Optimizing the Particle Mesh method for extreme scale simulations

Yuki Kaneko and Tomoaki Ishiyama (Chiba University)

Introduction

Gravitational N-body simulations and Methods

Gravitational N-body simulations numerically solve particle motion under mutual gravitational interactions.

Computational approaches:

- **Direct summation**：$O(N^2)$ complexity → impractical for large N
- **Particle Mesh (PM)[1]**： Approximate potential on uniform grid via FFT
- **TreePM[2]**：PM + Tree → widely used in cosmological simulations

FFT parallelization efficiency determines overall PM performance at scale.

Problem and Motivation

FFT libraries require specific data decompositions (slab, pencil, cube), but these differ significantly from the highly non-uniform domain decomposition optimal for tree methods[3].

➞ Additional data reshaping is required, introducing communication overhead.

This communication cost can be comparable to FFT computation time itself, meaning **simply improving FFT computational efficiency does not guarantee better overall performance**

Comprehensive performance evaluation including communication costs is needed to identify optimal FFT implementations at each scale.

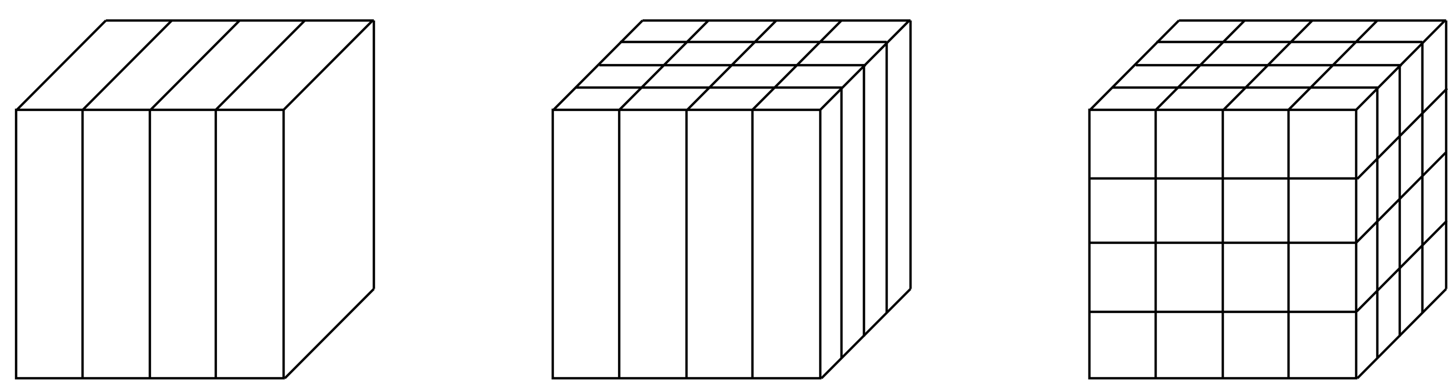
Method

Overview of Particle Mesh Method

1. Assign particle masses to a uniform grid using Cloud-in-Cell (CIC) method to obtain density distribution ρ.
2. Solve Poisson's equation $\nabla^2\phi = 4\pi G\rho$ in Fourier space via FFT to compute gravitational potential ϕ :
$$\tilde{\boldsymbol{\phi}} = -\frac{4\pi G}{k^2}\tilde{\boldsymbol{\rho}}$$
(G: gravitational constant, k: wave vector magnitude)
3. Transform back to real space via inverse FFT and interpolate forces to particle positions using CIC.
4. Update particle positions and velocities from computed forces.

Steps 1 through 4 are iterated to advance the simulation.

Differences in FFT Input Data Structures



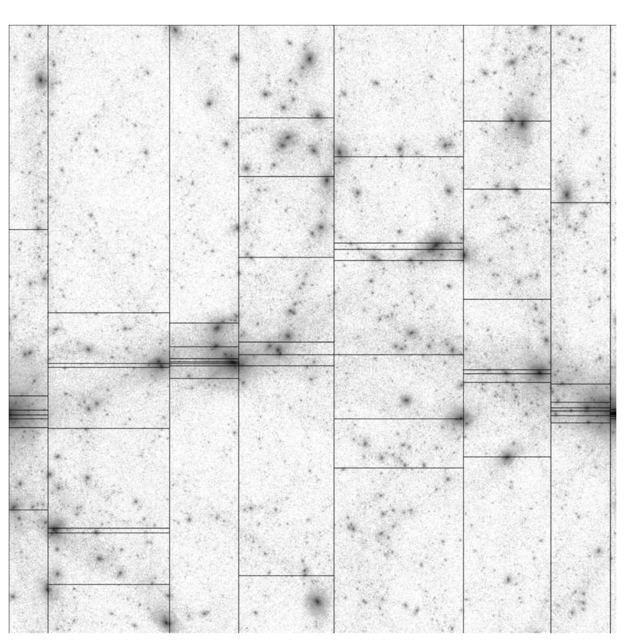
FFT libraries employ different parallelization structures: slab, pencil, or cube decomposition.

*heFFTe accepts 3D input but internally performs FFT in pencil configuration.

Particle Distributions Used

Two types of particle distributions were used in this study:

1. **Uniform distribution:** Particles are evenly distributed, corresponding to early universe conditions.
2. **Non-uniform distribution:** Uses actual cosmological simulation data, representing late-stage universe conditions where particles are clustered.
   Box size: 1.0 Gpc/h
   Redshift: $z = 0$



(Ishiyama et al. 2009)

Setup

Number of particles: 4096^3 Grid sizes: 2048^3 and 4096^3

System: Supercomputer Fugaku

- 48 cores per node, 2.0 GHz
- Total nodes: 158976
- Nodes used in this study: 128-2048
- Configuration: 12 threads, 4 processes per node

Results

Uniform distribution（Grid size：2048^3）

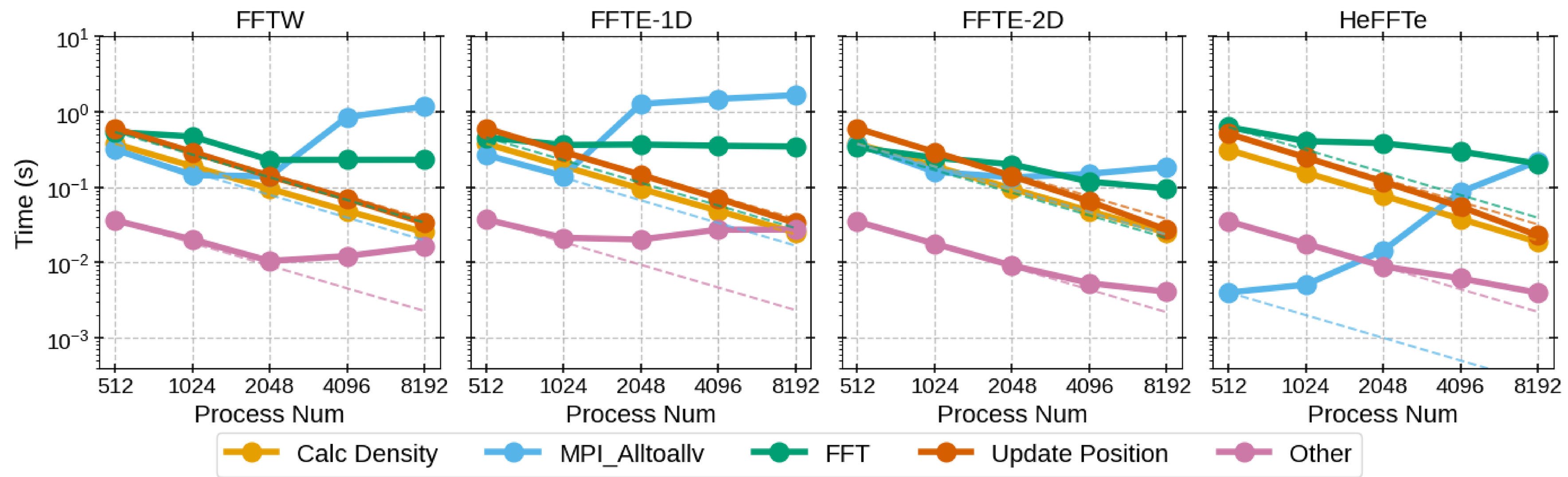


Fig1:Execution time breakdown for uniform distribution. Solid and dashed curves show actual and ideal scaling.

- **FFTW & FFTE-1D:** Slab decomposition limits maximum processes, causing performance plateau.
- **FFTE-2D:** Higher process limit enables continued scaling.
- **heFFTe:** Additional overhead from data conversion to pencil format.

Non-uniform distribution（Grid size：2048^3）

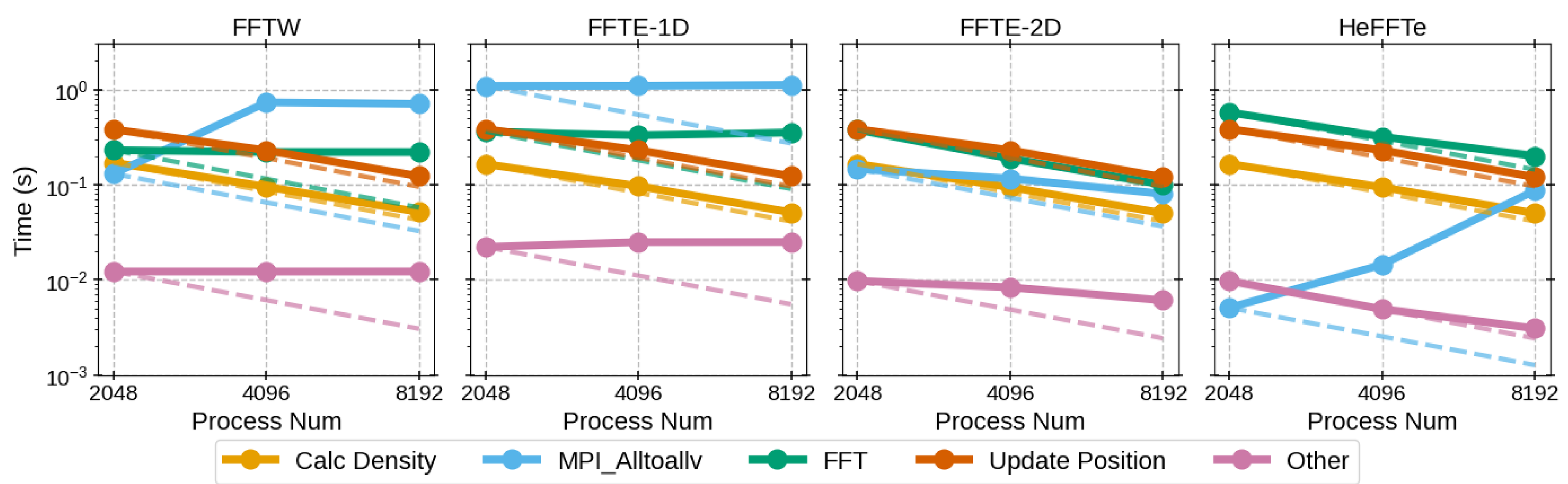
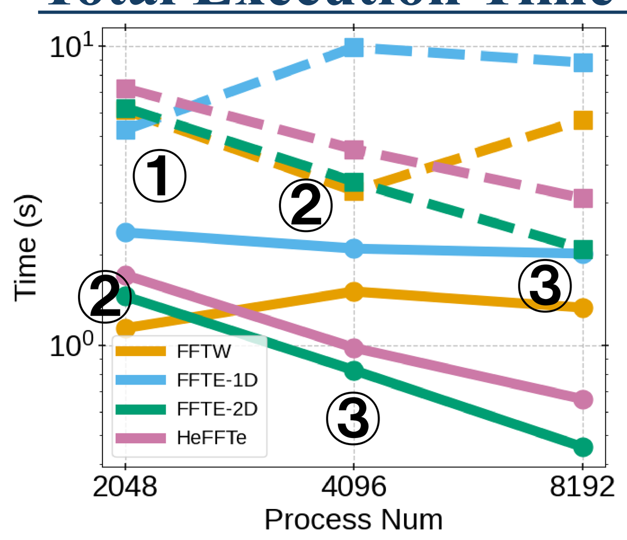


Fig2:Execution time breakdown for non-uniform distribution. Solid and dashed curves show actual and ideal scaling.

- Process range is limited due to particle imbalance in non-uniform distribution
- Execution time increases compared to uniform case
- Scaling efficiency slightly degrades due to non-uniformity

Total Execution Time Comparison



Optimal FFT Implementation by Scale:

1. Processes < Ng: FFTE-1D is fastest
2. Processes = Ng: FFTW is fastest
3. Processes > Ng: FFTE-2D is fastest

(Ng: grid size)

Fig. 3: Total execution time (non-uniform distribution)
Solid: 2048^3, dashed: 4096^3

Summary

Slab decomposition plateaus at process count = grid size, while pencil decomposition scales beyond this limit.

Optimal choice depends on scale:

P < Ng: FFTE-1D, P = Ng: FFTW, P > Ng: FFTE-2D

Future work includes evaluation of heFFTe on GPU-based supercomputers, where it is optimized. Additionally, implementing the relay mesh method[2] will improve scalability of data redistribution for FFT.

References

[1] R. W. Hockney, J. W. Eastwood, "Computer Simulation Using Particles", 1988.
[2] T. Ishiyama, "GreeM: Massively Parallel TreePM Code for Large Cosmological N-body Simulations", PASJ, vol. 61, no. 6, pp. 1319-1330, 2009.
[3] T. Ishiyama et al., "4.45 Pflops astrophysical N-body simulation on K computer", SC '12, 2012.
[4] M. Frigo, S. G. Johnson, "FFTW: An adaptive software architecture for the FFT", ICASSP '98, 1998.
[5] D. Takahashi, "FFTE: A Fast Fourier Transform Package", IEICE Trans., vol. E80-A, 1997.
[6] A. Ayala, S. Tomov, A. Haidar, and J. Dongarra, "heFFTe: Highly Efficient FFT for Exascale", *ICCS 2020*, LNCS 12143, 262 (2020)