Cloud Task Scheduling Based on Proximal Policy Optimization Algorithm for Lowering Energy Consumption of Data Center

Yongquan Yang[1], Cuihua He[1], Bo Yin[1], Zhiqiang Wei[1], and Bowei Hong[1*]
[1] Department of Computer Science and technology (Ocean University of China)
QingDao, China
[e-mail: yangyq@ouc.edu.cn, hecuihua@stu.ouc.edu.cn, ybfirst@ouc.edu.cn, weizhiqiang@ouc.edu.cn, hongbowei@ouc.edu.cn]
[*]Corresponding author: Bowei Hong

Received January 5, 2022; revised April 21, 2022; accepted May 30, 2022; published June 30, 2022

Abstract

As a part of cloud computing technology, algorithms for cloud task scheduling place an important influence on the area of cloud computing in data centers. In our earlier work, we proposed DeepEnergyJS, which was designed based on the original version of the policy gradient and reinforcement learning algorithm. We verified its effectiveness through simulation experiments. In this study, we used the Proximal Policy Optimization (PPO) algorithm to update DeepEnergyJS to DeepEnergyJSV2.0. First, we verify the convergence of the PPO algorithm on the dataset of Alibaba Cluster Data V2018. Then we contrast it with reinforcement learning algorithm in terms of convergence rate, converged value, and stability. The results indicate that PPO performed better in training and test data sets compared with reinforcement learning algorithm, as well as other general heuristic algorithms, such as First Fit, Random, and Tetris. DeepEnergyJSV2.0 achieves better energy efficiency than DeepEnergyJS by about 7.814%.

Keywords: cloud computing, cloud task scheduling, deep reinforcement learning, energy consumption, proximal policy optimization

1. Introduction

Cloud computing has become a trend in high-performance computing and is characterized by its large-scale, heterogeneous computing resources, and flexible computational architecture. In recent years, active work has appeared in cloud computing areas such as scheduling, placement, energy management, privacy and policy, security [1–4], and more. Algorithms for task scheduling have attracted extensive attention as a part of cloud computing technology. Task scheduling refers to mapping several tasks to computational resources. With the development of cloud service providers (CSPs), huge energy consumption and carbon dioxide emissions have become a serious challenge. Developing an energy-saving task scheduling strategy has practical importance.

Q-learning [5] is a classic algorithm for reinforcement learning. Deep neural networks have shown strong fitting ability in many fields, and reinforcement learning has an excellent ability for decision-making. Deep reinforcement learning (DRL) [6] combines deep learning (DL) [7] and reinforcement learning (RL) [8] algorithms, which enable it to solve complex control problems with a large state/action space. Deep learning captures the features of dynamic scenes from the current environment, and RL learns the best strategy guided by the corresponding reward obtained from interactions with the environment.

In our previous work, we proposed DeepEnergyJS [9], a cloud task scheduling framework based on deep reinforcement learning algorithms [10]. DeepEnergyJS obtained acceptable experimental results but still can be improved.

In this work we will upgrade our framework to DeepEnergyJSV2.0 by applying the Proximal Policy Optimization (PPO) algorithm as an alternative to the original policy gradient algorithm in DeepEnergyJS to improve the efficiency of reducing energy consumption.

To validate DeepEnergyJSV2.0, we updated the data set from Alibaba Cluster Data V2017[11] to Alibaba Cluster Data V2018[12], which consists of hybrid-type tasks that contain both independent tasks and tasks with inner task dependencies. The original data set does not contain tasks with dependencies, which makes it not very convincing to validate DeepEnergyJSV2.0. With Alibaba Cluster Data V2018, we can verify DeepEnergyJS on hybrid-type tasks, and our simulation experiment is more in line with real-world cases.

2. Related Works

Many approaches have been proposed to reduce the energy consumption of data centers through task scheduling. A. Francis Saviour Devaraj et al. [13] proposed an algorithm based on best-worst (BWM) and the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) methodology, in which they presented a modified particle swarm optimization algorithm that achieves great results in balancing energy efficiency. Peng and Wen et al. [14] proposed an optimal task workflow scheduling scheme based on the dynamic voltage and frequency scaling technique and the whale optimization algorithm, which can achieve a balance between performance and energy consumption. However, these offline algorithms have difficulties dealing with online dynamic tasks and large inputs. The dynamics and complexity of an enterprise strategy environment make scheduling more challenging. Ding et al. [15] proposed a Q-learning-based task scheduling framework for energy-efficient cloud computing (QEEC) to minimize task response time and maximize each server's CPU utilization simultaneously. Seth et al. [16] discussed the dynamic heterogeneous shortest job first (DHSJF) model, which considers both dynamic heterogeneities of workload and resources.

The task scheduling problem is an NP-hard problem, and various meta-heuristic algorithms can provide a feasible solution under certain conditions. The current research on cloud computing task scheduling focuses on independent tasks with traditional heuristic algorithms. Deep reinforcement learning (DRL) has attracted attention in recent years and has an outstanding ability to solve complicated control problems with high-dimensional state spaces and low-dimensional action spaces. Research work on how to apply DRL to obtain an efficient task scheduling strategy and make full use of system resources is missing.

In the study of dynamic task scheduling, the widely used methods are either heuristic algorithms or the DRL algorithm based on deep Q-networks. The problem of how to apply the DRL algorithm to cloud computing task scheduling strategy needs to be studied.

Compared with DQN, which adapts the ε-greedy strategy, the policy gradient can represent a random strategy and is free from the adjustment of the ε parameter. Based on this, DeepEnergyJS is the first system to present a policy-gradient-based task scheduling method to minimize energy consumption and improve the energy efficiency in a cloud computing system, which has also been proven to be effective for independent tasks and as well as tasks with dependencies. Our previous studies have exclusively focused on hybrid-type tasks that contain both independent tasks and tasks with inner task dependencies. In this paper, we prove that DeepEnergyJS is also applicable to hybrid-type tasks and adopt another gradient-based algorithm called proximal policy optimization (PPO) [17] to update DeepEnergyJS to DeepEnergyJSV2. 0 for better performance.

3. Theory

3.1 Proximal Policy Optimization Algorithm

The policy gradient method is a type of reinforcement learning algorithm, the original version of the policy gradient algorithm, sample data based on the Monte Carlo method, and the variance of the estimated gradient will be higher. REINFORCE is an online learning gradient descent algorithm that minimizes the cost function, which means that the agent that interacts with the environment and the agent that updates its model parameters using environmental feedback is the same. In the algorithm training phase, the agent has to sample a batch of samples under policy π and update the parameters of the same policy network; for the next iterative learning, the agent needs to interact with the environment again to collect the new data, that is, interacting with the environment while updating the parameters of the policy network. There are numerous problems with this online learning approach.

1. It consumes a significant amount of time for the agent to resample new data for iterative parameter updating.
2. Previously collected data could not be reused, which in turn led to low data utilization.

The PPO algorithm is another policy gradient algorithm that is suitable for continuous control problems [18], and it is simpler in its mathematical implementation compared to other policy gradient-based method (PGM)-based RL algorithms [19]. PPO is an offline learning method, and the strategy it adopts to interact with the environment and the strategy to be learned differ. The main idea of the PPO algorithm is to transfer online learning to offline learning based on importance sampling and adopt two networks to improve the network convergence rate. One policy network π' is used to process environment interactions and data gathering, and the other network tweaks its parameters by observing the effective interaction between π' and the environment. Different networks indicate different distribution functions, and the importance sampling mechanisms introduced to the sample from the original distribution. In the PPO algorithm, the original distribution is π'. When it is difficult to sample from the original distribution p(x), it can sample from another distribution q(x), and a weight (p(x))/(q(x)) can be multiplied to correct the difference between the two distributions. The derivation is shown in (1) to (3).

$$E_{x\sim p}[f(x)] = \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx \tag{1}$$

$$\int f(x)\frac{p(x)}{q(x)}q(x)dx = E_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right] \tag{2}$$

$$E_{x\sim p}[f(x)] \approx E_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right] = \frac{1}{N}\sum_{i=1}^{N} f(x_i)\frac{p(x_i)}{q(x_i)} \tag{3}$$

By importance sampling, the adverse effect of large deviation under the original distribution is improved, and the weighting factor acts as a regulator. To make importance sampling to be effective, the new and old distributions cannot differ significantly. Therefore, in the actual application of the PPO algorithm, a constraint is added to limit the difference between the two distributions.

3.2 The Derivation Process of Policy Gradient in PPO Algorithm

Policy π' provides collected data for policy π's parameter updating, which is the main idea behind the PPO algorithm. The derivation of the objective function $J^{\theta'}(\theta)$ is given by (4).

$$\begin{aligned} J^{\theta'}(\theta) &= E_{(s_t,a_t)\sim\pi_\theta}[A^{\theta}(s_t,a_t)\nabla\log\pi_\theta(a_t|s_t)] \\ &= E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{\pi_\theta(s_t,a_t)}{\pi_{\theta'}(s_t,a_t)}A^{\theta'}(s_t,a_t)\nabla\log\pi_\theta(a_t|s_t)\right] \\ &= E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}\frac{\pi_\theta(s_t)}{\pi_{\theta'}(s_t)}A^{\theta'}(s_t,a_t)\nabla\log\pi_\theta(a_t|s_t)\right] \\ &\approx E_{(s_t,a_t)\sim\pi_{\theta'}}[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}A^{\theta'}(s_t,a_t)\nabla\log\pi_\theta(a_t|s_t)] \end{aligned} \tag{4}$$

In formula (4), $\frac{\pi_\theta(s_t)}{\pi_{\theta'}(s_t)}$ is generally ignored; the pro rata coefficient $r_\theta = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}$ denotes the difference between distribution π' and π. To narrow the difference, the theory TRPO [19] suggests using an adaptive KL penalty coefficient, as expressed in (5).

$$J_{ppo}^{\theta'}(\theta) = J^{\theta'}(\theta) - \beta KL(\theta,\theta') \approx \sum_{(s_t,a_t)} r_\theta A^{\theta'}(s_t,a_t) - \beta KL(\theta,\theta') \tag{5}$$

The KL divergence is used to quantify the two different distributions. If the two distributions are identical, the value of the total KL divergence is zero, whereas a smaller value indicates a higher similarity between the two distributions conversely, a larger discrepancy. The main objective is represented by (6).

$$J_{ppo2}^{\theta'}(\theta) \approx \sum_{(s_t,a_t)} \min\left(r_\theta A^{\theta'}(s_t,a_t), \mathrm{clip}(r_\theta, 1-\epsilon, 1+\epsilon)A^{\theta'}(s_t,a_t)\right) \tag{6}$$

In formula (6), the function $\mathrm{clip}(r_\theta, 1-\epsilon, 1+\epsilon)$ is displayed in **Fig. 1**. The function $\min\left(r_\theta A^{\theta'}(s_t, a_t), \mathrm{clip}(r_\theta, 1-\epsilon, 1+\epsilon)A^{\theta'}(s_t, a_t)\right)$ is displayed in **Fig. 2**.

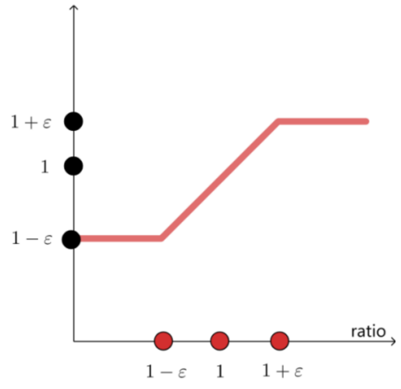

Fig. 1. Function $\mathrm{clip}(r_\theta, 1-\epsilon, 1+\epsilon)$

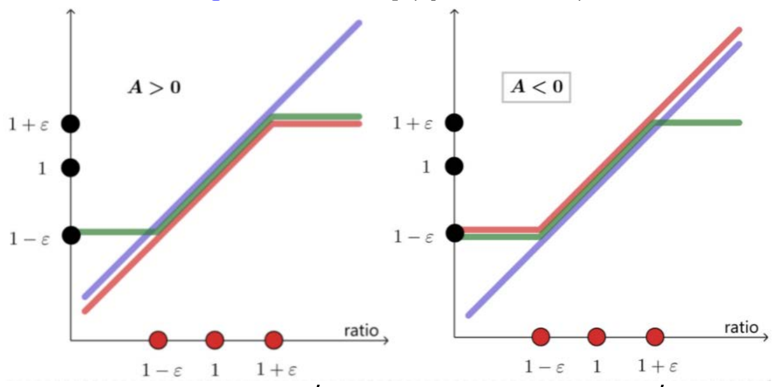

Fig. 2. Function $\min\left(r_\theta A^{\theta'}(s_t, a_t), \mathrm{clip}(r_\theta, 1-\epsilon, 1+\epsilon)A^{\theta'}(s_t, a_t)\right)$

The red line in the plot represents the objective function, when $A^{\theta'}(s_t, a_t)$ is greater than 0, the probability of state-action pair (s_t, a_t) selected will increase, which means the value of $\pi_\theta(s_t, a_t)$ will also increase, but the $\pi_\theta(s_t, a_t)/\pi_{\theta'}(s_t, a_t)$ ratio cannot exceed $1+\epsilon$. Likewise, when $A^{\theta'}(s_t, a_t)$ is less than 0, the probability of state-action pair (s_t, a_t) being selected will decrease which result in the decreasing $\pi_\theta(s_t, a_t)$, and the $\pi_\theta(s_t, a_t)/\pi_{\theta'}(s_t, a_t)$ ratio can also not be lower than1 $-\epsilon$. Furthermore, limiting the distance between π and π'.

Subsequent to the above analysis, the calculation of the policy gradient is shown in (7).

$$g_b(\tau) = \sum_{t=0}^{T} \min\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)} \nabla \log \pi_\theta(a_t|s_t) \cdot (R(\tau) - b), \mathrm{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right) \cdot (R(\tau) - b)\right] \quad (7)$$

4. Method

4.1 MDP Model

Task scheduling is an NP-hard problem, and the Markov decision process can provide a framework for modeling complex cloud task scheduling decision processes. As in our previous work, the state space is described by a list

$$S = [\langle m_1, t_1\rangle, \langle m_2, t_2\rangle, \ldots, \langle m_i, t_i\rangle, \ldots, \langle m_{total}, t_{total}\rangle]$$

, where each element in the list is a $\langle m_i, t_i \rangle$ pair, indicating the task-instance t_i can be scheduled to machine m_i. The set of indexes of the state space list $A = [1,2, \ldots, i, \ldots, \text{total}]$ is designed to denote the action space, action i means selecting $\langle m_i, t_i \rangle$ pair to allocate t_i to m_i. The added value of the power of the data center after the current acts multiplies (-1) represents the reward signal. The extracted properties in task t_i and machine m_i were presented in our previous work [9].

4.2 Formulation of Objective Optimization

The major research objective of this work is to propose an energy-minimizing method to lower energy consumption in a data center. The energy consumption for each machine can be accumulated by the instantaneous power. The instantaneous power of time t is determined using (8).

$$\text{power}(u) = P_{\text{idle}} + \left(P_{\text{busy}} - p_{\text{idle}}\right). u^r \tag{8}$$

While P_{idle} represents static power, P_{busy} the maximum power, u is the CPU usage, r is determined by the machine type, and it should be obtained from the best-fit curves.

Assuming that there are M machines, the total energy consumption is calculated using (9).

$$E = \sum_{i=1}^{M} \sum_{t=0}^{T_i} \text{power}(u)_t \tag{9}$$

4.3. The Overall Framework of Method

The overall framework of the task-scheduling method is shown in **Fig. 3**. Because the simulation experiments are initiated at each instance of time, when tasks initiate, the state space is created. For every taskInstance_i arriving, if machine_j satisfies the resource demand of taskInstance_i, the pair $< \text{machine}_j, \text{taskInstance}_i >$ is then incorporated into the list of the state space. It is worth noting that if machine_k also meets the requirements, pair $< \text{machine}_k, \text{taskInstance}_i >$ is also required to be added to the state space list. All of the machine-task instance pairs in the state space are directly input into the neural networks, and the fitness for each pair will be output. Suppose that $< \text{machine}_{\text{action}}, \text{taskInstance}_{\text{action}} >$ has the maximum fitness, $\text{taskInstance}_{\text{action}}$ will be scheduled to $\text{machine}_{\text{action}}$. Based on this, the state space is re-constructed for the remaining unscheduled task instances until the observed state space is empty. It is worth noting that time will not elapse during the period when the state space is not empty. There is more than one method for the neural networks to update the parameters, and in our previous study, REINFORCE was adopted; in this study, we performed the PPO algorithm and verified the factor validity first, and then contrast the two algorithms in terms of convergence rate, converged value, and stability. The flow chart of the experiment is shown in **Fig. 4**.

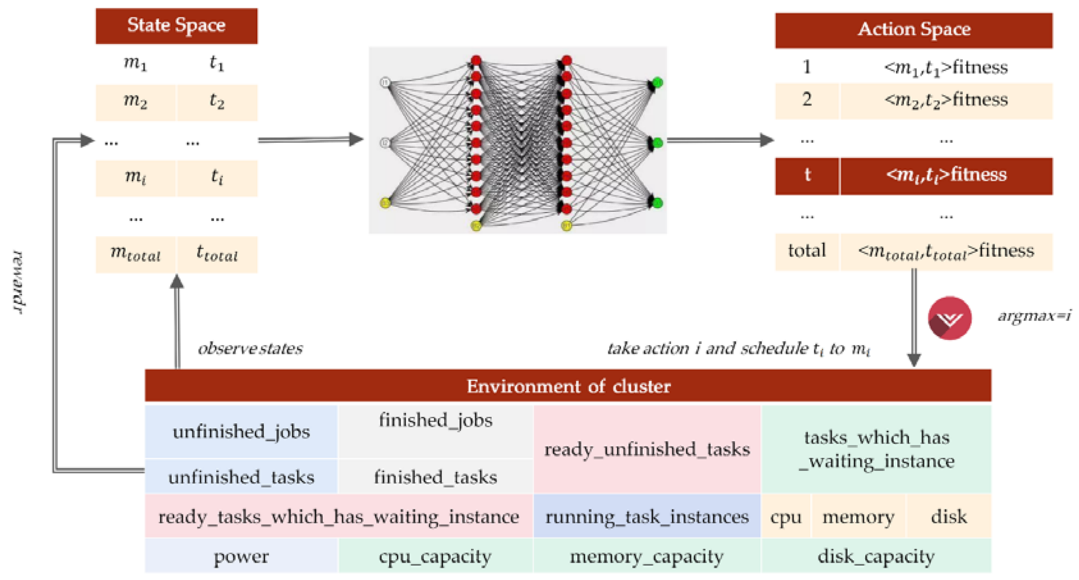

Fig. 3. The Overall Framework of task scheduling method.

Algorithm 1. The procedures of the simulation.

```
Algorithm 1: PPO Algorithm
1: Initialization of all parameters： $\theta = \theta_0$; $\theta' = \theta_0$
2: The simulation is performed through an iterative algorithm until convergence.
3: Create N threads and use policy $\pi_{\theta'}$ to collect N independent trajectories $[\tau_1, \tau_2, \tau_3, \dots, \tau_n]$
4: Compute the baselines$[b_1, b_2, b_3, \dots, b_n]$
5: Calculate the gradient for each trajectory according to equation (9) and obtain
$[g(\tau_1), g(\tau_2), \dots, g(\tau_N)]$
6: Calculate $\nabla_\theta \mathcal{J}(\theta)$
7: Update parameters of $\pi_\theta$： $\theta = \theta + \alpha_{iter} \nabla_\theta \mathcal{J}(\theta)$
8: if times == C: copy parameters $\theta$ to $\theta'$, i.e. $\theta' = \theta$ time = 0
   else: time ++;
```

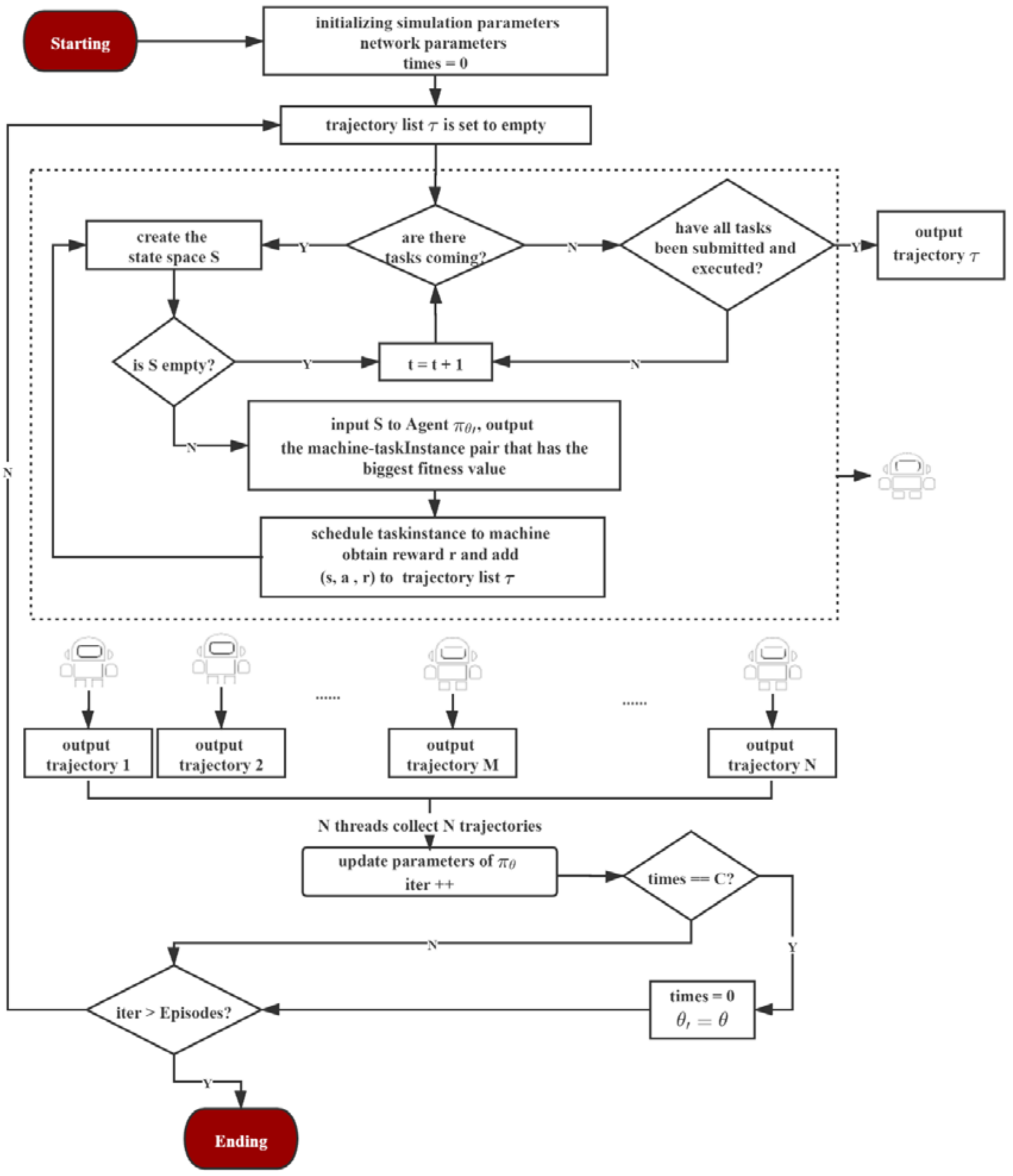
Starting
initializing simulation parameters
network parameters
times = 0
trajectory list τ is set to empty
are there tasks coming?
create the state space S
have all tasks been submitted and executed?
output trajectory τ
is S empty?
t = t + 1
input S to Agent $\pi_{\theta'}$, output the machine-taskInstance pair that has the biggest fitness value
schedule taskinstance to machine obtain reward r and add (s, a , r) to trajectory list τ
output trajectory 1
output trajectory 2
output trajectory M
output trajectory N
N threads collect N trajectories
update parameters of π_θ
iter ++
times == C?
iter > Episodes?
times = 0
$\theta_r = \theta$
Ending
Y
N

Fig. 4. The flow chart of the experiment

5. Experiment

5.1 Experimental Environment

Deep learning approaches require large amounts of memory because of the computation intensity. In this study, we performed the simulation experiments on a 32 GB computer running 119-Ubuntu (x86_64) with an Intel Xeon E5-2667 (3.20 GHz) processor with 16 cores. The method was simulated using Python and Python libraries such as Matplotlib [20], simply3[21], pandas [22], NumPy[23], and TensorFlow[24]. Programs were developed by

JetBrains PyCharm 2020. In order to compute the energy consumption during the experimentation process based on (9), we downloaded data from some mainstream servers that represent correspondence between CPU utilization and real power from the website (http://www.spec.org/power_ssj2008/results/power_ssj2008.html), as shown in **Table 1** to fit the EM power model and the fitted curve is depicted in **Fig. 5**. The EM power model parameters *r* for each kind server is as follows: 0.9569, 0.7257, 1.5767, 0.7119 and 1.5324.

Table 1. The power of selected server at different CPU utilization

CPU utilization	PowerEdge C5220	HP ProLiant DL2000	IBM System x3630 M4	IBM System x iDataPlex dx360 M3	System x3200 M3
0%	194	178	58.1	92.7	45.0
10%	254	306	79.7	162	48.6
20%	303	349	89.6	182	52.8
30%	345	382	100	199	57.4
40%	386	415	112	216	62.9
50%	427	451	128	234	69.0
60%	481	490	146	255	77.8
70%	539	532	171	276	88.0
80%	597	576	196	298	98.3
90%	635	617	227	318	109
100%	672	660	269	341	119

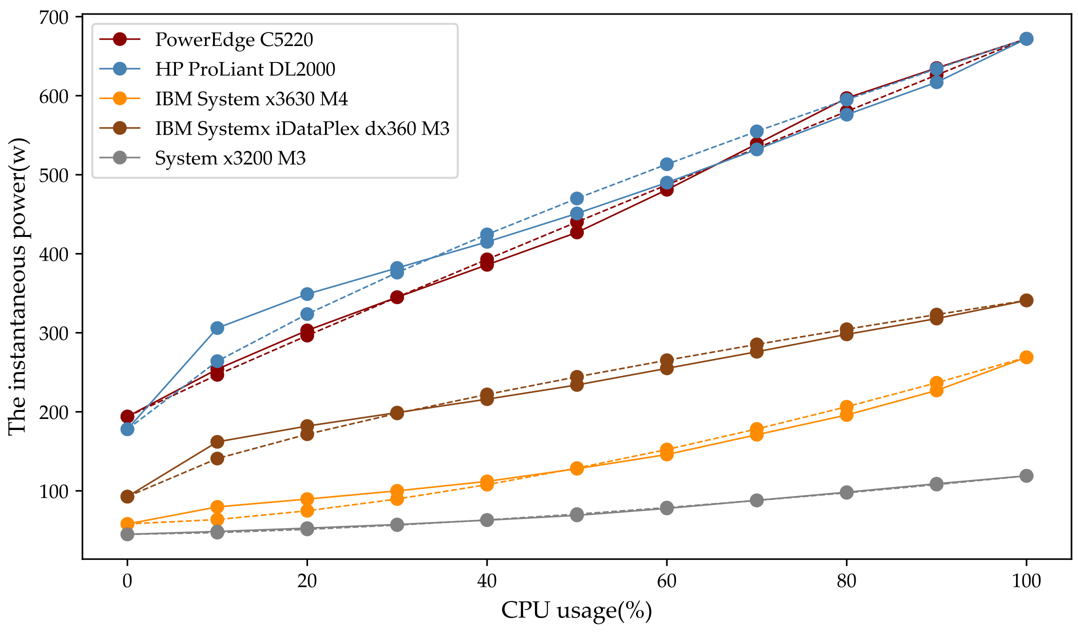

Fig. 5. Fitted curve for the EM power model

The experiments mainly focus on the task scheduling problem in heterogeneous cloud environments, and the number of CPU cores and memory units are listed in **Table 2**.

Table 2. Machines configuration

Machine Type	CPU cores	Memory Units	r of EM power model
PowerEdge C5220	32	1	0.9569
HP ProLiant DL2000	80	2	0.7257
IBM System x3630 M4	120	3	1.5767
IBM System x iDataPlex dx360 M3	80	1.5	0.7119
System x3200 M3	40	1	1.5324

5.2 Dataset

In this study, the data set from Alibaba Cluster Data V2018[20] was used as the benchmark dataset. Compared to Alibaba Cluster Data V2017[11], V2018 has both independent tasks and tasks with dependencies. In practice, we divide the jobs in V2018 into several chunks, and each chunk has 10 jobs arriving in sequential order. We trained DeepEnergyJSV2.0 on the first six chunks. The number of job chunks seen by DeepEnergyJSV2.0 and the number of training iterations are accumulated. It is important to note that the number of tasks in each job chunk varies and that each task contains a different number of task instances, which means that the workload varies with the change of time.

5.3. Experimental Results and Analysis

5.3.1 Convergence and Generalization of PPO Algorithm

In this section, we compared the PPO algorithm with the REINFORCE algorithm in the same training procedures (iteration: 300). The training curve is shown in **Fig. 6**.

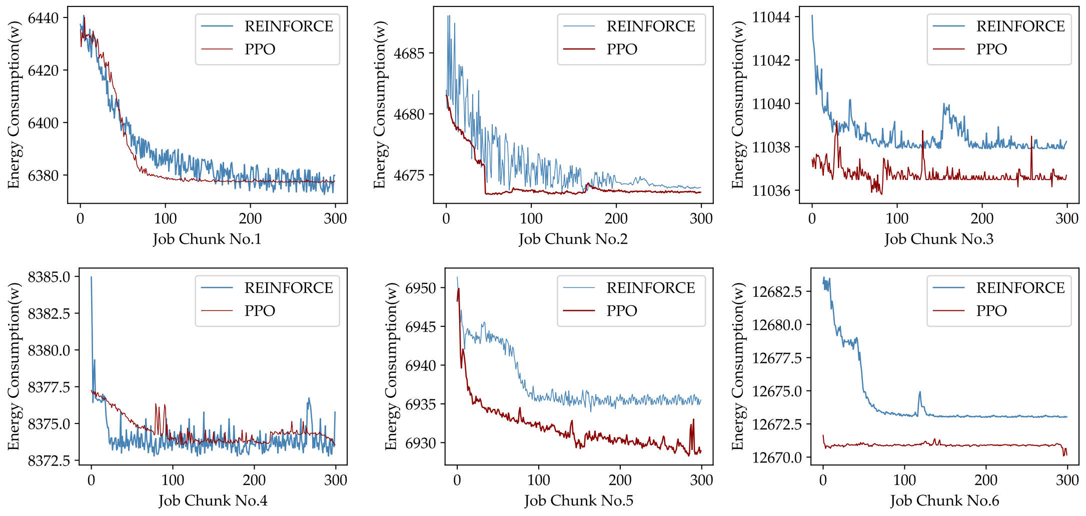

Fig. 6. Training curves of PPO algorithm and REINFORCE algorithm

The figure shows the convergence of PPO proved by the experimental results. In the case of algorithm stability, the curve of PPO is smoother than that of REINFORCE, which indicates better performance. In terms of convergence speed, except for Job Chunk No.4, the number of iterations of PPO to achieve convergence is less than that of REINFORCE. For the converged value, the two algorithms have similar values in job chunks No.1, No.2, and No.4. However, in Job Chunk No.3, No.5, and No.6, PPO has lower energy consumption. From the overall training process, PPO has more advantages over REINFORCE.

5.3.2 PPO and REINFORCE Comparison

In this section, we compare how well the PPO and REINFORCE perform on the test job chunks. **Table 3** lists the energy consumption for job chunks No.7 to No.10 in tabular form, and the line diagram is presented in **Fig. 7**.

Table 3. Energy consumption on test set

Energy (w) / Job Chunk No.	REINFORCE	PPO	First Fit	Random	Tetris
7	2662.112	2641.308	2784.635	2781.0287	2778.182
8	12934.447	12869.129	13941.556	13948.686	13950.213
9	12987.001	12774.974	14831.202	14027.881	13992.043
10	4223.319	4191.319	4330.059	4401.123	4325.423

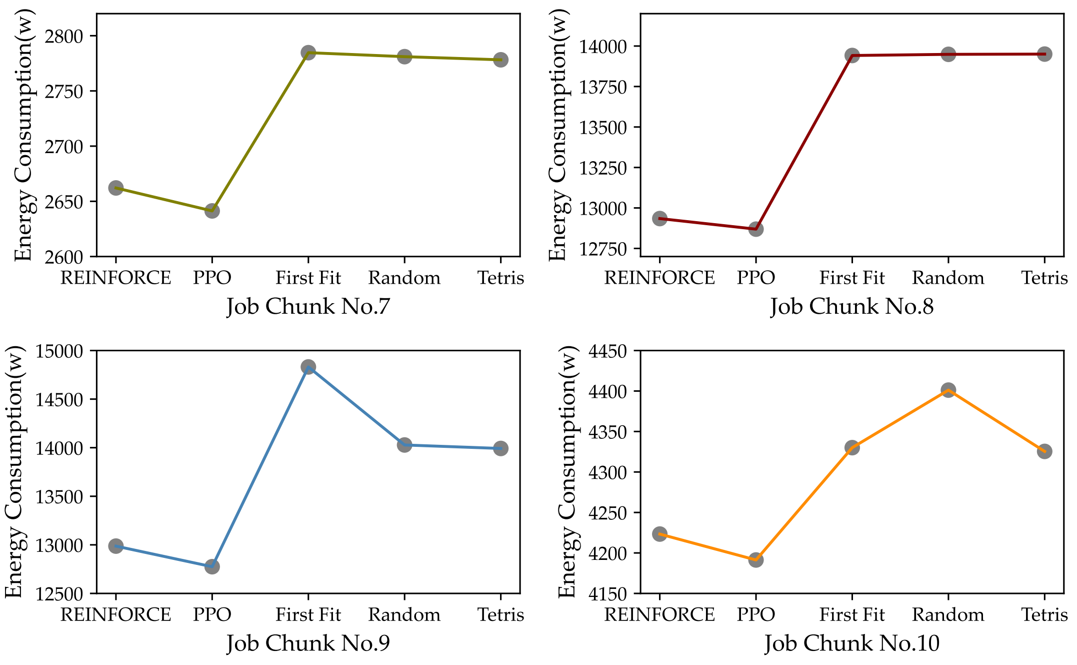

Fig. 7. Energy consumption on test set

As shown in **Fig. 7**, the PPO algorithm performs better than the other algorithms, that is, REINFORCE, First Fit, Random, and Tetris, and REINFORCE performs the second best. In addition, it emphasized that deep reinforcement learning (DRL) algorithms have significant advantages over general heuristic algorithms in solving difficult cloud task scheduling problems.

Table 4. Percentage reduction of energy consumption that PPO compare to the other algorithms

Energy (w) / Job Chunk No.	REINFORCE	First Fit	Random	Tetris
7	7.814%	5.024 %	5.147%	4.927%
8	5.050%	7.739%	7.692%	7.750%
9	1.633%	8.932%	13.864%	8.698%
10	7.577%	4.767%	3.204%	3.100%

Table 4 demonstrates the percentage of energy consumption decreased for the PPO compared to other algorithms. Compared to REINFORCE, PPO reached a maximal reduction of approximately 7.814%, 8.932% versus First Fit, 13.864% compared to Random, and 8.698% to Tetris. For all of the outcomes from the different job chunks, the PPO algorithm was verified as the most effective among the scheduling algorithms used in this study, and the generalization of the method was also validated.

6. Conclusion and Future Work

With the goal of minimizing energy consumption, the work in this paper is an extension and based on previous work. We changed the DRL algorithm included in the task scheduling method from REINFORCE to PPO to update the previous DeepEnergyJS to DeepEnergyJSV2.0. The experimental results show that DeepEnergyJSV2.0 achieves excellent results in hybrid-type tasks that contain both independent tasks and tasks with inner task dependencies to optimize the objective of energy consumption. In addition, DeepEnergyJSV2.0 achieves better overall performance on the training set and can find a better near-global optimal solution than the common heuristic algorithms, such as First Fit, Random, and Tetris, on the test set.

If applied in an actual cloud environment, our proposed DeepEnergyJSV2.0 will be an alternative to many existing task scheduling methods based on heuristic algorithms. However, the shortcomings of our research and directions for future studies are as follows:

1. Real cloud computing environments are more complex, but our research was conducted only for a single data center.
2. The research in this study is only aimed at the optimization of a single goal. In practice, a comprehensive consideration of multi-objective optimization will be more rewarding.
3. Containerization technology is an emerging virtualization technique that plays an increasingly important role in the future. Developing efficient strategies for scheduling tasks in containers is a focus for future research.
4. The performance of PPO is not stable enough. As shown in **Fig. 6**, the results of PPO are not better than REINFORCE in job chunks No.1, No.2, and No.4. The reason is that the task types in different job chunks differ. This result indicates that the performance of the PPO algorithm is not superior to the reinforcement learning algorithm in some specific scenarios. These specific scenarios need to be studied in the future to determine their underlying patterns.
5. In the future, cloud computing needs to combine with other computing model like edge computing, fog computing, serverless computing and quantum computing [25]. How to integrate these computing models to complete computing tasks and use AI/ML to optimize them is the main research direction in the future, which also poses a huge challenge to us.

Acknowledgement

This paper is financially supported by Shandong Key R&D Program (Major Science and Technology Innovation Project) (2020CXGC010704), Key R & D Projects of Shandong Province(2020JMRH0201), Key Projects of New and Old Kinetic Energy Conversion 2020, Qingdao independent innovation major project(20-3-2-12-xx), project of introducing urgently needed talents in key supported regions of Shandong Province.

References

[1] Y. Yin, Y. Xu, W. Xu, M. Gao, L. Yu, and Y. Pei, “Collaborative Service Selection via Ensemble Learning in Mixed Mobile Network Environments,” *Entropy*, vol. 19, no. 7, Jul. 2017. Article (CrossRef Link)

[2] J. Yu, Z. Kuang, B. Zhang, W. Zhang, D. Lin, and J. Fan, “Leveraging Content Sensitiveness and User Trustworthiness to Recommend Fine-Grained Privacy Settings for Social Image Sharing,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1317–1332, May 2018. Article (CrossRef Link)

[3] J. Yu, B. Zhang, Z. Kuang, D. Lin, and J. Fan, “iPrivacy: Image Privacy Protection by Identifying Sensitive Objects via Deep Multi-Task Learning,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 5, pp. 1005–1016, May 2017. Article (CrossRef Link)

[4] L. Y. Zuo and Z. B. Cao, “Review of scheduling research in cloud computing,” *Application Research of Computers*, vol. 29, no. 11, pp. 4023–4027, 2012. Article (CrossRef Link)

[5] C. J. C. H. Watkins, “Learning from delayed rewards,” Ph.D. dissertation, King’s College, Cambridge United Kingdom, 1989.

[6] V. Mnih et al., “Playing Atari with Deep Reinforcement Learning,” *arXiv:1312.5602 [cs]*, Dec. 2013. Article (CrossRef Link)

[7] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. Article (CrossRef Link)

[8] J. Stuart, Norvig, and Peter, *Artificial Intelligence: A Modern Approach*, 1995.

[9] C. He, Y. Yang, and B. Hong, “Cloud Task Scheduling Based on Policy Gradient Algorithm in Heterogeneous Cloud Data Center for Energy Consumption Optimization,” in *Proc. of 2020 International Conference on Internet of Things and Intelligent Applications (ITIA)*, pp. 1–5, Nov. 2020. Article (CrossRef Link)

[10] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Mach Learn*, vol. 8, no. 3, pp. 229–256, May 1992. Article (CrossRef Link)

[11] C. Lu, K. Ye, G. Xu, C.-Z. Xu, and T. Bai, “Imbalance in the cloud: An analysis on Alibaba cluster trace,” in *Proc. of 2017 IEEE International Conference on Big Data (Big Data)*, pp. 2884–2892, Dec. 2017. Article (CrossRef Link)

[12] “Alibaba Cluster Trace Program,” Alibaba, 2021. Accessed: Jan. 04, 2022. [Online]. Available: https://github.com/alibaba/clusterdata/blob/4221e02342dd01fd30a9800b19b7f365a3fd5ac8/cluster-trace-v2018/trace_2018.md

[13] A. F. S. Devaraj, M. Elhoseny, S. Dhanasekaran, E. L. Lydia, and K. Shankar, “Hybridization of firefly and Improved Multi-Objective Particle Swarm Optimization algorithm for energy efficient load balancing in Cloud Computing environments,” *Journal of Parallel and Distributed Computing*, vol. 142, pp. 36–45, Aug. 2020. Article (CrossRef Link)

[14] H. Peng, W.-S. Wen, M.-L. Tseng, and L.-L. Li, “Joint optimization method for task scheduling time and energy consumption in mobile cloud computing environment,” *Applied Soft Computing*, vol. 80, pp. 534–545, Jul. 2019. Article (CrossRef Link)

[15] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, “Q-learning based dynamic task scheduling for energy-efficient cloud computing,” *Future Generation Computer Systems*, vol. 108, pp. 361–371, Jul. 2020. Article (CrossRef Link)

[16] S. Seth and N. Singh, "Dynamic heterogeneous shortest job first (DHSJF): a task scheduling approach for heterogeneous cloud computing systems," *Int. j. inf. tecnol.*, vol. 11, no. 4, pp. 653–657, Dec. 2019. Article (CrossRef Link)
[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347 [cs]*, Aug. 2017. Article (CrossRef Link)
[18] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. Article (CrossRef Link)
[19] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust Region Policy Optimization," in *Proc. of the 32nd International Conference on Machine Learning*, pp. 1889–1897, Jun. 2015. Article (CrossRef Link)
[20] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, vol. 9, no. 03, pp. 90–95, May 2007. Article (CrossRef Link)
[21] "SimPy," Team SimPy, 2020. [Online]. Available: https://simpy.readthedocs.io/en/latest/index.html
[22] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proc. of the 9th Python in Science Conference*, pp. 56–61, 2010. Article (CrossRef Link)
[23] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Computing in Science Engineering*, vol. 13, no. 2, pp. 22–30, Mar. 2011. Article (CrossRef Link)
[24] Martín Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: https://www.tensorflow.org/
[25] Sukhpal Singh Gill et al., "AI for Next Generation Computing: Emerging Trends and Future Directions," *Internet of Things*, 2022. Article (CrossRef Link)

Yongquan Yang is currently a lecturer in Ocean University of China, Qingdao, China. He received a Ph.D. degree from the Ocean University of China in 2013. His current research interests are in the fields of cloud Computing and big data processing.

Cuihua He is currently a postgraduate student in Ocean University of China, Qingdao, China. Her current research interests are in the fields of cloud computing and deep reinforcement learning.

Bo Yin is currently a Professor in Ocean University of China, Qingdao, China. He received the Ph.D. degree from the Department of Computer Science and Technology, Ocean University of China, Qingdao, China, in 2006. His current research interests are in the fields of acoustic systems, embedded system designing, and intelligent control technology.

Zhiqiang Wei is currently a Professor in Ocean University of China, Qingdao, China. He received Ph.D. degree from Tsinghua University, Beijing, China, in 2001. His current research interests are in the fields of intelligent information processing, intelligent computing of big data.

Bowei Hong is currently a postdoctoral researcher in Ocean University of China, Qingdao, China. She received a Ph.D. degree from the Ocean University of China in 2018. Her current research interests are in the fields of cloud computing and deep learning.