Job level parallel search in software auto-tuning

Yuga Yajima
Kogakuin University, Japan
em23045@ns.kogakuin.ac.jp

Akihiro Fujii
Kogakuin University, Japan
fujii@cc.kogakuin.ac.jp

Teruo Tanaka
Kogakuin University, Japan
teru@cc.kogakuin.ac.jp

1 Introduction

Software auto-tuning (AT) is technology to improve performance by automatically controlling parameters which affect performance (performance parameters) in a program. In AT, to search for appropriate values for the performance parameters, the program is iteratively executed while fitting various values to the performance parameters. On the other hand, from 2000s on, machine learning field is attracting a lot of attention. In machine learning program, it is important to select proper hyperparameters. Therefore, hyperparameters of machine learning can be treated as performance parameters. However, machine learning programs take a long time per execution. Therefore, we have reduced the time by parallelizing the search [1].

By setting the number of parallels appropriately, the search can be properly conducted. The appropriate number of parallels depends on the amount of available computing resource, but if other tasks are being executed, the available computing resource is not constant. Therefore, it is difficult to set the appropriate number of parallels. In this study, we implemented a mechanism that dynamically recognizes the available computing resource on a supercomputer and sets the appropriate number of parallels.

2 Parallel searching using “Jobs”

There are three parallel levels of supercomputers: threads, processes, and jobs. Our searching is using job for parallelize. When user requests the supercomputer system to execute a program, user sets a job. Supercomputer system always checks usage of all users and if node is free, job is executed. A job is a level that requests the system to execute a program, and the number of parallels cannot usually be managed from the program.

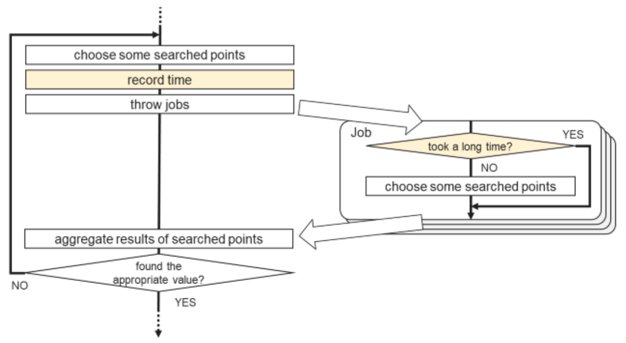
choose some searched points
record time
throw jobs
Job
took a long time?
YES
NO
choose some searched points
aggregate results of searched points
found the appropriate value?
NO
YES

Figure 1: Adding a mechanism to dynamically recognize the amount of computing resource

If the amount of computational resource is not taken into account, a part of computational resource is wasted because fractions of parallel execution are created. Therefore, we implemented a mechanism to dynamically recognize available computing resource from the program. Figure 1 shows the mechanism. If there is no room in the node, job execution is in pending. So, by measuring the time from setting job to start execution, we can recognize how crowded system is. It also allows for the appropriate number of parallel runs by cancelling the execution of jobs when there is no room in the computing resource.

3 Experiments and results

We experimented with the aforementioned search by applying it to a machine learning program. The execution environment was “Flow” Type II Subsystem in Nagoya University. Figure 2 shows the execution time of each job. The vertical axis represents the time elapsed since the first job started running. And each vertical line in the figure is the time when each job is being executed.

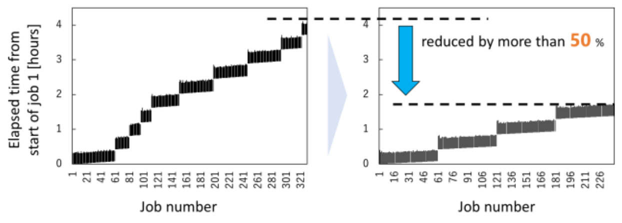

Figure 2: Reduction of job execution time

The conventional method often produced fractions of time that did not allow the appropriate number of parallels to be specified, but the addition of the new mechanism allowed the maximum number of parallels to be specified at all times. As a result, the search time was reduced by more than 50 percent while maintaining the final performance.

4 Conclusion

AT requires an enormous amount of time due to repeated trial and error, and we have made the search more efficient through parallelization. The maximum number of parallels was recognized by measuring the period from the job submission time to the start time, and a time reduction of more than 50 percent was achieved.

ACKNOWLEDGMENTS

This work is supported by “Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures” in Japan (Project ID: jh220044 and jh230045) and JSPS KAKENHI Grand Number JP 18K11340 and 23K11126.

REFERENCES

[1] Sorataro Fujika, Yuga Yajima, Teruo Tanaka, Akihiro Fujii, Yuka Kato, Satoshi Ohshima, and Takahiro Katagiri. Parallelization of Automatic Tuning for Hyperparameter Optimization of Pedestrian Route Prediction Applications using Machine Learning. HPC Asia 2023.