DRL-based task scheduler. The function of TQ is mainly to record the task submitted by end users, and the TQ allows for multiple types of tasks. The SM is used to obtain status information of all VMs, which includes VM's MIPS (Million Instructions Per Second), RAM utilization, waiting time of task. After that, we sent the length and CPU utilization of each task in the TQ, and information in SM into the DRL-based task scheduler as input, and the output of task scheduler is the tasks assignment solution. Finally, the task scheduler assigns the task in TQ to a suitable VM for execution. After each round of allocation, TQ will be emptied to prepare for the next round.

III. Scheduling Algorithm

In this section, we propose a task scheduling algorithm based on deep reinforcement learning, which can guarantee

the load balancing among VMs under the SLAs constraint. We first give a mathematical representation of our problem, and then introduce the proposed algorithm in details.

A. Problem Formulation

In our scheduling model, tasks are submitted online. The scheduler does not know the size and number of tasks to be reached in advance. Based on the above assumptions, each task can be expressed as

$$task_i = (mi_i, cpuUtilization_i), \tag{1}$$

where i indicates the number of the task, mi_i represents the task's length (i.e., the number of millions of instructions, MI), and $cpuUtilization_i$ represents the CPU usage of the task.

The application provider rents VMs with different configurations from the IaaS vendor. Each VM instance is defined as

$$vm_j = (mips_j, ram_j), \tag{2}$$

where j indicates the number of the VM, $mips_j$ represents how many million instructions are executed per second by the VM, ram_j represents the size of the VM's memory. Supposing the task scheduler assigns $task_i$ to mv_j, then the expected execution time of $task_i$ is

$$exT_{ij} = \frac{mi_i}{cpuUtilization_i \times mips_j}, \tag{3}$$

the waiting time of $task_i$ is

$$waT_{ij} = \sum_{k=1}^{C} exT_{kj}, \tag{4}$$

where C means the number of tasks in the waiting queue in vm_j. In here, the task's response time $resopnseTime_{ij}$ is defined as the total amount of time that $task_i$ stays in the vm_j. It can be calculated by the following equation:

$$responseTime_{ij} = waT_{ij} + exT_{ij}. \tag{5}$$

Furthermore, the average response time for assigning I tasks $task_I$ to J VMs vm_J is

$$avgRT_{IJ} = \frac{\sum_{i=1}^{I} responseTime_{ij}}{I}, \ \forall j = 1, \ldots, J. \tag{6}$$

The CPU utilization of vm_j at time t is $utilization_j^t$, and the average CPU utilization of J VMs at time t is

$$avgU_J^t = \frac{\sum_{j=1}^{J} utilization_j^t}{J}. \tag{7}$$

The task scheduler has a total of T times task assignments. The average of the CPU utilization of these T times is

$$avgU_J^T = \frac{\sum_{t=1}^{T} avgU_J^t}{T}. \tag{8}$$

The std of CPU utilization of J VMs at all times is

$$stdU_J^T = \sqrt[2]{\frac{1}{T}\sum_{t=1}^{T}\left(avgU_J^t - avgU_J^T\right)^2}. \tag{9}$$

Based on the formulations (6) and (9), the optimal returns can be getting by minimizing the average response time and CPU utilization std, it can be expressed as follows:

$$\min\left(\frac{\sum_{i=1}^{I}\left(exT_{ij} + \sum_{k=1}^{C} exT_{kj}\right)}{I} + \sqrt[2]{\frac{1}{T}\sum_{t=1}^{T}\left(avgU_J^t - avgU_J^T\right)^2}\right) \tag{10}$$

Subject to

$$\begin{cases} \forall i = 1, \ldots, I, \\ \forall j = 1, \ldots, J, \\ \forall t = 1, \ldots, T. \end{cases} \tag{11}$$

To achieve optimal returns, an efficient online task scheduler is needed to dispatch tasks to the appropriate VM instances.

B. Background of Reinforcement Learning and DRL

The basic idea of RL is learning through interaction [9]. An RL agent interacts with its environment, and upon observing the consequences of its actions, can learn to alter its own behavior in response to rewards received. The best sequence of actions is determined by the rewards provided by the environment. Each time the environment transitions to a new state s_{t+1} it also provides a scalar reward r_{t+1} to the agent as feedback. The goal of the agent is to learn a policy (control strategy) π that maximizes the expected return (cumulative, discounted reward).

The original deep Q-network (DQN) can only be used in tasks with a discrete action space. To extend it to continuous control, [10], [11] proposed DDPG to use deep neural networks on the actor-critic reinforcement learning method, where both the policy and value of the reinforcement learning are represented through hierarchical networks.

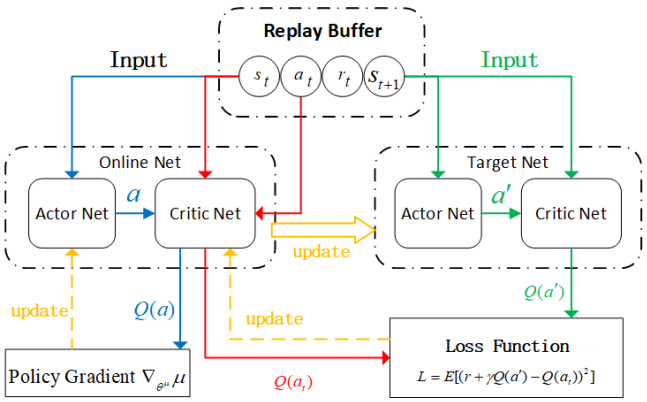

Fig. 3. DDPG algorithm framework.

The network framework of DDPG is shown in Fig.3. The DDPG algorithm maintains a parameterized actor function $\mu(s|\theta^\mu)$ which specifies the current policy by deterministically

mapping states to a specific action. The actor parameters are updated by

$$\begin{aligned}\nabla_{\theta^\mu}\mu &\approx \mathbb{E}_{\mu'}\left[\nabla_{\theta^\mu}Q\left(s,a|\theta^Q\right)|_{s=s_t,a=\mu(s_t|\theta^\mu)}\right] \\ &= \mathbb{E}_{\mu'}\left[\nabla_a Q\left(s,a|\theta^Q\right)|_{s=s_t,a=\mu(s_t)}\nabla_{\theta_\mu}\mu\left(s|\theta^\mu\right)|_{s=s_t}\right].\end{aligned} \tag{12}$$

The critic $Q(s,a)$ is learned using the Bellman equation as in Q-learning. The Target Net is a copy of the actor and critic network, its role is to improve the stability of learning. Suppose the Main Net's parameter is θ, the Target Net's parameter is θ', θ' is then updated by (13) with $\tau \ll 1$.

$$\theta' \leftarrow \tau\theta + (1-\tau)\,\theta' \tag{13}$$

Additionally, to make efficient use of hardware optimizations, it is essential to learn in minibatches, rather than online. And it maintains a replay buffer, at each timestep, the actor and critic network are updated by sampling a minibatch uniformly from the buffer.

C. DDPG-Based Task Scheduling Algorithm

Cloud computing environments are characterized by large scale and scalability. The workload of cloud-based applications is unpredictable and highly dynamical. It is impossible for us to adapt traditional scheduling methods to such flexible scenarios. RL can learn in the case of zero samples through interaction with the environment, and automatically adapt to the environment as the environment changes, hence RL is particularly suitable for cloud task scheduling. DRL enhances the representation ability of traditional RL, which can solve the complex decision problem of high-dimensional state space and action space in cloud task scheduling.

In the following, we introduce our proposed DDPG-based task scheduling algorithm (see Algorithm 1). We explain the two parts separately: online decision and offline training.

1) ONLINE DECISION: The task of online decision is to assign the tasks submitted by users to the appropriate VMs. When the user submits a batch of tasks, the agent will observe the state of the environment and assign these tasks to the appropriate VMs according to the current policy. In addition, the agent will record the reward for this task scheduling action.

In our model, the scheduler can assign tasks to any VM, and each VM can accept any number of tasks. Thus, the action space can be represented as a vector whose length is the number of tasks submitted at a time:

$$action_{IJ} = \left[vmid^1, \ldots, vmid^i, \ldots, vmid^I\right]. \tag{14}$$

It means assigning I tasks to J VMs, and $vmid^i$ represents the number of the VM assigned to $task_i$.

In the scheduling algorithm, when the agent is aware of the environment, the actual perceived state of the environment includes the state of the tasks and the state of the VMs. More specifically, state of $task_i$ consist mi_i and $cpuUtilization_i$. For VM j, its state consists 2 elements: $mips_j$, waT_j. Therefore, the current state can be expressed as

$$\begin{aligned}state_{IJ} =& [mi_i, cpuUtilization_i, \ldots, mips_j, waT_j, \ldots] \\ & i = 1, \ldots I,\ j = 1, \ldots, J.\end{aligned} \tag{15}$$

Algorithm 1 DDPG-Based Task Scheduling Algorithm

Randomly initialize critic network $Q(s,a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$.
Initialize replay buffer $\mathfrak{R}$.
Initialize exploration probability ϵ, exploration steps n.
for each batch of task $task_I$ arrive at time $t = 1, \ldots, T$ **do**
  Receive observation state s_t
  if $t > n$ **then**
    Sample a random minibatch of $\mathcal{N}$ transitions (s_i, a_i, s_{i+1}) from $\mathfrak{R}$, which all the selected tasks have completed
    Calculate $avgRT_{IJ}^i$ according to $responseTime$ of the selected tasks
    Calculate r_i according to $avgResponseTime_{IJ}^i$
    Set $y_i = r_i + \lambda Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q)^2)$
    Update the actor policy using the sampled gradient:

$$\nabla_{\theta^\mu}\mu|s_i \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s=s_i,a=\mu(s_i)}\nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

    With probability ϵ generate a action $a_t = \mu(s_t|\theta^\mu)$ and $1-\epsilon$ generate a random action a_t
  else
    Generate a random action a_t
  end if
  Transform the action a_t by Algorithm 2
  Schedule tasks to virtual machines according to action a_t and observe reward
  Store transition (s_t, a_t) in $\mathfrak{R}$
  Store completed task's $responseTime$ in $\mathfrak{R}$
end for

In reinforcement learning, rewards are feedback on a scheduling strategy that allows agents to learn better strategies. In our model, we use the average response time of a batch of tasks as reward. Thus for each batch of tasks, its reward is given by

$$reward_{IJ} = \omega \times avgRT_{IJ}, \tag{16}$$

where ω is a control factor. In fact, we want to minimize the $reward_{IJ}$.

Unlike the original DDPG algorithm, the technique used here is more like the trick in Deep Q-learning. We use ϵ-greedy policy to generate the action. With probability ϵ generate an action $a_t = \mu(s_t|\theta^\mu)$ and $1-\epsilon$ generate a random action a_t, the ϵ will gradually decrease as the number of training

rounds increases, so that agent will be more inclined to use its own experience to make decisions with its decision-making experience builds up.

In order to promote load balancing of each VM, we transform the actions generated by the network through Algorithm 2. This can also achieve the purpose of increasing noise and realizing the exploration of the agent. Through our experiments, we found that this can increase the stability of the algorithm.

Algorithm 2 Action Transformation Algorithm

Require: action_list, vm_list
Ensure: action_list
1: $X = \lceil \frac{len(action\_list)}{len(vm\_list)} \rceil$
2: **for** $i = 0, \ldots, len(action\_list)$ **do**
3: **if** action_list[i] of occurrences in the action_list $> X$ **and** randomly generated probability > 0.5 **then**
4: action_list[i] = randomly generate an integer in the range 0 to len(vm_list)
5: **end if**
6: **end for**
7: **return** action_list

State is the agent's perception of the environment when making scheduling decisions. All state transitions occur when the user submits a batch of tasks, at which point the state transitions from s_t to s_{t+1}.

2) OFFLINE TRAINING: Offline training has two purposes. One is to make the critic network more accurate in scoring the actions generated by the action network, and the other is to make the actions generated by the action network score higher. In order to achieve this goal, we used experience replay [12] and target network [10].

Experience replay's main role is to overcome the problem of correlation and non-stationary distribution of empirical data. We use a fixed size replay buffer $\mathfrak{R}$ to solve this problem. At each timestep, we store (s_t, a_t) and completed task's responseTime in $\mathfrak{R}$, and randomly select a fixed size sample from to train. Because DDPG is an off-policy algorithm, the replay buffer can be large, allowing the algorithm to benefit from learning across a set of transitions uncorrelated transitions.

Directly implementing Q-learning with neural networks proved to be unstable in many environments. Target network can eliminate the divergence and oscillations of DNN parameters during training processes. We created a copy of the actor and critic networks named target network, and used it to slowly modify the online networks instead of directly copying the weights.

IV. EXPERIMENTS

A. Experiment setup

We evaluate our proposed task scheduling algorithm through a lot of experiments in this section. All experiments are conducted on a tower server, which includes 2.1GHz Intel Xeon E5 CPU, 250GB RAM, Ubuntu 18.04LTS operation system, JDK1.8, Python3.6, PyTorch 1.0 and CloudSim Plus 4.0.

CloudSim Plus is an emerging simulation open-source package, it is widely used in [13], [14]. Based on it, we build our simulation environment for evaluating the performance of proposed task scheduling method. We consider that an application provider rents 20 long-term reserved VMs from IaaS vendor, and serves lots of end users. As shown in Table I, these VM instances are divided into 4 categories for handling different types of tasks, such as computation intensive and I/O intensive tasks.

TABLE I
VM SETTINGS

	MIPS	RAM(MB)
VM1	2000	1024
VM2	1000	1024
VM3	500	2048
VM4	250	2048

Alibaba-Cluster-trace-v2018 is used in this paper as the tested workload case. It includes about 4000 VMs in a periods of 8 days. The content of each record (i.e, task) in the file batch_instance.csv includes commit time, task length, cup utilization, and ram utilization. Through CloudSim Plus, tasks can be submitted dynamically. To simplify our experiments, we selected data for the first few hours of the second day, with up to 50 tasks a time being submitted to the data center. We divide the data into two data sets, which represent the application scenarios with lower and higher workload fluctuation.

In our proposed task scheduling algorithm, we adopt Actor_eval, Critic_eval, Actor_target, Critic_target four neural networks, each neural network has four fully connected layers. Both of Actor_eval and Critic_eval networks updated in real time, while the Actor_target and Critic_target networks will be update by using (13) with $\tau = 0.01$. We set the capacity of replay memory $\mathfrak{R} = 1000$, the size of minibatch $\mathcal{N} = 64$. We use the Adam optimizer to optimize the network. The learning rate for Actor_eval and Critic_eval network are 0.0006 and 0.001 respectively. After 400 steps of exploration, the network start to train, the probability of exploration is reduced from 0.95 to 0.05, and the rate of decline is 0.996.

To evaluate the performance of our proposed algorithm (DDPG), we compare it with four existing solutions: random task scheduling approach, round-robin task scheduling approach, earliest scheduling approach, and deep Q-learning scheduling approach.

Random task scheduling approach (Random) is a common scheduling method that randomly assigns tasks submitted by users to VMs. Round-robin task scheduling approach (Round-robin) is an effective scheduling method that assigns tasks to each VM in turn. Earliest scheduling approach (Earliest) is a greedy algorithm, it assigns task to the first idle VM according to the arrival time of the task. Because Random, Round-robin and Earliest are standard algorithms in task scheduling, many

researchers adopt them as the baseline methods. Deep Q-learning (DQN) task scheduling approach is based on reinforcement learning [15].

B. Experiment on lower-fluctuation workload

First, we evaluate the performance of DDPG in the face of lower-fluctuation workload. Fig.4 shows the change in lower-fluctuation workload on the second day. The length of the tasks in this data set is between 10MI and 300MI. The maximum number of tasks submitted per 10 minutes in this data set is 29,576, the minimum is 11,392, the maximum task length is 3,931,416MI, and the minimum is 517,536MI. In this paper, we intercepted the first 500,000 pieces of data as our data set.

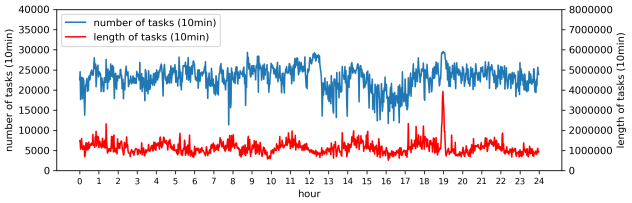

Fig. 4. The second day's lower-fluctuation workload in Alibaba-Cluster-trace-v2018. It shows the fluctuations in the number and total length of tasks submitted by users every 10 minutes.

In our experiments, we are concerned with task response time and VM's CPU utilization std. The results are shown in Fig.5. Due to the randomness of the experiment, we performed 10 repeated experiments on each algorithm and finally recorded the average results. For DQN, the state is a combination of the state of each task and the state of the VM.

Fig.5(a) shows the average response time for tasks submitted at different time periods. We can summarize four conclusions: (1) The average response time varies with the number and length of tasks submitted by the user. Since the VM's ability to handle tasks under higher workloads is diminished, this can result in longer response times for tasks assigned to the VMs. (2) The proposed algorithm has an average response time greater than the other algorithms in the first period of time, because it is exploring and training the neural network, but it converges quickly, partly because of Algorithm 2. (3) The average response time generated by different task scheduling methods is different. It can be seen from Fig.5(b) that the average response time generated by our proposed method is smaller than the rest in most cases. This means that the method we proposed is better than the other methods. (4) The DQN algorithm performs poorly in this scenario because it does not take into account the relationship between the same batch of tasks as DDPG does.

In addition, the CPU utilization std is shown in Fig5. (c). We can see that The CPU utilization std of our proposed algorithm is the lowest, which is 27% better than the worst. It demonstrate that that our proposed algorithm can promote load balancing of VM.

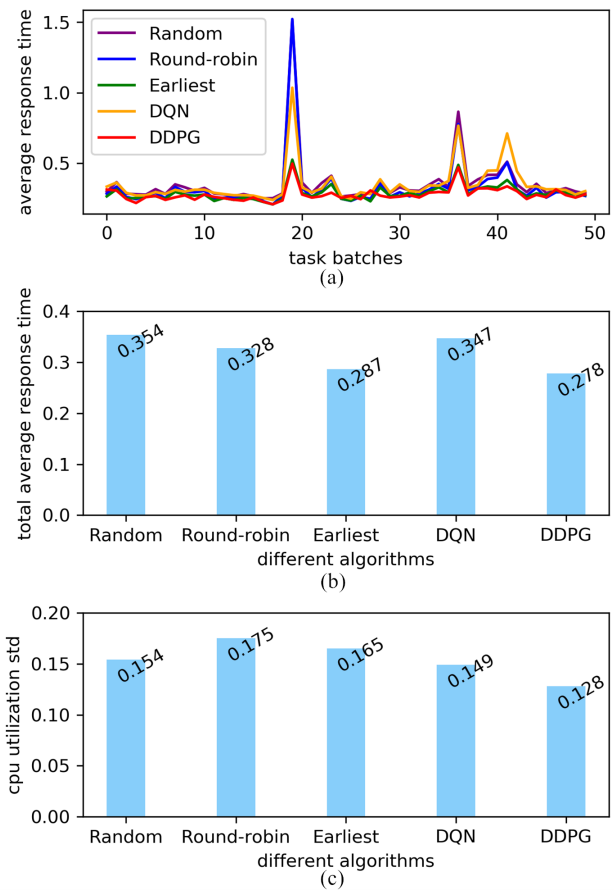

Fig. 5. The performance of different task scheduling algorithms on lower-fluctuation workload. Fig.5(a) shows the average response time for every 10,000 tasks. Fig.5(b) shows total average response time of 500,000 tasks. Fig.5(c) shows the CPU utilization std comparison of VMs in different scheduling algorithms.

C. Experiment on higher-fluctuation workload

In this section, we keep the task of length range from 10MI to 3000MI in the second day's data set to increase the strength and volatility of cluster workloads.

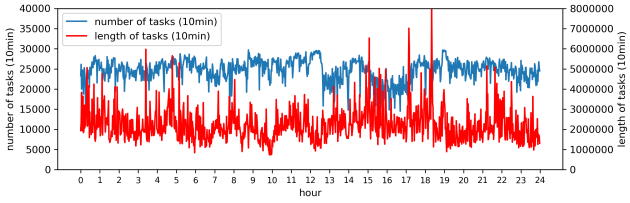

Fig. 6. The second day's higher-fluctuation workload in Alibaba-Cluster-trace-v2018. It shows the fluctuations in the number and total length of tasks submitted by users every 10 minutes.

Fig.6 shows the change in higher-fluctuation workload on the second day. The maximum number of tasks submitted per 10 minutes in this data set is 29,691, the minimum is 13,276; the maximum task length is 7,985,645MI, and the minimum is 730,726MI. It can be seen from the figure that the data

set is much larger than lower-fluctuation workload data set as shown in Fig.4, regardless of the intensity or fluctuation of the workload. We intercepted the first 500,000 pieces of data as our data set.

As shown in Fig.7(a), the average response time of the task in all algorithms has increased significantly compared to Fig.5, due to the increased workload. This magnifies the performance difference between the different methods. It can be seen that the proposed method is more adaptable to the scenario where the workload fluctuates significantly. Earliest is still the best performing algorithm except the DDPG, but our proposed algorithm has increased the average response time by 39% as shown in Fig.7(b). This is due to our proposed algorithm takes all the tasks in the TQ into consideration, and assigns the task to the most suitable VM. The random algorithm is the worst performer due to it can not assign tasks to the right VMs, for example, assign computationally intensive tasks to VMs with strong computing power. In comparison, our proposed algorithm reduces the average response time by 77%.

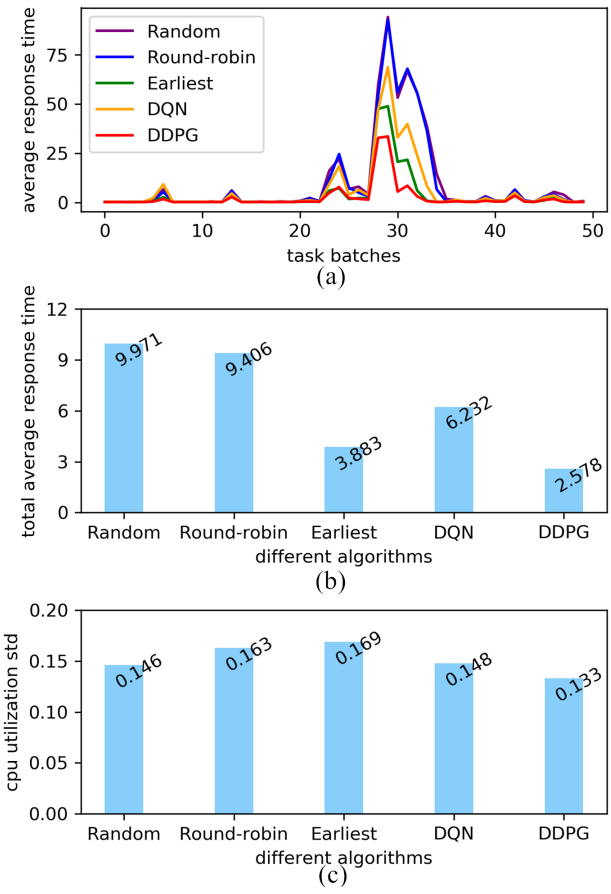

Fig. 7. The performance of different task scheduling algorithms on higher-fluctuation workload. Fig.7(a) shows the average response time for every 10,000 tasks. Fig.7(b) shows total average response time of 500,000 tasks. Fig.7(c) shows the CPU utilization std comparison of VMs in different scheduling algorithms.

Algorithm 2 is our optimization of the load balancing of the cluster. In our experiments, it plays a very good role. The VM's CPU utilization std of the algorithm our proposed in Fig.7(c) is lower than the rest of the algorithm, which is 21% better than the worst. This shows that our proposed algorithm can better guarantee load balancing in the case of high workload fluctuations.

V. Related Work

Achieving load balancing between VMs without sacrificing the performances of applications in the cloud environment is an important and challenge research issue. In this context, a well-designed cloud task scheduling method is considered as one of the most promising and efficient way to address the problem. A number of task scheduling approaches based on variant methods and ideas have been proposed, which heuristic algorithms are the popular and widely used in cloud task scheduling [16]–[21]. For example, Ebadifard et al. [16] proposed a static task scheduling method based on the particle swarm optimization (PSO) algorithm, it improved the performance of the basic PSO method by using a load balancing technique. Huang et al. [18] proposed a task assignment algorithm based on particle swarm optimization and simulated annealing (PSO-SA) in Ad-hoc mobile cloud. Keshanchi et al. [19] proposed an improved genetic algorithm, it completed the task scheduling of the DAG graph via the hybrid variation of chromosomes. Cheng et al. [21] proposed a heterogeneity-aware task assignment approach, E-Ant, which aims to improve the overall energy consumption in a heterogeneous Hadoop cluster without sacrificing job performance. These above heuristics based methods shows good performance in the offline task scheduling cases. However, they are inefficient to achieve online task scheduling in cloud.

With the repaid development of artificial intelligence in recent years, the machine learning method is increasingly applied to cloud task scheduling, such as [15], [22]–[27]. In particular, reinforcement learning (RL) is popular machine learning method, which is used to solve challenging decision problems [28]. It learns the optimal strategy for accomplishing the goal by maximizing the cumulative reward value, where the agent obtains from the environment. Hence, it has been widely used in cloud computing environments. Farahnakian et al. [24] proposed a Reinforcement Learning-based Dynamic Consolidation method (RL-DC) to minimize the number of active hosts, according to the current resources requirement. Barrett et al. [25] dynamically scales the resources of applications running on the IaaS cloud through the parallel Q-learning approach. Peng et al. [26] design a task scheduling scheme based on reinforcement learning and queuing theory, and it reduces the dimension of the state space by clustering the states. However, those traditional RL based solutions are subject to dimensional constraints. Deep reinforcement learning (DRL) has made effective progress in addressing the curse of dimensionality [27]. Wei et al. [15] proposed an intelligent QoS-aware job scheduling framework, it uses deep Q-learning algorithm to realize online scheduling of single task. However, this method can not solve the problem of action space explosion effectively. Therefore, this paper designs a

DDPG-based online task scheduling framework, which can effectively solve the problem of action space explosion.

VI. Conclusion

For IaaS vendors and cloud-based application providers, load balancing without sacrificing the performances of cloud-based applications is a highly changing problem in cloud computing. In order to address this issue, we propose an online task scheduling framework, it consist of a task queue, states monitor and task scheduler. The main purpose of task scheduler is to dispatch massive and highly dynamical tasks to limit resources under the SLAs requirement. Furthermore, we formulate the online task scheduling as a dynamical optimal problem with constraints. Benefit from the powerful perception and decision-making ability of DRL in automatic control problems, we adopt the DDPG network to find the optimal task assignment solution, which can dynamically adapt to the uncertainty and highly fluctuation workloads. We evaluate our proposed algorithm in two real world workload scenarios, compared to several other solutions, our proposed method has a shorter average task response time with meeting the load balancing target.

Acknowledgment

This research is partly supported by the National Natural Science Foundation of China under Grants 61602434, in part by Chongqing research program of key standard technologies innovation of key industries under grant cstc2017zdcy-zdyfX0076, cstc2018jszx-cyztzxX0025, in part by Youth Innovation Promotion Association CAS, No.2017393.

References

[1] P. Li, J. Li, Z. Huang, C.-Z. Gao, W.-B. Chen, and K. Chen, "Privacy-preserving outsourced classification in cloud computing," *Cluster Computing*, vol. 21, no. 1, pp. 277–286, 2018.

[2] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of "big data" on cloud computing: Review and open research issues," *Information systems*, vol. 47, pp. 98–115, 2015.

[3] S. Xu, L. Liu, L. Cui, X. Chang, and H. Li, "Resource scheduling for energy-efficient in cloud-computing data centers," in *2018 IEEE 20th International Conference on High Performance Computing and Communications(HPCC)*. IEEE, 2018, pp. 774–780.

[4] "Alibaba-cluster-trace-v2018," https://github.com/alibaba/clusterdata/blob/master/cluster-trace-v2018, 2018.

[5] X. Shi, J. Dong, S. M. Djouadi, Y. Feng, X. Ma, and Y. Wang, "Papmsc: power-aware performance management approach for virtualized web servers via stochastic control," *Journal of Grid Computing*, vol. 14, no. 1, pp. 171–191, 2016.

[6] S. K. Garg, S. K. Gopalaiyengar, and R. Buyya, "Sla-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter," in *International conference on Algorithms and architectures for parallel processing*. Springer, 2011, pp. 371–384.

[7] S. Burr, "Synthesis lectures on artificial intelligence and machine learning," *Act Learn*, vol. 6, no. 1, pp. 1–114, 2012.

[8] "Cloudsim plus," http://cloudsimplus.org, 2018.

[9] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.

[10] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[11] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning*, 2016, pp. 1329–1338.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[13] M. C. Silva Filho, R. L. Oliveira, C. C. Monteiro, P. R. Inácio, and M. M. Freire, "Cloudsim plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2017, pp. 400–406.

[14] D. P. Abreu, K. Velasquez, M. R. M. Assis, L. F. Bittencourt, M. Curado, E. Monteiro, and E. Madeira, "A rank scheduling mechanism for fog environments," in *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE, 2018, pp. 363–369.

[15] Y. Wei, L. Pan, S. Liu, L. Wu, and X. Meng, "Drl-scheduling: An intelligent qos-aware job scheduling framework for applications in clouds," *IEEE Access*, vol. 6, pp. 55 112–55 125, 2018.

[16] F. Ebadifard and S. M. Babamir, "A pso-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 12, p. e4368, 2018.

[17] A. Al-Maamari and F. A. Omara, "Task scheduling using pso algorithm in cloud computing environments," *International Journal of Grid and Distributed Computing*, vol. 8, no. 5, pp. 245–256, 2015.

[18] B. Huang, W. Xia, Y. Zhang, J. Zhang, Q. Zou, F. Yan, and L. Shen, "A task assignment algorithm based on particle swarm optimization and simulated annealing in ad-hoc mobile cloud," in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2017, pp. 1–6.

[19] B. Keshanchi, A. Souri, and N. J. Navimipour, "An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing," *Journal of Systems and Software*, vol. 124, pp. 1–21, 2017.

[20] Y. Xu, K. Li, T. T. Khac, and M. Qiu, "A multiple priority queueing genetic algorithm for task scheduling on heterogeneous computing systems," in *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*. IEEE, 2012, pp. 639–646.

[21] D. Cheng, X. Zhou, P. Lama, M. Ji, and C. Jiang, "Energy efficiency aware task assignment with dvfs in heterogeneous hadoop clusters," *Ieee transactions on parallel and distributed systems*, vol. 29, no. 1, pp. 70–82, 2018.

[22] Y. Wang, H. Liu, W. Zheng, Y. Xia, Y. Li, P. Chen, K. Guo, and H. Xie, "Multi-objective workflow scheduling with deep-q-network-based multi-agent reinforcement learning," *IEEE Access*, 2019.

[23] 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*. IEEE Press, 2018, pp. 129–134.

[24] F. Farahnakian, P. Liljeberg, and J. Plosila, "Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning," in *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, 2014, pp. 500–507.

[25] E. Barrett, E. Howley, and J. Duggan, "Applying reinforcement learning towards automating resource allocation and application scalability in the cloud," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 12, pp. 1656–1674, 2013.

[26] Z. Peng, D. Cui, J. Zuo, Q. Li, B. Xu, and W. Lin, "Random task scheduling scheme based on reinforcement learning in cloud computing," *Cluster computing*, vol. 18, no. 4, pp. 1595–1607, 2015.

[27] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.

[28] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.