Resource Scheduling Based on Reinforcement Learning Based on Federated Learning

Yabin Wang*, Jing Yu
The Information System Engineering Important Laboratory, China.

* Corresponding author. Tel.: 15850770993; email: 517129269@qq.com
Manuscript submitted July 10, 2020; accepted October 18, 2020.
doi: 10.17706/jsw.16.1.39-45

Abstract: The emergence of edge computing makes up for the limited capacity of devices. By migrating intensive computing tasks from them to edge nodes (EN), we can save more energy while still maintaining the quality of service. Computing offload decision involves collaboration and complex resource management. It should be determined in real time according to dynamic workload and network environment. The simulation experiment method is used to maximize the long-term utility by deploying deep reinforcement learning agents on IOT devices and edge nodes, and the alliance learning is introduced to distribute the deep reinforcement learning agents. First, build the Internet of things system supporting edge computing, download the existing model from the edge node for training, and unload the intensive computing task to the edge node for training; upload the updated parameters to the edge node, and the edge node aggregates the parameters with the The model at the edge node can get a new model; the cloud can get a new model at the edge node and aggregate, and can also get updated parameters from the edge node to apply to the device.

Key words: Edge computing, resource scheduling, federated learning, reinforcement learning, distributed learning.

1. Introduction

Edge computing nodes located in multiple BS are always composed of high-performance servers or distributed data centers. BS edge computing nodes are distributed in different geographical areas, and can have different service areas, rental costs and available resources to meet the service needs of mobile computing nodes. Like thin clients, these computing nodes connect to edge computing nodes through mobile networks without accessing the remote cloud.

Cloud computing servers deploy wireless APS (access points) located in intelligent roadside units, such as street lights, intersections, stores or buildings. These nodes provide various local capabilities and services, including computing, networking, storage and application, for computing nodes in the coverage area through limited transmission coverage communication technology [1]. This type of node in the hierarchical edge computing environment is very important because it is closer to the user, has strong computing potential, and is well connected through high-speed local communication technology, which is conducive to providing low latency computing and rich computing resources [2].

Offloading services to edge computing nodes can extend the limited capabilities of intelligent computing nodes. BS edge computing node has large-scale coverage and high-performance computing. If applications require high computing power, it is more desirable to unload to the BS edge computing node. However, the weakness of BS edge computing node is that users may experience a long waiting time for data exchange with corresponding remote nodes through 3G / 4G access network. Long delays can damage interactive responses because humans are very sensitive to delays and jitters. For AP edge computing nodes, if too many computing nodes access the same wireless network at the same time, the scarce bandwidth may affect the application QoS [3]. For real-time applications, it will be more beneficial for users to execute services locally on self-organizing computing node nodes to avoid long communication time. Because the current computing node service is always complex, including multiple tasks with various requirements, we should consider communication delay, computing power and computing node mobility, and make unloading decision for each task.

Traditionally, the decision-making algorithm of unloading depends on human experience. Relevant knowledge is extracted by researchers, including system formula, algorithm solution and method optimization. However, manual knowledge extraction has three main limitations. First of all, environment including edge computing node and access network cannot have a comprehensive accurate, real-time evaluation. For example, for the computing node service of heterogeneous nodes, task execution progress is always abstracted as a queuing model, which is difficult to solve and ensure fairness; in addition, the goal of unloading decision model is almost non convex function, and Lagrange relaxation needs to deal with constraints or use some specific heuristic algorithms to find the approximate optimal solution [3]; finally, every computing node Edge calculation When the environment changes, the offload decision should be recalculated, which leads to more service delays and higher costs. Therefore, it is necessary to find an intelligent way to learn knowledge and provide foresight to unload decision. Fortunately, edge computing nodes have the advantage that they are deployed with network devices and can capture multidimensional data from the environment, including computing node behavior, task information and network status. It is feasible to use environmental data to continuously learn decision-making related knowledge and provide process policies.

Deep learning is a hot field of computer vision, speech recognition, natural language processing and computer network, and has made some remarkable achievements. It includes DNN (deep neural network), CNN (convolution neural network), RNN (regression neural network), RESNET, dense net, etc. [4] These different types of neural networks use multi-layer network structure and nonlinear transformation to build the underlying features, and formulate abstract and differentiated high-level expressions, so as to perceive and express things.

Service offload decision in edge computing environment is affected by many factors. It can capture nonconvex optimization problems through complex objective functions and constraints. The deep learning model divides the complex mapping problem into several embedded simple mappings described by multiple layers of the model. In the training process, the gradient based iterative optimization minimizes the loss function representing the approximation degree. After the supervised learning process based on the historical data of human markers or the solution of heuristic algorithm, the deep learning model can obtain the approximate optimal solution of the complex mapping problem [5]. When the model is deployed in the real word setting after training, the approximate optimal solution can be obtained independently and in real time according to the environment

Considering the change of future environment, deep learning method needs new tag data. Service offloading decisions for complex services with different requirements should have the ability of long-term programming and continuous learning [6]. Deep reinforcement learning (DRL) combines the perceptual ability of deep learning with the decision-making ability of reinforcement learning. Reinforcement learning is based on MDP (Markov decision process) theory, but it does not need to make state transition probability.

KD service unloading decision framework for computing node edge computing provides a unique platform for various services and all controlled computing nodes [8]. It includes the decision-making model of DRL algorithm for learning the knowledge of service unloading, and the observation function for obtaining the environmental data of computing node mobility and edge computing nodes. Because services have a variety of tasks, KD service offload decision framework reserves a basic decision model for each service. The basic decision model is trained on a powerful edge computing node (such as BS edge computing node), and then distributed to the computing node for the actual service unloading decision. After this training, the system learns the long-term optimal decision-making knowledge of data dependent tasks in complex services from the experience of historical unloading rewards. Since the DRL model includes incentives for each decision, the model can be continuously trained online while the service is running. During this process, parameters are sent from the computing node to the BS edge calculation node for periodic updating of the basic model.

The number of accessible edge computing nodes for a computing node is different, and it can change as the computing node moves [7]. The observation function sorts the accessible nodes of the three types of edge computing nodes according to the computing power, and uses the fixed number of each type of node as the candidate unloading destination.

The heterogeneous resources provided by computing node edge computing nodes are abstracted as multiple containers with specific functions and parameters. Multitasking in a complex application is modeled as a specific data flow graph and represented by a DAG [8]. Due to differences in edge computing nodes, including several cloudlets, BS with computing and storage capabilities, and adjacent computing nodes on the road, each task in the service has multiple unloading destinations. We analyze task delay, node performance and computing node mobility to determine computing node service delay. Symbols are defined in Table 1. (choose where to uninstall by calculating service latency? What's the use of computing service delay. What is the observation function? Is it the function to calculate the service delay.

This paper uses federated learning to train deep reinforcement learning agents, which are used for joint allocation of communication and computing resources. The experimental results show that the method is better than the centralized training method.

2. Federal Learning

When multiple data owners (e.g. enterprises) f_i, I = 1 When n wants to train machine learning model with their own data, the traditional way is to integrate the data to one side and use the data d = {Di, I = 1 N} After training, we get the model. However, the scheme is usually difficult to implement because of the legal issues such as privacy and data security [9]. To solve this problem, we propose federal learning. Federated learning means that the data owner can train the model and get the calculation process of the model M < fed without giving his own data d < I, and can ensure that the gap between the effect V < fed of the model M < fed and the effect V < sum of the model M < sum is small enough, that is, < V < fed-v < sum < δ, where δ is a positive value of any small value.

We will classify federated learning based on the distribution of island data. Considering that there are multiple data owners, the data set D - I held by each data owner can be represented by a matrix. Each row of the matrix represents a user, and each column represents a user characteristic. At the same time, some data sets may also contain label data. If we want to build a prediction model of user behavior, we must have label data. We can call user features X and label features y. For example, in the field of finance, the user's credit is the label y that needs to be predicted; in the field of marketing, the label is the user's purchase desire y; in the field of education, it is the degree of knowledge that students master, etc. User characteristic x and label y constitute complete training data (x, y). However, in reality, it is often encountered that the users of each data set are not identical, or the characteristics of users are not identical.

Specifically, taking federated learning with two data owners as an example, data distribution can be divided into the following three situations:

User characteristics of two data sets (x1, X2, ...) The overlap is large, while the user (U1, U2) The overlap is small; Users of two datasets (U1, U2) The overlap is large, while the user characteristics (x1, X2,...) The overlap is small; Users of two datasets (U1, U2) Overlap with user features (x1, X2,...) Some of them are relatively small. In order to cope with the above three data distribution, we divide federated learning into horizontal federated learning, vertical federated learning and federated transfer learning.

2.1. Horizontal Federal Learning

In the case that the user characteristics of two data sets overlap more and the user overlaps less, we segment the data set according to the horizontal (i.e. user dimension), and take out the part of data with the same user characteristics but different users for training. This method is called horizontal Federation learning. For example, there are two banks in different regions, and their user groups come from their respective regions, with very small intersection. However, their business is very similar, so the user characteristics of the records are the same. At this point, we can use horizontal federated learning to build a joint model. In 2017, Google put forward a joint data modling scheme for Android phone model update: when a single user uses Android phone, it constantly updates the model parameters locally and uploads the parameters to Android cloud, so that each data owner with the same feature dimension can establish a joint model.

2.2. Longitudinal Federal Learning

In the case of more user overlaps and less user features overlaps in the two data sets, we segment the data set according to the vertical (i.e. feature dimension), and take out the part of data with the same user and different user features for training. This method is called longitudinal Federation learning. For example, there are two different institutions, one is a bank in a certain place, the other is an e-commerce in the same place.

3. Approach

If you are using *Word,* use either the Microsoft Equation Editor or the *MathType* add-on (http://www.mathtype.com) for equations in your paper (Insert | Object | Create New | Microsoft Equation *or* MathType Equation). “Float over text” should *not* be selected.

3.1. Architecture

This paper uses the system model with energy collection in the Internet of things environment shown in for analysis. In this case, set D = {1,2 Devices in, D} are in set n = {1,2 In the service scope of edge node (EN) in n}, edge node provides communication and computing offload. Among them, Internet of things devices are information sensing devices that have certain computing power, can collect energy units from edge nodes, and can communicate with edge nodes; edge nodes are communication and computing servers that are close to users and deployed on the edge of the core network.

Each device can select an edge node from n to establish communication and unload intensive computing tasks, and be allocated w-frequency bandwidth. For quantitative analysis, the time range i is discretised into I-index time period with a duration of ζ (unit s). In the aspect of IOT devices, a typical IOT device is taken as a representative to illustrate the model. It can collect energy units from edge nodes and store them in an energy queue with maximum length Le max for wireless transmission and calculation. At the same time, it allows computing tasks to perform specific services, and these tasks form independent and identically distributed Bernoulli random variable sequences, with the common parameter γ t ∈ [0,1] in Le max. In addition, there is a local task queue with the maximum length of LT Max in the Internet of things device, which can maintain the unprocessed and unsuccessfully processed tasks in the way of first input first output (FIFO).

If a task is generated during I, the device's task arrival indicator is represented as at I = 1 during time period I, otherwise at I = 0. Computing tasks are modeled as (D, V), where d (unit byte) and V respectively represent the size of transmission data needed to unload tasks and the number of CPU cycles needed to process tasks. With regard to computing tasks, you can determine the computing tasks extracted from the task queue for local execution or unloading to the edge nodes on the Internet of things devices for processing. Back to the Internet of things device, it should make a joint operation (CI, EI) at the beginning of each time period I to make a decision: (1) whether the task is processed locally (CI = 0) or unloaded to the edge node (CI ∈ n), pay attention to CI ∈ {0} ∪ n; (2) how many energy units (EI ∈ ℕ +) should be allocated from the energy queue storing energy. In this case, when the calculation task is assigned to be processed locally in the allowed energy unit EI (if any), i.e. CI = 0, the CPU frequency fi assigned to the task can be the maximum limit of the model.

Table 1. Experiment Result

Latency		
Regular Approach(s)	**Federated learning(s)**	**Iterating Times**
1.1	1.5	1000
2.3	2.6	2000
1.4	1.9	3000
1.4	1.5	4000
2.9	2.7	5000

3.2. Model

In real-time changing real-world scenarios, energy queues and task queues should be paid special attention, because they represent computing resources and workload respectively. In this paper, Le I is used to represent the length of the energy queue inside the Internet of things devices at the beginning of time period I. Using the available energy unit provided by the energy queue, the achievable task execution delay (including communication and calculation) is the focus of the research. In addition to processing delay and transmission delay of tasks, handover delay is also considered.

Specifically, if task execution will be regarded as failure, in two cases, these tasks will remain in the task queue until successful execution: (1) IOT devices can not process computing tasks until a period of time has ended; (2) IOT devices choose to unload tasks to special Fixed edge node, and long-time transmission failure caused by insufficient energy allocation or poor radio channel quality. For the convenience of description, the length of the task queue can be dynamically calculated as: lt I + 1 = min {LT I - 1 {0 < Ti < ζ} + at I, lt Max} (9). Of course, if the task queue is full of waiting tasks, the newly generated tasks must be deleted, which should be avoided under ideal conditions. Then the number of calculation tasks discarded in a time period I is described as follows: ξ I = max {LT I - 1 {0 < Ti < ζ} + at I - LT max, 0} (10). In addition, since not every task can be successfully processed in a time period ζ, the unscheduled queuing delay of calculation tasks will be generated. The queuing delay in time period I is regarded as the length of internal task queue LT I, that is, ε I = LT I - 1 {Ti > 0} (11). Meanwhile, if the calculation task fails to execute, the corresponding penalty will be given: σ I = 1 {Ti > ζ} (12). In addition, when the IOT device decides to unload its calculation task to the edge node, it should compensate the occupied edge node. Such compensation is weighted by the time taken to receive and process task input data. $\pi \in \mathbb{R} +$ is defined as the price paid per unit time. The expression of payment can be written as: $\phi\ I = \pi \times (\min \{Ti,\ \zeta\} - HI) \times 1 \{CI \in n\}$ (13)

4. Experiment

The latency of the computing nodes is calculated according to different reinforcement learning iterating times. The results show that reinforcement learning with federated learning can achieve the similar latency with regular approach.

5. Conclusion

This paper studies the combination of deep reinforcement learning and federated learning in the Internet of things environment which supports edge computing. Deep learning in deep reinforcement learning has strong perception ability, while reinforcement learning has decision-making ability. At the same time, federated learning also guarantees the privacy of data. The computational offload experiment in this paper proves the validity of the algorithm offload strategy based on federated learning. In the future, we will use non independent and identically distributed data to study whether deep reinforcement learning has model compression technology, and how to arrange federated based learning training with more fine-grained.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Yabin Wang conducted the research, analyzed the data and wrote the paper. Chenhao Guo and Jing Yu approved the final version of the paper.

References

[1] Qiu, T., Zheng, K., Han, M., *et al*. (2017). A data-emergency-aware scheduling scheme for Internet of things in smart cities. IEEE Transactions on Industrial Informatics, *14(5)*, 2042-2051.

[2] Chen, X., Zhang, H., Wu C., *et al*. (2019). Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning. *IEEE Internet of Things Journal*, *6(3)*, 4005-4018.

[3] Zhao, Y., Li, M., Lai, L., *et al*. (2018). Federated learning with nonIID data.

[4] Valerio, L., Conti, M., & Passarella, A. (2018). Energy efficient distributed analytics at the edge of the network for IoT environments. *Pervasive and Mobile Computing*, *51*, 27-42.

[5] Li, X., Wang, X., Wan, P. J., *et al*. (2018). Hierarchical edge caching in device-to-device aided mobile networks: modeling, optimization, and design. *IEEE Journal on Selected Areas in Communications*, *36(8)*, 1768-1785.

[6] Chen, X., Jiao, L., Li, W., *et al*. (2015). Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ ACM Transactions on Networking*, *24(5)*, 2795-2808.

[7] Van, H. H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double Q-learning. *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

[8] Mnih, V., Kavukcuoglu, K., Silver, D. *et al*. (2015). Human-level control through deep reinforcement learning. *Nature*, *518(7540)*, 529-533.

[9] Burd, T. D., & Brodersen, R. W. (1996). Processor design for portable systems. *Journal of VLSI Signal Processing Systems for Signal*, 203-221.

Yabin Wang is a PhD graduate student from Nanjing University. His interest is cloud computing and edge computing. His research results on software test optimization were employed and reported by sere 2012. He was visited University of Dallas, visit and cooperate in papers and contribute to JSS. He cooperates with Baidu company to develop automation testing tools and obtain patents. He develops software performance testing platform for Jiangsu software product quality supervision and inspection center.