Cloud Resource Allocation Based on Deep Q-Learning Network

Zuocong Chen(✉)

College of Computer Science and Technology, Hainan Tropical Ocean University, Sanya 572022, China
Twsf2005@163.com

Abstract. The goal of resource allocation is to allocate the optimal resource to the candidate tasks, so that all the tasks can be finished in less time and the users' demands can be satisfied. To have better performance on the time span, CPU usage ratio and the load balance compared with existed methods, it proposes an allocation method that can map the tasks to the resources effectively, where an optimal allocation program will be generated. Firstly, the resource allocation model for tasks was proposed and the goal function was designed. Afterward, the deep Q-learning algorithm was defined to get an optimal allocation program, and the algorithm was analyzed in detail. The experiment was implemented to verify the proposed method. The simulation experiments prove that the method in this paper can effectively implement task scheduling, which has the advantages of high CPU utilization, short scheduling time and strong load balancing ability.

Keywords: Deep Q learning · Resource allocation · Reinforcement learning · Cloud computing

1 Introduction

Cloud computing (Cloud Computing) is a computing model that uses computing resources as a service and then provides them to users through the network, so that users can use computing resources on demand [1]. These resources mainly include data, software, hardware, and bandwidth. The services provided by cloud computing mainly include three categories: software as a service, platform as a service, and infrastructure as a service. Resources in cloud computing have the characteristics of geographical dispersion, heterogeneous resources, and dynamic changes [2]. Therefore, it is necessary to comprehensively consider these heterogeneous communication and computing resources when making offloading decisions for users' computing tasks, and optimize resource scheduling to ensure the user experience.

In recent years, some research work on tasks for resource allocation in cloud computing environment. There are many research results on offloading strategies and resource scheduling issues. Some studies have proposed a distributed task off-loading scheme based on game theory, which optimizes energy consumption and maximizes the game among users.

J. Zeng et al. (Eds.): ICPCSEE 2020, CCIS 1257, pp. 666–675, 2020.
https://doi.org/10.1007/978-981-15-7981-3_49

Resource scheduling is a core issue of cloud computing, which is directly related to the stability of cloud computing services, the efficiency of resource use, and user satisfaction. The research of cloud computing resource scheduling has very important meaning from the theory and technology itself. There have been many researches about how to achieve resource allocation in the cloudy environment.

Li et al. [3] apply the successive approximation method to study the generalization-form model and present a gradient-based resource allocation scheme to solve the approximation problems, to solve the problems of non-strict convexity and non-separation in generalization-form optimization problem. Jin et al. [4] propose an incentive-compatible auction mechanism (ICAM) for the resource trading between the mobile devices as service users (buyers) and cloudlets as service providers (sellers), so that an auction mechanism that holds certain desirable properties for the cloudlet scenario can be designed. Zhang et al. [5] combined energy-efficient user scheduling and power allocation schemes for the downlink NOMA heterogeneous network for perfect and imperfect CSI to study the trade-off between data rate performance and energy consumption in NOMA. Cánovas et al. [6] advocate an intelligent media distribution system architecture for delivering video streaming, where the network parameters such as bandwidth, jitter, delay and packet loss that influence the QoE of the end-users and the other parameters of the energy consumption such as CPU, RAM, temperature and number connected users that impact the result of the QoE are considered. Rezaee et al. [7] propose three quality management resources (human, organizational and technological) and eight different strategies related to quality, to explore the relationship between effective strategies and improve the quality and quality management of allocated resources for the successful implementation of the strategies. Lei et al. [8] propose a heterogeneous resource allocation approach, called skewness-avoidance multi-resource allocation (SAMR), to allocate resource according to diversified requirements on different types of resources, so alieving the problem that the existing homogeneous resource allocation mechanisms cause resource starvation. Ghribi et al. [9] mapped the allocation problem to the combination of an optimal allocation algorithm with a consolidation algorithm relying on migration of VMs at service departure, where the optimal allocation algorithm is solved as a problem with a minimum power consumption objective.

To have a better performance on the time span, CPU usage ratio and the load balance, we proposed an allocation method that can map the tasks to the resources effectively by deep learning, where an optimal allocation program can be generated. The combination of deep learning and resource allocation can provide a vivid prospect to improve the efficiency and generality.

2 The Goal for Resource Allocation

The goal of resource allocation is to allocate the optimal node to the task, so that all the tasks can be finished in less time. The factors considered in resource allocation in cloud computing mainly include the estimated execution time $time\_\cos t(r)$, the network bandwidth $band\_width(r)$ and the network delay $delay(r)$.

The estimated execution time $time\_\cos t(r)$ defined as the earliest finished time for t_i at the node tn_j, and it is denoted as Eq. (1)

$$time\_\cos t(r) = ECT_{ij} + start(tn_j) \tag{1}$$

where $start(tn_j)$ is represented as the earliest available time for the node, ECT_{ij} is the required execution time of task t_i on node tn_j.

The network bandwidth $band\_width(r)$ is the maximum bandwidth provided by the route r, and the maximum network delay generate in the route r can be denoted as $delay(r)$.

By combining the above, the goal function of the resource allocation can be denoted as Eq. (2).

$$F{=}\min \frac{atime\_\cos t(r) + bdelay(r)}{cband\_width(r)} \tag{2}$$

Satisfying:

$$\begin{cases} time\_\cos t(r) < TLT \\ band\_width(r) < ELT \\ delay(r) < DLT \end{cases} \tag{3}$$

where a, b and c are the weights for $time\_\cos t(r)$, $band\_width(r)$ and $delay(r)$ with the corresponding boundaries for them are TLT, ELT and DLT.

3 Allocation Algorithm Based on Improved Deep Q-Learning Network

3.1 MDP Model

Reinforcement learning [10] is an unsupervised machine learning method. It can get optimal policies by maximizing the long-term return. The interaction between agent and the environment in reinforcement learning can be denoted a Markov process. The MDP can be defined a four-tuple, namely, $M = <X, U, f, u>$, where $X \in \mathbb{R}^k$ is the state space, $U \in \mathbb{R}^n$ denotes the action space, $f : X \times U \times X \to [0, +\infty]$ is defined as the state transition function, $r : X \times U \times X \to R$ is the reward function. At the time t, the state is denoted as x_t, according to the environment dynamics $f(x_t, u_t, x_{t+1})$, the immediate reward is $r(x_t, u_t, x_{t+1})$. The Agent attempt to find the optimal policy though maximizing the accumulative rewards by maximizing the expected rewards.

Policy $h : X \to U$ is the mapping from the state space X to the action space U, where the mathematical set of h depends on specific domains. The goal of the agent is to find the optimal policy h^* that can maximize the cumulative rewards. The cumulative rewards are the sum or discounted sum of the received rewards, and here, we use the latter case.

Under the policy h, the value function $V^h : X \rightarrow \mathbb{R}$ denotes the expected cumulative rewards, which is shown as:

$$V^h(x) = \mathrm{E}_h\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | x = x_t\right\}, \tag{4}$$

where $\gamma \in [0, 1]$ represents the discount factor and x_t is the current state.

The optimal state-value function $V^*(x)$ is computed as:

$$V^*(x) = \max_h V(x), \forall x \in X. \tag{5}$$

Therefore, the optimal policy h^* in state x can be obtained by

$$h^*(x) = \arg\max_h V^h(x), \forall x \in X. \tag{6}$$

3.2 Deep Q-Learning Network

Q-Learning is a temporal difference algorithm proposed by Watkins in 1989. It is characteristic of the different of the applied policy in exploration and in evaluation. The exploration policy generally adopts $\varepsilon - greedy$, while the evaluation takes the *greedy* policy, the update equation for action value function can be denoted as:

$$Q(x_t) = Q(x_t) + \alpha(r_{t+1} + \gamma \max_{u_{t+1}} Q(x_{t+1}) - Q(x_t)) \tag{7}$$

The deep Q-Learning network (DQN) use the deep neural network to approximate the action value function. Directing using the samples collected from on-line are sequential to learn the value function will make the learning process unsteady. Therefore, the samples are firstly stored in an experience pool, and then randomly sample it in the learning process. There are also two networks, the one is the value function network and the other is the goal value function network. The goal value function network can be a copy from the value function network, and it is changed after several batches of samples are learned in the learning process.

In the cloud environment, there is no determined goal terminal node. Therefore, we need to get an episode of samples, where the number of the numbers is equal to the tasks. The immediate task in the allocation process will not get any reward until the final task. Therefore, the former allocated samples will get a reward of 0 and the final task will get a reward shown as Eq. (2).

At every time step, the agent tries to select the optimal action to execute, but it also with some probability to explore the action space. Namely, the candidate node. We take the $\varepsilon - greedy$ exploration policy, there the exploration probability of ε is set as the following equation:

$$\varepsilon = \varepsilon_0 * \left(1 - \left(\frac{i}{n}\right)^2\right) \tag{8}$$

where the initial value of ε is set to 0.9, n is the sum of the iterations and i is the current iteration.

The allocation process of task to the node by using improve DQN can be describes as:

(1) Initialize the action space and state space. The state space X is the history allocated tasks, and the action space U is the set of available nodes. The current episode is set to $t = 0$, and the number of the episodes is T;
(2) Initialize the action value network and the goal action value network. For the state-action pair $\forall(x, u) \in X \times U$ in the state action space, the state action value for any state action pair can be initialized as $Q_{0\mathrm{e}}(x, u) = 0$, and the goal action value function is set as $Q_{0g}(x, u) = Q_{0\mathrm{e}}(x, u) = 0$;
(3) Set the current state as $x = t_0$, x will be the first task in the task queue $T = t_0, t_1, \ldots t_n$;
(4) According to $\varepsilon - greedy$ exploration policy, the optimal action $a \in \{node_1, \ldots node_m\}$ will be selected as the probability $1 - \varepsilon$, and the other actions will be executed as the exploration probability ε;
(5) If the current task is the final task in the task queue, the reward r will be computed according to Eq. (2), else the reward will be $r = 0$. The next state is represented as $x = t_0 t_1$;
(6) Construct the current sample $(t_0, node_0, t_0 t_1, r)$;
(7) Update the value of ε according to $\varepsilon = \varepsilon_0 * \left(1 - \left(\frac{i}{n}\right)^2\right)$
(8) Transfer to the step (4), until the current state $x = t_0 t_1, \ldots, t_n$ and the number of the episodes equal to T;
(9) Put all the episodes to the experience pool.
(10) Randomly sample an episode e from the experience pool. From the final state $x = t_0 t_1, \ldots, t_n$ to the initial state of this episode e, all the samples are used to update the action value network. The action value network for evaluation can be updated as:

$$Q_e(x, u) = Q_e(x, u) + \alpha(r + \gamma \max_{u'} Q_g(x', u') - Q_e(x, u)) \tag{9}$$

(11) The network of goal action value network is copied form the evaluation action value network $Q_g(x, u) = Q_e(x, u)$
(12) Transfer to the step (9) until all the episodes are iterated.

After the follow algorithm has been implemented, the action value function can be learned. The action to execute at each time step can be shown as:

$$h^*(x) = \arg\max_a Q^h(x,a), \forall x \in X. \tag{10}$$

Therefore, the whole algorithm is defined:

Algorithm 1: DQN-based method for resource allocation

```
Initialize: $t=0$ , $\forall(x,u) \in X \times U$ , $Q_{0e}(x,u)=0$ , $Q_{0g}(x,u)=Q_{0e}(x,u)=0$
Input: $X$ ,$U$ ,$T$
Step1: Repeat T times:
Step2:     set $x=t_0$, $T=t_0,t_1,...t_n$;
Step3:     select $a \in \{node_1,...node_m\}$ with 1-$\varepsilon$, and the others with 1-$\varepsilon$;
Step4:     if current task is the last:
               compute reward $F=\min\frac{atime\_\cos t(r)+bdelay(r)}{cband\_width(r)}$
           else:
               reward=0
Step5:     update the next state $x=t_0t_1$
Step6:     update the current sample: $t_0, node_0, t_0t_1, r$ to pool;
Step7:     update $\varepsilon=\varepsilon_0*(1-(\frac{i}{n})^2)$
Step8:     update $Q_e(x,u)=Q_e(x,u)+\alpha(r+\gamma\max_{u'}Q_g(x',u')-Q_e(x,u))$;
Step9:     update the goal network $Q_g(x,u)=Q_e(x,u)$
Step10:end repeat
Output: obtain the optimal action $h^*(x)=\arg\max_a Q^h(x,a), \forall x \in X$.
```

4 Experiment Result

4.1 Experiment and Parameter Setting

The grid computing simulation tool Cloudsim is used to simulate the proposed method. There is three node clusters in the cloud environment are Cluster1, Cluster2 and Cluster

3. The number of initial nodes in the three clusters are 10, 12 and 15. The number of CPU is of 4. The numbers of the resources in three clusters are 40, 48and 60. The numbers of the users are 10, 8 and 9, and the corresponding tasks are 100, 300, 500.

The setting of the parameters are set as the exploration factor $\varepsilon = 0.01$, the discount factor is $\gamma = 0.8$, the learning rate $\alpha = 0.1$, the maximum number of episodes is T.

We will compare with the representative methods, such as Literature [8] and Literature [9] in the time for task scheduling, CPU usage ratio and the load balance.

4.2 The Time for Task Scheduling

The time span for executing all the tasks are simulated, and the compared result of the time for task scheduling is shown as Fig. 1.

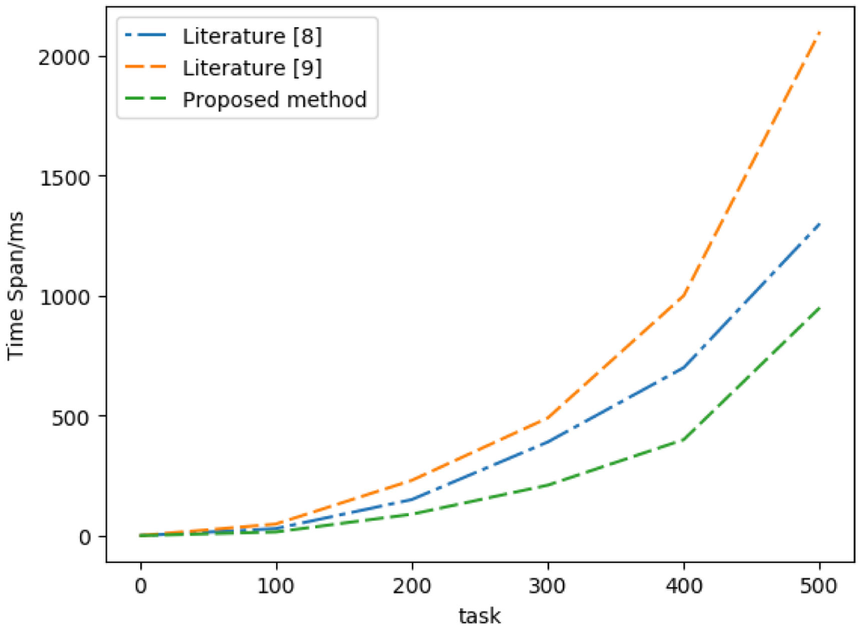

Fig. 1. The compared result of the time for task scheduling

It is easy to see from Fig. 1, the time span for the three methods are increasing in the whole simulation. However, the method of Literature [9] requires most time to execute all the tasks. Our method get the best performance in the whole experiment. The time span for Literature [8], Literature [9] and our method are about 1300 ms, 2100 ms and 850 ms, respectively. The deep reinforcement learning method can optimize the allocation program, so that it can get the far better performance on time span.

4.3 CPU Usage Ratio

The CPU usage ratio is a very import index to evaluate the reasonability of the resource allocation program. The results of the three methods after allocating all the tasks are shown in Fig. 2.

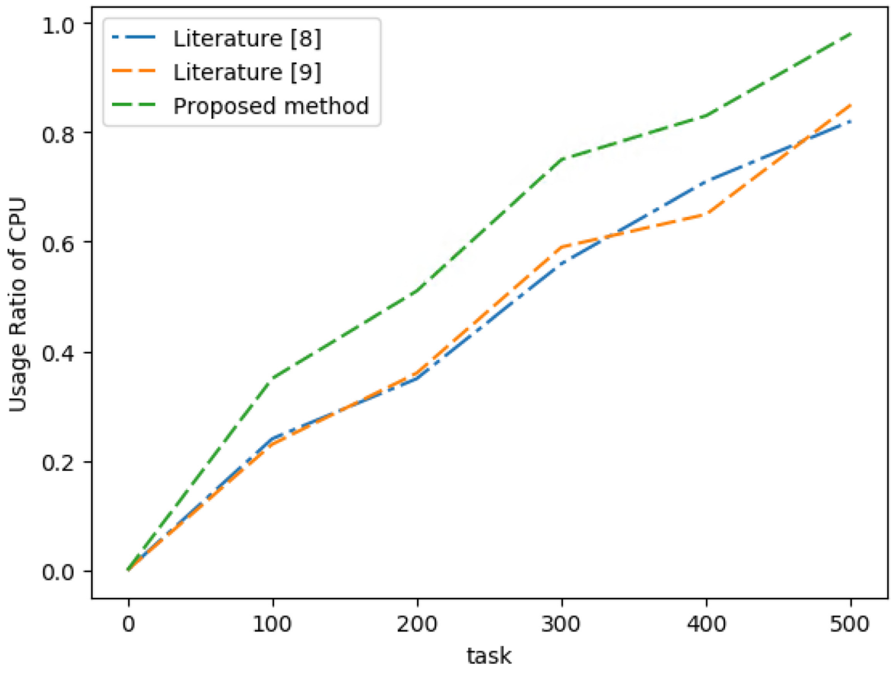

Fig. 2. Comparison of CPU usage ratio

From Fig. 2, we can evidently see that the CPU usage ratio for Literature [9] and Literature [10] are almost the same in the former 200 tasks. Afterward, the method in Literature [8] behaves a litter better than that of Literature [9]. But after 450 tasks are executed, the method in Literature [9] outperforms that of Literature [8] again. However, the CPU usage ratio is still higher than of those in Literature [8] and Literature [9], with a value about 0.98 after all tasks are finished.

4.4 Load Balance

The load balancing dispersion reflects the load balancing degree of the system, and its value can be calculated by the following formula:

$$\phi = \sqrt{\frac{\sum_{j=1}^{m} (LB_j - \bar{LB_j})^2}{m-1}} \tag{11}$$

where LB_j is the load balancing factor, and its value is the number of the tasks allocated to the node j, and $\bar{LB_j}$ is the average value of load balancing factor. The simulated result of the load balancing dispersion is shown as:

Though the load balancing dispersion obtained from three methods are increasing with the tasks, but it is clear that the prosed method has the smallest load balancing dispersion with the increase of the tasks. The method in Literature [8] is higher that of Literature [9] in the former 140 tasks. Afterward, the method in Literature [8] behave better until 350 tasks have been executed. The method in Literature [8] performs better between 350 tasks and 470 tasks. However, the result of the method in Literature [9] is still outperforms that of Literature [8] (Fig. 3).

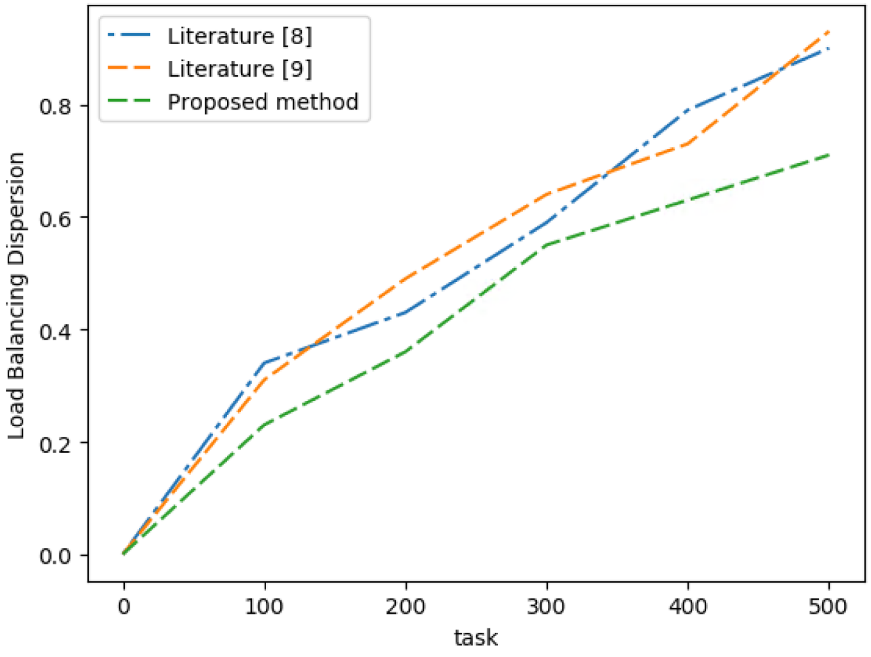

Fig. 3. Comparison of load balancing dispersion

5 Conclusion

Aiming at the shortcomings of long scheduling time and low CPU utilization of traditional scheduling algorithms in complex cloud environments, a cloud resource allocation algorithm based on deep reinforcement learning is proposed. Firstly, the resource allocation model for tasks is proposed. Afterward, the deep Q-learning algorithm is designed to get an optimal allocation program, and the algorithm is analyzed. Simulation experiments prove that the method in this paper can effectively implement task scheduling, which has the advantages of high CPU utilization, short scheduling time and strong load balancing ability, and strong feasibility.

Acknowledgments. This work was financially supported by Hainan Provincial Department of Science and Technology under Grant No. ZDKJ2016021.

References

1. Chen, X., Lin, J., Ma, Y., et al.: Self-adaptive resource allocation for cloud-based software services based on progressive QoS prediction model. Sci. China Inf. Sci. **62**, 1–3 (2019)
2. Sharma, V., Choudhary, G., You, I., et al.: Self-enforcing game theory-based resource allocation for LoRaWAN assisted public safety communications. J. Internet Technol. **19**, 515–530 (2018)
3. Li, S., Sun, W.: Utility maximization for resource allocation of migrating enterprise applications into the cloud. Enterp. Inf. Syst., 1–33 (2020)
4. Jin, A., Song, W., Zhuang, W.: Auction-based resource allocation for sharing cloudlets in mobile cloud computing. IEEE Trans. Emerg. Top. Comput. **6**, 45–57 (2018)
5. Zhang, H., Fang, F., Cheng, J.: Energy-efficient resource allocation in NOMA heterogeneous networks. IEEE Wireless Commun. **25**, 48–53 (2018)
6. Cánovas, A., Taha, M., Lloret, J., et al.: Smart resource allocation for improving QoE in IP Multimedia Subsystems. J. Netw. Comput. Appl. **104**, 107–116 (2018)
7. Rezaee, E., Pooya, A.: Resource allocation to strategies of quality management with FANP and Goal Programming approach. TQM J. **31**, 850–870 (2019)
8. Wei, L., Foh, C.H., He, B., et al.: Towards efficient resource allocation for heterogeneous workloads in IaaS clouds. IEEE Trans. Cloud Comput. **6**, 264–275 (2018)
9. Ghribi, C., Hadji, M., Zeghlache, D.: Energy efficient VM scheduling for cloud data centers: exact allocation and migration algorithms. In: IEEE/ACM International Symposium on Cluster, Cloud & Grid Computing. ACM (2013)
10. Zhong, S., Liu, Q., Zhang, Z., Fu, Q.: Efficient reinforcement learning in continuous state and action spaces with Dyna and policy approximation. Front. Comput. Sci. **13**(1), 106–126 (2019). https://doi.org/10.1007/s11704-017-6222-6
11. Chen, Z.: Method for extraction and fusion based on KL measure. In: Cheng, X., Jing, W., Song, X., Lu, Z. (eds.) ICPCSEE 2019. CCIS, vol. 1058, pp. 42–51. Springer, Singapore (2019). https://doi.org/10.1007/978-981-15-0118-0_4
12. Yang, L., Wen, K., Gao, Q., et al.: SVM based multi-label learning with missing labels forimage annotation. Pattern Recogn. **78**, 307–317 (2018)
13. Chen, Z.-c.: Task scheduling algorithm based on campus cloud platform. In: Sun, X., Pan, Z., Bertino, E. (eds.) ICAIS 2019. LNCS, vol. 11633, pp. 299–308. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24265-7_26
14. Cheng, J., Xu, R., Tang, X., et al.: An abnormal network flow feature sequence prediction approach for DDoS attacks detection in big data environment. Comput. Mater. Continua **55**, 95–119 (2018)
15. Ebrahimi, M.A., Khoshtaghaza, M.H., Minaei, S., et al.: Vision-based pest detection based on SVM classification method. Comput. Electron. Agric. **137**, 52–58 (2017)