DATE: 2018/1/9

Department of Electrical and Electronic Information Engineering		ID	143261
Name	Shintaro Nishiwaki		

Supervisor	Shuichi Ichikawa Naoki Fujieda

Abstract

Title	Performance evaluation of Xeon Phi by parallel SAT solver

The operation frequency of CPU has reached a peak by the limitation of the heat radiation performance as the increase of power consumption. Multi-core or many-core processors were designed to deal with this problem. Although the performance of each core is reduced, the performance of the whole system can be improved by parallel processing. In addition, a coprocessor is adopted to speed up some applications. Since the coprocessor has different architectures from CPU, they have suited and unsuited problems.

Intel Xeon Phi is a many-core coprocessor. Since there are many machines equipped with Xeon Phi in the list of TOP500, it is known that Xeon Phi systems are suited for the applications with high parallelism and little communication, such as High-performance Linpack used in TOP500.

This study aims at performance evaluation of Xeon Phi using Glucose syrup of parallel SAT solver which solves Satisfiability Problem (SAT) known as an NP complete problem. Our experiments adopted Xeon Phi 5110P coprocessor, Xeon Phi 7250 coprocessor and Xeon E5-2680v2 as a comparison target. Because Glucose syrup uses classes and structures, it can not be executed with Xeon 5110P as it is. Therefore the program was rewritten to be executed on Xeon Phi 5110P. Xeon Phi 7250 is processor version, where the original program is executed without modification. The problems used in evaluation were chosen from SAT Competition 2016 convention. Each problem was executed 20 times with 40, 60, 80 and 100 threads, and the time limit was set to 1000 seconds. For contrast, it was also executed 20 times with 20 threads on Xeon E5-2680 processor.

In comparison with Xeon, stable results were not obtained with both 5110 and 7250. However, it is still possible that Xeon Phi can solve more quickly than Xeon with an optimal specification of the solver. The number of problems solved often decreases when the number of threads increases. One of the reasons is the increase of synchronization overhead between threads. When executing with 40 and 60 threads, the number of synchronization between threads decreases within 1000 seconds and the number of active threads becomes the number of execution threads. But when executing with 80 and 100 threads, the number of active threads decreases.

To reduce the synchronization overhead and increase the number of active threads, learnt clauses shared with other threads ware limited. Nevertheless, better results than Xeon were not obtained, though it increased the number of active threads compared with Xeon. Since the method limits learnt clauses to be shared, the solution space might not be reduced properly. In a future work, it is necessary to improve the synchronization mechanism of learnt clauses.