42 TFlops Hierarchical N-body Simulations on GPUs with Applications in both Astrophysics and Turbulence

Tsuyoshi Hamada, Rio Yokota, Keigo Nitadori, Tetsu Narumi, Kenji Yasuoka, Makoto Taiji

Overview

- Introduction
- GPU implementation of the tree algorithm
- Results of cosmological N-body simulation using GPU-treecode
- GPU implementation of the FMM
- Results of a vortex particle simulation of homogeneous isotropic turbulence using GPU-FMM
- Overall cost performance of the simulations
- Conclusions

Introduction

- Groundbreaking N-body simulations have won the Gordon Bell prize in 1992, 1995-2003, 2006.
- In the field of N-body simulations during the past few years, graphics processing units (GPUs) have been preferred.
- Hierarchical algorithms such as the tree algorithm and fast multipole method (FMM) are also indispensable requisites, because they bring the order of the operation count from $O(N^2)$ down to O(NlogN) or even O(N).
- Challenges: inefficient use of parallel pipelines when the number of target particles per cell was small.
- Motivation: a method that could pack “multiple walks” that are evaluated by the GPU simultaneously via redesign the tree algorithm and FMM

GPU implementation of the tree algorithm

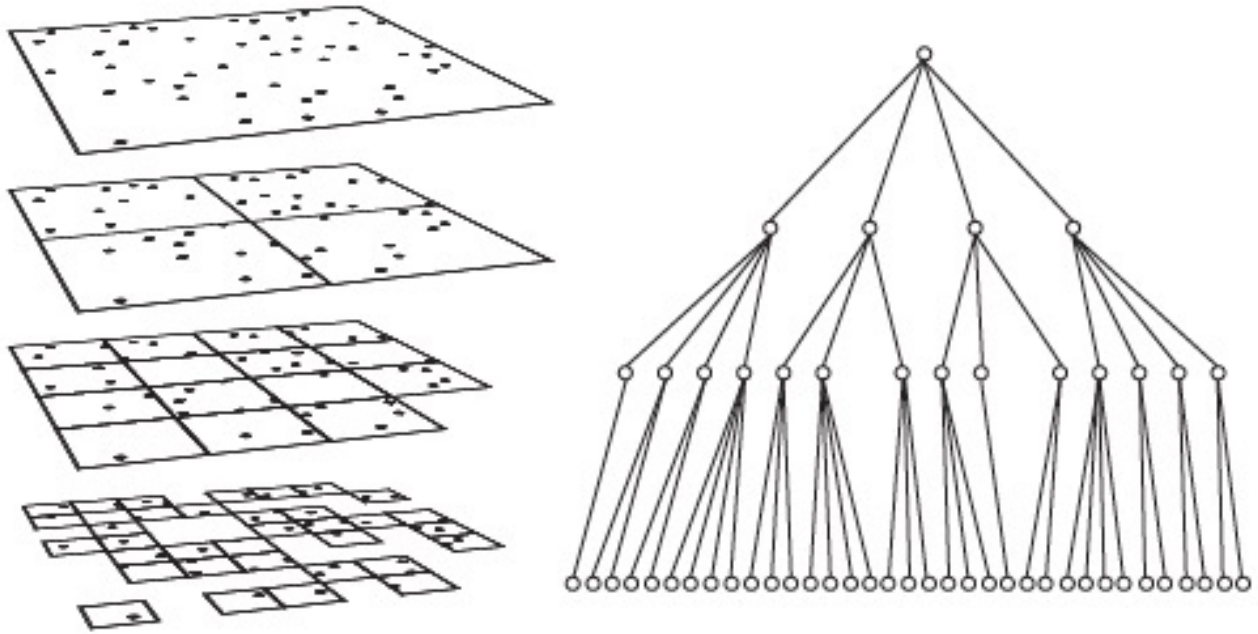


Figure 1: Hierarchical division of the calculation domain and the corresponding tree structure

- The treecode by Barnes and Hut represents a system of N particles in a hierarchical manner by the use of a spatial tree data structure.
- Whenever the force on a particle is required, the tree is traversed, starting at the root.
- At each level, a gravity center of particles in a cell is added to an “interaction list” if it is distant enough for the truncated series approximation to maintain sufficient accuracy in the force.

GPU implementation of the tree algorithm

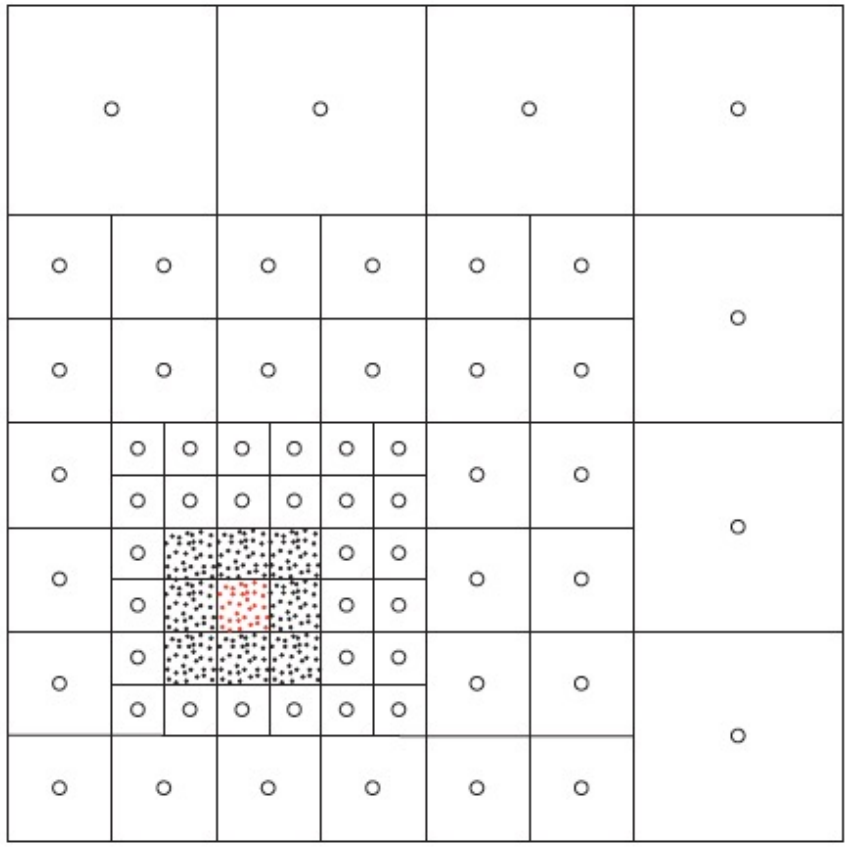


Figure 2: Interaction list in Barnes' modified tree algorithm. The small black dots represent the interacting particles. The white circles represent the truncated series expansions for each cell. Both the dots and circles are included in an interaction list.

- The modified tree algorithm terminates the tree traversal when the cell contains less than Ncrit particles.
- This results in an average number of particles per cell N_g in between $N_{crit}/8$ and N_{crit}
- The modified tree algorithm reduces the calculation cost of the host computer by roughly a factor of N_g

Comparison of GPU tree codes

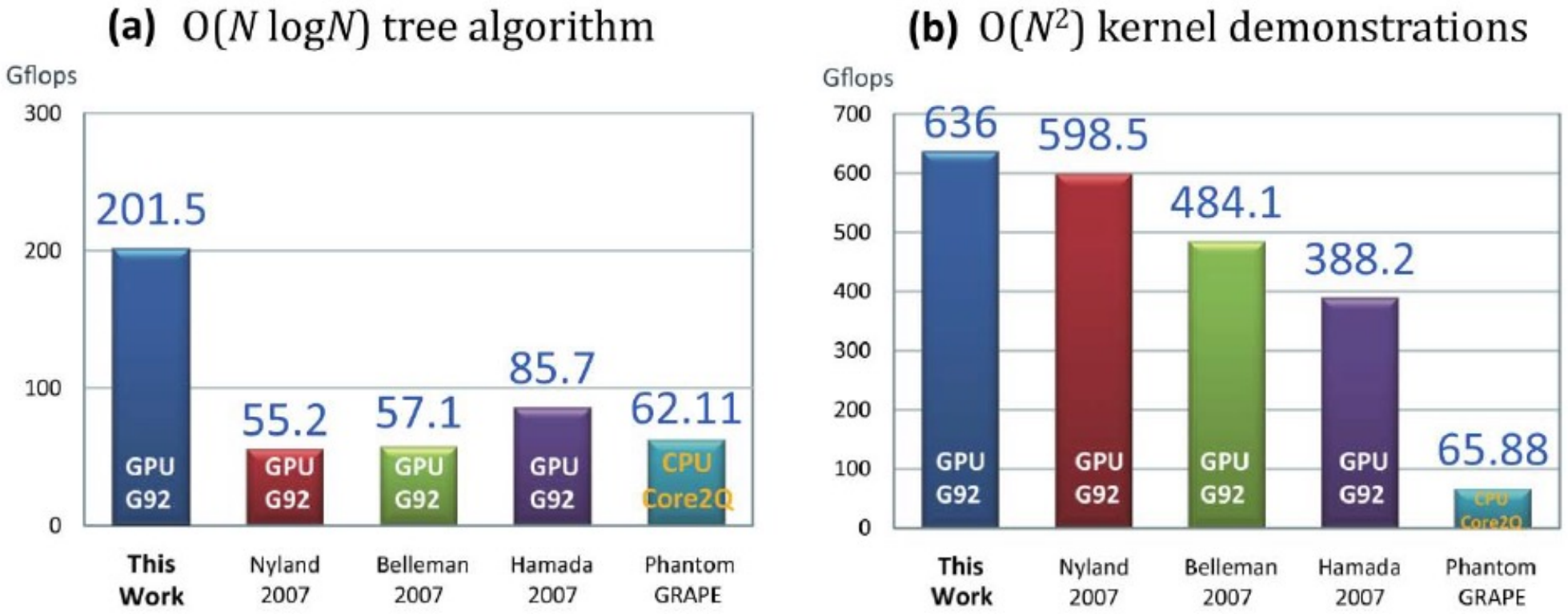


Figure 3: Comparison of the performances of the N-body kernel using (a) the $\mathcal{O}(N \log N)$ tree algorithm and (b) the $\mathcal{O}(N^2)$ direct-summation approach with shared timestep scheme on a single GeForce 8800 GTS. For the Flops count, we used a unified operation count of 38 for the calculation between a pair of particles.

Details of proposed code

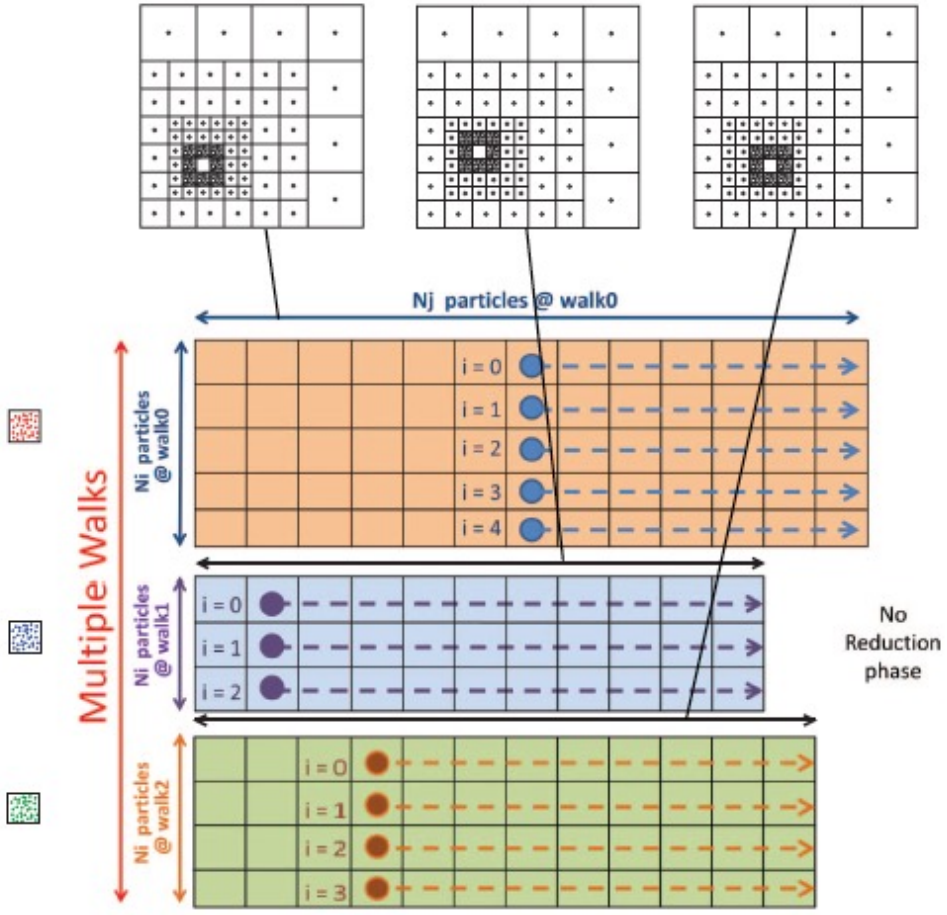


Figure 7: Implementation of the novel GPU code reported by this paper –the multiple walks method

The sequence of the algorithm is as follows:

1. The data of the multiple walks are prepared on the host CPU
2. Multiple walks are then sent to the GPU.
3. Calculations are performed.
4. The forces calculated by the different blocks in the GPU are received in a bundle.
5. The orbital integration and other caculations are performed on the host CPU.
6. The process is repeated.

Cosmological N-body simulation using GPU-treecode

- A cosmological N-body simulation of a sphere of radius 136.28 Mpc (mega parsec) with 1,608,044,129 particles for 327 timesteps.
- A particle represents 8.20 × 104 solar masses.
- Simulation from z = 21.1, where z is red-shifted, to z = 18.5.

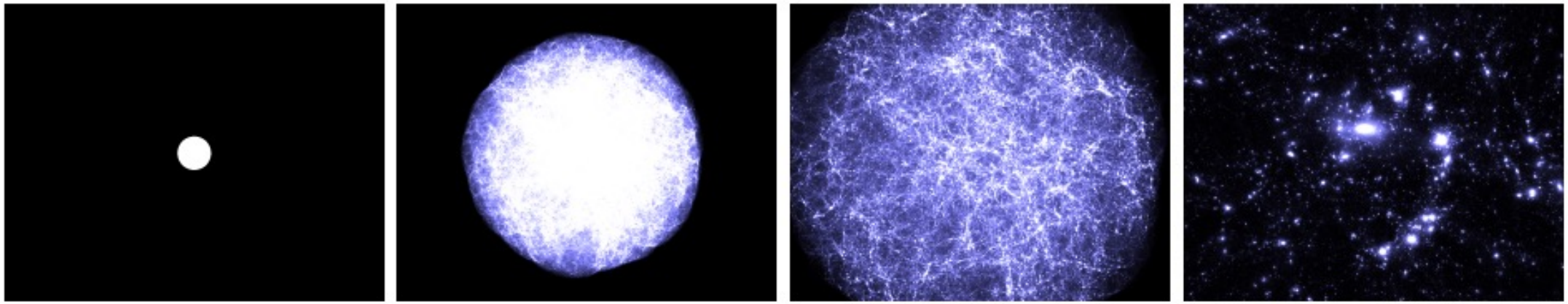


Figure 10: Snapshots of the simulation with 64M particles at z =18.8 (left), z = 4.5 (middle left), z = 2.6 (middle right), and z = 0 (right).

Performance validation

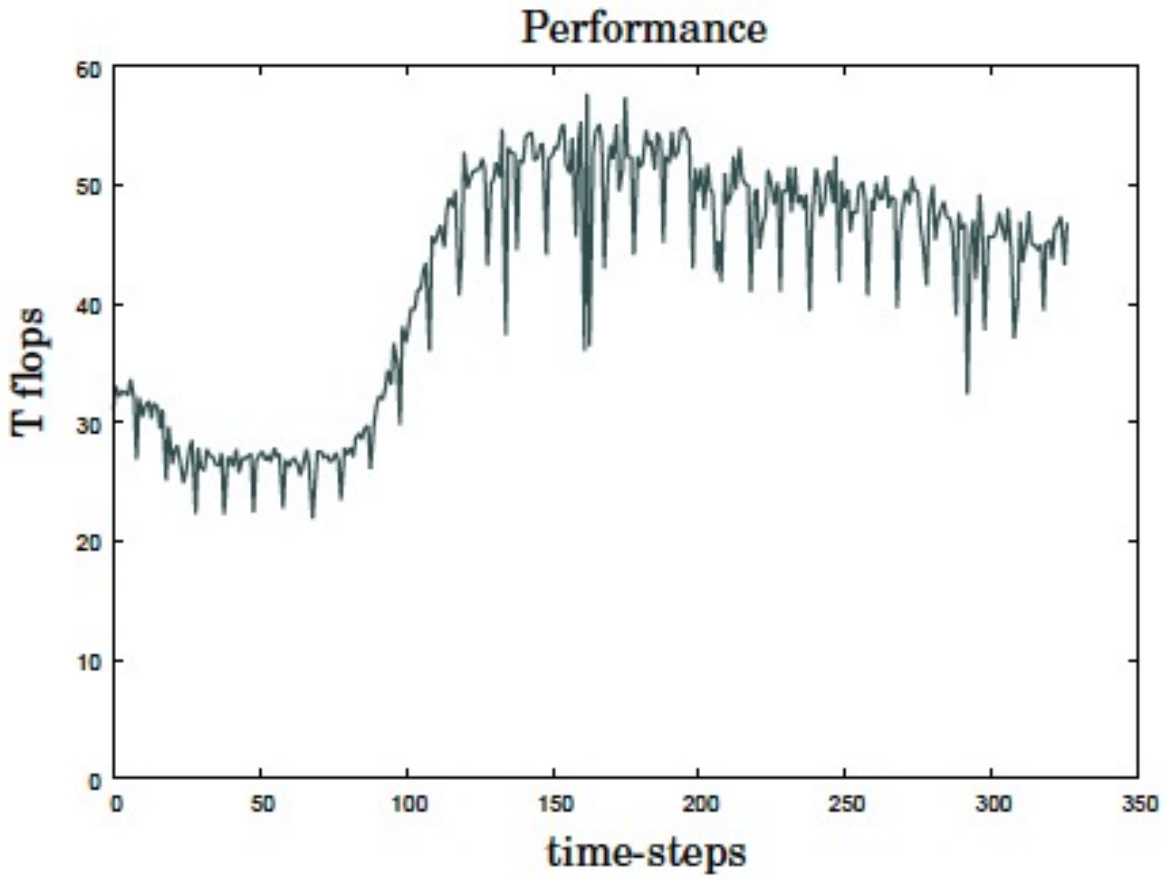


Figure 8: Measured performance of each timestep

- Due to the evolution of a large-scale structure in the universe, the number of interactions increased, and as a result, the performance also increased. After 150 steps the performance reached a constant state

GPU implementation of the FMM

In order to execute the FMM efficiently on the GPU, the following modifications were made to the FMM:

1. The complex spherical harmonics were transformed to real basis functions.
2. All of the translation matrices were generated on-the-fly.
3. The box structure and interaction list of the FMM are restructured and renumbered to match the number of threads per block.
4. The same multipole walks technique is implemented for all stages of the FMM algorithm.

A vortex particle simulation of homogeneous isotropic turbulence using GPU-FMM

- Vortex method: a particle based method for fluid dynamics simulations.
- It is especially well suitable for solving turbulence, because it calculates the vortex interactions seen in turbulent flows.
- The Navier-Stokes equation is solved in the velocity-vorticity form, and discretized with vortex particles.
- Velocity equation:

$$\mathbf{u}_i = \sum_{j=1}^{N} \boldsymbol{\alpha}_j g_\sigma \times \nabla G$$

- Vorticity equation:

$$\frac{D\boldsymbol{\alpha}_i}{Dt} = \sum_{j=1}^{n} \boldsymbol{\alpha}_j \nabla (g_\sigma \times \nabla G) \cdot \boldsymbol{\alpha}_i.$$

Performance validation

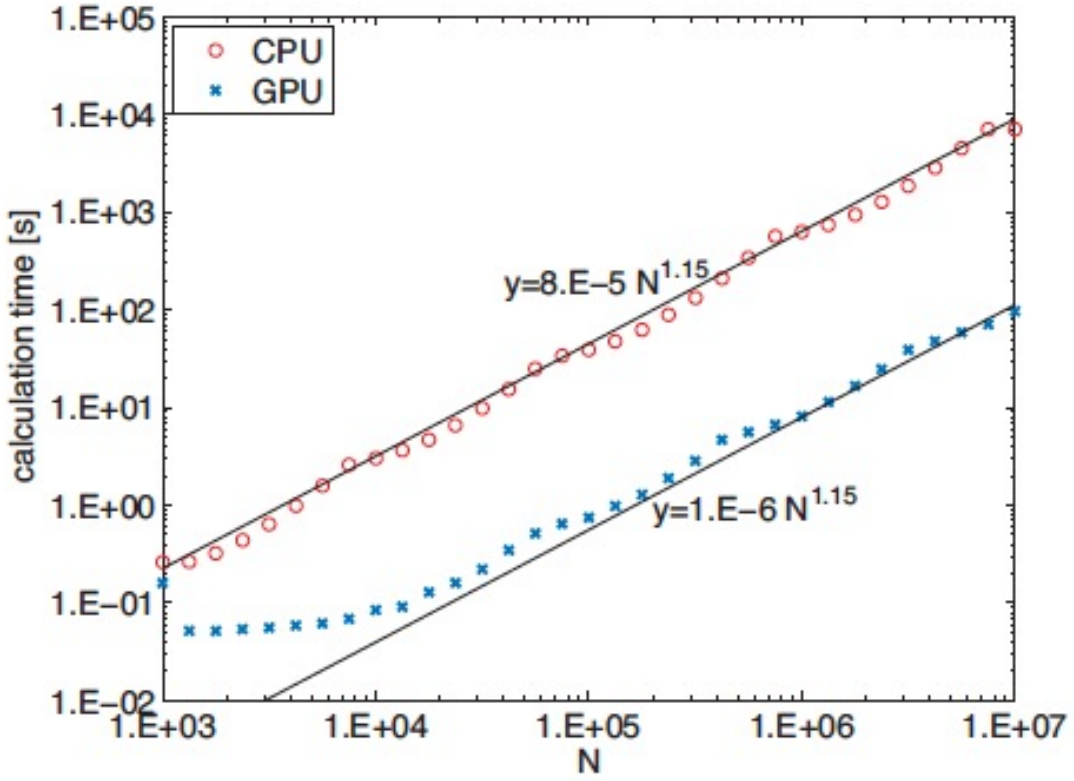


Figure 11: Cputime of the FMM on a serial CPU and GPU ($p = 10$). Dotted lines are functional fits as $y = f(N)$, where y is the time, and N is the number of particles.

- FMM does not scale exactly as O(N), but rather shows a scaling close to $O(N^{1.15})$.
- The FMM on the GPU is approximately 80 times faster than the FMM on the CPU.

Performance validation

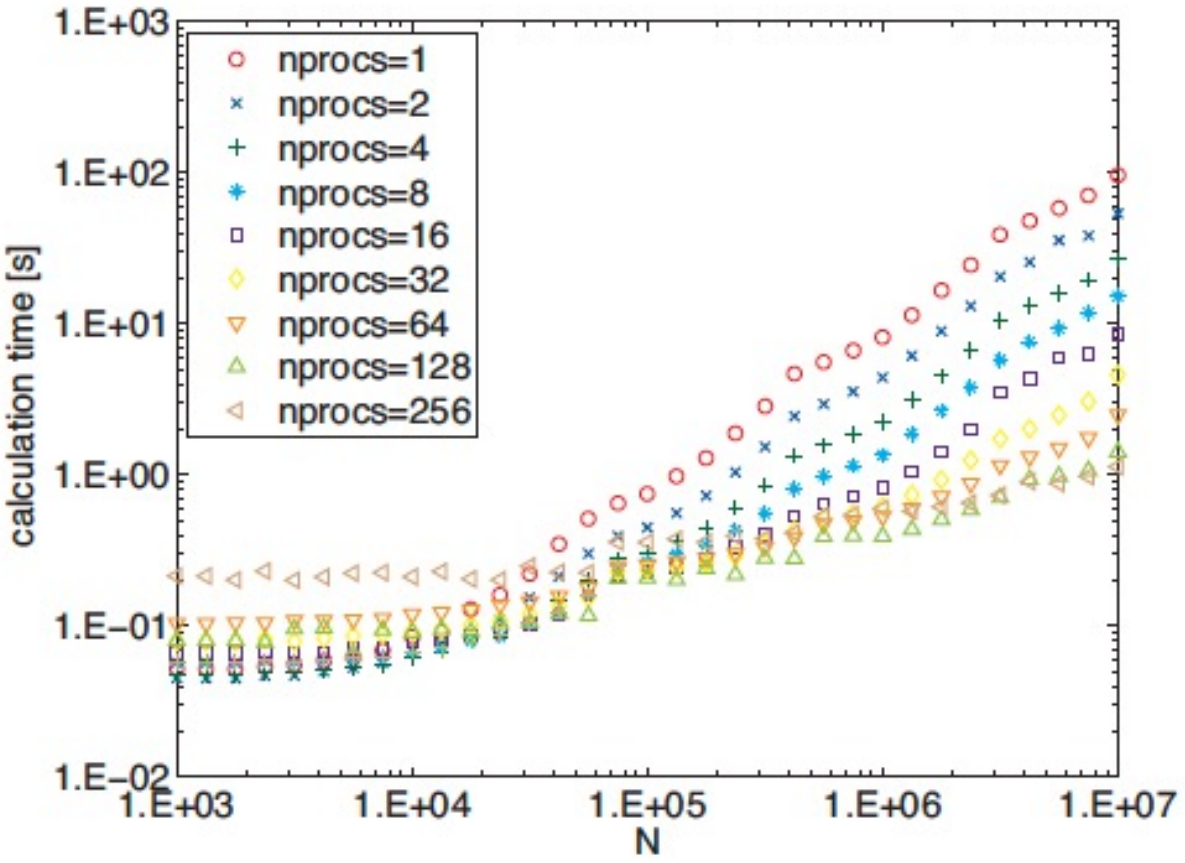
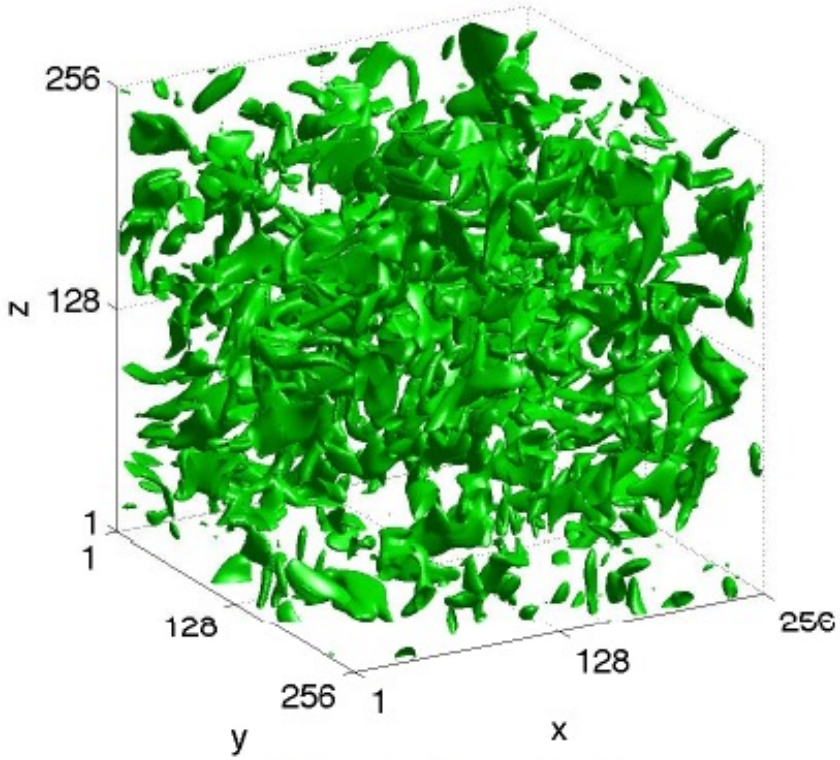


Figure 12: Cputime of the FMM on parallel GPUs

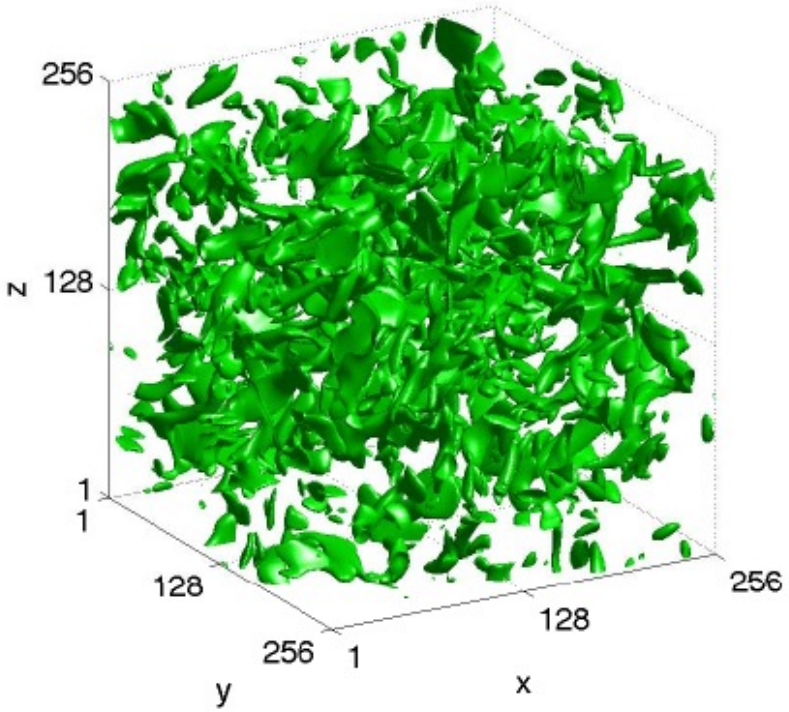
- The low parallel efficiency for small N is a direct consequence of the low performance of GPUs for small N.
- The calculation time for $N < 10^4$ increases for nprocs >= 16, because time spent on communication becomes large compared to the calculation time.

Calculation Results

- The larger structures match between the two methods, the smaller structures behave differently



(a) *Spectral method*



(b) *Vortex method*

Figure 15: Isosurface of the second invariant (II) of the velocity derivative tensor

Overall cost performance of the simulations

Table 2: Price of the GPU cluster

Elements	Quantity	Price (JPY)	Price ($)
GPUs	256	12,160,000	$ 118,345
Host PCs	128	10,716,032	$ 104,292
Network switch	4	644,800	$ 6,275
Total		**23,520,832**	**$ 228,912**

Table 3: Comparison with SC06 GB finalist

	'06 finalist	This work	Ratio
number of particles	2,159,038	1,608,044,129	745
price	$ 2,384	$ 228,912	96
GFlops (uncorrected)	36.31	42,150	1,161
GFlops (corrected)	15.39	28,100	1,826
$/GFlops	158	8.5	19

- The performance is 1870 times higher and the price/performance is 18.59 times better

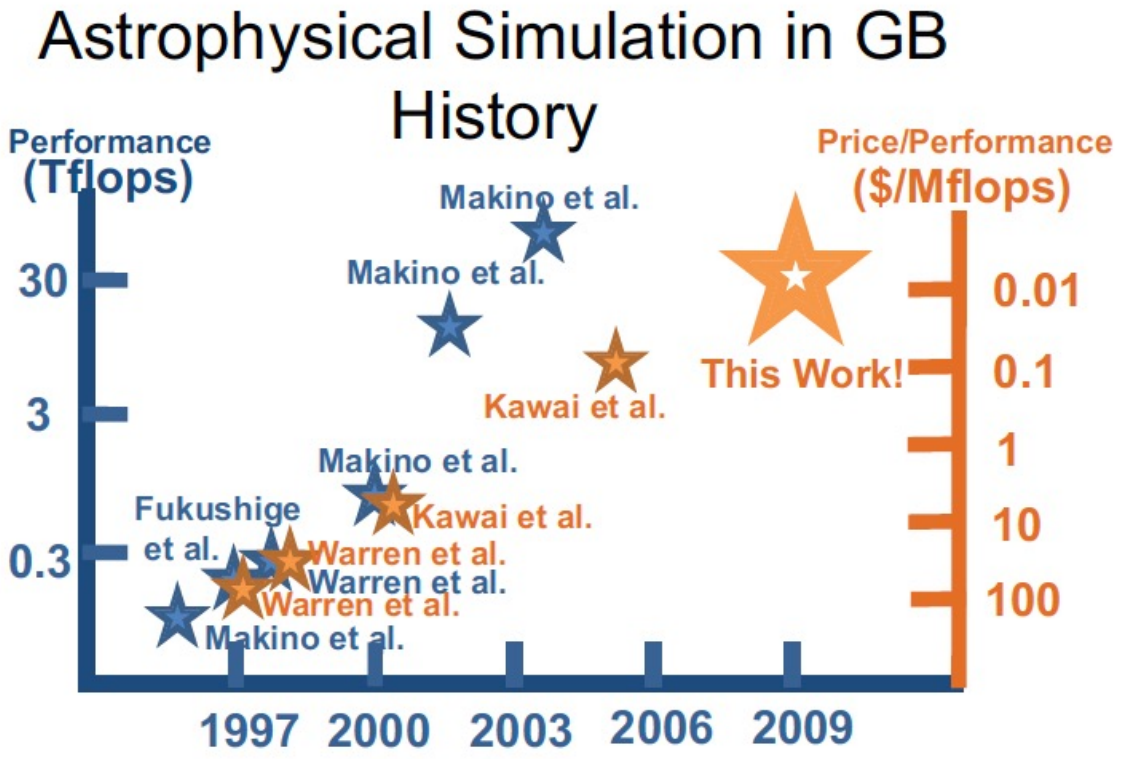


Figure 17: Comparison with previous Gordon Bell winners

- Compared to the Gordon Bell finalist of 2006 [13], the number of particles used in the present gravitational simulation is 768 times larger

Conclusions

- A hierarchical N-body simulation has been performed on a cluster of 256 graphics processing units (GPUs) using the treecode.
- A vortex particle simulation of homogeneous isotropic turbulence using 16,777,216 particles was performed using FMM.
- Both the tree algorithm and FMM show a significant performance gain when executed on GPUs as compared to the performances of the hierarchical algorithms running on CPUs.
- Using the novel approach, a GPU cluster can outperform a PC cluster from the viewpoints of cost/performance, power/performance, and physical dimensions/performance.