DRL-Cloud: Deep Reinforcement Learning-Based Resource Provisioning and Task Scheduling for Cloud Service Providers

Junquan Yu (junquany@usc.edu)

Rui Guo (ruiguo@usc.edu)

Zhendong Ju (zhendonj@usc.edu)

1, Project Description:

Cloud computing has become the standard paradigm in almost every industry due to booming growing size of data. Cloud Services Providers (CSPs) that owns data centers provides hardware resources and software resources to customers. Profit-driven CSPs charge users for service access, but also, they need to reduce energy consumption and electrical cost as much as possible. This is the reason they need to keep finding ways to minimize energy cost. Resource Provisioning and Task Scheduling are two essential parts in reducing energy. We adopt Deep Reinforcement Learning (DRL), an idea in Machine Learning field, to try to tackle these problems.

2, Challenge:

We face a bunch of challenges in this project.

At the very beginning, there are few models in the paper we need to realize. For example, the Energy Consumption Model and Realistic Pricing Model are the two models we need to implement. Fortunately, the data Google provides has already been classified into specific tasks that we no longer need to implement User Workload Model and Cloud Platform Model. And Energy Consumption Model and Realistic Pricing Model are quite simple to implement.

Secondly is Round Robin Method, Theoretically, Round Robin is a brute force method that it simply goes through all servers in sequence to check if the incoming task can be allocated in to a server, otherwise, reject it. Difficulty of Round Robin Method in our case is as followed: Normally, servers farm is processing incoming jobs and tasks in real-time. But our implementation is to process recorded tasks and jobs that already have timestamp and other information like CPU requested usage, etc. Meaning we need to simulate how time passes by using recorded data.

Secondly, we need to we are asked use C++ mandatorily. As far as we know, python is good at analyzing huge bunch of data, and it is extremely powerful in machine learning algorithm, such as what we will use in the future, DRL. In this case, we need to figure out which part we need C++ to implement.

Next step, all three members have zero idea about machine learning and just started learning how to use python. We need to get familiar with python and at the same time try to familiarize DRL implementation using python.

Finally, according to our initial idea, we first extract data from like .txt, .csv files using C++ and put this huge amount of data into our DQN which is implemented in python. This is also a big challenge, because the data we used to train DQN is of several Gigabytes.

DQN is short for Deep Q-learning Network. This neural network can adjust parameters and choose an action to achieve the biggest Q value. We now are still learning one famous implementation of DQN, how it is used for an explorer to find the treasure. We have 70% figured out the implementation of this instance, like how neural layer is built and how data is input and how to choose action based to previous action and corresponding reward it achieved.

3, Progress Report:

A, Round Robin

In order to solve the problem mentioned above. We place the tasks in time order following our scheme or so-called algorithm.

B, DQN

We now can use DQN to play a explorer game. Next step is to combine the DQN with idea in the paper.

4, Implementation:

A, Round Robin

In order to solve the problem I mentioned above. We place the tasks in time order following our scheme or so-called algorithm. Round Robin is a naive method used to schedule tasks. The basic idea is to assign tasks to server based on circular order. In our program, the core principle is the requested CPU of current running tasks shouldn't exceed 70% of the total CPU of the server, 100% of total RAM, meanwhile, meet the requested DDL. We let the task go from certain server and check whether it is suitable to assigned to the current server, if not, we continue do this. However, if we cannot find a suitable server after a round, we then push the task into the queue and handle it later. In this algorithm, the priority of the task is based on time. For a server, there are basically three types of situation when the task comes. The first situation is that adding new task won't exceed 70% total CPU that means the task can be executed at once. The second situation is that adding new task will make CPU used between 70% - 100% and RAM used below 100%. In this case, we consider putting it into a queue which means that it will be executed later. To choose which queue, we should also follow the core principle. If there is no queue meet the core principle, we go to the next server to do this again. The third situation is that adding the task

would make used CPU or used RAM exceed 100%. In this case, we simple go to the next server. Besides, we also implement the price model to calculate the cost of all task.

B, DQN

Before using DQN to achieve our goal, we need to learn how DQN works.

So, we found an online tutorial about how to play treasure-searching game using DQN idea. The explorer starts at origin, try to find the treasure hiding somewhere and at the same time try not to fall into hell. If he falls into one of the hells, game over and he needs to restart at origin.

The game works well, and I think we can modify it to our cloud computing project. We now have the structure, later we need to figure out in cloud computing how environment updates and what parameters we need to input into our Deep Q-learning Network.