计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200200112

# 基于定点压缩技术的双层粒子网格算法的设计与优化

程盛淦[1] 于浩然[2] 韦建文[1] 林新华[1]
1 上海交通大学高性能计算中心 上海 200240
2 厦门大学天文学系 福建 厦门 361005
(chengshenggan@sjtu.edu.cn)

**摘 要** 现代天体物理学的研究离不开大规模N-body模拟。N-body模拟常用的算法之一是粒子网格(Particle-Mesh,PM)算法,但是PM算法需要消耗较多的内存容量。内存限制成为了N-body模拟在现代超算平台大规模扩展的瓶颈。因此,文中使用了利用定点压缩技术减少内存消耗的方法,将存储每个N-body粒子相空间的内存消耗减少到最低6个字节,比传统PM算法低近一个数量级。文中实现了基于定点压缩技术的双层粒子网格算法,并使用包括混合精度计算、通信优化在内的方法对其性能进行了优化。这些优化技术显著降低了定点压缩带来的性能损耗,将压缩和解压在程序总耗时中的占比从21%降低至8%,并且在核心计算热点上达到了最高2.3倍的加速效果,使得程序在较低的内存消耗下保持较高的计算效率和扩展性。

**关键词**:N-body模拟;粒子网格算法;混合精度计算;大规模并行
**中图法分类号** TP391

## Design and Optimization of Two-level Particle-mesh Algorithm Based on Fixed-point Compression

CHENG Sheng-gan[1], YU Hao-ran[2], WEI Jian-wen[1] and James LIN[1]
1 Center for High Performance Computing, Shanghai Jiao Tong University, Shanghai 200240, China
2 Department of Astronomy, Xiamen University, Xiamen, Fujian 361005, China

**Abstract** Large-scale N-body simulation is of great significance for the study of modern physical cosmology. One of the most popular N-body simulation algorithms is particle-mesh(PM). However, the PM-based algorithms cost considerable amounts of memory, which becomes the bottleneck to scale the N-body simulations in the modern supercomputer. Therefore, this paper proposes to use fixed-point compression to reduce memory footprints per N-body particle to only 6 bytes, nearly an order of magnitude lower than the traditional PM-based algorithms. This paper implements the two-level particle-mesh algorithm with fixed-point compression and optimizes it with mixed-precision computation and communication optimizations. These optimizations significantly reduce the performance loss caused by fixed-point compression. The proportion of compression and decompression in the total time of the program reduces from 21% to 8% and achieves up to 2.3 times speedup on computing hotspots which make the algorithm maintain high efficiency and scalability with low memory consumption.

**Keywords** N-body simulation, Particle-mesh method, Mixed-precision calculation, Large-scale parallelism

## 1 引言

天体物理学中,计算机数值模拟技术是重要的研究手段,为现代宇宙学研究和宇宙观测对比提供了数值工具。自20世纪90年代以来,超算平台的发展极大地推动了宇宙学的发展。在目前的超算平台上,发展了各种高性能并行计算模型,使得模拟复杂的天文过程成为可能。N-body模拟在球状星团动力学、星系演化、星系之间相互作用[1-2]等研究领域都有着非常重要的作用。高精度的宇宙学参数测量要求对宇宙大尺度结构(Large Scale Structure,LSS)的形成进行精确的数值建模,从而需要粒子数目较大的N-body模拟[3]。例如,在暗物质和暗能量的研究中,需要分析宇宙中微弱星系的结构和演化历史,这要求对拥有较大范围的物理尺度和较高的粒子质量分辨率进行模拟。中微子质量等其他研究也需要N-body模拟达到4~5个数量级的动态范围,所需要的粒子数量超过万亿。

对于粒子数较大的情况,直接求和法消耗的庞大计算开销是无法承受的。因为这种粒子-粒子对(Particle-Particle,PP)的引力计算复杂度为$O(N^2)$。目前已有许多算法可以将引力求解的计算复杂度降低到$O(N\log N)$,例如基于树的方法和基于粒子网格混合型(Particle Mesh,PM)的方法。在宇宙学N-body模拟中,很少需要单个粒子的精确轨迹,只需要

到稿日期:2020-02-25 返修日期:2020-06-11 本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家重点研发计划(2016YFB0201800,2018YFA0404603)
This work was supported by the National Key R&D Program of China (2016YFB0201800,2018YFA0404603).
通信作者:林新华(james@sjtu.edu.cn)

最终的模拟结果符合理论的统计分布即可满足研究所需的精度。在这种情况下，基于 PM 改进的算法通常可以拥有较快的求解速度，同时满足精度要求，适合粒子规模较大的模拟。

但是，即使在目前世界上最大的超算平台使用基于 PM 的算法，还是会受到内存容量的限制。例如，世界上最大的 N-body 体模拟之一 TianNu[5] 使用 P3M 算法[6] 在天河二号超级计算机上完成了包含 3 万亿粒子的宇宙学 N-body 体模拟。因为 P3M 算法所需的计算资源和内存资源不均衡，该模拟在使用了天河二号全系统所有内存的情况下，只利用了 30% 的计算资源。因此，在现代超算平台上进行更大规模的模拟面临的主要挑战是如何减少其巨大的内存消耗。

为了应对这一挑战，可以使用定点化压缩[7]，以获得尽可能低的内存消耗。N-body 模拟通常需要大量内存来存储粒子的位置和速度信息。如果使用单精度或双精度浮点数存储粒子的三维位置和速度信息，则每个粒子就需要占用 24 字节或 48 字节。通过使用优化的定点化压缩算法，可以利用定点数将粒子的状态信息进行压缩并存储在内存中。定点化压缩可以显著地减少内存占用，最低可以达到 6 bpp(byte per particle)，比传统 PM 算法低近一个数量级，从而打破了大规模 N-body 模拟所面临的内存容量瓶颈。文献[7]在宇宙学意义上，对压缩方法做了可行性分析和正确性验证。

本文使用 C 语言在高性能计算平台上实现了基于定点压缩技术的双层粒子网格算法，并针对混合精度计算性能和扩展性进行了深度优化。本文的主要贡献如下：

1)使用 C 语言，基于 MPI 和 OpenMP 的混合并行模式，实现了基于定点压缩技术的双层粒子网格算法；

2)基于定点压缩的方法，将存储每个粒子相空间信息的内存消耗减少到最低 6 个字节，比传统 PM 算法低近一个数量级；

3)针对定点压缩带来的性能损耗，在混合精度计算、通信等方面进行了性能和可扩展性的优化，将压缩和解压在程序总耗时中的占比从 21% 降低至 8%，并且在核心计算热点上达到了最高 2.3 倍的加速效果。

## 2 相关工作

### 2.1 N-body 算法

使用计算机进行天文 N-body 模拟最早可追溯到 1970 年 Peeble[8] 在计算机上完成了包含 300 个粒子的 N-body 模拟。而随着高性能计算硬件和软件技术的高速发展，现代超算平台已经可以支持超过万亿个粒子的 N-body 模拟[9-11]。最初，复杂度为 $O(n^2)$ 的 PP 方法很快被诸如 PM 的方法取代。随后，由于 PM 方法无法提供足够高的分辨率和模拟精度，继而演化出了 P3M(Particle-Particle PM)方法和 TreePM 方法。相对于 PM 方法，P3M 方法和 TreePM 方法可以在相同复杂度的情况下达到更高的精度，是目前宇宙学 N-body 模拟中最为流行的方法。目前最为流行的 N-body 软件主要包括基于 TreePM 方法的 GADGET-2[12] 和基于 P3M 的 CUBEP3M 等。

2012 年 Gordon-Bell 奖[13] 的工作使用了基于 TreePM 的 N-body 算法，在 K Computer 上使用 82 944 个节点进行了 1 万亿粒子的 N-body 模拟，并达到了 5 PFlops 的峰值计算性能。但是，该模拟从红移 400 演化至红移 31，这不是一个典型宇宙学 N-body 覆盖的红移范围。这个红移范围的计算特征也与典型红移范围下的计算特征有所区别。其次，因为该模拟中的 FFT 采用了平面分解(slab-decomposed)的方法，扩展性较差，所以该模拟采用较粗的 PM 格点划分。而 TianNu 模拟[5] 是基于 CUBEP3M，使用天河二号 14 000 个节点累计计算 96 h，成功地完成了 3 万亿粒子的数值模拟，是当时世界上粒子数目最多的宇宙学 N-body 模拟。在 TianNu 模拟结果中发现的中微子凝聚效应，推动了中微子质量研究领域的前沿研究。

### 2.2 数据压缩

数据压缩被广泛用于高性能计算[14-15] 和人工智能领域[16-17]，减少了应用程序的存储或内存消耗。来自计算机数值模拟、科学观察和实验的高精度数值数据通常以浮点数进行表示，可以达到 TB 到 PB 的存储量。将如此大的数据集通过网络在计算节点之间甚至内存与磁盘之间来回移动成为了严重制约科学计算应用的瓶颈。在该类型应用上应用数据压缩技术，可以大大减少存储和移动的数据量。在现代超算平台上，一些应用也会在内存中使用压缩技术，以降低内存带宽或者 PCIe 带宽限制带来的性能损失。

## 3 算法设计与实现

设计并实现高性能的 N-body 程序面临许多挑战，主要包括：保持较小的内存占用量并支持较大的动态范围，最大程度减少跨计算节点通信，加速对大型数组的内存访问，高效利用高性能计算库来加速计算过程(如快速傅里叶变换)等。

### 3.1 算法设计

本文采用了基于两层架构的引力求解算法。在引力求解过程中，其混合网格方法和粒子方法。其中，网格方法更适合对较大动态范围的引力进行快速求解，粒子方法更适合对较小动态范围的引力进行精确求解。为了计算较大动态范围的引力，混合两种引力计算技术，需要在代码设计和结构上进行创新，整体代码遵循格点计算的方式，在底层级上又可以与粒子计算的方式相结合。遵循这个核心的设计，使用混合式的并行结构，将引力求解分为基于网格的长程力和中程力，以及基于粒子的短程力。而根据计算平台架构的特点和模拟所需要的精度，可以调整短程力的计算范围。

通过混合网格方法和粒子方法，将引力的计算分为不同的部分，使其可以同时具备格点方法较高的计算效率，又可以通过粒子方法提升模拟结果的精确度。

使用两层 PM 的长程力和中程力独立进行计算，并累加得到粒子受到的引力，以此更新粒子的速度。这样的计算方法需要将空间划分为两个层级的格点。先将整个空间均匀划分成 $N_c^3$ 个粗格点(coarse mesh)，再将每个粗格点继续细分为 $R^3$ 个细网格(fine mesh)，所以整个空间包含 $N_f^3 = R^3 N_c^3$ 个细网格。第一层 PM，也就是长程力部分，是在粗格点上进行计算的，需要完整模拟区域的信息，并且涉及跨节点的 MPI 通信。第二层 PM，即中程力部分，是在细格点上进行计算的，只需要模拟局部区域的信息，不涉及跨节点间的通信。

粒子-粒子相互作用力需要计算一定范围的细网格内所有粒子之间的相互作用力，以此更新粒子的速度。为了避免

粒子过于接近时产生的碰撞效应，需要将距离小于软化长度 $r_{soft}$ 的粒子对的相互作用力设置为零。累积每个粒子受到来自附近细格点所有其他粒子的引力，得到该粒子受到的局部粒子引力，用来更加精确地更新粒子速度。

### 3.2 算法实现

在算法实现中，本文采用了 PM 算法常用的并行模式。网格在 3 个维度方向上被均匀划分，并分配到不同的计算节点上进行计算；同时在每个节点分配的区域周围设置一定宽度的缓冲区域，任何在这个节点中的粒子在速度更新时如果超出了本网格的物理范围，就会进入缓冲区域；在每个时间步结束时，物理空间上相邻的节点需要交换缓冲区中的所有粒子，并更新粒子密度场和速度场，再进行下一步迭代。算法的整体流程如图 1 所示。

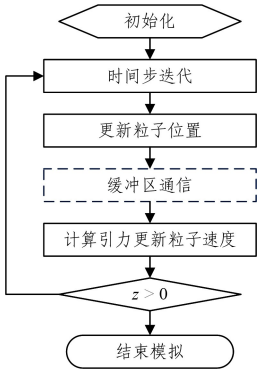



图 1 模拟的整体流程

Fig. 1 Workflow of simulation

在综合考虑计算效率和内存消耗的情况下，程序采用双层划分的数据划分方式，如图 2 所示。首先，将整个模拟的宇宙空间按照 3 个维度划分为子立方体，每个子立方体都被分配给一个 MPI 进程进行独立计算。第二层次的数据划分发生在每个 MPI 进程内部，其中子立方体被三维分解为多个称为“切片”(Tile)的局部立方体，每个切片包含相同数量的粗格点。切片将依次计算，并使用 OpenMP 在线程级并行化切片内部的计算。由于全局空间在两个层次上进行三维分解，因此进程的数量必须是立方数。

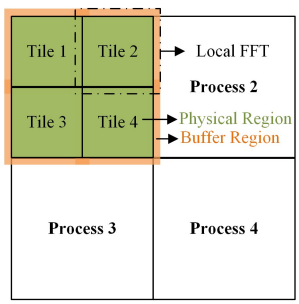



图 2 空间分解的二维示意图

Fig. 2 Two-dimensional schematic of spatial decomposition

## 4 算法优化

在使用基于双层粒子网格的算法进行较大规模的模拟时，仍然会面临内存容量、计算性能和扩展效率等挑战。因此，本文在双层粒子网格算法的基础上，引入了 3 种算法优化技术，包括粒子存储格式优化、计算优化和扩展性优化。

### 4.1 粒子存储格式优化

N-body 模拟中最消耗内存的部分通常是粒子相空间数组，该相空间数组将每个粒子的位置和速度坐标信息存储为 6 个浮点数(单精度 24 bpp，双精度 48 bpp)。其包括物理区域和缓冲区域中的粒子相空间。除了粒子相空间数组，实际程序中还包括全局粗网格和局部细网格相关的链表、数组和与 FFT 相关的内存占用。

但是，浮点数提供的数值精度高于模拟所需要的精度，因此可以使用定点化技术进行信息压缩以减少模拟所消耗的内存。定点化技术通过使用定点数表示(1 字节或 2 字节)存储粒子相对的相空间坐标，而不是用全局值来解决此问题。通过密度场和速度场(可忽略的内存)以及相空间数组中粒子的排序关系(无需额外的内存)，可以将局部坐标恢复为全局坐标。通过这种定点化压缩技术，可以将粒子相空间数组的内存消耗从28 bpp 减少到 6 bpp。

使用定点化技术压缩粒子相空间信息，可以显著减少模拟过程中所需要的内存总量，同时可以降低模拟检查点(Checkpoint)所需要的存储总量。在粒子缓冲区域通信中，因为只需要传输压缩后的粒子信息，所以通行部分对网络带宽的消耗有了显著降低。对于进行较大规模的 N-body 模拟，定点化压缩技术的引入可以打破现代超算平台的内存限制的瓶颈，并进一步提高程序的 I/O 性能和扩展性。

#### 4.1.1 粒子位置压缩

在模拟过程中，整个模拟空间被划分成均匀的网格，每个粒子都属于网格中的某个单元。因此在存储粒子位置信息时，不需要存储全局坐标，而是存储其相对于包含该粒子的父单元的偏移量。我们将单元的每个维度 $d$ 平均分为 $2^8=256$ 个单元，并使用 1 个字节的整数 $\chi_d \in \{-128,-127,\cdots,127\}$ 表示这个粒子在这个维度上属于哪个单元。粒子的全局位置由内存空间中的顺序格式给出，并且该网格中的粒子数量(密度场)将提供有关粒子在网格中分布的完整信息。结合粒子的全局位置，可以计算出粒子在第 $d$ 维 $x_d$ 的全局坐标为 $x_d=(n_c-1)+(\chi_d+128+1/2)/256$，其中 $n_c=1,2,\cdots,N_c$ 为粗网格上的索引。

如果需要更高精度的模拟结果，则可以使用 2 个字节的整数代替。更为一般的粒子位置转换公式如下：

$$\chi_d=\lfloor 2^{8n_\chi}(x_d-\lceil x_d\rceil)\rfloor-2^{8n_\chi-1} \tag{1}$$

$$x_d=(n_c-1)+2^{-8n_\chi}(\chi_d+2^{8n_\chi-1}+1/2) \tag{2}$$

其中，$n_\chi \in \{1,2\}$ 是定点数存储所需要的字节数，$x_d$ 和 $\chi_d$ 是粒子位置坐标的浮点数表示和定点数表示。

#### 4.1.2 粒子速度压缩

类似地，在第 $d$ 维上，粒子的速度 $v_d$ 被分解为同一粗网格上的平均速度场 $v_c$ 和粒子相对于该场的残差 $\Delta v$：

$$v_d=v_c+\Delta v$$

使用单独的数组来维护 $v_c$，并在粒子速度更新后重新计算 $v_c$。因为粗网格的数量远少于粒子的数量，所以维护平均速度场的内存消耗是比较低的。由于在实际情况中，特别是在低红移阶段，粒子的速度不是均匀分布的，速度较低的粒子比速度较高的粒子丰富，且粒子速度的分布范围会随着模拟发生较大的变化。因此，我们选择将速度空间 $\Delta v$ 分为不均

匀的区间，并使用 $n_v$ 个字节的定点数来表示粒子位于哪个 $\Delta v$ 区间。在粒子分布较为丰富的区域采用更为细致的划分，在粒子分布较为稀疏的区域采用更为粗略的划分，从而在较少字节的表示上可以充分地利用定点数的表达范围，并根据模拟过程动态调整表达的动态范围，获得比均匀划分更好的精度。

粒子速度信息的浮点数表示和定点数表示的转换公式如下：

$$V_d=\left[(2^{8n_v}-1)\pi^{-1}\arctan((v_d-v_c)\sqrt{\pi/(2\sigma_\Delta^2)}\right] \tag{3}$$

$$v_d=v_c+\tan\left(\frac{\pi V_d}{2^{8n_v}-1}\right)\sqrt{2\sigma_\Delta^2/\pi}) \tag{4}$$

其中，$n_v\in\{1,2\}$ 是定点数存储所需要的字节数；$v_d$ 和 $V_d$ 是粒子速度的浮点数表示和定点数表示；$\sigma_\Delta$ 是每个粗网格中粒子速度分布的方差，该方差会在每次粒子速度更新后重新统计并更新。

### 4.2 计算优化

定点化压缩大大降低了 N-body 模拟所需的内存占用，并且降低了缓冲区域的粒子通信所需要的通信总量。但是，压缩和解压的过程会带来额外的计算成本，并且会降低程序的向量化程度。压缩和解压的过程占程序总耗时的 21%，影响了程序整体的计算效率。本文提出了计算预先缓存和低精度近似函数等针对压缩数据的优化技术，极大地降低了压缩和解压带来的性能损耗。

#### 4.2.1 解压计算优化

在涉及粒子位置和速度的计算中，通常需要先对压缩后的定点数表示进行解压，再进行计算。较慢的解压速度会降低整体的计算效率。例如，由粒子速度解压公式（见式(4)）可知，解压粒子的速度信息十分耗时，降低了粒子速度计算部分的运行效率。为了降低解压带来的额外开销，可以采用计算预先缓存技术，即预先计算所有定点数表示的解压结果并将其存储在一个缓存数组中。在进行预先缓存计算时，只需要遍历定点数的所有表示，求解其对应解压的浮点数结果，并使用缓存数组进行存储即可；在后续需要使用粒子速度的计算中，可以直接访问缓存数组获得解压结果代替解压计算，从而减少了解压带来的性能损耗。由于低比特数的定点数表示数量有限，对于 1 字节定点数仅存在 256 个表示，2 字节定点数存在 65536 个表示，因此时间和内存成本均可以忽略不计。

在 PP 引力计算中，需要计算一定范围内每对粒子之间的欧几里得距离。由于粒子位置被压缩为定点数，在这种情况下无法直接使用传统的 AVX512 向量化指令进行优化加速，需要将粒子的位置信息解压成浮点数，然后使用向量化指令进行加速，解压过程降低了 PP 整体的向量化程度和并行效率。因此，可以使用 Intel 的 AVX512 VNNI（矢量神经网络指令）扩展（Intel Cascade Lake 处理器开始支持）直接对定点化格式下的数据进行引力计算，而不需要经过解压步骤。VPDPBUSD 指令将第一个源地址的各个字节（8 位）与第二个源地址的对应字节（8 位）相乘，产生中间结果（16 位），并将它们累加到目标地址的双字（32 位）中。将 16 对粒子的 $x$-$y$-$z$ 轴相对矢量填入源地址，并填充零，然后使用 VPDPBUSD 和 VSQRTPS 指令直接计算获得 16 对粒子距离计算的结果。在理论情况下，与传统 AVX 指令相比，使用 VPDPBUSD 指令可以增加 3 倍的操作吞吐量，并将内存占用减少为原来的 1/3。

#### 4.2.2 压缩计算优化

使用定点化压缩技术将单精度浮点数（32 位）转换为 1 字节或 2 字节整数时，某些计算代价较高的高精度数学函数是冗余的。因此，可以使用低精度的近似函数来代替它们，以减少定点化压缩带来的性能损耗，而不会影响最终科学结果的准确性。例如，在粒子速度压缩式(3)中的 $\arctan(x)$ 函数，我们可以使用二次多项式作为其近似函数。在使用低精度近似函数时，其与原始函数的最大绝对误差远低于粒子信息被定点化压缩表示所带来的精度误差。因此，低精度近似函数可以在不影响最终模拟结果精度的情况下，降低原始函数的计算代价。

### 4.3 扩展性优化

本文针对两个主要的通信部分，即全局 3D-FFT 计算和缓冲区域通信，分别进行了扩展性优化的工作。

#### 4.3.1 全局 3D-FFT

我们使用 PFFT 库[16]支持程序中全局 3D-FFT 的计算。PFFT 是针对分布式内存体系结构设计的大规模并行快速傅里叶变换软件库。PFFW 基于 FFTW-MPI 或 MKL，对高维数据分解有更好的支持；分布并行优化也有着更好的扩展效率。

#### 4.3.2 缓冲区域通信

在缓冲区域通信部分，每个网格需要将缓冲区域的粒子位置和速度在相邻的网格之间传输。为了减少缓冲区域通信的开销，我们使用了通信整合策略和 3D MPI 进程分配策略。

使用通信整合策略将缓冲区域通信的部分进行整合，减少了调用 MPI 进行发送和接收的次数，并使用 MPI 提供的单边通行的异步模式减少了通信部分所占用的时间。

使用 3D MPI 进程分配策略代替默认的 1D MPI 进程分配，使原来在物理域上相邻的进程可以尽可能地分布在同一个节点上。在程序中，每个节点分配 8 个进程，默认的 1D MPI 进程分配策略会按照一维的方式将连续的进程分配在一个节点上，而 3D MPI 进程分配策略根据进程所对应的物理区域进行分配，将互相可以组成立方体的进程分配在一个节点上。3D MPI 进程分配策略可以有效地减少跨节点的缓冲区域通信，从而提高程序整体的扩展效率。

## 5 实验设计

### 5.1 实验环境

所有性能和扩展性测试均在上海交通大学超级计算机 π2.0 上完成。π2.0 有 650 个计算节点，峰值性能为 2 PFlops。每个节点配备两颗 Intel Cascade Lake 6248 处理器（包含 20 个核心）和 192 GB DDR4 内存。其中，Cascade Lake 是第一代支持 AVX512 VNNI 扩展指令集的架构。集群中，所有节点都通过 Intel 100Gbps Omni-Path Architecture（OPA）互相连接。

本文使用 C 语言，基于 MPI 和 OpenMP 的混合并行模式进行了算法实现。因此，我们使用了 Intel C/C++ Compiler18.0.5 编译套件完成程序的编译，并使用 Intel MPI 2018（Update 4）支持实现中的 MPI 部分。

### 5.2 实验方法及参数配置

本文进行了一系列实验，以验证提出的优化技术的效果，

并评估程序整体的计算性能和扩展效率。

所有的实验都保持相同的宇宙学模型和参数。在初始化宇宙初始条件时，使用 Zel'dovich 逼近确定红移 $z=99$ 时粒子的初始位置和速度，然后使用主程序将宇宙演化至红移 $z=0$。模拟使用 Hubble 参数 $H_0=72\,\mathrm{kms}^{-1}\mathrm{Mpc}^{-1}$ 的 ΛCDM 宇宙模型进行建模，冷暗物质密度 $\Omega_{ch}^2=0.214\times0.72^2=0.1109$，重子密度 $\Omega_b h^2=0.044\times0.72^2=0.0228$，初始条件参数 $\sigma_8=0.80$，$n_s=0.96$。

在测试压缩优化实验中，使用 1 字节的定点数格式进行粒子相空间的压缩存储。在测试压缩计算优化时，每个进程负责 $256^3$ 个粗格点(全模拟共 $4096^3$ 个)。每个进程分配的空间被进一步分解为 8 个切片。每个切片在每个维度上有 512+2×6=524 个细格点(物理区域有 512 个，缓冲区有 2×6 个)。

## 6 实验结果

### 6.1 压缩优化结果

图 3 展示了一个 MPI 进程的内存消耗。按照程序的主要模块划分，可以将每个进程的内存消耗划分为 3 个部分：粒子、粗格点和细格点。粒子部分包括物理区域和缓冲区域中粒子相空间信息的内存占用，占程序总内存消耗的 72.40%，是内存占用中最主要的部分。粗格点和细格点分别表示与粗网格和细网格相关数组和 FFT 部分的内存占用，分别占程序总内存消耗的 17.30% 和 9.36%，其中粗格点和细格点的 FFT 部分均采用本地计算(In-Place Transform)的模式，以减小 FFT 部分的内存消耗。每个 MPI 进程总计的内存消耗为 13.75 GB，包含 $1024^3$($=1.074\times10^9$)个粒子。因此，该程序的内存消耗为 $13.75\times10^9/(1.074\times10^9)=12.8$ bpp。

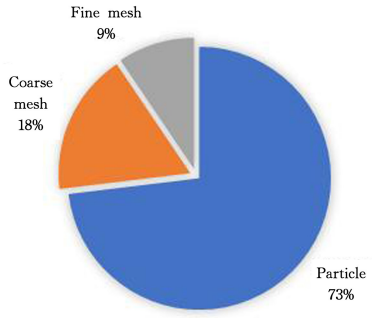



图 3 单进程的内存消耗

Fig. 3 Memory consumption for one MPI process

由于使用了两层 PM 算法和定点压缩技术技术，该程序的内存消耗明显小于其他主流的宇宙学 N-body 模拟代码。例如，TianNu[5] 在天河二号上进行了 2.97 万亿个粒子的模拟，平均的内存消耗为 186 bpp，是该程序的 14.5 倍。

### 6.2 计算优化结果

图 4 展示了应用 4.2 节的算法优化后，各个计算 kernel 的性能提升。可以看到，利用预先计算缓存、混合精度计算等技术，可以将压缩和解压在程序总耗时中的占比从 21% 降低至 8%，极大地降低了定点化压缩给整体程序带来的性能损耗，提高了程序整体的计算效率。

预先计算缓存和低精度近似函数主要作用在 update_x(粒子位置更新)kernel 中，分别获得了 120% 和 20% 的性能提升。在 PP(粒子互作用力计算)kernel 中，主要使用了混合精度等技术，共获得了 2.3 倍的加速。通信部分使用 PFFT 库进行优化，在 particle_mesh(3D FFT)部分获得了 20% 的提升。通信整合和 3D MPI 进程分配策略为缓冲区域通信(buffer)部分带来了 34% 的性能提升。

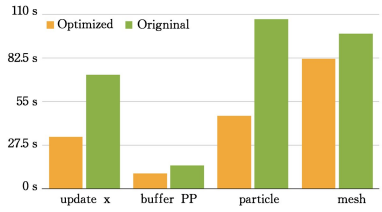


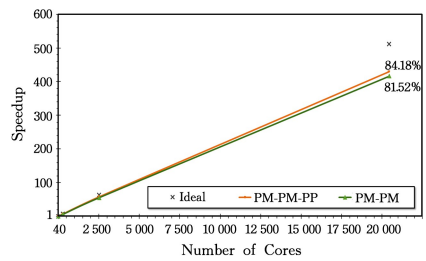

图 4 各 kernel 的性能提升

Fig. 4 Performance improvemen to feach kernel

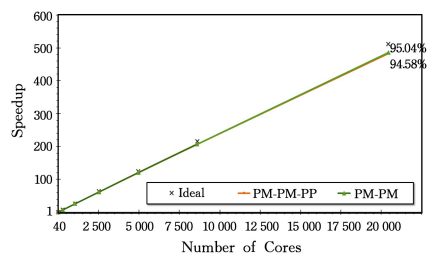
### 6.3 扩展性优化结果

为了进行弱扩展性测试，我们在实验中保持每个 MPI 进程处理 $1024^3$ 个细格点来进行$(200\,\mathrm{Mpch}^{-1})^3$ 体积的模拟，并逐渐从 40 颗核心扩展到 20480 颗核心。图 5(a)显示了有无 PP 力(图例中为 PM-PM-PP 和 PM-PM)时的弱扩展效率。在两种情况下，程序都实现了 95% 左右的并行效率，表现出了几乎完美的线性扩展。该程序相比其他的 N-body 代码拥有更好的扩展效率。比如，使用 CUBEP3M 的 TianNu 模拟在天河二号上的弱扩展效率为 72%[17]。

在强扩展性测试过程中，我们固定模拟规模为$(200\,\mathrm{Mpch}^{-1})^3$，其中包含 $1024^3$ 个细格点。在全局规模固定的情况下，逐渐增加核心数量，并在每个进程中分配相应数量的细格点。图 5(b)显示了程序的强扩展效率。在扩展到 20480 个核心时，程序在 PM-PM 和 PM-PM-PP 下分别保持 81.5% 和 84.2% 的并行效率。随着核心数量的增加，强扩展性降低，其原因主要来自两个方面：计算时间和通信时间的比例降低，以及缓冲区域与物理区域的比例增加。随着核心数量的增加，更多比例的时间花费在通信和缓冲区域的计算上，使得强扩展性有较为明显的降低。




(a)弱扩展




(b)强扩展

图 5 弱扩展和强扩展效率

Fig. 5 Weak-scaling and strong-scaling efficiency

**结束语** 本文设计实现并优化了基于定点压缩技术的双层粒子网格算法。基于定点压缩技术，将存储每个粒子相空间信息所需的内存容量降低至 12.8 bpp，比传统 N-body 算法低了近一个数量级，本文针对混合精度计算性能进行了深度优化。这些优化技术显著降低了定点压缩带来的性能损耗，将压缩和解压在程序总耗时中的占比从 21%降低至 8%，并且在核心计算热点上到达了最高 2.3 倍的加速效果。同时，本文针对程序的通信部分进行了优化，并成功扩展到 2 万核心以上。

实验结果表明，使用定点压缩技术优化的双层粒子网格算法在大规模宇宙学 N-body 模拟方面具有巨大潜力。在接下来的工作中，我们提出了两个改进方向。1)将该方法移植到异构平台(如英伟达 GPU 计算平台)上，利用 GPU 在混合精度计算方面的优势，对程序中耗时较多的部分进行移植和加速，在模拟速度上可以获得更大的优势。2)将该方法移植到国产处理器平台上，使用超过千万核心进行数值模拟计算。利用更大数量的核心和内存，可以使得问题的规模提升 2 到 3 个数量级，例如在“神威·太湖之光”平台上进行更大规模的并行[18-19]。这些大规模的 N-body 模拟将对宇宙大尺度模拟和其他宇宙学方向的研究产生深远影响。

## 参 考 文 献


[1] FENG L, ZHU W. The simulation techniques and applications in modern cosmology[J]. SCIENTIA SINICA Physica, Mechanica & Astronomica, 2013(6): 1.

[2] SI Y, WEI J, SEE S, et al. Parallel Design and Optimization of Galaxy Group Finding Algorithm on Comparation of SGI and Distributed-memory Cluster [J]. Computer Science, 2017, 44(10): 80-84.

[3] YANG X, FENG L, ZHE Y. Wavelet power spectrum analysis of cosmic large-scale structures: methods and numerical simulation tests[J]. Science in China(Series A), 2001, 31(3): 278-288.

[4] HUANG W. Neutrino Mass and the Superstructure of the Universe[J]. HIGH Energy Physics and Nnclear Physics, 1991, 15(12): 1135-1136.

[5] YU H R, EMBERSON J, INMAN D, et al. Differential neutrino condensation onto cosmic structure [J]. Nature Astronomy, 2017, 1(7): 1-5.

[6] HARNOIS-DÉRAPS J, PEN U L, ILIEV I T, et al. High-performance P3M N-body code: CUBEP3M[J]. Monthly Notices of the Royal Astronomical Society, 2013, 436(1): 540-559.

[7] YU H R, PEN U L, WANG X. CUBE: An Information-optimized Parallel Cosmological N-body Algorithm[J]. The Astrophysical Journal Supplement Series, 2018, 237(2): 24.

[8] PEEBLES P J, YU J. Primeval adiabatic perturbation in an expanding universe [J]. The Astrophysical Journal, 1970, 162: 815.

[9] ISHIYAMA T, ENOKI M, KOBAYASHI M A, et al. The ν2GC simulations: Quantifying the dark side of the universe in the Planck cosmology[J]. Publications of the Astronomical Society of Japan, 2015, 67(4): 61.

[10] HEITMANN K, FRONTIERE N, SEWELL C, et al. The Q continuum simulation: harnessing the power of GPU accelerated supercomputers[J]. The Astrophysical Journal Supplement Series, 2015, 219(2): 34.

[11] HEITMANN K, FINKEL H, POPE A, et al. The Outer Rim Simulation: A Path to Many-core Supercomputers[J]. The Astrophysical Journal Supplement Series, 2019, 245(1): 16.

[12] SPRINGEL V. The cosmological simulation code GADGET-2 [J]. Monthly Notices of the Royal Astronomical Society, 2005, 364(4): 1105-1134.

[13] ISHIYAMA T, NITADORI K, MAKINO J. 4.45 Pflops astrophysical N-body simulation on K computer — The gravitational trillion-body problem [C] // Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. 2012: 1-10.

[14] LINDSTROM P. Fixed-rate compressed floating-point arrays [J]. IEEE Transactions on Visualization and Computer Graphics, 2014, 20(12): 2674-2683.

[15] LINDSTROM P, ISENBURG M. Fast and efficient compression of floating-point data[J]. IEEE Transactions on Visualization and Computer Graphics, 2006, 12(5): 1245-1250.

[16] PIPPIG M. PFFT: An extension of FFTW to massively parallel architectures[J]. SIAM Journal on Scientific Computing, 2013, 35(3): C213-C236.

[17] EMBERSON J, YU H-R, INMAN D, et al. Cosmological neutrino simulations at extreme scale[J]. Research in Astronomy and Astrophysics, 2017, 17(8): 85.

[18] WANG Y, LIN J, CAI L, et al. Porting and Optimizing GTC-P on TaihuLinght Supercomputer with Sunway OpenACC [J]. Journal of Computer Research and Development, 2018, 55(4): 875-884.

[19] MENG D, WEN M, WEI J, et al. Porting and Optimizing OpenFOAM on Sunway TaihuLight System[J]. Computer Science, 2017, 44(10): 64-70.


**CHENG Sheng-gan**, born in 1997, bachelor. His main research interests include heterogeneous computing and parallel computing.

**James LIN**, born in 1979, Ph.D, associate professor, is a senior member of China Computer Federation. His main research interests include HPC and so on.