An Implementation of Effective Server Resource Management Scheme Using Deep Reinforcement Learning

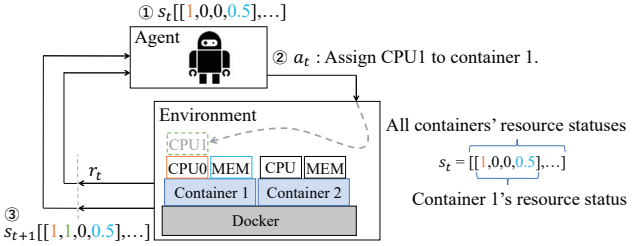
Toshiki KAWAKITA[†a)], ***Nonmember***, **Kimihiro MIZUTANI**[†,††], **Satoru KOBAYASHI**[†††], **Kensuke FUKUDA**[†††], ***and*** **Osamu AKASHI**[†††], ***Members***

1. Introduction

A container virtualization technology has been expanding rapidly in recent years, which can construct multiple application execution environments on a server. However, the server computing resources are shared among containers, and the performance of the containers may be degraded if appropriate computing resources are not allocated in busy case. In this study, to improve the performance degradation of the running application, we dynamically changed the resource allocation of the container while the container was running. In addition, we evaluate it on the real container virtualization environment (i.e., Docker Environment [1]) while a past literature [2] evaluated it by simulation.

2. Proposed Scheme

In this study, we use Docker as a container virtualization platform, and build multiple containers environment using it. For realizing effective resource allocation for the containers, we adopt a reinforcement learning scheme. In the reinforcement learning, a learning agent transits a state s_{t+1} from a state s_t by an action a_t. Then, the learning agent obtains the reward r_t and adds the reward to value $Q(s_t, a_t)$ in each state noting that $Q(s_t, a_t)$ is zero in the initial state. The learning agent finds the optimal action through the state transitions for gaining the maximum reward in each state. Here, we formulate the parameters of s_t, a_t, r_t. A state s_t is an array of resource statuses of containers at time t. A resource status of a container contains the allocated CPU and memory. If a container can use up to N CPU and M gigabyte memory, the length of a resource status is as $N + 1$, in detail, the i-th element ($1 \leq i \leq N$) of a resource status indicates i-th CPU is used (i.e., 1) or not (i.e., 0). The last element is the normalized value of the allocated memory between 0 and 1. For example, a resource status $[1, 0, 0, 0.5]$ indicates that a container is allocated the first CPU and about half of the memory size against the memory size limit noting that the second and third elements represent other CPU resources. In addition, s_t is an array of resource statuses if multiple containers exist. An agent takes an action a_t for a state transition at time t. Through an action a_t, the agent selects a target container and determines an action from $allocate/deallocate/keep$ for CPU or memory; $keep$ indicates no action for the container and the memory allocation is executed on 512MB units. A reward r_t is calculated from the degree of performance improvement after a resource allocation. Figure 1 shows the implementation of our proposed scheme. When an agent takes an action a_t in current state s_t, the container resource is changed. Then, the agent calculates the reward r_t by measuring performance improvement, and updates own $Q(s_t, a_t)$.



1. An agent observes current state s_t[[1,0,0,0.5],…].
2. The agent executes action a_t; assign CPU1 for container 1.
3. The agent obtains reward r_t ; performance improvement (i.e., RTT improvement). In addition, the agent observes state transition s_t to s_{t+1}.

Fig. 1 The implementation of our proposed scheme.

3. Evaluation and Conclusion

The proposed scheme is based on a deep reinforcement learning algorithm implemented with OpenAI GYM [3] and Chainer [4], which adopts 3 layer's fully connected neural network and took some learning parameters: discount factor γ and randomness value ϵ for action determination as 0.95 and 0.2. With this learning algorithm, our scheme controls 2 Docker containers with Docker API. Video streaming applications are launched in both containers and we loaded the websites with Apache Benchmark which keeps 100 http connections for the containers. For the reward calculation, we use the reciprocal value of RTT (Round Trip Time) as the degree of performance improvement. With these settings, our proposed scheme adjusts the number of CPUs and the amount of memory for improving RTT. After 1,000 epochs, we confirmed that our scheme kept to gain higher reward constantly and converged reward gain. As a result, we could conclude our scheme can realize effective resource allocation as the number of controlling becomes larger.

Acknowledgement

This work is partially supported by JSPS Kakenhi 20H04185.

References

[1] Docker, https://www.docker.com/
[2] Y. Xu et al, "A Survey on Resource Allocation for 5G Heterogeneous Networks: Current Research, Future Trends, and Challenges," in IEEE Communications Surveys & Tutorials, vol. 23, no. 2, pp. 668-695, 2021.
[3] OpenAI Gym, https://gym.openai.com/
[4] Chainer, https://tutorials.chainer.org/ja/

[†]Graduate School of Science and Engineering, Kindai University 3-4-1 Kowakae, Higashi Osaka, Osaka, Japan
[††]Cyber Informatics Research Institute, Kindai University, Same location as Kindai University.
[†††]National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo.
a) E-mail: 2233340448g@kindai.ac.jp

Copyright © 2022 The Institute of Electronics, Information and Communication Engineers

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