IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 5, 2020, accepted December 16, 2020, date of publication January 14, 2021, date of current version January 22, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3051672

A High-Efficient Joint 'Cloud-Edge' Aware Strategy for Task Deployment and Load Balancing

YUNMENG DONG, GAOCHAO XU, MENG ZHANG, AND XIANGYU MENG
College of Computer Science and Technology, Jilin University, Changchun 130012, China

Corresponding author: Xiangyu Meng (xymeng512@jlu.edu.cn)

This work was supported in part by the National Key Research and Development Plan of China under Grant 2017YFA0604500, in part by the Jilin Provincial Industrial Innovation Special Foundation Project under Grant 2017C028-4, and in part by the Jilin Province Science and Technology Development Plan Project under Grant 20190201180JC.

ABSTRACT Task deployment has become a research hotspot for load balancing in joint "cloud-edge" datacenter. In view of the problem that most of the hosts are overloaded in the current joint "cloud-edge" datacenter, which may cause unbalanced load in the center, existing research mainly pay attention to the problem of unilateral load balancing of cloud computing center or edge computing center. In order to realize efficient deployment of "cloud-edge" tasks and overall load balancing, on the basis of the deployment mode of joint "cloud-edge", this paper proposes a resource management and task deployment strategy JCETD (Joint Cloud-Edge Task Deployment) based on pruning algorithm and deep reinforcement learning. The main idea consists of two parts: firstly, the set of "cloud-edge" hosts is pruned according to the attribute value of the physical host. Then, there will be a non-dominated set of joint hosts which reduces the computational complexity of the whole algorithm and improve the computational efficiency of the system. Secondly, the problem of task deployment is simulated as a deep reinforcement learning process under the "cloud-edge" model. Through the continuous exploration and utilization of the system environment, the tasks are reasonably and efficiently deployed in the cloud computing center and edge computing center. Finally, the "cloud-edge" system can achieve an efficient computing performance and overall load balancing. The experimental results show that the proposed algorithm significantly reduces the total completion time and average response time compared with the existing research, which effectively optimizes the service ability and realizes the load balancing of the joint "cloud-edge" system.

INDEX TERMS Task deployment, joint "cloud-edge", pruning, deep reinforcement learning, load balancing.

I. INTRODUCTION

Joint "cloud-edge" computing is a research direction with particular prospects after distributed computing, cloud computing and edge computing [1], It is a hot topic in current research. Joint "cloud-edge" computing is a new architecture model which is able to supplement the computing, storage and other resources of the cloud computing center in real time when the computing, storage, and bandwidth resources of the edge computing center are insufficient, and to meet the resource requirements of edge applications. A large number of physical hosts are deployed in the resource pool of the joint "cloud-edge" datacenter, and the remaining resource of physical hosts varies from time to time. When the amount of task request resources submitted by users is greater than the remaining resources of the current physical host or the current physical host resources are occupied and not completely released, the task deployment efficiency will be reduced and the deployment will fail. When the amounts of task request resources approach the remaining resource of the currently deployed host, the current task will be processed slowly. At the same time, it will also make subsequent tasks unable to be deployed effectively, and make the load of the "cloud-edge" datacenter unbalanced. Therefore, it fails to provide users with real-time calculation results and fails to show the advantages of joint "cloud-edge" computing.

At present, the task deployment problem of load balancing for the joint "cloud-edge" datacenter has become a hot research and famous topic. In the environment of joint "cloud-edge" mode, it is necessary to design a task

The associate editor coordinating the review of this manuscript and approving it for publication was Zhibo Wang.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

deployment model with the efficient computing ability and load balancing efficiency. The key to achieve the above goals is to place the requested task on a reasonable resource. By placing the requested task on the optimal resource at the user level, the average response time of the task could be reduced, and the calculated results could be returned to users in a short time, which improves the service quality to users. At the system level, it not only increases the throughput of the joint "cloud-edge" system, but also realizes the long-term load balancing of the joint cloud-edge system. However, the current research on the task deployment strategy of the joint cloud-edge system is not very complete, and there is no guarantee that the task will be placed on the optimal resource all the time.

In order to achieve efficient computing ability and load balancing in the joint "cloud-edge" system environment, this paper proposes a resource management and task deployment strategy based on pruning algorithm [2] and deep reinforcement learning. First of all, the physical host set of the current cloud computing center and the edge computing center are combined to form a joint host set. The unpromising physical hosts are pruned through the idea of pruning algorithm to obtain a non-dominated host set, which is taken as the initial state of the deep reinforcement learning algorithm. Then, the task deployment process is simulated as a process of the deep reinforcement learning algorithm. Through continuous exploration of the environment, the efficient deployment of tasks and load balancing of joint "cloud-edge" are finally realized.

This paper aims to achieve efficient computing performance and load balancing of joint "cloud-edge" computing, and provide users with a better service ability. The crux of the matter to achieve the above goals is to efficiently allocate the requested tasks to the physical hosts of the "cloud-edge" computing for calculations, so that the joint "cloud-edge" datacenter can have a stronger computing ability. Thus, the best quality system service performance can be provided to users, and the load balancing of joint "cloud-edge" system is realized.

The main contributions of this paper are as follows:

- On the basis of the joint "cloud-edge" architecture, the deep reinforcement learning algorithm is used to realize the task efficient deployment and long-term load balancing of the "cloud-edge" system.
- The idea of pruning algorithm is seamlessly connected to the process of the deep reinforcement learning, which not only prunes the unreasonable physical host, but also reduces the state space of deep reinforcement learning algorithm.
- Through the DDPG algorithm of deep reinforcement learning, tasks can be continuously and efficiently deployed in the continuous action space.

The rest of this paper is organized as follows. In Section II, the paper briefly introduces the related work of task deployment methods in the current edge computing and cloud computing environments. In Section III, the premise of the problem is briefly described, and then the question is formalized. In Section IV, the system architecture of task deployment in the joint "cloud-edge" computing environment is designed. Then, we introduce the design and implementation process of the algorithm in detail. In Section V, the experiment results are introduced in detail, which proves that the proposed algorithm is efficient. We conclude the paper in Section VI.

II. RELATED WORKS

Task deployment has become a hot topic in the research of current popular computing paradigms. Through effective management of current resources and reasonable deployment of computational tasks, efficient computing performance and load balancing of the system are achieved. In current research, popular task deployment problems can be roughly divided into three categories according to different implementation methods: collaborative computing, algorithms based on reinforcement learning, and algorithms based on load balancing.

Collaborative computing [3]–[5] is mainly to solve the current computing system when the computing ability and scale cannot meet the current needs, and it needs to be jointly completed by other partners. Literature [6] Sahni Y *et al.* proposed a task allocation model based on data-aware to jointly schedule tasks and network flows in collaborative edge computing. By mathematically modeling of joint problems, the total completion time of the application is minimized. Literature [7] Schafer D *et al.* put forward a method of edge and cloud collaborative hybrid scheduling, the corresponding perceptual scheduler will extract the characteristics of the task, and then decide whether the task is executed in the cloud or at the edge. Literature [8] Li *et al.* proposed a two-level scheduling optimization scheme in the "edge-cloud" environment, the first level scheduling deploys most of the tasks in the edge computing center, the second level scheduling is deployed in the cloud and edge according to a certain strategy under the condition of insufficient edge center computing resources, this scheme has better performance in minimizing delay and completion time, and reducing total cost. Literature [9] Xie *et al.* proposed a workflow scheduling strategy DNCPSO in the cloud-edge environment, which reduced the maximum completion time and scheduling cost to some extent, and obtained a compromise result. The above-mentioned studies all start from the perspective of the task terminal or the physical host alone, which means modeling the problem as a time delay or energy consumption minimization problem. However, it does not consider the impact of task scheduling on the overall load balance of the system.

Reinforcement learning is an important branch of machine learning, more importantly, it does not learn from the training set with labels. In contrast, it learns from the feedback information of the environment, which is very important for task scheduling problems, because high-quality labeled data cannot be generated. Literature [10] Orhean *et al.* used reinforcement learning to solve the problem of workflow scheduling of heterogeneous distributed resources, reducing task execution time to a certain extent. Deep reinforcement learning is the integration of reinforcement learning and deep learning, which can solve more complex scientific problems. Literature [11] Wang *et al.* used DQN model to solve the problem of workflow scheduling with the goal of minimizing the completion time and cost. Literature [12] Dong *et al.* proposed a task scheduling algorithm based on Deep Reinforcement Learning Architecture (RLTS), which dynamically scheduled tasks with priority relationship to the cloud server to minimize the task execution time. Literature [13] Xiong *et al.* proposed a resource allocation strategy for the edge computing system of the Internet of Things, which formalized the resource allocation problem as a Markov decision process, and used deep reinforcement learning to solve the problem, the goal is to minimize the long-term weighted sum of the average task completion time and the average number of requested resources. However, this strategy only considers the cost of the system, and does not consider the load balancing of the system too much. Literature [14] Cheng *et al.* proposed a novel resource allocation and task scheduling system based on deep reinforcement learning (DRL), the two-stage RP-TS learns the ever-changing environment (user request modes and realistic electricity prices) to automatically generate the best long-term decisions, and minimize the energy costs of large CSPs (Cloud Service Providers).

Load balancing is an important way to achieve efficient task deployment. Literature [15] Tham C *et al.* proposed a load balancing scheme for distributed computing at the edge of the network, with the objective of minimizing the overall processing time of the application while still satisfying the wireless channel capacity and link contention constraints, and modeling the load balancing problem between edge nodes as an optimization problem, then the gradient descent algorithm is used to solve the problem. Literature [16] Jyoti *et al.* proposed a novel dynamic resource allocation method based on load balancing and service broker, by predicting the execution environment of tasks, tasks are deployed to a virtual machine according to their priority, and load balancing is performed in the virtual machine, which increases the throughput of the system and reduces the response time of the users. Li *et al.* in [17] proposed a load balancing strategy for task allocation in edge computing based on intermediate nodes to solve the problem of load balancing between different edge nodes, it uses the intermediate nodes to monitor the global information, gets the attributes of the edge nodes in real time, and distributes the tasks to the nodes with the minimum load according to the task allocation model. Experiments show that this method could balance the load between edge nodes and reduce the completion time of tasks. Literature [18] Ghasemi *et al.* proposed a multi-objective virtual machine replacement algorithm based on reinforcement learning (RLVMrB) to achieve load rebalancing by obtaining the virtual machine-to-physical host mapping matrix available in the datacenter, it succeeds to achieve load balancing between and within physical hosts. Literature [19] Tong *et al.* proposed a dynamic scheduling algorithm DQTS, which aims to balance the load in the cloud computing environment. Through the combination of Q learning and deep neural network to form a deep Q learning method, it has better scalability and can perform load balancing more effectively than other algorithms.

On the basis of above research, this paper integrates the joint "cloud-edge" model with deep reinforcement learning effectively, explores the environment continuously under the deep reinforcement learning model, and deploy task sets in the cloud computing center and edge computing center respectively, so that tasks can be processed efficiently and the average response time is minimized. Moreover, better computing ability and load balancing of the joint "cloud-edge" datacenter can be achieved.

III. THE PROPOSED PROBLEM AND ITS FORMALIZATION

A. PROBLEM STATEMENT

In the environment of unilateral computing (cloud computing or edge computing), there are a large number of compute-intensive tasks in the system that need to be processed. Generally, the system will randomly deploy computational tasks on the hosts in the unilateral datacenter (cloud or edge). When the amount of resources requested by users in the system is greater than the remaining resources of physical host of the current unilateral datacenter, it may result in the decline of the computing and service capacity of the datacenter. Besides, it is impossible to return the calculation results to the current user in real time, and the datacenter may be unable to reach the state of load balancing at the current moment. In addition, when the computing resources of the unilateral datacenter are completely occupied and not released in a timely manner, the task is placed in the datacenter, which may lead to the loss of users' data and fail to provide effective computing services for users. Obviously, in the face of large-scale computational tasks, different task deployment mode and resource allocation schemes will lead to different computational efficiency and state of system load balancing. In the case of limited computing resources in unilateral data centers, the optimal deployment mode and strategy is undoubtedly an important way to achieve high-quality computing service capabilities and load balancing of the system. Therefore, in view of the above shortcomings, the use of the joint "cloud-edge" mode, the design of a high-quality task deployment model and strategy are necessary conditions to achieve the above goals, as shown in Figure 1.

B. FORMALIZATION OF THE PROBLEM

In the joint "cloud-edge" computing model, the task deployment problem can be formalized as follows: in a Δt time period, the system collects n task requests, which are independent of each other and have no dependencies. They need to be deployed to a joint datacenter composed of edge computing center and cloud computing center. There are h tasks to be

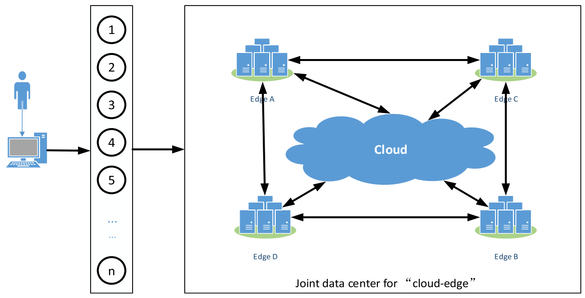


FIGURE 1. A joint architecture for "cloud-edge" datacenter.

deployed in the edge center, and *n-h* tasks to be deployed in the cloud computing center. Assuming that in the same network environment, there are *m* available hosts in the edge computing center and *q* available hosts in the cloud computing center, which are heterogeneous and dynamic, and use the space sharing allocation strategy. The problem can be described as follows: the edge computing center and the cloud computing center are combined to form a joint datacenter, and the high-quality deployment strategy enables the task set within each Δt time to quickly complete the calculation and response. Through high-quality deployment mode and strategy, the problem of joint "cloud-edge" efficient task deployment and load balancing has been solved from a long-term perspective.

Assume that each computational task is represented by $TK_n(B_n, D_n, T_n)$, the deployment is completed in the joint "cloud-edge" datacenter according to certain strategy. Where B_n is the size of the input data, D_n is the number of CPU cycles required to complete the calculation task, B_n and D_n are positively correlated, T_n is the maximum tolerable delay of task TK_n which means the total time to complete the calculation should not exceed T_n. We regard the task average response time as an important constraint of the system optimization problem, and it is also the key to ensuring the QoS experience of users. The average response time is defined as follows:

$$ART = \frac{1}{|T|} \cdot \sum_{P_k} \sum_{\varphi_i^j} PT_k\left(\varphi_i^j\right) \tag{1}$$

where φ_i^j represents the current task, i represents the task type, j represents the task arrival sequence number, and P_k represents the k-th processing unit. $PT_k(\varphi_i^j)$ represents the processing time of task φ_i^j on P_k including the waiting and processing time after the task itself arrives.

In order to measure the degree of the "cloud-edge" datacenter's load balancing, the degree of joint load balancing is used as a metric. Load balancing degree refers to the degree of balancing of load distribution on each processing node of parallel system, which is an important factor affecting parallel efficiency. When the entire system has a large number of tasks, the load on each node may be unbalanced, which will reduce the utilization of the entire system. In this paper, the variance of residual load rate of host resource is used to represent the load balancing degree, where the residual load rate of host i can be expressed as follows:

$$RLR_i = \frac{R_i}{total} \tag{2}$$

R_i is the remaining resources of the current physical host i, and *total* is the total remaining resources of "cloud-edge" datacenter, where R_i and *total* can be expressed as follows:

$$R_i = \alpha R_{ic} + \beta R_{im} \tag{3}$$

$$total = \sum_{i=1}^{n} R_i \tag{4}$$

$$\alpha + \beta = 1 \tag{5}$$

R_{ic} is the amount of remaining resources of the host CPU; R_{im} is the amount of remaining resources of the host memory; α is the weight of the CPU, and β is the weight of the memory.

The residual load rate of each node can be calculated through formula (2)(3)(4)(5), and then the standard deviation formula can be used to calculate the load balancing degree of joint "cloud-edge" datacenter, as shown below:

$$LBD = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left(RLR_i - \frac{1}{M}\sum_{i=1}^{M} RLR_i\right)^2} \tag{6}$$

From the above optimization goals, the ultimate goal of this paper is to achieve load balancing in the joint "cloud-edge" datacenter based on minimizing the average response time. This paper is a two-goal optimization problem, which is load balancing degree and average response time, and they are all as small as possible. When the minimum value is reached, it is the optimal solution of this paper. Therefore, our overall optimization goal can be expressed as follows:

$$V = \mu \cdot LBD(RLR) + \lambda \cdot ART(\varphi) \tag{7}$$

In summary, the problem of this paper can be formalized as follows:

$$\begin{aligned} \min_{A} \mu \cdot & \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left(RLR_i - \frac{1}{M}\sum_{i=1}^{M} RLR_i\right)^2} \\ & + \lambda \cdot \frac{1}{|T|} \cdot \sum_{P_k} \sum_{\varphi_i^j} PT_k\left(\varphi_i^j\right) \\ \text{s.t. C1} \quad & \lambda + \mu = 1 \\ \text{C2} \quad & \frac{D_n}{f_n} + \frac{S_n}{v} \le T_n \\ \text{C3} \quad & \sum_{i=1}^{I} (\alpha R_{ic} + \beta R_{im}) \le T_{\text{total}} \quad \forall i \in I. \end{aligned} \tag{8}$$

In the above optimization goals, $A = \{e_1, e_2, e_3, \ldots, c_i, \ldots, c_n\}$ is the deployment plan within the current time Δt, e indicates that the task is deployed on the host of edge computing center, and c indicates that the task is deployed on the host of cloud computing center. Constraint C1 is the sum of the weight of the task average response time and load balancing degree to 1. Constraint C2 is that the transmission time and the completion time of the computing node should not exceed the completion time set by the task. Constraint C3 is that the remaining resources of all physical hosts should be less than the total resources of physical host of current cloud computing center and edge computing center. Since the above problem is not a convex optimization problem, but an NP-hard problem, therefore, this paper uses a deep reinforcement learning algorithm to solve the problem.

In order to improve the deployment efficiency and reduce the complexity of the state space, this paper uses the idea of pruning algorithm to prune the current set of joint "cloud-edge" host. It mainly uses the dominance relationship between physical host attributes to pruning unpromising physical hosts, and obtain a non-dominated physical host set. This achieves the purpose of improving the efficiency of the deployment task of the "cloud-edge" datacenter, reducing the state space of the system, and thereby reducing the complexity of the algorithm. The dominance relationship is defined as follows:

The attribute value set of the physical host P_i can be expressed as:

$$V\left(P_i\right) = \left(v_1\left(P_i\right), v_2\left(P_i\right), \cdots, v_r\left(P_i\right)\right) \tag{9}$$

where $v_1(P_i)$ is the first attribute of physical host i, $v_r(P_i)$ is the r-th attribute of physical host i. Different attributes may have different optimization directions. To maximize the value of the attribute, we multiply the value of attributes by -1. In this way, our optimization goal can be expressed as:

$$\min_{P_i}\left(v_1\left(P_i\right), v_2\left(P_i\right), \cdots, v_r\left(P_i\right)\right) \tag{10}$$

Suppose there are two physical hosts P_i and P'_i, their attribute values are represented by formula (9) and (11), respectively. If formula (12) (13) can be satisfied, we think that P_i dominates P'_i.

$$V\left(P'_i\right) = \left(v_1\left(P'_i\right), v_2\left(P'_i\right), \cdots, v_r\left(P'_i\right)\right) \tag{11}$$

$$\forall r \in \{1, 2, \cdots, r\}, v_r\left(P_i\right) \leq v_r\left(P'_i\right) \tag{12}$$

$$\exists r \in \{1, 2, \cdots, r\}, v_r\left(P_i\right) < v_r\left(P'_i\right) \tag{13}$$

For simplicity, we use $P_i \succ P'_i$ to represent that P_i dominates P'_i. Once P'_i was dominated, we can infer that the P'_i is not the optimal solution. The dominated solution is pruned through the pruning algorithm, and then the remaining hosts form a set of non-dominated solution, which can effectively improve the deployment efficiency of the joint "cloud-edge" datacenter and reduce the state space of the deep reinforcement learning algorithm to a certain extent.

The first step of modeling the system environment using the deep reinforcement learning algorithm is to define the state space of the system. When the agent perceives the environment, the system state is the state of the remaining resources of "cloud-edge" physical host and the current task. The state space is defined as:

$$X = \{x_1, x_2, x_3, \cdots, x_n\} \tag{14}$$

where x_i is the state of task i, that is, x_i is:

$$x_i = \{B_i, D_i, T_i, R_{1c}, R_{1m}, \cdots, R_{kc}, R_{km}\} \tag{15}$$

where B_i,D_i,T_i are the state of task i, and R_{kc},R_{km} are the resources remaining state of the CPU and memory of host k.

This paper mainly deploys the tasks collected in Δt time to the set of joint "cloud-edge" host processed by the pruning operation, so that each task can be deployed quickly. Therefore, the system action corresponds to the edge host or cloud host for each task. The action space is as follows:

$$Y = \left\{y_1^{\text{edge}}, y_2^{\text{edge}}, \cdots, y_{k-1}^{\text{cloud}}, y_k^{\text{cloud}}\right\} \tag{16}$$

System reward refers to the feedback value given by the environment after making a certain action in a certain system state, indicating the degree of good or bad taking a certain action in that state. In this paper, we use the degree of load balancing of the system after each deployment action as a reward. The smaller the load balancing degree, the more balanced the load in the "cloud-edge" datacenter. A system reward *Syreward* is expressed as follows:

$$Syreward = \begin{cases} 1, & \text{if} \quad lb(x_t, y_t) \leq 0; \\ -1 & \text{if} \quad lb(x_t, y_t) > 0. \end{cases} \tag{17}$$

$$lb\left(x_t, y_t\right) = LBD\left(x_{t+1} \mid x_t, y_t\right) - LBD\left(x_t\right) \tag{18}$$

where LBD is the cost function of load balancing degree, x_t is the state of the system at time t, and y_t is the action of the system.

As mentioned above, in order to achieve the above optimization goals, the proposed problem is formalized as a process of deep reinforcement learning. Through the deep reinforcement learning algorithm, the best computing nodes are found for the tasks in the set for processing to maximize the joint load balancing of the edge computing center and the cloud computing center. The detailed algorithm process will be given below.

IV. A "CLOUD-EDGE" ALGORITHM FOR EFFICIENT TASK DEPLOYMENT

A. SYSTEM ARCHITECTURE DESIGN

Figure 2 describes the system architecture in the joint "cloud-edge" computing environment. It shows the interaction between the JCETD proposed and other entities, and also reflects the important role that the algorithm plays in the entire architecture. First, the monitor gets information about the task set and "cloud-edge" physical hosts, and combines the physical hosts in the cloud computing center and the edge computing center into a set, then the physical host set is pre-processed by pruning sub-module of JCETD. After the pre-processing is completed, the current task set and the pre-processed host set are sent to the sub-module DDPG of JCETD, then the deployment strategy is generated using the algorithm in this paper. Finally, the deployment strategy is applied to the deployment controller, and the set of tasks received in Δt time is deployed to the corresponding physical

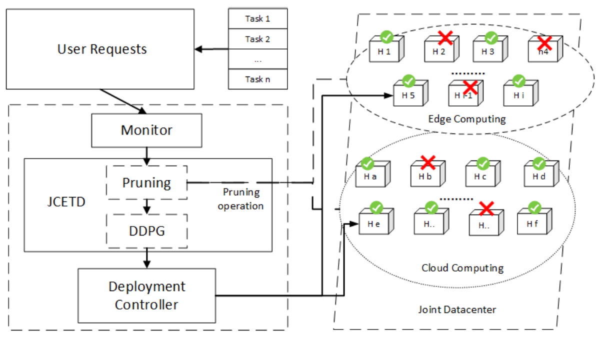


FIGURE 2. The view of JCETD's architecture.

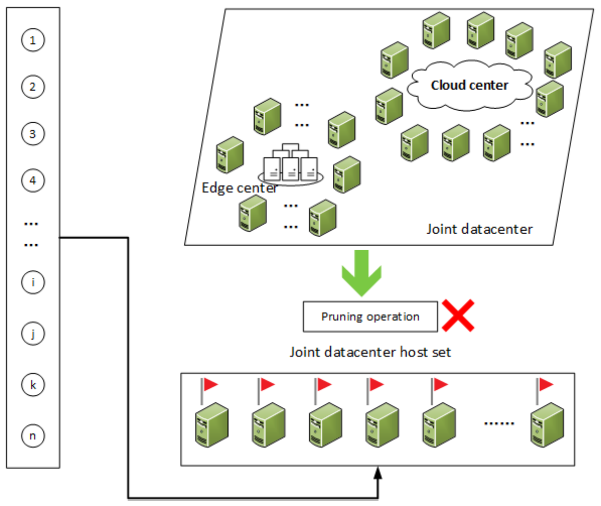


FIGURE 3. The process of JCETD task deployment.

host of "cloud-edge" datacenter through the deployment strategy in this paper.

B. MAIN IDEA OF JCETD

The JCETD proposed in this paper is a task deployment strategy for load balancing of joint "cloud-edge" datacenter based on the idea of pruning algorithm and the deep reinforcement learning algorithm, which is used to deploy the collected tasks on the physical hosts of "cloud-edge". Firstly, the physical hosts of cloud computing center and edge computing center are formed into a set. Several attributes of physical host are selected, and the physical hosts in the set is compared in pairs by using the idea of pruning algorithm, so as to obtain a non-dominated set of physical hosts. Secondly, the set of physical hosts obtained by the pruning operation is used as the initial state space of the deep reinforcement learning algorithm to simulate the process of deep reinforcement learning, and the tasks collected by each timestamp are deployed on the host in the joint "cloud-edge" datacenter. Details of the implementation are given below.

C. IMPLEMENTATION OF JCETD

Step 1: Monitor the user's task request and initialize the parameters of the algorithm. The initial stage of the algorithm will collect the task in Δt time and form a set, which will be used as the object of this problem, and the core algorithm will be executed once in each Δt time, and the task set collected during the execution of the algorithm will be used as the next problem to be processed. As mentioned earlier, the number of tasks in the set, and the number of hosts in the edge computing center and in the cloud computing center in a Δt time are denoted as n, m, q, respectively.

Step2: Initialization of the set of physical hosts of "cloud-edge" datacenter. It is known that the physical host set of cloud computing center is $PHC = \{phc_1, phc_2, \cdots, phc_q\}$, and the physical host set of edge computing center is $PHE = \{phe_1, phe_2, \cdots, phc_m\}$. Define an empty set $TOTAL = \{\}$, merge the set of physical hosts of the cloud computing center and the edge computing center into a new host set and assign it to the empty set $TOTAL$, that is, $TOTAL = \{PHC + PHE\}$. The $TOTAL$ set of hosts in the "cloud-edge" datacenter serves as the initial set of the pruning algorithm.

Step 3: The calculation of attribute values of the physical host. Take the set of $TOTAL$ as the initial set of pruning operations, i.e., $TOTAL = \{phc_1, phc_2, \cdots, phc_q, \cdots, phe_1, phe_2, \cdots, phc_m\}$, the size of the set is $q+m$. Then, we select three attributes of the physical host as the basis of pruning algorithm, which are the memory resources and the CPU resources of remaining of the physical host, and the posterior probability of physical host. The remaining resources of memory and CPU are known, the posterior probability of physical host is calculated according to Bayesian theorem. Defines that a task to be deployed to a physical host for processing as event A, and define event Bi as physical host i to be selected for processing tasks. In a Δt time, among the task requests received by the system, the task with the largest resource demand is selected, and the ratio of its resource demand R_t to the current computing capacity R_i of the physical host is taken as the load proportion, in fact, the larger load ratio of a physical host is, the less likely it could be selected for processing tasks. So, the prior probability of a physical host can be expressed as:

$$P(A \mid B_i) = 1 - \frac{R_t}{R_i} \tag{19}$$

And the "cloud-edge" datacenter has m+q physical hosts, then

$$P(B_i) = \frac{1}{m+q} \tag{20}$$

Then, the posterior probability $P(B_i|A)$ of each physical host in "cloud-edge" datacenter can be obtained by bringing formulas (19) and (20) into the Bayesian formula, as follow:

$$\begin{aligned} P(B_i \mid A) &= \frac{P(B_i)\,P(A \mid B_i)}{\sum_{i=1}^{n'} P(B_i)\,P(A \mid B_i)} \\ &= \frac{R_i - R_t}{R_i(m+q)\cdot\sum_{i=1}^{m+q}\left(\frac{R_i-R_t}{(m+q)R_i}\right)} \end{aligned} \tag{21}$$

The attribute value of memory can be expressed as follows:

$$v_1^m = \alpha_1 M_1 + \alpha_2 M_2 \tag{22}$$

where M_1 represents the memory allocated to the physical host, and M_2 represents the utilization of memory, the weight coefficients add up to 1.

$$\alpha_1 + \alpha_2 = 1 \tag{23}$$

Similarly, the attribute value of CPU can be expressed as:

$$v_2^c = \beta_1 C_1 + \beta_2 C_2 \tag{24}$$
$$\beta_1 + \beta_2 = 1 \tag{25}$$

The attribute value of the physical host is $V_{ph} = (v_1^m, v_2^c, v_3^p)$, where v_3^p is equal to $P(B_i|A)$.

Step4: The pruning operates of physical host set of the ''cloud-edge'' datacenter. Through the above steps, the physical host set of the ''cloud-edge'' datacenter and the attribute values of physical host are obtained, and then the pruning operation is performed. The set of current physical hosts and the attribute value set of physical hosts are known. First, set up an empty set $PH_{opt} = \{\}$, and then we use the attribute values of physical host to compare the physical hosts in the candidate set in pairs, and thus, we can obtain the optimal physical hosts and remove the dominated physical hosts. However, the selection of the first candidate element in the enumeration process will affect the efficiency of pruning, so we choose an optimal candidate as the first element for pairwise comparison. We define the grade g of the candidate physical host as follow:

$$g(phi) = \sum_{r=1}^{l} d_r(phi) \tag{26}$$

d_r is the value of r-th attribute of the physical host. It is easy to see that the candidate with the smallest grade must be optimal. That is, if $phi* \in TOTAL$, and $g(phi*) = \min_{phi \in TOTAL} g(phi)$. In this way, we consider $phi*$ as the optimal candidate in $TOTAL$. Let $phi*$ be the first element of the enumeration process, and continuously compare it with other physical hosts in the set through formulas (12) and (13). If $phi*$ is dominated by other elements, it will be eliminated and this comparison will terminate, and then it makes a comparison for the next round. When the last round of comparison is completed, we get the final set of non-dominated physical hosts PH_{opt}. At this point, the pruning process ends, and PH_{opt} is used as the initial state space for deep reinforcement learning.

Step5: Initialize the network parameters ω and θ of Critic and Actor, and copy ω and θ to the corresponding Target network parameters ω^* and θ^*, respectively. Initialize the experience pool R.

Step6: Perform the following iterative loop within a time step.

(1) Initialize the Uhlenbeck-Ornstein stochastic process (UO process for short), and the purpose is to introduce random noise;

(2) The Actor network obtains an action a_t according to the current strategy π and random UO noise. The expression of a_t is as follows:

$$a_t = \pi\left(s_t \mid \theta\right) + \mathbb{N}_t \tag{27}$$

(3) After the agent executes the action a_t, the environment returns the current system reward value r_t and the state s_{t+1} at the next moment.

(4) Store each state transition process (s_t, a_t, r_t, s_{t+1}) in the experience pool R as a data set for network training. In addition, N data of state transition process are randomly sampled in the experience pool as a mini-batch training set of Critic network and Actor network, the single state transition process data can be expressed as (s_t, a_t, r_t, s_{t+1}).

(5) To calculate the gradient of critical network: using the method similar to supervised learning, loss is defined as MSE (mean squared error). The calculation formula of loss value L of the critical network is as follows:

$$L = \frac{1}{N}\sum_i \left(y_i - Q\left(s_i, a_i \mid \omega\right)\right)^2 \tag{28}$$

where the y_i can be regarded as a ''label''.

$$y_i = r_i + \gamma Q'\left(s_{i+1}, \pi'\left(s_{i+1} \mid \theta^*\right) \mid \omega^*\right) \tag{29}$$

Based on the standard Back Propagation algorithm, the gradient $\nabla_\omega L$ of L relative to ω can be obtained.

(6) Update the critic network and use Adam Optimizer to update ω;

(7) Calculate the policy gradient of the Actor network, and use the mini-batch training set data according to the Monte Carlo method to estimate its expected value without bias. The expression is:

$$\nabla_\theta J \approx \frac{1}{N}\sum_i \left(\nabla_a Q(s, a \mid \omega)\big|_{s=s_i, a=\pi(s)} \cdot \nabla_\theta \pi(s \mid \theta)\big|_{s=s_i}\right) \tag{30}$$

(8) Update the Actor network and use Adam Optimizer to update θ.

(9) To update the target critical network and target actor network, the following formula is used to update ω^*, θ^*, where t is taken as 0.001 generally.

$$\omega^* = \tau\omega + (1-\tau)\omega^* \tag{31}$$
$$\theta^* = \tau\theta + (1-\tau)\theta^* \tag{32}$$

Step 7: the time step ends and the iteration loop terminates.

The pseudo-code of JCETD's algorithm is shown in Algorithm 1.

From the perspective of the macro framework, this paper fuses the idea of Deterministic Strategy Gradient Algorithm (DDPG) of Deep Reinforcement Learning on the basis of the ''cloud-edge'' architecture, which integrates the ''cloud-edge'' architecture and machine learning effectively. That is, ''cloud-edge'' is regarded as a joint datacenter, and a deterministic strategy gradient algorithm (DDPG) deployment strategy is built on this basis. Through continuous interaction with the surrounding environment, and using the

Algorithm 1 Algorithm JCETD

```
Input: The set of tasks $TK$, set of edge hosts $PHE$, set of cloud hosts $PHC$, $\omega$, $\theta$, related parameters of algorithm and "cloud-edge" physical hosts;
Output: final deployment solution vector $S$;
1:  The set of cloud hosts and the set of edge hosts are combined to form a new host set $TOTAL = \{PHC + PHE\}$;
2:  Initializes an empty set $PH_{opt} = \{\}$;
3:  The value of posterior probability of host set is calculated by formula (21);
4:  Compute the attribute values of memory and CPU through formulas (22) and (24);
5:  Sets the memory, CPU, and posterior probabilities as a set of attributes for the physical host, i.e., $V_{ph} = (v_1^m, v_2^c, v_3^p)$;
6:  Get the smallest grade candidate by formula (26) and make it the first element in the set;
7:  IsDominated(i)=false
8:  for $i = 1$ to $k$ do
9:      if IsDominated(i)==true then
10:         continue;
11:     for $j = i + 1$ to k do
12:         if $c_i \succ c_j$ then
13:             IsDominated(j)=true
14:         else
15:             if $c_j \succ c_i$ then
16:                 IsDominated(i) =True;
17:                 break;
18: if IsDominated(i)==false then
19:     $PH_{opt}.add(c_i)$;
20: Randomly initialize weights $\theta$, $\omega$, $\omega^* = \omega$, $\theta^* = \theta$, and initialize replay buffer $R$;
21: for $t = 1$ to $T$ do
22:     Select action $a_t$ according to the current policy and exploration noise;
23:     Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$;
24:     Store transition $(s_t, a_t, r_t, s_{t+1})$ in set $R$;
25:     Sample a random minibatch of $N$ transitions $(s_t, a_t, r_t, s_{t+1})$ from $R$
26:     Update the parameters of the Critic network $\omega$;
27:     Update the actor policy using the sampled policy gradient;
28:     Update the target networks $\omega^*$, $\theta^*$;
```

generated data to modify its own behavior, after several iterations of learning, the optimal deployment of the corresponding tasks can finally be completed. Through the exploration process of deep reinforcement learning, the traditional mode of task deployment is changed, and the shortcomings of the traditional intelligent algorithm, such as slow convergence speed and high complexity, are solved effectively. To a certain extent, high-quality QoS requirements of users and long-term load balancing of "cloud-edge" datacenter are realized.

From the micro point of view, in the initial stage of deep reinforcement learning, this paper uses the idea of pruning algorithm to realize the preprocessing process of the algorithm. Through the attributes of the physical host, the dominated physical host is pruned, and finally a non-dominated set of physical hosts is obtained. Pruning operation pruned the unpromising physical hosts, which not only reduces the state space of reinforcement learning, but also reduces the complexity of reinforcement learning, and makes the algorithm more robust in the running phase. At the same time, it is also an indirect way to achieve efficient task deployment and load balancing of "cloud-edge" datacenter. It can be seen through analyzing the proposed algorithm that the computational complexity of the algorithm is O(n^2).

V. PERFORMANCE EVALUATION AND ANALYSIS

In this part, we mainly evaluate the performance of JCETD algorithm, and verify the effectiveness of the algorithm. First, we selected the representative deployment methods FERPTS [20] and FIFO respectively to compare experiments with the JCETD algorithm proposed in this paper. Then, the metrics to verify the effectiveness of JCETD algorithm are given. Finally, through the load balancing degree, average response time, Makespan and throughput of the "cloud-edge" datacenter, the JCETD algorithm and the above deployment algorithm are simulated, and the final simulation results are displayed in the form of chart. In this paper, we use Python simulator on Tensorflow to implement the comparison experiment.

The experimental results show that, compared with other algorithms, JCETD reduces the total completion time and the average response time of tasks, and improves the throughput of the joint system greatly, and provides users with better service ability. On this basis, the JCETD proposed in this paper enables the joint "cloud-edge" datacenter to have a better load balancing effect.

FERPTS: It is a modern deployment algorithm, which realizes the perceptual decision of the current task through the access control strategy. It allocates the corresponding computing resources to realize the reasonable task scheduling, and reduces the complexity of algorithm effectively and minimizes the running time, and then it achieves energy consumption reasonably.

FIFO: the first-in-first-out strategy is to assign the physical host at the edge center closest to the user to the current tasks until there are no available computing resources. After that, the strategy sends all remaining tasks to the cloud computing center.

A. EXPERIMENTAL METRICS

The experiment metrics in this paper are mainly selected from the perspective of user experience and the performance of the "cloud-edge" joint system, including the average task response time, total completion time (Makespan), throughput and load balancing degree. Through the dual simulation experiment of users and the system, the effectiveness of JCETD algorithm proposed in this paper is finally verified.

The average response time is a performance metric from the perspective of users, which can be calculated by formula (1) to maximize users' QoS experience by reducing the average response time.

Total completion time (Makespan), that is, the time it takes to process all tasks. It is shown in formula (33):

$$Makespan = \max\left\{CT_{ij} = \frac{D_n}{f_n} | i=1, 2, \cdots, n; j=1, 2, \cdots, m\right\} \tag{33}$$

where CT_{ij} is the completion time of the i-th task executed on the j-th host, f_n is the computing ability allocated to the task by the current host node.

In this paper, the throughput of the "cloud-edge" system is taken as a metric to evaluate the external service performance of the system. By measuring the total amount of work or the total number of tasks that the entire system can complete in a given time, the more tasks are processed in unit time, the better the external service performance of the system is proved. The throughput formula is as follows:

$$F = VU * R/T \tag{34}$$

where F is throughput, VU is the number of users, R is number of task requests submitted by users, and T is time.

Load balancing degree is an important metric to evaluate the degree of load balancing of each node in the "cloud-edge" datacenter, and it is also one of the key embodiments of system performance. Through formula (6), we can get the load balancing degree of "cloud-edge" system, and the goal is that the smaller the value, the better.

The above four experimental metrics are used to verify the effectiveness of the algorithm. under the premise of satisfying the user's QoS experience, the "cloud-edge" system can show high-quality external service performance, and it will realize the long-term load balancing of the "cloud-edge" datacenter.

B. COMPARISON IN MAKESPAN

In the first set of experimental scenarios, we will compare the total completion time (Makespan), the smaller the value, the better the performance of system. Comparing the proposed JCETD algorithm in this paper with FERPTS and FIFO, the value of Makespan can be calculated by formula (33), and the changes of the Makespan of three algorithms can be reflected by increasing the number of tasks. As shown in Figure 4, the Makespan values of three algorithms continue to increase with the increase of the number of tasks. When the number of tasks is 100, the difference of Makespan value among the three is not too large. Among which, the FERPTS algorithm has the smallest Makespan value, followed by FIFO, and the JCETD has the maximum Makespan value. This situation occurs because when the number of tasks in the initial stage is small, the FERPTS algorithm can perceive the current task and make decisions

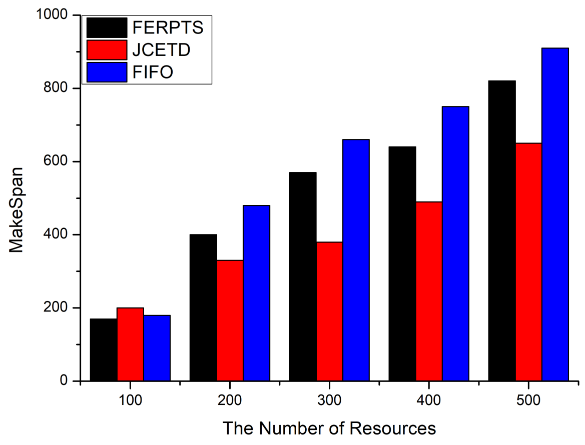


FIGURE 4. Comparison of FIFO, FERPTS, and JCETD on Makespan.

based on its own access control strategy, moreover, the corresponding computing resources are configured to realize reasonable task scheduling, so that tasks can be deployed rapidly when there are fewer tasks. FIFO rapidly deploys tasks to the nearest edge service node according to its own strategy, so it can complete the task calculation rapidly. In contrast, the proposed JCETD algorithm in this paper needs some calculation in the initial stage, and the data in the experience pool is too small to achieve the optimal deployment of tasks, as a result, the Makespan value is higher than the other two algorithms when the number of tasks is small. As the number of tasks increases, the Makespan value of the three algorithms continues to increase, the Makespan value of the JCETD is always far smaller than the other two algorithms, followed by FERPTS, and FIFO is the largest. The main reason is that, compared with the other two algorithms, the JCETD has the potential for long-term efficient deployment tasks. It combines the advantages of "cloud-edge" computing and deep reinforcement learning, and optimizes it into an intelligent cloud-edge deployment problem. Therefore, its Makespan value is better than the other two algorithms to a certain extent.

C. COMPARISON IN AVERAGE RESPONSE TIME

In the second set of experimental scenarios, we will compare the average response time of the three algorithms. The smaller the average response time is, the better the user experience will be. By comparing with the other two algorithms in average response time, the proposed JCETD algorithm in this paper has an excellent user experience. The average response time of each algorithm can be calculated by formula (1). As shown in Figure 5, as the number of tasks increases, the average response time of the three algorithms is increasing, and the average response time changes in a wavelike manner, only the JCETD has small fluctuations, and the average response time is always lower than the other two algorithms in the whole process. This is because the proposed JCETD algorithm in this paper simulates the process of deep reinforcement learning in the process of computing tasks,

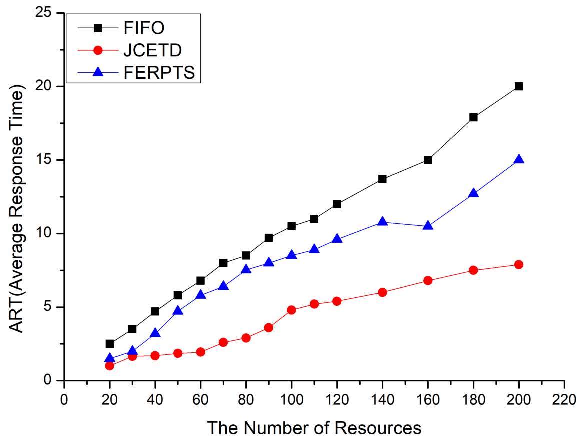


FIGURE 5. Comparison of FIFO, FERPTS, and JCETD on ART.

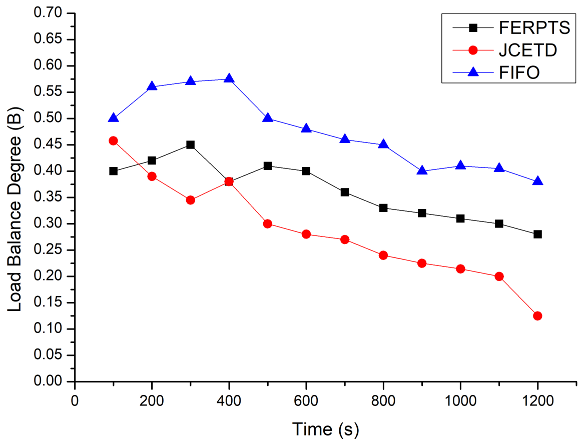


FIGURE 6. Comparison of FIFO, FERPTS, and JCETD on LBD.

and it uses the experience replay mechanism and deterministic strategy of the DDPG algorithm to improve its own behavior continuously, and the current experience will be directly applied to the next deployment, and the high-quality deployment and calculation will always be maintained in the whole process, thereby achieving the rapid deployment and response of tasks and improving the user experience.

D. COMPARISON IN LOAD BALANCING DEGREE

In the third set of experimental scenarios, we compare the JCETD with FIFO and FERPTS on the degree of load balancing. The real-time load balance degree of the system can be calculated from the formula (6). The smaller the value, the more balanced the load of the whole system. It can be seen from the Figure 6 that FERPTS has a better degree of load balancing at the beginning, the JCETD takes the second place and FIFO is the worst. With the increase of time, the load balancing degree of the three algorithms is decreasing, at t=400, the JCETD and FERPTS have the same value of load balancing degree. When $t > 400$, the value of load balancing degree of JCETD is always lower than the other two algorithms, followed by FERPTS, and FIFO is the worst. This is because the proposed JCETD algorithm in this paper has less previous experience data in the experience pool at the beginning, and the correlation between the before and after states is large relatively, which leads to the slow convergence of the algorithm and inability to complete the deployment and calculation of tasks with high-quality. Therefore, the effect of load balancing is not ideal in the initial stage. With the increase of time, compared with other algorithms, the JCETD shows its own advantages gradually. The algorithm reduces the state space of deep reinforcement learning to a certain extent through pruning preprocessing in the initial stage. In the process of the deep deterministic strategy gradient algorithm, tasks are continuously and reasonably deployed on the optimal physical host of the "cloud-edge" datacenter, so that the joint "cloud-edge" datacenter has a better effect of load balancing.

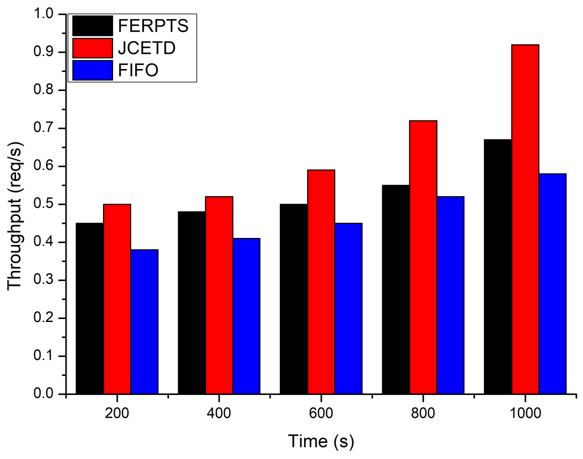


FIGURE 7. Comparison of FIFO, FERPTS, and JCETD on throughput.

E. COMPARISON IN THROUGHPUT

In the fourth set of experimental scenarios, the throughput is taken as the performance metric of the comparison experiment, and the throughput of the proposed JCETD algorithm is compared with the other two algorithms to verify the performance of the algorithm on the "cloud-edge" system. The throughput of the system can be calculated by formula (34). As shown in Figure 7, the throughput of the three algorithms shows an upward trend during the whole process. FIFO has a worse throughput in the initial stage, as time increases, the throughput of the system increases slowly. FERPTS is also increasing, but the rate of growth is not much different from FIFO. The throughput growth rate of the JCETD is higher than that of the other two algorithms significantly, and the real-time throughput is also the largest. This is because the proposed JCETD algorithm in this paper performs pruning preprocessing operations on the set of "cloud-edge" host in the initial stage, which reduces the state space of DDPG algorithm to a certain extent, and accelerates the execution efficiency of the algorithm, and improves the throughput of the system greatly.

TABLE 1. Comparison of Cloud-only, Edge-only, and JCETD on throughput, Makespan, and ART.

Metrics		Task Numbers			
		200	400	600	800
Cloud-only	Makespan	320	550	780	1100
	Thoughout	0.45	0.48	0.5	0.59
	ART	186.4169	238.31	344.9521	589.7327
Edge-only	Makespan	450	720	990	1300
	Thoughout	0.4	0.42	0.45	0.51
	ART	212.3526	365.5133	520.3805	990.4713
JCETD	Makespan	230	450	680	890
	Thoughout	0.58	0.62	0.88	0.92
	ART	127.65	204.2	303.2833	422.8

F. COMPARISON WITH UNILATERAL COMPUTING

In the fifth set of experimental scenarios, the joint "cloud-edge" computing proposed in this paper is compared with edge computing and cloud computing, through the comparison of Makespan, throughput and ART to verify the advantages of joint "cloud-edge" framework from different perspectives. As can be seen from table 1, with the increase of the number of tasks, three performance metrics of the joint "cloud-edge" system proposed in this paper are always better than cloud computing and edge computing. This is because the joint deployment of this paper is guided by deep reinforcement learning, looking for the optimal physical host for tasks constantly, which may be on the cloud or on the edge. Therefore, this kind of "cloud-edge" joint mode of deep reinforcement learning can not only maintain high-quality user experience, but also realize the load balancing of joint "cloud-edge" system. However, in single edge computing, the current computing resources may not be able to meet the task requests of a large number of users, which will degrade the quality of service and performance of the system. As for single cloud computing, as the amount of user access and data increase together with that the communication link to the cloud is too long, it may cause the cloud computing center to load unbalanced and fail to return the calculation results timely. The experimental results show that the performance of joint "cloud-edge" computing is better than that of cloud computing and edge computing.

VI. CONCLUSION AND FUTURE WORK

Based on the joint "cloud-edge" computing, this paper proposes a resource management and task deployment strategy JCETD based on pruning algorithm and deep reinforcement learning, and gives its main ideas, process implementation and evaluation. First of all, the resource management process adopts the idea of pruning algorithm, pruning the set of "cloud-edge" host by using the attribute values of physical host to get a non-dominated host set, which has the potential to realize tasks efficient deployment and load balancing of "cloud-edge" datacenter to a certain extent. Then, the deep reinforcement learning algorithm is integrated on the basis of the joint "cloud-edge" computing mode, and the set of "cloud-edge" host obtained in the resource preprocessing stage is used as the system state to simulate the deep reinforcement learning process. Through the continuous exploration and utilization of the environment, the efficient computing ability and load balancing under the "cloud-edge" architecture are finally realized. In order to evaluate the JCETD algorithm, a number of simulation experiments of performance metrics have been carried out to verify the efficiency of the proposed JCETD algorithm from the perspective of users and systems. The experimental results show that the JCETD not only enables efficient deployment and calculation of tasks, but also makes the overall load of the "cloud-edge" datacenter more balanced.

In the proposed JCETD algorithm, there are some open issues that need further study and empirical issues that need a lot of experiments to gradually obtain a better solution. Where the value of CPU weight and memory weight is an empirical issue, and multiple experiments are required to obtain the optimal value such that $\alpha + \beta = 1$. In this paper, all parameters are set to appropriate values.

In order to further improve the deployment performance and load balancing of joint "cloud-edge" computing, we plan to analyze and research from the perspective of "cloud-edge" intelligent task deployment in the next work. And on the premise of providing high-quality services for users, tasks can be deployed in edge computing center and cloud computing center reasonably, so as to maximize the benefits of "cloud-edge" datacenter and users. In the future work and experiments, the empirical and open questions raised in this paper will be further studied.

REFERENCES

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[2] Y. Chen, J. Huang, C. Lin, and X. Shen, "Multi-objective service composition with QoS dependencies," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 537–552, Apr. 2019.

[3] X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, and L. Qi, "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Gener. Comput. Syst.*, vol. 95, pp. 522–533, Jun. 2019.

[4] H. Tang, C. Li, J. Bai, J. Tang, and Y. Luo, "Dynamic resource allocation strategy for latency-critical and computation-intensive applications in cloud–edge environment," *Comput. Commun.*, vol. 134, pp. 70–82, Jan. 2019.

[5] Y. Huang, Y. Zhu, X. Fan, X. Ma, F. Wang, J. Liu, Z. Wang, and Y. Cui, "Task scheduling with optimized transmission time in collaborative cloud-edge learning," in *Proc. 27th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2018, pp. 1–9.

[6] Y. Sahni, J. Cao, and L. Yang, "Data-aware task allocation for achieving low latency in collaborative edge computing," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3512–3524, Apr. 2019.

[7] D. Schafer, J. Edinger, J. Eckrich, M. Breitbach, and C. Becker, "Hybrid task scheduling for mobile devices in edge and cloud environments," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2018, pp. 669–674.

[8] C. Li, C. Wang, and Y. Luo, "An efficient scheduling optimization strategy for improving consistency maintenance in edge cloud environment," *J. Supercomput.*, vol. 76, pp. 6941–6968, Sep. 2020.

[9] Y. Xie, Y. Zhu, Y. Wang, Y. Cheng, R. Xu, A. S. Sani, D. Yuan, and Y. Yang, "A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud–edge environment," *Future Gener. Comput. Syst.*, vol. 97, pp. 361–378, Aug. 2019.

[10] A. I. Orhean, F. Pop, and I. Raicu, "New scheduling approach using reinforcement learning for heterogeneous distributed systems," *J. Parallel Distrib. Comput.*, vol. 117, pp. 292–302, Jul. 2018.

[11] 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*, vol. 7, pp. 39974–39982, 2019.

[12] T. Dong, F. Xue, C. Xiao, and J. Li, "Task scheduling based on deep reinforcement learning in a cloud manufacturing environment," *Concurrency Comput., Pract. Exper.*, vol. 32, no. 11, Jun. 2020, Art. no. e5654.

[13] X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1133–1146, Jun. 2020.

[14] M. Cheng, J. Li, and S. Nazarian, "DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," in *Proc. 23rd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2018, pp. 129–134.

[15] C.-K. Tham and R. Chattopadhyay, "A load balancing scheme for sensing and analytics on a mobile edge computing network," in *Proc. IEEE 18th Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2017, pp. 1–9.

[16] A. Jyoti and M. Shrimali, "Dynamic provisioning of resources based on load balancing and service broker policy in cloud computing," *Cluster Comput.*, vol. 23, no. 1, pp. 377–395, Mar. 2020.

[17] G. Li, Y. Yao, J. Wu, X. Liu, X. Sheng, and Q. Lin, "A new load balancing strategy by task allocation in edge computing based on intermediary nodes," *EURASIP J. Wireless Commun. Netw.*, vol. 2020, no. 1, pp. 1–10, Dec. 2020.

[18] A. Ghasemi and A. T. Haghighat, "A multi-objective load balancing algorithm for virtual machine placement in cloud data centers based on machine learning," *Computing*, vol. 102, pp. 2049–2072, Sep. 2020.

[19] Z. Tong, H. Chen, X. Deng, K. Li, and K. Li, "A scheduling scheme in the cloud computing environment using deep Q-learning," *Inf. Sci.*, vol. 512, pp. 1170–1191, Feb. 2020.

[20] H. Li, J. Li, W. Yao, S. Nazarian, X. Lin, and Y. Wang, "Fast and energy-aware resource provisioning and task scheduling for cloud systems," in *Proc. 18th Int. Symp. Qual. Electron. Design (ISQED)*, Mar. 2017, pp. 174–179.

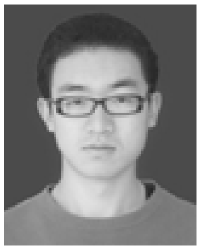
YUNMENG DONG received the M.S. degree in software engineering from Jilin University, China, in 2015, where he is currently pursuing the Ph.D. degree.

His current research interests include distributed systems, cloud computing, edge computing, big data and data mining, and machine learning.

GAOCHAO XU received the B.S., M.S., and Ph.D., degrees from the College of Computer Science and Technology, Jilin University, China, in 1988, 1991, and 1995, respectively. He is currently a Professor and a Ph.D. Supervisor with the College of Computer Science and Technology, Jilin University. As a person in charge or a principal participant, he has finished more than ten national, provincial, and ministerial-level research projects of China. His main research interests include distributed systems, grid computing, cloud computing, the Internet of Things, information security, software testing, and software reliability assessment.

MENG ZHANG received the Ph.D. degree in computer science from Jilin University, in 2003.

He is currently a Professor with the College of Computer Science and Technology, Jilin University. His research interests include distributed computing, string algorithms, and computational biology.



XIANGYU MENG received the Ph.D. degree from the College of Computer Science and Technology, Jilin University, Changchun, China, in 2017. He is currently a Postdoctoral Researcher with Jilin University. His research interests include data mining, network security, cloud computing, and mobile edge computing.