RAA 2018 Vol. **18** No. **6**, 62(10pp) doi: 10.1088/1674–4527/18/6/62
© 2018 National Astronomical Observatories, CAS and IOP Publishing Ltd.
http://www.raa-journal.org *http://iopscience.iop.org/raa*

Research in Astronomy and Astrophysics

PHoTo*N*s–A parallel heterogeneous and threads oriented code for cosmological *N*-body simulation

Qiao Wang[1], Zong-Yan Cao[1,2], Liang Gao[1,3], Xue-Bin Chi[2], Chen Meng[2,4], Jie Wang[1] and Long Wang[2,4]

[1] Key Laboratory of Computational Astrophysics, National Astronomical Observatories, Chinese Academy of Sciences, Beijing 100101, China; *qwang@nao.cas.cn*
[2] Supercomputing Center, Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China
[3] Institute for Computational Cosmology, Department of Physics, Durham University, Science Laboratories, South Road, Durham Dh1 3LE, England
[4] System Department, Baidu Group Ltd., Beijing 100085, China

Received 2017 November 7; accepted 2018 February 26

Abstract We introduce a new code for cosmological simulations, PHoTo*N*s, which incorporates features for performing massive cosmological simulations on heterogeneous high performance computer (HPC) systems and threads oriented programming. PHoTo*N*s adopts a hybrid scheme to compute gravitational force, with the conventional Particle-Mesh (PM) algorithm to compute the long-range force, the Tree algorithm to compute the short range force and the direct summation Particle-Particle (PP) algorithm to compute gravity from very close particles. A self-similar space filling a Peano-Hilbert curve is used to decompose the computing domain. Threads programming is advantageously used to more flexibly manage the domain communication, PM calculation and synchronization, as well as Dual Tree Traversal on the CPU+MIC platform. PHoTo*N*s scales well and efficiency of the PP kernel achieves 68.6% of peak performance on MIC and 74.4% on CPU platforms. We also test the accuracy of the code against the much used Gadget-2 in the community and found excellent agreement.

Key words: methods: numerical — galaxies: interactions — dark matter

1 INTRODUCTION

During the last decades, cosmological N-body simulation has become an essential tool for understanding the large scale structure of the Universe, due to the nonlinear physical nature of the formation and evolution of cosmic structure. Demanded by future large scale galaxy surveys, the new generation of cosmological simulations requires a huge number of particles and a huge simulation volume in order to obtain good statistics and simultaneously achieve high resolution to resolve faint galaxies that can be observed in galaxy surveys. For example, a recent simulation by Potter et al. (2017) evolved an unprecedented 2 trillion particles in a 3 Gpc^3 volume for the upcoming Euclid survey. To perform such a simulation, an exquisite simulation code is a must. Indeed many cosmological simulation codes have been continuously developed in recent years (Efstathiou et al. 1985; Bertschinger & Gelb 1991; Jing & Suto 2002; Teyssier 2002; Makino et al. 2003; Makino 2004; Wadsley et al. 2004; Springel 2005; Stadel et al. 2009; Bagla & Khandai 2009; Ishiyama et al. 2009; Klypin et al. 2011; Prada et al. 2012; Ishiyama et al. 2012; Warren 2013; Wang et al. 2015; Emberson et al. 2017; Yu et al. 2017).

At the same time, high performance computing architecture has undergone significant changes, and many new high performance computer (HPC) systems have a mixed CPU+GPU architecture rather than the traditional pure CPU one (Makino et al. 1997, 2003). Many existing cosmological simulation codes were developed based on the pure CPU systems and thus are not able to take advantage of the power provided by popular heterogeneous HPC machines (Guo et al. 2011). In this paper, we introduce a novel cosmological simulation code PHoTo*N*s. It has two characteristics; the first feature is that the code is programmed based on threads and the other is that it is highly optimized for heterogeneous architectures, such as Intel Xeon Phi (MIC).

The organization of the paper is as follows. We briefly introduce the background of the physical model in Section 2. The force calculation algorithm is presented in Section 3. Section 4 discusses our parallelization and implementation strategy. In Section 5, we describe the numerical accuracy and performance of PHoTo*N*s.

2 EQUATIONS OF MOTION

In cosmological N-body simulations, the underlying matter of the Universe is usually sampled with collisionless dark matter particles which are only governed by gravity. Thus the Lagrangian of a particle reads

$$\mathcal{L} = \frac{\boldsymbol{p}^2}{2a^2} - \frac{\phi}{a}, \tag{1}$$

where the momentum $\boldsymbol{p} = a\boldsymbol{v}$ and ϕ refers to comoving potential (Peebles 1980), which is determined by Poisson's equation

$$\nabla^2\phi = 4\pi G a^3 \left[\rho_m\left(\boldsymbol{r}, t\right) - \bar{\rho}_m\left(t\right)\right]. \tag{2}$$

The density field ρ can be constructed from discrete mass points

$$\rho\left(\boldsymbol{r}\right) = \sum_{\forall i} \frac{m_i}{a^3}\delta_{\mathbf{D}}\left(\boldsymbol{r} - \boldsymbol{r}_i\right). \tag{3}$$

Here the Dirac delta function δ_D can be modified by considering different mass distributions.

According to the Lagrangian, the motion equations of an individual particle are given by

$$\begin{aligned} \dot{\boldsymbol{r}} &= \frac{\boldsymbol{p}}{a^2}, \\ \dot{\boldsymbol{p}} &= -\frac{\nabla\phi}{a}. \end{aligned} \tag{4}$$

It is apparent that the gravity between two particles satisfies the inverse-square law due to the form of the Poisson equation. The acceleration of a particle is solved by the gradient of potential and pairwise Newtonian gravity in the simulation.

3 FORCE CALCULATION ALGORITHM AND IMPLEMENTATION

The most challenging problem for cosmological N-body simulation is how to compute the gravitational force efficiently and accurately. In PHoTo*N*s we adopt a hybrid scheme to do the task, namely using the Particle-Mesh (PM) algorithm to compute the long range force, and a novel combination of Tree algorithm and direct summation Particle-Particle (PP) to compute the short range force. We describe our force calculation scheme in detail below.

3.1 PM-Tree-PP Method

The PM method (Hockney & Eastwood 1988) is a common and efficient algorithm to solve the Poisson equation (Eq. (2)). In Fourier space, the Poisson equation can be expressed as an algebraic equation, and its solution can be simply obtained by convolution of the Green function under the periodic boundary condition. The Green function of the Poisson equation has a simple form of $-/k^2$, and its 3-point difference in k space reads

$$g_k(l, m, n) = \frac{\pi G \Delta_g^2}{\sin^2\left(\frac{\pi}{N}l\right) + \sin^2\left(\frac{\pi}{N}m\right) + \sin^2\left(\frac{\pi}{N}n\right)}, \tag{5}$$

where Δ_g denotes the width of a mesh, N is the mesh number on one side, and l, m, n denote the discrete wave number. The convolution of density field and Green function is a multiple in Fourier space and can be implemented with a Fast Fourier Transformation (FFT). As FFT works on regular meshes, one needs to assign the density field sampled by a number of discrete particles into a regular mesh. Then the Poisson equation is solved in Fourier space, and an inverse FFT transformation is applied to obtain gravitational potential on meshes. Finally, the acceleration of each particle can be linearly interpolated from the potential meshes.

The force accuracy of the PM method is nearly exact at large scales but drops dramatically within a few mesh sizes (Bagla 2002). In past years many other gravity solver algorithms were invented to improve force accuracy on small scales. Among these, a popular gravity solver is the *Tree method*, introduced by Barnes & Hut (1986), which provides a robust efficiency even for a highly clustered system. In their original approach, the simulation volume is recursively divided into eight smaller subcubes until each subcube only contains one particle. Each subcube contains information about the center of mass and the multiple moments associated with the mass distribution enclosed in the subcube, as well as other information. Considering every particle as a leaf, one can organize the adjoined particles in hierarchical branches (tree nodes) of a tree data structure, so that the gravity from distant tree nodes can be computed as individual mass points, otherwise the closer nodes are opened. The force calculation of a particle is complete when the tree-walk recursively goes though all branches. The time complexity of such a Tree method is $\mathcal{O}(N\log N)$.

Bagla (2002) suggests a hybrid approach to combine the advantages of the Tree and PM methods in order to achieve adequate force accuracy at both large and small scales. The idea is to elaborately compute long range gravity with the PM method and calculate the short range force with the Tree algorithm. The combination of these two parts achieves an accurate model of gravity at all scales. To this end, for each particle, when doing the PM calculation, an additional low-pass Gaussian filter, $1 - \exp(-s^2/2R_s^2)$, is convolved to take out the short range force of the PM calculation. Here s is the spatial separation of two particles, and R_s is a characteristic parameter to control the splitting scale at which the long and short range force calculations take effect. Correspondingly, the expression of the short range force also needs to be modified by the following factor

$$\boldsymbol{f}_i^{\rm sh}(s) = {\rm erfc}\left(\frac{s}{2R_s}\right) + \frac{s}{R_s\sqrt{\pi}}\exp\left(-\frac{s^2}{4R_s^2}\right), \quad (6)$$

where s is the spatial separation from the particles i. R_s is the splitting radius. From the expression one can readily see that the short range force drops rapidly as the separation increases. Beyond a certain scale $R_{\rm cut}$, contribution from the short force is negligible. Here the exp and erfc functions in Equation (6) are both computationally expensive, and so Equation (6) is usually estimated based on interpolation from a pre-built-*float* table.

Following Bagla (2002) and Springel (2005), PHoTo*N*s also carries out most of the force calculation with the PM-Tree scheme (Xu 1995). For the PM part, PHoTo*N*s employs a conventional Cloud-in-Cell (CIC) scheme to assign particles into regular grids. For the Tree calculation part, we adopt a much used oct-tree (Barnes & Hut 1986) structure for the tree construction. In contrast to some existing Tree codes, we follow Springel (2005) to adopt monopole moments for tree nodes. As discussed in Springel (2005), there are some attractive advantages in using the monopole moments scheme, for example, less memory consumption which also improves the efficiency of tree operations. However, the monopole scheme usually requires a more strict opening criterion in order to achieve the same level of force error when compared to multiple moments. In addition to the geometric effect, accuracy of the tree method is also affected by the dynamical state, and we follow the criterion of Gadget-2,

$$\frac{GM}{s^2}\left(\frac{l}{s}\right)^2 \le \alpha|\mathbf{a}|\,, \quad (7)$$

where s is the separation between the target particles, a tree node has a width of l, α is a control parameter and $\boldsymbol{a}$ is acceleration of a particle. This opening criterion results in higher force precision for particles with larger acceleration and is more effective than the pure geometric opening criterion; please refer to Springel (2005) for details.

When considering the scale at which the PM and Tree calculations are split, Bagla (2002) found that the gravity error is less than 1% if $R_{\rm cut}$ is 3.5 times larger than R_s. A more strict cut scale is used in PHoTo*N*s, which follows Gadget-2 (Springel 2005), $R_{\rm cut} = 4.5R_s$ and $R_s = 1.2\Delta_g$. Springel (2005) shows that, with these parameters, a force error of more than 99.99% for the particles is smaller than 0.005% for a typical value of $\alpha = 0.001$. This implies the $R_{\rm cut}$ should be larger than $5.4\Delta_g$. Thus our choice of $R_{\rm cut} = 6\Delta_g$ is sufficient.

We demonstrate the idea of force splitting in Figure 1. The coarse grains (thick solid lines) are referred to as the *ground* tree-node and the number on one side ($N_{\rm side}$) should be 2 to the power of n. The fine grains (dotted lines) represent the PM mesh. The size of the coarse grains is set to be exactly 6 times the fine one ($> 5.4\Delta_g$), which has the advantage that the tree calculation part for the target point (a filled square in the diagram) only needs to consider its adjacent coarse grains and itself. Our domain decomposition strategy is also based on this splitting scheme; we will discuss this further in later sections.

In the standard PM-Tree method, the short range force is computed all the way with the tree method. PHoTo*N*s implements some modifications, and instead we use the PP method to replace the tree calculation when a tree branch is left a given number of particles, $N_{\rm pp}$. This implementation has two advantages. Firstly by doing this, the depth of a tree is reduced by a factor of about $15\%-20\%$, hence reducing the memory consumption and the levels of tree-walk. Secondly, compared to the Tree method, it is more efficient to compute force of a group of particles with the PP if the number is small. In practice, we found $N_{\rm pp} = 100$ is a good number to keep the force calculation efficiency and at the same time reduce memory consumption. PHoTo*N*s also includes several improvements to the Tree-building part as described below.

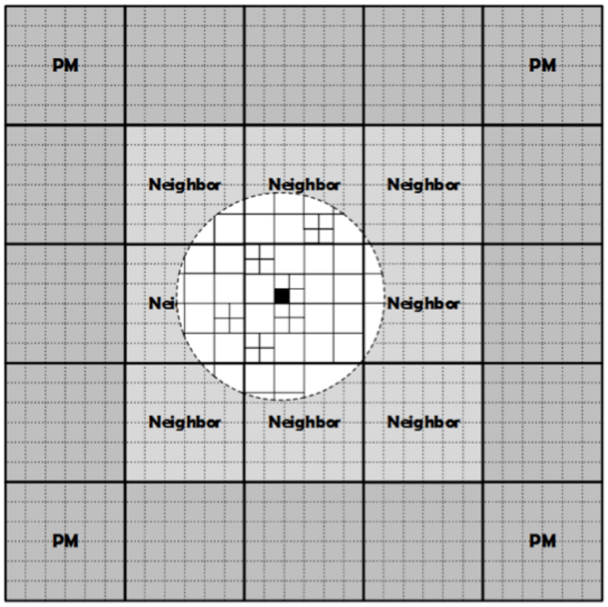


Fig. 1 Diagram illustrating the long and short range force decomposition. The *coarse grain* corresponds to the ground tree node and *fine grid* is for the PM calculation. Gravitational force in any target area (the *filled square*) is a summation of the long range PM and short range Tree force. The radius of the *dashed circle* is exactly six times the PM grid, thus the adjoining ground tree node (*coarse grain*) contains all information needed for the short range force calculation.

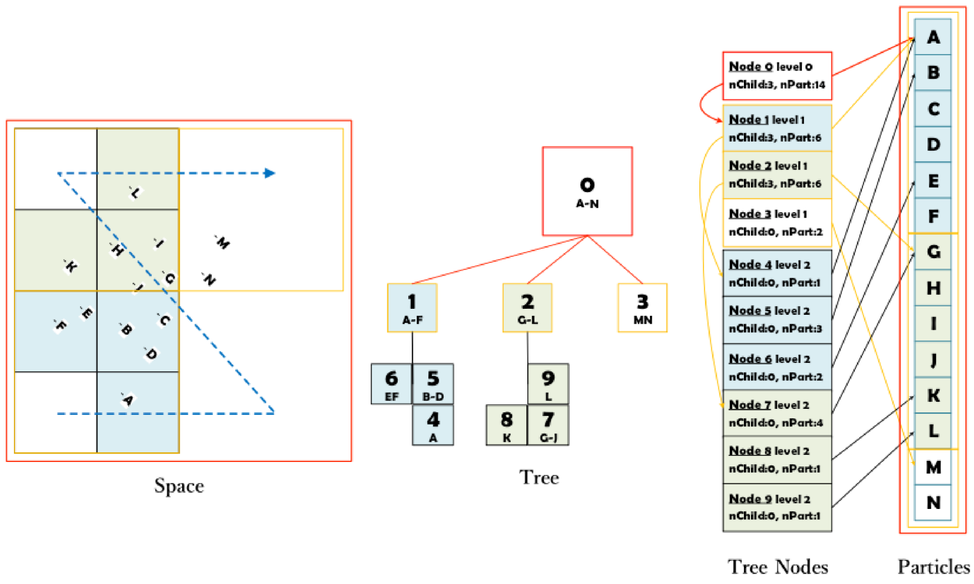
3.2 Tree Building

PHoTo*N*s initially builds a conventional oct-tree as other Tree codes do, with a series of pointers that are defined in the tree nodes structure to record the relationship between parent and offspring nodes. During the procedure, a tree is built recursively by inserting particles one by one. As a result, tree nodes are not contiguously ordered in memory, which is not friendly for tree-walk. Some modifications have been made to improve the tree-walk efficiency by rearranging the tree nodes later in the order of tree-walk, however even by doing this the data storage for particles is still incontiguous. Such a scheme is neither friendly for memory access nor hardware optimization, and it is even more serious on some heterogeneous architectures, for instance Intel MIC due to the communication and memory allocation strategy on MICs.

Desirably, all particles associated with the same tree node should be in a contiguous sequence in memory, and PHoTo*N*s adopts two steps to achieve this. Firstly, all *ground* grids are marked with the index of $(i\times N_{\rm side} + j)\times N_{\rm side} + k$. Then we label each particle with the index of the grid in which it resides and apply a bucket sorting to collect the particles with the same index into the same grid. Now all particles in the same grid should be adjacent in the *ground* level. In the second step we need to refine the *ground* tree nodes to make them continuous in memory. We adopt a Morton key to rearrange all particles of each grid. Morton key is a spatial filling sequence, and the order of the key is exactly the same as the structure of the oct-tree. After doing this, all particles are assigned into the proper tree node at each level, and the number of particles in a node and the position (offset) of the first particle in the global array are recorded in the tree node data structure. After all these steps, not only the children of a branch node are contiguously stored in the memory, but also a particle in any node can also be easily identified in just one memory block without much time latency caused by the cache miss. This scheme greatly improves the memory access efficiency in the consequent tree-walk procedure, especially on MIC. The memory arrangement for tree and particle is shown in 2D in Figure 2.

3.3 Dual Tree Traversal

PHoTo*N*s allows an option to run on Intel MIC architecture by employing a Dual Tree Traversal (DTT) scheme to calculate gravitational force. In the conventional Tree method, the gravitational force of a target particle is the interaction between the target particle and tree nodes and so the program needs to walk through the entire tree, while the DTT computes gravity based on interactions between tree nodes as described in detail below.



Space
Tree
Tree Nodes
Particles
0 A-N
1 A-F
2 G-L
3 MN
6 EF
5 B-D
4 A
9 L
8 K
7 G-J
Node 0 level 0 nChild:3, nPart:14
Node 1 level 1 nChild:3, nPart:6
Node 2 level 1 nChild:3, nPart:6
Node 3 level 1 nChild:0, nPart:2
Node 4 level 2 nChild:0, nPart:1
Node 5 level 2 nChild:0, nPart:3
Node 6 level 2 nChild:0, nPart:2
Node 7 level 2 nChild:0, nPart:4
Node 8 level 2 nChild:0, nPart:1
Node 9 level 2 nChild:0, nPart:1
A
B
C
D
E
F
G
H
I
J
K
L
M
N

Fig. 2 A diagram for the tree structure. The *left panel* displays a group of particles (from A to N) residing in a cube on the configuration space, the *middle panel* shows the tree structure and the *right panel* displays these 14 particles and 10 tree nodes that are stored in the contiguous memory after operations, see text for details.

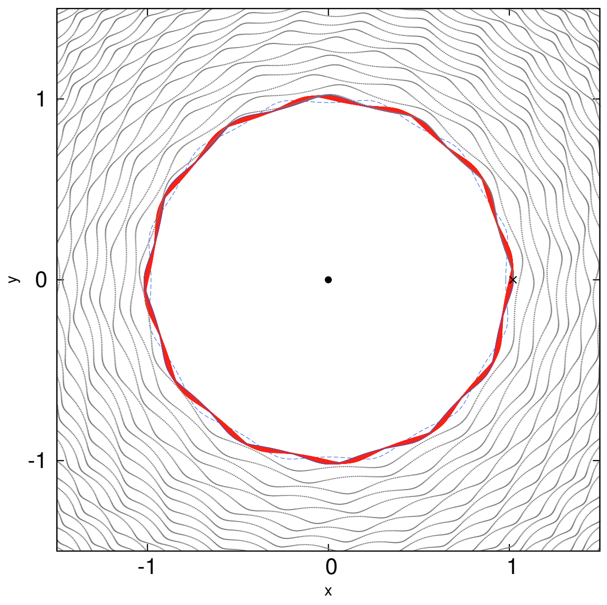


Fig. 3 Adaptive KDK scheme. The *central point* is a massive point. The *cross* marks the initial position of one close binary, and the *blue solid curve* traces its exact closed orbit (nearly overlapping with the *red curve*) run with a sufficiently small time step. The *red curves* indicate the KDK scheme with an insufficiently small time step and the *dotted (gray) curve* represents the DKD scheme with the same time step. The result suggests that KDK has better ability to conserve energy during long time evolution for an N-body system.

Given the side width of the two nodes, A and B respectively, we define the two opening criteria $\theta_1 = (B + A)/R$ and $\theta_2 = B/(R - A)$. If the opening angle is larger than the criteria, such as $\theta = 0.3$, one node should be opened and its offsprings are checked with respect to the other node recursively, until all node pairs are accepted or end up as a leaf (Benoit Lange & Fortin 2014). In our framework, a leaf is a package with a given number of particles. Once the criteria are met, all particles of the target node gain the gravity from the mass center of the other node (in the case of 1 to N). If traversal occurs between two fat leaves (in the case of N to M), a subroutine, the PP kernel, is called; this case usually occurs in dense regions. Since particles are contiguous in memory after the tree is built as we described in the last section, the force accumulation can be readily optimized.

In regards to the opening criteria, a previous study (Benoit Lange & Fortin 2014) suggested opening a relatively larger node. In our experience, however, we found this is a good choice only on the pure CPU platform. On MIC architecture, it performs better to preserve the target node and open the other one, no matter which one is larger.

3.4 Adaptive KDK Stepping

The N-body problem is a Hamiltonian system whose long time behavior can often be changed by non-Hamiltonian perturbations introduced by ordinary numerical integration methods, thus a symplectic integration method is desired. Here we adopt the cosmic symplectic scheme proposed by Quinn et al. (1997). Based on the Lagrangian of Equation (1), two symplectic operators can be defined. The one to update the particle position is referred to as *Drift* and the other one to update momentum is *Kick*. The specific expressions read

$$\begin{aligned} D(t_1;t_2)\ &:\ \boldsymbol{r}(t_2) = \boldsymbol{r}(t_1) + \boldsymbol{p}\int_{t_1}^{t_2}\frac{dt}{a^2}, \\ K(t_1;t_2)\ &:\ \boldsymbol{p}(t_2) = \boldsymbol{p}(t_1) - \nabla\phi\int_{t_1}^{t_2}\frac{dt}{a}. \end{aligned} \tag{8}$$

Drift and Kick must be alternately applied. This can be implemented with two feasible updating schemes, Kick-Drift-Kick (KDK) or Drift-Kick-Drift (DKD). For the KDK scheme, the momentum is updated by a half time step (from t_1 to $t_1 + (t_2 - t_1)/2$) (Kick), then the position is updated by an entire step from t_1 to t_2 (Drift), and finally the momentum is updated again by the remaining half step (from $t_1 + (t_2 - t_1)/2$ to t_2) (Kick). The integration order of DKD is just reversed for KDK.

In order to examine the robustness of those schemes under decomposition of long and short range force calculation schemes, we set up a 3-body system, in which we define two mass points orbiting around a central massive object. Fine-tuning the mass ratio, velocities and configurations is set so that the scale of the revolved orbit about the central mass point is larger than the split scale and the close binary is dominated by the short-range force. The exact orbits can be solved by an extremely high resolution integration with a direct 3-body gravity, which is a closed orbit denoted by the (blue) solid curve in Figure 3.

Then we decrease the stepping rate. In the KDK scheme (the red curve), the lower stepping rate is still stable but causes an extra procession. However the trajectory of DKD (the gray dotted curve) is unstable and spiraling outwards. It suggests that KDK is more robust for scale splitting as well. So, PHoTo*N*s also employs the KDK scheme.

In cosmological N-body simulation, as matter becomes more and more clustered, a higher stepping rate (or shorter step) is needed to accurately follow the more rapid change of trajectory. PHoTo*N*s follows the criterion of Gadget-2,

$$t_{\rm acc} = \frac{2\eta\epsilon}{\sqrt{|\boldsymbol{a}|}},$$

to evaluate whether to refine the step length. If it is not satisfied, the current step length is divided by 2, so that the step length of any particle in a variant environment always has a power of 2 at the top level. Such a half-and-half refinement is flexible for synchronizing the adaptive steps among the particles in variant environments at different levels.

4 IMPLEMENTATION AND PARALLELIZATION

4.1 Domain Decomposition

The essential task for parallelization of an N-body problem on scalar architectures is how to distribute particles into individual processors. We follow the scheme proposed by Springel (2005) with some improvements. The basic idea of the scheme is to use a self-similar Peano-Hilbert curve to map 3D space into a 1D curve which can be cut further into pieces that define the individual domain. In practice, we firstly partition the simulation box into 2^{3n} grids and label each grid with a unique integer key, which is referred to as a PH key, labeling the order of a point in the curve. Next, we sort the grids in order of the key and find the cutting positions of the filling curve, then we collect the particles in the same segment as a domain. In our implementation, we follow the approach of Ishiyama et al. (2012) to partition the 1D curve according to the workload by using wall clock time of each segment as the weight to assess the workload in those segments. As shown by Ishiyama et al. (2012), this can significantly improve load imbalance. Note, while the Peano-Hilbert curve is not the only option, its decomposition is relatively spatially compact and has a low ratio of surface to

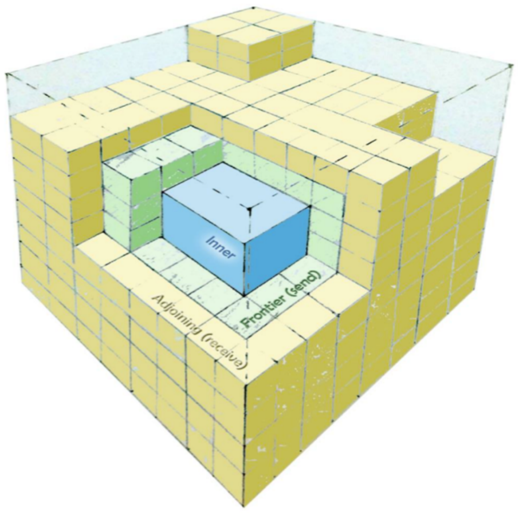


Fig. 4 A sketch of a domain in PHoTo*N*s. The domain in a computing node is a simply connected region determined by the Peano-Hilbert filling curve method. A domain is surrounded by the ghost layer (*yellow*). The frontier part of the domain (*green*) needs to export data to its adjacent domains. The inner (*blue*) part is independent of other domains. The subcube denotes *ground* nodes.

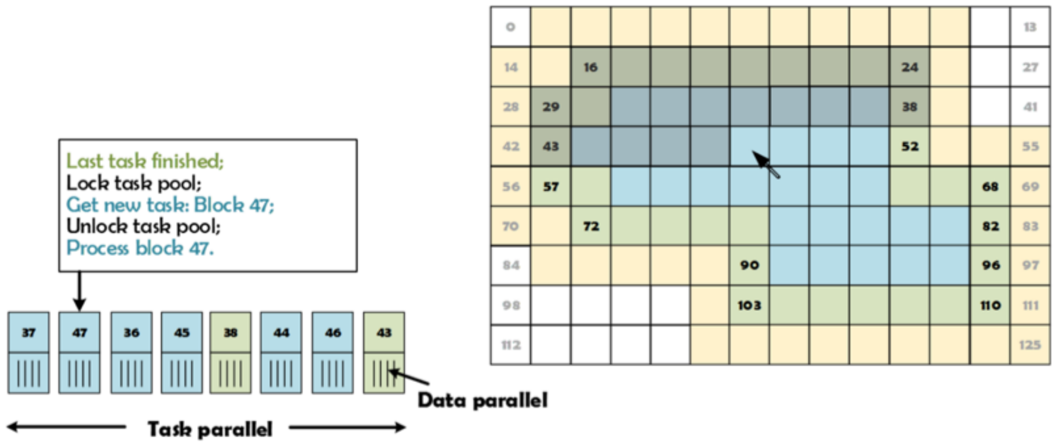

Last task finished;
Lock task pool;
Get new task: Block 47;
Unlock task pool;
Process block 47.

Fig. 5 A sketch of the task queue. The index labels the order of ground grids (*right panel*) and it forms a natural queue of tasks (*left queue*). Multiple threads roll the task queue in ascending order. In each task, PHoTo*N*s is optimized by the strategy of data parallelization.

volume; the latter is advantageous in communications of the domain boundaries.

As an example, Figure 4 illustrates the structure of a domain. Each grid is one of 2^{3n} described above, and a group of them makes up of a domain. Since the short range force needs information from particles in adjacent domains, an entire domain includes a ghost layer containing data of adjacent domains. In Figure 4, the ghost layer (“adjoining” in the label) (yellow) stores information on the particles in its adjacent domains. Correspondingly, frontier layer (green) also needs to send its interior particle information to its adjacent domains. Therefore we allocate an extra buffer to store information for adjacent ghost layers, while the inner part (blue) of the domain does not require information from other domains.

4.2 Particle Mesh Implementation

PHoTo*N*s employs the PM method to compute the long-range force. It involves Fourier transformation for which we use the publicly available FFTW package in PHoTo*N*s. However the data storage for FFTW does not match the strategy of our domain decomposition. In the FFTW, the mass density field must be assigned into the slabs along a specific direction. In addition to the conventional **MPI_COMM_WORLD** for the domain decomposition, we add an exclusive MPI communicator, **PM_COMM_WORLD** for the PM. The rank of **MPI_COMM_WORLD** increases along the Peano-Hilbert key, while the rank of **PM_COMM_WORLD** increases along the z direction. We construct the density mesh for the PM on the local domain, send the mesh information to the proper rank, then fill the information into the correct position.

PHoTo*N*s forks one thread for the PM and one thread for domain communications. The most time consuming part of the calculation, tree-walk, uses all the remaining resources on each computing node. For instance, a CPU with 12 cores has to create a thread for domain decomposition, then create a thread for the PM and a thread for the Tree-building at the same time. Since PM calculation decouples with the Tree walking part, the PM and Tree traversal can be carried out at the same time. This improves scalability of the code because scalability of the FFT decreases with the number of employed meshes.

4.3 Task Queues

In our framework, the tree walking does not start from the root but from *ground* grids. Each ground grid is a branch of a local tree. Since the grid size is exactly 6 times larger than the PM grid (see Fig. 1), each grid will complete the short-range force calculation by walking through the 26 surrounding adjoined local grids and itself. The right panel of Figure 5 illustrates a domain consisting of the ground tree node. It is a natural queue pool to identify one grid as one task. This represents an effective task-parallelization strategy in which multiple threads roll the task queues, e.g. each thread fetches a grid data point for the tree walking and the PP calculation. Such a task strategy can also better take advantage of the DTT method (Yokota & Barba 2013; Yokota 2012).

4.4 Heterogeneity

PHoTo*N*s is designed for a heterogeneous system. In this work, we focus on the Intel co-processor MIC which has heterogeneous architecture, the advantage of which is high concurrency and wide Single Instruction Multiple Data (SIMD) instruction. Since the average bandwidth and memory access latency on an MIC core is relatively lower than that of a CPU, we offload particles onto MIC memory to accelerate the short-range force calculation. Because the particles are recorded on the *ground* grain, the computing workload can be readily estimated. After assessing the particle numbers in grains, we can determine an optimized partition workload ratio between CPU and MIC in task queues. PHoTo*N*s also provides a dynamic mechanism to adjust the partition ratio according to the real elapsed time on MIC and CPU. The MIC run with offload mode and the data of the particles and tree nodes are transferred to MIC memory at the beginning of each time step. This can optimize communication between MIC and main memory and avoid repeated data exchange.

The forces for particles in a leaf are computed with the direct summation method PP. To speed up this procedure, we optimize its performance with the idea of data parallelism. This allows us to put more particles into leafs without consuming additional time. This substantially reduces depth of the tree and so decreases the tree-walk. According to our experiment, the maximum particle number in a leaf can be set between 512 and 2048 on an MIC and between 16 to 128 on a CPU (Habib et al. 2013).

5 TEST AND PERFORMANCE

In order to test the accuracy of our code, we carry out two cosmological N-body simulations starting from an identical initial condition but run with PHoTo*N*s and Gadget which has been commonly used in the community. The simulations follow 128^3 particles within a comoving box with a side length of $100\,h^{-1}$ Mpc. The initial condition of our simulations is generated by 2LPTic at redshift $z = 49$, and the cosmological parameters are assumed to be $\Omega_M = 0.25, \Omega_\Lambda = 0.75, H_0 = 0.7$ and $\sigma_8 = 0.8$. In Figure 6, we provide visualization of the matter density map for both simulations at $z = 0$. Clearly the two maps are indistinguishable.

In Figure 7 we provide a quantitative comparison of both simulations. In the left panel of the figure, we compare the matter power spectrum of both simulations at different epochs, $z = 0, 0.2, 1$ and 4. Solid lines are results from the run with PHoTo*N*s and symbols are from the run with Gadget. As can be seen clearly, matter power spectra in both simulations are identical. The middle panel of the figure presents a comparison of the halo mass function in both simulations. Again results from the two simulations are indistinguishable. In the right panel of the figure, we plot the density profile of the most massive halo in both simulations. The vertical dashed lines show the softening length of our simulations, and the

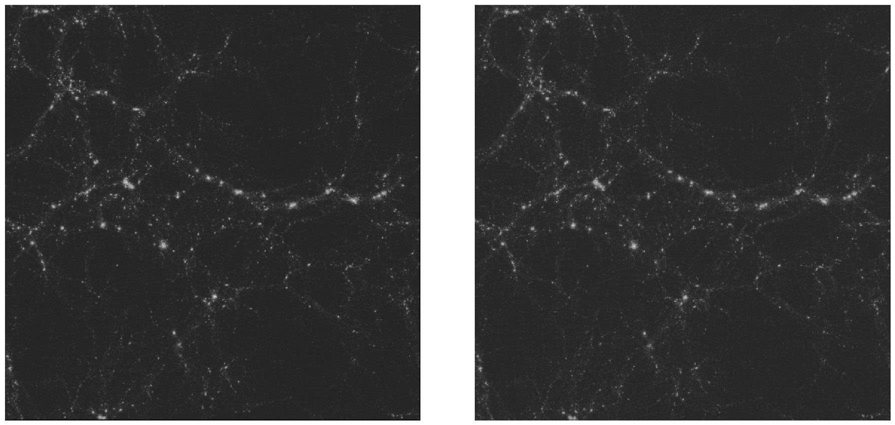


Fig. 6 Comparison of density map from our simulations run with different codes. The *left panel* shows a slice of the simulation run with Gadget-2 and the *right panel* shows the result from PHoTo*N*s.

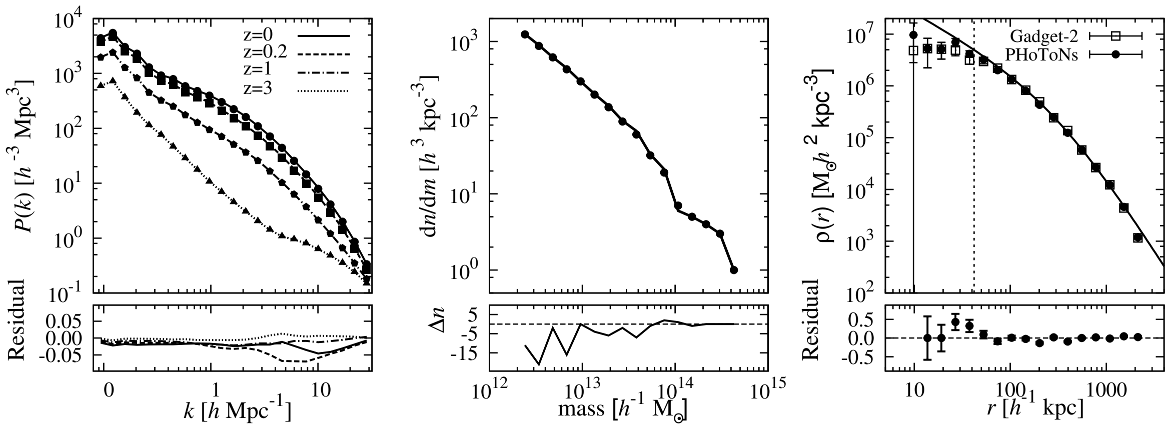


Fig. 7 Comparisons of various properties from simulations run with Gadget-2 and PHoTo*N*s: power spectrum (*left panel*), dark matter halo mass function (*middle panel*) and density profile of the largest halo in each simulation (*right panel*). In the *left* and *middle panels*, *lines* indicate results from Gadget and various symbols are from PHoTo*N*s. In the *right panel*, *open squares* are from Gadget-2 and *filled squares* are from PHoTo*N*s, as indicated in the label. Δn is the difference of halo number in each mass bin.

solid curve is the best Navarro-Frenk-White (NFW) fit to the profiles. Clearly, density profiles of both simulations agree well with each other down to the softening length. This suggests that the numerical accuracy of PHoTo*N*s is comparable to the popular code–Gadget (Heitmann et al. 2008; Kim et al. 2014).

Our benchmark runs on the *Era* supercomputer[1], which is a 300 Tflop/s (double precision) cluster. The computer contains 256 CPU nodes, 40 MIC nodes and 30 GPU nodes. Each node comprises two twelve 2.8 GHz Ivy Bridge Xeon E5-2680v2 cores and 64/128 GB DDR3 of system memory. These nodes are interconnected with the EDR Infiniband which provides 56 Gbps peer to peer bandwidth. Each of the 40 MIC codes has two Xeon Phi 5110p cards with 60 cores (four threads per core) running at 1.05 GHz and 8 GB of GDDR5 memory. The peak single precision (SP) performance is 896 Gflops for CPU node and 4.926 Tflop/s for MIC node.

Table 1 shows that PHoTo*N*s has good scalability both on pure CPU and CPU+MIC platforms. When reaching the limit of our test machine, 40 nodes on *Era*, the wall clock time roughly decreases linearly. CPU+MIC architecture generally speeds up the CPU platform by roughly 3 times. Assuming 22 flops in one interaction (Nitadori et al. 2006), the efficiency goes up to 68.6% of peak performance on MIC and 74.4% on two Sandy Bridge CPUs.

6 SUMMARY

We describe a new cosmological N-body simulation code PHoTo*N*s. The code is designed to run on both

[1] *http://www.sccas.cn/yhfw/yjzy/xyd/*

Table 1 Scalability of PHoTo*N*s on the *Era* supercomputer. $N_{\rm side}$ denotes the number of PM meshes used in tests. CPU and C+M indicate wall clock time run with pure CPUs and CPU+MIC, respectively.

#Node	2	4	8	16	24	32	40
$N_{\rm side} = 192$							
CPU (s)	721.0	364.0	188.4	97.7	66.7	51.6	42.0
C+M (s)	240.9	123.1	66.4	34.2	24.8	19.4	17.5
$N_{\rm side} = 384$							
CPU (s)	270.8	137.5	71.9	38.7	28.3	21.4	20.3
C+M (s)	54.9	30.9	17.3	11.0	10.2	8.2	9.5

pure CPU based and heterogeneous platforms. In particular, our implementation on the heterogeneous platform is dedicated to performing massive simulations on the CPU+MIC architecture, for instance, the Tianhe-2 supercomputer.

PHoTo*N*s adopts a hybrid scheme to compute gravity solver, including the conventional PM to calculate the long-range force and the Tree method for short-range force, and the direct summation PP to compute force between very close particles. A merit of PHoTo*N*s is it better takes advantage of the multi-threads feature of the new generation of supercomputers by using DTT and a flexible task queue to make use of multiple threads and improve the load imbalance. In addition, our PM computation is independent of our tasks and thus can be hidden during the tree-walk, which improves scalability of the code. The performance is effectively improved by optimizing the PP kernel with data parallelism on the heterogeneous architecture of MIC. The future development of PHoTo*N*s will include hydrodynamics, which will be presented in the another paper.

Acknowledgements We acknowledge support from the National Key Program for Science and Technology Research and Development (2017YFB0203300) and the National Natural Science Foundation of China (Grant Nos. 11403035, 11425312 and 11573030). LG acknowledges support from Royal Society Newton advanced Fellowships.

References

Bagla, J. S. 2002, Journal of Astrophysics and Astronomy, 23, 185

Bagla, J. S., & Khandai, N. 2009, MNRAS, 396, 2211

Barnes, J., & Hut, P. 1986, Nature, 324, 446

Benoit Lange, B., & Fortin, P. 2014, Euro-Par 2014 Parallel Processing, 716, *http://hal.upmc.fr/hal-00947130v2*

Bertschinger, E., & Gelb, J. M. 1991, Computers in Physics, 5, 164

Efstathiou, G., Davis, M., White, S. D. M., & Frenk, C. S. 1985, ApJS, 57, 241

Emberson, J. D., Yu, H.-R., Inman, D., et al. 2017, RAA (Research in Astronomy and Astrophysics), 17, 085

Guo, Q., White, S., Boylan-Kolchin, M., et al. 2011, MNRAS, 413, 101

Habib, S., Morozov, V., Frontiere, N., et al. 2013, in SC'13 Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis, Article #6

Heitmann, K., Lukić, Z., Fasel, P., et al. 2008, Computational Science and Discovery, 1, 015003

Hockney, R. W., & Eastwood, J. W. 1988, Computer Simulation Using Particles (Bristol: Hilger)

Ishiyama, T., Fukushige, T., & Makino, J. 2009, PASJ, 61, 1319

Ishiyama, T., Nitadori, K., & Makino, J. 2012, arXiv: 1211.4406

Jing, Y. P., & Suto, Y. 2002, ApJ, 574, 538

Kim, J.-h., Abel, T., Agertz, O., et al. 2014, ApJS, 210, 14

Klypin, A. A., Trujillo-Gomez, S., & Primack, J. 2011, ApJ, 740, 102

Makino, J. 2004, PASJ, 56, 521

Makino, J., Fukushige, T., Koga, M., & Namura, K. 2003, PASJ, 55, 1163

Makino, J., Taiji, M., Ebisuzaki, T., & Sugimoto, D. 1997, ApJ, 480, 432

Nitadori, K., Makino, J., & Hut, P. 2006, New Astron., 12, 169

Peebles, P. J. E. 1980, The Large-scale Structure of the Universe (Princeton: Princeton Univ. Press)

Potter, D., Stadel, J., & Teyssier, R. 2017, Computational Astrophysics and Cosmology, 4, 2

Prada, F., Klypin, A. A., Cuesta, A. J., Betancort-Rijo, J. E., & Primack, J. 2012, MNRAS, 423, 3018

Quinn, T., Katz, N., Stadel, J., & Lake, G. 1997, astro-ph/9710043

Springel, V. 2005, MNRAS, 364, 1105

Stadel, J., Potter, D., Moore, B., et al. 2009, MNRAS, 398, L21

Teyssier, R. 2002, A&A, 385, 337

Wadsley, J. W., Stadel, J., & Quinn, T. 2004, New Astron., 9, 137

Wang, L., Spurzem, R., Aarseth, S., et al. 2015, MNRAS, 450, 4070

Warren, M. S. 2013, arXiv:1310.4502

Xu, G. 1995, ApJS, 98, 355

Yokota, R. 2012, Journal of Algorithms & Computational Technolog, 3, 301

Yokota, R., & Barba, L. A. 2013, Computers & Fluids, 80, 17

Yu, H.-R., Emberson, J. D., Inman, D., et al. 2017, Nature Astronomy, 1, 0143