文章编号：1000-8055(2022)00-20250348-10 doi: 10.13224/j.cnki.jasp.20250348

CUDA 平台的非结构网格 DSMC 并行算法

王 志，王学德
（南京理工大学 能源与动力工程学院，南京 210094）

摘 要： 为提升稀薄气体流动模拟的计算效率，针对非结构网格在复杂几何建模中的适应性特点，构建了一种基于统一计算设备架构(CUDA)平台的非结构网格直接模拟蒙特卡洛(DSMC)并行算法。建立通用的 DSMC 求解框架，设计高效的网格搜索与粒子追踪机制，并在 CUDA 平台上对流场初始化、粒子推进、碰撞、排序及结果取样等核心模块进行了并行优化。通过合理划分线程与存储资源，解决了图形处理器(GPU)计算中数据一致性、线程发散与负载不均衡等问题。以超声速圆柱绕流为算例进行验证，结果表明：该算法在单卡 GPU 上整体加速比达到 52.5；基于网格并行策略的各模块取得了 24.7～53.7 的加速比，基于分子并行策略的各模块取得了 47～68 倍的加速比。该算法显著提升了非结构网格 DSMC 模拟的并行性能，为高马赫数稀薄气体流动的高效数值模拟提供了一种可行方案。

关 键 词： 直接模拟蒙特卡洛(DSMC)方法；非结构网格；图形处理器(GPU)并行计算；
统一计算设备架构(CUDA)平台；稀薄气体动力学

中图分类号： V411.4 **文献标志码：** A

Parallel DSMC algorithm for unstructured grids on CUDA platform

WANG Zhi，WANG Xuede
（School of Energy and Power Engineering，
Nanjing University of Science and Technology，Nanjing 210094，China）

Abstract: To enhance the computational efficiency of rarefied gas flow simulations, a compute unified device architecture (CUDA) platform-based parallel direct simulation Monte Carlo (DSMC) algorithm for unstructured grids was developed considering the adaptability of unstructured meshes to complex geometries. A general DSMC framework was established, and efficient grid-search and particle-tracking mechanisms were designed. The core computational modules—including flow-field initialization, particle movement, collision, sorting, and sampling—were parallelized and optimized on the CUDA platform. By rationally allocating threads and memory resources, issues such as data consistency, thread divergence, and load imbalance in graphics processing unit (GPU) computing were effectively addressed. The algorithm was validated through simulation of supersonic flow over a circular cylinder. Results showed that the overall speedup on a single GPU reached 52.5; the speedups of individual modules based on the grid-parallel strategy ranged from 24.7 to 53.7, while those based on the molecule-parallel strategy ranged from 47 to 68. The proposed algorithm significantly improved the parallel performance of DSMC simulations on unstructured grids, thus providing a feasible approach for efficient numerical simulation of high-Mach-number rarefied gas flows.

Keywords: direct simulation Monte Carlo (DSMC) method；unstructured mesh；
graphics processing unit (GPU) parallel computing；
compute unified device architecture (CUDA) platform；rarefied gas dynamics

收稿日期： 2025-07-23

作者简介： 王志(2000—)，男，硕士生，主要从事稀薄气体动力学研究。E-mail: wangzhinjust@qq.com

通信作者： 王学德(1977—)，男，副教授，博士，主要从事稀薄气体动力学研究。E-mail: wangxuede2000@njust.edu.cn

引用格式： 王志，王学德. CUDA 平台的非结构网格 DSMC 并行算法[J]. 航空动力学报，2022，X(X)：20250348. WANG Zhi，WANG Xuede. Parallel DSMC algorithm for unstructured grids on CUDA platform[J]. Journal of Aerospace Power，2022，X(X)：20250348.

美国太空探索(SpaceX)、一网(OneWeb)、亚马逊(Amazon)等商业公司相继提出大规模低轨通信星座计划[1],其中,美国太空探索技术公司已成功实现了可重复、低成本、高可靠的航天发射。近年来我国在月球探测、载人航天和空间站建设等一系列航天活动中取得重大进展,对可重复使用火箭等类似用于空间探测的产品的研发不断涌现、更新,这无疑需要对空间飞行器的气动特性进行更快速的计算模拟。经过数十年的不断发展,直接模拟蒙特卡洛(direct simulation Monte Carlo, DSMC)方法[2]已成为研究稀薄气体动力学的强有力工具,该方法也已在稀薄流域问题的求解上得到广泛应用[3-11]。

DSMC 的应用主要采用的网格分为直角网格、结构网格和非结构网格 3 类。其中,非结构网格 DSMC 方法,在复杂几何外形的计算网格生成方面具有自动化程度高、生成周期短、分布控制灵活等优点,已被 DSMC 工程计算广泛采用[12]。

限制 DSMC 发展的主要因素是它需要模拟几百万甚至数以亿计的粒子运动与碰撞,计算量十分庞大,对计算机的要求很高。通过将 DSMC 算法并行化进而提高计算效率,不仅能满足高性能计算需求,还能实现更高效的稀薄气体流动模拟,这对航空航天、微纳米技术等领域的发展具有重要推动作用。

20 世纪 90 年代,分布式存储消息传递接口(message passing interface, MPI)并行编程模型标准发布。MPI 被用于不同处理器之间的通信,是并行计算领域消息传递最为权威的设计标准[13],逐渐成为 DSMC 并行化的标准工具之一。20 世纪后,随着大规模分布式计算系统的发展应用(如超级计算机),使得 DSMC 方法得以在更大规模的流体动力学模拟中应用。通过大规模的计算域分解和 MPI 并行化,DSMC 被应用于复杂的三维问题中,如航天器外部稀薄气体环境的模拟[14]。但是该系统的性能受到并行通讯速度的影响很大,虽然减少中央处理器(central processing unit, CPU)的使用量可以减少通讯,但这要求 CPU 具有更高的性能。然而,根据基准测试软件开发商 PassMark Software[15]的数据,2023—2025 年最近 3 年各个平台旗舰级 CPU 的性能分数几乎没有提升。其中主要原因便是芯片的制造工艺逐渐达到极限,CPU 的摩尔定律逐渐失效,因此 CPU 的计算速度逐渐遇到瓶颈。近年来,计算机体系结构变得越来越复杂,主流的计算机架构从最初的单核转变为多核甚至异构多核,呈现出多核并行和异构加速的典型特征。通用可编程图形处理器(graphics processing unit, GPU)技术飞速发展,CUDA(compute unified device architecture)[16]为 GPU 编程提供了高效的并行计算平台,极大降低了基于 GPU 的编程难度。这些技术的发展为大规模 DSMC 在 GPU 上并行提供了新的可能,对 DSMC 方法进行大规模的高效并行处理已经是大势所趋。

已有一些工作将 DSMC 方法在异构平台进行并行加速,陈伯君[17]首次在 GPU 上以一维 Rayleigh 问题为例对 DSMC 方法进行了并行计算。严立等[18]在异构系统架构下基于 CUDA 对结构网格下 DSMC 方法的粒子运动模块进行了并行化改造和优化,并行算法的粒子运动模块的加速比随着计算规模的增大而增加,获得了 4～9 倍的加速效果。邱天林[19]使用 CUDA Fortran 在 GPU 上进行了直角坐标网格 DSMC 方法的并行计算,取得了 7.7 倍的加速比。然而他们的工作均未涉及非结构网格 DSMC 算法,且取得的加速比有限,并未充分发挥出 CUDA 设备的并行加速潜能。

本文旨在充分发挥非结构网格对复杂外形的高度贴体性[20],便于局部自适应细化,实现自动化网格生成的优势,基于 CUDA Fortran 设计一种面向 GPU 的非结构网格 DSMC 并行算法。针对 CUDA 的单指令多线程(single instruction multiple threads, SIMT)架构执行模式的特性,并结合 Amdahl 定律[21]对 DSMC 算法各模块的加速潜力进行分析,识别出适合 GPU 加速的核心计算模块,并进行并行实现。对于算法中分支密度高,不适合直接 GPU 并行的模块,采用标量扩展预处理与循环解耦分步法,分解出这些模块中适合进行 GPU 并行化的计算步骤并进行并行实现,以减少线程束分化(warp divergence)带来的性能损耗,实现高效的并行计算。最终通过典型算例进行验证。

1 非结构网格 DSMC 方法

1.1 DSMC 方法

DSMC 方法是用于模拟稀薄气体动力学的数值方法,对于稀薄流区,通过随机取样粒子运动和碰撞,从粒子动力学角度将粒子的碰撞和运动解耦,对非结构网格内的微观粒子取样统计来表征宏观物理特性。核心步骤为:①初始化模块:根据来流条件对计算域内的网格进行初始参数的

分配，包括网格内的粒子数、粒子的速度、网格初始温度等信息。②粒子运动模块：主要考虑粒子运动前后的位置、速度及能量的变化、粒子与物面的相互作用，溢出计算域粒子的删除。③粒子排序模块：基于网格单元内粒子数量重新对粒子排序与编号。④粒子碰撞模块：以网格为单位，依次考虑每个网格内的每组取样碰撞对，计算碰撞前后碰撞对能量分配的问题。⑤取样模块：经过一个时间步长的粒子运动和碰撞后，以网格为单位统计网格中粒子的信息，对网格中粒子的能量信息进行累加。⑥重复步骤②～步骤④，经过一定的时间步长后对流场信息进行取样平均，输出流场宏观特性。

本文 DSMC 中的粒子碰撞模型采用变径硬球模型[2]；碰撞对的选取采用非时间计数器法[2]；利用漫反射模型粒子与物面碰撞[2]；能量交换采用 L-B（Larsen-Borgnakke）模型[13]。

1.2　非结构网格粒子搜索方法

非结构网格中粒子位置搜索困难是因为非结构网格节点分布无序，无法像直角坐标网格那样直接根据粒子的位置坐标确定粒子与网格的从属关系，对于粒子在非结构网格单元间的运动追踪采用面积元搜索方法[22]，如图 1 所示，其核心思想是：将非结构网格的节点都按逆时针方向进行编号，$\triangle ABC$ 的面积公式为

$$S_{\triangle ABC}=\begin{vmatrix}1 & x_1 & y_1\\ 1 & x_2 & y_2\\ 1 & x_3 & y_3\end{vmatrix} \tag{1}$$

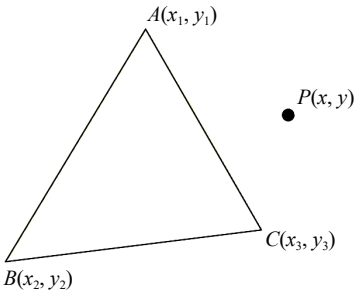


图 1　网格搜索示意图

Fig. 1　Schematic diagram of grid search

类似地，按式（1）分别计算 $\triangle ABP$、$\triangle BCP$、$\triangle CAP$ 的面积，若得到的面积元都为正值，则粒子 P 在网格 $\triangle ABC$ 内；而若粒子 P 在 $\triangle ABC$ 外，则得到的计算结果中至少有一个值为负。当判定目标粒子在一网格外部时，下一步要做的就是将搜索范围扩展到相邻网格，重复此操作，直至找到粒子所在的网格。

2　CUDA 和线程束分化

2.1　统一计算设备架构 CUDA

2006 年底，NVIDIA 推出了针对通用计算 GPU 的一个全新软硬件平台——统一计算设备架构（CUDA），旨在将高性能并行处理能力从图形渲染领域扩展到科学计算、工程仿真和数据分析等广泛的应用领域，从而显著提升应用程序的计算速度和能效，它使开发者能够利用图形处理器（GPU）的大量并行处理单元进行通用计算。

一个标准的 CUDA 程序执行流程如图 2 所示，通常由以下两个主要部分构成，主机代码（host code）和设备代码（device code），当主机代码调用 CUDA 核函数后，执行流程通常如下：

1）数据准备与传输：主机端首先初始化所有输入数据，然后通过 CUDA 内存拷贝函数（CUDA memory copy function，记为 CudaMemcpy）将这些数据从主机内存传输到设备内存。

2）核函数启动与并行计算：主机代码通过 CUDA 核函数启动语法（CUDA kernel launch syntax，记为 kernel<<<gridDim,blockDim>>>）的指令调用启动核函数。CUDA 会按照指定的网格和线程块配置，在 GPU 上分派大量线程执行核函数。每个线程按照其计算得到的全局索引独立地操作对应的数据。

3）同步与数据回传：当核函数执行结束后，主机代码使用 CUDA 设备同步函数（CUDA device synchronization function，记为 CudaDeviceSynchronize）等同步函数确保所有 GPU 线程完成计算。然后，利用 CudaMemcpy 将输出结果从设备内存传回主机内存以供后续处理或显示。

2.2　线程束分化

与 CPU 的多指令流多数据流（multiple instruction multiple data，MIMD）[23] 不同，CUDA 的基本架构基于 SIMT 执行模式，每 32 个线程组成一个线程束（warp），数以千计的线程以线程束为单位在 GPU 上同时执行相同的指令流。这种大规模并行性使得 CUDA 在处理数据并行问题（例如离散模拟、矩阵运算和图像处理等）方面具有显著优势，但在这种架构下，如果分支条件在同一线程束中不同线程上表现不一致，则会发生线程束分化。线程束分化问题使得一个线程束需要顺序地执行不同的分支路径，导致内存的非线性

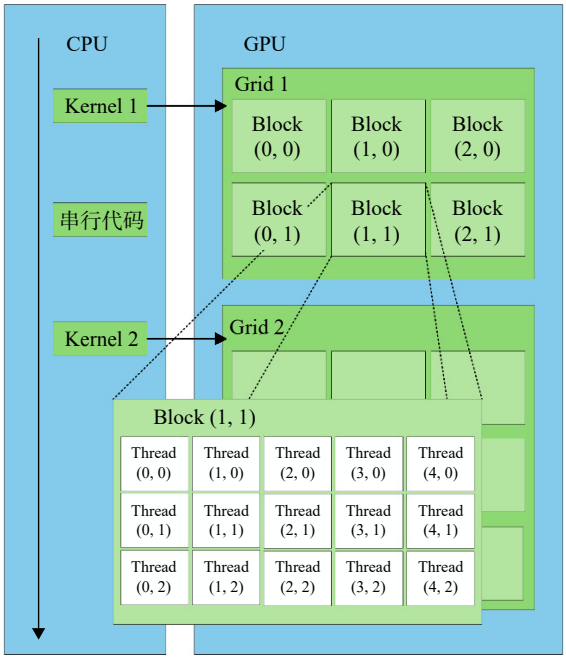


图 2 GPU 编程模型

Fig. 2 GPU programing model

访问和线程等待，从而大幅降低并行计算的效率，尤其在内核函数中存在大量逻辑判断分支时更为明显，会严重降低 GPU 的计算效率[24-25]。较高的分支分化率会显著降低 GPU 的吞吐量，在 CUDA 的 SIMT 执行模式下线程束分化是导致并行优化效率降低乃至优化失效的主要因素。

3 基于 CUDA 的非结构网格 DSMC 算法的实现

3.1 非结构网格DSMC算法各模块加速潜力分析

Amdahl 定律是并行计算领域用来评估程序加速理论上限的一个经典公式

$$S(N)=\frac{1}{(1-Q)+\frac{Q}{N}} \tag{2}$$

其中$S(N)$为理论加速比，Q为算法中可并行执行部分的比例，N为可用的并行处理单元数量(与计算设备硬件相关，对于 CUDA 设备而言可认为N等于 CUDA 核心数量)。根据该公式，算法中可并行执行部分的比例越高可获得的加速比越高。经统计，非结构网格 DSMC 算法各模块的可并行部分比例如表 1 所示。

DSMC 算法模拟过程中每个网格或者每个粒子的行为是独立的，单个迭代步内不存在网格间或者粒子间的数据依赖，因此 DSMC 方法具有较强的数据并行性。其中初始化、取样和粒子排序模块的分支行为相对较少，在并行实现后一个线程束内的大多数线程可以沿着相同的执行路径运行，在 CUDA 的 SIMT 执行模式下并行时几乎不会引发严重的线程束分化，能够充分发挥 GPU 并行计算的优势，因而可并行部分的比例较高。

表 1 各模块可并行部分比例

Table 1 Proportion of parallelizable parts for each module

模块名称	比例 Q/%	核心可并行部分	
		子模块名称	子模块功能
初始化	92.6	Volume	网格面积计算
		Subcell	子网格编号计算
		Reassign	取样赋值初始化
粒子运动	23.6	Move	粒子坐标计算
粒子排序	94.1	Index	粒子排序
粒子碰撞	10.1	Colnsel	碰撞对数目计算
取样	100	Sample	网格信息取样

粒子运动模块和粒子碰撞模块则存在大量的随机分支行为。其中粒子运动模块需要判断粒子是否溢出计算域、是否与物面作用、计算反射粒子速度和能量、搜索运动后粒子网格信息。粒子碰撞模块以网格为单位，根据网格内粒子的数量确定碰撞对的数目，然后随机选择两个粒子组成碰撞对，判断是否满足碰撞条件，如果满足，则根据碰撞模型进行碰撞处理。这两个模块中大量存在的基于随机数的逻辑判断或者跳转操作，导致每个粒子行为不一致，如果直接对这两个模块进行 GPU 并行，这些随机操作会导致在 GPU 并行时各个线程在相同时间内沿着不同的代码路径执行，引发线程束分化和内存的非线性访问问题，严重降低并行算法的执行效率，甚至导致了计算性能下降，因此在 CUDA 的 SIMT 执行模式下这些存在大量随机条件判断以及跳转操作的计算不能作为可并行执行部分。

可并行部分比例值量化了这些模块代码路径一致性，即代码的分支密度的高低，将 DSMC 算法模块中可并行部分比例高的模块作为低分支密度模块，包括初始化、取样和粒子排序模块，相对应高分支密度模块为粒子运动模块和粒子碰撞模块。针对低分支密度模块中不同计算场景，设计网格并行策略和粒子并行策略进行并行实现。针对高分支密度模块，采用标量扩展预处理和循环解耦分步法进行计算分解，在此基础上通过网格并行策略和粒子并行策略并行实现。

3.2　低分支密度模块并行实现

DSMC 过程中初始化、取样和粒子排序模块的分支密度较低，这类模块可直接将原本的顺序循环计算映射到 GPU 上，根据不同计算场景设计了①网格并行策略：每个线程处理一个网格的计算行为；②粒子并行策略：每个线程处理一个粒子的计算行为。

对于初始化模块中每个网格单元的面积计算、每个网格单元初始粒子数的生成，不同网格单元之间互不依赖，天然具备高度并行性。这些场景的并行实现策略为将串行计算中顺序循环 I 个网格的顺序计算映射到 GPU 的 I 个线程进行并行计算。

以初始化模块中网格面积计算子模块 Volume 的并行算法为例，通过建立线程全局索引序号与串行算法的顺序循环操作的循环次数 I 的映射关系，将原本循环 I 次的计算操作映射到 GPU 中 I 个线程并行执行。并行化后 Grid(线程网格)、Block(线程块)、Thread(线程)与网格的映射关系示意图见图 3。

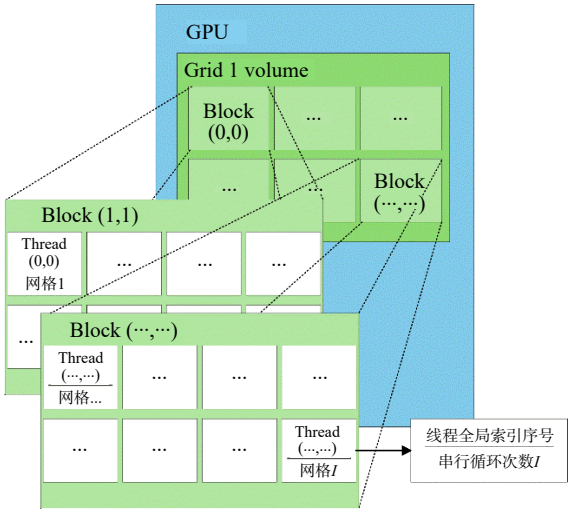


图 3　线程与网格映射关系图

Fig. 3　Thread and grid allusion relationship diagram

粒子排序模块的核心操作是每个粒子在对应网格中累加得到网格内总粒子数和为每个粒子在所属网格中分配编号。以每个粒子在对应网格中累加得到网格内总粒子数 N 为例，该场景的并行实现策略为通过将串行算法的顺序循环操作的循环次数 N 与并行实现时的线程全局索引序号建立映射关系，将原本循环 N 次的计算操作映射到 N 个线程并行执行，每个线程独立处理一个粒子在对应网格内数量的累加计算。对于累加操作在并行实现时造成的线程间数据冲突，利用原子累加函数 ATOMICADD 实现，确保对同一内存地址的操作按顺序执行，防止线程之间的数据冲突。

在原始串行算法中，取样模块的主要流程为：外层循环遍历所有网格单元，内层循环遍历每个网格内的所有粒子，并通过累加粒子的速度、转动能等微观信息以获得网格对应的宏观物理量，其中每个粒子的贡献互不依赖，因此整体上仍具有较高的并行性。若直接采用网格并行策略——每个线程处理一个网格的计算行为来粗粒度地并行外层循环，由于各网格内粒子数量差异较大，将导致不同线程间的工作负载不均衡，进而引发线程束分化，降低并行效率。该模块计算的本质为对所有粒子的遍历与统计，因此可采用粒子并行策略——每个线程处理一个粒子的计算行为，跳过外层循环直接遍历所有粒子，实现粒子级别的高效细粒度并行。具体而言，每个线程首先确定自身粒子所属网格，然后将该粒子的微观物理量贡献累加到对应的网格单元中。通过细粒度的并行有效避免了线程束分化问题，更充分地利用了 GPU 的并行计算能力，从而提高了模拟效率。

3.3　高分支密度模块并行实现

粒子碰撞、粒子运动模块的分支密度较高，该计算模块的并行化采用标量扩展预处理和循环解耦分步法，核心思路为：将原有高分支密度循环计算任务分解为若干 CPU 计算步骤和若干个 GPU 计算步骤。具体操作方法为：首先在预处理阶段进行标量扩展，将原串行算法循环计算内原为标量的变量扩展为数组，使其循环过程中的计算结果独立存储，为循环解耦分步和并行化创造必要条件，在此基础上通过扩展的中间数组完成循环的解耦分步——将单一复杂循环计算拆分为多个独立简单循环计算，拆分后的每个循环仅处理原循环中的部分计算，最终将拆分后的多步循环计算中分支密度较低的若干步骤进行并行化，从而避免高分支密度计算阶段造成的严重并行加速损失。

以粒子运动模块中粒子运动及网格搜索算法为例，原串行算法流程如图 4 所示。

算法存在两个紧密耦合的部分包括粒子运动前后坐标计算和基于坐标信息的运动后粒子处理及所在网格的搜索，其中粒子初始坐标 X_i、Y_i，位移 D_X、D_Y，终点坐标 X、Y 均为标量，每次串行循环仅覆盖旧值，循环计算之间这些值之间不存在数据依赖，通过标量扩展将循环内原为标量数据扩展为数组 $X_i(N)$、$Y_i(N)$、$D_X(N)$、$D_Y(N)$、

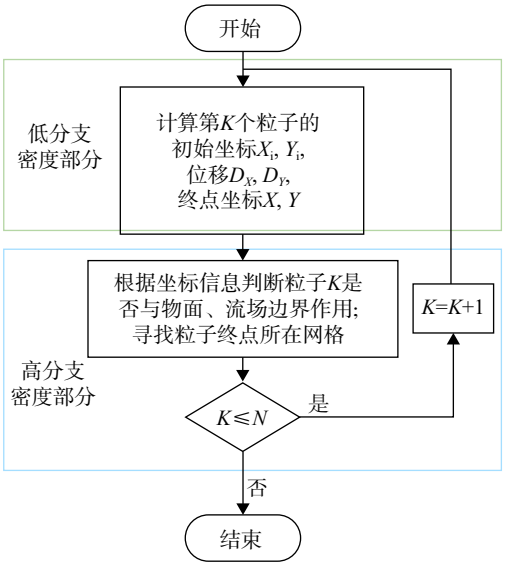


图 4 粒子运动模块串行算法流程图

Fig. 4 Flowchart of serial algorithm for molecular motion module

$X(N)$、$Y(N)$，其中 N 为粒子总数，使原本每个循环中的粒子的坐标结果独立存储。在此基础上进行循环分步操作，将粒子运动后坐标计算与运动后粒子处理及其所在网格的搜索解耦分步，解耦分步后其中分支密度较低的粒子运动前后坐标计算部分按照前述粒子并行策略并行——启动 N 个线程进行并行计算。后续高分支密度部分——运动后粒子处理及所在网格的搜索，则基于扩展的中间数组保留在 CPU 端完成后续计算。最终得到的算法流程如图 5 所示。

粒子碰撞模块的优化方法和粒子运动模块相

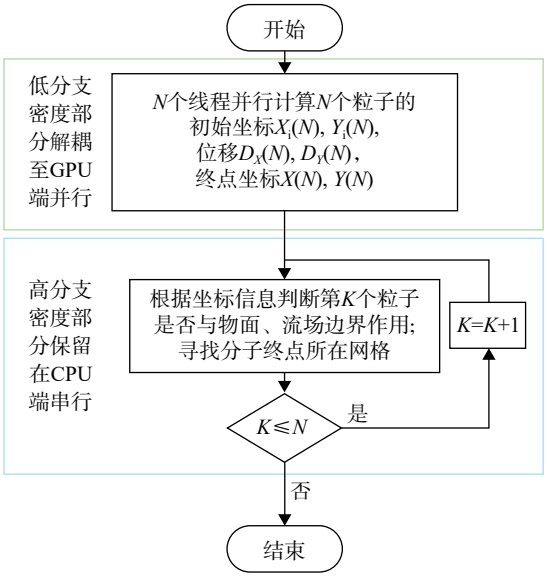


图 5 粒子运动模块 GPU 并行算法流程图

Fig. 5 Molecular motion module GPU parallel algorithm flowchart

同，通过标量扩展将循环内原为标量的每个网格内选取的碰撞对数目 S 扩展为数组 $S(I)$，其中 I 为网格数，在此基础上进行循环解耦分步，最终将其中分支密度较低的每个网格碰撞对数量的计算按照网格并行策略并行实现——启动 I 个线程，每个线程计算对应网格产生的碰撞对个数。后续随机选择两个粒子组成碰撞对、判断是否满足碰撞条件和能量分配等计算操作基于扩展的中间数组 $S(I)$ 完成后续计算。

3.4 数据传输优化

本研究的 GPU 并行实现主要集中在 DSMC 的低分支密度计算部分。然而，由于整个 DSMC 流程中还存在分支密度较大的流程，以及数据读取、写入等存在逻辑顺序依赖的流程，这些部分无法充分并行化，从而不可避免地需要在 CPU 与 GPU 之间进行数据传输。在基于 CUDA 的 GPU 并行计算中，数据传输通常是影响整体性能的关键瓶颈之一。为降低数据传输对总体性能的影响，采用 GPU 端数据常驻方法：网格信息、节点信息以及流场参数等数据通常在整个计算过程中保持不变，将这些数据在算法初始化阶段一次性从主机内存传输到 GPU 全局内存，使这些数据在 CPU 和 GPU 端分别常驻，在后续所有核函数调用中减少了频繁的数据复制操作，以此利用 GPU 的内存架构来提高数据访问的效率。

3.5 线程越界保护与动态线程分配策略

CUDA 的 SIMT 执行模式以线程束（32 线程）为最小执行单元，每次启动的线程数总是 32 的整数倍。当实际所需线程数并非 32 的倍数时，多余的线程会启动但不执行有效计算，若不加以判断，可能导致越界访问。为此，在每个核函数开始阶段即加入线程索引越界检测：当索引 ID 超出有效范围时，线程立即退出，既保证了数据安全，也避免了额外的无效指令执行。

DSMC 运行过程中，计算规模是不断变化的，粒子数 N 也会不断变化，如果每次都固定按最大可能线程数启动，会造成资源浪费。为提高调度效率，在每一次核函数调用之前，根据总粒子数的值决定启动的线程数，以避免线程束过多引起的计算资源的浪费，提升了 GPU 资源利用率并降低了启动开销。

4 算例验证与结果分析

4.1 并行算法正确性验证

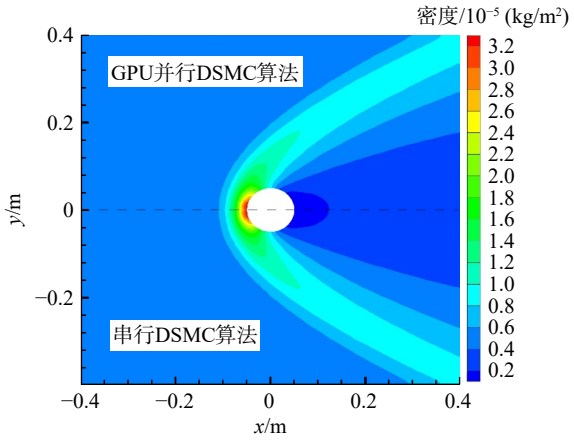
以超声速圆柱绕流典型验证算例为例，验证

基于 GPU 的非结构网格 DSMC 并行算法的正确性。参考算例来自文献 [14], 圆柱半径为 0.05 mm, 来流气体为氩气, 温度为 300 K, 壁面温度为 500 K, 气体粒子数密度为 1.0×10^{20} m^{-3}, 来流速度为 1 412.5 m/s, 马赫数为 4.4。

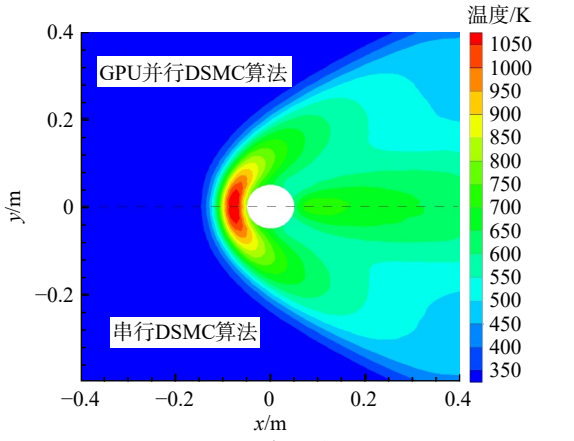
分别对串行非结构网格 DSMC 算法和并行 DSMC 算法进行模拟, 图 6 给出串行和并行算法密度云图和温度云图, 可以看出在圆柱上游形成一道脱体弓形激波, 激波前后气体密度和温度显著跃升, 最高温度出现在激波与圆柱驻点之间的压缩区, 圆柱侧缘气流膨胀温度密度骤降, 圆柱下游形成低压区, 密度低于自由来流。两种算法所得各物理量的结果一致。

图 7 给出串行算法和并行算法气流沿驻点线的密度分布, 并和文献 [14] 中数据进行对比分析, 两种算法的模拟结果与文献给出的结果曲线图趋势高度吻合, 密度和温度的峰值均出现在激波后的压缩区, 验证了 GPU 并行算法的准确性。

图 8 对两种算法壁面特征量与文献 [14] 进行



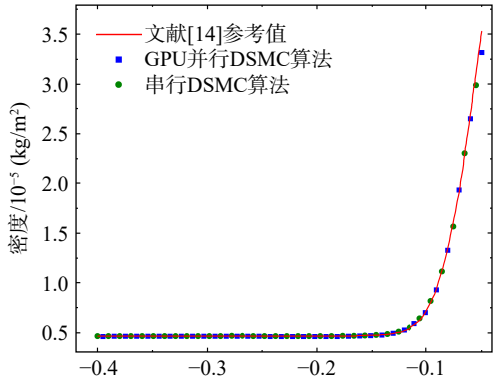
(a) 密度云图



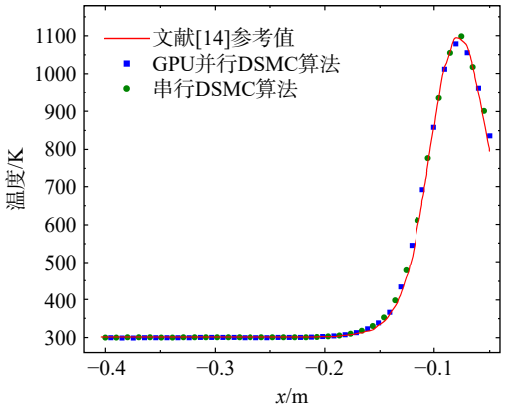
(b) 温度云图

图 6　串行算法并行算法密度和温度比较

Fig. 6　Comparison of density and temperature between Serial and Parallel Algorithms



(a) 沿驻点线密度分布

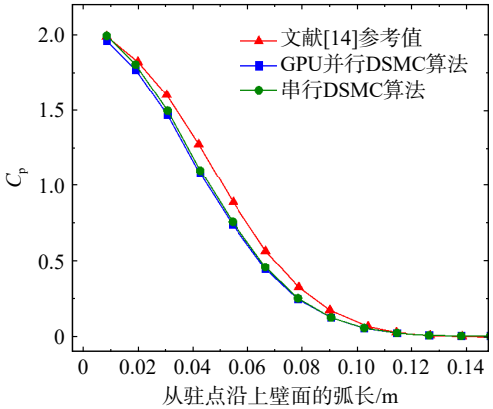


(b) 沿驻点线温度分布

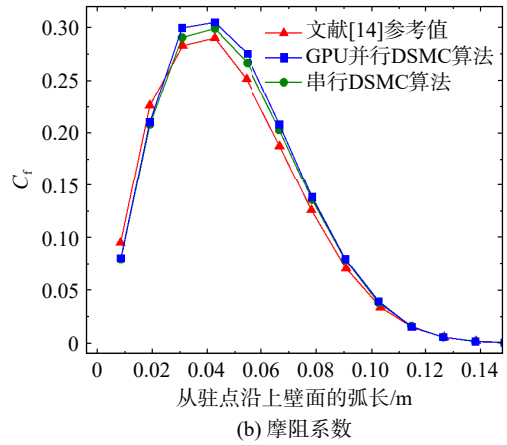
图 7　沿驻点线密度和温度比较

Fig. 7　Comparison of density and temperature along the stagnation line

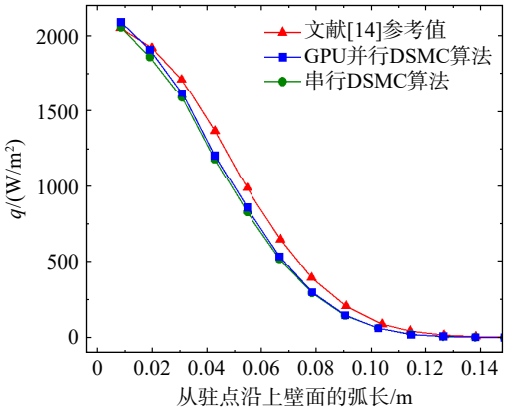
对比。压力系数 C_p 在驻点达到最大值, 随后在圆柱侧缘因急剧下降, 膨胀区进一步减小至趋于零。摩阻系数 C_f 在驻点至加速区快速增大, 膨胀区进一步减小至趋于零。热流通量 q 在驻点处因激波压缩和滞止效应出现峰值, 随后沿加速区逐渐降低, 膨胀区因低温气流附着减弱而进一步减小至趋于零。算法模拟结果与文献 [14] 参考结果高度符合, 进一步验证了 GPU 并行算法的准确性。



(a) 压力系数



(b) 摩阻系数



(c) 热流通量

图 8 圆柱壁面特征值比较

Fig. 8 Comparison of wall characteristics for cylindrical walls

4.2 并行算法性能分析

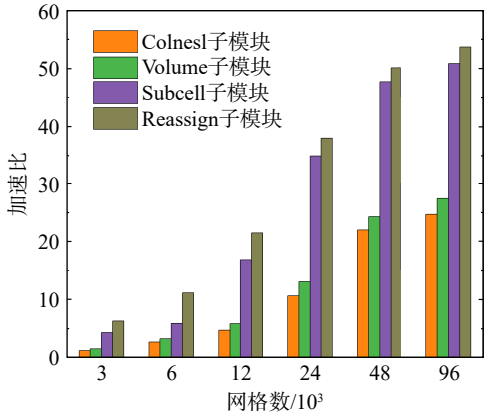
在流场稳定后，统计串行算法和并行算法各个优化模块每 100 步的平均计算时长，计算得到各优化模块的加速比数据，以验证并行算法的高效性。本算例所用测试设备环境见表 2。

图 9(a)展示了基于网格并行策略的主要优化部分在不同网格数算例下的加速比，图 9(b)则给出了粒子并行策略下主要优化部分在不同粒子数算例下的加速比。可以看出，两种并行策略的加速比随处理数据量的增加均呈现出先快速提升、随后趋于平稳的典型趋势。在小规模数据下，内核启动与数据传输开销占比高，导致加速比偏低；随着数据规模增大，计算占比逐渐提升，GPU 并行效率显著增强，使加速比快速上升；当数据量进一步增大至足以饱和 GPU 的计算能力后，加速比逐渐接近算法和硬件的理论上限。在大规模算例中，各优化部分均获得了 25～70 倍的加速效果。图 9(c)展示了程序迭代过程中的所有加速模块

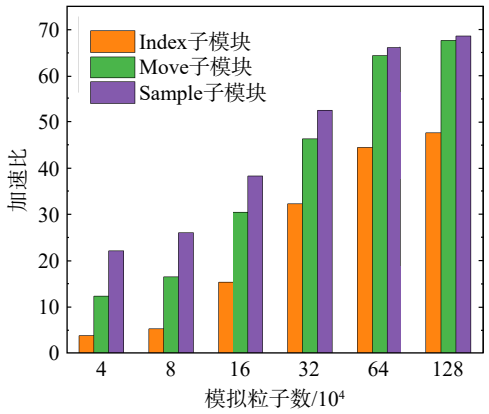
表 2 算法测试环境

Table 2 Algorithm testing environment

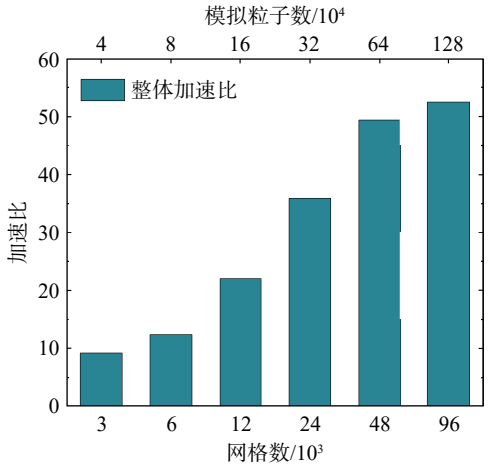
类别	名称	配置
硬件环境	CPU	Intel(R) Core(TM) i7-14700KF
	GPU	NVIDIA GeForce RTX 4070 SUPER
	内存	48 G DDR5
软件环境	操作系统	Ubuntu 22.04
	编译器	NVFORTRAN 25.3



(a) 基于网格并行策略各模块加速比



(b) 基于粒子并行策略各模块加速比



(c) 各并行模块加速随计算规模的变化

图 9 各子模块及并行模块整体加速比随计算规模的变化

Fig. 9 Changes in acceleration of parallel modules and overall with computing scale

整体(剔除仅在初始化过程中执行的 Volume 和 Subcellz 子模块)在不同算例下的加速比，获得了 9～53 倍的加速效果，充分验证了所提出并行算法的高效性。

在网格并行策略下的大规模计算中，不同模块的加速效果存在差异。碰撞模块的 Colnsel 子模块中存在随机分支分化，初始化模块中的 Volume 子模块计算密度较低且以简单的数据搬运操作为主，因而仅获得约 25 倍的最低加速比；相比之下，初始化中的 Subcell 子模块和 Reassign 子模块的分支条件规律可预测，且具有较高的计算密度，从而实现了约 50 倍的最高加速比。

在粒子并行策略下的大规模计算中，粒子排序模块的 Index 子模块因引入原子操作导致线程竞争，最终仅获得约 45 倍的最低加速比；而取样模块中 Sample 子模块虽仍包含原子操作，但通过算法重构将其由网格并行策略优化为粒子并行策略，实现了更细粒度的并行度，从而取得约 68 倍的最高加速比。

5　结论和展望

本研究在 CPU+GPU 异构平台上，对原有串行 DSMC 算法进行了深入分析和基于 GPU 的并行实现。首先，通过对各模块可并行执行部分比例的计算量化了算法各模块的分支密度，识别出最适合在 GPU 上加速的计算部分，并基于 CUDA Fortran 对其实现了高效并行化；对于分支密度较高、不宜直接并行的模块，进一步通过循环解耦分步法将其中的低分支密度子任务分解提取，并在 GPU 上并行执行。

实验结果表明，无论是直接并行化的计算模块，还是分解后并行化的计算模块，都获得了显著的加速效果，且随着网格数目的增加，粒子数目的增加，加速比呈现出先快速提升、随后趋于平稳的典型趋势，充分验证了所开发算法对 GPU 并行能力的高效利用。

下一步研究将针对当前仍保留在 CPU 端执行的高分支密度模块，探索基于 MPI 的分布式并行优化方案，并最终构建 MPI+CUDA 混合异构并行算法框架，以期在更大规模的算例和更多节点的计算环境中，实现更高效、更可扩展的 DSMC 模拟。

参考文献：

[1] RADTKE J, KEBSCHULL C, STOLL E. Interactions of the space debris environment with mega constellations: using the example of the OneWeb constellation[J]. Acta Astronautica, 2017, 131: 55-68.

[2] BIRD G A. Molecular gas dynamics and the direct simulation of gas flows[M]. Oxford: Oxford University Press, 1994.

[3] 许啸，王学德，谭俊杰. 近空间高超声速流场通量分裂型 DSMC-IP 方法[J]. 航空动力学报，2015, 30(2): 315-323.
XU Xiao, WANG Xuede, TAN Junjie. Flux splitting DSMC-IP method in near space hypersonic flow field[J]. Journal of Aerospace Power, 2015, 30(2): 315-323. (in Chinese)

[4] 王保国，黄伟光，钱耕，等. 再入飞行中 DSMC 与 Navier-Stokes 两种模型的计算与分析[J]. 航空动力学报，2011, 26(5): 961-976.
WANG Baoguo, HUANG Weiguang, QIAN Geng, et al. Computation and analysis of DSMC and Navier-Stokes model in reentry flight[J]. Journal of Aerospace Power, 2011, 26(5): 961-976. (in Chinese)

[5] 朱荣丽，曹义华，李栋，等. 高超声速飞行器复杂流场过渡区 DSMC 数值模拟的一种新方案[J]. 宇航学报，2006, 27(2): 167-171.
ZHU Rongli, CAO Yihua, LI Dong, et al. A new version of hypersonic vehicle numerical simulation in direct simulation of Monte Carlo method of three dimensional flow in transitional regime[J]. Journal of Astronautics, 2006, 27(2): 167-171. (in Chinese)

[6] 李中华，李志辉，李海燕，等. 过渡流区 N-S/DSMC 耦合计算研究[J]. 空气动力学学报，2013, 31(3): 282-287.
LI Zhonghua, LI Zhihui, LI Haiyan, et al. Study on N-S/DSMC coupling calculation in transition flow region[J]. Acta Aerodynamica Sinica, 2013, 31(3): 282-287. (in Chinese)

[7] 王保国，李学东，刘淑艳. 高温高速稀薄流的 DSMC 算法与流场传热分析[J]. 航空动力学报，2010, 25(6): 1203-1220.
WANG Baoguo, LI Xuedong, LIU Shuyan. DSMC algorithm and heat transfer analysis of high temperature and high velocity rarefied gas flow[J]. Journal of Aerospace Power, 2010, 25(6): 1203-1220. (in Chinese)

[8] 梁杰，阎超，李志辉，等. 稀薄过渡流区横向喷流干扰效应数值模拟研究[J]. 空气动力学学报，2013, 31(1): 27-33.
LIANG Jie, YAN Chao, LI Zhihui, et al. Numerical investigation of lateral jet interaction effects in rarefied transition flow regime[J]. Acta Aerodynamica Sinica, 2013, 31(1): 27-33. (in Chinese)

[9] 王保国，李耀华，钱耕. 四种飞行器绕流的三维 DSMC 计算与传热分析[J]. 航空动力学报，2011, 26(1): 1-20.
WANG Baoguo, LI Yaohua, QIAN Geng. Three-dimensional DSMC calculation and heat transfer analysis of four capsules for hypersonic rarefied conditions[J]. Journal of Aerospace Power, 2011, 26(1): 1-20. (in Chinese)

[10] 蔡国飙，刘世俭，王慧玉，等. 真空羽流场的 DSMC 并行数值模拟[J]. 航空动力学报，1999, 14(2): 113-118.
CAI Guobiao, LIU Shijian, WANG Huiyu, et al. DSMC parallel numerical simulation of vacuum plume flow field[J]. Journal of Aerospace Power, 1999, 14(2): 113-118. (in Chinese)

[11] 杜松蔚，王学德. 一种基于双重空间网格结合的高效 DSMC 实现方法[J]. 航空动力学报，2022, 37(9): 1846-1854.
DU Songwei, WANG Xuede. High-efficiency DSMC implement method based on combination of dual spatial grids[J]. Journal of Aerospace Power, 2022, 37(9): 1846-1854. (in Chinese)

[12] 张哲铭. 非结构网格 DSMC 的大规模并行计算研究[D]. 成都：电子科技大学，2022.
ZHANG Zheming. Research on large-scale parallel computing of unstructured grid DSMC applications[D]. Chengdu: University of

Electronic Science and Technology of China, 2022. (in Chinese)

[13] LUSK E, HUSS S, SAPHIR B, et al. MPI: a message-passing interface standard[J]. International Journal of Supercomputer Applications, 2009, 8(3/4): 623.

[14] 王学德. 高超声速稀薄气流非结构网格 DSMC 及并行算法研究[D]. 南京: 南京航空航天大学, 2006.
WANG Xuede. DSMC method on unstructured grids for hypersonic rarefied gas flow and its parallelization[D]. Nanjing: Nanjing University of Aeronautics and Astronautics, 2006. (in Chinese)

[15] PassMark Software. Top CPU performance to date [EB/OL]. (2025-06-28)[2025-07-06]. https://www.cpubenchmark.net/year-on-year.html.

[16] NVIDIA. NVIDIA CUDA toolkit documentation [EB/OL]. (2025-06-09) [2025-07-06]. https://developer.nvidia.com/cuda-faq.

[17] CHEN Bojun. High performance parallel computing of DSMC method based on CUDA[EB/OL]. (2009-12-21) [2025-07-06]. https://wenku.baidu.com/view/f8f381ccbb4cf7ec4afed0cd.html?_wkts_=1763396744283

[18] 严立, 戴欣怡, 陈佳洛, 等. 基于计算统一设备架物 Fortran 的直接模拟蒙特卡洛方法并行优化[J]. 上海交通大学学报, 2013, 47(8): 1198-1204.
YAN Li, DAI Xinyi, CHEN Jialuo, et al. Parallel optimization of direct simulation Monte Carlo method using compute unified device architecture fortran[J]. Journal of Shanghai Jiao Tong University, 2013, 47(8): 1198-1204. (in Chinese)

[19] 邱天林. 直角坐标网格下 DSMC 方法的 GPU 并行研究[D]. 南京: 南京航空航天大学, 2017.
QIU Tianlin. Parallel research on GPU-based DSMC method using Cartesian grid[D]. Nanjing: Nanjing University of Aeronautics and Astronautics, 2017. (in Chinese)

[20] 陈飞同, 王学德. 三维复杂界面非结构网格 N-S/DSMC 耦合方法[J]. 航空动力学报, 2025, 40(9): 20240329.
CHEN Feitong, WANG Xuede. N-S/DSMC coupling method using three-dimensional unstructured mesh for complex interfaces[J]. Journal of Aerospace Power, 2025, 40(9): 20240329. (in Chinese)

[21] AMDAHL G M. Validity of the single processor approach to achieving large scale computing capabilities[C]//Proceedings of the Spring Joint Computer Conference. New York: Association for Computing Machinery, 1967: 483-485.

[22] 王学德, 伍贻兆, 夏健, 等. 三维非结构网格 DSMC 并行算法及应用研究[J]. 宇航学报, 2007, 28(6): 1500-1505.
WANG Xuede, WU Yizhao, XIA Jian, et al. A parallel algorithm of 3D unstructured DSMC method and its application[J]. Journal of Astronautics, 2007, 28(6): 1500-1505. (in Chinese)

[23] 白洪涛. 基于 GPU 的高性能并行算法研究[D]. 长春: 吉林大学, 2010.
BAI Hongtao. Research on high performance parallel algorithms based on GPU[D]. Changchun: Jilin University, 2010. (in Chinese)

[24] 傅萌. 基于 CPU+GPU 并行计算加速的电力系统状态估计技术[D]. 南京: 东南大学, 2023.
FU Meng. Power system state estimation technology based on CPU+GPU parallel computing acceleration[D]. Nanjing: Southeast University, 2023. (in Chinese)

[25] 鞠鹏飞, 宁方飞. GPU 平台上的叶轮机械 CFD 加速计算[J]. 航空动力学报, 2014, 29(5): 1154-1162.
JU Pengfei, NING Fangfei. Accelerated CFD computing of turbomachinery on GPU platform[J]. Journal of Aerospace Power, 2014, 29(5): 1154-1162. (in Chinese)

（编辑：王碧珺）