计算机科学
COMPUTER SCIENCE

一种基于GPU的核苷酸分子系统发育树条件似然概率可扩展并行计算方法

黄佳为，李晓鹏，凌诚

引用本文

黄佳为，李晓鹏，凌诚.一种基于GPU的核苷酸分子系统发育树条件似然概率可扩展并行计算方法[J]．计算机科学, 2022, 49(11A): 210800189-7.

HUANG Jia-wei, LI Xiao-peng, LING Cheng. Scalable Parallel Computing Method for Conditional Likelihood Probability of Nucleotide Molecular Phylogenetic Tree Based on GPU [J]. Computer Science, 2022, 49(11A): 210800189-7.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210800189

一种基于GPU的核苷酸分子系统发育树条件似然概率可扩展并行计算方法

黄佳为　李晓鹏　凌　诚
北京化工大学信息科学与技术学院　北京 100000
(867454915@qq.com)

摘　要　贝叶斯与Metropolis-Hastings算法的高效实现让MrBayes成为使用广泛的分子序列系统发育分析工具。然而,分子序列与进化参数的增加导致候选分子树样本空间急剧扩大,使得系统发育树的重构工作面临巨大计算挑战。为降低MrBayes系统发育分析中分子树条件似然概率的计算时间,提高分析效率,近年来出现一批基于图形处理器(GPU)的并行加速方法。为提高并行方法的可扩展性,提出了一种优化的似然概率多线程并行计算方法。根据位点间可变进化速率模型中分子状态似然概率的计算需要对应不同转移概率矩阵,将前期使用多线程对不同位点似然概率的并行计算,进一步分解为多位点间不同转移概率矩阵下的条件似然概率的计算。该策略在不改变单个线程计算传输比的基础上,通过增加线程数量,优化了线程warp间的并行重叠度,提高了并行效率。此外,由于每个线程warp只计算同一种转移概率矩阵下的似然概率,避免了在使用共享内存时不同warp间的同步开销,进一步提升了内核计算效率。所提方法与前期方法在4组实际数据和30组模拟数据上的计算结果表明,在核心似然函数的计算加速上,本文取得的计算性能超过tgMC^3(2.0版)和nMC^3(2.1.1版)方法,最高达1.78和2.04倍。

关键词:MrBayes;似然计算;GPU;并行计算;CUDA编程
中图法分类号　TP399

Scalable Parallel Computing Method for Conditional Likelihood Probability of Nucleotide Molecular Phylogenetic Tree Based on GPU

HUANG Jia-wei, LI Xiao-peng and LING Cheng
School of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100000, China

Abstract　The efficient implementation of Bayesian and Metropolis Hastings algorithms makes Mrbayes a widely used tool for molecular sequence phylogenetic analysis. However, the increase of molecular sequences and evolutionary parameters leads to the rapid expansion of the sample space of candidate molecular trees, which makes the reconstruction of phylogenetic trees face great computational challenges. In order to reduce the calculation time of conditional likelihood probability of molecular tree in mrbayes phylogenetic analysis and improve the analysis efficiency, a number of parallel acceleration methods based on graphics processor (GPU) have emerged in recent years. In order to improve the scalability of parallel methods, an optimized likelihood probability multithreaded parallel computing method is proposed in this paper. As the calculation of molecular state likelihood probability in the variable evolution rate model between sites needs to correspond to different transition probability matrices, this method further decomposes the parallel calculation of likelihood probability of different sites using multithreading into the calculation of conditional likelihood probability under different transition probability matrices between multiple sites. This strategy optimizes the parallel overlap between threads and improves the parallel efficiency by increasing the number of threads without changing the calculation transmission ratio of a single thread. In addition, because each thread warp only calculates the likelihood probability under the same transition probability matrix, it avoids the synchronization overhead between different warps when using shared memory, and further improves the computing efficiency of the kernel. Calculation results of 4 groups of actual data and 30 groups of simulated data show that the computational performance of this method is 1.78 and 2.04 times higher than that of tgMC^3 (version 2.0) and nMC^3 (version 2.1.1) in the calculation acceleration of core likelihood function.

Keywords　MrBayes, Likelihood computing, GPU, Parallel computing, CUDA

1　引言

系统发育学是研究进化关系的一门学科,有别于个体发育,它关注类群的形成和发展过程,即系统发育。系统发育树是一种对物种进化过程的可视化描述,系统发育树的客观分析研究方法的发展是进化理论和生物信息学的重要版块。系统发育分析在各种生命科学领域的应用越来越广泛,如药物发现、医学诊断、人类流行病学、病毒传播、法医学和保护[1-2]。

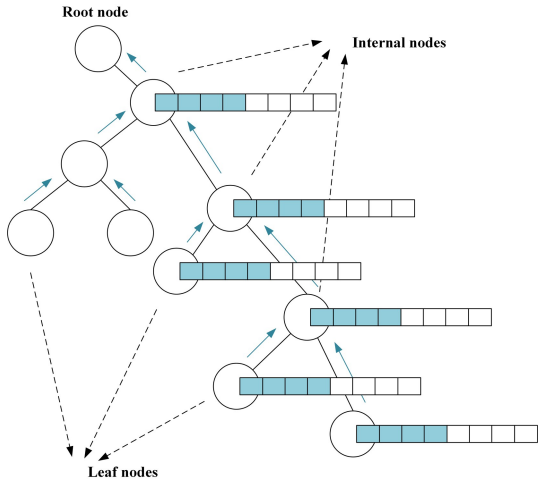
基金项目:国家自然科学基金(61602026)
This work was supported by the National Natural Science Foundation of China(61602026).
通信作者:凌诚(lingcheng@mail.buct.edu.cn)

计算机科学和生物学的进步促进了分子序列数据的快速增长，也推动了重建系统发育树方法的快速发展。在这些方法中，应用最广泛的是最大似然法（Maximum Likelihood，ML）[3]和贝叶斯推理法[4]，它们的共同点是利用经验进化模型来估计系统发生树的似然概率。不同的是，ML 使用统计学的方法，通过给予特定的系统发育树的似然概率，来估计置换模型中的替换概率。贝叶斯方法假设每棵概率树具有先验概率分布，通过建立马尔可夫链蒙特卡罗（Markov Chain Monte Carlo，MCMC）算法和不同分子数状态移动规则进行空间采样，重新评估新状态的似然概率后计算分子树的后验概率[5-7]。后验概率的值随着相邻样本间似然的估计、先验比和建议比的变化而变化。最后，从这些树状态的后验分布中总结出最终一致树。

随着类群数量的增加，被评估的候选分子树的数量将呈指数增长，给系统发育树的分析过程带来巨大的计算挑战。为解决由于数据规模成倍增加所带来的计算问题，近些年以来产生了在不同的平台下的系统发育分析并行计算方法，如，基于 CPU 多核[8]和图形处理器（Graphic Processing Unit，GPU）[9]的高性能计算方案。此外，线程处理粒度的缩小推动着多层混合并行计算方案的发展，例如，在单机上进行三层混合并行计算[10]。并行架构在处理高密度的计算问题下能显著提高计算效率，有助于处理快速增长的生物序列数据的难题，因此计算速度的提高在系统发生的实践中具有重要意义。在同等浮点算力条件下，基于 Nvidia Compute Unified Device Architecture（CUDA）的通用显卡，比基于 CPU 多线程的大型集群的计算成本更低。MrBayes 是一种使用广泛的系统发育分析工具，对其而言，gMC^3[11]方法是第一个基于 GPU 的 MrBayes 似然概率并行方法，该方法通过分配大量的 GPU 线程并行处理计算分子树内部节点的条件似然概率（Conditional Likelihood Probabilities，CLP）。在此基础上，为了提高 Host 与 Device 之间的数据传输效率，并增加进程与内核间的计算并行度，Zhou 等提出了一种改进的 GPU 并行方法，即 nMC^{3}方法[12]。Bao 等通过动态调整任务粒度适应输入序列和硬件配置，同时开发了一种自适应的 DNA 序列分离和组合方法优化了 nMC^{3}方法[13]。Ling 等[14]提出 tgMC^{3}方法，通过合并离散内核减少了内核启动次数和冗余数据的传输，从而大大提高了计算性能。此外，Ling 等为了提高数据存取效率，减少 bank 冲突，将 CLP 矩阵从行存储转成列存储，降低了 GPU 多线程计算 CLP 的复杂度，在 4 块 Tesla K40 GPU 上取得了 178 倍的加速比[15]。Kuan 等提出了 sMC^{3}方法[16]，利用任务级与数据级并行性，最小化内核启动与 CPU-GPU 数据传输的开销，相比原始的 MrBayes 取得了高达 103 倍的应用加速，同时相比使用 BEAGLE[17]加速库的 MrBayes 取得了 3.3 倍加速比。BEAGLE 是一个用于系统发育分析的高性能似然计算库，BEAGLE 3[18]实现了新的并行方案，改进了可伸缩性以及可用性。在一个分区实例中，运行性能较之前提高了 5.9 倍。

到目前为止，MrBayes 的所有并行优化方法中，对于较长序列集的系统发育分析加速效果较好的是 nMC^3（2.1.1 版）和 tgMC^3（tgMC^3++，2.0 版）方法，本文针对 tgMC^{3}方法中存在的计算瓶颈，提出一种根据转移概率矩阵数量进行 GPU 线程扩展的并行方法。与前期方法在 4 组实际数据和 30 组模拟数据中的计算结果表明，在主要核心的似然计算函数（down_3，down_12，down_0，后文介绍差别）的加速上，对 down_3 函数的加速性能最高达到 nMC^{3}和 tgMC^{3}方法的 1.29 和 1.22 倍；对 down_12 函数的加速性能最高达到 nMC^{3}和 tgMC^{3}方法的 1.97 和 1.71 倍；对 down_0 函数的加速性能最高达到 nMC^{3}和 tgMC^{3}方法的 2.04 和 1.78 倍。

MrBayes 是一种使用广泛的系统发育分析工具，它实现了基于贝叶斯推理、Metropolis-Coupled Markov Chain Monte Carlo（MCMCMC，MC^3）多链采样算法、似然函数计算以及多种分子进化模型在内的分子树样本后验概率估计。MC^{3}算法是 MCMC 的一种变体，可以实现运行多个马尔可夫采样链。每条采样链在候选分子树空间内，通过不同概率方法修改候选树状态，寻找具有最大似然值的系统发育树，采样过程在每次迭代更新分子树状态中完成。为了计算每条采样链上的分子树的似然概率，图 1 给出了一棵有根树的计算流程，从节点间蓝色的箭头指向可以发现，根节点的似然概率计算是从叶子节点逐级计算而来。以核苷酸序列为例，图中的每一排方框代表序列连续位点残基状态的 CLP，蓝色填充的方框描述了一个位点包含的 4 个 CLP，即 A，C，G，T 的概率值。叶子节点计算完成后，就从内部节点逐级向上计算，同时内部节点的计算依赖于子代节点，直到根节点计算完成后才结束该分子树的似然计算。

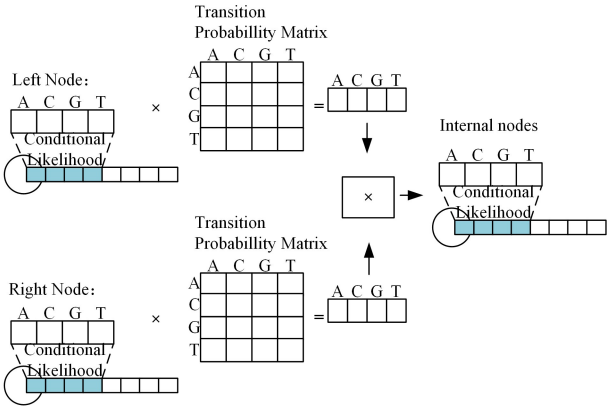


注：蓝色的小箭头表示似然概率计算的方向，虚线表示当前节点的属性

图 1 有根树拓扑结构和分子数节点（电子版为彩图）

Fig. 1 Root tree topology and molecule number nodes

图 2 给出了单个内部节点似然概率的计算流程。内部节点的 CLP 值计算由左右子节点的 CLP 值和 Tip（Transition Probabilities）值计算而来。提取子代左右节点的 A，C，G，T 的似然概率值，并加载当前进化速率下的 TiP 矩阵，使子代节点的 CLP 乘上 TiP 矩阵得到 CLP 中间值，最后对子代左右节点的中间值进行处理从而得到内部节点对应的似然概率值。贝叶斯系统发育分析过程中最为耗时的是位点的似然概率计算。通常情况下，MrBayes 在进行系统发育分析时默认使用 $GTR+I+\Gamma$，即广义时间可逆模型（General Time Reversible，GTR）加一定比例速率不变位点加位点间可变进化速率，可变位点间速率变化一般服从伽马分布曲线。为了计算的可行性，从伽马分布曲线中提取的 4 个离散的伽马参数表示 4 种不同的碱基替换速率[19]。



注：中间方框里的乘号表示将左右子节点计算的中间值相乘后得到当前节点的 CLP 值

图 2　内部节点的似然概率计算

Fig. 2　Likelihood probability calculation of the internal node

2　相关工作

2.1　MrBayes tgMC³ 方法概述

tgMC³ 是一种在 nMC³ 早期版本基础上发展出的 MrBayes MC³ 加速算法，nMC³ 方法的实现分为 5 个模块，每个模块有一个单独的内核实现。

(1)内核 1：通过 GPU 多线程对非终端节点的转换概率矩阵处理，并对转移概率矩阵转置后的矩阵(preLike 矩阵)填充模糊数据。并非所有非终端节点都需要执行该过程，因为其中一些可能没有叶子节点。

(2)内核 2：从全局内存加载转移概率矩阵或 prelike 矩阵到共享内存，以及它的子代的终端状态或 CLP，并在线程之间分配 CLP 的计算。计算的 CLP 值需要迭代地保存在全局内存中，直到到达根节点。

(3)内核 3：如果激活节点的 scalarsSet 参数，则该节点所属的每个站点的 lnscaler 值都要减去旧的 scaler，使用内核 3 来实现这个函数。同时，CPU 进程在 GPU 旁边翻转 scalersSet 位。

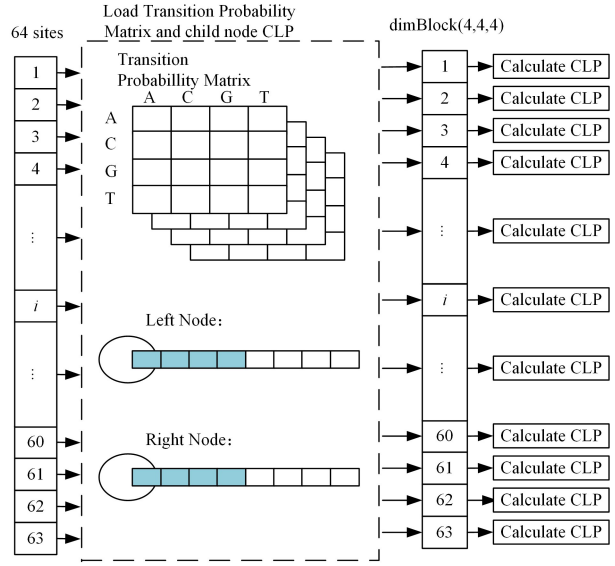
(4)内核 4：对于一个标量节点，MrBayes MC³ 规定将每个残差中的最大 CLP 值作为残差的标量，这只需遍历所有 CLP 即可实现。然后用这个新的标量除以剩余的所有 CLP。内核使用多个线程来遍历相应的 CLP 且对其进行并行扩展，这需要从全局内存重新加载 CLP。

(5)内核 5：根据进化设定中不变站点比例，核苷酸分子状态基础频率和每个位点的权重来计算根节点的似然概率。

由于 MrBayes 采样链从运行到结束有多达百万次的采样，每一次采样获得的分子树的分析都需要经过以上 5 个内核的调用，会造成较大的时间开销。tgMC³ 方法将这 5 个内核变成 1 个内核中的 5 个步骤，该方法(1.0 版)具有 3 个关键特征：1)将 Host 端似然概率计算的步骤集成在单个 CUDA 内核中完成运算，即采用了紧凑的 GPU 内核，减少了 Host 端与 Device 端的数据传输频率；2) 在单个内核中缓存需要频繁使用的参数，减少相同参数在 GPU 全局内存的存取次数和时间开销；3)针对数据集的特征，采用两种不同的任务并行化策略，充分利用 GPU 硬件资源。tgMC³ ++，即 tgMC³ 的 2.0 版，主要对 tgMC³ 的存储策略进行了优化。计算的过程分为 3 个步骤。第一步，对输入序列进行处理，由于为似然计算函数分配的线程块中包含的线程数为 64，因此将输入序列以 64 个位点为一组进行划分。当划分的最后一组数据数小于 64 个位点时，通过在数据的末端添加虚字符来完成对这组数据的填充处理，从而使输入的序列长度是每个线程块所能处理的位点数的整数倍。第二步，似然概率计算之前，需要把进行似然估计的数据从 Host 端传递到 Device 端的全局内存中，例如加载的数据 TiP，子代结点的 CLP 值。为了提高内核函数计算过程中参数的复用性，tgMC³ ++方法将重复使用的数据从 GPU 的全局内存加载到共享内存中，使线程获取数据的速度更快，从而提高计算效率。第三步，以内部节点似然计算为例，将子代左右节点的 TiP 加载到共享内存中，然后再将子代左右节点的 CLP 存储到寄存器中，线程从共享内存中获取 TiP 数据，从寄存器中获取子代节点 CLP 值后，计算该位点的似然概率。非常重要的一点是，为了解决行主序存储带来的数据冗余传递问题，tgMC³ ++方法采取列主序存储模式，保证多序列参数访问的一致性。

2.2　似然概率计算

根据分子树内部节点的后代节点属性，所有分子树内部节点可以分为 3 类：down_3，down_12 和 down_0。down_3 对应的节点的左右子节点均为叶子节点；down_12 对应节点的后代节点中 1 个为叶子节点，另 1 个为非叶内部节点；down_0 对应的 2 个子节点均为非叶内部节点。nMC³ 和 tgMC³ 方法针对这 3 类节点设计了对应的 GPU 内核函数。为统一描述，这里分别称为 gpu_down_3，gpu_down_12，gpu_down_0。以 gpu_down_0 函数为例，多线程的计算流程如图 3 所示。



注：每个位点似然估计之前需要加载转移概率矩阵和子节点的 CLP

图 3　gpu_down_0 函数中位点与线程的关系

Fig. 3　Relationship between position and thread in gpu_down_0 function

第 1 步　分配线程块 dimBlock(threadIdx.x，threadIdx.y，threadIdx.z)，其中 threadIdx.x 和 threadIdx.y 表示线程块行、列，Threadlx.z 表示线程块的层数。tgMC³ 方法设定线程块为 dimBlock(4,4,4)，前两个 4 表示线程块的大小为 4×4 的线程矩阵，第三个 4 表示线程块有 4 层；然后传递左右子节点的 CLP 和 TiP 存储位置信息，分别用 clL/clR，tiPL/tiPR 表示。

第 2 步　声明与似然估计相关的变量，如，表示左右子节点似然概率值的变量；申请 GPU 共享内存空间，然后利用多线程以行主读的顺序，将子代左右节点的似然概率值的数据拷贝到共享内存中；声明共享内存变量 numChars 用于表示总位点数。

第 3 步 单个位点似然概率计算的循环次数为第一步传递的进化速率的数量值，然后提取不同速率下的 TiP 值以及子代左右节点的 CLP 值存储到寄存器变量中。准备好所有的变量后，计算当前转移概率矩阵下位点 A，C，G，T 的条件似然概率值，存储每一次计算后获得的 CLP 值时也需要将 CLP 数组的下标加上 numChars，计算完成当前转移概率矩阵下的似然概率后，通过循环重新载入其他进化速率条件下产生的转移概率矩阵，完成对应的状态似然概率的计算。

3 算法优化与实现

3.1 优化方法

在 CUDA 的并行计算架构下，使用并行计算函数时，需要为内核函数设定 dimGrid 和 dimBlock 两个核心参数集。线程的管理分为两层，第一层为 grid，第二层为 block，每一层都是三维的。Streaming Multiprocessor(SM)是该架构下的执行核心，所有需要执行的线程都在 SM 下进行处理，执行的最小单位是 warp，每个 warp 有 32 个线程。由于 SM 的数量远远小于线程数，因此在大规模的并行计算时，除执行线程外的线程以 warp 为单位等待执行，为充分利用硬件性能，设计分配线程数为最小执行单位的整数倍。当更多的 warp 被调用时，可以隐藏数据传输延迟，从而提升并行效率。

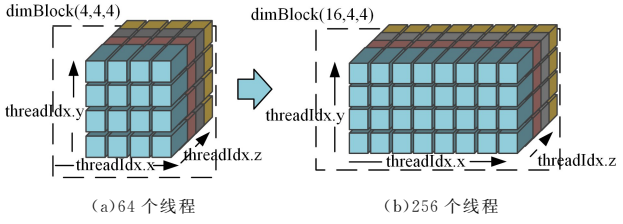
基于这个思路，根据位点间可变进化速率模型中分子状态似然概率的计算需要对应不同转移概率矩阵，本文方法将前期使用多线程对不同位点似然概率的并行计算，进一步分解为多位点间不同转移概率矩阵下的条件似然概率的计算。该策略在不改变单个线程计算传输比的基础上，通过增加线程数量，优化线程 warp 间并行重叠度来提高并行效率。此外，每个线程 warp 只计算同一种转移概率矩阵下的似然概率，避免了在使用共享内存时不同 warp 间的同步开销，可以进一步提升内核计算效率。

3.2 优化实现

在 tgMC^3 方法似然概率计算中，线程与位点一一对应，线程与位点的 CLP 计算的关系是一对多的；在指定的进化模型下，以 DNA 序列为例，每个位点需要计算 16 个 CLP(4 个 DNA 残基状态乘以 4 个伽马类别)，其似然概率估计函数分配的线程计算循环 4 次，在每一种转移概率矩阵下进行 4 个 CLP 计算。本文针对单个线程循环 4 次计算 16 个 CLP 以及单个线程计算量较大的问题进行优化，分解循环，每个线程似然计算 4 个 CLP 值，优化步骤分为 3 步。

第一步 如图 4 所示，将分配的线程数扩大 4 倍，执行单元 warp 由原来的 2 个扩大到 8 个，极大地提升了 GPU 线程的利用率；dimBlock 的参数设计十分精巧，由原来的 dimBlock(4,4,4)变为 dimBlock(16,4,4)；threadIdx.x 扩大 4 倍，线程块大小变为 16×4；threadIdx.z 保持不变，如图中所示，线程块均为 4 层；同时巧妙利用 threadIdx.z 来表示位点间不同进化速率的数量，为后面的不同数量参数的设定提供了可扩展空间。

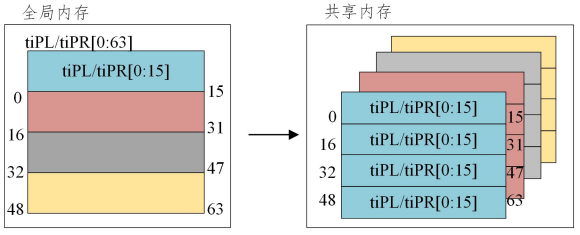
第二步 若按照 tgMC^3 方法中的数据分配模式，增加的 warp 线程数将不能获取到对应数据，从而不能进行位点的 CLP 计算。如图 5 所示，本文用 s_开头表示共享内存的数据，对原有 s_tiPL 和 s_tiPR 的共享内存空间从 16×4 扩大为 16×4×4，然后通过多线程进行数据的拷贝存储。根据 threadIdx.y 的数值，将原来的转移概率矩阵数据每份拷贝为 4 份，存储于共享内存中。如图 5 所示，全局内存中的 4 种颜色的子数组表示不同的转移概率矩阵，对应着 4 种不同颜色扩展在共享内存中。本文利用 threadIdx.z 控制不同转移概率矩阵的读取，使用 threadIdx.x 和 threadIdx.y 控制每个 half-warp 线程的操作，属于同一个 half-warp 的 16 个线程，根据线程的 threadIdx.z 值从全局内存读取对应转移概率矩阵。这样的设计可以让线程数与转移概率值一一对应，避免了数据读写冲突，解决了不同 warp 间在计算 CLP 之前需要进行的同步操作。



(a)64 个线程 (b)256 个线程

图 4 核函数线程数量分配变化

Fig. 4 Changes inallocation of number of threads in the core function

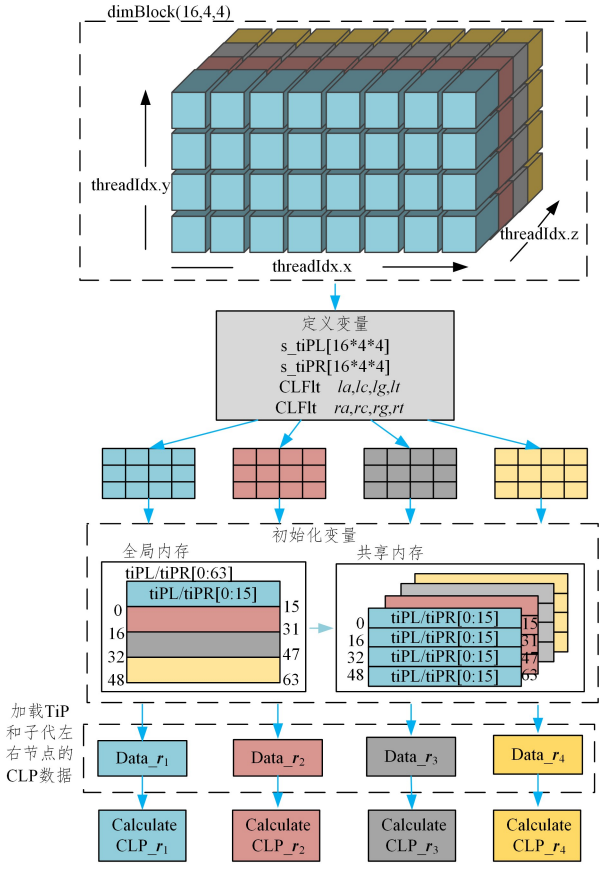


注：箭头表示 TiP 数据从全局内存中加载到共享内存中，同时对 TiP 进行 4 次复制

图 5 转移概率矩阵数据的拷贝

Fig. 5 Copy of transition probability matrix data

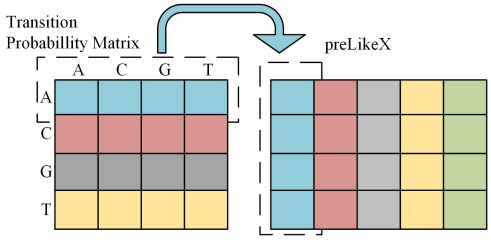
第三步 首先，每个 half-warp 线程从全局内存中 clL/clR 取出子代左右节点的条件似然概率值后，存入每个线程的寄存器中，并且与声明的左右子节点似然概率值的变量一一对应。每个 half-warp 线程从共享内存中 s_tiPL/s_tiPR 读取 16 个转移概率值放入 half-warp 的共享内存中，然后 half-warp 线程中的每一个线程取出子代左右节点的条件概率值与线程共享内存中的转移概率值进行似然概率计算得到线程对应位点的 CLP 值，并将该位点的结果迭代存入共享内存中。如图 6 所示，线程立方体的 4 种颜色表示 4 个线程块，每一种颜色的线程块有 64 个线程；为了绘图的美观性，将线程块的 threadIdx.x 绘制成 8 块，实际值为 16。$\boldsymbol{r}_1$，$\boldsymbol{r}_2$，$\boldsymbol{r}_3$，$\boldsymbol{r}_4$ 表示 4 种不同速率下的转移概率矩阵，线程块的颜色与转移概率矩阵的颜色一一对应。以蓝色线程块为例，将蓝色线程块中的 64 个线程分成 4 组 half-warp 线程块，这 4 组线程块从全局内存 tiPL/tiPR 读取相同的 16 个转移概率值即 $\boldsymbol{r}_1$ 矩阵存入共享内存 s_tiPL/s_tiPR 中。加载数据：首先 4 组 half-warp 线程从共享内存中读取出 64 个转移概率值，16 个为一组存入 4 组 half-warp 的共享内存中，再从全局内存 clL/clR 中提取子代左右节点的 CLP 值存入每个线程的寄存器中，且将数据分别赋值给 la/ra，lc/rc，lg/rg，lt/rt 变量。位点的似然计算：每个线程从寄存器中取出寄存器中的子代左右节点 CLP 变量，再从 half-warp 的共享内存中读取 TiP 值，然后分别计算内部节点在该速率下的 4 个 CLP，最后将线程计算的 CLP 结果迭代存入全局内存中。



注：$r1$，$r2$，$r3$，$r4$ 表示 4 种不同速率下的转移概率矩阵，每个线程块计算所有位点在对应速率下的似然概率

图 6　gpu_down_0 计算模型(电子版为彩图)

Fig. 6　gpu_down_0 calculation model



注：绿色是添加的常量，用于表示非 A,C,G,T 字符的转移概率

图 7　转移概率矩阵的转置与常量添加(电子版为彩图)

Fig. 7　Transposition and constant addition of transition probability matrix

此外，如图 7 所示，gpu_down_12 对应的节点的子代节点中 1 个为叶子节点，另 1 个为非叶内部节点。由于叶子节点的值是确定的，内部节点是不确定的，tgMC^3 方法中为方便在数组内存中提取转移概率，将转移概率矩阵进行转置，并将转置后的结果存入 preLike 矩阵。程序中使用 preLikeX 代替子代左右节点的转置后的转移概率矩阵，由于输入序列处理时填充了虚字符，增加了一列数据来处理非 A,C,G,T 字符的转移概率，因此将矩阵大小设定为 20×进化速率的值。将图中转移概率矩阵中蓝色的一行数据，存入 preLikeX 数组的第一列，再依次存入 preLikeX 数组中，直到 4 个字符的转移概率值全部完成后，最后一列存入常量为 1.0。

4　实验

4.1　实验环境

所有性能验证实验均在同一个 dell Inspiron 台式机上进行，该计算机的主要硬件设备有酷睿 i7-8700 CPU，显卡为 1 块 NVIDIA GeForce 1060，内存为 16 GB，硬盘 1T。操作系统版本为 Ubuntu 20.04，Host 端和 Device 端代码的编译工具及版本分别为 gcc(9.3.0)和 CUDA Tookit(11.4)。

4.2　实验数据

实验首先使用 4 个真实数据集：第 1 个数据集包含 37 个物种，每个物种的序列长度为 2 238；第 2 个数据集包含 111 个物种，每个物种的序列长度为 1 506；第 3 个数据集包含 234 个物种，每个物种的序列长度为 1 790；第 4 个数据集包含 288 个物种数，每个物种的序列长度为 3 386。此外，我们还使用了由 seq-gen v1.3.2x[20] 生成的 30 个模拟数据集。使用模拟数据是为了充分比较不同加速方法在不不同数据扩展条件下计算性能的变化。这些模拟数据集分为 2 组，第 1 组的数据集包含相同数量的物种，但序列长度不同，序列长度等于 $1\,000*i$，$i\in\{1,\cdots,15\}$。第 2 组数据集包含相同数量的位点数，但物种数量不同，物种数量等于 $4*j$，$j\in\{1,\cdots,15\}$。

4.3　评估方法

我们对 nMC^3(2.1.1 版)、tgMC^3(2.0 版)和本文方法的系统发育分析性能进行对比，需要注意的是，由于 nMC^3 方法中似然函数的调用次数与另外两个方法存在一定差别，我们不将总体分析时间作为性能的评价标准。为准确比较 3 个方法的加速性能，本文将单位时间内 CLP 的浮点运算次数作为评价标准。以本文指定的进化模型下的核苷酸序列为例，每个位点需要计算 16 个 CLP(4 个核苷酸分子状态乘以 4 个分子进化速率)，每个函数的总浮点运算次数＝(调用次数×位点数×16)/(调用时间×10^9)GFLOPS。我们通过 nvprof 工具统计每种似然函数调用次数和运行时间。

4.4　性能对比

表 1 列出了 3 种方法在真实数据集上的分析性能比较，通过表 1 分别计算出每个函数每秒的浮点运算次数(见表 2)。对 3 个函数的浮点运算次数进行分析，结果如图 8 所示。

表 1　nMC^3、tgMC^3、本文方法在 4 组实际数据上的主要函数运行时间和调用次数比较

Table 1　Comparison of running time and number of calls of main functions nMC^3, tgMC^3, and the proposed method on four realistic data sets

数据集		1(37×2238)		2(111×1506)		3(234×1790)		4(288×3386)	
方法	函数	时间/s	调用次数	时间/s	调用次数	时间/s	调用次数	时间/s	调用次数
nMC^3	down_3	0.64	181 257	1.39	416 870	1.81	505 700	2.60	721 296
	down_12	1.87	343 205	5.72	1 034 872	13.75	2 368 756	13.14	2 727 090
	down_0	3.34	527 389	5.46	855 992	13.70	1 972 821	13.87	1 977 480
tgMC^3	down_3	0.62	176 571	1.38	415 353	1.68	500 461	2.70	709 824
	down_12	1.75	356 044	4.58	991 078	10.43	2 171 251	17.28	2 809 681
	down_0	3.02	528 266	4.62	833 392	11.11	1 834 100	17.07	2 063 208
本文	down_3	0.51	176 571	1.17	415 353	1.41	500 461	2.22	709 824
	down_12	1.02	356 044	2.85	991 078	6.51	2 171 251	11.30	2 809 681
	down_0	1.70	528 266	2.73	833 392	6.86	1 834 100	12.30	2 063 208

表 2 nMC³、tgMC³、本文方法在 4 组数据上的主要函数的浮点运算性能对比

Table 2 Comparison of floating-point calculation performance of main functions nMC³, tgMC³, and the proposed method on four realistic data sets

数据集		1(37×2238)	2(111×1506)	3(234×1790)	4(288×3386)
方法	函数	GFLOPS	GFLOPS	GFLOPS	GFLOPS
nMC³	down_3	4.23	5.39	7.59	14.12
	down_12	2.73	3.25	4.68	10.55
	down_0	2.34	2.82	3.91	7.25
tgMC³	down_3	4.38	5.54	8.24	13.44
	down_12	3.13	3.99	5.75	8.32
	down_0	2.68	3.33	4.56	6.19
本文	down_3	5.31	6.53	9.82	16.39
	down_12	5.34	6.41	9.22	12.73
	down_0	4.77	5.63	7.39	8.59

图 8 显示，对于全部实际数据的系统发育分析中，本文方法对于似然计算的 3 个核心函数的加速性能全部超过了 nMC³ 和 tgMC³ 方法。在这 4 组实际数据中，本文方法的计算性能均超过了前期方法，获得了最高 2.04 倍的加速效果。

图 9(a)—图 9(c)是在模拟数据集 1 的条件下的结果图。对于 down_3 函数，本文方法单位时间内的浮点运算次数是 nMC³ 方法的 1.00～1.30 倍，是 tgMC³ 方法的 1.20～1.42 倍。对于 down_12 函数，本文方法单位时间内的浮点运算次数是 nMC³ 方法的 1.03～2.00 倍，是 tgMC³ 方法的 1.44～1.62 倍。对于 down_0 函数，本文方法单位时间内的浮点运算次数是 nMC³ 方法的 1.07～1.84 倍，是 tgMC³ 方法的 1.35～1.73 倍。综上，在这 15 组模拟数据中，本文方法的计算性能均超过了前期方法，获得了最高 2 倍的加速效果。

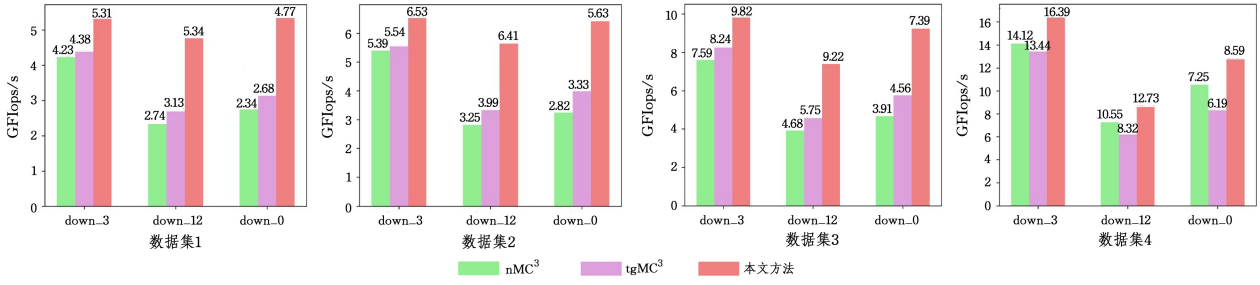


图 8 3 种方法在 4 组真实数据上调用似然计算函数获得的浮点运算性能对比

Fig. 8 Performance comparison of floating-point operations of likelihood estimation functions of three methods on four realistic data sets

图 9(d)—图 9(f)是在模拟数据集 2 的条件下的结果示意图，对于 down_3 函数，本文方法单位时间内的浮点运算次数是 nMC³ 方法的 1.00～1.24 倍，是 tgMC³ 方法的 1.19～1.50 倍。对于 down_12 函数，本文方法单位时间内的浮点运算次数是 nMC³ 方法的 1.04～1.82 倍，是 tgMC³ 方法的 1.50～1.67 倍。对于 down_0 函数，本文方法单位时间内的浮点运算次数是 nMC³ 方法的 1.19～1.66 倍，是 tgMC³ 方法的 1.44～1.65 倍。综上，在这 15 组模拟数据中，本文方法的计算性能均超过了前期方法，获得了最高 1.82 倍的加速效果。

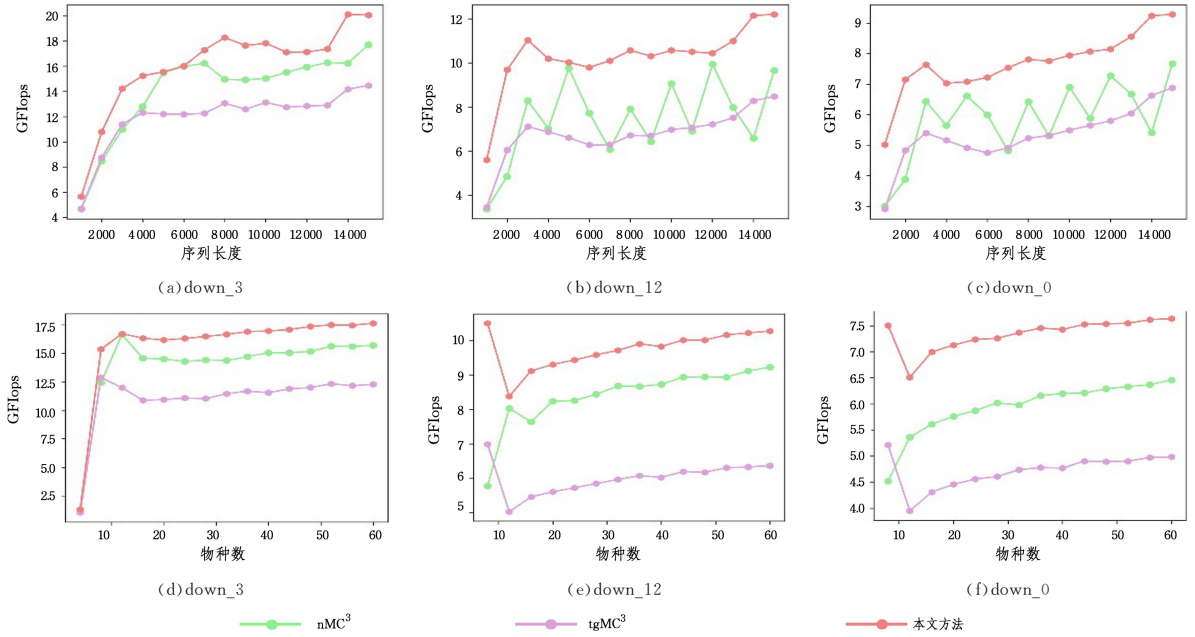


图 9 本文中的 3 种方法分别在固定类群数量和固定位点数量条件下的浮点运算性能对比

Fig. 9 Floating-point performance comparison of the three methods in this paper under the condition of fixed number of clusters and fixed number of sites

结束语 本文围绕 MrBayes 的高性能计算的并行优化，在 nMC³ 和 tgMC³ 方法的似然并行计算方法上进行优化获得了最高 2.04 倍的加速。本文方法将前期使用多线程对不同位点似然概率的并行计算，进一步分解为多位点间不同转移

概率矩阵下的条件似然概率的计算。该策略在不改变单个线程计算传输比的基础上,通过增加线程数量,优化了线程 warp 间并行重叠度,提高了并行效率。

下一步考虑将似然计算核心函数位点数由 64 个提升到 256 个位点,充分挖掘 GPU 多线程计算能力。进一步对似然概率估计任务级粒度进行优化,计算位点的似然概率由单块 GPU 拓展到 4 块 GPU 并行计算,单块 GPU 只需计算所有位点一种速率下的似然概率值,从而实现更快的系统发育分析。

参 考 文 献

[1] ARROWSMITH C,BOUNTRA C,FISH P,et al. Epigenetic protein families:a new frontier for drug discovery. Nature Reviews Drug Discovery,2012,11(5):384-400.

[2] RAJAPAKSA S,RASANJANA W,PERERA I,et al. GPU Accelerated Maximum Likelihood Analysis for Phylogenetic Inference[P]. Software and Computer Applications,2019.

[3] GUINDON S,GASCUEL O. A Simple,Fast,and Accurate Algorithm to Estimate Large Phylogenies by Maximum Likelihood [J]. Systematic Biology,2003,52(5):696-704.

[4] YANG Z,RANNALA B. Bayesian phylogenetic inference using DNA sequences:a markov chain monte carlo method[J]. Molecular Biology & Evolution,1997(7):717-724.

[5] HASTINGS W K. Monte Carlo Sampling Methods Using Markov Chains and Their Applications[J]. Biometrika,1970,57(1):97-107.

[6] GREEN P J. Reversible Jump Markov Chain Monte Carlo Computation andBayesian Model Determination [J]. Biometrika,1995,82(4):711-732.

[7] METROPOLIS N,ROSENBLUTH A W,ROSENBLUTH M N,et al. Equation of State by Fast Computing Machines[J]. Journal of Chemical Physics,1953,21(6):1087-1092.

[8] PRATAS F,TRANCOSO P,STAMATAKIS A,et al. Fine-grain parallelism using multi-core,Cell/BE,and GPU systems:Accelerating the phylogenetic likelihood function[C]// 2009 International Conference on Parallel Processing. IEEE,2009:9-17.

[9] PANG S A,STONES R J,REN M M,et al. GPU MrBayes V3.1:MrBayes on Graphics Processing Units for Protein Sequence Data.[J]. Molecular Biology and Evolution,2015,32(9):2496-2497.

[10] ZHAO M,REN Q,WANG Y,et al. A Three-Level Parallel Algorithm for MrBayes 3.2[C]// 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC). IEEE, 2017: 1246-1250.

[11] PRATAS F,TRANCOSO P,SOUSA L,et al. Fine-grain parallelism using multi-core,Cell/BE,and GPU Systems[J]. Parallel Computing,2012,38(8):365-390.

[12] ZHOU J F,LIU X G,STONES D S,et al. MrBayes on a Graphics Processing Unit[J]. Bioinformatics,2011,27(9):1-7.

[13] BAO J,XIA H J,ZHOU J F,et al. Efficient implementation of MrBayes on multi-GPU[J]. Molecular Biology and Evolution,2013,30(6):1471-1479.

[14] LING C,HAMADA T,BAI J N,et al. MrBayes tgMC^3:A Tight GPU Implementation of MrBayes[J]. PLOS ONE,2013,8(4):1-9.

[15] LING C,HAMADA T,GAO J Y,et al. MrBayes tgMC^3++:A High Performance and Resource-Efficient GPU-Oriented Phylogenetic Analysis Method[J]. IEEE/ACM Transactions on Computational Biology and Bioinformatics,2016,13(5):1-9.

[16] KUAN L,PRATAS F,SOUSA L,et al. MrBayes sMC^3:Accelerating Bayesian inference of phylogenetic trees[J]. The International Journal of High Performance Computing Applications,2018,32(2):754-755.

[17] AYRES D L,DARLING A,ZWICKL D J,et al. BEAGLE:An Application Programming Interface and High-Performance Computing Library for Statistical Phylogenetics[J]. Systematic Biology,2012,61(1):170-173.

[18] AYRES DANIEL L,CUMMINGS MICHAEL P,BAELE G,et al. BEAGLE 3:Improved Performance,Scaling,and Usability for a High-Performance Computing Library for Statistical Phylogenetics. [J]. Systematic Biology,2019,68(6):1052-1061.

[19] YANG Z. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites:approximate methods[J]. Journal of molecular evolution,1994,39(3):306-314.

[20] ANDREW R,NICHOLAS C. Grass. Seq-Gen:an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees[J]. Bioinformatics,1997,13(3):235-238.

HUANG Jia-wei, born in 1996, postgraduate. His main research interests include high-performance GPU computing in bioinformatics and so on.

LING Cheng, born in 1987, Ph.D, associate professor. His main research interests include high-performance GPU computing on computational biology and bioinformatics.