An Investigation on the High-portability Software Framework for Implementing Large-scale FDTD Simulation with Many-core Processor Systems

Yukihisa Suzuki[1], Ryo Imai[1], and Kan Okubo[1]
(1) Tokyo Metropolitan University,
1-1, Minamiosawa, Hachioji, Tokyo, 192-0397, JAPAN,
e-mail: y_suzuki@tmu.ac.jp

1 Introduction

Finite-Difference Time-Domain (FDTD) schemes has suitable nature for massively parallel processing, which is executed on the many-core processors for example Graphics Processing Unit (GPU), Intel Many Integrated Core (MIC), and so on. By implementing FDTD schemes to those many core architectures appropriately, it achieves several to several tens of times higher calculation speed than by using conventional multi core CPU systems. However, it is difficult to implement FDTD schemes on the different types of many core architectures for the people who are not familiar to the art of parallel programming. The purpose of this study is to establish the software framework with high portability to implement the large-scale FDTD simulation on the many-core processors.

2 Concept of software framework

Figure 1 shows software stack diagram for the support tool to parallelize FDTD schemes we have proposed. In this framework, users only write FDTD scheme as function objects in C++ language, and parallelization is automatically performed by the support tool. In compiling FDTD simulation program, an execution file for NVIDIA GPU, Intel MIC, or multicore CPU is generated depending on the users' choice.

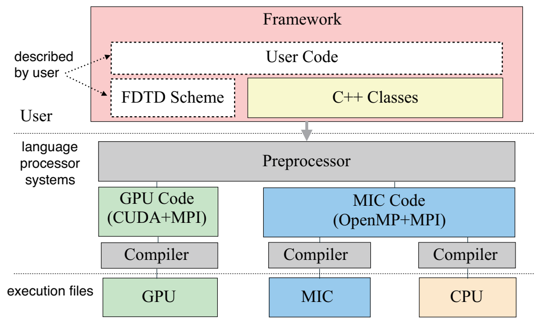

Figure 1. The stack diagram of software framework

3 Measurement of performances

We compared the performance of manual implementation with that of framework implementation for GPU (Nvidia Tesla k40), MIC (Intel Xeon Phi 7120P), and CPU (Dual Xeon E5-2620), respectively. Calculation scale is 512 x 512 x 512 cells. Compilers used in this investigation are CUDA7.5 for Nvidia GPU and Intel C/C++ Compiler 15.0 for MIC and CPU. In manual implementation, results of measured performances are 12.8, 39.1, and 65.1 GFlops for CPU, MIC, and GPU, respectively. On the other hand, when the framework implementation is applied, measured performances are 11.0, 39.6, and 67.3 GFlops for CPU, MIC, and GPU, respectively. Consequently, the software framework implementations we have developed indicate almost equivalent performance with the manual implementation. Furthermore, this framework reduces complicated procedures to parallelize FDTD schemes for each kind of many-core system, such as GPU, MIC, and so on.