Auto task assignment in MapReduce on Heterogeneous Environment

2230019 Kurumi Wakai

In recent years, with the explosive growth of data volume and the increasing diversity of computational tasks, data centers have evolved from homogeneous cluster architectures to heterogeneous ones. This transition is driven by the widespread adoption of specialized hardware components such as GPU, ARM CPU, and FPGA. However, MapReduce were originally developed with homogeneous environments. MapReduce has proven exceptionally effective in large-scale data processing on homogeneous clusters of commodity machines, its performance can degrade significantly in heterogeneous environments.

To address this challenge, a method known as MrHeter was proposed, aiming to optimize task assignment in MapReduce running on heterogeneous clusters. Specifically, MrHeter distributes map and reduce tasks according to the actual capabilities of each node. MrHeter can dramatically reduce the execution time of MapReduce jobs, achieving improvements of 30% to 70% over traditional, homogeneous scheduling approaches.

The MrHeter method solves the calculation models MrHeter-MS and MrHeter-R to find the optimal number of task assignments. To solve these computational models, we use the average execution time of the map and reduce tasks for each heterogeneous node, but this needed to be measured manually. The more heterogeneous nodes there are, the more time-consuming it is to measure the average execution time. For this reason, the MrHeter method had the disadvantage of being expensive to use.

In this study, we will conduct a demonstration experiment to perform task allocation using Optuna, a Python hyperparameter optimization library. Optuna takes in hyperparameters and an objective function, and outputs the hyperparameter values that minimize or maximize the objective function. Therefore, it is not necessary to measure things such as average execution time in advance, and labor savings can be achieved.

The advantage of using Optuna to automatically obtain parameters to reduce MapReduce execution time is that it requires less effort to things such as measure average execution time than the MrHeter method. The main contributions of this research are as follows. The first point is that, where in the past, the parameters for task allocation had to be measured manually, by using the Python hyperparameter optimization library Optuna, manual measurement becomes unnecessary. The reason that manual measurement is no longer necessary is that when you input the hyperparameters and objective function into Optuna, you can obtain the hyperparameters that minimize the

objective function, in other words, the execution time of MapReduce. The second reason is that manual measurement is no longer necessary, so the cost of measurement can be reduced, and a heterogeneous environment consisting of many different types of nodes can be constructed at low cost.

Many studies have been conducted to improve the performance of MapReduce in heterogeneous environments. LATE is the first study in the academic field to address this issue and has been very successful. LATE identifies tasks with slow execution speeds, called stragglers, and reduces the overall execution time of the job by speculatively executing them on other nodes. TPR, Dolly, and Mantri are also studies that focus on stragglers. In heterogeneous environments, predictable and stable computing power is the main cause of stragglers. However, most research does not investigate the causes of stragglers, but only tries to eliminate their effects, so the problem is not solved at its root. On the other hand, there are also studies that optimize jobs by improving the data imbalance between nodes in a cluster. For example, there are SkewReduce, Topcluster, and Libra. The aim is to distribute data according to the processing power of the nodes, but it is difficult to determine the appropriate coefficient because there is variation depending on the application. There have also been studies on improving MapReduce performance through configuration optimization. However, there are so many parameters that affect job performance that tuning them requires multiple tests, which is inefficient.

Unlike previous research, the MrHeter method identifies the causes of the performance degradation of MapReduce in heterogeneous environments and designs a computational model to solve the problem from the roots. One drawback of the MrHeter method is that it is costly to manually search for and input performance indicators and other data that are used as input variables for the computational model. In particular, measuring the average execution time of local map tasks and remote map tasks for each type of node is costly, so we propose an automated program. We also aim to reduce MapReduce execution time by using Optuna, a black-box optimization method, to skip the manual search for optimal parameters and use automatically derived parameters.

Local map task average execution time and Remote map task average execution time measurements are costly, so we have proposed a program to automate the measurement process. This program calculates the average task execution time for a specified server based on the container ID, which is unique for each job, from the resourcemanager log. The input information required is the path to the log file, the server name to be measured, and the container ID. When measuring the Local map task, the log of the Local map task execution is extracted from the resourcemanager log, and when

measuring the Remote map task, the log of the Remote map task execution is extracted from the resourcemanager log, and the path of these log files is specified as input information.

In this paragraph, we will explain how to execute MapReduce task assignment using Optuna. The flow of the code for parameter optimization using Optuna is as follows: First, the results of the previous execution using Optuna are evaluated and a parameter search is performed, and the parameters to be set for the next trial are determined. In the case of the first trial, only the parameter search is performed. Next, the parameters for each server (mapreduce.job.maps and mapreudce.job.reduces) derived by Optuna to minimize the execution time of the job are set in the Hadoop configuration file mapred-site.xml for each server, and the server is restarted to reflect the settings. This parameter setting and server restart process is performed on all servers. After that, the job to be minimized is executed and the execution time is measured. Optuna evaluates this execution time and specifies the parameters to be used in the next trial. The program ends when the number of times the job is executed reaches the specified number.

The experiment used a heterogeneous cluster environment. We used 14 servers, including one Opteron 2 (AMD Ryzen 7 3700X 8-Core Processor 4.4GHz), one P100 (Intel(R) Xeon(R) CPU E5-2637 v4@ 3.50GHz), and icl00-icl09 (Intel(R) Celeron(R) J4105 CPU @ 1.50GHz) 10 units, and icl10-icl11 (Intel(R) Celeron(R) J4005 CPU @ 2.00GHz) 2 units.

In terms of the cost of the procedure, the procedure for the MrHeter method takes 1+(number of server types * 2) steps, and the procedure for the proposed method takes 4 steps, so we can say that the labor saving of parameter determination using Optuna has been achieved. The more server types there are in a cluster environment, the greater the benefit of using the proposed method to save labor.

From the execution results of the wordcount job, it was found that the proposed method using Optuna was able to reduce the time by 16-50% compared to the default settings of Hadoop. The previous study, the MrHeter method, reduced the time by 4-48% compared to the default settings of Hadoop, so it was found that the proposed method had execution times that were equivalent to or shorter than the MrHeter method.

In this study, the execution time of MapReduce was used as a performance benchmark. In the future, it will be necessary to consider whether it is possible to use the parameters automatically derived by Optuna in situations where it is necessary to satisfy multiple criteria other than execution time, such as power consumption and memory usage.