科学通报 2014 年 第 59 卷 第 14 期: 1358 ~ 1368
论 文 www.scichina.com csb.scichina.com
《中国科学》杂志社
SCIENCE CHINA PRESS

颗粒凝并动力学 Monte Carlo 方法的高效 GPU 并行计算

赵海波*, 徐祖伟, 刘昕, 史家伟, 郑楚光

华中科技大学煤燃烧国家重点实验室, 武汉 430074
* 联系人, E-mail: klinsmannzhb@163.com

2013-10-15 收稿, 2014-01-21 接受, 2014-04-01 网络版发表
国家自然科学基金(51276077, 51021065)、教育部新世纪优秀人才支持项目(NCET-09-0395)和多相复杂系统国家重点实验室开放课题(MPCS-2011-D-02)资助

摘要 Monte Carlo(MC)方法作为一种求解颗粒群平衡方程(PBE)的有效方法(PBMC), 由于它对多维问题的适应性、符合实际颗粒动力学特征的离散和随机本质、程序结构相对简单、易于编程实现等优点受到人们持久、普遍的关注. 但在涉及到颗粒凝并问题时, 常规的 PBMC 方法计算代价较高, 与模拟颗粒数目的平方成正比, 限制了其工程应用. 并行计算技术的快速发展, 特别是近年来 NVIDIA 公司提出的计算统一设备架构(CUDA)为 PBMC 的快速高效模拟提供了一个良好的平台. 本文在CUDA平台上实现了颗粒凝并动力学PBMC的图形处理器(GPU)并行计算(分别实现了累计概率法和接受-拒绝法选择凝并对)及中央处理器(CPU)的协同处理, 与目前广泛运行于 CPU 的串行计算相比, 取得了精确的计算结果和非常明显的加速, 计算代价仅与颗粒数目成正比, 在当前主流 GPU/CPU 设备上能够达到上百倍的加速比.

关键词
颗粒群平衡模拟
凝并
随机模拟
并行计算
CUDA
计算效率

颗粒群平衡模拟(PBM)研究的是离散单元(如多相流中离散的固体颗粒、液滴或气泡等, 下文均用颗粒表示)在各种动力学事件(包括碰撞、凝并(或团聚、凝聚)、冷凝/蒸发、成核、沉积等)主导下颗粒群的内部变量(通常为单组分颗粒的尺度, 也可包括颗粒中各组分质量、颗粒表面积、孔隙率、分形维数等)的分布函数的时间和空间演变过程. PBM 既可以作为一种单独的数值模拟手段来研究离散系统的动力学演变过程, 也可以与多相流模型耦合起来以考虑离散相-离散相、离散相-连续相间的相互作用(即四向耦合的多相湍流模型), 包括离散单元的碰撞、聚并、破碎以及相变等关键热点, 从而推动了多相流研究向更为科学的定量描述和精确控制的转变.

PBM 在诸多自然和工程领域得到了广泛应用[1~3], 包括燃烧(如燃烧合成纳米颗粒; 碳烟和多环芳香烃的生成长大; 燃烧源可吸入颗粒物的生成、生长和迁徙)、大气环境(如室内、室外气溶胶的生成、团聚等微物理化学过程; 自然环境中雨、雾、冰雹等的形成和降落)、化学工程(如流化床、精馏塔等过程设备中团聚体和气泡的破碎和聚并; 乳化液滴颗粒的聚合和表面化学反应; 结晶过程中晶体的生成和生长; 纳米粉体制备过程中的快速化学反应、凝并、摩擦破碎和干燥; 胶体的团聚、微混合及结构重建现象; 异类液相萃取过程中液滴颗粒的聚并、破碎、相间质量交换和相分离); 生物医学工程(如细胞的生成、生长和分裂以及内部蛋白质、DNA 和 RNA 的再分布)等领域.

在 PBM 中, 离散系统被视为 1 个服从统计学规律的整体, 每个动力学事件的发生服从某种概率(或称为核函数(kernel 函数)), 这些概率与整体系统属性(如多相流场等)、颗粒群内部变量(如单组

引用格式: 赵海波, 徐祖伟, 刘昕, 等. 颗粒凝并动力学 Monte Carlo 方法的高效 GPU 并行计算. 科学通报, 2014, 59: 1358–1368
Zhao H B, Xu Z W, Liu X, et al. Efficient GPU parallel computing of population balance — Monte Carlo method for coagulation (in Chinese). Chin Sci Bull (Chin Ver), 2014, 59: 1358–1368, doi: 10.1360/972013-76

分或多组分颗粒尺度、分形维数、孔隙率、表面积、组分浓度等)等密切相关, 统计上可由核模型定量表征; 颗粒群内部变量的动力学演变过程可由颗粒群平衡方程(PBE)或称为通用动力学方程(GDE)所描述:

$$\begin{aligned}\frac{\partial n(p,t)}{\partial t}=&\left\{\frac{1}{2}\int_{\Omega_p}\beta(p-s,s,t)n(p-s,t)n(s,t)\mathrm{d}s\right.\\&\left.-n(p,t)\int_{\Omega_p}\beta(p,s,t)n(s,t)\mathrm{d}s\right\}_{凝并}\\&+\left\{\int_{\Omega_p}\gamma(p,s,t)b(s,t)S(s,t)n(s,t)\mathrm{d}s\right.\\&\left.-S(p,t)n(p,t)\right\}_{破碎}\\&+\left\{\int_{\Omega_p}\gamma(p,s,t)b(s,t)S(s,t)n(s,t)\mathrm{d}s\right.\\&\left.-S(p,t)n(p,t)\right\}_{破碎}\\&+\left\{J_0(p,t)\delta(p_{\mathrm{nul}},p)\right\}_{成核}\\&-\left\{\partial\left(I(p,t)n(p,t)\right)/\partial p\right\}_{冷凝/蒸发}\\&-\left\{E(p,t)n(p,t)\right\}_{沉积}+\left\{R_{\mathrm{other}}\right\},\end{aligned}\tag{1}$$

其中 p 为颗粒群内部变量集, 如颗粒尺度 v、颗粒表面积 a、颗粒孔隙率 ε、颗粒分形维度 D_{f}、颗粒内部组分 i 的浓度 C_i 等, $\Omega_p\in R^{\xi}$ (ξ=1,…, 为内部变量的个数)表示颗粒群内部变量空间域; $n(p,t)$为颗粒群内部变量 p 的分布函数, $n(p,t)\mathrm{d}p$ 表示时刻 t, 变量范围在 p~p+dp 内的颗粒的数目浓度; $\beta(p,s,t)$是时刻 t、内部属性为 p 和 s 的 2 颗粒的凝并速率; $S(p,t)$, $J_0(p,t)$, $I(p,t)$和 $E(p,t)$分别为颗粒(p)的破碎、成核、冷凝/蒸发和沉积事件的发生速率; $\gamma(p,s,t)$为母颗粒(p)破碎产生子颗粒(s)的概率; $b(p,t)$为母颗粒(p)产生的子颗粒数目.

PBE 是 1 个典型的部分积分微分方程, 对于通常的多分散性、多组分的颗粒群和非线性的核模型而言, 在高维相空间 $R^{\xi\times t}$(ξ=1,…, p 表示颗粒群内部变量域, t 表示时间域)进行积分微分是非常困难的. 已有的 PBM 数值方法主要包括确定性的矩方法、分区法等以及基于随机描述的 Monte Carlo(MC)方法[4]. 确定性方法的优点是计算代价较小, 易于与计算流体力学(CFD)耦合, 但主要缺点是难以求解多变量 PBE. 实际上, 往往需要多个颗粒内部变量来准确描述颗粒“状态”(如纳米颗粒制备, 颗粒尺度和表面积是最常关注的 2 个状态, 而对于多组分纳米颗粒制备, 还需关注颗粒内部组分分布[5]), 因此需要同时获知颗粒群多个内部变量分布函数的动力学演变信息. MC 方法对高维问题的求解具有天然优势, 特别适合于多变量 PBM. 颗粒群平衡模拟 MC(PBMC)方法的基本特点是直接描述颗粒群的动力学特性而非“暴力”求解复杂的 PBE, 与实际研究对象具有一致的离散本质和随机特性, 易于得到颗粒轨道经历效应和历史效应以及颗粒内部结构的细节信息, 可直接扩展处理多变量高维工况, 算法相对简单易于编程实现. 但在涉及颗粒凝并问题时, 常规 PBMC 方法需要对模拟颗粒采取双重遍历的跟踪, 计算代价与模拟颗粒数目的平方成正比, 如此高的计算代价限制了其工程应用, 需要发展快速高效的求解算法.

目前已有多类 PBM 的 MC 方法. Kruis 等人[6]发展一种阶梯式常体积法模拟多变量系统, 可考虑颗粒尺度、荷电量、组分浓度等随时间的演变过程; Matsoukas 等人[7]发展常数目法研究液滴内双化学组分的微混合过程; Laurenzi 等人[8]则引入“凝并概率密度函数”和“凝并表”等概念而发展了一套新的事件驱动 MC 模拟多组分颗粒凝并过程, 用来降低计算代价和内存需求; Kraft 等人[9]发展强核函数法, 通过虚拟 Markov 步进高效模拟纳米颗粒演变过程; Irizarry[10]发展一种复杂的点集合 MC(PEMC), 通过把颗粒区间类比于不同的化学组分和构建随机产物 Markov 步进模型, 可较好地加速 MC 和模拟多变量系统; 本文发展了基于“异权值”模拟颗粒思想的 MC 方法[11~15]建立了异权值模拟颗粒相互作用(碰撞、凝聚等)的新模型, 通过主动调控有限数目的模拟颗粒使其相对均匀分布在颗粒群内部变量的高维空间内, 异权值 MC 方法对于统计噪声的降低和全尺度范围颗粒群的跟踪有着明显优势, 而以上等权值 MC[6~10]实际上可视为异权值 MC 的一种特例. 尽管异权值 MC 能较好地解决计算精度和统计噪声等问题, 但仍然与常规 MC 方法一样具有计算代价较大的缺点, 特别是对于双颗粒事件(如凝并), 其计算代价通常与颗粒数目平方成正比. 减少 MC 的计算代价使其适合工程模拟的方案之一是采取并行计算的策略. 基于中央处理

器(CPU)的并行计算是常用的技术, 包括单机上多个 CPU 核心的并行计算(基于多核心(Open MP))和集群中多节点 CPU 的并行计算(基于消息传递接口(MPI))等. 本文实现了异权值 MC 的 CPU 并行计算, 发现在 10 节点集群上基于 MPI 的并行计算可获得 4.64 倍的加速比, 而在 4 核心 CPU 上基于 Open MP 的并行计算可得到约 3 倍的加速比, 此时对于一个实际工况(如柱塞流气溶胶反应器制备 TiO_2 纳米颗粒的过程模拟)仍然需要几小时以上.

随着技术的发展, 图形处理器(GPU)的计算速度已经远远超出了桌面 CPU, 例如, Intel® Xeon™ E5-2680 的峰值单精度浮点计算能力为 345.6 Gflops, 而 NVIDIA® Tesla™ C2075 GPU 的峰值单精度浮点计算能力可达 1030 Gflops. 基于 GPU 的并行计算是近几年涌现的新技术, NVIDIA 公司提出了代表性的解决方案, 并发布了主流的 GPU 编程工具——计算统一设备架构(CUDA), 显示了强大的并行计算能力, 在分子动力学模拟[16,17]、格子-Boltzmann 方法[18~22]、计算流体动力学[23,24]等方面获得了惊人的成绩, 可实现几个数量级的计算能力提升, 得到了全世界的广泛关注, 如中国科学院过程工程研究所建立了单精度峰值超过 100 Tflops 的 CPU+GPU 并行计算系统, 并在其上实现多尺度离散模拟[25]. 对于上面所列举的确定性数值方法, 从 CPU 移植到 GPU 是当前的一个研究热点, 且效果良好. 随机模拟方法(如 MC)在某些问题(如计算金融的期权估计、核物理中的中子输运、化工过程中的多维多变量群平衡、稀薄气体动力学等)的求解方面具有无法取代的地位, 而且目前的计算效率普遍较低, 然而目前随机模拟的 GPU 并行计算研究相对较少, 加速效果也有待改进. 例如, 对于近连续流直接模拟蒙特卡罗(DSMC)的大规模 GPU 并行计算[26], 与单核 Intel Xeon X5670 CPU 相比, 使用单 GPU 和 16 个 GPU 可以使得计算代价分别减小 15 倍和 185 倍. 因此有必要研究随机模拟方法的 GPU 并行加速技术, 特别是需要根据 GPU 并行计算的特点来设计 MC 方法的计算策略. 本文主要任务是针对计算代价最大的凝并事件实施 PBMC 的 GPU 并行计算. 由于 GPU 以共享内存、单指令多数据(SIMD)模式实现大规模并行处理, 为了获得更大的加速比, PBMC 的算法结构、计算流程、数据存储、CPU 和 GPU 之间的数据通讯等应该与 GPU 并行计算策略相匹配, 使其发挥最大功效. 本文将 PBMC 在 CPU 上执行的串行代码重新进行规划设计, 使 CPU 与 GPU 协同计算处理, 成功地实现了高性能的 GPU 并行加速.

1 颗粒凝并动力学 MC 方法简介

常规 CPU 运行的颗粒凝并动力学 MC 方法具有如下流程[27].

(ⅰ) 初始化过程. 设定模拟时间长度, 输入模拟工况初始条件, 包括: 初始时刻实际颗粒群的尺度分布、凝并核模型、模拟系统的边界条件.

(ⅱ) 生成模拟颗粒. 采用"加权虚拟颗粒"方法产生数目权值不等的模拟颗粒群.

(ⅲ) 设置时间步长. 基于当前时刻模拟颗粒群内发生凝并事件的速率 R_c, 计算 2 次凝并事件之间的静止时间 τ(量纲 s)

$$\tau = 1/\left(V_S R_c\right) = 2\Big/\left(V_S \sum_{i=1}^{N_S} C_i\right), \tag{2}$$

其中 V_S 为计算区域体积、量纲 m^3, N_S 为计算区域内模拟颗粒总数目; R_c 为单位体积内凝并事件的发生速率(量纲为 $m^{-3}\,s^{-1}$)

$$R_c = \frac{1}{2}\sum_{i=1}^{N_S} C_i, \tag{3}$$

C_i 为单位时间单位体积内模拟颗粒 i 发生凝并事件的总概率, 量纲为 $m^{-3}\,s^{-1}$,

$$C_i = \frac{1}{V_S^2}\sum_{j=1,i\neq j}^{N_S}\left(\beta_{ij} w_j \frac{2\max\left(w_i, w_j\right)}{w_i + w_j}\right) = \frac{1}{V_S^2}\sum_{j=1,i\neq j}^{N_S}\beta'_{ij}, \tag{4}$$

其中凝并核 β_{ij} 为单位时间内模拟颗粒 i 和 j 发生凝并的概率、量纲 $m^3\,s^{-1}$; 规整化凝并核 $\beta'_{ij} = 2\beta_{ij} w_j \max\left(w_i, w_j\right)\Big/\left(w_i + w_j\right)$, w_i 和 w_j 分别为 2 颗粒的统计权值. 需要注意的是, 第 1 次计算 C_i 时采用式(4), 得到所有模拟颗粒的凝并概率需要进行双重遍历, 而之后 C_i 可采用智能薄记技术[28]进行更新, 见步骤(ⅵ), 可以提高 30~40 倍的计算效率.

(ⅳ) 凝并事件检测. 随机判断在静止时间 Δt_E 后凝并事件的发生情况; 依据当前时刻凝并速率 R_c, 采用累积概率法(CPM)或接受-拒绝法(ARM)选择发生凝并事件的 2 颗模拟颗粒 i 和 j.

CPM: 依次搜寻模拟颗粒群如果满足关系式

$$\tau V_S S_{i-1}/2 < r \leqslant \tau V_S S_i/2, \tag{5}$$

则模拟颗粒 i 为参与一次凝并事件的主颗粒, 这里

$S_i=\sum_{m=1}^{i}C_m$；而检索其余模拟颗粒如果满足

$$\frac{\tau}{2V_{\mathrm{s}}}\sum_{m=1,m\neq i}^{j-1}\beta'_{im}<r-\frac{\tau V_{\mathrm{s}}}{2}S_{i-1}\leqslant\frac{\tau}{2V_{\mathrm{s}}}\sum_{m=1,m\neq i}^{j}\beta'_{im},\quad(6)$$

则模拟颗粒 j 为模拟颗粒 i 的凝并伙伴. 其中 r 是位于[0,1]区间满足均匀分布的随机数.

ARM: 从模拟颗粒群中随机选取 2 个颗粒 i 和 j 如果满足:

$$r\leqslant\beta'_{ij}\Big/\max_{\forall k,\forall m}\{\beta'_{km}\}\quad(7)$$

则模拟颗粒 i 和 j 发生凝并事件.

(ⅴ) 凝并后果处理. 对步骤(ⅳ)所检测的模拟颗粒 i 和模拟颗粒 j 的凝并事件的后果进行处理, 模拟颗粒 i 数目权值为 w_i, 尺度为 v_i; 模拟颗粒 j 数目权值为 w_j, 尺度为 v_j, 凝并后的属性为

$$\begin{aligned}
&w_i>w_j,\begin{cases}w_i^*=w_i-w_j;v_i^*=v_i,\\ w_j^*=w_j;v_j^*=v_i+v_j;\end{cases}\\
&w_i<w_j,\begin{cases}w_i^*=w_i;v_i^*=v_i+v_j,\\ w_j^*=w_j-w_i;v_j^*=v_j;\end{cases}\\
&w_i=w_j,\begin{cases}w_i^*=w_i/2;v_i^*=v_i+v_j,\\ w_j^*=w_j/2;v_j^*=v_i+v_j,\end{cases}
\end{aligned}\quad(8)$$

其中上标“*”表示凝并事件之后.

(ⅵ) 智能薄记技术. 参与凝并事件的 2 颗粒 m 和 n 的凝并概率仍然采用式(4)计算, 而其余颗粒的凝并概率可根据下式进行更新

$$C_k^*=C_k-\beta'_{km}-\beta'_{kn}+\beta'^*_{km}+\beta'^*_{kn},\quad(9)$$

其中 $\beta'^*_{km}=2\beta^*_{km}w^*_m\max\left(w_k,w^*_m\right)\Big/\left(w_k+w^*_m\right)$, β^*_{km} 为基于新的颗粒状态(体积、直径等)而计算得到的常规凝并核.

(ⅶ) 时间长度判断. 如果时间步长 τ 的累加值超过了初始设定的模拟时间长度 t_{e}, 进入步骤(ⅷ), 否则返回步骤(ⅲ).

(ⅷ) 输出结果. 可以采用不同的随机数产生器种子进行多次 MC 模拟, 对模拟结果进行平均, 输出最终的模拟结果.

2 基于 CUDA 的 PBMC 并行算法设计

实际上, CPU 擅长复杂指令调度、循环、分支、逻辑判断以及执行等程序任务, 但难以实现数百个线程(thread)并行, 而 GPU 擅长计算密集度高而无逻辑关系数据的并行处理, 浮点运算快, 它基于大规模多线程(SIMT)架构, 由硬件管理轻量级线程, 可同时实现成千上万个相对简单线程的并行计算及零开销的线程切换. GPU 的另外 2 个主要优点是能耗小和成本低, 但存在“编程墙”的缺点. 2007 年提出并不断升级的 CUDA 大大改善了 GPU 通用并行计算的可编程性. CUDA 提供与 C 语言相似的编译器和运行库, 其并行计算的主要思想是: 对不同数据执行相同操作的应用程序(称为内核, kernel 函数)通过应用程序接口(API)加载在 GPU 计算设备上, 实现许多不同的 thread 同步执行, thread 组成线程块(block), block 又组成网格(grid), 内核函数以 block 为单位执行, block 间通过快速共享内存实现数据交流, 并同步执行以协调内存访问. CUDA 基于 2 层并行模型, 即 grid 中 block 之间的并行和 block 中的 thread 之间的并行.

PBMC 方法在算法、研究对象以及信息处理等 3 个层面上具有较高的并发性. (1) MC 方法的并行性: MC 计算是多处理器并行计算的最佳对象, 因为它满足并行计算的基本要素: 数据独立; 负载平衡; 计算粒度大; I/O 通讯少. 因此算法上可以获得高加速倍率[29]. (2) 颗粒运动的并行性: 颗粒基于拉格朗日描述追踪, 在碰撞之前相互独立, 碰撞的概率依赖于颗粒对的特征量(碰撞核)和群体的统计量(平均碰撞频率), 每个颗粒对的组合可以对应 1 个 thread, 颗粒对的随机选择过程能够对应多个 thread 的并行, 颗粒群的统计量可以通过 thread 的归约得到. (3) 研究对象与数据结构的映射: 颗粒群信息能够矩阵化, 所有颗粒的内部变量能够以矩阵的形式存储于共享内存, 颗粒信息的索引、计算和更新能够分配给不同层次的 thread 并行处理(图 1). 对应于 CUDA 逻辑上和物理上多层次并行模型和结构, 根据 PBMC 的基本特点, 并行计算的基本思路是: 利用 GPU 执行高度线程化的数据并行处理任务, 而由 CPU 进行复杂逻辑和事务处理等串行计算, 最大限度地发挥计算机的处理能力, 实现桌面上的超级计算. 为了匹配 GPU 并行计算策略, PBMC 的 CUDA 程序流程设计如图 2 所示.

软硬件性能的配合对并行计算的效果具有重要影响, 本文的工作是在 1 台塔式工作站(HP® Z820)上实施的. 操作系统(OS): Red hat enterprise 6.0 Linux x86-64; CPU: Intel® Xeon™ E5-2680(主频 2.7 GHz, 8 核心); 显卡: NVIDIA® Tesla™ C2075 GPU(448 个

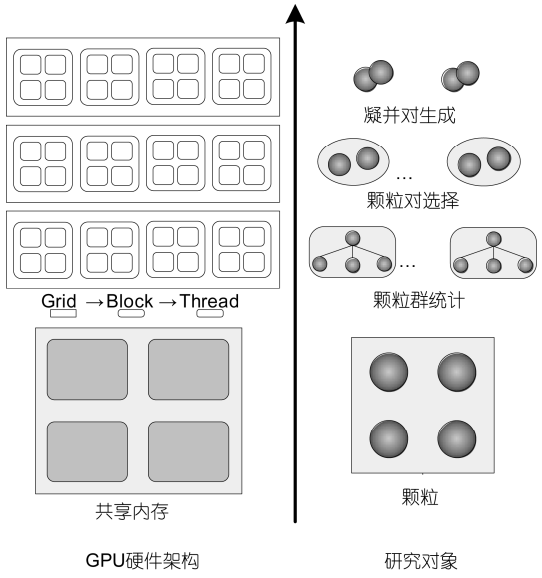

图 1 颗粒凝并问题并行结构示意图

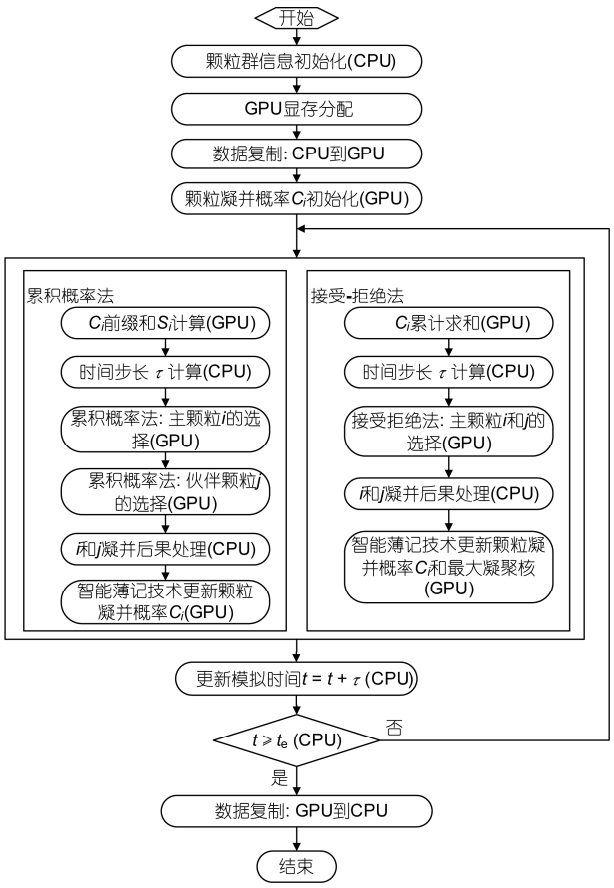

图 2 PBMC 的 CUDA 程序流程图

CUDA 核心, 14 个多重处理器(Multiprocessor, 每个包含 32 个 CUDA 核心), 计算能力为 2.0, 显卡驱动为 295.41 版); CUDA Toolkit: 4.20 版; C 语言编译器: GCC 4.4.6.

CPU 与 GPU 协同计算中 CPU 执行的部分以及在 CPU 上运行的对照工况均是在 Intel® Xeon™ E5-2680 处理器上单核串行. 以下介绍 GPU 并行计算的具体算法设计, 重点阐述执行于 GPU 的应用程序内核函数. 假设初始模拟颗粒数目 N_s=10000(整个计算过程保持常数目), 执行于 GPU 设备上的 kernel 函数的 block 大小为 1024(GPU 的硬件架构限制), 且 block 和 grid 均采用一维设计.

2.1 凝并概率初始化

在初始化中涉及到 3 个 kernel 函数, 首先计算某 1 个模拟颗粒 k 与其他 N_s-1 个模拟颗粒的凝并概率 $\{\beta'_{kl}\}(l\neq k)$, 需要 9999 个 thread, 即 $\left\lceil \frac{9999}{1024} \right\rceil$=10 个 block, 故计算所有颗粒的凝并概率 β' 需要 10×10000 个 block. 由于目前 CUDA 计算能力的限制, 每个 grid 包含的 block 最大数量为 65535(GPU 的硬件架构限制), 所有 thread 运行一次可以计算出 $\left\lfloor \frac{65535}{10} \right\rfloor$=6553 个颗粒的 β', 因此在初始化 β' 的过程中需要对该 kernel 进行 2 次循环. 为了降低访问全局存储器所带来的访存延迟, 在每个 block 中分配大小为 1024×sizeof(float)的共享存储器存储 β', 在每个 block 计算完成后, 将 block 内所有的 β' 采用归约(reduce)方法求和并存储在全局存储器内, 需要 10×10000×sizeof(float)字节的空间, 这是第 2 个 kernel 函数. 对于 CPM, 根据式(4)计算每个模拟颗粒与其他 N_s-1 个模拟颗粒的凝并概率之和, 将模拟颗粒 i 所包含的 block 和再加起来得到 C_i, 并存储在 10000×sizeof(float)字节的全局存储器内; 对于 ARM, 采用并行归约方法计算得到总的凝并核 $\sum_{i=1}^{N_s} C_i$ 和最大凝并核 $\beta'_{\max}$, 这是第 3 个 kernel 函数, 由于后面对 C_i 和 $\beta'_{\max}$ 的更新过程采用智能簿记技术, 所以从第二次时间循环开始不需要再调用此 kernel 函数. 初始化过程如图 3 所示.

2.2 智能薄记核

凝并事件后对 C_i 进行更新分成两部分, 第一部

分是参加凝并的颗粒 m 和 n 的凝并概率按照式(4)进行计算，分配 20 个 block 并行计算 $\{\beta'(m,0),\beta'(m,1),\cdots,\beta'(m,9999)\}$ 和 $\{\beta'(n,0),\beta'(n,1),\cdots,\beta'(n,9999)\}$，然后计算出新的 C_m 和 C_n；第二部分是为每个未参与凝并的颗粒分配 1 个 thread，根据式(9)计算新的凝并概率 C_k^*. 对于 ARM 中最大凝并核的更新，采用并行归约的方法求得序列 $\{\beta'(m,0),\beta'(m,1),\cdots,\beta'(m,9999)\}$ 和 $\{\beta'(n,0),\beta'(n,1),\cdots,\beta'(n,9999)\}$ 中的最大值.

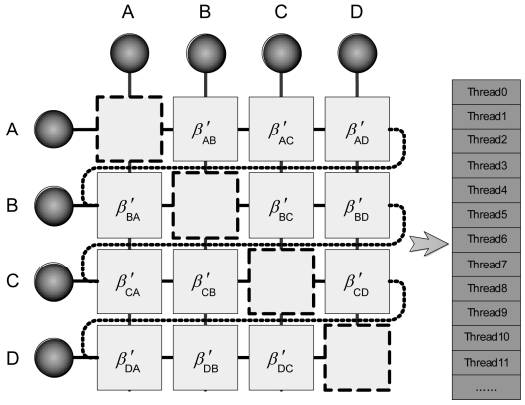

图 3　凝并概率 β' 初始化 kernel 线程结构示意图

2.3　前缀和核

颗粒凝并概率的前缀和 $S_i=\sum_{m=0}^{i}C_m$，初始化核已经得到了每个颗粒的凝并概率 C_i，为每个颗粒分配 1 个 thread，需要 $\left\lceil\frac{10000}{1024}\right\rceil$=10 个 block. 首先，计算每个 block 内 1024 个颗粒的前缀和，在 block 求和中可以利用共享存储器暂存数据，减少访存延迟，计算方法如图 4 所示(以 8 颗颗粒作为 1 个 block 为例). 在完成每个 block 内前缀和后，启动第 2 个 kernel 函数计算整个 grid 中的 S_i. 比如对于 block 2，需要将 block 0, block 1 中最后 1 个 thread 前缀和与 block 2 中每个前缀和相加，在图 5 中，以前 3 个 block 为例说明 S_i 的计算过程.

2.4　主颗粒选择核

由前面计算的前缀和 S_{9999}，根据式(2)可以得到 2 次凝并事件之间的静止时间 $\tau=2/(V_S S_{9999})$，为每个模拟颗粒分配1个thread，根据式(5)将前缀和序列 $\{S_0,S_1,\cdots,S_{9999}\}$ 与CPU产生的随机数 r 按 $\tau V_S S_i-r$ 进行差值处理，每个thread得到的差值存储在共享存储器内，并把每个block两端的差值复制到全局存储器内；然后从第1个thread开始将相邻的2个差值进行符号比较，找到符号相异的2个差值对应的线程索引，其中较大的线程索引为凝并主颗粒的序号，如图6所示.

2.5　伙伴颗粒选择核

确定凝并主颗粒 i 后，根据式(6)搜索凝并伙伴 j，这与凝并主颗粒的选择过程相同，而且这里需要对凝并概率序列 $\{\beta'_{i0},\beta'_{i1},\cdots,\beta'_{i9999}\}$ 求前缀和，与前面的 C_i 前缀和计算过程类似.

2.6　接受-拒绝核

从模拟颗粒群中随机选取满足式(7)的 2 个颗粒 i 和 j 作为凝并对. 第 1 个 kernel 是采用 CURAND 库

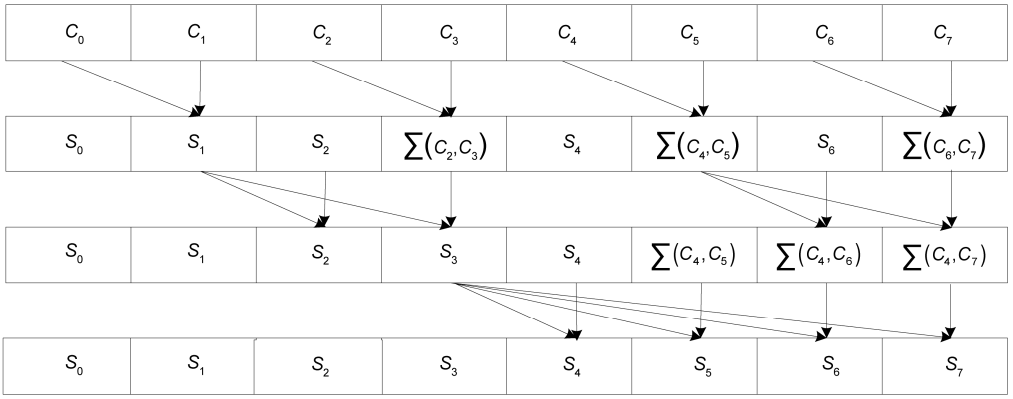

图 4　Block 内颗粒凝并概率前缀和算法

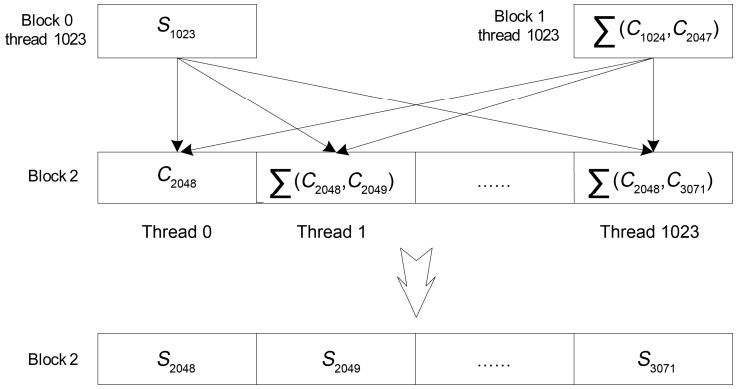

图 5 Grid 内颗粒凝并概率前缀和算法

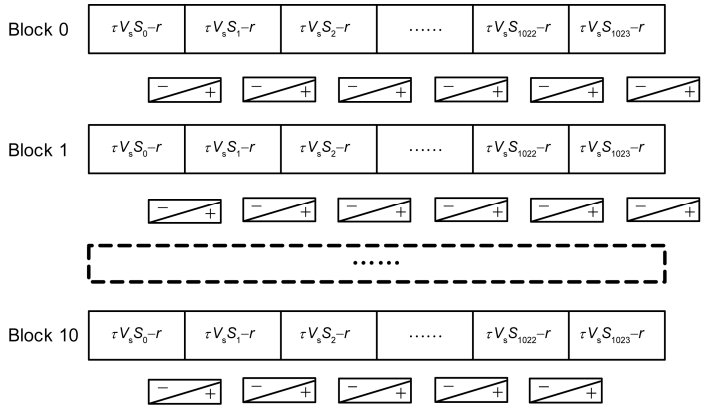

图 6 凝并主颗粒的搜寻

中的梅森旋转(Mersenne twister)算法在 GPU 并行产生随机数序列, 存储在全局存储器内. 梅森旋转算法基于有限二进制段上的矩阵线性递归, 可以快速产生高质量的伪随机数, 是目前比较高效的一种并行随机数算法[29]. 第 2 个 kernel 是分配 256 个 thread 依次读取随机数序列 $\{r_0,r_1,\cdots,r_{255}\}$, 对应模拟颗粒索引 $\{r_0N_s,r_1N_s,\cdots,r_{255}N_s\}$, 计算相邻 2 个 thread 中随机颗粒对的凝并概率 $\{\beta_0',\beta_1',\cdots,\beta_{255}'\}$, 然后再读取 256 个随机数 $\{r_0',r_1',\cdots,r_{255}'\}$, 根据式 (7) 计算 $r_i'\beta_{\max}'-\beta_i'$ (i 为线程索引), 把所有线程计算的结果写入 1 个数组复制到主机内存里, 然后在 CPU 上检测这个数组是否存在不大于 0 的元素, 如果存在, 则第 1 个检测到的元素对应的颗粒对就被选为凝并对, 如果不存在, 则返回第 2 个 kernel 函数, 直到成功选取凝并对. 计算过程如图 7 所示.

3 算例结果与讨论

下面采用一种应用较广的实际工况——自由分子区布朗凝并来验证基于 CUDA 的 PBMC 并行算法. 初始颗粒群为单分散性, 粒径 d_0=3 nm, 密度 ρ_p=1000 kg m^{-3}, 实际颗粒数目 N_0=1.0×10^{17}, 采用异权值模拟颗粒策略对实际颗粒群进行加权处理, 生成 10^3~10^4 个模拟颗粒, 初始数目权值 $w_0=N_0/N_s$, 模拟过程中 N_s 保持不变. 模拟颗粒的凝并核为

$$\beta'\left(d_i,d_j\right)=2w_j\frac{\max\left(w_i,w_j\right)}{w_i+w_j}\left(\frac{3k_BT}{\rho_p}\right)^{\frac{1}{2}}\times\left(d_i+d_j\right)^2\left(d_i^{-3}+d_j^{-3}\right)^{\frac{1}{2}},\tag{10}$$

式中 Boltzmann 常数 k_B=1.38×10^{-23} J K^{-1}, 气体热力学温度 T=300 K. 模拟过程中保存计算结果的窗口时间点为 τ_{coag}, $3.6\tau_{coag}$, $4\tau_{coag}$, $4.3\tau_{coag}$, $5.3\tau_{coag}$, $10\tau_{coag}$, $50\tau_{coag}$, $100\tau_{coag}$, $500\tau_{coag}$, $1000\tau_{coag}$, 其中特征时间尺度 $\tau_{coag}=n_0^{-1}\left(\frac{3k_BTd_0}{\rho_p}\right)^{-\frac{1}{2}}$.

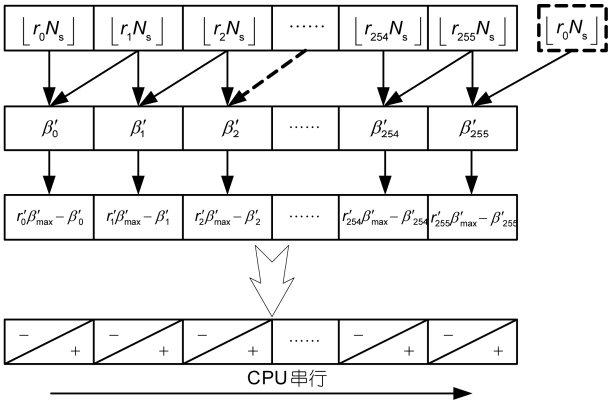

图 7　ARM 选择凝并对

3.1　计算精度比较

在模拟颗粒数N_s=10000的情况下分别对CPM和ARM在GPU上的并行计算结果与CPU的串行计算结果进行比较，包括颗粒群演变特定时刻点($1000\tau_{coag}$)的颗粒尺度分布(PSD)和整个过程无量纲颗粒尺度分布的零阶矩(M_0)和一阶矩(M_1)变化过程，如图8所示. 由于布朗凝并的PSD时间演变没有理论分析解，这里将计算得到的 PSD 自保持分布与文献[30]的自保持分布数值解进行对照，其中无量纲颗粒尺度 $\eta=v/\bar{v}=Nv/M$，颗粒尺度分布的无量纲概率密度函数 $\psi(\eta,t)=Mn(v,t)/N^2$，v 为颗粒尺度(体积)，N为颗粒总的数目浓度，M为颗粒总的质量浓度(或体积分数). 无量纲颗粒尺度分布 $p(k,t)=\dfrac{n_k(t)}{\sum_{k=1}^{K}n_k(t)}$，零阶矩 $M_0(t)=\sum_{k=1}^{K}v_k^0(t)p(k,t)$，一阶矩 $M_1(t)=\sum_{k=1}^{K}v_k^1(t)p(k,t)$，无量纲尺度 $v_k=\dfrac{v_k^+ + v_k^-}{2\bar{v}}$，$k$为颗粒尺度分区序号，$K$ 为总的颗粒尺度区间数，n_k 为尺度区间k内的颗粒数目，v_k^+ 和 v_k^- 分别为k区间的尺度上限和下限. 理论分析表明 M_0 和 M_1 均为 1[30].

计算结果按照离散-分区法将颗粒尺度分成 120 个区间，2 种方法(CPM 和 ARM)在 GPU 上的并行运算结果与 CPU 的串行计算结果相比，在颗粒群演变时间 $t_e=1000\tau_{coag}$ 时，颗粒尺度分布在统计上没有明显差别，均与自保持曲线吻合得很好；颗粒尺度的无量纲化零阶矩(M_0)均一直保持为 1，一阶矩(M_1)有微

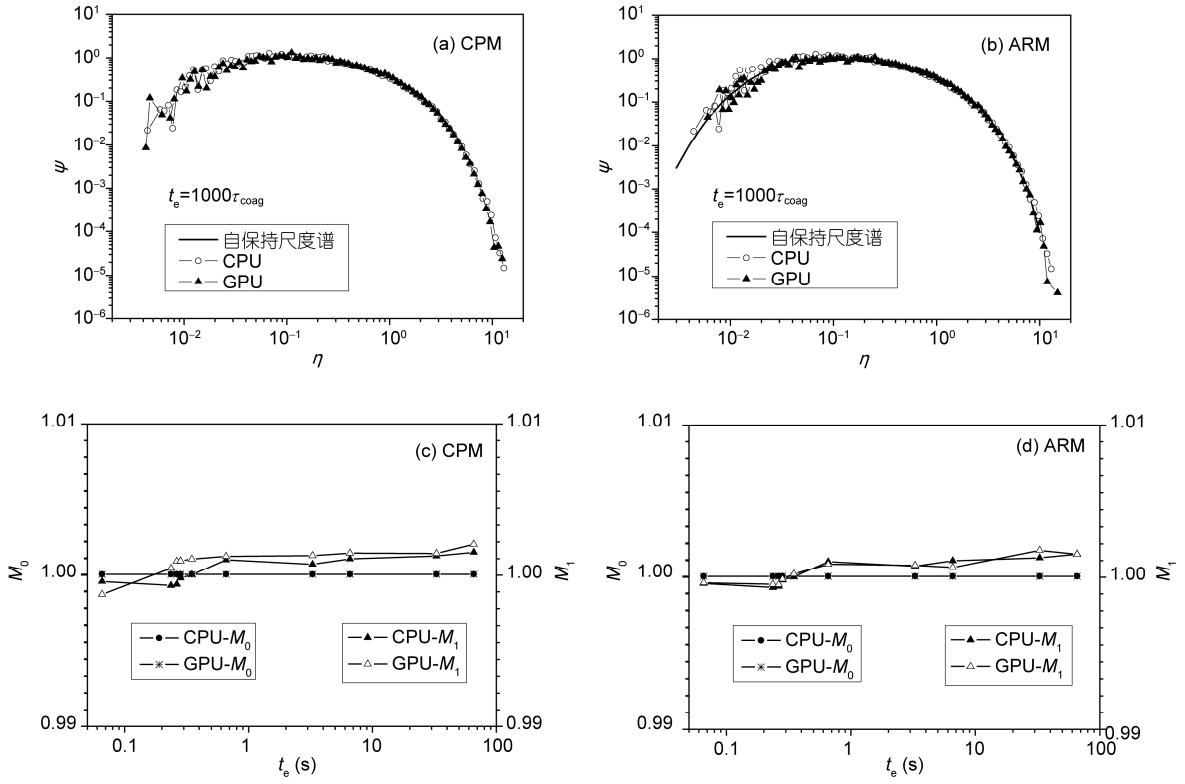

图 8　计算精度比较(N_s=10000)

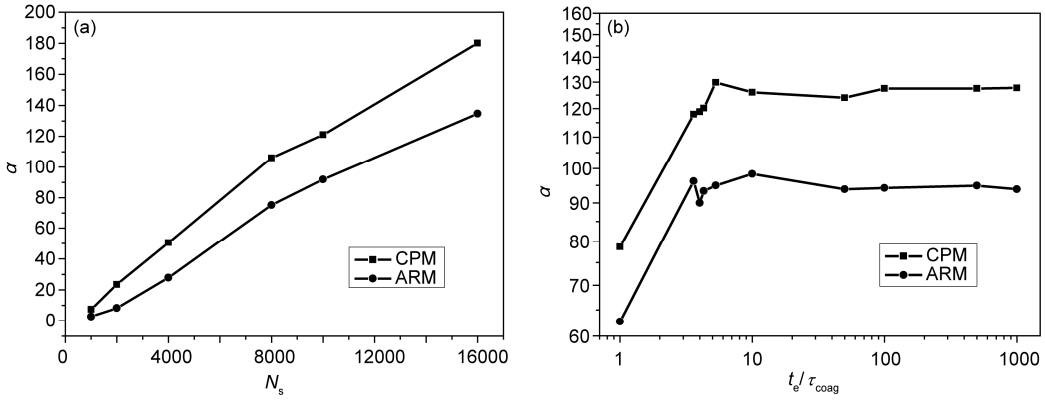

图 9 并行加速比

(a) 加速比α与模拟颗粒数 N_s; (b) 加速比α与演变时间 t_e(N_s=10000)

小的波动(±2‰), 这是 MC 方法的随机噪声引起的误差.

3.2 计算效率比较

并行加速效果是本文最关心的问题, 分别比较了 CPM 和 ARM 在不同模拟颗粒数目(N_s=1000, 2000, 4000, 8000, 10000, 16000)情况下 GPU 并行对 CPU 串行的加速比, 定义加速比 $\alpha = t_{CPU}/t_{GPU}$. 如图 9(a)所示, 随着模拟颗粒的增加, 加速比有显著的提升, 近似呈现正比的线性关系, 这表明 GPU 程序计算时间与模拟颗粒数目成正比. 在 N_s=10000 的情况下, CPM 能达到 120 倍的加速, ARM 有 90 倍的加速, 图 9(b)展示了加速比随模拟时间的变化关系, 开始时的初始化过程需要 CPU 和 GPU 通过主机内存和设备内存通信, 消耗较多的时钟周期, 加速比较低, 后续的计算都保持较稳定的加速比, 整个计算过程具有时间均匀性. 需要说明的是在 CPU 上运行时, ARM 比 CPM 的效率稍高, 但在 GPU 上运行时, ARM 的效率较低, 其中的主要原因是 ARM 的 GPU 并行随机数生成算法(Mersenne Twister)需要 624 个字长的静态数组, CUDA 的静态数组只能通过全局存储器实现, 随机数生成器需要反复读写全局内存, 开销较大; 而且 ARM 的凝并对选择过程(图 7)需要 GPU 和 CPU 之间通信, 消耗很多的时钟周期.

4 结论

颗粒凝并动力学的 PBMC, 在一个时间步长内具有很高的并行度, 特别是颗粒群的初始化过程、凝并颗粒对的选择、颗粒群信息的智能薄记更新, 而且模拟颗粒的序号很方便与并行线程索引映射, 符合单指令多线程的并行计算构架, 在 CUDA 平台上能够很好地发挥 GPU 的并行处理能力. 在模拟过程中, 对各个任务细分成多个内核函数, 实现了深度的并行, 使 CPM 和 ARM 都得到了较高的加速效果, 特别是 CPM 的加速更明显. 传统 CPU 串行计算时间与模拟颗粒数目平方成正比, 而 GPU 并行计算时间与模拟颗粒数目成正比, 同时计算精度几乎一致. 但 PBMC 在 GPU 上的并行仍有一定的优化空间, 如 thread 的同步、block 之间的通信、CPU 与 GPU 的通信以及 CUDA 存储结构的合理应用. 总之, GPU 的并行计算对于计算代价高的 PBMC 方法有非常广阔的应用前景, 为复杂的实际颗粒群平衡问题(多维、多变量、CFD 耦合)提供了一个潜力巨大的解决方案.

致谢 感谢与Institute for Nano-Structure (University of Duisburg-Essen)的Prof. Dr. Kruis F. E. 和Dr. Wei J. 的交流和讨论.

参考文献

1 Friedlander S K. Smoke, Dust and Haze: Fundamentals of Aerosol Dynamics. New York: Oxford University Press, 2000

2 Seinfeld J H, Pandis S N. Atmospheric Chemistry and Physics: From Air Pollution to Climate Change. New York: Wiley-Interscience, 1997
3 Ramkrishna D. Population Balances: Theory and Applications to Particulate Systems in Engineering. San Diego: Academic Press, 2000
4 赵海波, 郑楚光. 离散系统动力学演变过程的颗粒群平衡模拟. 北京: 科学出版社, 2008
5 Zhao H, Kruis F E, Zheng C. Monte carlo simulation for aggregative mixing of nanoparticles in two-component systems. Ind Eng Chem Res, 2011, 50: 10652–10664
6 Kruis F E, Maisels A, Fissan H. Direct simulation Monte Carlo method for particle coagulation and aggregation. AIChE J, 2000, 46: 1735–1742
7 Matsoukas T, Lee K, Kim T. Mixing of components in two-component aggregation. AIChE J, 2006, 52: 3088–3099
8 Laurenzi I J, Bartels J D, Diamond S L. A general algorithm for exact simulation of multicomponent aggregation processes. J Comput Phys, 2002, 177: 418–449
9 Kraft M, Wagner W. An improved stochastic algorithm for temperature-dependent homogeneous gas phase reactions. J Comput Phys, 2003, 185: 139–157
10 Irizarry R. Fast Monte Carlo methodology for multivariate particulate systems-I: Point ensemble Monte Carlo. Chem Eng Sci, 2008, 63: 95–110
11 Zhao H, Kruis F E, Zheng C. Reducing statistical noise and extending the size spectrum by applying weighted simulation particles in monte carlo simulation of coagulation. Aerosol Sci Tech, 2009, 43: 781–793
12 Zhao H, Zheng C. Correcting the multi-Monte Carlo method for particle coagulation. Powder Technol, 2009, 193: 120–123
13 Zhao H, Zheng C. A new event-driven constant-volume method for solution of the time evolution of particle size distribution. J Comput Phys, 2009, 228: 1412–1428
14 Zhao H, Kruis F E, Zheng C. A differentially weighted Monte Carlo method for two-component coagulation. J Comput Phys, 2010, 229: 6931–6945
15 Zhao H, Zheng C. A population balance-Monte Carlo method for particle coagulation in spatially inhomogeneous systems. Comput Fluids, 2013, 71: 196–207
16 陈飞国, 葛蔚, 李静海. 复杂多相流动分子动力学模拟在 GPU 上的实现. 中国科学 B 辑: 化学, 2008, 38: 1120–1128
17 徐骥, 葛蔚, 任瑛, 等. Particle-Mesh Ewald(PME)算法的 GPU 加速. 计算物理, 2010, 27: 548–554
18 李博, 李曦鹏, 张云, 等. 耦合 Nvidia/AMD 两类 GPU 的格子玻尔兹曼模拟. 科学通报, 2009, 54: 3177–3184
19 黄昌盛, 张文欢, 侯志敏, 等. 基于 CUDA 的格子 Boltzmann 方法: 算法设计与程序优化. 科学通报, 2011, 56: 2434–2444
20 Zhou H, Mo G, Wu F, et al. GPU implementation of lattice Boltzmann method for flows with curved boundaries. Comput Method Appl M, 2012, 225-228: 65–73
21 Xiong Q, Li B, Xu J, et al. Efficient parallel implementation of the lattice Boltzmann method on large clusters of graphic processing units. Chin Sci Bull, 2012, 57: 707–715
22 Li C, Maa J Y, Kang H. Solving generalized lattice Boltzmann model for 3-D cavity flows using CUDA-GPU. Sci China Phys Mech Astron, 2012, 55: 1894–1904
23 王健, 许明, 葛蔚, 等. 单相流动数值模拟的 SIMPLE 算法在 GPU 上的实现. 科学通报, 2010, 55: 1979–1986
24 董廷星, 李新亮, 李森, 等. GPU 上计算流体力学的加速. 计算机系统应用, 2011, 20: 104–109
25 多相复杂系统国家重点实验室多尺度离散模拟项目组. 基于 GPU 的多尺度离散模拟并行计算. 北京: 科学出版社, 2009
26 Su C C, Smith M R, Kuo F A, et al. Large-scale simulations on multiple Graphics Processing Units (GPUs) for the direct simulation Monte Carlo method. J Comput Phys, 2012, 231: 7932–7958
27 赵海波, 郑楚光. 描述颗粒凝并动力学的事件驱动常体积法. 中国科学 E 辑: 技术科学, 2008, 38: 1836–1849
28 Zhao H, Kruis F E, Zheng C. Reducing statistical noise and extending the size spectrum by applying weighted simulation particles in Monte Carlo simulation of coagulation. Aerosol Sci Tech, 2009, 43: 781–793
29 陶应龙, 王建国, 牛胜利, 等. MCATNP 蒙特卡罗粒子输运程序的 MPI 并行化. 核电子学与探测技术, 2011, 31: 490–494
30 Matsumoto M, Nishimura T. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. ACM Trans Model Comput Simul, 1998, 8: 3–30
31 Vemury S, Pratsinis S E. Self-preserving size distributions of agglomerates. J Aerosol Sci, 1995, 26: 175–185

Efficient GPU parallel computing of population balance — Monte Carlo method for coagulation

ZHAO HaiBo, XU ZuWei, LIU Xin, SHI JiaWei & ZHENG ChuGuang

State Key Laboratory of Coal Combustion, Huazhong University of Science and Technology, Wuhan 430074, China

The population balance-Monte Carlo (PBMC) method has become increasingly popular because the discrete and stochastic nature of the MC method is especially suited for particle dynamics. However, for the two-particle events (typically, particle coagulation), the double looping over all simulation particles is required in normal PBMC methods, and simulating particle coagulation is in general a challenging computational task due to its numerical complexity and the computing cost. The compute unified device architecture (CUDA) is a programming approach for performing scientific calculations on a graphics processing unit (GPU) as a data-parallel computing device. In this article we present an implementation of accelerating PBMC method based on the Inverse scheme and the Acceptance-rejection (AR) scheme for simulating particle coagulation on the GPU. The main idea is to implement the highly threaded data-parallel processing tasks by using GPU and serial computing of complex logic and transaction processing by CPU. Furthermore, the computation accuracy of the PBMC on GPU was validated with a benchmark, a CPU-based discrete-sectional method. To evaluate the accelerating performance, the computing time on the GPU against its sequential counterpart on the CPU was compared. The speedups show that the GPU can accelerate the PBMC by a factor from decades to more than one hundred, depending on the number of simulation particle.

population balance modeling, coagulation, Monte Carlo method, parallel computing, CUDA, computational efficiency

doi: 10.1360/972013-76