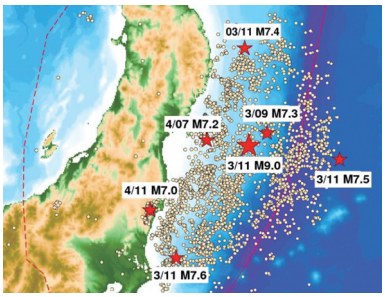
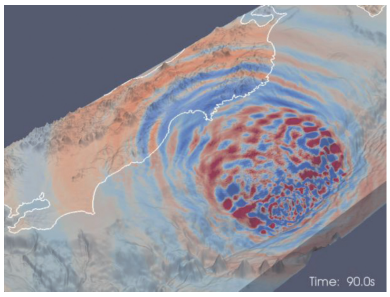
e-Science Journal
Global Scientific Information and Computing Center
Vol. 6

TSUBAME ESJ.





Turbulence Simulation Using 4096^3 Vortex Particles on 4096 GPUs

An Ultra-fast Computing Pipeline for Metagenome Analysis with Next-Generation DNA Sequencers

GPU-Accelerated Large-Scale Simulation of Seismic-Wave Propagation

http://www.gsic.titech.ac.jp/TSUBAME_ESJ

Turbulence Simulation Using 4096^3 Vortex Particles on 4096 GPUs

Rio Yokota* Lorena Barba** Tetsu Narumi*** Kenji Yasuoka****

*King Abdullah University of Science and Technology **Department of Mechanical Engineering, Boston University
***Department of Computer Science, The University of Electro-Communications ****Department of Mechanical Engineering, Keio University

The simulation of high Reynolds number turbulence is one of the most computationally demanding applications in high performance computing. In the present study, we perform the simulation of turbulence in a periodic box using 4096^3 vortex particles. The accuracy of the calculation is verified by comparing with a pseudo-spectral code. The relative parallel efficiency is compared between the FMM based vortex particle method, and the FFT based pseudo-spectral method on up to 4096 GPUs on the TSUBAME 2.0 system.

1 Introduction

Turbulence is ubiquitous in our daily lives and also appears in various engineering problems involving energy and environmental applications. The non-linearity of the governing equation and the large degrees of freedom in the system, make the simulation of turbulence one of the most computationally demanding problems in computational physics. Recent advances in the computational capability of supercomputers has made it possible to perform direct numerical simulations of the Navier-Stokes equation for reasonably high Reynolds numbers.

Although typical engineering applications for turbulent flows involve complex boundary conditions and require strong coupling with other domains of physics, there exist certain canonical flow fields where the physics of turbulence itself can be investigated without the complications of these extra constraints. The homogeneous isotropic turbulence under periodic boundary conditions is one such case, where the fundamental characteristics of high Reynolds number turbulence can be analyzed in isolation from mean shear and near wall effects.

The numerical method of preference for solving isotropic turbulence with periodic boundary conditions has been pseudo-spectral methods[1]. The largest of such calculations was performed with 4096^3 grid points at a Taylor-microscale Reynolds number of $\mathrm{Re}_\lambda=1130$ on the Earth Simulator back in 2002[2]. Their calculations revealed that the scaling of the high order turbulence statistics is significantly different at high Re_λ, while the local scaling exponents seemed universal[3]. Such conclusions would not have been attainable without the use of supercomputers and an idealized flow configuration, and symbolizes the importance of such efforts.

We would like to point out however, that the record for the largest turbulence calculation has not been broken for a decade[1], despite the 300 fold increase in LINPACK performance between the Earth Simulator and K computer. This can be either interpreted as; a) the difficultly to parallelize FFT on large distributed memory systems with torus networks, or b) the inability of LINPACK to serve as an indicator for the actual performance that can be achieved in real applications. The former interpretation prompts the development of alternative algorithms that have better scalability. The latter perception would most likely facilitate an early adaptation of an alternative benchmark, which could lead to better co-design between the software and hardware in high performance computing.

In the present study, we propose an alternative method for solving isotropic turbulence that can scale to the full node of TSUBAME 2.0. We caution the reader that ``scalability'' does not necessarily reflect the ``performance'' of the code – slow codes usually scale better because it is easier to hide the communication. By comparing both the time-to-solution and scalability of two different algorithms for the same application, we hope to provide data that supports decision making in the choice of algorithms and co-design of next generation software and hardware in high performance computing.

[1]There have been reports of spectral method calculations for 8192^3 grid points, but these are benchmarks that are not run for a significantly long time, and do not provide any data on the actual physics of turbulence

2 Vortex Method

There are a variety of methods that are referred to as ``vortex methods''[4]. Some are Lagrangian while others are semi-Lagrangian. Some are based on the vorticity-streamfunction formulation while other use the vorticity-velocity formulation. There are also many different ways to treat the viscous diffusion

in the Lagrangian frame of reference. A common advantage of these methods is the treatment of convection in the Lagrangian frame. This allows the use of larger time step sizes while eliminating numerical diffusion and dispersion effects. Furthermore, the vorticity-based formulation allows the points to be placed only in regions with non-zero vorticity. This leads to substantial savings in computational time and storage for external flows with unsteady vortex dynamics, such as vortex rings and trailing edge vortices.

In this work, we use the full Lagrangian vortex method with vorticity-velocity formulation[5]. The vorticity field is discretized by a superposition of Lagrangian vortex particles with Gaussian basis functions. In the vorticity-velocity formulation the velocity Poisson equation.

$$\nabla^2\mathbf{u}=-\nabla\times\boldsymbol{\omega}$$

is solved along with the vorticity equation

$$\frac{\partial\boldsymbol{\omega}}{\partial t}+\mathbf{u}\cdot\nabla\boldsymbol{\omega}=\boldsymbol{\omega}\cdot\nabla\mathbf{u}+\nu\nabla^2\boldsymbol{\omega}$$

The velocity Poisson equation becomes the Biot-Savart equation when formulated as an integral equation with the Green's function as the kernel. This becomes an N-body problem that can be solved using the fast multipole method (FMM) described in the next section. In the vorticity equation, the convection term is calculated by advecting the Lagrangian particles. The stretching term is calculated by substituting the Biot-Savart equation into the velocity, which results in another N-body problem. The diffusion term is calculated by increasing the size of the Gaussian basis function according to the analytical solution of the diffusion equation. In order to maintain sufficient overlap between the vortex particles, the coordinates are reinitialized every few steps, and the corresponding vortex strength of each particle is calculated to reproduce the vorticity field before the reinitialization. This is done using radial basis function interpolation with a Gaussian basis.

3 Fast Multipole Method

The N-body problem resulting from the Biot-Savart equation and the stretching term of the vorticity equation results in $O(N^2)$ operations if solved directly. We use the fast multipole method (FMM)[6] to bring the number of operations down to $O(N)$.

Since the Biot-Savart equation originates from a Poisson's equation, the FMM for Laplace kernels can be used here. The curl in the source term of the Biot-Savart equation and the gradient in the stretching term each require an extra a derivative to be applied to the Laplace kernel. However, since the multipole expansion and local expansion already consist of these derivative terms, it is trivial to extract this information without any extra computation.

3.1 Load Balancing

When parallelizing FMMs, the global nature of the N-body interaction and the dynamic nature of the particle distribution pose a unique challenge. Unlike grid based methods, the use of domain decomposition will not localize the data dependency –the halo region will be the entire domain but with hierarchically coarsened far regions. A common approach is to use a global tree structure to minimize the communication[7]. When the particle distribution is irregular and the tree structure is adaptive, equally partitioning the domain will result in load imbalance. Furthermore, the dynamic nature of the N-body simulation necessitates frequent repartitioning, so the overhead must be small.

A clever way to solve the load balancing issue is to record the load imbalance from the previous time step, and use it to repartition the tree for the present step. Such strategies were first introduced in the early 90's on both shared memory[8] and distributed memory architectures[9]. This idea of using information from the previous step to update the present step can be used with any existing partitioning scheme such as orthogonal recursive

bisection (ORB)[7], partitioning of Morton/Hilbert keys[9], and graph based partitioning[10]. The work/communication load from the previous step can be used as weights in any of these originally unweighted partitioning schemes. Our current FMM code supports both ORB and Morton key based partitioning schemes.

Once the domain is partitioned, each process will own a local portion of the global tree structure. It is then necessary to communicate the local essential tree (LET), which is a significantly smaller subset of the global tree that is required for the tree traversal on each process. The communication of the LET is global in nature, and is a potential bottleneck for the scalability of FMMs. A common technique in both treecodes[11] and FMMs[12] is to repeatedly exchange the LET on the other side of the bisector of the equivalent binary tree structure. This N-D hypercube type communication is extended in our code to work with number of processes that are not a power of two. We have further extended this framework to work with periodic boundary conditions.

3.2 Dual Tree Traversal

Both treecodes and FMMs build the same tree structure and partition it in the same way. The difference between the two methods lies in the way the tree is traversed. Treecodes loop over the target leaf cells and traverse the entire source tree for each target leaf cell. FMMs usually do not traverse the tree, but loop over all target cells and construct an explicit source cell list per target cell. For an adaptive tree, the construction of such source cell lists becomes complicated, especially if one were to introduce the concept of multipole acceptance criteria (MAC) into FMMs.

The dual tree traversal[13] can be used to simplify the construction of source cell lists in FMMs. It can be thought of as a generalization of U,V,W,X-lists[12] that also allows the use of MAC in FMMs. A conceptual diagram of the dual tree traversal is shown in Figure 1. Two trees are traversed simultaneously, one for the target and one for the source. The MAC is applied for each pair during the dual traversal. If the MAC is satisfied, the multipole-to-local translation is calculated and the pair is removed from the traversal queue. If the MAC is not satisfied, the traversal continues until both trees reach their leaf cells, at which point the direct interaction will be calculated between all particles in the cells. In other words, the dual tree traversal provides a simple bookkeeping strategy for constructing a MAC-based interaction stencil that is mutually exclusive at each level, by letting the target cells inherit a unique list of source cells from their parents.

3.3 Auto-tuning on CPU/GPU

In hierarchical N-body methods such as treecodes and FMMs, the dominant part of the calculation is the particle-particle interaction (P2P) kernel for the near field and the multipole-to-local (M2L) or multipole-to-particle (M2P) kernel for the far field.
The ratio between these near field and far field calculations must be optimized by properly selecting the number of particles per leaf cell during the tree construction. This is a non-trivial task on heterogeneous architectures, since each kernel will accelerate at different rates on GPUs[14]. There are also a wide variety of mathematical formulations for each of these kernels, some of which are better suited for a particular architecture.

A simple strategy for selecting the optimal kernel on any given architecture is to measure the time of dry kernel calls at the beginning of the simulation, and to use those timings to select the fastest option during the tree traversal. By providing the option to select between P2P (direct), M2P (treecode), and M2L (FMM) kernels, the resulting algorithm will always run faster than any of these methods alone, on any architecture, for any problem size. Our tests show that this is indeed the case[15].

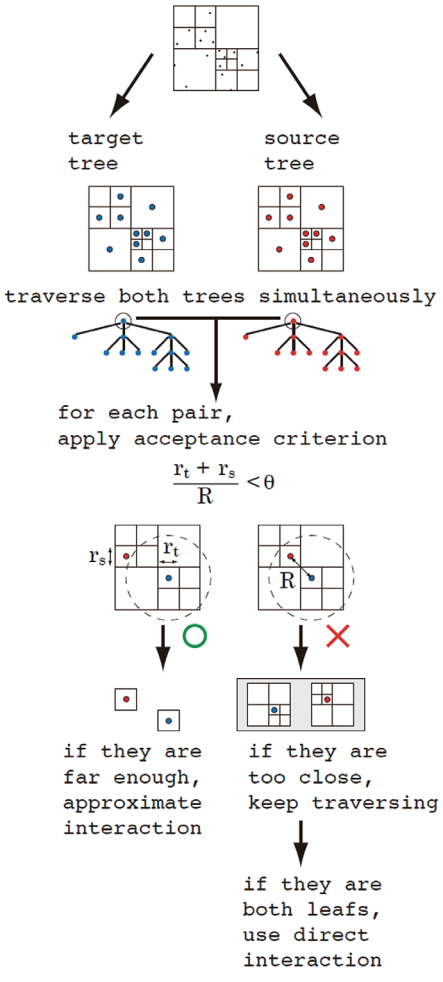


Figure 1 Diagram of the dual tree traversal

4 Scalability Results

The present simulations are run on the TSUBAME 2.0 system, which has 1408 nodes equipped with two Intel Xeon5670 CPUs, three NVIDIA M2050 GPUs, 54 GB of RAM, and 120 GB of local SSD storage. The inter-node connection is a full bisection/non-blocking fat-tree, where each node has two QDR Infiniband links.

The breakdown of the weak scaling runs for $N=4096^2$ particles per GPU is shown in Figure 2. We assign one MPI process per GPU so the number of processes is equal to the number of GPUs. The execution time is for the combined Biot-Savart + stretching kernel. In the legend, "P2P evaluation" is the evaluation time of the P2P kernel in the near field, "FMM evaluation" is the sum of all other FMM kernels, "MPI communication" is the total time spent on all MPI communications except the ones in the tree construction, "GPU buffering" is the time spent on reordering/buffering the data and sending it to the GPU, and "Tree construction" is the parallel construction/ partitioning of the global tree. The local portion of the P2P evaluation is overlapped with the MPI communication. In Figure 2 the MPI communication time is subtracted from the P2P evaluation time so that the total height of the bar reflects the actual wall clock time. The largest run with $N=4096^3$ vortex particles on 4096 GPUs results in a sustained performance of 1.01 Pflop/s.

We see that the GPU buffering is taking a significant amount of time, but we have found this to be necessary in order to achieve high efficiency in the P2P evaluation and FMM evaluation on GPUs. However, this part of the calculation scales perfectly, and does not affect the scalability of the FMM. The parts that do affect the scalability are the tree construction and MPI communication. Actually, the tree construction involves MPI communication for the partitioning so it is the efficiency of MPI communication that determines the weak scalability.

It may be worth noting that the N-D hypercube type communication of the LET (described in section 3.1) turned out to be slower than a simple call to MPI_Alltoallv for sending the entire LET at once. Therefore, the results shown in Figure 2 are with MPI_Alltoallv and not the hypercube communication.

We performed a similar weak scalability test for the spectral method, and compared the parallel efficiency with the FMM in Figure 3. The spectral method was calculated on the CPU because we did not have access to a GPU based FFT code that gave us significant acceleration over FFTW (when host-device data transfer is included) at the time of the comparison. The parallel efficiency of the FMM was 74 % at 4096 GPUs, while the parallel efficiency of the spectral method was 14 % on the same number of CPUs. The time-to-solution was approximately 100s for both the FMM and spectral method.

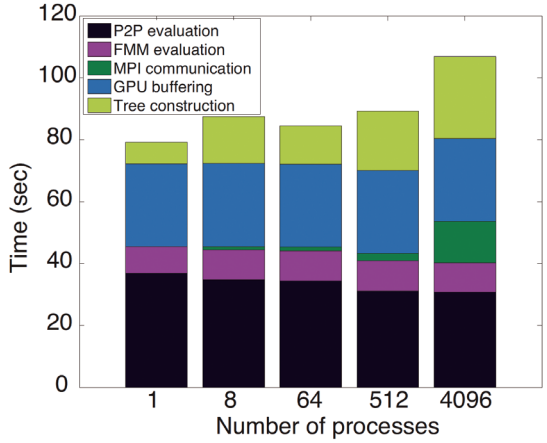


Figure 2 Weak scaling of FMM on up to 4096 GPUs for $N=4096^2$ particles per GPU.

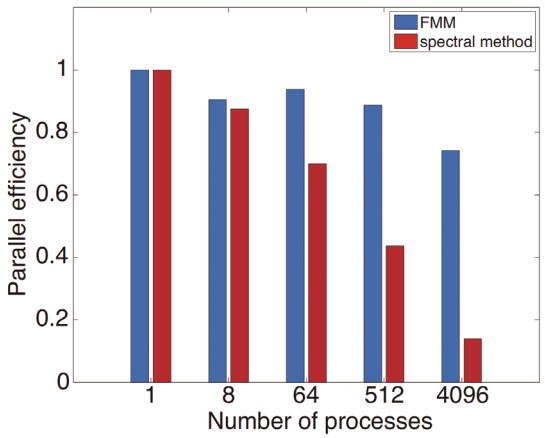


Figure 3 Parallel efficiency of FMM and spectral method on up to 4096 processes.

5 Isotropic Turbulence

The calculation of isotropic turbulence for Re_λ=500 was performed by the FMM based vortex particle method and FFT based pseudo-spectral method. The calculation domain was a periodic box of $[-\pi,\pi]^3$ and the number of particles/grid-points was 4096^3. For the FMM, we used 27^3 periodic images to approximate the periodic boundary condition, and the order of multipole expansion was set to p=14. The initial condition was generated using the spectral code, by generating a velocity field in wave space with a prescribed energy spectrum and random phases. This initial velocity field is used directly as an input to the pseudo-spectral method. For the vortex method, the velocity field is first converted to real space and the strength of the vortex particles are adjusted to reproduce this velocity field. The vortex particles are initially positioned at the cell centers (same location as the pressure on a staggered grid), and radial basis function interpolation is used to determine the strength of each particle. The initial core radius of the vortex particles are set to $\sigma=\Delta x$ so that the overlap ratio is 1.

Figure 4 shows an isosurface of the second invariant of the velocity gradient tensor at time t/T=2, where T is the large eddy turn-over time. Many small vortex structres can be observed, but no large scale coherent structures are seen at this early stage of the calculaiton. The duration of the run was severely limited by the window of access to the full node of TSUBAME 2.0. Therefore, the results of the present turbulence simulation are only preliminary, and are only interesting in terms of demonstrating the scalability and validity of FMM based vortex methods on large scale GPU systems, which is orthogonal to the issue of what new physics of turbulence was obtained.

The kinetic energy spectra for the pseudo-spectral method and vortex method at t/T=2 are compared in Figure 5. There is a small deviation at the high end of the wave numbers, but otherwise the results agree quantitatively. The improvement over similar comparisons in the past[5][16] was achieved by using a higher order of multipole expansion (p=14), using more periodic images (27^3), more frequent reinitialization (every 5 steps), higher accuracy in the RBF iteration (relative $|L|^2$=1e-5), and of course much more vortex particles. We have confirmed that all the conditions mentioned above must be satisfied in order to achieve the high accuracy in vortex methods that we show in Figure 5.

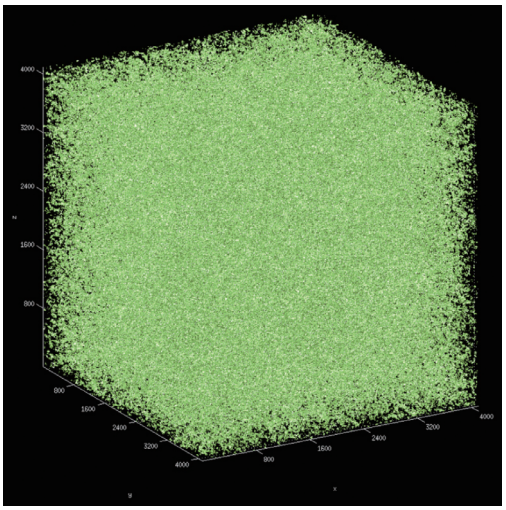


Figure 4 Isosurface of the second invariant of the velocity gradient tensor from the vortex method simulation

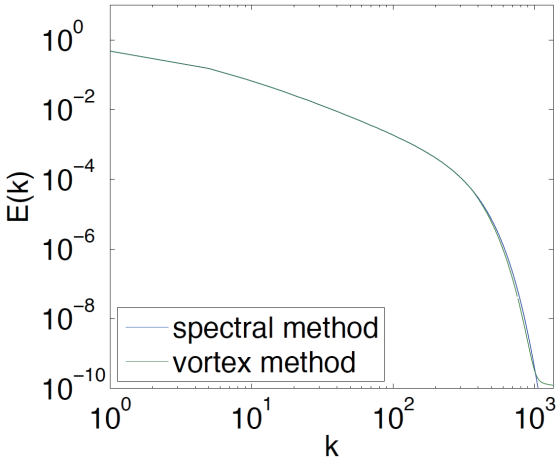


Figure 5 Energy spectra of the pseudo-spectral spectral method and vortex method at t/T=2

Conclusions 6

We compared the scalability of a FMM based vortex method against a FFT based pseudo-spectral method for the simulation of isotropic turbulence with 4096^3 particles/grid points on 4096 CPU/GPUs on TSUBAME 2.0. The FMM based calculation achieved 74 % parallel efficiency, while the FFT based method had 14 % parallel efficiency on 4096 processes. The calculation using 4096^3 vortex particles reached 1.01 Pflop/s, which is a new record for FMMs as far as the authors are aware. The comparison of the results between the vortex method and pseudo-spectral method showed quantitative agreement in the energy spectra. However, due to the limited availability of the full node of TSUBAME 2.0, we were not able to compare high order turbulence statistics between the vortex method and spectral method. Future generation supercomputers may offer us the opportunity to make use of our FMM based turbulence solver.

Acknowledgements

Computing time in the TSUBAME 2.0 system was made possible by the Grand Challenge Program of TSUBAME. The current work was partially supported by the Core Research for the Evolution Science and Technology (CREST) of the Japan Science and Technology Corporation (JST). LAB acknowledges partial support from NSF grant OCI-0946441, ONR award #N00014-11-1-0356, and Boston University College of Engineering.

References

[1] S. A. Orszag and G. S. J. Patterson: Numerical Simulation of Three Dimensional Homogeneous Isotropic Turbulence. Phys. Rev. Lett. Vol. 28, pp. 76-79 (1972)

[2] M. Yokokawa, K. Itakura, A. Uno, T. Ishihara, and Y. Kaneda: 16.4-Tflops Direct Numerical Simulation of Turbulence by a Fourier Spectral Method on the Earth Simulator, Proc. ACM/IEEE Supercomput. Conf., Washington DC, pp. 50-66 (2002)

[3] T. Ishihara, T., Gotoh, and Y. Kaneda: Study of High-Reynolds Number Isotropic Turbulence by Direct Numerical Simulation, Annu. Rev. Fluid Mech., Vol. 41, pp. 165-180 (2009)

[4] G. H. Cottet and P. Koumoutsakos: Vortex Methods, Cambridge University Press (2000)

[5] R. Yokota, T. K. Sheel, and S. Obi: Calculation of Isotropic Turbulence Using a Pure Lagrangian Vortex Method, J. Comput. Phys., Vol. 226, pp. 1589-1606 (2007)

[6] H. Cheng, L. Greengard, and V. Rokhlin: A Fast Adaptive Multipole Algorithm in Three Dimensions, Vol. 155, pp. 468-498 (1999)

[7] M. S. Warren and J. K. Salmon: Astrophysical N-body Simulation Using Hierarchical Tree Data Structures, Proc. ACM/IEEE Supercomput. Conf., pp. 570-576 (1992)

[8] J. P. Singh, C. Holt, J. L. Hennessy, and A. Gupta: A Parallel Adaptive Fast Multipole Method, Proc. ACM/IEEE Supercomput. Conf., pp.54-65 (1993)

[9] M. S. Warren and J. K. Salmon: A Parallel Hashed Oct-tree N-body Algorithm. Proc. ACM/IEEE Conf. Supercomput., pp. 12-21 (1993)

[10] S.-H. Teng. Provably Good Partitioning and Load Balancing Algorithms for Parallel Adaptive N-body Simulation. SIAM J. Sci. Comput., Vol. 19, pp. 635-656 (1998)

[11] T. Hamada, K. Nitadori, K. Benkrid, Y. Ohno, G. Morimoto, T. Masada, Y. Shibata, K. Oguri, and M. Taiji: A Novel Multiple-walk Parallel Algorithm for the Barnes-Hut Treecode on GPUs – Towards Cost Effective, High Performance N-body Simulation, Comput. Sci. – Res. Dev., Vol. 24, pp. 21-31 (2009)

[12] I. Lashuk, A. Chandramowlishwaran, H. Langston, T.-A. Nguyen, R. Sampath, A. Shringarpure, R. Vuduc, L. Ying, D. Zorin, and G. Biros: A Massively Parallel Adaptive Fast Multipole Method on Heterogeneous Architectures, Proc. Conf. High Perform. Comput. Netw. Stor. Anal., (2009)

[13] W. Dehnen: A Hierarchical O(N) Force Calculation Algorithm, J. Comput. Phys., Vol. 179, pp. 27-42 (2002)

[14] R. Yokota and L. A. Barba: Treecode and Fast Multipole Method for N-body Simulation with CUDA, GPU Computing Gems, Morgan Kaufmann (2011)

[15] R. Yokota and L. A. Barba: Hierarchical N-body Simulations with Auto-tuning for Heterogenous Systems. Comput. Sci. Eng., Vol. 14, pp. 30-39 (2012)

[16] R. Yokota, T. Narumi, R. Sakamaki, S. Kameoka, S. Obi, and K. Yasuoka: Fast Multipole Methods on a Cluster of GPUs for the Meshless Simulation of Turbulence. Comput. Phys. Comm., Vol. 180, pp. 2066-2078 (2009)

An Ultra-fast Computing Pipeline for Metagenome Analysis with Next-Generation DNA Sequencers

Takashi Ishida* Shuji Suzuki* Yutaka Akiyama*
*Department of Computer Science, Tokyo Institute of Technology

Metagenome analysis is useful for not only understanding symbiotic systems but also watching environment pollutions. However, metagenome analysis requires sensitive sequence homology searches which require large computation time and it is thus a bottleneck in current metagenome analysis based on the data from the latest DNA sequencers generally called a next-generation sequencer. To solve the problem, we developed a faster homology search program based on GPU-calculation and a large-scale computing pipeline for metagenome analysis on TSUBAME2.

1 Introduction

Metagenome analysis is the study of the genomes of uncultured microbes obtained directly from microbial communities in their natural habitats such as soils, seas, and human bodies. The analysis is useful for not only understanding symbiotic systems but also watching environment pollutions[1]. These days, the latest DNA sequencers, generally called next-generation sequencer, became to produce huge amount of genomic data in a short time and it is expected that metagenomic researches are promoted based on such huge genomic data.

However, metagenome analysis requires comparisons of sequence data obtained from a sequencer with sequence data of remote homologues in databases. Because, current databases do not include sequence data for most of microbes in the sample. Therefore, sensitive sequence homology search processes are required in metagenome analysis. Unfortunately, this process needs large computation time and is thus a bottleneck in current metagenome analysis based on the data from a next-generation sequencer[2].

Here, we developed a large-scale automated computing pipeline for analyzing huge amount of metagenomic data obtained from a next-generation sequencer. This pipeline enables us to analyze metagenomic data from a next generation sequencer in real time by utilizing huge computational resources on TSUBAME2. Also, we developed a fast homology search program based on GPU-calculation, which is named GHOSTM[3]. This program has sufficient search sensitivity for metagenomic analysis and is much faster than BLASTX program[4], which is generally used for previous metagenomic researches. By using these new program and pipeline, we can process metagenome information obtained from a single run of a next generation sequencer in a few hours. We believe our pipeline will accelerate metagenome analyses with next-generation sequencers.

2 Metagenome mapping

Current sequencers produce only information in short fragments, whose lengths range between 50 and 700 base-pair (bp). Thus, mapping or assembling processes are required for obtaining meaningful results from the output.

Mapping is a process to identify the location of each DNA sequence fragment on known genome information. In single-organism genomics, it is easy because the reference genome has already been obtained and we do not have to mention about many mismatches and gaps. Thus, the process is similar to general text search and does not require large computation. Therefore, by using efficient short-read mapping programs such as BWA[5] and Bowtie[6], we can process the output of a next generation sequencer on a few workstations.

In contrast, in metagenomic analysis, the DNA sequence fragments obtained from environmental samples frequently include DNA sequences from many different species, and closely related reference genome sequences are often unavailable. Thus, more sensitive search, generally called a homology search, is required for the identification of novel genes. In addition, in the typical metagenomic analyses, sequenced DNA fragments are translated into protein coding sequences and then further assigned to protein families to improve search sensitivity. This also increases the computational cost of homology search.

The BLAST algorithm is sufficiently sensitive for searching protein families and is much faster than the classical dynamic programming method. Thus, BLASTX program has been used in previous metagenomic researches[7]. However, its performance is

insufficient for analyzing the large quantities of data produced by a next-generation sequencer. In practice, by using BLASTX program, approximately 25,000 CPU days are needed for querying 600 GB of short reads generated from one run of the latest sequencer Illumina's Hiseq 2000. Thus, the metagenomic mapping based on a homology search is a bottleneck in current analysis and its acceleration is highly required.

GHOSTM: a GPU-accelerated homology search tool for metagenomics 3

To address the issue, we developed a new and efficient homology search algorithm suitable for GPU calculation and implemented the system on GPUs. The system accepts a large number of short DNA fragment sequences produced by a next-generation sequencer as the input and, like the BLASTX program, performs DNA sequence homology searches against a protein sequence database. We used NVIDIA CUDA to implement the GPU-calculation. Thus, the system requires CUDA 2.2 or higher. The search system, which we named GHOSTM, demonstrated a calculation speed that was 130 times faster with one GPU than BLAST on a CPU.

3.1 Algorithm

The GHOSTM is mainly composed of three components, as shown in Figure 1. The first component searched the candidate alignment positions for a sequence from the database using the indexes. The second component calculated local alignments around the candidate positions using the Smith-Waterman algorithm[8] for calculating the alignment scores. Finally, the third component sorted the alignment scores and output the search results.

Both the candidate search and local alignment components required a large amount of computing time. Therefore, we processed queries on both components in parallel and mapped them onto GPUs. Thus, multiple queries were simultaneously processed on different GPU cores.

3.2 Search sensitivity and speed

Because metagenome analyses require highly sensitive searches, it is difficult to use homology search program with high speed but low sensitivity, such as BLAT[9]. In contrast, GHOSTM has sufficient search sensitivity for metagenomic analysis. Figure 2 shows the comparison of search sensitivity for each homology search program. To evaluate the search sensitivity, we used the search results obtained with the Smith-Waterman local alignment method implemented in SSEARCH[10], and these results were assumed to be the correct answers. We analyzed the performance of a particular method in terms of the fraction of its results that corresponded to the correct answers obtained by SSEARCH. The search accuracy of GHOSTM was clearly higher than BLAT. Low-scoring hits (e.g., <50) are generally not used in practice because such hits can occur by chance. With the exception of the low-score hits, GHOSTM successfully identified more than 90 % of the hits identified by SSEARCH. This result suggests that GHOSTM is sufficiently accurate for general usage.

Table 1 shows the computational times of BLAST, BLAT, and GHOSTM for 100 thousand reads. Each query read has the

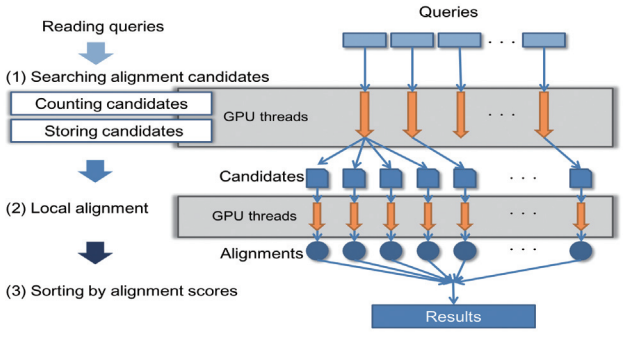


Figure 1 Data flow and processing within GHOSTM.

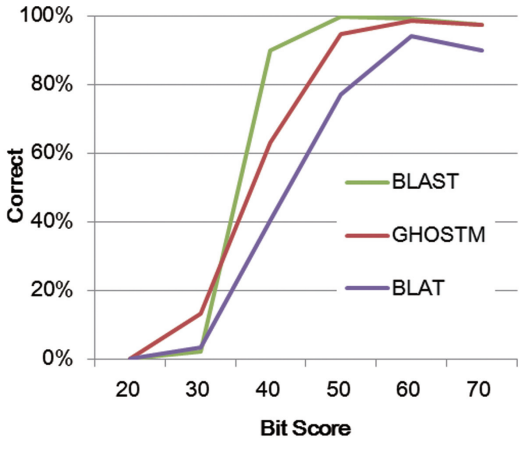


Figure 2 Comparison of search accuracy

length from 60 to 75 bp and the search target is KEGG Genes ("genes.pep") database with approximately 2.5GB. The GHOSTM program achieved a calculation speed approximately 130 and 400 times faster than the BLAST program using 1 thread and 4 threads, respectively. Moreover, GHOSTM was approximately 3.4 times faster than BLAT despite of its higher search sensitivity. GHOSTM achieves both high search speed and high search sensitivity compared with previous homology search programs.

Program	#GPUs	Time (sec.)	Acceleration ratio
GHOSTM	1	2855	129.5
GHOSTM	4	909	406.7
BLAT		9898	37.3
BLASTX (1 thread)		369678	1
BLASTX (4 threads)		102255	3.6

Table 1 Comparison of search speed

Automated computing pipeline on TSUBAME2 4

Even using GHOSTM, it is difficult to analyze huge amount of metagenomic data on a single workstation. Thus, we developed a large-scale automated computing pipeline for analyzing huge amount of metagenomic data obtained from a next-generation sequencer. This pipeline enables us to analyze metagenomic data from a next generation sequencer in real time by utilizing huge computational resources on TSUBAME2. File I/O processes including a database copy and writing search results caused a contention problem when we used many computation nodes. Thus, we changed to employ a sophisticated file transfer manner where data are simultaneously copied from local disk of a node to another in a binary-tree manner.

4.1 Large-scale experiments

By the assistance of TSUBAME Grand Challenge (Large Scale GPU application) program, we performed a large-scale metagenome analysis by using our pipeline on whole TSUBAME2 system. We used data sampled from polluted soils and obtained by using a next-generation sequencer. Original metagenomic data has 224 million DNA reads with 75 bp, and after excluding low-quality data, the dataset became 71 million DNA reads. Thus, homology searches were performed with 71 million DNA reads for NCBI nr amino-acid sequence DB (4.2GB). We evaluated the effective performance of the pipeline with both BLASTX on CPUs and GHOSTM on GPUs.

As results, the pipeline shows almost linear speedup to the number of computing cores. When we used BLASTX as a homology search program, the pipeline achieved to process about 24 million reads per an hour with 16,008 CPU cores (1,334 computing nodes) (Figure. 3). When we used GHOSTM as a homology search program, the pipeline achieved to process about 60 million reads per an hour with 2,520 GPUs (840 computing nodes) (Figure. 4). When we used 2,520 GPUs, the speed up is only 20% compared with the case that we used 1,260 GPUs. This was because the dataset was too small for 2,560 GPUs and it might cause the failure of the load balancing. Therefore, the speedup will be linear even with more than 2,520 GPUs by processing much more metagenomic data.

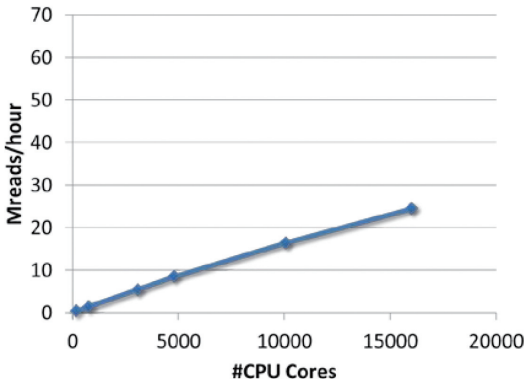


Figure 3 Speedup of the BLASTX-based system for the number of CPU cores

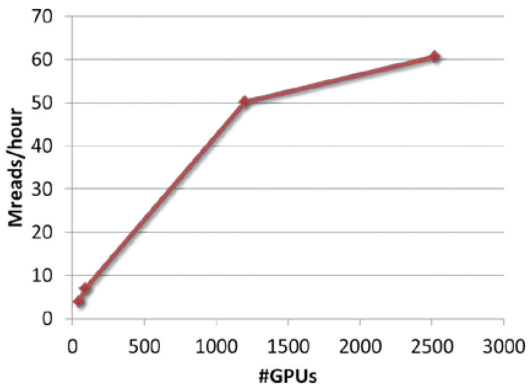


Figure 4 Speedup of the GHOSTM-based system for the number of GPUs

Summary 5

For analyzing metagenomic data obtained from a next generation sequencer in real time, we developed a new efficient GPU-based homology search program GHOSTM and a large-scale automated computing pipeline which enable us to utilize huge computational resources on TSUBAME2.
The results of the experiment with whole TSUBMAME2 system indicate the pipeline can process genome information obtained from a single run of next generation sequencers in a few hours. We believe our pipeline will accelerate metagenome analysis with next generation sequencers.

Acknowledgements

This research was supported in part by HPCI Strategic Program Computational Life Science and Application in Drug Discovery and Medical Development by MEXT of Japan, Cancer Research Development funding by National Cancer Center, Japan, and the CUDA COE Program by NVIDIA.

References

[1] M. Arumugam et al., "Enterotypes of the human gut microbiome.," Nature, vol. 473, no. 7346, pp. 174-80, May 2011.
[2] J. C. Wooley, A. Godzik, and I. Friedberg, "A primer on metagenomics.," PLoS computational biology, vol. 6, no. 2, p. e1000667, Jan. 2010.
[3] S. Suzuki, T. Ishida, K. Kurokawa, and Y. Akiyama, "GHOSTM: A GPU-Accelerated Homology Search Tool for Metagenomics," PLoS ONE, vol. 7, no. 5, p. e36060, May 2012.
[4] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool.," Journal of molecular biology, vol. 215, no. 3, pp. 403-10, Oct. 1990.
[5] H. Li and R. Durbin, "Fast and accurate short read alignment with Burrows-Wheeler transform.," Bioinformatics (Oxford, England), vol. 25, no. 14, pp. 1754-60, Jul. 2009.
[6] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome.," Genome biology, vol. 10, no. 3, p. R25, Jan. 2009.
[7] P. J. Turnbaugh, R. E. Ley, M. a Mahowald, V. Magrini, E. R. Mardis, and J. I. Gordon, "An obesity-associated gut microbiome with increased capacity for energy harvest.," Nature, vol. 444, no. 7122, pp. 1027-31, Dec. 2006.
[8] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences.," Journal of molecular biology, vol. 147, no. 1, pp. 195-7, Mar. 1981.
[9] W. J. Kent, "BLAT--the BLAST-like alignment tool.," Genome research, vol. 12, no. 4, pp. 656-64, Apr. 2002.
[10] W. R. Pearson, "Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms.," Genomics, vol. 11, no. 3, pp. 635-50, Nov. 1991.

GPU-Accelerated Large-Scale Simulation of Seismic-Wave Propagation

Taro Okamoto* Hiroshi Takenaka** Takeshi Nakamura*** Takayuki Aoki****

*Department of Earth and Planetary Sciences, Tokyo Institute of Technology **Department of Earth and Planetary Sciences, Kyushu University
***Earthquake and Tsunami Research Project for Disaster Prevention, Japan Agency for Marine-Earth Science and Technology
****Global Scientific Information and Computing Center, Tokyo Institute of Technology

Simulating seismic wave propagation is important for the study of earthquake sources, the generation of strong ground motions and the excitation of large tsunamis. We describe methods for accelerating large-scale finite-difference time-domain simulation of the seismic wave propagation by the use of graphics processing units (GPUs). We then present examples of the wave-filed from the 2011 Tohoku-Oki earthquake simulated by using the GPUs of TSUBAME supercomputer. The simulated wave-field exhibits strongly complex pattern reflecting the source complexity and the heterogeneities around the source region.

Introduction 1

The Tohoku-Oki earthquake on March 11, 2011 (M_{JMA} 9.0; Fig. 1) generated strong shaking reaching the maximum intensity (seven) on the JMA's scale and caused devastating tsunamis with run-up heights exceeding 30 m. 15,845 people were lost, and 3,368 people are still missing (January 27, 2012). Such magnitude 9 earthquake was not expected to occur along the plate interface off the northeastern Japan. Thus it is very important to study this event for understanding the geophysical condition of the generation of mega-thrust earthquake, the characteristics of the induced strong ground motions, and the mechanism of the excitation of the large tsunamis.

The ground motion records of this event are quite important data for the quantitative studies on the earthquake source and the induced damages. However, modeling of the ground motions is not a simple task because of the strong lateral heterogeneity in and around the Japan trench: steeply varying topography, oceanic water layer, thick sediments, crust with varying thickness and subducting oceanic plate can all affect the seismic waves radiated from suboceanic earthquakes.

The modeling of the ground motion induced by this event is a computational challenge: large memory size and fast computing devices are required because the huge source size of the earthquake imposes a very large domain size for the simulation.

In this paper we review our 3-D finite-difference time domain (FDTD) method [1-3]. In order to simulate the wave propagation with a large grid size, we adopt the GPU (graphics processing unit) computing to our finite-difference program. We then present the results of the simulation of the wave propagation based on a preliminary source model of the 2011 Tohoku-Oki earthquake.

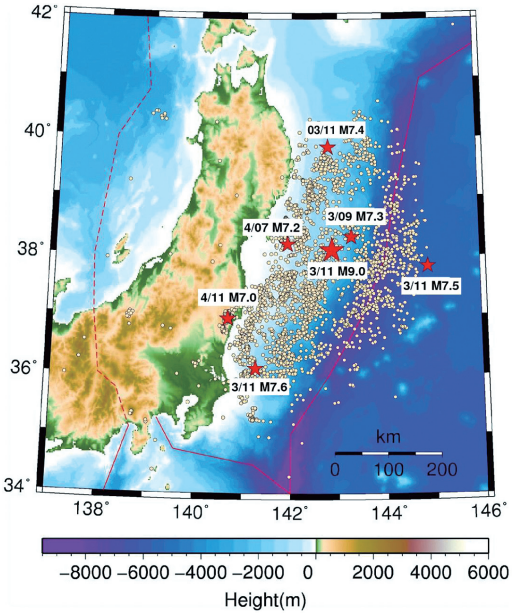


Figure 1
Aftershock distribution of the 2011 Tohoku-Oki earthquake. The epicenters of earthquakes ($M_{JMA} \geqq 4.0$) from Mar. 09, 2011 to Apr. 11, 2011 (JST) are shown. Earthquakes with magnitude larger than or equal to 7 are shown by red stars. Hypocentral data determined by Japan Meteorological Agency are used.

Finite-diference time domain method 2

We apply the time domain, staggered-grid three-dimensional finite-difference scheme [4]. The components of the particle velocity (v_i; i=x, y, z) and the stress tensor (τ_{ij}; i, j=x, y, z) are the field variables. We use three material parameters (Lamé coefficients and density) as we assume isotropic and elastic material for the simulation. Thus twelve variables are assigned

to a unit cell (i.e., a heterogeneous formulation: Fig. 2). The accuracies of finite-differences are fourth-order in space and second-order in time, respectively. In order to incorporate the effects due to heterogeneities and irregularities in the structure, we have adopted an approach to model structures with both of the land and ocean topography and heterogeneity in 3D seismic modeling with the finite-difference method [5]. The approach unifies the implementation for irregular free-surface and that for irregular liquid-solid interface.

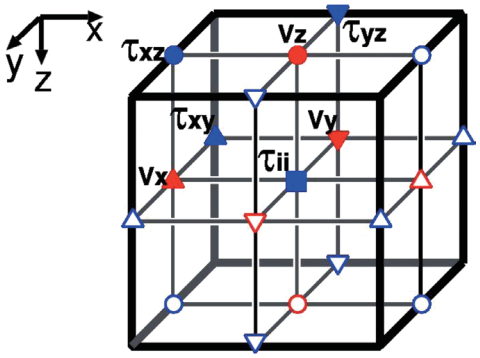


Figure 2 A single unit cell of the staggered grid. Open symbols denote the variables that belong to the neighboring cells [2].

GPU computing 3

The GPU (graphics processing unit) is a remarkable device for its many-core architecture and high memory bandwidth (Fig. 3). The GPU delivers extremely high computing performance at a reduced power and cost compared to conventional central processing units (CPUs). The simulation of seismic wave propagation is a typical memory-intensive problem which involves a large amount of data transfer between the memory and the arithmetic units, while the number of arithmetic operations is relatively small. This is the reason we adopt the GPU computing to the simulation of seismic wave propagation: it can benefit from the high memory-bandwidth of the GPUs.

We use the *TSUBAME 2.0* supercomputer in Global Scientific Information and Computing Center, Tokyo Institute of Technology, for the seismic wave simulation employed in this paper. The TSUBAME 2.0 is a large-scale GPU cluster equipped with 1408 nodes and 4224 of NVIDIA M2050 GPUs, and has a peak performance of 2.4 PFlops (peta-flops). It is ranked as the world fifth fastest supercomputer in the November 2011 TOP-500 list (www.top500.org).

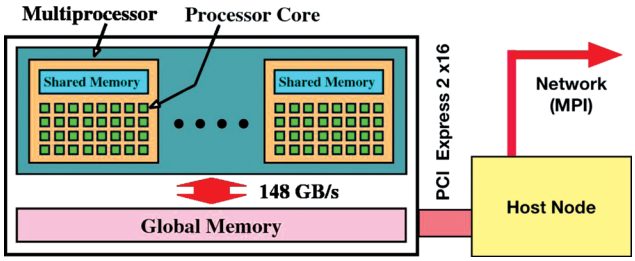


Figure 3 A simplified diagram of a GPU (NVIDIA M2050) used in this study. A single GPU has 14 processors (multiprocessors) and a single multiprocessor has 32 processor cores. Thus 448 processor cores are integrated in a GPU. The size of the global memory is 3 GB.

Memory optimization

The GPU is characterized by its hundreds of cores for arithmetic instructions (Fig. 3): we need to provide sufficient amount of data to these cores. The bandwidth of the main memory (called "global memory" in NVIDIA GPUs) connected to the GPU is much faster than that of the memory system connected to the conventional CPUs. However, 400 to 600 clock cycles of memory latency still occurs in transferring the data between the global memory and the GPU. It is thus critical for performance to optimize the memory manipulation in memory-intensive applications such as the simulation of seismic wave propagation. Therefore, we use the fast (but small) memories in the GPU, the *registers* and the *shared memory*, as software managed cache memories to reuse the data and hence to reduce the data transfer from the global memory [1, 6-9] (see Fig. 4 for the method).

In order to reduce the access to the global memory, we define the material parameters only at the center of the unit cell. The material parameters at the grid points for particle velocities and shear stresses are computed by using the values defined at the center of the unit cells at every time steps. The unified implementation method [5] allows us to compute the appropriate values of the material parameters. We further use the look-up table method for the material parameters. In addition to these techniques we optimize the number of grid points (i.e., the *block size*) assigned to a single execution unit (called a *multiprocessor*),

because the memory transfer rate is better for grouped memory transaction using blocks of proper size (typically 16 to 64) than that for serialized, one-by-one memory transaction.

Fig. 5 shows the performance of the (multi-GPU-capable) FDTD program executed on a single-GPU. As a comparison we show the performance of a FDTD program executed on the host CPUs. Note that this example for CPU does not involve MPI inter-node communications: the program is parallelized with OpenMP and executed on a single node. For these programs, the performance of a single GPU is roughly three-fold faster than that of a single node (2 CPUs, 12 cores).

Parallel computing with multi-gpus

For parallel computing with multi-GPUs, we decompose the FDTD domain into *subdomains* and allocate a single subdomain to a single GPU. This is necessary for large-scale simulations because the size of the global memory of a GPU is not large (e.g., 3 GB in the case of NVIDIA M2050). We here use the MPI library for our parallel FDTD program.

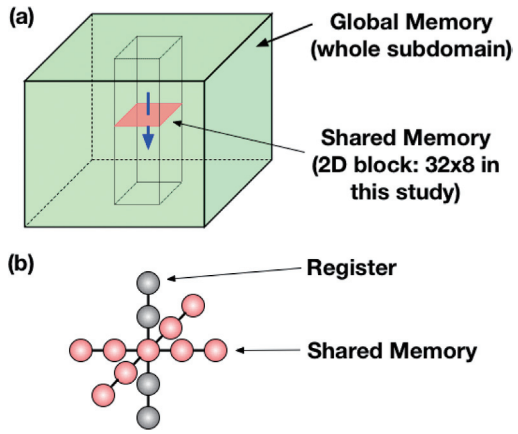


Figure 4
Illustrations showing the usage of the shared memory and registers. (a) All the variables in the whole subdomain is stored in the global memory in the GPU. We further copy the variables on a chosen level into the small, two-dimensional (2D) blocks assigned in the shared memory. We assign a single thread to each unit cell. Thick blue arrow denotes the marching direction of the computation. (b) Illustration of grid points for the finite-difference. Values on red points are stored in shared memory and those on gray point in registers. To the variables not on the 2D plane, only "vertical" finite-differences are operated so that they are not required to be shared among different threads.
When the computation proceeds to the next level, the variables in the shared memory are copied to the registers and vice versa. This reduces the data transfer from the global memory.

In the parallel computing we need to exchange the data in the ghost zones between the neighboring subdomains. We adopt the *three-dimensional* (3D) domain decomposition (Fig. 6). Here we note that, in GPU computing, the data within the ghost zones in the GPU memory must be once copied to the host memory in order to exchange them with other GPUs installed on the other nodes (see Fig. 3). Thus in decomposing the domain, we use dedicated memory buffers for the ghost zones to improve the data transfer rate between the GPU memory and the host memory [1,3].

We also overlap the procedures for the communication and the computation to reduce the total processing time. In our program we first compute for the grid points at and near the sides of the subdomains for ghost point exchange. Then we start the exchange procedure for the ghost points and the computations for the internal grid points simultaneously.

With these optimizations, the performance of our 3D FDTD program scales well with the increasing number of GPUs (Fig. 7): by using 400 nodes of the TSUBAME supercomputer and under the condition of 2 GPUs activated per a node, a very high single-precision performance of about 50 TFlops is achieved in the case of 800 GPUs (85 % of the complete scalability). Even higher performance of about 61 TFlops (69 %) is achieved in the case of 1200 GPUs under the condition of 3 GPUs per a node [3].

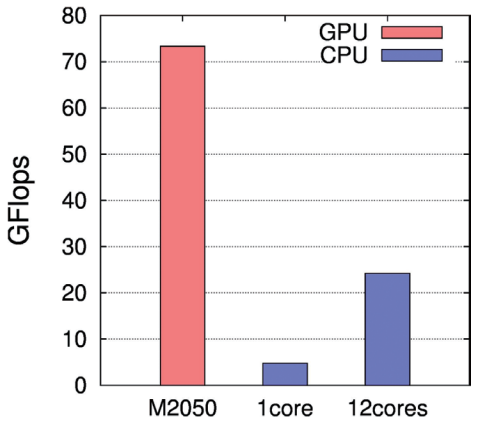


Figure 5
Performance of GPU and CPU in GFlops.
A single GPU is used to execute the (multi-GPU-capable) FDTD program with a domain size of 320 × 320 × 320.
On CPUs we parallelized a Fortran source code by using OpenMP with 1D domain decomposition, compiled it by PGI Fortran compiler with "-fastsse" option, and executed the program on a single host node with 12 cores (2 CPUs) in total.

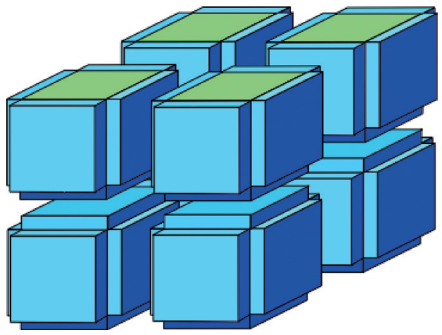


Figure 6
Schematic illustration of the 3D domain decomposition.
Blue regions indicate the ghost zones.

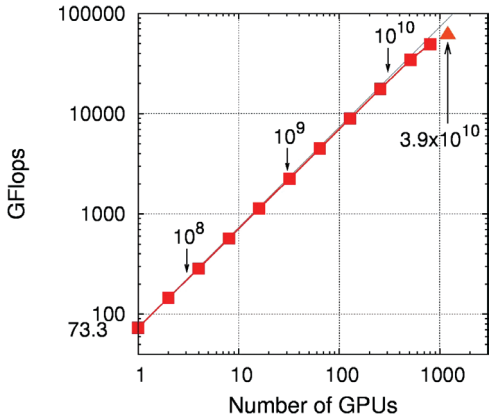


Figure 7
Weak scaling curve of Multi-GPU case on TSUBAME–2.0.
The subdomain size was fixed to 320 × 320 × 320.
The total number of subdomains is equal to the number of used GPUs. The numbers of unit cells are also shown for several points. The experiments were performed with 2 GPUs per a node, except in the case of 1200 GPUs that was performed with 3 GPUs per a node.

Simulation of Tohoku-Oki earthquake 4

The aftershock distribution of the 2011 Tohoku-Oki earthquake indicates the approximate source area (i.e., size of the fault) of about 500 km long and 200 km wide. We use a preliminary source model determined by ourselves [10] for the results shown in this paper. The preliminary source model was determined by using waveform data recorded at world-wide stations that are far from the source region and hence the model represents long-wavelength feature of the source. The source model consists of point sources distributed on the grid points (23 × 11) on the fault plane with a spacing of about 20 km. We compile several detailed structure models [11-14] in and around the northeastern Japan region to construct the 3D model for the simulation. The 3D model incorporates the irregular topography of the land and ocean area, ocean water layer, sediments, crust and subducting plates. For the example shown here we were required to use 1000 GPUs (334 nodes) of the TSUBAME 2.0 supercomputer. This is about 24% of the entire resources of the TSUBAME 2.0. The FDTD parameters are as follows: the grid spacing 0.15 km, the time increment 0.005 s, the grid size 6400 × 3200 × 1600, time steps 44000, and the maximum frequency 0.61 Hz.

Fig. 8 and 9 show the snapshots of the ground motion (particle velocity) computed for the preliminary source model of the 2011 Tohoku-Oki earthquake [2]. The ground motions at the free-surface in the land area and at the solid side of the ocean bottom in the oceanic area are visualized in color scale. Although not visualized, the propagation of the pressure waves within the ocean water layer is also modeled in the simulation. We are able to observe strong interference due to the complex source model and the heterogeneous structure. The complexity is especially significant in the later part of the simulation. It is also remarkable that, from about 130 s to 170 s, regions with strong positive motion (red) and negative motion (blue) emerge near the coast of the Fukushima prefecture (Fig. 9, yellow circle). Motions in the regions overwhelm the wave-field propagated from around the epicenter, and largely distort the pattern of the wave propagation. The source that provides this feature is the relatively large slip near the Fukushima prefecture retrieved in the source model [2]. Such a feature was also observed in the strong ground motion records and the strong-motion-generation-areas (SMGAs) were identified below the coast of Miyagi to Fukushima prefectures [15]. These characteristics must be studied further in the future analysis, with improvements in the source model as well as in the 3D structure model.

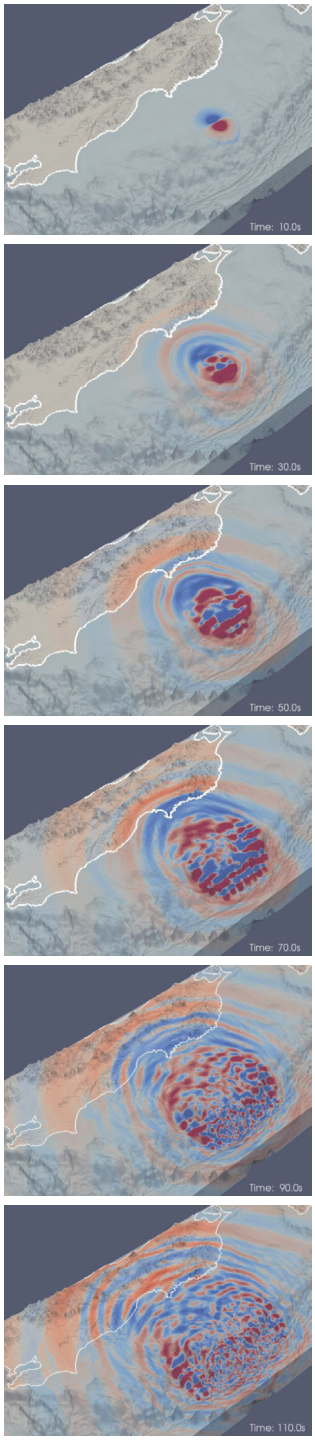


Figure 8
Snapshots of the simulated wave-field. The vertical component of the particle velocity is visualized with color scale. Red and blue colors denote upward and downward motions, respectively.
From left to right, top to bottom the snapshot at 10 s after the onset of the rupture, at 30 s, 50 s, 70 s, 90 s, and 110 s, respectively, are shown.

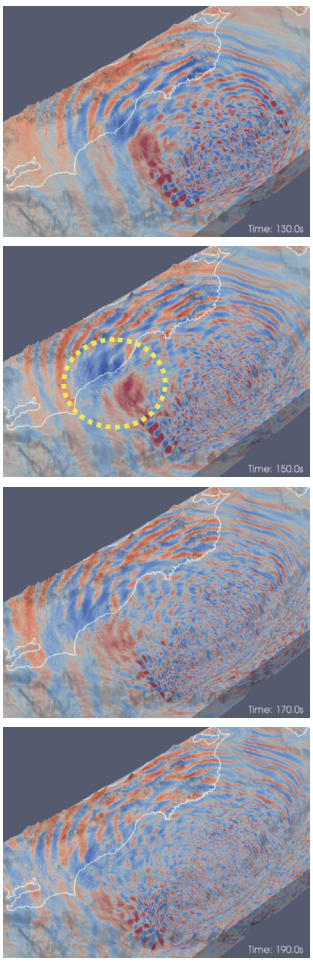


Figure 9
Same as Fig. 8 but for snapshots at 130 s, 150 s, 170 s and 190 s.

Conclusion 5

In order to simulate the wave propagation from the 2011 Tohoku-Oki earthquake, we constructed a preliminary 3D structure model for the northeastern Japan region by compiling the models for topography, sediments, crust, and subducting plates. We adopted the GPU computing to accelerate the large-scale finite-difference simulation of the seismic wave propagation. By using about 24% of the resources of the GPU supercomputer, TSUBAME 2.0, at Tokyo Institute of Technology, Japan, we are now able to simulate the seismic wave propagation from the source of the huge fault plane up to

0.61 Hz within a tolerable computation time. The snapshots of simulated wave-field exhibit strongly complex pattern because of the complex source model, the irregular topography and the shallow heterogeneities. Such effect must be considered in improving the source model, the 3D structure model, and in the future quantitative study on the observed ground motion records.

Acknowledgements

We are grateful to the researchers who provided the structural models. We are also grateful to Global Scientific Information and Computing Center (GSIC), Tokyo Institute of Technology, for providing opportunity (Grand Challenge) to use large number of nodes of TSUBAME. This research was partially supported by Grant-in-Aid for Scientific Research (KAKENHI 23310122) from Japan Society for the Promotion of Science (JSPS).

References

[1] Okamoto, T., Takenaka, H., Nakamura, T. and Aoki, T. "Accelerating large-scale simulation of seismic wave propagation by multi-GPUs and three-dimensional domain decomposition", Earth, Planets and Space, vol. 62, no. 12, pp. 939-942 (2010).

[2] Okamoto, T., Takenaka, H., Nakamura, T., and Aoki, T. "Large-scale simulation of seismic-wave propagation of the 2011 Tohoku-Oki M9 earthquake", Proceedings of the International Symposium on Engineering Lessons Learned from the 2011 Great East Japan Earthquake, pp. 349-360 (2012).

[3] Okamoto, T., Takenaka, H., Nakamura, T. and Aoki, T. "GPU-accelerated simulation of seismic wave propagation", in GPU Solutions to Multi-scale Problems in Science and Engineering, Yuen, D., Wang, J., Johnsson, L., Chi, C.-H., Shi, Y. (Eds.), 250 pp., Springer, in press.

[4] Graves, R. W. "Simulating seismic wave propagation in 3D elastic media using staggered-grid finite differences", Bull. Seism. Soc. Am., vol. 86, pp. 1091-1106 (1996).

[5] Takenaka, H., Nakamura, T., Okamoto, T. and Kaneda, Y. "A unified approach implementing land and ocean-bottom topographies in the staggered-grid finite-difference method for seismic wave modeling", Proc. 9th SEGJ Int. Symp., CD-ROM Paper No.37 (2009).

[6] Abdelkhalek, R., Calandra, H., Coulaud, O., Roman, J. and Latu, G. "Fast seismic modeling and reverse time migration on a GPU cluster", International Conference on High Performance Computing & Simulation, pp. 36-43 (2009).

[7] Micikevicius, P. "3D finite-difference computation on GPUs using CUDA", GPGPU-2: Proc. 2nd Workshop on General Purpose Processing on Graphics Processing Units, pp. 79-84, Washington DC, USA (2009).

[8] Michéa, D. and Komatitsch, D. "Accelerating a three-dimensional finite-difference wave propagation code using GPU graphics cards", Geophys. J. Int., doi: 10.1111/j.1365-246X.2010.04616.x (2010).

[9] Okamoto, T., Takenaka, H. and Nakamura, T. "Simulation of seismic wave propagation by GPU", Proceedings of symposium on advanced computing systems and infrastructures, pp. 141-142 (2010).

[10] Okamoto, T., Takenaka, H., Hara, T., Nakamura, T. and Aoki, T. "Rupture Process And Waveform Modeling of The 2011 Tohoku-Oki, Magnitude-9 Earthquake", American Geophysical Union, Fall Meeting, U51B-0038, San Francisco, USA (2011).

[11] Kisimoto, K. "Combined bathymetric and topographic mesh data: Japan250m.grd", Geological Survey of Japan, Open-file Report, No. 353 (1999).

[12] Fujiwara, H., Kawai, S., Aoi, S., Morikawa, N., Senna, S., Kudo, N., Ooi, M., Hao, K. X.-S., Hayakawa, Y., Toyama, N., Matsuyama, H., Iwamoto, K., Suzuki, H. and Liu, Y. "A study on subsurface structure model for deep sedimentary layers of Japan for strong-motion evaluation", Technical Note of the National Research Institute for Earth Science and Disaster Prevention, No.337 (2009).

[13] Baba, T., Ito, A., Kaneda, Y., Hayakawa, T. and Furumura, T. "3-D seismic wave velocity structures in the Nankai and Japan Trench subduction zones derived from marine seismic surveys", Abstr. Japan Geoscience Union Meet., S111-006, Makuhari, Japan (2006).

[14] Nakamura, T., Okamoto, T., Sugioka, H., Ishihara, Y., Ito, A., Obana, K., Kodaira, S., Suetsugu, D., Kinoshita, M., Fukao, Y. and Kaneda, Y. "3D FDM simulation for very-low-frequency earthquakes off Kii Peninsula", Abstr. Seism. Soc. Japan, P1-06, Hiroshima, Japan (2010).

[15] Kurahashi, S. and Irikura, K. "Source model for generating strong ground motions during the 2011 off the Pacific coast of Tohoku earthquake", Earth Planets Space, Vol. 63, 571-576 (2011).

● TSUBAME e-Science Journal No.6

Published 07/31/2012 by GSIC, Tokyo Institute of Technology ©
ISSN 2185-6028
Design & Layout: Kick and Punch
Editor: TSUBAME e-Science Journal - Editorial room
Takayuki AOKI, Thirapong PIPATPONGSA,
Toshio WATANABE, Atsushi SASAKI, Eri Nakagawa
Address: 2-12-1-E2-1 O-okayama, Meguro-ku, Tokyo 152-8550
Tel: +81-3-5734-2087 Fax: +81-3-5734-3198
E-mail: tsubame_j@sim.gsic.titech.ac.jp
URL: http://www.gsic.titech.ac.jp/

TSUBAME

International Research Collaboration

The high performance of supercomputer TSUBAME has been extended to the international arena. We promote international research collaborations using TSUBAME between researchers of Tokyo Institute of Technology and overseas research institutions as well as research groups worldwide.

Recent research collaborations using TSUBAME

1. Simulation of Tsunamis Generated by Earthquakes using Parallel Computing Technique
2. Numerical Simulation of Energy Conversion with MHD Plasma-fluid Flow
3. GPU computing for Computational Fluid Dynamics

Application Guidance

Candidates to initiate research collaborations are expected to conclude MOU (Memorandum of Understanding) with the partner organizations/ departments. Committee reviews the “Agreement for Collaboration” for joint research to ensure that the proposed research meet academic qualifications and contributions to international society. Overseas users must observe rules and regulations on using TSUBAME. User fees are paid by Tokyo Tech’s researcher as part of research collaboration. The results of joint research are expected to be released for academic publication.

Inquiry

Please see the following website for more details.
http://www.gsic.titech.ac.jp/en/InternationalCollaboration

GSIC
Global Scientific Information and Computing Center