ISSN 2096-742X
CN 10-1649/TP

文献DOI：
10.11871/jfdc.issn.
2096-742X.2020.
02.013

文献PID：
21.86101.2/jfdc.
2096-742X.2020.
02.013

页码：155-164

开放科学标识码
（OSID）

基于FMM-PM方法的宇宙N体模拟在GPU上的实现和优化

扶月月[1,2]，王武[1*]，王乔[2,3]

1. 中国科学院计算机网络信息中心，北京 100190
2. 中国科学院大学，北京 100049
3. 中国科学院国家天文台，北京 100101

摘　要：【目的】本文在多 GPU 平台上，对基于快速多极子方法（FMM）和粒子网格方法（PM）的天文 N 体模拟软件 PHoToNs 的核心函数进行 CUDA 加速实现和性能优化。【方法】主要优化方法包括算法的参数优化、页锁定内存和 CUDA 流优化、混合精度和快速数学库优化等。【结果】优化后的短程力相互作用核心函数在 Titan V 的 GPU 平台上采用 4 张 GPU 卡的计算速度相对采用 4 个 Intel Xeon CPU 核提高了约 410 倍。【结论】本文的优化技术可为其它高性能 GPU 异构平台上的进一步算法研究和超大规模天文 N 体模拟提供支撑。

关键词：N体模拟；快速多极子方法；GPU；优化

The Implementation and Optimization of Cosmological N-Body Simulation by FMM-PM Method on GPUs

Fu Yueyue[1,2], Wang Wu[1*], Wang Qiao[2,3]

1. Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China
2. University of Chinese Academy of Sciences, Beijing 100049, China
3. National Astronomical Observatories, Chinese Academy of Sciences, Beijing 100012, China

Abstract: **[Objective]** In this paper, the kernel functions of PhoToNs, which is an astronomical N-body simulation software based on the fast multipole method (FMM) and particle grid method (PM), are accelerated and optimized for CUDA on a multi-GPU platform. **[Methods]** The main optimization methods adopted in CUDA kernels include: algorithm parameter optimization, use of page-locked memory and CUDA streams, and use of mixed precision and fast math library. **[Results]** The kernel function of short range force interaction is deeply optimized, which achieves a speedup of about 410 times faster on four Titan V GPUs than the pure MPI code running on four Intel Xeon CPU cores. **[Conclusions]** Optimization methods in this paper can support further algorithm research and hyper-scale N-body simulation on other high performance GPU-based heterogeneous platforms.

Keywords: N-Body simulation; fast multipole method; GPU; optimization

基金项目：国家重点研发计划项目“宇宙学高性能异构模拟系统”(2017YFB0203302)；中国科学院“十三五”信息化专项“科研信息化应用工程”(XXH13506-405)；中国科学院战略性先导科技专项（C类）（XDC01040100）

*通讯作者：王武（E-mail:wangwu@sccas.cn）

引言

N 体问题计算相互作用的 N 个粒子的相互作用和运动，是高性能计算中具有代表性和挑战性的问题之一，在天体物理、分子动力学、磁流体动力学等领域具有广泛的应用。模拟 N 体问题的数值方法包括直接法（Particle-Particle，PP）、树方法（Tree Method）和粒子网格方法 (Particle Mesh，PM) 等，其中树方法有 BH 方法 (Barnes-Hut)[1] 和快速多极子方法 (Fast Multipole Method，FMM)[2]。PP 方法的复杂度为 $O(N^2)$，其中 N 为粒子规模，这限制了模拟问题的规模；BH 方法的复杂度为 O(NlogN)，且计算精度可控；FMM 的复杂度为 O(N)，兼具精度和速度优势，被 IEEE 计算机协会和美国计算物理学会列为 20 世纪十大算法之一 [3]；基于快傅里叶变换（Fast Fourier Transform，FFT）的 PM 方法 [4] 复杂度为 O(NlogN)，虽然计算速度较快，但计算短程力时不精确，只适用于长程力计算。为了弥补 PM 方法计算精度不足的缺点，PM 混合算法应运而生，PM 方法结合直接法和树方法的混合算法（P3M[5]、TreePM[6]）应用广泛。

随着高性能计算技术的发展，以及天文学和分子动力学等领域对大规模计算需求的增长，仅从算法上降低计算复杂度或者增加 CPU 核数量并不足以满足领域科学家的计算需求。采用硬件加速来加快计算速度，从而扩大计算规模也是较为可行的途径。GPU 的最新发展已经能够以低廉的成本提供高性能的通用计算，基于 CPU+GPU 的并行异构架构也已成为当今大型超级计算机的主流体系结构之一。例如，2019 年 11 月全球 TOP500 排行榜前 10 名的超级计算机中，有 5 台使用了 CPU+GPU 的异构体系结构。

近些年 N 体问题成为了 GPU 上的热点应用之一，国际上有不少针对各个算法使用 GPU 来加速 N 体模拟的研究，很多显著的研究成果都是对 PP 方法的 N 体模拟软件进行 GPU 优化加速 [7-9]，基于 BH 方法和 FMM 方法在大规模 GPU 集群上的性能提升的研究成果也不少 [10-12]，但是利用 CPU+GPU 异构平台对 PM 混合算法进行优化加速的研究并不多，目前还没有在 GPU 集群上实现 FMM-PM 耦合算法的 N 体模拟。本文采用 FMM-PM 耦合的方法实现 N 体问题的模拟，既可兼顾计算速度和精度，又能重叠短程力的计算与长程力的通信，因此较适合大规模并行。我们的工作为今后对多体问题和异构体系结构的研究工作奠定了坚实的基础。

宇宙模拟软件 PHoToNs 是针对宇宙暗物质大尺度结构设计的 N 体模拟软件 [13]，其第二个版本改进了引力的计算算法，采用了 FMM-PM 耦合算法实现，其中 FMM 的核心函数在 GPU 上实现，FMM 中计算量较小的算子、PM 函数和通信函数在 CPU 上实现，本文给出了该软件在 CPU+GPU 架构上的实现过程和性能优化技术。第 1 节介绍 FMM-PM 耦合算法的基本原理和计算流程，第 2 节介绍 FMM 在 GPU 上的实现策略，第 3 节介绍 FMM 在 GPU 上的三种性能优化方法，第 4 节给出测试结果。

1 FMM-PM 算法原理

1.1 快速多极子方法

天文 N 体问题主要计算引力场中的 N 个粒子（星体、星系或暗物质都可以抽象为粒子）的相互作用和运动，粒子所受作用力可表示为：

$$F_i = \sum_{j=1}^{N} \frac{G m_i m_j r_{ij}}{|r_i - r_j|^3},$$

其中 $i = 1,2,...,N$，G 是引力常量，m_i 和 m_j 是粒子 i 和 j 的质量，$r_{ij} = x_i - x_j$ 为粒子 i 到 j 的位移矢量。采用该公式直接计算每对粒子之间的相互作用，其精度较高且易于并行实现，然而需要每次重复计算各个时间步，这个过程的时间复杂度为 $O(N^2)$，模拟问题的粒子规模大大受限。

FMM 将粒子的相互作用分为远场和近场作用，远场作用通过近似方法来计算，在近场作用的计算

过程中，通过层次划分和位势函数的多极子展开计算各点位势，然后将位势转换为各点所受的力。位势和引力满足：

$$\Phi(r_i)=\sum_{j=1}^{N}\frac{Gm_im_j}{|r_i-r_j|},F(r_i)=-\nabla\Phi(r_i)$$

其中 i = 1,2, ... N, m_i, r_i, $\Phi(r_i)$，$F(r_i)$ 分别为粒子的质量、位置、位势和作用力。虽然近场部分使用直接法，但由于短程力作用范围有限，近场作用的粒子对数量远小于远场作用，所以 FMM 整体复杂度仍为 $O(N)$。

FMM 的核心思想是：首先将空间进行多层组划分，然后通过核函数多极子展开，将粒子的位势聚集到所在盒子的中心，逐层向上递归计算各层盒子的展开系数，最后把粒子间的远场相互作用转换为盒子中心之间的相互作用，逐层向下递归到所有叶盒子。FMM 树结构递归过程的示意图见图 1，向上箭头表示 M2M 逐层聚集，向下箭头表示逐层发散，水平虚线表示次相邻转移。

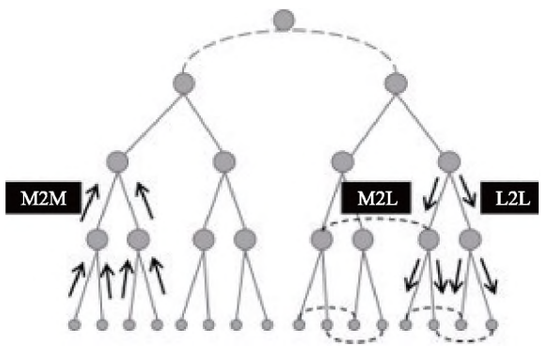

图 1 FMM 树结构算法示意图

Fig.1 FMM tree structure algorithm diagram

FMM 逐层递推过程包括以下七个基本步骤：

(1) 空间多层盒子划分，自下而上构造树；

(2) 计算每个叶盒子的多极子展开系数和粒子与叶盒子的相互作用（P2M），然后逐层上聚到父盒子中心（M2M）；

(3) 计算次相邻盒子的相互作用，即级数展开系数的转换（M2L）；

(4) 逐层向下计算各盒子的局部展开系数以及父盒子对子盒子的作用（L2L）；

(5) 采用直接法计算每个叶盒子和相邻盒子里所有粒子的近相互作用（$F_{near\text{-}field}$）；

(6) 计算所有粒子的位势和受到的长程力相互作用（$F_{far\text{-}field}$）；

(7) 计算所有粒子的加速度，基于蛙跳格式更新下一时刻的速度和位置。

1.2 粒子网格方法

粒子网格（Particle Mesh，PM）方法基于谱空间的位势场泊松方程求解粒子系统的引力，它应用于具有周期边界的 N 体模拟。PM 方法根据粒子信息得到均匀网格上的密度分布函数，求解出网格点的位势，然后根据粒子所处的网格单元位置，通过差分和插值得到每个粒子所在点的位势和所受的力。PM 方法的基本步骤如下：

(1) 构造密度分布：基于粒子的分布情况计算构造均匀网格，以及格点的密度分布函数；

(2) 估算位势：根据密度分布ρ和引力势泊松方程$\Delta\Phi=4\pi G\rho$，算出格点的位势，通常采用快速傅里叶变换方法（FFT）或多重网格（MG）等算法，本文采用 FFT；

(3) 计算作用力：通过有限差分和插值方法从格点的位势得到粒子所受的力；

(4) 计算加速度：根据粒子质量，将每个粒子的受力转换为加速度；

(5) 速度和位置更新：基于时间步进积分（可采用蛙跳格式、龙格 - 库塔格式等）计算各粒子下一时刻的速度和位置。

与 PP 方法相比，PM 方法的网格点数量与粒子数相当或者更少，而且基于规则网格的密度场分布通过谱空间的 FFT 求解各点的引力势，仅涉及到两次 FFT，其复杂度为 $O(N_glogN_g)$，其中 N_g 为离散网

格点数，所以 PM 计算速度较快，但是如果粒子间的距离较小时，该方法算出的粒子受力计算精确不能满足要求，因此 PM 只适合计算粒子的长程力相互作用。

1.3 FMM-PM 耦合方法

耦合方法把引力势拆分成长程和短程部分，其中长程部分通过谱空间位势的傅立叶变换算出，短程部分通过位形空间泊松方程求解得到，计算公式如下：

$$\Phi_k^{long} = \Phi_k \, exp(-k^2 r_s^2) \, ,$$

$$\Phi_k^{short} = -G \sum_i \frac{m_i}{r_i} erfc\left(\frac{r_i}{2r_s}\right)$$

其中Φ_k^{long}和Φ_k^{short}分别表示长程引力势和短程引力势，r_i为粒子 i 相对 x 的距离，r_s为长程和短程分界的特征空间尺度，k 为谱空间的波数，$erfc()$ 为误差修正函数。

Hockney 等人在 1974 年提出 P3M[5] 方法，采用 PP 方法计算短程力，它早期用于等离子体模拟，后来应用于宇宙模拟；Xu G 等在 1995 年提出 TreePM[6] 方法，采用树方法计算短程力，TreePM 方法带来计算性能的显著提升，在天体物理和分子动力学中得到广泛应用。目前国际主流的宇宙 N 体模拟软件就是采用 TreePM 方法（如 GADGET[14]）。

天文 N 体模拟软件 PHoToNs-2.0 采用 FMM-PM 耦合算法，长程力采用基于 FFT 的 PM 方法求得，短程力通过 FMM 计算。基于 FFT 的 PM 方法虽然速度更快，但受限于精度，只适用于计算长程力，而且会带来全局通信。FMM 不仅具有近似线性的复杂度和较高的精度，而且适合并行。其中近场相互作用是计算密集型的，可有效利用高性能异构平台的加速器件（如 GPU、MIC 等）。采用 FMM-PM 耦合算法来实现 N 体问题的大规模并行，既兼顾了计算速度和精度，又重叠了短程力的计算与长程力的通信。

图 2 为 FMM-PM 耦合算法示意图，虚线内为 FMM 计算区域，其中粒子间的短程力相互作用（P2P）占 FMM 计算时间的 90% 以上，适合在加速卡上实现；虚线外的区域基于三维 FFT 的 PM 方法计算长程力，占整体运行时间的比例约 1%，因此不需要采用 GPU 加速，可在 CPU 端调用 FFT 函数库 (如 FFTW、P3DFFT 等) 实现，PHoToNs-2.0 采用二维区域分解的三维并行 FFT 函数库 2decomp-FFT[15]。

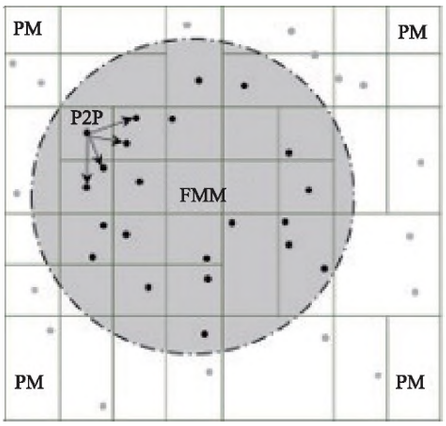

图 2 FMM-PM 耦合算法示意图

Fig.2 FMM-PM algorithm diagram

2 FMM 在 GPU 上的实现

FMM 的本地和非本地短程作用（P2P）的计算时间占 FMM-PM 运行时间的比例最高（对于大规模问题，通常在 90% 以上），考虑到 P2P 算子是计算密集型和数据密集型，可采用 GPU 进行加速。FMM 其它计算（构造树，盒子 - 粒子作用等）是访存、传输密集型函数，计算量相对较小。核心函数 P2P_kernel 是计算短程力的主要函数，其作用是通过粒子的位置、质量等信息算出粒子的受力和加速度。P2P_kernel 在 GPU 上实现的过程如下：

(1)CPU 获取任务队列，做数据传输准备，开辟显存空间；

(2)CPU 把需要计算的粒子信息传输到显存；

(3)GPU 上计算粒子的短程力相互作用；

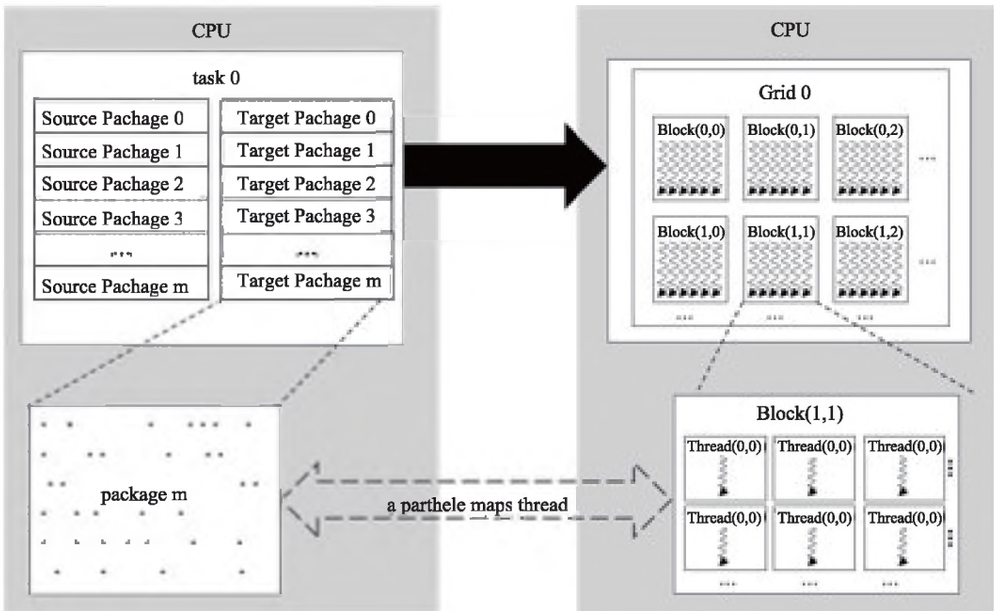

图 3 计算任务（包）与 GPU 线程映射关系示意图

Fig.3 Mapping relationship diagram between computing tasks (packages) and GPU threads

(4) GPU 把运算结果通过显存传输回 CPU 主存；

(5) CPU 端更新粒子信息。

在具体实现，把按顺序存放的粒子信息以及其邻居粒子分批存入 CPU 中开辟的缓冲池中，之后一起存入 GPU 显存中保证完整性，而且使 GPU 有序访问显存，提高存取效率。如图 3 所示，根据 CUDA 编程模型，一个线程映射一个粒子信息，充分发挥 GPU 计算线程较多的优势。

3 FMM 在 GPU 上的性能优化

3.1 核心函数的参数优化

在 FMM 中 MaxPackage 参数至关重要，它表示树结构最细层粒子包中的粒子数上限，其大小与任务量和数据传输开销有关。随着 MaxPackage 增大，虽然单次执行核函数的任务量增大，但总任务次数减小，总数据传输开销也随之减小。反之，MaxPackage 减小，总数据传输开销增大。短程力数据传输的复杂度可表示为：

$$T_{trans} = O\left(\frac{K * N_{part}}{MaxPackage}\right)$$

其中 N_{part} 表示每个进程的粒子数，常数 K 表示每个盒子的近场相邻盒子数，包括该盒子本身，取值与截断半径和判断准则有关。

MaxPackage 参数分别与数据传输量和数据传输时间关系如图 4 和图 5 所示。

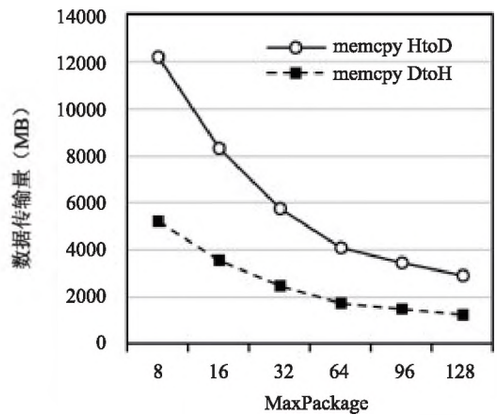

图 4 MaxPackage 与数据传输量的关系

Fig.4 The relationship between MaxPackage and the amount of data transmitte

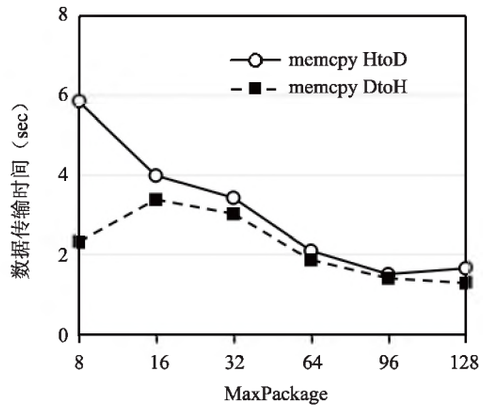

图 5 MaxPackage 与数据传输时间的关系

Fig.5 The relationship between MaxPackage and the Data transfer time

图 6 为使用 nvprof 工具分析内存拷贝时间与核心函数 p2p_kenel 的执行时间分别占 P2P 计算总执行时间的比重。其中，memcpyHtoD 和 memcpyDtoH 分别表示从主机端到设备端的内存拷贝和从设备端到主机端的内存拷贝。可见，在多 GPU 系统中，数据传输时间占总 P2P 短程作用力的计算总时间的 90% 以上。虽然短程力计算量与 MaxPackage 成正比，但测试表明在 GPU 上传输比计算更耗时间，即 P2P 计算复杂度可表示为：

$$T_{p2p} = O(K*N*MaxPackage)$$

因此，在 GPU 上通过增大 MaxPackage 参数，可以减少数据传输开销，进一步优化 P2P 短程作用力的计算和传输总时间。

3.2 页锁定内存优化和 CUDA 流优化

对 CUDA 架构而言，传统的内存拷贝方式是由操作系统 API malloc( ) 在主机上分配的同时用 cudaMalloc( ) 分配设备内存，这种内存，方式称为可分页内存（page-able memory）。可分页内存的内存拷贝非常耗时，为了加速内存拷贝，CUDA 提供了页锁定内存（page-lock memory 或 pinned memory）机制来而分配主机内存，即采用 API cudaHostAlloc( )

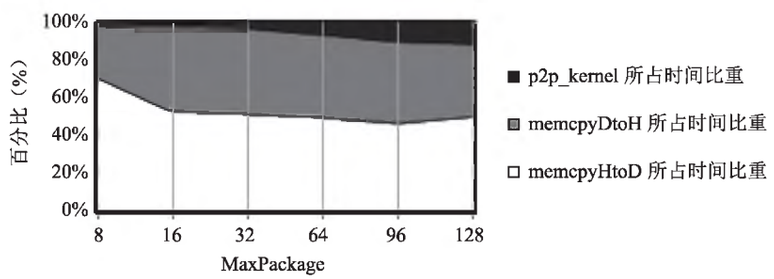

图 6 memcpyHtoD、memcpyHtoD 以及 p2p_kernel 所占的时间比重

Fig.6 Time proportion of memcpyHtoD, memcpyHtoD and p2p_kernel

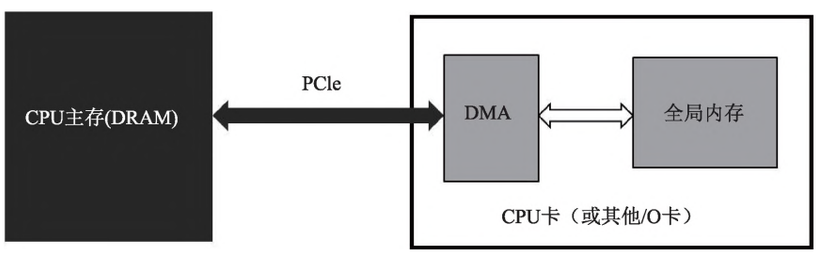

图 7 CPU 与 GPU 之间的内存交互示意图

Fig.7 Memory interaction diagram between CPU and GPU

分配主机内存。使用页锁定内存方式分配的主机内存会常驻物理内存中，操作系统将不会对该内存进行分页和交换到磁盘上，这块内存不会被破坏或者重新定位，所以操作系统能够安全地使某个应用程序访问该内存的物理地址。CPU 与 GPU 内存交互方式如图 7 所示，CPU 和 GPU 之间的总线是 PCIe，是双向传输的，CPU 和 GPU 之间的数据拷贝使用 DMA 机制来实现。

在进行内存复制操作时，若采用传统的可分页方式，CUDA 驱动程序仍会通过 DRAM 把数据传给 GPU，这样复制操作会执行两遍：先从可分页内存复制一块到临时的页锁定内存，再从这个临时的页锁定内存复制到 GPU 上。当从可分页内存中执行复制时，复制速度将受限制于 PCIE 总线的传输速度和系统前段速度相对较低的一方。使用了页锁定内存时，GPU 可以直接确定内存的物理地址，从而能够使用 DMA 技术在 GPU 和 CPU 之间拷贝数据。显然，使用页锁定内存大大减少了 CPU 与 GPU 之间数据传输时间，最大化内存吞吐量，提高内存带宽。

CUDA 流包括一系列异步的 CUDA 操作（主机与设备间的数据传输，内核启动以及由主机发起但由设备处理的一些命令）。CUDA 流按照主机程序确定的顺序在设备端执行，相对于主机是异步的。可以使用 CUDA 的 API 来确保一个异步操作在运行结果被使用前完成。在同一个 CUDA 流中的操作有严格的执行顺序，而在不同 CUDA 流中的操作在执行顺序上不受限制。

使用多个 CUDA 流同时启动多个 kernel 可实现网格级并发，因为所有排队的操作都是异步的，所以在主机与设备系统中可重叠执行部分操作。在同一时间内将流中排队的操作与其他并发操作一起执行，可以隐藏部分操作的时间开销，使设备利用率最大化。图 8 为使用 2 个 CUDA 流的性能提升示意图，可以看出，在一个流中设备端的计算 kernel 与另一个流中设备到主机的数据传输是重叠的。

3.3 混合精度和快速数学库优化

单精度和双精度浮点运算在通信和计算上的性能差异是不可忽略的。在我们的程序中，使用双精度数值能够使程序运行总时间增加近一倍 (这个结果取决于程序是计算密集型还是 I/O 密集型)，在设备端进行数据通信的时间也是使用单精度的两倍，这是由于传输数组的字节长度相差两倍。随着全局内存输入 / 输出数量和每条指令执行的位操作数量的增加，设备上的计算时间也会增加。为了最大化指令吞吐率，我们使用单精度数 (float) 常量、变量和单精度计算函数，仅在有必要的时候使用双精度运算。

内部函数 __fdividef 与除法运算相比，在执行浮点除法时速度更快但数值精度相对较低。用功能上等价的 __fdividef 来替换除法操作，可以加速程序运行。另外，快速数学库（Fast Math Library）可以

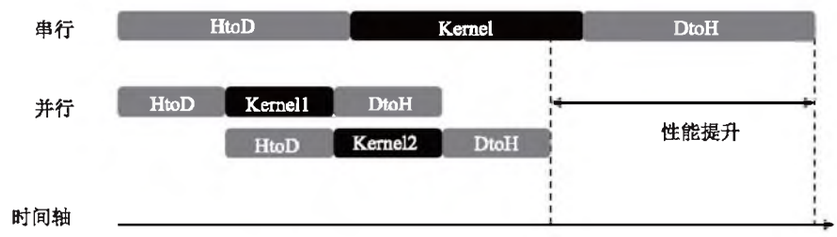

图 8 使用 2 个流的 CUDA 操作性能提升示意图

Fig.8 Performance improvement diagram of CUDA operation using 2 streams

将核函数中使用的单精度计算函数替换为 CUDA 内部实现的高速版本。我们对核函数中单精度浮点类型的超越函数（erfc）调用快速数学库，程序性能大大提升。

4 数值结果

在浪潮 NF5280M5 服务器上进行测试。系统环境配置：2 个 Intel Xeon Gold 6148（20 核心 *2），主频为 2.4GHz，12 个内存条 DDR4 2400 DDR4（32GB * 12）；4 张 NVIDIA TITAN V GPU 卡，显存容量 12GBS，CUDA 版本为 9.1，采用 Intel 编译器和“-O3”优化。

对优化后的 CUDA 版本程序进行加速效果测试，粒子规模为两百万（128^3），使用 4 个进程，在浪潮服务器上单个时间步的模拟。图 9 为不同 MaxPackage 三个实现版本程序的数据传输时间对比。其中，版本 1 是未进行优化的 CUDA 程序，版本 2 是表示使用页锁定内存和 CUDA 流优化后的程序，版本 3 为使用混合精度调用快速数学库优化的程序。可见，MaxPackage 越大，数据传输越快，系统性能越好。

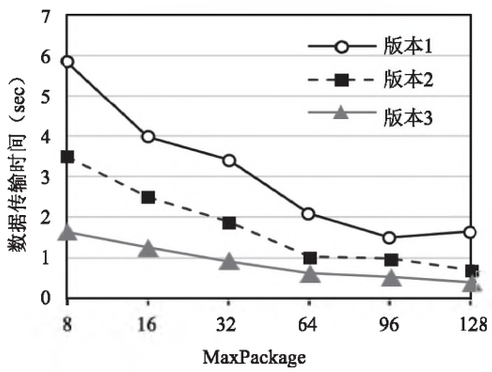

图 9 不同 MaxPackage 三个优化版本的数据传输时间对比

Fig.9 Data transfer time comparison between three optimized versions of different MaxPackage

图 10 为 PHoToNs-2.0 的纯 MPI 程序和三个 CUDA 实现版本程序的短程力计算函数 P2P 运行时间的对比，图 11 为三个 CUDA 实现版本采用不同 MaxPackage 值时 P2P 的加速倍数。可以看出，当 MaxPackage 为 128 时，核心函数 P2P 执行时间从 475.9s 降到了 1.16 秒，加速了 410.3 倍。

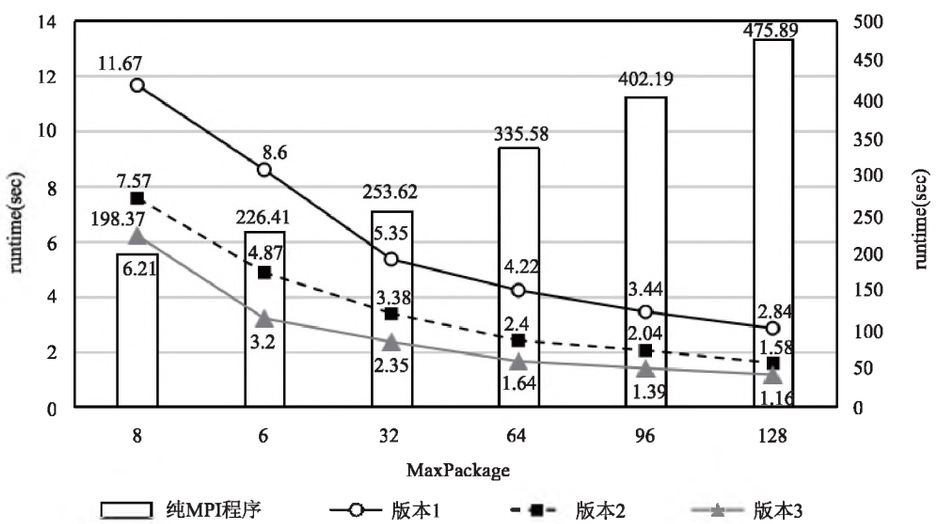

图 10 纯 MPI 程序与三个 CUDA 实现版本程序的 P2P 函数执行时间对比

Fig.10 Comparison of P2P function execution time between pure MPI program and three CUDA implementation version programs

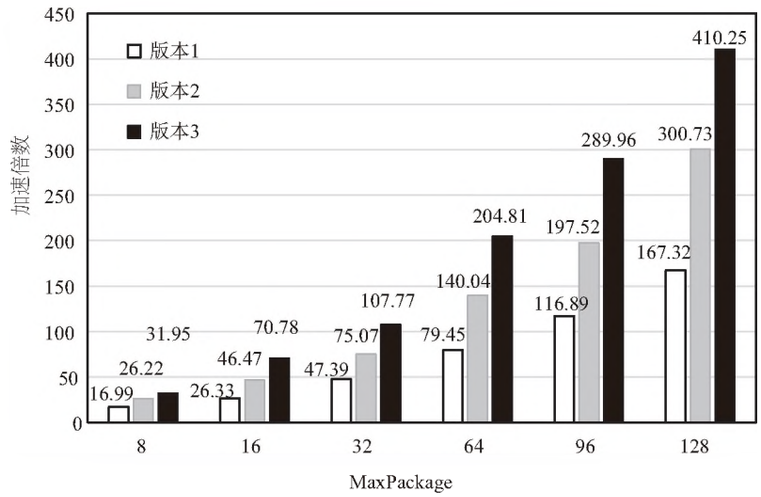

图 11 不同 MaxPackage 三个 CUDA 实现版本程序的 P2P 函数加速倍数

Fig.11 P2P acceleration multiples of three CUDA versions of different MaxPackages

5 结论与展望

本文在 CPU+GPU 混合异构平台的，对基于快速多极子方法和粒子网格方法的 N 体模拟软件 PHoToN-2.0 进行 CUDA 实现和性能优化，包括优化粒子包的参数以减少数据传输开销、使用页锁定内存提高内存带宽和最大化内存吞吐量、使用 CUDA 流优化最大化设备利用率、采用混合精度优化最大化指令吞吐率、有效利用快速数学库等手段。最终，FMM-PM 耗时 90% 以上的核心函数 P2P 在在 Titan V 的 GPU 平台上采用 4 张 GPU 卡的计算速度相对采用 4 个 Intel Xeon CPU 核提高了约 410 倍。这些优化方法为在大规模 GPU 异构平台上进一步优化和超大规模天文 N 体模拟奠定了基础。

利益冲突声明

所有作者声明不存在利益冲突关系。

参考文献

[1] Barnes J, Hut P. A hierarchical O(nlogn) force-calculation algorithm [J]. Nature, 1986, 324: 446-449.

[2] Greengard L, Rokhlin V. A Fast Algorithm for Particle Simulations [J]. Journal of Computational Physics, 1997, 135(2):280-292.

[3] Dongarra J, Sullivan F. Guest Editors Introduction to the top 10 algorithms [J]. Computing in Science and Engineering, 2000, 2(1):22-23.

[4] Aubert D, Amini M, David R. A Particle-Mesh Integrator for Galactic Dynamics Powered by GPGPUs [C]. Proceedings of 9th International Conference on Computational Science, 2009, 874-883.

[5] Hockney R W, Goel S P, Eastwood J W, et al. Quiet high resolution computer models of a plasma[J]. Journal of Computational Physics, 1974, 14(2): 148-158.

[6] Xu G. A new parallel N body gravity solver: TPM[J]. Astrophysical Journal Supplement Series, 1995, 98(1):335-363.

[7] Hamada T, Nitadori K, Benkrid K, et al. A novel multiple-walk parallel algorithm for the Barnes-Hut treecode on GPUs - Towards cost effective, high performance N-body simulation[J]. Computer Science - Research and Development, 2009, 24(1-2):21-31.

[8] Miki Y, Takahashi D, Mori M, et al. Highly scalable implementation of an N-body code on a GPU cluster[J]. Computer Physics Communications, 2013, 184(9): 2159-2168.

[9] Wang L, Spurzem R, Aarseth S J, et al. nbody6++gpu: ready for the gravitational million-body problem[J]. Monthly Notices of the Royal Astronomical Society, 2015, 450(4): 4070-4080.

[10] Robert G Bellemana,Jeroen Bédorfa,Simon F Portegies Zwart. High performance direct gravitational N-body simulations on graphics processing units II: An implementation in CUDA [J]. New Astronomy, 2008,13(2):103-112.

[11] Bédorf J, Gaburov E, Portegies Zwart S. A sparse octree gravitational N-body code that runs entirely on the GPU processor[J]. Journal of Computational Physics, 2012, 231(7): 2825-2839.

[12] Potter D, Stadel J, Teyssier R, et al. PKDGRAV3: beyond trillion particle cosmological simulations for the next era of galaxy surveys[J]. Computational Astrophysics and Cosmology, 2017, 4(1):2-14.

[13] Wang Q, Cao Z Y, Gao L, Chi X B, Meng C, PHoToNs–A parallel heterogeneous and threads oriented code for cosmological N-body simulation [J]. Research in Astronomy and Astrophysics, 2018, 18(6):9-18.

[14] Springel V, Yoshida N, White S. Gadget: a code for collisionless and gasdynamical cosmological simulations [J]. New Astronomy, 2001, 6(2):79-117.

[15] Ning L, Laizet S, 2DECOMP&FFT – A highly scalable 2D decomposition library and FFT interface [C]. Cray User Group 2010 conference, Edinburgh, 2010.

收稿日期：2020 年 2 月 4 日

扶月月，中国科学院计算机网络信息中心，硕士研究生，研究方向为并行计算。
本文承担的工作为设计和实现快速多极子方法在 GPU 上的实现和优化。

Fu Yueyue is a master student at Computer Network Information Center, Chinese Academy of Sciences. Her main research interest is parallel computing algorithm.
In this paper, she undertakes the following tasks: design, optimization and implementation of fast multipole method on GPU.
E-mail: fuyueyue@cnic.cn

王武，中国科学院计算机网络信息中心，博士，副研究员，研究方向为并行算法、高性能计算。
本文承担的工作为设计指导快速多极子方法在 GPU 上的实现和优化。

Wang Wu, Ph.D., is an associate research fellow at Computer Network Information Center, Chinese Academy of Sciences. His main research interests are parallel computing algorithm and high performance computing.
In this paper, he is the director for design, optimization and implementation of the fast multipole method on GPU.
E-mail: wangwu@sccas.cn

王乔，中国科学院国家天文台，博士，副研究员，研究方向为宇宙大尺度结构、计算宇宙学。
本文承担的工作为设计和实现并行快速多极子方法与粒子网格方法。

Wang Qiao, Ph.D., is an associate research fellow at National Astronomical Observatories, Chinese Academy of Sciences. His main research interests are cosmic large scale structure and computational cosmology.
In this paper, he undertakes the following tasks: design and implementation of the parallel fast multipole method and particle mesh method.
E-mail: qwang@nao.cas.cn

引文格式：扶月月,王 武,王 乔.基于FMM-PM方法的宇宙N体模拟在GPU上的实现和优化[J]. 数据与计算发展前沿,2020,2(2):155-164.DOI:10.11871/jfdc.issn.2096-742X.2020.02.013.PID:21.86101.2/jfdc.2096-742X.2020.02.013.
Fu Yueyue, Wang Wu, Wang Qiao.The Implementation and Optimization of Cosmological N-Body Simulation by FMM-PM Method on GPUs[J].Frontiers of Data & Coputing,2020,2(2): 155-164.DOI:10.11871/jfdc.issn.2096-742X.2020.02.013.PID:21.86101.2/jfdc.2096-742X.2020.02.013.