Japan Geoscience Union Meeting 2014
(28 April - 02 May 2014 at Pacifico YOKOHAMA, Kanagawa, Japan)

STT60-P01 Room:Poster Time:April 30 18:15-19:30

Parallel Performance of Particle Method in Many-Core System

FURUICHI, Mikito[1*] ; NISHIURA, Daisuke[1]

[1]Japan Agency for Marine-Earth Science and Technology

We present a computational performance of the smoothed particle hydrodynamics (SPH) simulation on three types of current shared-memory parallel computer devices: many integrated core (MIC: Intel Xeon Phi) processor, graphics processing units (GPU: Nvidia Geforce GTX Titan), and multi-core Central Processing Unit (CPU: Intel Xeon E5-2680 and Fujitsu SPARC64 processors). We are especially interested in the efficient shared-memory allocation methods with proper data access patterns on each chipset. We first introduce several parallel implementation techniques of SPH code for shared-memory system. Then they are examined on our target architectures to find the best algorithms for each processor unit. In addition, the computing and the power efficiency, which are increasingly important to compare multi device computer systems, are also examined for SPH calculation. In our bench mark test, GPU is found to mark the best arithmetic performance as the standalone device and the most efficient power consumption. The multi-core CPU shows the best computing efficiency. On the other hand, the computational speed by the MIC on Xeon Phi approached to that by two Xeon CPUs. This indicates that using MIC is attractive choice for the existing SPH codes parallelized by OpenMP to gain the computational acceleration by the many many-core processors.

Keywords: high-performance computing, many core, SPH, Parallel Computing, Performance analysis, Shared memory

Japan Geoscience Union Meeting 2014
(28 April - 02 May 2014 at Pacifico YOKOHAMA, Kanagawa, Japan)

STT60-P02 Room:Poster Time:April 30 18:15-19:30

Numerical investigation of efficient parallelization of large scale quasi-dynamic earthquake generation cycle simulation

HYODO, Mamoru[1*] ; ANDO, Kazuto[1] ; HIYOSHI, Yoshihisa[1] ; HORI, Takane[1]

[1]Japan Agency for Marine-Earth Science and Technology

Recently, Ohtani et al. (2011) applied an efficient compression method of full matrix to the problem of earthquake cycles. Since an original full matrix is approximated by a set of sub-matrices with the hierarchical structure, a compressed full matrix is called as H-matrices. By multiplying H-matrices to a column vector, they found that the required floating-point operation reduces to O(N)-O(NlogN) where N means the number of discretization of the model fault, though the original multiplication operation using the full matrix is $O(N^2)$. Owing to H-matrices, required memory and computation time are largely reduced for the problems of M8 earthquake cycles with $N=10^5$-10^6, and it enables us to execute capacity computing consisting of many earthquake scenarios using massively parallel computers like the K computer. However, for more realistic simulation with multi-scale earthquakes and their interactions, we must use 100 times larger N at least, and capability computing with massively parallel CPUs will be indispensable.

Following Ohtani et al. (2011), we have implemented MPI parallelization of earthquake cycle simulation with H-matrices. First, we applied a 1D division in the row direction to H-matrices. Then, each MPI process took charge of a divided row band region of H-matrices. Since the original H-matrices have a hierarchical structure consisting of many sub-matrices with large variation in size, it is difficult to divide all sub-matrices into MPI processes without overlapping through the 1D row division. Hence, we arrowed the overlapping sub-matrix to be calculated in both adjacent MPI processes for the simplification of parallelization. Then, through the simulation with $N=3\times10^5$, we confirmed a gradual speed-up with the increase of MPI processes up to about 100. However, further increases of MPI processes caused stagnation of speed-up, because the overlapping operation that is not reduced by the increase of MPI processes became dominant.

Accordingly, for more large-scale simulation with many MPI processes, it is necessary to reconsider the parallelization. At first, based on the current 1D division code, we limit the division number in the row direction so as not to increase the ratio of operations with respect to overlapping sub-matrices to the total operations. Then, each row region is divided into further small sub-regions in the column direction, thus we will apply the 2D division of H-matrices. In dividing a particular row region into further sub-regions, we introduce a reference size for the division, B (Block size). The column directional division of H-matrices requires data transfers between sub-matrices. Moreover, depending on the value of B, we also need data transfer inside the large sub-matrix. Though such 2D division increases the data transfers between neighboring MPI processes, the appropriate choices of B and division number in the column direction will realize the equal load balancing among MPI processes in row bands. Accordingly, parallel implementations with 2D division of H-matrices may overcome the overhead due to the increase of data communications.

As tentative results, for $N=1.3\times10^6$ problem, we implemented parallel calculations with both 1D and 2D divisions. Though the 1D parallelization cannot reduce the computational time with the increase of MPI processes, 2D parallelization successfully achieves speed-up with the increase of the number of parallelization. For the same number of MPI processes (1024 processes), the 2D implementation is more than two times faster than that of 1D.

In the presentation, we will show the more detail of our parallelization algorithm and its dependencies on the values of N, B, and division numbers.

Acknowledgement: Part of the results is obtained by using the K computer at the RIKEN Advanced Institute for Computational Science (Proposal number hp120278). We thank the Fujitsu tuning team for helping us to develop the parallelization of simulation codes.

Keywords: earthquake cycle, capability computing, parallel computing, H-matrices