CUBE – Towards an Optimal Scaling of Cosmological N-body Simulations

Shenggan Cheng*, Hao-Ran Yu†, Derek Inman‡, Qiucheng Liao*, Qiaoya Wu† and James Lin*
*Center for High Performance Computing, Shanghai Jiao Tong University, Shanghai, 200240, China
Email: {chengshenggan, keymorrislane, james}@sjtu.edu.cn
†Department of Astronomy, Xiamen University, Xiamen, Fujian 361005, China
Email: haoran@xmu.edu.cn, wuqiaoya@stu.xmu.edu.cn
‡Center for Cosmology and Particle Physics, Department of Physics, New York University, New York, 10003, USA
Email: derek.inman@nyu.edu

***Abstract*—N-body simulations are essential tools in physical cosmology to understand the large-scale structure (LSS) formation of the universe. Large-scale simulations with high resolution are important for exploring the substructure of universe and for determining fundamental physical parameters like neutrino mass. However, traditional particle-mesh (PM) based algorithms use considerable amounts of memory, which limits the scalability of simulations. Therefore, we designed a two-level PM algorithm CUBE towards optimal performance in memory consumption reduction. By using the fixed-point compression technique, CUBE reduces the memory consumption per N-body particle to only 6 bytes, an order of magnitude lower than the traditional PM-based algorithms. We scaled CUBE to 512 nodes (20,480 cores) on an Intel Cascade Lake based supercomputer with $\simeq$95% weak-scaling efficiency. This scaling test was performed in Cosmo-π – a cosmological LSS simulation using $\simeq$4.4 trillion particles, tracing the evolution of the universe over $\simeq$13.7 billion years. To our best knowledge, Cosmo-π is the largest completed cosmological N-body simulation. We believe CUBE has a huge potential to scale on exascale supercomputers for larger simulations.**

***Index Terms*—N-body; particle-mesh method; large-scale simulation; fixed-point compression; mixed-precision calculation;**

I. Problem Overview

In astrophysics, N-body simulations are used to study the dynamics of globular clusters, galaxy evolutions, galaxy and intergalactic interactions and cosmology [1]. Cosmological parameters are measured to the percent or even sub-percent levels. This precision requires equally accurate numerical modeling of the *large scale structure* (LSS) formation and might lead to cosmological N-body simulations with a very large N. For example, for studies of dark matter and dark energy, we need to resolve the structures and assembly histories of faint galaxies in a cosmological volume matching modern spectroscopic galaxy surveys. This requires a mass resolution of N-body particles $M_p \sim 10^8 M_\odot$ and physical scale of the simulation $L \sim 1000\,\mathrm{Mpc}/h$. Another example is the study of neutrino mass using cosmological effects. To model small but nonlinear cosmic neutrino background, we need a large N to suppress the Poisson noise (e.g., [2]). These problems need a dynamical range of 4 to 5 orders of magnitude, corresponding to, at least, 10^{12} (trillion) particle N-body simulations.

A direct-summation algorithm is unaffordable for a large N because the computational complexity of such a *pairwise force calculation* (PP) is $O(N^2)$. Many algorithms have been designed to reduce the complexity to $O(N \log N)$, such as the *particle-mesh methods* (PM) or *tree methods*, or even $O(N)$, such as the *fast multipole method*. Fortunately, cosmological N-body simulations rarely require accurate trajectories of individual particles, but rather just correct statistical distributions. In this scenario, the PM-based algorithms usually meet the accuracy requirements and have a potential to simulate with a very large N.

However, achieving the largest N with the PM-based algorithms is typically bound by memory capacity, instead of computing capacity. For example, *TianNu* [2], one of the world's largest N-body simulations, used the P^{3}M algorithm [3] to complete an $N \simeq 3 \times 10^{12}$ cosmological N-body simulation on the *Tianhe-2 supercomputer*. The simulation used all the memory of Tianhe-2, but only 30% of its computing resource[1]. The challenge to further increase the scale of N-body simulations is to reduce its memory consumption.

To tackle this challenge, we designed a *two-level PM* (PMPM) algorithm CUBE [4] to obtain the lowest memory consumption possible. N-body simulations typically require considerable amounts of memory to store the positions and velocities of N-body particles. These six numbers occupy 24/48 *bytes per particle* (`bpp`) if they are stored as single/double precision floating numbers. By using an information-optimized algorithm, fixed-point format can be used to store the phase-space information of particles in memory. This *fixed-point compression* can significantly minimize the memory footprint toward only 6 `bpp`, thus break the memory capacity bound for scaling large N-body simulations.

In this work, we implemented the CUBE algorithm in `C` and optimized the `C` code for performance and scalability. We then scaled the optimized code on the Intel Cascade Lake based supercomputer π 2.0, to 512 nodes (20,480 cores) with $\simeq$95% weak-scaling efficiency. This cosmological simulation, Cosmo-π, evolved $\simeq$4.4 trillion *cold dark matter* (CDM) par-

[1]TianNu used only CPUs in Tianhe-2 which contribute to about 30% of the computing resource. The rest are contributed from the coprocessors (Intel Knights Corner, KNC) which associate with small, independent memory, and TianNu did not use.

ticles from *redshift*[2] 99 to 0 in a *cubic comoving volume* $L = 3,200\,\mathrm{Mpc}/h$ per side. To our best knowledge, Cosmo-π is the largest completed cosmological N-body simulation.

II. Application Scenario

High-performance N-body simulation codes must address many challenges – maintaining a low memory footprint given a large N, minimizing the communication across computing nodes, reducing and accelerating the memory accesses to large arrays, and efficiently using high-performance libraries to speed up standard calculations such as FFTs. A series of cosmological N-body codes, including "Moving-PM" [5], PMFAST [6], CUBEP^{3}M [3], and CUBE [4], are designed especially for weak-scaling on supercomputers[3]. Our scaling test is based on our continuous development of CUBE, aiming for optimizations on all the above aspects.

CUBE solves the gravitational force using the PMPM algorithm, with optional extended-PP force modules for increased accuracy. The traditional PM-based algorithm is suboptimal in parallel computing as it requires a full resolution parallel FFT. The PMPM algorithm solves this problem by splitting the gravitational force F_G into a *short-range force* F_s and a *long-range force* F_l, $F_G = F_s + F_l$. The short-range force F_s has a cutoff, i.e., $F_s(r \geq r_{\mathrm{cutoff}}) = 0$, and the gravity beyond the cutoff is fully provided by the long-range force, $F_l(r \geq r_{\mathrm{cutoff}}) = F_G(r)$. When $r < r_{\mathrm{cutoff}}$, both force components contribute to F_G. In particular, for smaller r, F_G is gradually dominated by F_s. The force-matching optimizations under different criteria are discussed in Ref. [3], [6].

For either F_s or F_l, the force calculation is a convolution in real space, equivalent to multiplying the density field with component-wise force kernels in Fourier space,

$$\tilde{F}^i_{s/l}(\boldsymbol{k}) = \tilde{\rho}(\boldsymbol{k})\tilde{K}^i_{s/l}(\boldsymbol{k}), \tag{1}$$

where ρ is the density field obtained by interpolating particles onto a mesh (mass assignment), K is the force kernel, a tilde $\tilde{}$ indicates that the variable is in Fourier space, and $i = 1, 2, 3$ corresponds to three spatial dimensions. The PMPM algorithm essentially computes the short-range ($_s$) and long-range ($_l$) components of Eq.(1) separately. The two PM algorithms are applied on two meshes with different resolutions.

For the long-range force, because $F_l(r \to 0) = 0$ it requires a lower resolution – F_l is usually computed on a 4-times-coarser mesh (coarse-mesh).

In contrast, for F_s, the short-range PM is computed on full-resolution fine-meshes. Because $F_s(r > r_{\mathrm{cutoff}}) = 0$, we can divide the simulation volume L^3 into many cubic sub-volumes $V_{\mathrm{tile}} = (L/N_{\mathrm{tile}})^3$, called *tiles*, and F_s on each V_{tile} is computed by a fine-mesh PM on a slightly enlarged volume $V_{\mathrm{PM-fine}} = (L/N_{\mathrm{tile}} + 2r_{\mathrm{cutoff}})^3$. So F_s in L^3 is done locally using N^3_{tile} local FFTs.

If the PP force is implemented, the gravity between particles within a preset number of adjacent neighboring fine-cells are computed explicitly, and the short-range force kernel K^i_s is further corrected accordingly.

Thus, only the coarse-mesh FFTs require global MPI communications[4]. The coarse-mesh PM scales as $O(N \log N)$ but with 1/4 of the full-resolution, so it typically consumes negligible CPU time. This makes the PMPM algorithm excellent for weak-scaling problems as computing time scales like $O(N)$, because an increase of the problem size leads to a proportional increase of N^3_{tile} when V_{tile} is fixed. It also dramatically reduces memory footprint requirements as the coarse grid has $4^3 = 64$ times fewer cells.

Computationally, the global simulation volume is first broken down into small cubical sub-volumes, each of which is assigned to one MPI process. A second level of cubical decomposition occurs inside each process, where the node sub-volumes are broken into a number of local volumes called tiles as mentioned above. Inside an MPI process, the tiles are calculated in turn, and using OpenMP parallelizes the calculations within the tile.

III. Solution

A. Fixed-point Compression

The PMPM algorithm is intrinsically memory efficient, and the memory consumption is thus dominated by the phase-space coordinates of particles. CUBE is information-optimized and further reduce this memory footprint by using fix-point formats instead of floating formats.

This format is based on the structure of the coarse-mesh. For the i-th dimension, instead of using a 4/8-byte floating number storing each particle's global coordinate, $x^i \in [0, L)$, we store its relative position w.r.t. its parent-cell in the coarse-mesh. In particular, if L is divided by N_c coarse-cells per dimension, we further divide each coarse-cell into 256 bins, and for a particle in the n^i_c-th cell, its coordinate in the i-th dimension can be expressed by

$$x^i = x^i_c + \Delta x^i = (n^i_c - 1)L/N_c + (m^i + 1/2)L/(256N_c) \tag{2}$$

where x^i_c is the left boundary of the cell, and Δx^i is the relative position. $m^i \in \{0, 1, ..., 255\}$ can be stored as an 1-byte integer. To have the correct n^i_c, particles' m^i must be stored in memory in a cell-ordered way. A complementary number count of particle numbers in this mesh (density field) will give complete information on the particle distribution in the mesh.

The particle velocities in each dimension v^i is decomposed as

[2]The cosmological redshift z is an indicator of cosmic time in an expanding universe. The scale factor a of the universe satisfies $a \propto 1/(1+z)$.

[3]They introduced, for example, the 2-level particle-mesh (PMPM) algorithm [6], optimized on cubic decompositions [3] and fixed-point information optimization technique [4].

[4]Besides computing the gravitational force, MPI communications are also used to send/receive particles and to synchronize density/velocity fields.

$$v^i = v_c^i + \Delta v^i = v_c^i(n_c^i) + f(u^i, z, L/N_c) \quad (3)$$

where $v_c^i = \langle v^i \rangle$ is the velocity field averaged on the n_c-th cell, and Δv^i is each particle's relative velocity w.r.t. v_c^i. Similarly, u^i can be stored as an 1-byte integer, and maps to Δv^i by a nonlinear function $f(u^i, z, L/N_c)$. The nonlinearity is due to the fact that the probability distribution of Δv^i is not uniform but is approximated by a Gaussian distribution with its variance σ_Δ^2 linearly predictable as a function of redshift z and smoothing scale L/N_c. In practice, f being the form similar to the inverse error function (the cumulative distribution function of Gaussian) minimizes the error in the velocity conversion, because it better solves the dominant slower moving particles. σ_Δ^2 becomes nonlinear at lower redshifts, and can be directly measured from simulations.

For each particle, $m^i, u^i\,(i = 1, 2, 3)$ are stored by six 1-byte integers. n_c^i and v_c^i information are provided by the density and velocity fields on the coarse-mesh. Usually each coarse-cell contains large number (64 in Cosmo-π) particles so the auxiliary density and velocity fields consumes negligible additional memory [4]. m^i and u^i can either or both stored as 2-byte integers, corresponding to a more accurate position/velocity storage. More detailed discussions and results are in Ref. [4].

B. Optimizations

1) Precompute Expensive Functions: The fixed-point compression dramatically lowers the required memory footprint of N-body simulations; however, the data compression and decompression introduce extra computing costs. The profiling results showed 21% of total elapsed time is spent on them. For example, calculating Eq.(3) is very time-consuming due to the expensive math functions in f. To implement them efficiently, we precomputed their value and stored them into arrays. As the number of compression fixed-point representation is limited, 256 for 1-byte integer, the time and memory cost of the precomputing are negligible.

2) Approximate Expensive Functions: As the fixed-point compression casts a float (32 bits) to just an 1- or 2-byte integer, the high accuracy of some expensive math functions is redundant. Therefore, we use approximate functions to replace them without losing the accuracy of final scientific results. For example, to accelerate the expensive $\arctan(x)$ function (which is used in Ref. [4] to approximate velocity compression function), we modified approximate function [7] and expanded its domain (Eq. 4). This fast approximate function has a maximum absolute error of 0.0038 radians, which has no effects on the accuracy of final results.

$$arctan(x) \approx \begin{cases} \frac{\pi}{4}x + 0.273x(1 - |x|), |x| \le 1 \\ \frac{\pi}{2} - \frac{\pi}{4x} - \frac{0.273}{x}(1 - |\frac{1}{x}|), x > 1 \\ -\frac{\pi}{2} - \frac{\pi}{4x} - \frac{0.273}{x}(1 - |\frac{1}{x}|), x < -1 \end{cases} \quad (4)$$

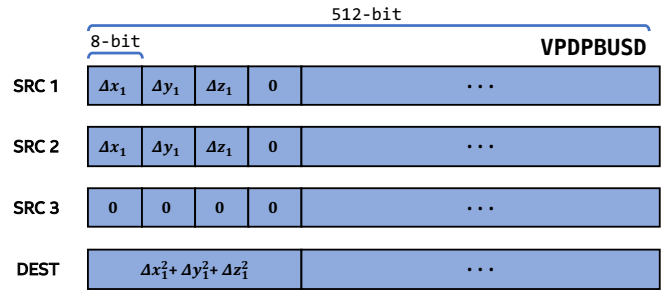

Fig. 1. Using Intel AVX512 VNNI instructions to calculate the PP force kernel in mixed-precision. Putting 16 relative vectors of x-y-z axes delta and filling with zero in the `SRC1` and `SRC2`, then getting the 16 pairs of particles with the `VPDPBUSD` and `VSQRTPS` instructions.

3) Compute PP Force Kernel in Mixed-precision: The PP force kernel requires calculating Euclidean distances between each pair of particles within a certain range. As the particles positions are compressed in integers, we cannot use traditional AVX512 vectorization instructions. Therefore, we adopt the new *AVX512 VNNI (Vector Neural Network Instructions) extension* [8] (shipped with Intel Cascade Lake processors and after) to compute the compression position for gravity calculation in *mixed-precision*. As shown in Fig. 1, the AVX512 VNNI `VPDPBUSD` instruction multiplies the individual bytes (8-bit) of the first source operand by the corresponding bytes (8-bit) of the second source operand, producing intermediate word (16-bit) results which are summed and accumulated in the double word (32-bit) of the destination operand. Theoretically, using the `VPDPBUSD` instruction can increase 3 times of operations throughput and use 3 times less memory footprint than scalar instructions.

4) Distribute MPI Processes: The buffer communication requires the positions and velocities of particles to be transferred between adjacent grids. To reduce the overhead of buffer communications, as illustrated in Fig. 2, we use *3D MPI rank distribution* to ensure the processes with adjacent simulation volumes are nearby in the physical domain.

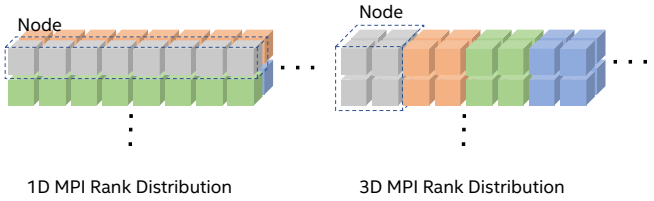

Fig. 2. The visual presentation of 1D and 3D MPI Rank Distributions. Each cube represents the physical region for one process. Processes in the same node have the same color.

IV. Performance Metrics and Results

A. Experimental Setup

1) Hardware & Software: We evaluated all scaling tests on supercomputer π 2.0, which has 650 computing nodes with 2PFlops peak performance. Each node has two Intel

Cascade Lake 6248 processors (20 cores for each socket) and 192GB DDR4 memory. All nodes are interconnected with Intel 100Gbps Omni-Path Architecture (OPA).

We compiled CUBE with Intel C/C++ Compiler 18.0.5. The FFT and MPI are supported by Intel MKL 2018 (Update 4) and Intel MPI 2018 (Update 4), respectively.

2) Simulation Parameters: We use 4,096 MPI processes on 512 nodes ($\sim 80\%$ of the full system of π 2.0) to evolve $16384^3 (\approx 4.39 \times 10^{12})$ cold dark matter particles in a $(3.2\ \mathrm{Gpc}/h)^3$ cosmological volume. We use the *Zel'dovich Approximation* [9] to determine the initial positions and velocities of particles at redshift $z = 99$ and then use CUBE to evolve the particles to $z = 0$. The simulation models a *ΛCDM universe* with Hubble parameter $H_0 = 100h\ \mathrm{km\,s^{-1}\,Mpc^{-1}} = 72\ \mathrm{km\,s^{-1}\,Mpc^{-1}}$, CDM density $\Omega_c h^2 = 0.214 \times 0.72^2 = 0.1109$, baryon density $\Omega_b h^2 = 0.044 \times 0.72^2 = 0.0228$ and initial conditions characterized by $\sigma_8 = 0.80$ and $n_s = 0.96$. For fixed-point compression, we use the 1-byte fixed-point format to store the particle phase space.

The PM-PM force calculation is computed using a coarse mesh composed of 256^3 cells per process (4096^3 in total). Each process volume is decomposed into 8 tiles resolved by $512+2\times6 = 524$ (512 for physical volume and 2×6 for buffer region) fine cells per dimension. This geometry is equivalent to a regular PM calculation with a 16384^3 fine cell mesh (or 1024^3 fine cells per process), but utilizing 8 times less memory footprint and substantially faster due to the decreased size of the global FFT.

B. Performance Metrics

We choose bytes per particle (`bpp`) and wall clock time as the two performance metrics for the scaling tests. We use the first metric to measure the memory consumption for each MPI process or each node by using `/proc/self/statm` and `MPI_Allreduce`; we use the second one to measure code speeds by using `MPI_Wtime`.

C. Performance Results

1) Memory Consumption: Tab. I lists the memory consumption for one MPI process of CUBE, which includes three parts: `Particle`, `Coarse mesh`, and `Fine mesh`. `Particle` includes three components: `Physical region` and `Image buffer` are used to store the particle phase space in the physical domain and buffered region whereas `Tile buffer` is the temporary buffer for each tile. `Coarse mesh` and `Fine mesh` are the arrays associated with the coarse mesh and fine mesh, respectively. In CUBE, each MPI process requires 13.75GB memory to simulate $1024^3 (= 1.074 \times 10^9)$ particles. The `bpp` of CUBE is thus $13.75 \times 10^9/(1.074 \times 10^9) = 12.8$.

Due to using fixed-point compression, CUBE has significantly smaller `bpp` than any other cosmological N-body simulation codes. For example, TianNu [2] simulates 2.97 trillion particles on Tianhe-2. Each Tianhe-2 node holds an average of 576^3 neutrino particles and 288^3 CDM particles, and uses 40GB memory. Its `bpp` is thus $(576^3 + 288^3)/(40 \times 10^9) = 186$, which is 14.5 times larger than CUBE's `bpp`.

Table I. Memory consumption for one MPI process to simulate 1024^3 particles.

Parts		Memory Consumption		
		/GB	/bpp	Percentage
Particles	Physical region	7.3	6	53%
	Image buffer	1.55	2.24	11.36%
	Tile buffer	1.11	1.03	8.04%
Coarse mesh		2.38	2.21	17.30%
Fine mesh		1.32	1.22	9.56%
Total		13.75	12.8	100.00%

2) Scalability: To study the weak-scaling of CUBE, we allow each process to evolve a 200 $\mathrm{Mpc}\,h^{-1}$ volume using 1024^3 fine cells and gradually scale from 40 cores to 20,480 cores. Fig. 3(a) shows CUBE's weak-scaling result both with and without the PP force (PM-PM-PP and PM-PM in the legend). We see an almost perfect linear speed achieving 95% parallel efficiency in both cases. For comparison, the TianNu simulation had 72% weak-scaling efficiency [10]; although we note that this scaling test was done at redshift $z = 5$ where nonlinear structure substantially increases iterations of the PP force kernel.

While often less relevant for cosmological applications, we also report CBUE's strong-scaling result. We evolved a fixed global simulation volume of 200 $\mathrm{Mpc}\,h^{-1}$ with 1024^3 fine cells, but slowly increased the number of cores and distributed corresponding fine cells per process. Fig. 3(b) shows the strong-scaling where we can see around 81.5% and 84.2% parallel efficiency at 20,480 cores, respectively. The degradation in performance with increasing cores comes from two aspects: the decrease in computation to communication

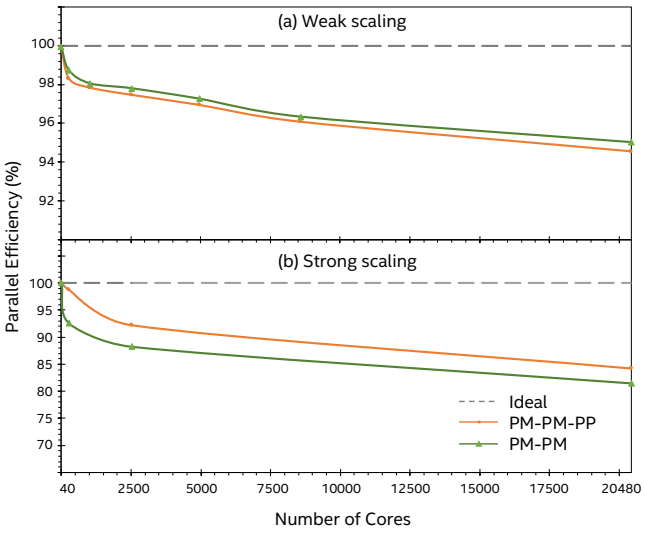

Fig. 3. Weak-scaling (a) and strong-scaling (b) from 40 to 20,480 cores. We show the parallel efficiency against core count for PM-PM-PP and PM-PM along with the ideal efficiency.

time, and the increase in the ratio of the buffer region to physical volume.

D. Validation

We validate the correctness of the Cosmo-π simulation visually and statistically. Fig. 4 illustrates the LSS distribution of CDM in a thin slice of the simulation volume. The N-body particles in Cosmo-π at the final checkpoint $z = 0$ are interpolated onto regular grids by the *cloud-in-cell* (CIC) method. The resulting 3D density field ρ is then partly projected onto the plane of Fig. 4 where dark/light colors show high/low column densities. The size of the figure corresponds to the box size $L = 3200\,\mathrm{Mpc}/h$ of the simulation, and the thickness of the slice is chosen to clearly visualize the structure of the cosmic web – nodes, filaments, and voids of certain scales. The hierarchically zoomed-in panels show more detailed CDM structures on smaller scales and the most zoomed-in panel shows the projected N-body particles' distribution directly.

A useful statistics of LSS is the *power spectrum*, corresponding to the Fourier transform of the two-point correlation function. From the CDM density field ρ we define the dimensionless density contrast $\delta \equiv \rho/\bar{\rho} - 1$, from which the power spectrum $P(k)$ is given by $\langle \delta^\dagger(\boldsymbol{k})\delta(\boldsymbol{k}')\rangle = (2\pi)^3 P(k)\boldsymbol{\delta}_{3\mathrm{D}}(\boldsymbol{k} - \boldsymbol{k}')$, where $\boldsymbol{k}$ is the Fourier wave vector with $k = |\boldsymbol{k}|$ and $\boldsymbol{\delta}_{3\mathrm{D}}$ is the 3D Dirac delta function. We usually plot the dimensionless power spectrum $\Delta^2(k)$, defined by $\Delta^2(k) \equiv k^3 P(k)/(2\pi^2)$. Cosmo-$\pi$ store checkpoints at various of redshifts and in Fig. 5 we plot their dimensionless power spectra compared with their linear and nonlinear predictions. Note that at high redshifts (e.g., the initial condition of Cosmo-π, $z = 99$), the LSS statistics are well described by linear theory; while at low redshifts, the linear/nonlinear power spectra[5] mismatch and we see $\Delta^2(k)$ follows the nonlinear predictions. The power spectrum suppression at high wavenumber corresponds to the limited sub-grid force resolution, which can be improved by using the extended PP force. The result here is consistent with Ref. [4].

V. Related Approaches

The gravitational N-body simulation is a critical tool for understanding modern cosmology that relies on supercomputing. The past ten years has seen the rise of trillion particle simulations starting with the Gordon-Bell Prize winning simulation [13] and a finalist [14]. Tab. II compares the large-scale cosmological N-body simulations in recent years. There are now several different codes that have been used to exceed the trillion particle mark, which differ primarily in their implementation of the gravitational force computation. Our work evolved 4.39 trillion particles and thus, to our best knowledge, is the largest cosmological N-body simulation.

[5] According to LSS formation theories, at first order, linear approximations where $|\delta| \ll 1$, Fourier modes evolve independently and $P(k)$ is scaled by a growth factor $D(a)$, independent of k. At lower redshifts, nonlinearities cannot be neglected and we use various nonlinear predictions (e.g., [11], [12]) for the power spectrum. The exact LSS evolution can only be accurately modeled by N-body simulations.

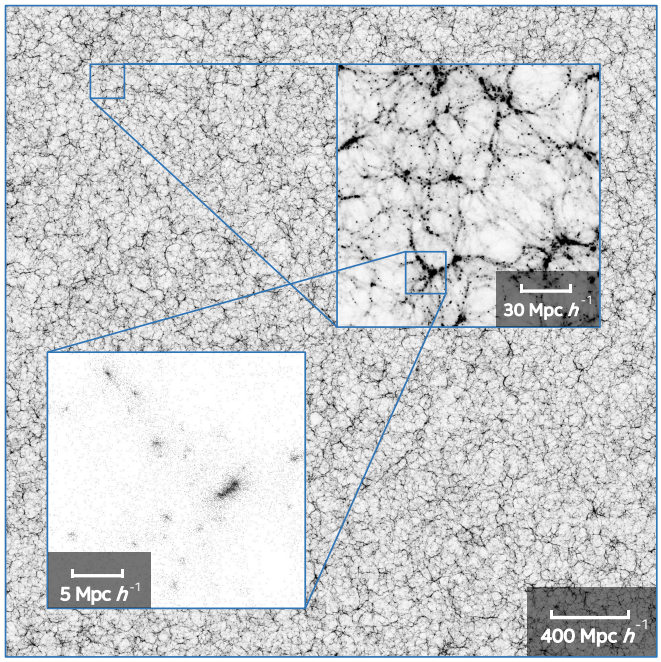

Fig. 4. Two-dimensional visualization of the CDM structures in Cosmo-π at redshift $z = 0$. A slice of volume $3200 \times 3200 \times 20\,(\mathrm{Mpc}\,h^{-1})^3$ is shown, while sub-panels show zoomed-in structures. The high/low column densities are rendered by black/white, while the most zoomed-in panel shows the direct projection of CDM N-body particles.

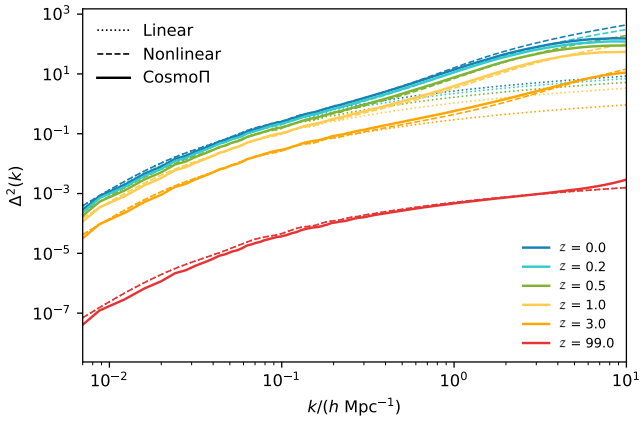

Fig. 5. Statistical validation of Cosmo-π. We show the dimensionless power spectra $\Delta^2(k)$ at redshifts $z = 0, 0.2, 0.5, 1, 3, 99$ as well as their linear and nonlinear predictions. The range of k is chosen to show the transition between linear and nonlinear scales.

VI. Impact of the Solution

We summarize the impact of the Cosmo-π simulation in three aspects. First, Cosmo-π is, to the best of our knowledge, the largest completed cosmological N-body simulation, evolving 4.39 trillion particles from redshift 99 to 0. Simulations such as Cosmo-π, as well as higher resolution ones using the PP force, will allow for improved LSS statistics and better understanding of halo assembly and substructure.

Second, we believe CUBE has a huge potential for large-scale cosmological simulations. Cosmo-π was able to evolve

Table II. Comparison of the large-scale cosmological N-body simulations on supercomputers. For each simulation, we summarize the computational scale in CPU cores (c) or GPU cards (g), the problem scale in the particles number (N_p), redshift range (z_i,z_f), box size (L), method used to compute the force, and force resolution (ϵ). To the best of our knowledge, this work is the largest cosmological N-body simulation (4.39 trillion particles).

Simulations	Years	Codes	Supercomputers	Scale	$N_p(\times 10^{12})$	z_i, z_f	L (Gpc/h)	Force	ϵ (kpc/h)
Dark Sky [15]	2014	2HOT	Titan	12,288g [1]	1.074	$93, 0$	8	Tree	36.8
ν^2GC [16]	2015	GreeM^3	K Computer	131,072c	0.55	$127, 0$	1.12	TreePM	4.27
Q Continuum [17]	2015	HACC	Titan	16,384g [1]	0.55	$200, 0$	0.923	P^{3}M	2
TianNu [2]	2017	CUBEP^{3}M	Tianhe-2	331,776c	2.97	$5, 0$	1.2	PM-PM-PP	13
Euclid Flagship [18]	2017	PKDGRAV3	Piz Daint	>4,000g [1]	2.0	$49, 0$	3	FMM	4.8
Outer Rim [19]	2019	HACC	MIRA	524,288c	1.074	$200, 0$	3	TreePM	2.84
Cosmo-π (**This work**)	2019	CUBE	π 2.0	20,480c	**4.39**	$99, 0$	3.2	PM-PM	195

1. These three simulations were carried out using NVIDIA Tesla K20X.

4.39 trillion particles using just 20,480 cores, a substantial improvement in N enabled by CUBE's memory consumption optimization. In the next few years, exascale supercomputers will be available which will allow for simulations using more than ten million cores [20] [21] , increasing the problem scale by at least three orders of magnitude. This will have a profound impact on studies of LSS and other astronomical simulations.

Finally, we show fixed-point compression and mixed-precision calculation can be extremely valuable tools for scientific applications. We expect traditional HPC applications will benefit more from emerging AI computing technologies.

ACKNOWLEDGMENT

We thank HPC Center of Shanghai Jiao Tong University for providing computing resource and excellent technical support. This research is partially supported by The National Key Research and Development Program of China 2016YFB0201800. Hao-Ran Yu acknowledges National Science Foundation of China No.11903021. Shenggan Cheng and Hao-Ran Yu contributed equally to this paper. James Lin is the corresponding author.

REFERENCES

[1] R. W. Hockney and J. W. Eastwood, *Computer simulation using particles*, 1988.

[2] H.-R. Yu, J. D. Emberson, D. Inman, T.-J. Zhang, U.-L. Pen, J. Harnois-Déraps, S. Yuan, H.-Y. Teng, H.-M. Zhu, X. Chen, Z.-Z. Xing, Y. Du, L. Zhang, Y. Lu, and X. Liao, "Differential neutrino condensation onto cosmic structure," *Nature Astronomy*, vol. 1, p. 0143, Jul. 2017.

[3] J. Harnois-Déraps, U.-L. Pen, I. T. Iliev, H. Merz, J. D. Emberson, and V. Desjacques, "High-performance P^{3}M N-body code: CUBEP^{3}M," *Monthly Notices of the Royal Astronomical Society*, vol. 436, pp. 540–559, Nov. 2013.

[4] H.-R. Yu, U.-L. Pen, and X. Wang, "CUBE: An Information-optimized Parallel Cosmological N-body Algorithm," *The Astrophysical Journal Supplement Series*, vol. 237, p. 24, Aug. 2018.

[5] U.-L. Pen, "A Linear Moving Adaptive Particle-Mesh N-Body Algorithm," *The Astrophysical Journal Supplement Series*, vol. 100, p. 269, Sep. 1995.

[6] H. Merz, U.-L. Pen, and H. Trac, "Towards optimal parallel PM N-body codes: PMFAST," *New A*, vol. 10, pp. 393–407, Apr. 2005.

[7] S. Rajan, S. Wang, R. Inkol, and A. Joyal, "Efficient approximations for the arctangent function," *IEEE Signal Processing Magazine*, vol. 23, no. 3, pp. 187–198, 2006.

[8] A. Rodriguez, E. Sen, E. Meiri, E. Fomenko, Y. Jin, H. Shen, and Z. Barukh, "Lower numerical precision deep learning inference and training," 2018.

[9] Y. B. Zel'dovich, "Gravitational instability: An approximate theory for large density perturbations." *Astronomy and Astrophysics*, vol. 5, pp. 84–89, Mar. 1970.

[10] J. D. Emberson, H.-R. Yu, D. Inman, T.-J. Zhang, U.-L. Pen, J. Harnois-Déraps, S. Yuan, H.-Y. Teng, H.-M. Zhu, X. Chen, and Z.-Z. Xing, "Cosmological neutrino simulations at extreme scale," *Research in Astronomy and Astrophysics*, vol. 17, p. 085, Aug. 2017.

[11] D. Blas, J. Lesgourgues, and T. Tram, "The Cosmic Linear Anisotropy Solving System (CLASS). Part II: Approximation schemes," *J. Cosmology Astropart. Phys.*, vol. 7, p. 034, Jul. 2011.

[12] R. E. Smith, J. A. Peacock, A. Jenkins, S. D. M. White, C. S. Frenk, F. R. Pearce, P. A. Thomas, G. Efstathiou, and H. M. P. Couchman, "Stable clustering, the halo model and non-linear cosmological power spectra," *Monthly Notices of the Royal Astronomical Society*, vol. 341, no. 4, p. 1311–1332, Jun 2003. [Online]. Available: http://dx.doi.org/10.1046/j.1365-8711.2003.06503.x

[13] T. Ishiyama, K. Nitadori, and J. Makino, "4.45 pflops astrophysical n - body simulation on k computer: the gravitational trillion-body problem," pp. 1–10, 2012.

[14] S. Habib, V. Morozov, H. Finkel, A. C. Pope, K. Heitmann, K. Kumaran, T. Peterka, J. Insley, D. Daniel, P. Fasel *et al.*, "The universe at extreme scale: Multi-petaflop sky simulation on the bg/q," *arXiv: Distributed, Parallel, and Cluster Computing*, 2012.

[15] S. W. Skillman, M. S. Warren, M. J. Turk, R. H. Wechsler, D. E. Holz, and P. M. Sutter, "Dark Sky Simulations: Early Data Release," *arXiv e-prints*, p. arXiv:1407.2600, Jul 2014.

[16] T. Ishiyama, M. Enoki, M. A. R. Kobayashi, R. Makiya, M. Nagashima, and T. Oogi, "The ν^2GC simulations: Quantifying the dark side of the universe in the Planck cosmology," *Publications of the ASJ*, vol. 67, no. 4, p. 61, Aug 2015.

[17] K. Heitmann, N. Frontiere, C. Sewell, S. Habib, A. Pope, H. Finkel, S. Rizzi, J. Insley, and S. Bhattacharya, "The q continuum simulation: Harnessing the power of gpu accelerated supercomputers," *Astrophysical Journal Supplement Series*, vol. 219, no. 2, p. 34, 2015.

[18] D. Potter, J. Stadel, and R. Teyssier, "PKDGRAV3: beyond trillion particle cosmological simulations for the next era of galaxy surveys," *Computational Astrophysics and Cosmology*, vol. 4, no. 1, p. 2, May 2017.

[19] K. Heitmann, H. Finkel, A. Pope, V. Morozov, N. Frontiere, S. Habib, E. Rangel, T. Uram, D. Korytov, H. Child, S. Flender, J. Insley, and S. Rizzi, "The Outer Rim Simulation: A Path to Many-core Supercomputers," *Astrophysical Journal Supplement Series*, vol. 245, no. 1, p. 16, Nov 2019.

[20] J. Lin, Z. Xu, L. Cai, A. Nukada, and S. Matsuoka, "Evaluating the sw26010 many-core processor with a micro-benchmark suite for performance optimizations," *Parallel Computing*, vol. 77, pp. 128 – 143, 2018.

[21] L. Cai, Y.-C. Wang, W. Tang, B. Wang, S. Ethier, Z. Liu, and L. James, "Openacc vs the native programming on sunway taihulight: A case study with gtc-p," 09 2018, pp. 88–97.