

PHoToNs –A Parallel Heterogeneous & Threads oriented code for cosmological N -body Simulation

Qiao Wang¹, Zongyan Cao^{2,1}, Liang Gao^{1,3}, Xuebin Chi², Chen Meng^{2,4}, Jie Wang¹, Long Wang^{2,4}

¹ Key Laboratory of Computational Astrophysics, National Astronomical Observatories, Chinese Academy of Sciences, Beijing, 100012 qwang@nao.cas.cn

² Supercomputing Center, Computer Network Information Center, Chinese Academy of Sciences, Beijing

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

Received 0000 xxxx 00; accepted 0000 xxxx 00

Abstract We introduce a new code for cosmological simulations, PHoToNs, which has features on performing massive cosmological simulations on heterogeneous high performance Computer (HPC) and threads oriented programming. PHoToNs adopts a hybrid scheme to compute gravity force, with the conventional PM to compute the long-range force, the Tree algorithm to compute the short range force, and the direct summation PP to compute the gravity from very close particles. A self-similar space filling Peano-Hilbert curve is used to decompose computing domain. Threads programming is highly used to more flexibly manage the domain communication, PM calculation and synchronization, as well as Dual Tree Traversal on the CPU+MIC platform. The scalability of the PHoToNs performs well and the efficiency of 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 becomes an essential tool to understand 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

statistics and simultaneously achieve high resolution to resolve faint galaxies to be observed in galaxy surveys. For example, a recent simulation of [Potter et al. \(2017\)](#) evolving unprecedented 2 trillion particles in 3 Gpc^3 volume to make galaxy mock for the upcoming Euclid survey. To perform such 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 mean time, the high performance computing architecture is undergoing significant changes, many new HPC systems have 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 of the popular heterogeneous HPC machines ([Guo et al. 2011](#)). In this paper, we introduce a novel cosmological simulation code PHoToNs. It has two characteristics, the first feature is that the code is programmed basing on threads, the other is highly optimized on heterogeneous architecture, such as Intel Xeon Phi (MIC).

The organization of the paper is as follows. We briefly introduce the background of physical model in section 2. The force calculation algorithm is present in Section 3. Section 4 discuss our parallelization and implementation strategy. In section 5, we describe the numerical accuracy and performance of the PHoToNs.

2 EQUATIONS OF MOTION

In cosmological N-body simulation, 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

$$\mathcal{L} = \frac{\mathbf{p}^2}{2a^2} - \frac{\phi}{a}, \quad (1)$$

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

$$\nabla^2 \phi = 4\pi G a^3 [\rho_m(\mathbf{r}, t) - \bar{\rho}_m(t)], \quad (2)$$

The density field ρ can be constructed from discrete mass points

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

Here the Dirac delta function $\delta_{\mathbf{D}}$ can be modified according to different mass distributions.

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

$$\begin{aligned} \dot{\mathbf{r}} &= \frac{\mathbf{p}}{a^2}, \\ \dot{\mathbf{p}} &= -\frac{\nabla \phi}{a}, \end{aligned} \quad (4)$$

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 particles is solved by the gradient of potential and pairwise

3 FORCE CALCULATION ALGORITHM & IMPLEMENTATION

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

3.1 PM-Tree-PP method

The Particle-Mesh (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, its solution can be simply obtained by convolution of Green function under the periodic boundary condition. The Green function of the Poisson equation has a simple form $-/k^2$, its 3-point difference in k space reads

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

where Δ_g denotes the width of a mesh, N is the mesh number on one side, l, m, n denotes the discrete wave number. The convolution of density field and Green function is a multiple in Fourier space and can be implemented with Fast Fourier Transformation(FFT). As FFT works on regular mesh, 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 others gravity solver algorithms were invented to improve force accuracy in the small scales. Among these, a popular gravity solver is *Tree method* invented by Barnes & Hut (1986), which provides a robust efficiency even for highly clustering system. In their original approach, the simulation volume is recursively divide into 8 smaller subcubes until each subcube only contains one particle. Each subcube contains information of the center of mass and the multiple moment for the mass distribution enclosed in the subcube etc. 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 go 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 advantage 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 the long range gravity with the PM method and calculate the short range force with the Tree algorithm. The combination of two parts achieves an accurate 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 convoluted to take out the short range force of the PM calculation. Here s is the spatial separation of two particles, R_s is a

effect. Correspondingly, the expression of the short range force also need to be modified by the following factor

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

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_{cut} , contribution from the short force is negligible. Here \exp and erfc functions in Eq. 6 both are computationally expensive, and so Eq. 6 is usually estimated based on the interpolation from a pre-built *-float* table.

Following Bagla (2002) and Springel (2005), PHoToNs also carries out the most force calculation with PM-Tree scheme (Xu 1995). For the PM part, PHoToNs employs a conventional Cloud-in-Cell (CIC) to assign particles into regular grids. For the Tree calculation part, we adopt a much used oct-tree (Barnes & Hut 1986) for the tree construction. In contrary 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. But the monopole scheme usually requires a more strict opening criterion in order to achieve the same level of force error when comparing to multiple moment. In addition to the geometric effect, the accuracy of tree method is also affected by dynamical state, we follow the criterion of Gadget-2,

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

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

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

We demonstrate the idea of the force splitting in Fig. 1. The coarse grains (heavy solid line) are referred to as the *ground* tree-node and the number of one side (N_{side}) should be 2 to the power of n . The fine grains (dotted line) represents 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 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. PHoToNs makes some modifications, instead we use the PP method to replace the tree calculation when a tree branch left a given number of particles, N_{pp} . This implementation has two advantages. Firstly by doing

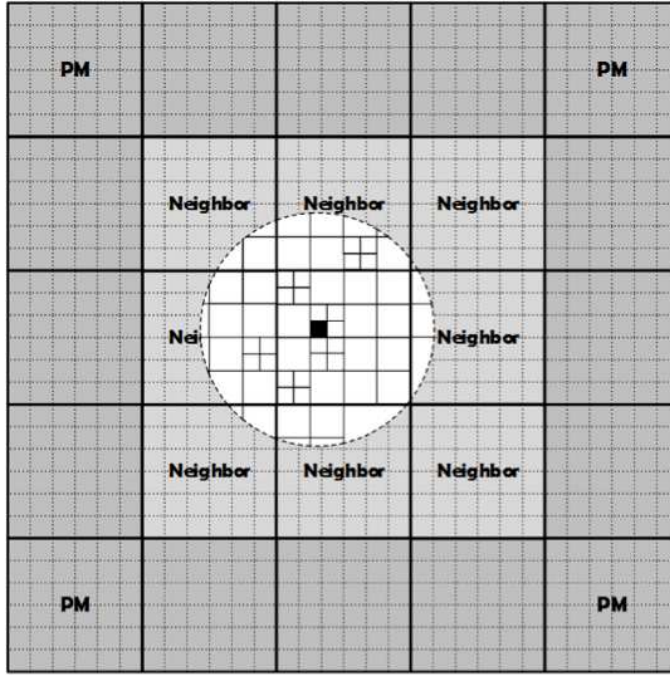


Fig. 1 Diagram for the long and short range force decomposition, the coarse grain corresponds to the ground tree node and fine grid is for the PM calculation. Gravitational force in any target area (the filled square) is a summation of the long range PM and short range Tree force. The radius of dash circle is exactly 6 times the PM grid, thus the adjoining ground tree node (coarse grain) contains all information needed for the short range force calculation.

and the levels of tree-walk. Secondly, compared to the Tree method, it is more efficient to compute force of a group particles with the PP if the number is small. In practice, we found $N_{pp} = 100$ is a good number to keep the force calculation efficiency and at the same time reduce memory consumption. PHoToNs also made several improvements on the Tree-building part as described below.

3.2 Tree building

PHoToNs initially builds a conventional oct-tree as other Tree codes does, namely a series of pointers are defined in the tree nodes structure to record the relationship of parent and off-spring nodes. During the procedure, 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 are 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 in-contiguous. Such a scheme is neither friendly for memory access nor hardware optimization, and it is even more serious on some heterogeneous architecture, for instance Intel MIC due to communication and memory allocation strategy on MICs.

Desirably, all particles of the same tree node should be in a contiguous sequence in memory, PHoToNs adopt two steps to achieve this. Firstly all *ground* grids are marked with the index of $(i \times N_{side} + j) \times N_{side} + k$. Then we label each particle with the index of the grid in which it resides and apply for a bucket sorting

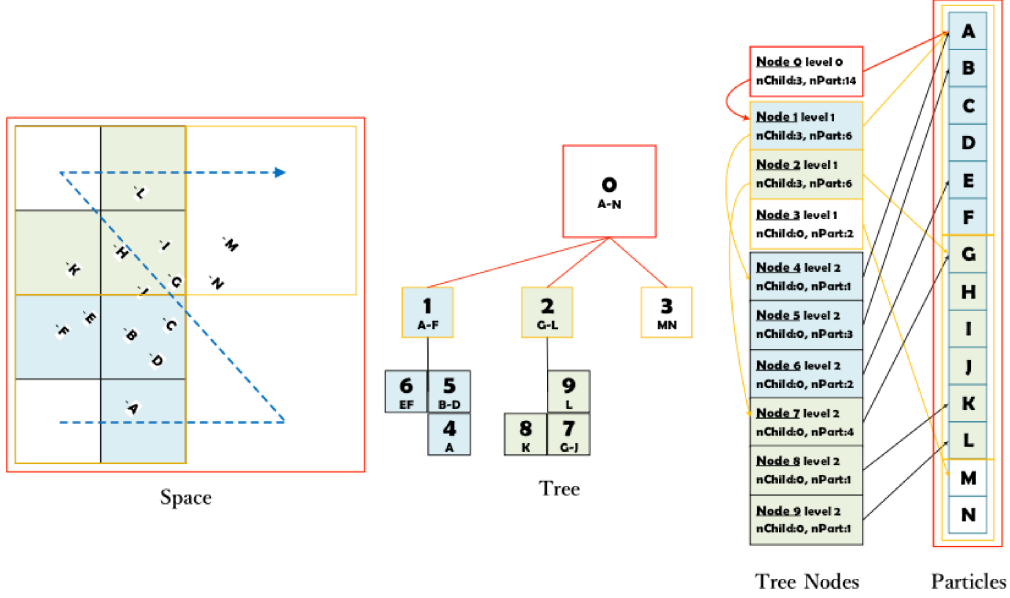


Fig. 2 A diagram for the tree structure. The left panel displays a group of particles (from A to N) residing in a cube on the configuration space, The middle panel show the tree structure, and the right panel displays these 14 particles and 10 tree nodes are stored in the contiguous memory after operations,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

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 Morton key to re-arrange all particles of each grid. Morton key is a spatial filling sequence, its 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 the particle 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 consequent tree-walk procedure, especially on MIC. The memory arrangement of tree and particle is shown in 2D in Fig. 2.

3.3 Dual Tree Traversal

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

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 criterion, such as $\theta = 0.3$, one node should be opened and its off-springs are checked respected 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 of a given number of particles. Once the criterion meet, all particles of the target node gain the gravity from

case of N to M), a subroutine of PP kernel is called, this case usually occurs in dense regions. Since particles are contiguous in memory after the tree building as we described in the last section, the force accumulation can be readily optimized.

Regards to the opening criterion, previous study (Benoit Lange & Fortin 2014) suggest to open the relatively bigger node (Benoit Lange & Fortin 2014). In our practice, however, we found this is a good choice only on 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 Eq. 1, two symplectic operators can be defined. The one to update the particle position is referred to *Drift* and the other one to update momentum is *Kick*. The specific expressions read

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

The Draft 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 (Draft), 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 to the KDK.

In order to examine the robustness of those schemes under the decomposition of long and short range force calculation scheme, we setup a 3-body system, in which we set up two mass points orbiting around a central massive object. Fine-tuning the mass ratio, velocities and configurations 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 (black) solid curve in Fig. 3. Then we decrease the stepping rate. In KDK scheme (the red curve), the lower stepping rate is still stable but cause 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 PHoToNs also employs the KDK scheme.

In cosmological N -body simulation, as matter become more and more clustered, a higher stepping rate (or shorter step) is needed to accurately follow the more rapid change of trajectory. PHoToNs follow the criteria of Gadget-2,

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

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

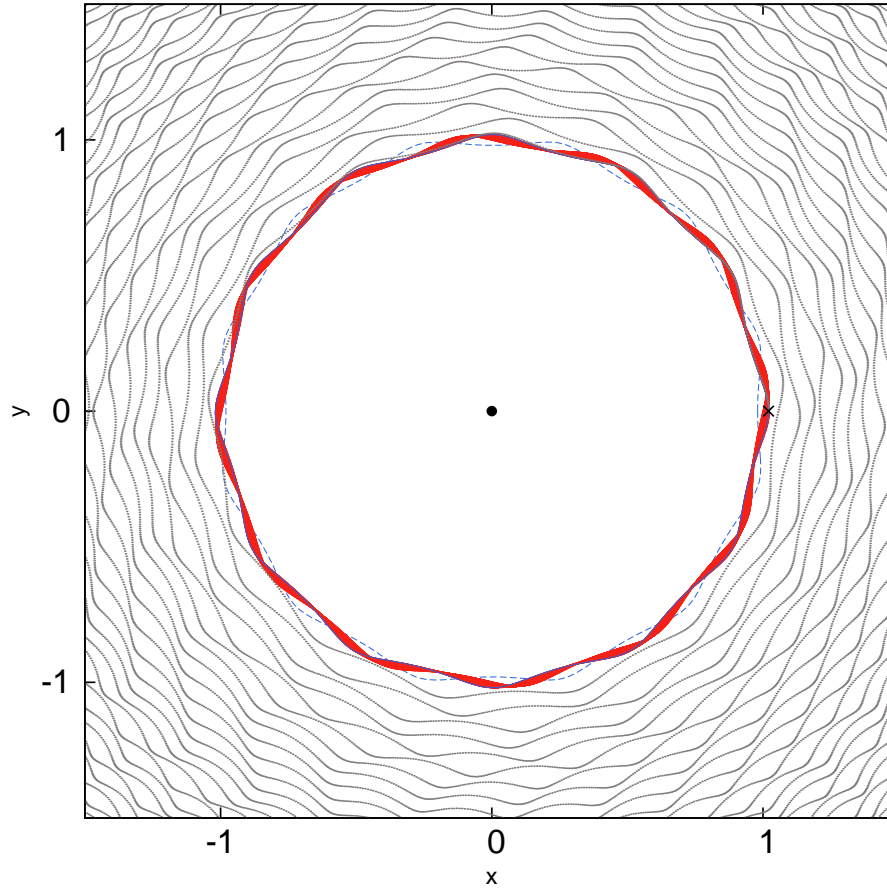


Fig. 3 Adaptive KDK scheme. The central point is a massive point. The cross denotes initial position of one of close binary, the black solid curves denotes exact its closed orbit (nearly overlapping with the red curve) run with sufficiently small time step. The red curves denote KDK scheme with an insufficiently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

4 IMPLEMENTATION AND PARALLELIZATION

4.1 Domain decomposition

The essential task for parallelization of 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 defines the individual domain. In practice, we firstly partition the simulation box into a 2^{3n} grids and label each grid with a unique integer key, which is referred to as 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 approach of [Ishiyama et al. 2012](#)) to partition the 1D Curve according to the

As shown by (Ishiyama et al. 2012), this can significantly improve load imbalance. Note, while the Peano-Hilbert curve is not the only option, but its decomposition is relatively spatial compact and has a low ratio of surface to volume, the latter has advantage in the communications of the domain boundaries.

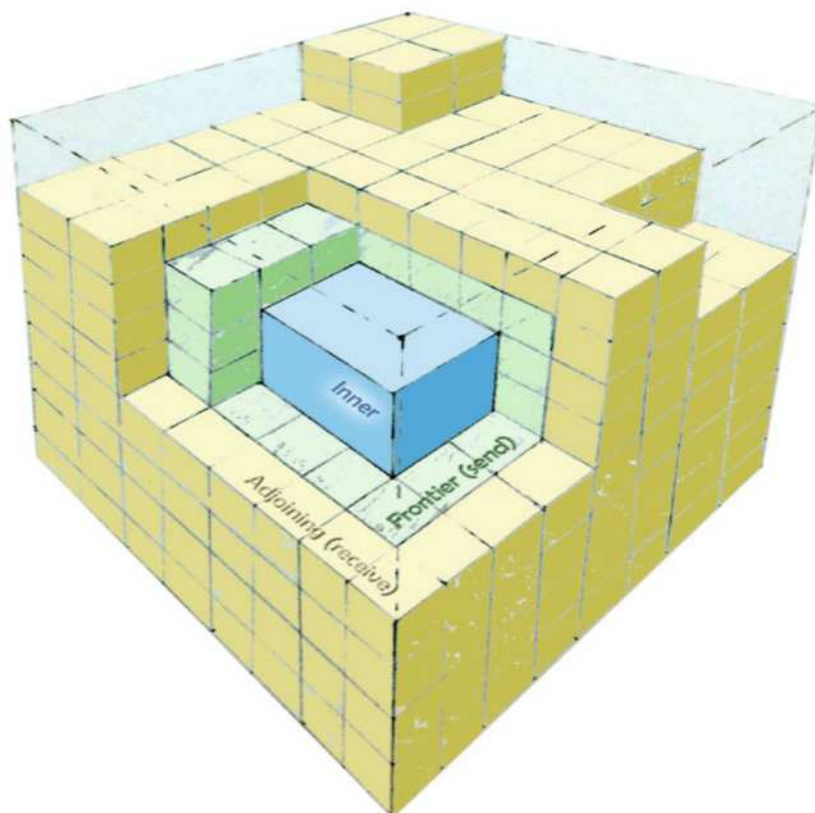


Fig. 4 A sketch of a domain in PHoToNs . The domain in a computing node is a simple connected region determined by Peano-Hilbert filling curve method. A domain is surrounded by the ghost layer (yellow). The frontier part of the domain (green) needs to export data to its adjacent domains. The inner (blue)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

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

4.2 Particle Mesh implementation

PHoToNs employs Particle-mesh method to compute the long-range force. It involves Fourier transformation which we use the public available FFTW package in PHoToNs . However the data storage for FFTW

signed 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.

PHoToNs forks one thread for the PM and one thread for domain communications. The most time consuming part of the calculation—tree-walk, use all the remaining resource 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 decouple 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 of the 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 local trees. 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 Fig. 5 illustrates a domain consisting of the ground tree node. It is a nature queue pool to identify one grid as one task. It is an effective task-parallel strategy that multiple threads roll the task queues, e.g. each thread fetches a grid data for the tree walking and the PP calculation. Such a task strategy can also better take advantage of the DTT method (Yokota & Barba 2013; Koyama et al. 2013).

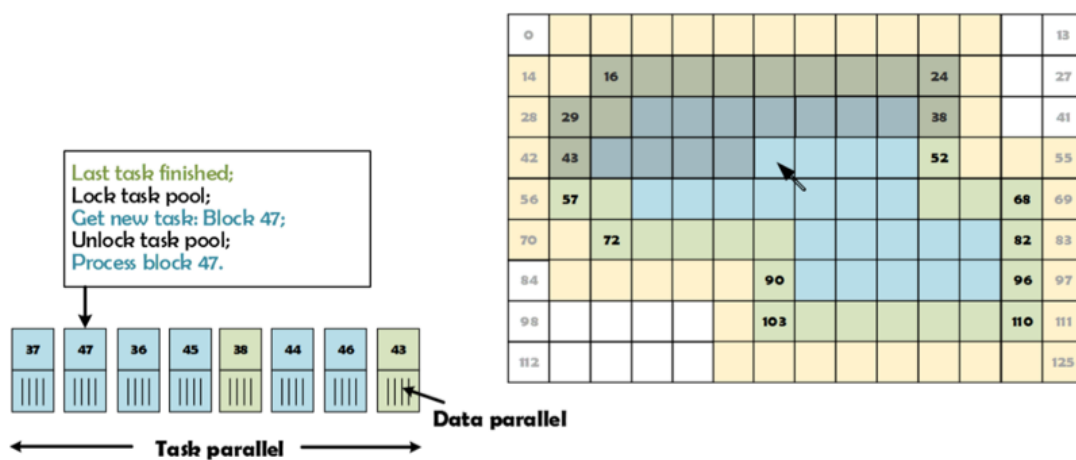


Fig. 5 The sketch of the task queue. The index label the order of ground grids (right panel) and it form a nature queue of tasks (left queue). Multiple threads roll the task queue in ascending order. In each task, PHoToNs is optimized by the strategy of data parallel.

4.4 Heterogeneity

PHoToNs is designed for heterogeneous system. In this work, we focus on the Intel co-processor MIC heterogeneous architecture, whose advantage is high concurrency and wide SIMD (Single Instruction Multiple Data) instruction. Since the average bandwidth and memory access latency on MIC core is relatively lower than that of 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 particles numbers in grains, we can determine an optimized partition workload ratio between CPU and MIC in task queues. PHoToNs also provide 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 is transferred to MIC memory at 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 direct summation method PP. To speed up this procedure, we optimize its performance with the idea of data parallelism. This allows us put more particles into leafs without additional time consuming. 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 MIC and between 16 to 128 on 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 PHoToNs and Gadget which has been commonly used in the community. The Simulations follow 128^3 particles within a comoving box with a side length $100 h^{-1} \text{Mpc}$. The initial condition of our simulations is generated by 2LPTic at redshift $z = 49$, and the cosmological parameters are assumed to be $\Omega_M = 0.25, \Omega_\Lambda = 0.75, H_0 = 0.7, \sigma_8 = 0.8$. In Fig. 6, we provide visualization of the matter density map of both simulations at $z = 0$. Clearly two maps are indistinguishable.

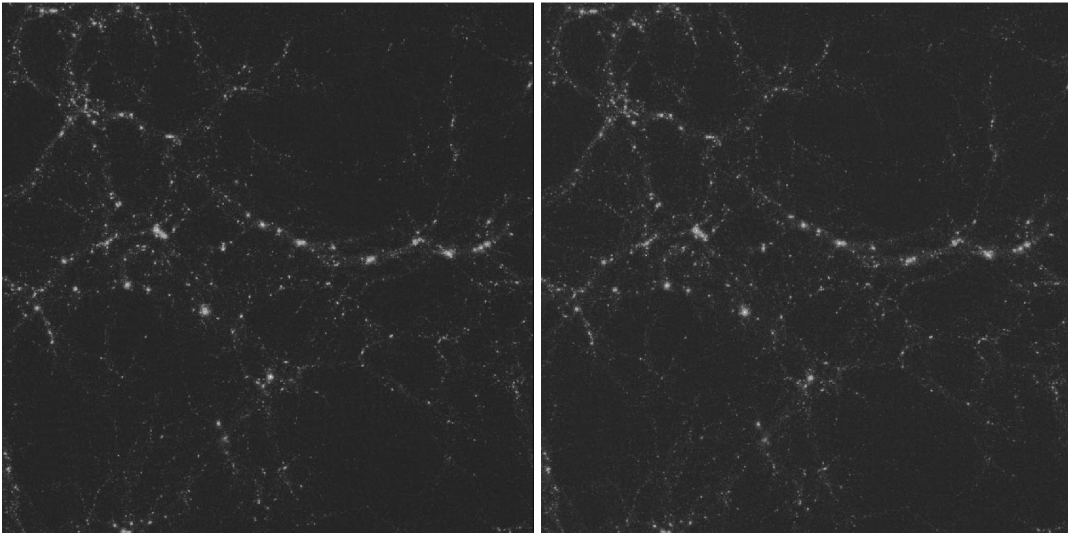


Fig. 6 Comparison of density map of our simulations run with different code. The left panel shows a slice of the simulation run with Gadget-2, the right panel shows result from PHoToNs .

In figure 7 we provide quantitatively comparison of both simulations. In left panel of the figure, we compare matter power spectrum of both simulations at different epoch, $z=0, 0.2, 1$ and 4 . Solid lines are results from the run with PHoToNs and symbols are from the run with Gadget. As can be seen clearly, matter power spectrum in both simulations are identical. The middle panel of the figure present comparison of halo mass function in both simulations. Again results from two simulation are indistinguishable. In the right panel of the figure, we plot density profile of the most massive halo in both simulations. The vertical dashed lines shows the softening length our simulations, and the solid curve is the best NFW fit to the profiles. Clearly, density profiles of both simulations agree well each other down to the softening length. This suggests that the numerical accuracy of the PHoToNs is comparable to the popular code—Gadget (Heitmann et al. 2008; Kim et al. 2014).

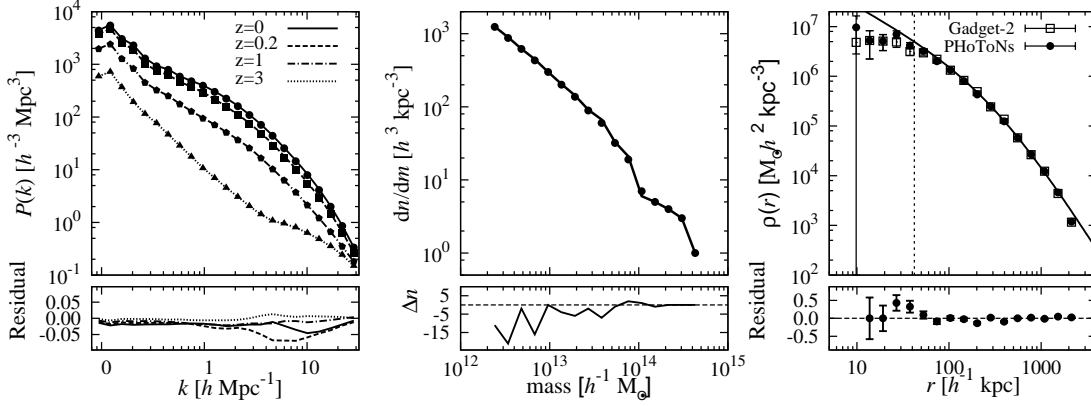


Fig. 7 Comparisons of various properties from simulations run with Gadget-2 and PHoToNs, power spectrum (left panel), dark matter halo mass function (middle panel) and density profile of the largest halo in each simulation (right panel). In the left and the middle panels, lines indicate results from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

Our benchmark runs on *Era* computer¹ of 300 Tflop/s (Double Precision) cluster. The computer contains 256 CPU nodes, 40 MIC nodes and 30 GPU nodes. Each node comprises two twelve 2.8GHz Ivy Bridge Xeon E5-2680v2 cores and 64/128 GB DDR3 system memory. These nodes are inter-connected 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 (4 threads per core) running at 1.05 GHz and 8GB GDDR5 memory. The peak SP (single precision) performance is 896 Gflops for CPU node and 4.926 Tflop/s for MIC node.

Table 1 shows that PHoToNs has good scalability both on pure CPU and CPU+MIC platforms. Upon the limit of our test machine, 40 nodes on *Era*, the wall-clocking 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 CPU.

Table 1 The scalability of PHoToNs on *Era* super computer. N_{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
$N_{side} = 192$							
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
$N_{side} = 384$							
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

6 SUMMARY

We describe a new cosmological N -body simulation code PHoToNs . The code is designed to run on both pure CPU based and Heterogeneous platforms. In particular, our implementation on heterogeneous platform is dedicated to perform massive simulation on the CPU+MIC architecture, for instance, Tianhe-2 supercomputer.

PHoToNs 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 the PHoToNs is to better take advantage of multi-threads feature of new generation of supercomputer by using Dual Tree Traversal 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 the scalability of code. The performance is effectively improved by optimizing the PP kernel with data parallelism on heterogeneous architecture of MIC. The future development of the PHoToNs will include hydrodynamics, which will be presented in the other paper.

Acknowledgements We acknowledge supports from the National Key Program for Science and Technology Research and Development (2017YFB0203300) and NSFC grants (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
- 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, *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*

- 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*
- 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
- Koyama, D., Lai, C., Yamamoto, N., & Yokota, R. 2013
- 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*
- 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., & Barba, L. A. 2013, *Computers & Fluids*, 80, 17
- Yu, H.-R., Emberson, J. D., Inman, D., et al. 2017, *Nature Astronomy*, 1, 0143