Performance and Cost-Aware Task Scheduling via Deep Reinforcement Learning in Cloud Environment

Zihui Zhao[1,2], Xiaoyu Shi[1,3(✉)], and Mingsheng Shang[1,3]

[1] Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China
xiaoyushi@cigit.ac.cn
[2] Beihang University, Beijing 100191, China
[3] School of Automotive Software, Chongqing School, University of Chinese Academy of Sciences, Chongqing 400714, China

Abstract. In the cloud computing environment, task scheduling with multiple objectives optimization becomes a highly challenging problem in such a dynamic and bursty environment. Previous studies have mostly emphasized assigning the incoming tasks in a specific scenario, with a weak generalization ability to various objectives automatically. Thus, they suffer the inefficient issue under large-scale and heterogeneous cloud workloads. To address this issue, we propose a deep reinforcement learning (DRL)-based intelligent cloud task scheduler, which makes the optimal scheduling decision only dependent on learning directly from its experience without any prior knowledge. We formulate task scheduling as a dynamical optimization problem with constraints and then adopt the deep deterministic policy gradients (DDPG) network to find the optimal task assignment solution while meeting the performance and cost constraints. We propose a correlation-aware state representation method to capture the inherent characteristics of demands, and a dual reward model is designed to learn the optimal task allocation strategy. Extensive experimental results on Alibaba cloud workloads show that compared with other existing solutions, our proposed DDPG-based task scheduler enjoy superiority and effectiveness in performance and cost optimization.

Keywords: Cloud computing · Task scheduling · Deep reinforcement learning · Cost optimization · Performance improvement

1 Introduction

Cloud computing is the most popular computing paradigm in IT society [1]. With virtualization technologies, the data center can easily abstract the different hardware infrastructures as a larger resource pool and provides elastic hardware resources as services to users through the Internet. For instance, Amazon EC2[1],

[1] https://aws.amazon.com/ec2/.

J. Troya et al. (Eds.): ICSOC 2022, LNCS 13740, pp. 600–615, 2022.
https://doi.org/10.1007/978-3-031-20984-0_43

Microsoft Azure[2], and Alibaba Cloud[3] offers customized hardware resource to customers in the form of virtual machine (VM), and charge based on actual usage. As a result, increasingly services have migrated to the cloud environment for fast development and cost-saving.

Regarding of cloud environment, the effective task scheduling of cloud services is one of the key enablers of large-scale cloud systems [2]. However, the unique features of the cloud environment make task scheduling among VMs more challenging. Firstly, the cloud workload featured highly dynamic variation and unexpected bursts. Thus, it requires a designed task scheduler with the characteristic of robustness against the unexpected burst in incoming cloud tasks. Secondly, the problem of task scheduling in cloud environment is NP-hard problem. Furthermore, the datacenter usually offers users various types of VM instances to meet users' customized requirements. Each type of VM has its own pricing model. Thus, the large-scale task scheduling problem becomes more complicated in such a heterogeneous environment. Last but not least, task scheduling in the cloud environment is a multi-objective optimization problem and is related to the profits of multiple stakeholders. For the date center, it is expected to maximize the utilization ratio of hardware resources. The cloud services providers emphasize minimizing the usage cost of rented VM instances, while the end users concern more about the service experience offered by cloud services providers. Thus, how to design an efficient task scheduler to guarantee the profits of different stakeholders is also an important problem.

Following this, several solutions have been proposed with using heuristic-based algorithms [3–7]. Some of these methods focus on scheduling problems for offline or static batch tasks, which are not incapable of the dynamic workloads in real-time scenarios. For the online task scheduling method, existing solutions using meta-heuristic algorithms can only assign tasks sequentially, which is inefficient to deal with the massive and dynamic workload case. Meanwhile, heuristic-based task scheduling methods emphasize on a specific scenario, lacking the generalization ability to adapt to a wide range of objectives. Meanwhile, they cannot utilize the inherent characteristics of workloads to improve the optimization effect.

To this end, we propose an effective task scheduling based on the deep reinforcement learning (DRL) framework in this paper. We consider online task scheduling as a constrained dynamical optimization problem. We formulate it as a Markov decision process (MDP) model, then adopt the deep deterministic policy gradient (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 for VMs for continuous online task requests. The main contributions of this paper include:

– We propose an RL model of the task scheduling problem in cloud environments. We also formulate the state representation and rewards to train DRL

[2] https://azure.microsoft.com/en-ca/.
[3] https://www.alibabacloud.com/.

agents to satisfy load balancing, reduce average response time and optimize the usage cost of a cloud cluster.
- We design a correlation-aware state representation method that leverages the Pearson's correlation coefficient (PCC) and standard deviation of the resources demands (STD) to help perceive the feature of the workloads. A dual-reward model is designed to improve the effectiveness of learning the optimal policy.
- We conduct extensive experiments on real-world workload trace. It clearly demonstrates the superiority of our method over other state-of-the-art approaches.

2 Related Works

Scheduling tasks in the cloud environment is an essential and challenging problem, which has been studied for decades. To improve the performance of cloud datacenter under the constraints of the Service Level Agreements (SLAs), various task assignment algorithms and approaches are proposed [8–10]. Several solutions view the task scheduling problem as an NP-hard problem, whose goal is to optimize the task assignment in a stable environment. Hence, some heuristic or meta-heuristic methods have been proposed to solve the task scheduling problem, such as SARO [6] and hybridized BA [7]. For instance, Luo et al. [5] proposed a Correlation-Aware Heuristic Search(CAHS) method to detect the inherent correlations of the demands of different types of computing resources. Note that, most heuristic-based solutions only focus on specific scenarios, which limits their generalization ability in a highly dynamic environment.

Recently, several reinforcement learning (RL) methods have been applied to cloud task scheduling [11,12]. Compared to the heuristic-based methods that focus on maximizing the immediate (short-term) reward, RL-based method can help cloud services learn the long-term optimal task scheduling policy on-the-fly. For example, Wei et al. [13] proposed a QoS-aware job scheduling method for applications in a cloud deployment. DeepRM [14] used REINFORCE, a policy gradient DeepRL algorithm for multi-resource packing in cluster scheduling. Rjoub et al. [15] combined DRL with LSTM to address large-scale workloads. However, the cluster of this model is assumed to be homogeneous. Therefore, they can not adapt to various scenarios easily and be widely used in the real-world environment.

In summary, most of the existing methods can only schedule single task at a time. Furthermore, these works only focus on performance improvement. In addition, most of the previous solutions only adapt to some specific scenarios, and cannot be generalized to adapt to various scenarios. In contrast, our method can schedule multiple tasks simultaneously. By perceiving the status of the batch tasks as a whole, it can use Pearson's correlation coefficient (PCC) and standard deviation of the resource demand to evaluate the feature of the workload and therefore come up with better strategies. What's more, it can adapt to multiple optimization objectives such as task response time and cost-efficiency.

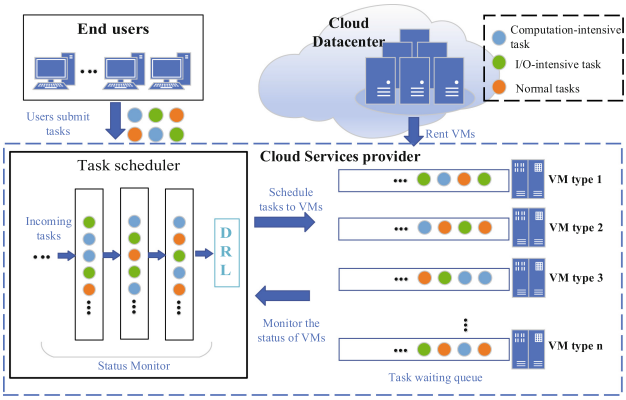


Fig. 1. Task scheduling framework in the cloud environment

3 System Architecture and Problem Statement

3.1 Overview of System Architecture

A typical cloud computing scenario is considered in this paper. It involves three stakeholders: datacenter, cloud services providers and end users. Figure 1 is the system architecture. On the cloud side, each VM maintains a task wait queue for received tasks. Here, we assume that each VM executes all assigned tasks in a non-preemptive way (i.e.,the first come first served way). On the other hand, users are allowed to submit multiple types of tasks at the same time. These tasks can be of different types, such as Computation-intensive tasks, I/O-intensive tasks ,and normal tasks. These tasks can be assigned to different VMs through the designed task scheduler. Please note that the arrival time, quantity, and type of tasks submitted by users are all unpredictable.

The task scheduler consists of three core components: a task queue (TQ) module, a status monitor (SM) module ,and a DRL-based scheduler module. The function of TQ is mainly to store heterogeneous tasks submitted by different end-users over time. The SM is used to collect status information of all assigned tasks and VMs, including task length, CPU utilization of each task, VM's MIPS (Million Instructions Per Second), RAM utilization, and waiting time of a task. For scheduler module, it is responsible for calculating the optimal task assignment solution, based on the collected status information from SM. After that, according to the result of the scheduler module, the tasks in TQ are assigned to suitable VM instances automatically for execution.

3.2 Problem Formulation

For the convenience of expression, we first define the related symbol in our model, as shown in Table 1. Then we formalize the cloud task scheduling as a constrained dynamic optimization problem.

Table 1. Definition of symbols

Symbol	Definition
J	The total number of VMs in the cluster
I	The total number of tasks in a batch
$speed$	The data transportation speed
vm_j	The jth VM in the cluster $1 \leq j \leq J$
$mips_j$	The speed that the vm_j executes the instructions
ram_j	The size of $vm_j's$ memory
$price_j$	The price of the vm_j per second
$task_i$	The ith task in the task batch $1 \leq i \leq I$
mi_i	The task length of $task_i$
cpu_i	The required cpu utilization rate of $task_i$
$data_i$	The data size of $task_i$
$coreNum_j$	The number of cores in the jth VM

To provide personal services and maximize resource utilization, the data center usually offers various types of VM instances with varied resource configurations. Hence, the VM instance can be described as $vm_j = (mips_j, ram_j, price_j)$ $(0 \leq j \leq J)$, where j indicates the id of the VM instance.

Generally, the tasks are submitted by numerous end-users simultaneously. On the other hand, the cloud services providers have no prior knowledge of the incoming workloads in advance, e.g., the amount and type of submitted tasks. Hence, in our model, a task is identified as $task_i = (mi_i, cpu_i, data_i)$ $(0 \leq i \leq I)$, where i is the task id.

The task scheduler is responsible for assigning user tasks to suitable VM instances. When a task is allocated to a specific VM instance, it firstly enters the corresponding waiting queue in a first-come-first-sever (FCFS) manner. The response time of task i deployed on VM j (i.e., RT_{ij}) is defined as the total amount of time that task i stays in VM j. In detail, it can be divided into the duration of task i will spend in the waiting queue (i.e., Q_j^{queue}) and the execution time in the CPU (P_{ij}^{exe}). Considering the transportation of $data_i$ consumes some time, the RT_{ij} can be defined as:

$$RT_{ij} = P_{ij}^{exe} + max\left(Q_j^{queue}, \frac{data_i}{speed}\right) \tag{1}$$

Based on the above assumption, the execution time P_{ij}^{exe} is defined as $P_{ij}^{exe} = \frac{mi_i}{mips_j \times cpu_i}$, and the waiting time Q_j^{queue} is $Q_j^{queue} = \sum_{k=1}^{C} P_{kj}^{exe}$, where C means the number of tasks (arrival earlier than task i) that are waiting in the queue of vm_j. Furthermore, the average response time of total $I(t)$ tasks at time t is:

$$AT(t) = \sum_{i=1}^{I(t)} \frac{RT_{ij}}{mi_i} \tag{2}$$

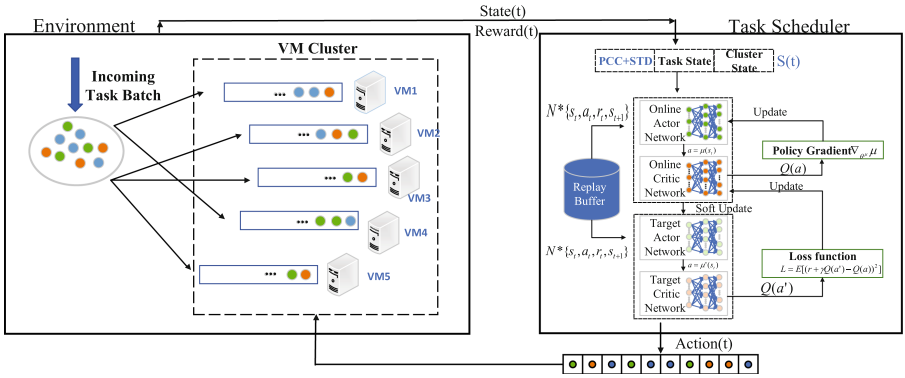


Fig. 2. The structure of DRL-based Task scheduler

Given the price of rented VM_j, the cost of executing $task_i$ is evaluated as $EC_i = P_{ij}^{exe} \times price_j$. Thus, the total cost of executing $I(t)$ tasks at time t is:

$$EC(t) = \sum_{i=1}^{I(t)} EC_i \tag{3}$$

Based on (2) and (3), the optimal target of the task scheduler can be defined as minimizing the average response time and total cost of all tasks during the whole running time K, it can be expressed as:

$$minimize \quad \beta \times \sum_{t=1}^{K} AT(t) + (1-\beta) \times \sum_{t=1}^{K} EC(t) \tag{4}$$

4 Performance and Cost-Aware Task Scheduler

In this section, we propose a DDPG-based method for performance and cost-aware task scheduling in the high dynamic cloud environment. The objectives of our method are to optimize the average response time and total cost when executing large-scale tasks in VM cluster. Considering the time-varying characteristics of workload, the dynamic tasking scheduling problem is constructed as a Markov decision process (MDP), then the DDPG algorithm is employed to obtain the optimal task scheduling strategy. Specifically, the correlation-aware state representation and dual rewards design are introduced.

4.1 MDP Model

In the cloud environment, the workloads are unpredictable and highly dynamic. Thus, it is impossible for us to adapt traditional scheduling methods to such flexible scenarios. In this scenario, we model the task scheduling in the cloud environment as a Markov Decision Process (MDP). Formally, we define the tuple of three elements (S, A, R) of MDP in the cloud task scheduler as follows:

- **State space** S: In the scheduling algorithm, a state $s \in S$ is defined by the correlation of the batch tasks s^{cor}, the current status of submitted tasks s^{Task} and rented VMs s^{VM} , i.e., $s = s^{cor} \bigcup s^{Task} \bigcup s^{VM}$.
- **Action Space** A: An action $a = \left\{vm^1, vm^2, \ldots, vm^J\right\} \in A$ is to assign a batch of tasks to rented J VMs based on the current state s.
- **Reward** R: The reward is used to guide the task scheduler to make the optimal assignment solution (i.e., action) under current states, based on the objective of the proposed task scheduler framework. In our model, the reward of assigning a batch of I tasks to a cluster of J VMs is: $reword_{IJ} = reward_{IJ}^{prior} + reward_{IJ}^{posterior}$.

The agent is to assign different tasks that users submit to appropriate VMs. The agent observes the state of tasks and VMs in the cluster and takes action according to trained police π. After that, it receives a reward immediately from the environment. In detail, the proposed DDPG-based task scheduling framework consists of two parts, i.e., online net and target net, to train and test the task assignment policy based on the online environment and offline historical data. For each part, it includes an Actor net and Critic net. The structure of our proposed task scheduler is illustrated in Fig. 2.

4.2 Correlation-Aware State Representation

State representation plays an important role in the DRL algorithms because it describes the current information of the cloud environment. In [5], it revealed the effectiveness of considering correlations while scheduling the tasks. Therefore, we propose a correlation-aware state representation method to help the agent better perceive the workload. In detail, PCC is an effective matrix to evaluate the correlation of any two task, and has been widely applied in various fields. Hence, in this paper, we adopt PCC to measure demand correlation. Considering the demand for three types of resources $(mi, cpu, data)$ in batch tasks, the PCC of the batch tasks can be calculated by:

$$\rho(mi, data) = \frac{cov(mi, data)}{\sigma(mi) \times \sigma(data)}, \tag{5}$$

$$\rho(mi, cpu) = \frac{cov(mi, cpu^{util})}{\sigma(mi) \times \sigma(cpu)}, \tag{6}$$

$$\rho(cpu, data) = \frac{cov(cpu, data)}{\sigma(cpu) \times \sigma(data)}, \tag{7}$$

Here, $cov(\cdot)$ and $\sigma(\cdot)$ represent the covariance and standard deviation, respectively. In addition, to evaluate the distributions of the resource demands between different tasks, we also employ the standard deviation(STD) of $std(mips)$, $std(data)$,and $std(cpu)$, which can be calculated by:

$$std(mips) = \sqrt{\frac{\sum_{i=1}^{I}(mips_i - \overline{mips})^2}{I}} \tag{8}$$

$$std(cpu) = \sqrt{\frac{\sum_{i=1}^{I}(cpu_i - \overline{cpu})^2}{I}} \tag{9}$$

$$std(data) = \sqrt{\frac{\sum_{i=1}^{I}(data_i - \overline{data})^2}{I}} \tag{10}$$

We applied the PCC and STD to the state space to let the agent make better decisions according to the feature of the batch tasks. Therefore the correlation part of the state s is:

$$s^{cor} = [\rho(mi, data), \rho(mi, cpu), \rho(cpu, data), std(mi), std(data), std(cpu)] \tag{11}$$

The second part of the vector is the batch's specification. Suppose there are in total I tasks in a batch, the batch can be described as:

$$s^{task} = [task^1_{mi}, task^1_{cpu}, task^1_{data}, ..., task^I_{mi}, task^I_{cpu}, task^I_{data}] \tag{12}$$

The third part of the vector represents the state of the VMs. Suppose there are in total J VMs in the cluster. Note that each vm_j may have a Task Queue waiting to be executed, therefore it has a feature that represents the time that a new task will wait in line Q_j^{queue} , so the cluster can be described as:

$$s^{VM} = [vm^1_{mips}, vm^1_{price}, vm^1_Q, ..., vm^J_{mips}, vm^J_{price}, vm^J_Q] \tag{13}$$

These three parts make up the state space vector. After each action, the cluster state will be updated and the next batch will come and make up the new space state.

4.3 Dual Rewards

To better guide the agent in learning an optimal task allocation scheme, we define a dual reward model for the proposed DDPG method, which includes a prior and a posterior rewards. Specifically, the prior reward means the agent can know the reward of a specific action before the VMs execute the tasks. We use $reward^{prior}_{IJ}$ to represent the total prior reward of a batch task:

$$reword^{prior}_{ij} = \begin{cases} reward_{pri}/I, & \text{if } Q_j^{queue} \text{ is not empty} \\ 0, & \text{if } Q_j^{queue} \text{ is empty} \end{cases} \tag{14}$$

$$reward^{prior}_{IJ} = \sum_{i=1}^{I} reword^{prior}_{ij} \tag{15}$$

Here, $reward_{pri}$ is a constant that is used to control the maximum of prior reward. The posterior reward means the agent can only know the reward after the VMs execute the tasks. Our posterior reward has two elements, one is the total cost of the tasks EC_{IJ}, the other is the average response time AT. For

a specific task in the batch, the execution of it will have a certain cost and a certain response time ratio, which are EC_{ij} and AT_{ij} respectively.

To make the training process stable, we normalized the EC and AT in a batch. We define the maximum and minimum AT in a batch as AT_{max} and AT_{min}, and the maximum EC in a batch as EC_{max}. Therefore, the normalized AT and *cost* are:

$$AT^*_{ij} = \frac{AT_{ij} - AT_{min}}{AT_{max} - AT_{min}} \tag{16}$$

$$EC^*_{ij} = \frac{EC_ij}{EC_{max}} \tag{17}$$

Hence, the posterior reward of this batch is:

$$reward^{posterior}_{IJ} = \beta \times \sum_{i=1}^{I} AT^*_{ij} + (1-\beta) \sum_{i=1}^{I} EC^*_{ij} \tag{18}$$

Note that $\beta \in [0, 1]$. β can be used to adjust the agent's optimization objectives. For example, if $\beta = 1$, then an agent is trained to reduce the average response time.

The two rewards are applied simultaneously to train the agent. We can adjust the value of $reward_{pri}$ to control the size relationship between two rewards. The goal of the DDPG agent is to minimize the rewards.

4.4 Algorithm Training

In the following, we introduce our proposed DDPG-based task scheduling algorithm (see Algorithm 1). Offline training can not only make the critic network evaluate the actions more accurately, but also let the action network generate higher scored actions. To achieve this, we applied experience replayed strategy and target network [16].

Experience replay's main purpose is to solve the problem of correlation and non-stationary distribution of empirical data. We introduced a fixed-size memory replay buffer $\mathfrak{R}$. At each time step, we will store the latest (a_t, s_t, r_t, s_{t+1}) sets to the replay buffer $\mathfrak{R}$, and randomly sample a mini-batch from the buffer to train the agent. Because the DDPG algorithm is an off-policy algorithm, the replay buffer can be relatively large which allows the algorithm to learn across a set of uncorrelated transitions. Therefore, the parameters of the actor network are updated by:

$$\begin{aligned} \nabla_{\theta^\mu} \mu &\approx E_{\mu'}[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t|\theta^\mu)}] \\ &= E_{\mu'}[\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_t}] \end{aligned} \tag{19}$$

Directly implementing Q-learning with a neural network is proved unstable in many situations. Introducing a target network can significantly reduce the oscillations of the neural network's parameters caused training process. The target network is a copy of the online network (actor and critic network), but it is

Algorithm 1. DDPG-based task scheduling Algorithm

Randomly initialize online critic network $Q(s, a|\theta^Q)$ and online actor network $\mu(s|\theta^\mu)$ with parameters θ^Q and θ^μ
Initialize target network Q' and μ' with parameters $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize memory replay buffer $\mathfrak{R}$
Initialize exploration probability ϵ and exploration warm up steps n
for each batch of I tasks arrive at time $t = 1, ..., T$ **do**
  if $t > n$ **then**
    Sample a mini-batch of $\mathcal{N}$ transitions (s_i, a_i, s_{i+1}) from $\mathfrak{R}$, which all the selected tasks have been completed
    Calculate $reward_{IJ}^i$ according to responseTime and exT
    Calculate r_i according to $reward_{IJ}^i$
    Set $y_i = r_i + \lambda Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update the 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:
    $\nabla_{\theta^\mu}\mu|s_i \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s=s_i}$
    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 $1 - \epsilon$ generate an action $a_t = \mu(s_t|\theta^\mu)$ and ϵ generate a random action a_t
  else
    Generate a random action a_t
  end if
  Complete task scheduling according to action a_t and observe reward
  Store transition (s_t, a_t) in $\mathfrak{R}$
  Store completed task's RT and P^{exe} in $\mathfrak{R}$
end for

updated slower instead of copying the weights directly, which ensures the neural network higher stability. At each time step, the parameters of the target network are updated by:

$$\theta' = \tau\theta + (1-\tau)\theta' \tag{20}$$

5 Performance Evaluation

5.1 Experimental Settings

Cluster Resources. We consider that there are 20 VMs deployed in the public cloud and provide services to the end-users. Meanwhile, we set four types of VM instances with various pricing models in the cluster. The details of cluster resources and price are shown in Table 2. Note that, the pricing model of the VM instances is identified with the Enterprise level Computation type(c7) (in China) provided by Alibaba Cloud. Following [17,18]. We also adopt Cloudsim Plus to build the cloud environment.

Table 2. Cluster resource details

Instance Type	CPU cores	Memory(GB)	Quantity	Price
m1	16	32	5	\$0.3624/h
m2	12	24	5	\$0.2739/h
m3	4	8	5	\$0.0972/h
m4	2	4	5	\$0.0530/h

Workloads. *Alibaba-Cluster-trace-v2018* is used to test the performance of the task scheduler, which contains 4000 VMs under workload of 8 d. To simplify the problem, we use the data of the second day. To make the model stable, there are up to 50 tasks can be submitted by the end-users in one second, which will not miss the feature of the workload according to our observation.

Parameter Settings. In our DDPG-based task scheduling algorithm, we employ four deep neural networks, which are Actor_online, Critic_online, Actor_target, Critic_target. Each one has four fully connected layers. Both two online networks are updated with each round of training, whereas the two target networks are updated by (20) with $\tau = 0.01$. We set the capacity of memory replay buffer $\mathfrak{R} = 10000$, the size of mini-batch $\mathcal{N} = 16$. We apply Adam optimizer to optimize the network and the learning rate for Actor_online and Critic_online networks are 0.0006 and 0.001 respectively. To store some memory before the training, the network begins to train after 400 steps. 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.

Baseline Schedulers. We compare our DDPG-based task scheduler with four baseline approaches. There are 1)Random, randomly selects a VM for each job; 2) Round-Robin(rr), assigns tasks to each VM in turn; 3) Earliest, assigns a task to the first idle VM according to the arrival time; 4) DQN, the newest DRL-based task scheduling method, and the design of the DQN method is similar to [11].

Note that, our proposed DDPG-based scheduler and all the baselines make dynamic decisions from the current state of the cluster and workload, and do not have prior knowledge of the whole workload. We performed 10 repeated experiments on each algorithm and recorded the average results.

Evaluation Metrics. In this paper, we apply three indicators to evaluate the performance of different methods in terms of satisfying different stakeholders profiles, which are response time ratio, total cost and the standard deviation of CPU utilization. Among them, response time ratio $ResTR = \frac{\sum_{m=1}^{M} \frac{mi_m}{RT_{mj}}}{M}$ describes the average response time of executing M tasks in total. Total costs represents the monetary cost of the cloud services provider during the whole execution time K. According to (3), the total costs can be defined as $Cost = \sum_{t=1}^{K} EC(t)$.

CPU utilization standard deviation among the cluster, which means the standard deviation of average CPU utilization of each VM instance in the cluster. The CPU utilization standard deviation can represent the level of load balance among the cluster. For the data center, a lower one means better resource utilization:

$$\Delta cpu = \sqrt{\frac{\sum_{j=1}^{J}(AvgCpu_j - \overline{AvgCpu_j})}{J}} \tag{21}$$

Here, $AvgCpu_j$ represents the average cpu utilization of VM_j in the time period of K time steps.

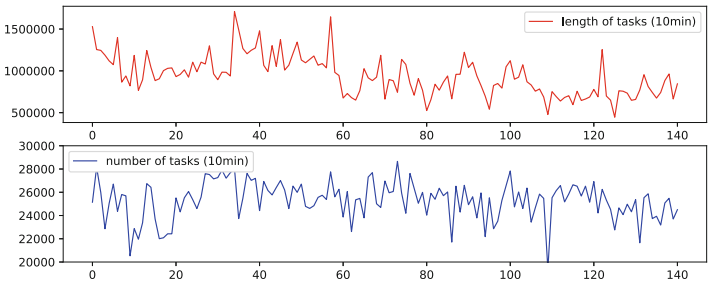
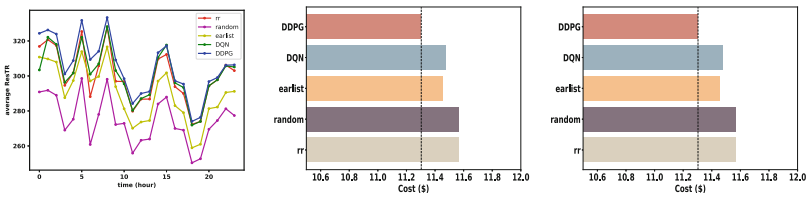


Fig. 3. The changes of Alibaba-Cluster-trace-v2018 workload on the second day.



(a) Response Time Ratio (b) Total Costs (c) CPU Utilization

Fig. 4. Performance comparison on real-world workload.

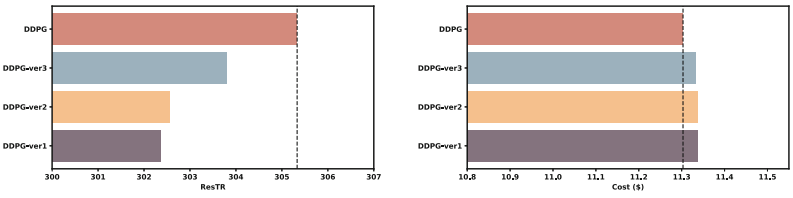
5.2 Performance on Real-World Workload

First, we evaluate the performance of DDPG in the face of real-workload with the characteristic of high dynamical. Figure 3 shows the average total task number and average total task length of Alibaba-Cluster-trace-v2018 every 10 min.

Figure 4 shows the experiment result of the comparison of these methods. We can summarize the following three conclusions: 1). Our proposed DDPG-based method has the best performance on all of those three metrics, this is because the reinforcement learning methods and our design of PCC and STD in the state space allows it to detect the feature of the workload efficiently. The dual rewards allow it to learn to adapt to proper strategy to realize global optimization. 2).

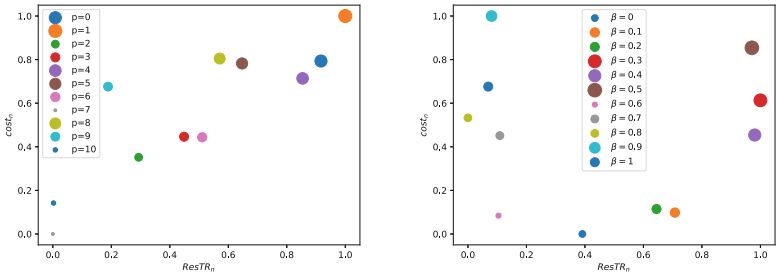
The RL methods (DDPG, DQN) have better performance than traditional methods, because they can learn from their interaction with the environment, whereas traditional methods are incapable of changing their strategies according to the environment. 3). Our proposed DDPG-based method has a better performance than the DQN method, especially on the metric of cost the average CPU utilization standard deviation. This is because the DDPG method can schedule the whole batch of tasks simultaneously, and DQN can only do a one-by-one schedule. This ability of DDPG is strengthened by the design of PCC and STD in the state space.

Ablation Study. To prove the effectiveness of each component in our refinement of the DDPG method, we gradually eliminate the corresponding model components by defining the following versions: 1). DDPG-ver1: it does not have PCC and STD in its state space, and it only has one posterior reward for the agent, which is a typical model for most DRL methods. 2). DDPG-ver2: In this model, we added PCC and STD in its state space. 3). DDPG-ver3: It has dual rewards as its reward function. The results are shown in Fig. 5. As expected, our proposed method outperforms all the other versions of the DDPG method. In detail, our refined versions DDPG-ver2 and DDPG-ver3 already achieved quite good performances compared with the original DDPG method. This is because the PCC and STD in the state space allow the DDPG-ver2 agent to better detect



(a) ResTR (b) Total Costs

Fig. 5. Comparison of DDPG methods in different versions



(a) Comparison of different p (b) Comparison of different β

Fig. 6. Influence of changing $reward_{pri}$ and β. The size of the points represents the sum of the two value.

the intrinsic correlation of the workload efficiently, and the dual rewards allow the DDPG-ver3 agent to adapt to a more balanced global optimization strategy. Lastly, our proposed method combines the advantages of DDPG-ver2 and DDPG-ver3 can reach the best performance among these versions.

As the optimization priority β and the $reward_{pri}$ of prior reward are pivotal for the agent to learn a proper strategy, we investigated the impact of changing the value of them. To better evaluate these two objectives, we normalized both of them. Among all of these algorithms, we can define the $ResTR_{max}$, $ResTR_{min}$ and $cost_{max}$, $cost_{min}$. Therefore, each one's $ResTR^* = \frac{ResTR - ResTR_{min}}{ResTR_{max} - ResTR_{min}}$, and $cost^* = 1 - \frac{cost - cost_{min}}{cost_{max} - cost_{min}}$. It is clear that a greater $ResTR^*$ and $cost^*$ means a better performance.

Effect of changing $reward_{pri}$. We use p in the figure to denote $reward_{pri}$. We changed $reward_{pri}$ in a range of $[0, 10]$. Figure 6a shows the experiment results. We have the following observations: 1). $p = 0$ does not have the best overall performance because it completely ignores the prior reward. 2). As the growing of p, the performance of the model declines. This is because the excessive large value of p will let the agent adapt to the strategy more like the earliest method which can not reach a global optimization strategy. 3). $p = 1 (reward_{pri} = 1)$ has the best performance on both $ResTR$ and $cost$, which indicates that this is the most proper value of the prior reward.

Effect of changing β. To evaluate which value of β can help the algorithm reach the best overall performance, we changed β in a range of $[0, 1]$. Figure 6(b) shows the experiment results. We have the following observations: 1). $\beta = 0$ and $\beta = 1$ do not have good performances on either metric, because these two are strongly correlated, and ignoring any of them will prevent the algorithm from finding the most proper strategy. 2). When $\beta = 0.5$, the overall performance is the best because it is in the upper right corner. This is because it can better balance the two optimization objectives and learn to adapt to the most appropriate strategies. 3). $\beta = 0.3$ has the best performance on $ResTR$ and $\beta = 0.9$ has the best performance on $cost$. According to different scenarios, these are also applicable choices of β.

6 Conclusion

Efficient task scheduling in the cloud environment is always an important and challenging problem because of its high dynamic and unpredictable workload and complex inherent VM characteristics. Traditional methods and heuristic-based approaches only focus on some specific scenarios with particular objectives. In this paper, we introduce an RL model for the problem of multiple-objective optimization-based task scheduling in the cloud datacenter. In addition, we apply this model to our DDPG-based algorithm. We have designed correlation-aware state representation and advanced reward signals, which help the DDPG agent to learn the task schedule performance and the total cost of VM cluster. The agents can learn to optimize multiple objectives under high dynamic workloads without previous knowledge of the VM cluster and the workload, but only from

its interaction with the environment and the rewards. Extensive experimental results have shown that our proposed method overperforms the baseline methods on response time and total cost when facing high dynamical workloads.

Acknowledgements. This work is partly supported by the key cooperation project of chongqing municipal education commission (HZ2021017,HZ2021018), in part by the "Fertilizer Robot" project of Chongqing Committee on Agriculture and Rural Affairs, in part by the Chongqing Research Program of Technology Innovation and Application under grants cstc2019jscx-zdztzxX0019, in part by West Light Foundation of The Chinese Academy of Sciences.

References

1. Arunarani, A., Manjula, D., Sugumaran, V.: Task scheduling techniques in cloud computing: a literature survey. Future Gener. Comput. Syst. **91**, 407–415 (2019)
2. Zhu, Q.-H., Tang, H., Huang, J.-J., Hou, Y.: Task scheduling for multi-cloud computing subject to security and reliability constraints. IEEE/CAA J. Automat. Sinica **8**(4), 848–865 (2021)
3. Houssein, E.H., Gad, A.G., Wazery, Y.M., Suganthan, P.N.: Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends. Swarm Evol. Comput. **62**, 100841 (2021)
4. Tawfeek, M.A., El-Sisi, A.B., Keshk, A.E., Torkey, F.A.: Cloud task scheduling based on ant colony optimization. In: 2013 8th International Conference on Computer Engineering & Systems (ICCES), pp. 64–69 (2013)
5. Luo, C., et al.: Correlation-aware heuristic search for intelligent virtual machine provisioning in cloud systems. In: Proceedings of the AAAI Conference on Artificial Intelligence **35**, 12363–12372 (2021)
6. Shu, W., Cai, K., Xiong, N.N.: Research on strong agile response task scheduling optimization enhancement with optimal resource usage in green cloud computing. Future Gener. Comput. Syst. **124**, 12–20 (2021)
7. Bezdan, T., Zivkovic, M., Bacanin, N., Strumberger, I., Tuba, E., Tuba, M.: Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm. J. Intell. Fuzzy Syst. **42**(1), 411–423 (2022)
8. Gill, S.S., Chana, I.: A survey on resource scheduling in cloud computing: issues and challenges. J. Grid Comput. **14**, 06 (2016)
9. Mathew, T., Sekaran, K.C., Jose, J.: Study and analysis of various task scheduling algorithms in the cloud computing environment. In: 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 658–664 (2014)
10. Liu, X., Tong, W., Zhi, X., ZhiRen, F., WenZhao, L.: Performance analysis of cloud computing services considering resources sharing among virtual machines. J. Supercomput. **69**(1), 357–374 (2014)
11. Islam, M.T., Karunasekera, S., Buyya, R.: Performance and cost-efficient spark job scheduling based on deep reinforcement learning in cloud computing environments. IEEE Trans. Parallel Distrib. Syst. **33**(7), 1695–1710 (2021)
12. Ran, L., Shi, X., Shang, M.: SLAs-Aware online task scheduling based on deep reinforcement learning method in cloud environment. In: 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 1518–1525, IEEE (2019)

13. Wei, Y., Pan, L., Liu, S., Wu, L., Meng, X.: DRL-scheduling: an intelligent QoS-aware job scheduling framework for applications in clouds. IEEE Access **6**, 55112–55125 (2018)
14. Mao, H., Alizadeh, M., Menache, I., Kandula, S.: Resource management with deep reinforcement learning. In: Proceedings of the 15th ACM workshop on hot topics in networks, pp. 50–56 (2016)
15. Rjoub, G., Bentahar, J., Abdel Wahab, O., Saleh Bataineh, A.: Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems. Concurrency and Computation: Practice and Experience, vol. 33, no. 23, p. e5919 (2021)
16. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. In: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings (Y. Bengio and Y. LeCun, eds.) (2016)
17. Abreu, D.P., et al.: A rank scheduling mechanism for fog environments. In: 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), pp. 363–369, IEEE (2018)
18. Silva Filho, M.C., Oliveira, R.L., Monteiro, C.C., Inácio, P.R., Freire, M.M.: 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), pp. 400–406, IEEE (2017)