计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2018.20160873
55(2):409-417,2018

面向 GPU 的单颗粒冷冻电镜软件 RELION 并行与优化

苏华友　温　文　李东升
(国防科技大学并行与分布处理重点实验室　长沙　410073)
(huayousu@163.com)

Optimization and Parallelization Single Particle Cryo-EM Software RELION with GPU

Su Huayou, Wen Wen, and Li Dongsheng
(*Science and Technology on Parallel and Distributed Processing Laboratory*, *National University of Defense Technology*, *Changsha* 410073)

Abstract　Single particle cryo-electron microscopy (cryo-EM) is one of the most important methods of macromolecular structure. RELION (regularized likelihood optimization) is an open-source computer program for the refinement of macromolecular structures by single-particle analysis of cryo-EM data. Due to its easy usage and high quality results, RELION has attracted a lot of attentions from researchers. However, the computation requirement of this program is too huge to solve some large molecular structures with CPU, which harpers the popularization of RELION. In this paper, we characterize the algorithm of RELION and parallelize it with GPU. Firstly, the mathematical theory, computer patterns and performance bottlenecks of RELION are analyzed comprehensively. Then, we optimize the program targeting at fine-grained many-core architecture processor, such as GPU. We propose an efficient multi-level parallel model to utilize the powerful computation capacity of many-core processor. In order to achieve high performance, we reconstruct the data structure for GPU continues memory access. To avoid the limitation of GPU memory size, we implement an adaptive framework. The experimental results show that the proposed GPU based algorithm can achieve good performance. When compared with the CPU implementation, the speedup ratio of the application is more than 36 times, while the speedup ratio of compute-intensive algorithm is about 75X. Moreover, the testing results on multi GPUs show that the GPU based implementation has good scalability.

Key words　cryo-electron microscopy (cryo-EM); regularized likelihood optimization (RELION); graphic processing units (GPU); parallel computing; high performance computing

摘　要　单颗粒冷冻电镜是结构生物学研究的重要手段之一,基于贝叶斯理论的冷冻电镜3维图像数据处理软件 RELION(regularized likelihood optimization)具有很好的性能和易用性,受到广泛关注.然而其计算需求极大,限制了 RELION 的应用.针对 RELION 算法的特点,研究了基于 GPU 的并行优化问题.首先全面分析了 RELION 的原理、RELION 程序的算法结构及性能瓶颈;在此基础上,针对 GPU 细粒度体系结构对程序进行优化设计,提出了基于 GPU 的多级并型模型.为了获得良好的性能,对

收稿日期:2016-11-21;**修回日期**:2017-07-04
基金项目:国家自然科学基金项目(21502509);霍英东青年教师基金项目(141066)
This work was supported by the National Natural Science Foundation of China (21502509) and the Fok Ying-Tong Education Foundation for Young Teachers in the Higher Education Institutions of China (141066).

RELION 的数据结构进行重组. 为了避免 GPU 存储空间不足的问题,设计了自适应并行框架. 实验结果表明:基于 GPU 的 RELION 实现可以获得良好的性能,相比于单 CPU,整个应用的加速比超过 36 倍,计算密集型算法的加速比达到 75 倍以上. 在多 GPU 上的测试结果表明基于 GPU 的 RELION 具有很好的可扩展性.

关键词 冷冻电镜;正则化似然优化;图形图像处理器;并行计算;高性能计算

中图法分类号 TP311.56; TP338.6

单颗粒冷冻电镜是结构生物学上研究分子结构的重要手段[1],采用冷冻电镜技术获取蛋白质分子的显微镜照片,通过单颗粒重构的方法获得蛋白质的 3 维结构. 近年来,随着科技的发展,例如样品制备技术、新的电子显微镜、新的重构算法的提出以及计算机计算能力的提升,冷冻电镜重构技术可以获得接近原子分辨率的 3 维结构,并因此获得结构生物学研究者的青睐.

冷冻电镜 3 维重构的数学原理是中央界面定理[2]:3 维实空间的物体的某个方向上的 2 维投影的傅里叶变换等于该物体在 3 维傅里叶空间的一个中央截面,该中央截面经过原点且垂直于投影方向. 单幅颗粒的电镜图片对应于该颗粒某个方向上的中央截面,得到足够多的、均一的电镜图片可求得颗粒在 3 维傅里叶空间的信息,对 3 维傅里叶信号进行逆变换可以获得颗粒的 3 维空间结构. 为了降低生物样品遭受电子辐射的损伤,使用的电子剂量较小,生成的电镜图片信噪比和对比度都不高. 为了提高 3 维结构的分辨率,往往需要对大量图片进行平均操作,同时重构方法需要将原始图片与参考模型的投影图像进行对比,整个重构过程往往要持续多次迭代,对计算的需求较高.

目前已经有很多针对单颗粒冷冻电镜图像处理的软件,包括 XMIPP[3], EMAN, EMAN2[4], FREALIGN[5], RELION[6] 等. 由于重构对计算的要求巨大,即使采用多 CPU 并行的方式,这些软件仍然难以满足需求. 相比较 CPU,图形图像处理器(graphic processing unit, GPU)有更强大的浮点计算能力和更高的存储带宽. 于此同时,GPU 已经在高性能科学计算领域获得广泛的应用. 目前,已经有研究人员对基于 GPU 的冷冻电镜图像处理技术进行了研究和探索. 例如:Li 等人[7]研究了 EMAN 在 CPU/GPU 异构系统中的并行化;Li 等人[8]研究了用 GPU 来加速 FREALIGN;Cianfrocco 等人[9]通过云计算来处理冷冻电镜数据. RELION 是由 MRC 的 Scheres 在 2012 年发布的针对单颗粒冷冻电镜图片进行处理的一个框架. 由于具有很好的效果及易用性,该软件自发布之后便引起了学术界和工业界的广泛关注. 研究人员通过 RELION 已获得一系列高分辨率的分子结构,获得结构生物学的广泛关注. 相比较传统的重构软件,RELION 改进了重构算法来提高计算速度和结果的精度[10],如:采用贝叶斯理论和最大后验估计来重构 3 维模型、采用不同采样粒度来加速颗粒取向的搜索、采用 2 批独立数据的模型来估计结果的精度防止过拟合等. 目前,RELION 已经成为冷冻电镜研究领域的主流软件之一,最近几年发现的分辨率达到 4Å 以上的分子结构,几乎都是采用 RELION 对电镜图像进行处理得到的. RELION 冷冻电镜图像处理算法对计算需求巨大,对于 10^5 数量级的图片,RELION 在 256 个主流(Intel sandy bridge E5 处理器)CPU 核上的运行时间达到 10 h 以上. 如果能将 GPU 的强大计算能力应用到 RELION 上,将大幅提高 RELION 的处理速度,极大提高科研人员的工作效率.

本文针对单颗粒冷冻电镜重构软件 RELION,开展基于 GPU 并行化研究. 首先全面分析了 RELION 的原理、RELION 程序的算法结构及性能瓶颈,然后针对 GPU 细粒度体系结构对程序进行优化设计,提出了基于 GPU 的 3 级并行 RELION 实现方案. 为了获得良好的性能,本文对程序结构进行了重构,对数据结构进行了重组以及使用高性能计算库. 实验结果表明,基于 GPU 的 RELION 实现可以获得 36 倍的加速比,并且具有很好的可扩展性.

1 RELION 的算法原理和特征分析

1.1 数学模型

RELION 是一个基于贝叶斯方法的冷冻电镜图像处理软件,其基本数学模型如傅里叶空间中的式(1)所示:

$$X_{ij} = CTF_{ij} \sum_{l=1}^{L} \boldsymbol{P}_{jl}^{\phi} V_{kl} + N_{ij}, \qquad (1)$$

其中,X_{ij} 表示第 i 幅图片在 2 维傅里叶空间上的第 j 个分量;CTF_{ij} 表示对应于第 i 幅图片的衬度传递函数的第 j 个分量;V_{kl} 表示第 k 类 3 维模型在 3 维傅里叶空间上的第 l 个分量,分类总数 K 通常假定为已知的数目. 所有分量 V_{kl} 满足均值为 0、方差为 τ_{kl}^2 的相互独立的高斯分布;$\boldsymbol{P}_{jl}^{\phi}$ 表示投影矩阵,$\sum_{l=1}^{L}\boldsymbol{P}_{jl}^{\phi}V_{kl}$ 表示将第 k 类 3 维模型往方向 ϕ 做投影的第 j 个分量,ϕ 包含 3 维旋转参数和 2 维平移参数. 类似的,$\sum_{l=1}^{L}\boldsymbol{P}_{jl}^{\phi\mathrm{T}}X_{ij}$ 表示将图片从 2 维傅里叶空间反向投影到 3 维傅里叶空间;N_{ij} 表示噪声,假设满足均值为 0、方差为 σ_{ij}^2 的相互独立的高斯分布.

冷冻电镜图像处理的目标是找到在先验信息 Y 下观察到样本 X 的最大概率模型参数 Θ(包括 V_{kl},τ_{kl}^2,σ_{ij}^2). 在贝叶斯理论中,这种后验分布包括 2 个部分,如式(2)所示:

$$P(\Theta|X,Y)\propto P(X|\Theta,Y)P(\Theta|Y), \tag{2}$$

$P(\Theta|X,Y)$ 表示给定模型参数 Θ 和先验信息 Y 下样本的概率分布,$P(\Theta|Y)$ 表示在先验信息 Y 下的模型参数 Θ 的概率分布. 式(2)可以使采用最大后验估计方法求解.

在 RELION 中,先验信息由冷冻电镜重构模型提供,而后验信息则由观察到图像样本提供. RELION 软件使用 EM(expectation maxi-mization)算法对式(1)进行求解,具体为式(3)~(7)迭代公式:

$$V_{kl}^{(n+1)}=\frac{\sum_{i=1}^{N}\int_{\phi}\Gamma_{ik\phi}^{(n)}\sum_{j=i}^{J}\boldsymbol{P}_{lj}^{\phi}\frac{CTF_{ij}X_{ij}}{\sigma_{ij}^{2(n)}}\mathrm{d}\phi}{\sum_{i=1}^{N}\int_{\phi}\Gamma_{ik\phi}^{(n)}\sum_{j=i}^{J}\boldsymbol{P}_{lj}^{\phi}\frac{CTF_{ij}X_{ij}}{\sigma_{ij}^{2(n)}}\mathrm{d}\phi+\frac{1}{\tau_{kl}^{2(n)}}}, \tag{3}$$

$$\sigma_{ij}^{2(n+1)}=\frac{1}{2}\sum_{k=1}^{K}\int_{\phi}\Gamma_{ik\phi}^{(n)}\left|X_{ij}-CTF_{ij}\sum_{l=1}^{L}\boldsymbol{P}_{jl}^{\phi}V_{kl}^{(n)}\right|^2\mathrm{d}\phi, \tag{4}$$

$$\tau_{kl}^{2(n+1)}=\frac{1}{2}|V_{kl}^{(n+1)}|^2, \tag{5}$$

$$\Gamma_{ik\phi}^{(n)}=\left\{P(X_i\mid k,\phi,\Theta^{(n)},y)P(k,\phi\mid\Theta^{(n)},y)\right\}\Big/\left\{\sum_{k'=1}^{K}\int_{\phi'}P(X_i\mid k',\phi',\Theta^{(n)},y)P(k',\phi'\mid\Theta^{(n)},y)\mathrm{d}\phi'\right\}, \tag{6}$$

$$P(X_i\mid k,\phi,\Theta^{(n)},y)=\prod_{j=1}^{J}\frac{1}{2\pi\sigma_{ij}^{2(n)}}\exp\left(\frac{\left|X_{ij}-CTF_{ij}\sum_{l=1}^{L}P_{jl}^{f}V_{kl}^{(n)}\right|^2}{-2\sigma_{ij}^{2(n)}}\right), \tag{7}$$

这里 $V_{kl}^{(n+1)}$ 表示第 $n+1$ 次迭代的 3 维模型,$\sigma_{ij}^{2(n+1)}$ 表示第 $n+1$ 迭代的噪声功率,$\tau_{kl}^{2(n+1)}$ 表示第 $n+1$ 次迭代的模型的信号功率,$\Gamma_{ik\phi}^{(n)}$ 表示第 i 幅图片对应第 k 类模型方向为 ϕ 的后验概率. 迭代由一个估计的初始模型 V_{kl} 开始,模型的类别数目 K 通过人为指定,$\sigma_{ij}^{2(n+1)}$ 和 $\tau_{kl}^{2(n+1)}$ 的初始值可通过初始模型 V_{kl} 和电镜样本图片 X_{ij} 计算得到. 在每次迭代中,都要对所有方向 ϕ 和可能的类别 k 做计算得到后验概率 $\Gamma_{ik\phi}^{(n)}$,主要的计算需求在于求解式(7),即对所有采样的图片在所有可能的方向上进行均方差的计算.

1.2 RELION 程序特征分析

RELION 的主要功能是通过基于贝叶斯的 EM 算法重构分子的 3 维结构,算法 1 给出了 RELION 的整个程序流程.

算法 1. RELION 程序的 EM 迭代算法.

输入:冷冻电镜颗粒图片(X);

输出:冷冻电镜重构模型(V).

① for $iter<nr\_iter$

/* 初始化 */

② $expectationsetup()$;

/* 计算每个颗粒在模型不同方向的权重 */

③ $expectation()$;

④ for each pacticle image X_i

/* 计算每幅图对应的傅里叶形式以及 CTF 矩阵 */

⑤ $getForuierTransformsAndCTFs()$;

⑥ for $ipass<1$

/* 计算式(7)的指数部分 */

⑦ $getAllSquredDifferences()$;

/* 计算式(6) */

⑧ $convertAllDiffToWeights()$;

⑨ end for

/* 计算式(4)和式(5) */

⑩ $storeWeightedSums()$;

⑪ end for

/* 冷冻电镜模型重构,计算式(3) */

⑫ $maximization()$;

⑬ $reconstruct()$.

⑭ end for

EM 算法是一个迭代逐步求精的过程,整个过程通常包括多次迭代,迭代次数由 nr_iter 制定. 在每次迭代中,EM 算法分为 $expectation$ 和 $maximization$ 2 个步骤,简称 E 步和 M 步. 在 E 步中,需要计算所有

颗粒图像在3维模型各个投影方向的概率分布和期望,在M步中利用E步计算得到的各颗粒图片在不同上方上的投影概率来更新3维模型.在RELION中,E步主要包含以下4个函数:*getFourierTransformAndCTFs*, *getAllSquaredDifferences*, *convertAllDiffToWeights*, *storeWeightedSums*.下文为了方便描述,本文将这4个部分简称为*getFourier*, *getAll*, *convertAll*, *storeWeight*.M步主要包括函数*reconstruct*.

我们在2个计算节点上对RELION程序进行热点分析.其中,每个节点含有2个Intel Xeon E5-2620 v3 CPU,其主频为2.40 GHz.每个节点运行12个MPI进程.我们同时对MPI+Pthread的模式进行了测试,其统计结果与基于MPI运行方式得到的结果类似.RELION运行的功能模式为3D-classification和auto-refine.

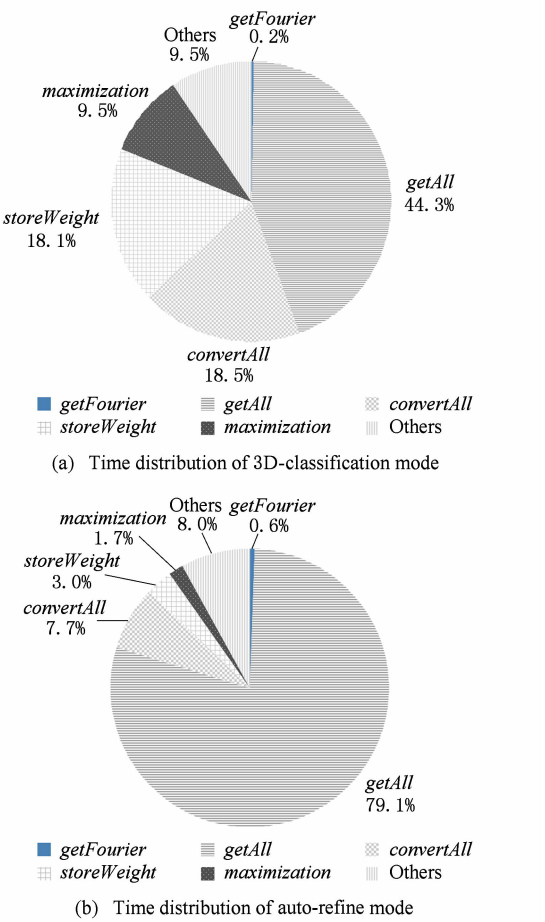


Fig. 1 RELION execution time distribution

图1 RELION执行时间分布

图1给出了RELION在运行在3D-classification模式和auto-refine模式时的程序执行时间分布.RELION的3D-classification模式和auto-refine模式都实现了3维重构,其中3D-classification模式用来确定可能的模型种类,auto-refine模式用来计算最终的重构模型.这2种模式的程序流程类似,参数有所不同,auto-refine模式求得的模型具有更高的分辨率,通常来讲需要更长的运行时间.从测试结果可以看到其中*getAll*和*convertAll*占到大部分的时间,对应公式中需要密集计算的部分,Others占到8%~9%,对应读写文件、MPI通信开销和其余函数.值得注意的是为了降低计算需求,RELION采用了一种自适应的方式实现EM算法,该实现方式通过2次采样来确定2维图像在3维模型中的投影方向,第1次($ipass=0$)是粗粒度采样,第2次($ipass=1$)是细粒度采样,采样方向为第1次粗粒度采样阶段计算得到的显著方向,即式(6)计算得到的结果中大于阈值的方向,如算法1中第⑦行和第⑧所示,对应式(6)和式(7)的求解.

通过对程序算法和热点代码分析,我们将RELION的计算模式分为为以下3类:计算密集型、控制密集型和全局规约操作.在第1次粗粒度采样时,对于每一张2维图像,程序将会计算该图像在所有可能的采样方向上的投影概率,即式(7)中的L_2范数.其基本计算过程是每一幅图和参考模型在可能方向上的2维投影图像以像素为单位进行点对点计算.在这一阶段,所有的图像以及所有的方向都是独立的,表现出极高的并行性,本文将该类计算归为计算密集型.式(7)当中L_2范数的计算量可以用式(8)表示:

$$nr\_particles\times nr\_orients\times nr\_trans\times image\_size\times ops, \tag{8}$$

其中$nr_particles$为颗粒图像数目,$nr_orients$为投影方向的数目,nr_trans为平移位置的数目,$image_size$为图像大小,ops为每个像素点对应的操作次数.对于高分辨率的3维结构的重构,通常需要对10万到百万量级的图像,而每一张图可能的投影方向多达上千,计算需求达到10^{15}量级.

式(3)和式(4)针对*convertAll*阶段计算得到的有效方向进行处理,每一次计算之前都需要判断该方向是否是有效的.类似地,在细粒度采样阶段时,式(6)和式(7)的计算也需要根据当前方向是否是有效的方向来决定是否执行.相对于全部的采样方向,实际上每一幅图的有效方向数是很少的,本文将这类计算归结为稀疏索引计算.

在对式(3)和式(4)进行求解时,每一个模型分量的值是所有图像在有效方向上反向投影的加权

和,这涉及到所有颗粒图像在某一方向的全局规约.由于存在多个图像需要在同一投向上进行投影,为了保证正确性,并行计算时需要引入原子操作来实现全局规约.

2 针对 RELION 的并行优化方案

本节介绍基于 GPU 的 RELION 并行实现,包括总体并行模型设计、存储优化以及高性能 GPU 库的使用.

2.1 多级并行模型

单颗粒冷冻电镜软件 RELION 的并行性主要表现在以下 2 个层次:不同的图像在不同投影方向上的计算过程可以同时处理;同一幅图内部,各个像素点之间的计算也可以并行.针对这一特征,本文设计了一个面向多 GPU 的多级并行模型,如图 2 所示,该模型基于MPI+CUDA 实现,每个 MPI 进程控制 1 个 GPU,并将其主要的工作量提交给 GPU 执行.首先,所有的图片以组为单位形成 1 个任务池,主进程每次从任务池中选取 1 组颗粒图片发送到空闲的 GPU,多 GPU 之间的并行实现了多组图片的同时执行,如图 2 的最上层所示.其次,在同一个 GPU 内部,组内不同图片在不同方向上的计算可以同时执行,每一个图片在 1 个方向上的计算交由 1 个线程块处理,如图 2 的中间层所示.最后,在 1 个线程块内部,不同的线程可以并行处理不同的像素.通过 3 层并行,可以充分发挥 GPU 的计算能力.而以任务池的方式对整个应用进行组织,可以避免不同处理能力的 GPU 之间可能面临的负载不均衡问题.该模型主要应用于函数 *getAll* 和函数 *storeWeight* 中 L_2 范数的计算.

在函数 *storeWeight* 中,涉及到大量的规约操作,本文针对不同情况分别使用共享存储器或者原子操作来完成.当使用共享存储器来完成规约操作时,采取减少线程块内线程数目的方式以及单个线程计算多个像素的方式来提高共享存储器的重用率.当使用原子操作完成规约操作时,减少线程块的数目,利用线程切换来隐藏访存的开销.

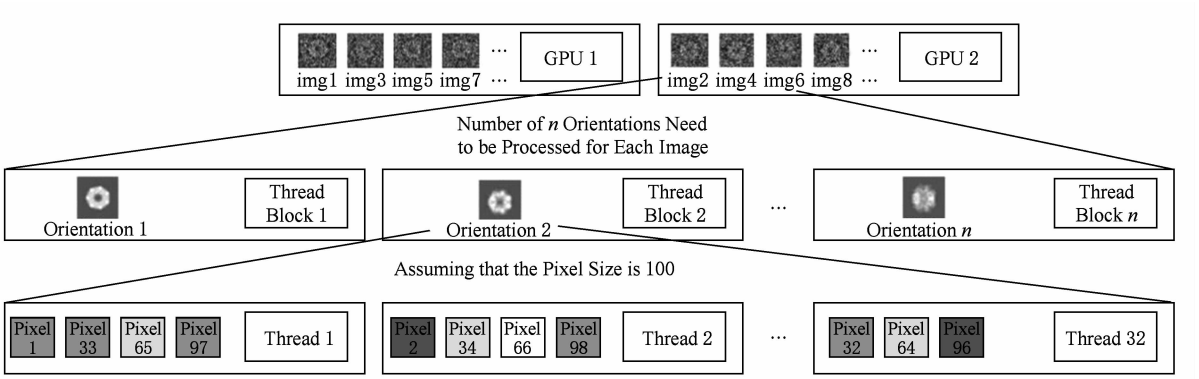


Fig. 2 Multi-level parallel model

图 2 多级并行模型

2.2 数据结构重组

由于 PCI-E 的传输速率远低于 GPU 存储器的访问速率,PCI-E 传输数据还有启动开销,为减少数据在主机内存和 GPU 之间的传输,需要将尽可能多的数据存放在 GPU 的存储器上.即使有的数据计算量不大或者具有许多串行的特性,考虑到 GPU 硬件的性能以及 RELION 程序数据前后的关联性,存储在 GPU 上由 GPU 完成计算会比存储在内存由 CPU 计算更高效.

RELION 中的变量一般采用多维数组对象作为基本数据结构存储不同的属性信息,多维数组采用的是 AOS(array of structures)的方式.AOS 方式在访问属性信息时为跨步访存,cache 命中率不高,造成访存延迟,这种存储方式并不适合 GPU 存储器连续访问的特性.为了更好地利用存储带宽和缓存,需要将图像的属性信息重新组织为利于使用 GPU 连续访问的模式,例如 SOA(structure of array)

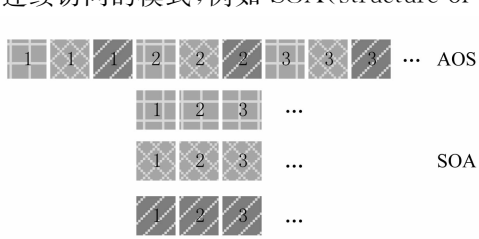


Fig. 3 Data structure reconstruction: From AOS to SOA

图 3 数据结构重组:从 AOS 到 SOA

方式,将同一属性的信息连续存储,之后将数据拷贝到 GPU 端进行计算,如图 3 所示. 虽然属性信息的重组需要在 CPU 端完成,会带来额外的时间开销,但是由于数据在 GPU 端的密集计算,重组的收益还是非常显著的.

2.3 自适应的任务分配

为了充分利用 GPU 的计算能力,我们总是希望尽可能多地往 GPU 加载任务,使得 GPU 能够满负荷的运作. 对于本文而言,在式(7)的计算过程中,我们希望能够同时处理所有图片在所有方向上的计算,然而这样很可能会造成内存不够的问题. 为了更好地利用 GPU 的计算能力,本文利用 MPI 来调度 GPU. 按照 RELION 的任务池的分配方式,将固定数目的颗粒图像分配给一个 MPI 进程,然后由该 MPI 进程启动对应的 GPU 进行计算,计算完成后再从任务池获取新的颗粒图像组,使得全部 GPU 达到负载平衡. 由于投影的方向数目很多,尤其是在细粒度采样阶段,可能在某些迭代步中 GPU 无法存储所有方向的数据. 因此,本文设计了一个自适应的任务分配方式. 首先根据需要计算的图片数量、图片大小以及可能的方向数量预测所需的存储空间;然后根据 GPU 当前可用存储空间对可以并行的方向数进行设置. 如果能够满足所有的投影方向对存储的需求,则同时处理所有的方向;否则将当前能够处理的方向减半,然后再一次进行比较,直到 GPU 的存储能够满足当前方向数目对内存空间的需求为止.

2.4 CUDA 高性能计算库的使用

CUDA 为开发者提供了一系列库函数,这些库函数对某些计算进行了针对性的优化,具有很好的性能和稳定性,同时也降低程序的开发难度. 本文使用 CUDA 提供的 cufft 库对 2 维图像和 3 维模型在实数和复数空间的傅里变换进行处理,主要应用在函数 *setFourier*、*getFourier* 以及 *reconstruct* 中. 在 RELION 的原始实现中,每次都对正、反 2 种傅里叶变换分配存储空间. 在 GPU 实现中,由于 3 维傅里叶变换所需的空间巨大,本文只根据当前需要进行的傅里叶变换类型分配对应的存储空间.

在函数 *convertAll* 中,涉及到显著方向阈值的计算,RELION 采用一系列排序、逐项求和、条件判断来完成,如算法 2 所示. 该过程是一个完全串行的实现,并不适合 GPU 执行. 然而如果这部分不能够有效的并行,需要将所有的权值都拷贝回 CPU 进行计算,不仅计算速度慢,而且数据的传输开销也十分昂贵.

算法 2. 计算每个颗粒权重阈值算法.

输入:电镜图片在不同方向的权重 W、所有权重因子总和 $total\_sum$、权重比例 α;

中间结果:非零权重 $sort\_W$;

输出:权重阈值 sig_weight.

① $i=np=0$

/ * 挑选非零权重 * /

② for ($;i<len(W);i++$)

③ if $W[i]>0$

④ $sort\_W[np++]=W[i]$;

⑤ end if

⑥ end for

/ * 对非零权重进行排序 * /

⑦ $sort$ ($sort\_W$)

/ * 计算有效方向的权重阈值 sig_weight * /

⑧ for ($i=0;i<len(sort\_W);i++$)

⑨ $sig\_weight=sort\_W[i]$;

⑩ $part\_sum+=sort\_W[i]$;

⑪ if ($part\_sum>total\_sum\times\alpha$)

⑫ break.

⑬ end if

⑭ end for

为了利用 GPU 并行实现算法 2 中描述的计算过程,本文采用 CUDA 的 thrust 库来完成一系列的操作,主要涉及以下原语:$thrust::sort$, $thrust::exclusive\_scan$, $thrust::count\_if$ 和 $thrust::copy\_if$. 完整的过程如算法 3 所示.

算法 3. 基于 GPU 并行计算颗粒权重阈值算法.

输入:电镜图片在不同方向的权重 W、所有权重因子总和 $total\_sum$、权重比例 α;

中间结果:非零权重 $sort\_W$;

输出:权重阈值 sig_weight.

/ * 挑选非零权重 * /

① $thrust::copy\_if(W, sort\_W, none\_zero())$;

② $thrust::sort(sort\_W)$;

/ * 对排序后的非零权重求前缀和 * /

③ $thrust::inclusive\_scan(sort\_W)$;

/ * 求阈值下标 * /

④ $n=thrust::count\_if(sort\_W, greater(total\_sum\times\alpha))$;

⑤ $sig\_weight=sort\_W[n]-sort\_W[n-1]$.

3 实验结果与分析

本节对基于 GPU 的单颗粒冷冻电镜图像处理软件的正确性和加速比进行评测. 实验测试平台由 2 个计算节点组成,每个计算节点包含 4 个 NVIDIA Kepler K40m GPU 和 2 个 Intel Xeon E5-2620 v3 CPU 组成. 其中,每个 K40 m 含有 15 个 SMX,每个 SMX 有 192 个流处理器,频率为 0.75 GHz,双精度浮点计算性能为 1.43TFLOPS,全局存储器的带宽为 288 GB/s. CPU 型号为 Intel Xeon E5-2620 v3 是 1 款 6 核处理器,处理器的频率为 2.4 GHz,单个 CPU(6 核)的浮点计算性能为 115.2GFLOPS. 本文采用 CUDA 7.5 工具包和 thrust[11]库进行 GPU 程序开发. 测试数据集包含 2 个,分别为 betagal[12]和 TRPV1[13]. 其中 betagal 包含 8 915 个颗粒图片,分辨率为 100×100;TRPV1 包含 35 645 幅颗粒图像,其分辨率为 256×256. 我们对这 2 个数据集在 3D-classification 和 auto-refine 两种功能模式下进行了测试.

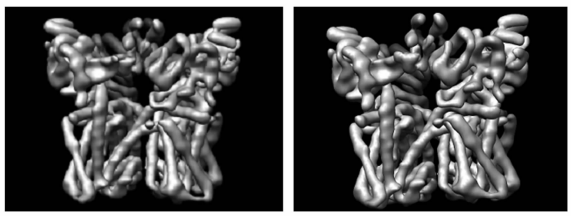
3.1 正确性验证

首先我们对程序的正确性进行了验证,表 1 给出了 CPU 和 GPU 两种实现在不同数据集上测试得到的模型分辨率,从表 1 中可以看出,GPU 的执行结果与 CPU 实现的结果基本一致. 为了进一步进行比较,我们对 auto-refine 之后的 TRPV1 数据进行渲染,得到图 4 所示的结果,从图 4 可以看出,本文的 GPU 实现也可以获得与原 CPU 实现一致的结果.

Table 1 The Generated Models' Resolutions with CPU or GPU RELION

表 1 CPU 和 GPU 程序在不同数据集上的模型分辨率 Å

Functions	Data Set	CPU	GPU
3D-classification	betagal	19.666 66	19.666 67
	TRPV1	7.836 923	7.836 923
auto-refine	betagal	7.695 652	7.695 652
	TRPV1	5.796 227	5.796 227



(a) Model on CPU (b) Model on GPU

Fig. 4 Comparison of model result

图 4 模型的对比

3.2 加速比分析

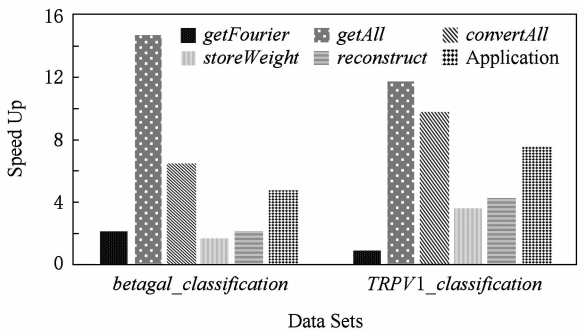
我们对基于 GPU 的单颗粒冷冻电镜图像软件的性能进行了测试. 表 2 给出了使用 CPU 和 GPU 程序在数据集 TRPV1 上进行 3D-classification 和 auto-refine 两种功能测试的时间统计结果. 从表 2 中可以看出,使用 8 个 GPU 可以获得比 24 个主流 CPU 快 10 倍左右的性能加速. 同时,经过我们对程序结构的改造和数据结构优化,程序中额外的时间开销也减少了,主要来自于中节结果的利用和访存量的降低. 图 5 给出 8 个 K40m GPU 相对 24 个 CPU 核的加速情况,其中 Application 表示整个应用的加速情况,其他各部分分别表示对应步骤的加速情况. 之所以使用 24 个 CPU 核,是因为单核 CPU 的执行时间太长. 从图 5 中可以看出,对于 auto-refine 过程而言,8 个 GPU 相对于 24CPU 的加速比超过了 12,归一化之后的加速比达到 36 以上. 而 3D-classification 的加速比为 5~7 倍,归一化之后的加速比为 15~20 倍. 3D-classification 加速比低于 auto-refine 的原因在于 3D-classification 过程中 getAll 的执行时间比重比 auto-refine 低. 而 auto-refine 过程中 *getAll* 部分的加速比达到了 25 以上,换算成单 GPU 对单 CPU,加速比超过了 75. 在所有的测试中,*getAll* 的加速比都是最高的,这也反映出该部分计算密集的特点. 在所有各部分性能中,*getFourier* 的加速效果最差,加速比接近 1,这是因为该部分是每一组图片计算的初始阶段,涉及很多的数据准备工作. 同时,该阶段的并行度比较低,

Table 2 The Execution Time Distribution of Different Configurations on Data Set TRPV1

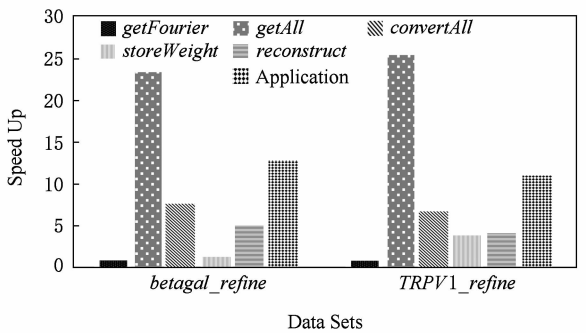
表 2 不同配置模式下 CPU 和 GPU 程序处理 TRPV1 数据集的执行时间分布 min

Functions	Configurations	Application	Expectation	Maximization	Others
3D-classification	24 CPUs	566.7	461	55.4	50.3
	8 GPUs	75	41	13.3	20.8
auto-refine	24 CPUs	1 008.78	907.2	20.2	80.7
	8 GPUs	92	76.5	9.4	6.2

只有一组图像中包含的图片数量,通常情况下为 8. 也就是说在 15 个 SM 上只执行 8 个线程块,因此其并行效果并不好. 需要注意的是,将这部分计算任务放在在 GPU 上执行,可以直接将结果保存在 GPU 中给 *getAll* 部分使用. 函数 *convertAll* 主要利用 thrust 库进行优化,加速比达到 5～10. 函数 *storeWeight* 的加速比同样不理想,仅仅为 2～5 倍,原因在于这部分包含大量的全局规约操作,GPU 优化中使用了一定数量的原子操作来完成. 函数 *reconstruct* 利用了 cufft 库来加速,加速比达到 5～10 倍.



(a) Speed up for 3D-classification mode



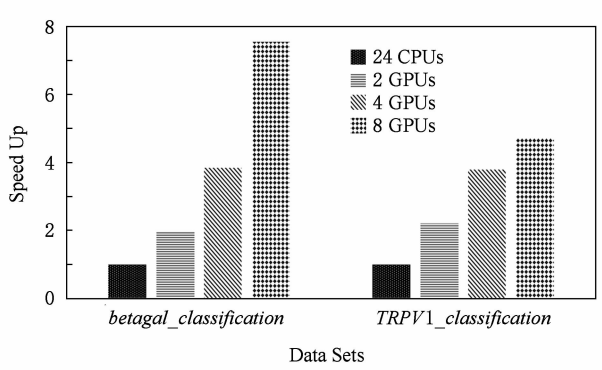
(b) Speed up for auto-refine mode

Fig. 5 Speed up of function and application

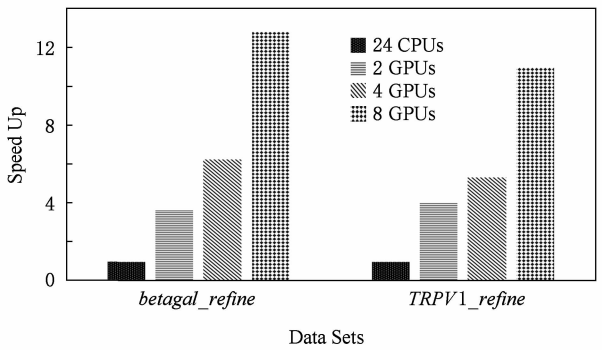
图 5 函数和应用的加速比

3.3 扩展性分析

我们还对 GPU 程序的可扩展性进行了分析,分别在 2,4,8 个 GPU 上进行了相应的测试,图 6 给出了使用不同 GPU 数目对于 24 个 CPU 核的加速情况,从图 6 中可以看出从 2 个 GPU 增加到 8 个 GPU 可以获得近似线性的加速比,尤其是针对 auto-refine 执行过程,说明本文实现的 GPU 算法具有良好的可扩展性. 具体而言,8 个 GPU 相比较 2 个 GPU 获得 2.5～3.5 的加速比. 其中部分数据显示出现了超线性加速,原因在于 RELION 是基于贝叶斯的 EM 算法,迭代次数受到初始模型的影响,初始模型具有一定的随机性.



(a) Scalability of 3D-classification mode



(b) Scalability of auto-refine mode

Fig. 6 Scalability of multiple GPU

图 6 多 GPU 的可扩展性

4 结束语

为解决冷冻电镜 3 维重构耗时的问题,本文针对 RELION 算法的特点,基于 GPU 开展并行化研究. 本文依次分析了 RELION 的原理、RELION 应用的加速热点,然后针对 GPU 细粒度体系结构对程序进行优化设计,提出了基于 GPU 的 RELION 并行优化方案并基于 CUDA 实现了整个程序. 实验结果表明,基于 GPU 的 RELION 实现可以获得 36 倍的加速比,对于计算密集型部分的加速比可以达到 75 倍以上,不同 GPU 数量的测试结果表明本文的实现具有良好的可扩展性.

下一步工作除了对程序进行进一步性能优化外,例如对 *storeWeight* 部分的原子操作进行优化,同时从性价比的角度出发,我们将考虑使用单精度浮点运算的可能性,在不降低结果精度的同时,进一步提高性能和降低硬件成本. 为了提高性能和实用性,我们还将考虑 GPU-CPU 协同计算的方案.

参 考 文 献

[1] Cheng Yifan, Nikolaus G, Pawel A, et al. A primer to single-particle cryo-electron microscopy [J]. Cell, 2015, 161(3): 438-449

[2] De Rosier D, Klug A. Reconstruction of three dimensional structure from electron micrographs [J]. Nature, 1968, 217(5124): 130-134

[3] Sorzano C O S, Marabini R, Velázquez-Muriel J, et al. XMIPP: A new generation of an open-source image processing package for electron microscopy [J]. Journal of Structural Biology, 2004, 148(2): 194-204

[4] Tang G, Peng L, Baldwin P R, et al. EMAN2: An extensible image processing suite for electron microscopy [J]. Journal of Structural Biology, 2007, 157(1): 38-46

[5] Grigorieff N. FREALIGN: High-resolution refinement of single particle structures [J]. Journal of Structural Biology, 2007, 157(1): 117-125

[6] Scheres S H W. A Bayesian view on cryo-EM structure determination [J]. Journal of Molecular Biology, 2012, 415(2): 406-418

[7] Li Xuemeng, Grigorieff N, Cheng Yifan. GPU-enabled FREALIGN: Accelerating single particle 3D reconstruction and refinement in Fourier space on graphics processors [J]. Journal of Structural Biology, 2010, 172(3): 407-412

[8] Li Linchuan, Li Xingjian, Tan Guangming, et al. Experience of parallelizing cryo-EM 3D reconstruction on a CPU-GPU heterogeneous system [C] //Proc of the 20th Int Symp on High Performance Distributed Computing. New York: ACM, 2011: 195-204

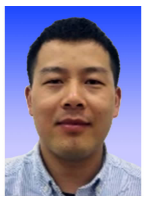
[9] Cianfrocco M A, Leschziner A E. Low cost, high performance processing of single particle cryo-electron microscopy data in the cloud [J/OL]. eLife, 2015, 4: e06664 [2016-03-07]. https://elifesciences.org/articles/06664v2

[10] Scheres S H W. RELION: Implementation of a Bayesian approach to cryo-EM structure determination [J]. Journal of Structural Biology, 2012, 180(3): 519-530

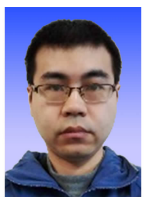
[11] Bell N, Hoberock J. Thrust: A productivity-oriented library for CUDA [M/OL] //GPU Computing Gems: Jade Edition. 2nd ed. San Francisco: Morgan Kaufmann, 2012 [2015-11-03]. https://pdfs.semanticscholar.org/0226/adea5e4f5f739633a83d159ca989045eefe5.pdf

[12] Scheres S H W, Chen Shaoxia. Prevention of overfitting in cryo-EM structure determination [J]. Nature Methods, 2012, 9(9): 853-854

[13] Liao Maofu, Cao Erhu, Julius D, et al. Structure of the TRPV1 ion channel determined by electron cryo-microscopy [J]. Nature, 2013, 504(7478): 107-112



Su Huayou, born in 1985. PhD, assistant professor. Member of CCF. His main research interests include high performance computing, GPU programming and parallel computing.



Wen Wen, born in 1986. Master. His main research interests include GPU programming and hybrid parallel computing.

Li Dongsheng, born in 1978. PhD, professor and PhD Supervisor. Member of CCF. His main research interests include distributed computing, network, big data, etc.