Japan Geoscience Union Meeting 2010
(May 23-28 2010 at Makuhari, Chiba, Japan)

SSS018-04 Room: IC Time: May 25 11:30-11:45

Accelerating Simulation of Seismic Wave Propagation with Multi-GPU

Taro Okamoto[1*], Hiroshi Takenaka[2], Takeshi Nakamura[3]

[1]Tokyo Institute of Technology, [2]Kyushu University, [3]JAMSTEC

The GPGPU (General Purpose Graphics Processing Unit), or simply GPU, is a highly parallel processor that executes the arithmetic instructions on many (more than one hundred) processing units. With its extremely high computing performance which now reaches to nearly one TFLOPS (tela flops) in single precision arithmetic, it provides highly cost-effective solutions for scientific and engineering computing. This paper presents our approach to accelerate the simulation of seismic wave propagation by adopting the GPU computing.

We use NVIDIA TESLA S1070 equipped in the Global Scientific Information and Computing Center, Tokyo Institute of Technology. A single TESLA GPU has a peak arithmetic performance of nearly one TFLOPS. It has 4 GB (gigabyte) of main memory called "global memory". The data transfer rate of the global memory is relatively low compared to the small (but fast) internal memory in each processing unit. So we have to use the internal memory as a "software-managed cache memory" to achieve high computational performance.

We have developed a velocity-stress, staggered grid, three-dimensional finite-difference program for the GPU (TESLA) by using the CUDA programming tools. We reported a preliminary version of the program at the last SSJ meeting (2009): the preliminary version was based on a second-order scheme and periodic boundary condition was applied on all the domain boundaries for simplicity. We have extended the previous program in this paper: we now adopt the fourth-order scheme, the absorbing boundary condition (Cerjan et al. 1985) on the side and the bottom boundaries, and free-surface condition at the top boundary.

The amount of data transfered from the global memory to the internal memory of the processing unit for the fourth-order scheme is larger than that for the second-order scheme: the former is about 1.38 times larger than the latter. This is because of the extra ghost points required in adopting the fourth-order scheme. On the other hand, the amount of arithmetic operations for the fourth-order scheme is about 1.55 times larger than that of the second-order scheme. Therefore, by assuming a same data transfer rate for both cases, improvement in the computational performance is expected for the fourth-order scheme (by about 12 percent). Indeed, the arithmetic operations per second on a single GPU for the fourth-order scheme is about 47.6 GFLOPS (giga flops: for a case of 384x384x384 grid points) which is about 10 percent larger than that for the second-order scheme (43.1 GFLOPS). Note that these performance is much better than that of the conventional CPU: e.g., the arithmetic operations per second for a second-order finite-difference program is only about 1.5 GFLOPS on a single core of AMD Opteron (2.4 GHz).

We have also implemented Multi-GPU functionality in our program by using the MPI library. We apply one dimensional decomposition to the (fourth-order) finite-difference domain. We achieve 11 5 GFLOPS on four GPUs for a grid of 384x384x1536 (the last 1536 grid points are divided into four domain). But in other cases speedup is marginal probably because of the one-dimensional decomposition (e.g., 74 GFLOPS for a grid of 768x768x512 on four GPUs with 1D-decomposition

of the last 512 grid points). In the Multi-GPU computations, the data on the ghost-grids have to be transfered from the global memory of GPU to the host computer and exchanged between the host computers, and then copied back to the GPU because direct data exchange functionality is not available currently. Thus it is highly important to optimize the data transfer between the GPUs by using optimized domain decomposition scheme.

Keywords: GPU, wave propagation, finite-difference method, parallel computing