DATE : 2016/01/08

Department of Computer Science and Engineering		ID	M113728
Name	Takahiro Douta		

Supervisor	Shuichi Ichikawa Naoki Fujieda

Abstract

Title	Preliminary performance evaluation of the Xeon Phi coprocessor

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 appeared 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.

Intel Xeon Phi is a many-core coprocessor. Xeon Phi has around 60 physical cores in a single chip, which is expected to accelerate the calculation that involves more parallelism than CPU. Xeon Phi is connected to the host processor via PCI Express. The communication time is required to exchange data with other processor cores and the host.

This study aims at preliminary performance evaluation for advanced performance tuning techniques suitable for Xeon Phi. Our experimental testbed includes a Xeon E5-2680v2 host processor and a Xeon Phi 5110P coprocessor. The performance of Xeon Phi with the various number of cores is measured in both the native mode and the offload mode. In the offload mode, the change of the quantity of data communication with the offloaded part of program is investigated. In this study, NAS Parallel Benchmarks (NPB), which is a benchmark for parallel computers developed in NASA Ames Research Center, was mainly used. This study examined four programs LU, FT, CG and MG, out of eight. This study used four sizes S, W, A and B size, out of seven.

Two or three methods were investigated in the offload mode. The first method (Method 1) offloads the most time-consuming function according to a profile. The second method (Method 2) offloads the whole of the benchmark body where the time is measured. The third method, only applicable to FT, offloads the part which are executed in parallel by OpenMP.

In Method 1, the number of offload becomes equivalent to the number of times the function is called. Therefore, communication became dominant and the execution time was not shortened by increasing the number of threads. In Method 2, communication time became minimal and the bottleneck resolved disappeared because the number of offload was reduced to one. The execution time became four times faster than Method 1 in LU (109 threads). In comparison with the host processor, the execution time of Method 2 with 109 threads in LU was smaller than that of the host processor with 4 threads. When comparing the native mode with the offload mode, the native mode is more effective than the offload mode if the problem size is small. Since the memory capacity of Xeon Phi is small the native mode cannot be applicable when the program size is large. It may be necessary to choose a proper mode and method according to the program to be executed.

This study examined the offload mode and the native mode. A future issue is the evaluation with the symmetric mode. In addition, evaluating with multiple nodes and Xeon Phi coprocessors and finding out the proper assignment of modes, methods, and threads are required.