Microsoft Research

Fast Multipole Method on the E2 Dynamic Multicore System

Naoya Maruyama
RIKEN (previously at Tokyo Institute of Technology, where the project was done)
nmaruyama@riken.jp
http://mt.aics.riken.jp/projects/fmm-on-e2.html

1. Project Goal

The E2 dynamic multicore architecture is an innovative microarchitecture that attempts to maximize the utilization and efficiency in a mixed application workload by adaptive composition and splitting of processor cores. Regular data-parallel computation can be run on a large number of small power-efficient cores, whereas irregular sequential code can be efficiently executed by a combined large scalar core, allowing more efficient usage of available chip resources while minimizing power consumption.

This project explores the effectiveness of the E2 architecture in particle simulation HPC code using the FMM algorithm, which simulates interactions of a large number of particles in multi-dimensional space by recursively constructing a tree of particles. We choose FMM because of its importance in scientific and engineering simulations and its multi-phase computation with different parallelism. The mixed workload of FMM can present an interesting opportunity for exploiting the core composition: Some phases may run more efficiently on a large number of small cores, whereas others run more efficiently on a small number of big cores.

2. Technical breakthrough

Summary: The achievements so far are limited to preliminary performance evaluation of one of the major computation phases of the FMM algorithm. Specifically, our experimental results indicate that core composition can improve the performance of a phase that computes multipole to local force update. We observe that composing four cores into one logical processor achieves approximately 10% of performance improvement compared to a single non-composed core.

Some more details: The composition uses the E2 runtime API and its performance is evaluated using the E2 timing simulator, which is being developed by a team led by Smith et al. The simulator allows for cycle-accurate timing simulations of up to 16 composable cores. We extended our evaluation workload with the composition API, and confirmed that it runs correctly on another E2 simulator that only allows for functional simulations without timing information. However, the same program currently fails on the timing simulator when executed with more than 8 composable cores, so for now we evaluated the performance effect by composition of 4 cores only.

In order to simplify the use of the composition API, we developed a very simple task-parallel API that allows for the user to specify a function to execution, the number of tasks, and the number of threads to use. When the API function is called, we use the E2 core composition API to change the number of (logical) cores, and create a thread on each core. The specified number of tasks is then executed by the threads in parallel. For functional portability, the same API is also implemented using the Pthreads API.

We have extended the original FMM program with the task-parallel API to exploit core composition. More specifically, the two major phases of the algorithm, P2P and M2L, are modified to use our task parallel API, and we confirmed that their performance is similar to the parallel version with OpenMP. The task-parallel API is designed to be a very simple C++ library, so we hope it should be easily usable by other applications.

Our evaluation is still ongoing. Smith's team is very actively developing a new version of the E2 timing simulator, which includes a lot of bug fixes with much faster simulation speeds, and is planned to become mature enough for our evaluation in a couple of weeks. Hopefully, we will be able to update the evaluation results using the new simulator soon with more cores and larger workloads.

3. Innovative Applications

The result can be also applied to other types of application workload where the amount of parallelism varies during the course of program execution. Especially, in high-performance computing, where parallel processing is norm, achieving good performance scalability is getting more difficult because of Amdahl's Law. For instance, reduction is one such example, which has limited parallelism at its last stages of computations. Techniques like adaptive core composition may be able to improve such limited-parallelism computations while simultaneously allowing for good performance scalability of computations with sufficient parallelism.

4. Collaboration with Microsoft Research

The project has been done in close collaboration with Dr. Aaron Smith of MSR Redmond (currently visiting University of Kyushu). The evaluation was done with the processor simulator developed by the team led by Smith.

5. Project Development

The study of adaptive HPC applications is ongoing with several government-funded projects.