In-situ analysis with OpenMP task for leveraging unused core

Tatsuya Akazawa
akazawa@cspp.cc.u-tokyo.ac.jp
The University of Tokyo
Kashiwa, Chiba, Japan

Toshihiro Hanawa
hanawa@cc.u-tokyo.ac.jp
The University of Tokyo
Kashiwa, Chiba, Japan

Yohei Miki
ymiki@cc.u-tokyo.ac.jp
The University of Tokyo
Kashiwa, Chiba, Japan

1 INTRODUCTION

In recent years, the number of physical cores of the processor in HPC clusters has been increasing. However, the total performance might be reduced if all the cores are used. For this reason, there are cases where the number of cores used is intentionally limited to maximize the CPU's performance. The cores that are not being used then are called "unused cores." We plan to leverage the unused cores to support the main computation. Our goal is to create a framework that allows unused cores to support computation. The framework is expected to provide functions including auto-tuning dynamically and in-situ analysis and visualization technology. In this study, we aimed to demonstrate that OpenMP task pragma can be used to manage efficiently the main computation and analysis processes and reduce the execution time.

2 THE OPENMP TASK

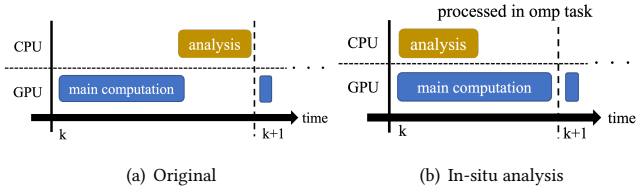
The code to which we have applied parallelization is Gravitational Oct-Tree code accelerated by HIerarchicaltime step Controlling codecite (GOTHIC)[1]. It is designed in such a way that the main computation, analysis, and output are processed sequentially. At first, this code computes the N-body on the GPU, then transfers the computation results from the GPU to the CPU. Next, analyze and output the results on the CPU. Figure 1 (a) shows the main computation and analysis flow of the gothic original.
We consider process the analysis in parallel because only one CPU is used during the main computation. To realize the In-situ analysis, we will use the OpenMP task pragma. it can be useful for parallelizing irregular algorithms such as pointer chasing. By using omp task, the computation results of the previous iteration can be analyzed in parallel with the main computation, which is expected to reduce the overall execution time.

3 EXPERIMENT AND EVALUATION

Figure 1 (b) shows the image of the parallel execution with omp task. The Core that was resting during the main computation on the GPU is assigned to analysis.
We compared the execution time between the case of parallelizing the main computaion and analysis and the case of sequential processing.

Table 1: Experiment Environment.

System	Wisteria/BDEC-01 Aquarius
CPU	Intel Xeon Platinum 8360Y
GPU	NVIDIA A100
Compiler	gcc-8.3.1



(a) Original (b) In-situ analysis

Figure 1: Image of parallel processing with omp task

For evaluation, we use the Wisteria/BDEC-01 system operated by Information Technology Center, the University of Tokyo. Table 1 shows the specifications of the system. Figure 2 shows the processing time per a loop. 1 is the sequential, 2 is the parallel execution, i.e. in-situ analysis. The execution time of in-situ analysis was reduced to about 52% of the time per loop compared to the time of the sequential processing.

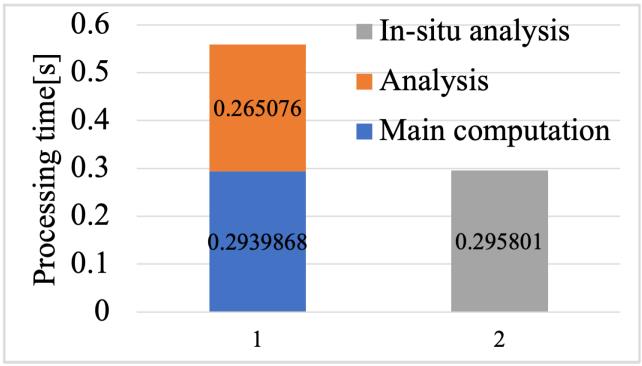


Figure 2: Experimental result

However, if the time for the main computation and the analysis differs greatly, the thread will wait longer. Therefore, it is necessary to consider a wait method that does not overload the CPU.

4 CONCLUSION AND FUTURE WORK

In this study, we parallelized the main computation and analysis using the OpenMP task pragma, and evaluated the execution time. As the future work, we would like to study issues such as how to wait for threads that do not overload the CPU.

REFERENCES

[1] Yohei Miki and Masayuki Uemura. 2017. GOTHIC: Gravitational oct-tree code accelerated by hierarchical time step controlling. *New Astronomy* 52 (2017), 65–81. https://doi.org/10.1016/j.newast.2016.10.007