ISSN 2096-742X
CN 10-1649/TP

文献DOI：
10.11871/jfdc.issn.
2096-742X.2020.
01.009

文献PID：
21.86101.2/jfdc.
2096-742X.2020.
01.009

页码：105-116

开放科学标识码
（OSID）

基于CPU/GPU异构系统架构的高超声速湍流直接数值模拟研究

党冠麟[1,4]，刘世伟[3,4]，胡晓东[2]，张鉴[2]，李新亮[1,4]*

1. 中国科学院力学研究所，北京 100190
2. 中国科学院计算机网络信息中心，北京 100190
3. 中国科学院数学与系统科学研究院，北京 1 00190
4. 中国科学院大学，北京 100049

摘　要：【目的】高超声速湍流直接数值模拟（DNS）对空间及时间分辨率要求高，计算量非常大。过大的计算量及过长的计算时间是导致DNS难以在工程中被大范围应用的重要原因。为加快计算速度，作者设计并开发了一套CPU/GPU异构系统架构（HSA）下的高性能计算流体力学程序OpenCFD-SCU。【方法】该程序以作者前期开发的高精度有限差分求解器OpenCFD-SC为基础，经GPU系统的移植及优化而得。GPU程序的计算部分使用CUDA编程，确保所有算术运算都在GPU上完成。【结果】利用GPU程序OpenCFD-SCU，进行了来流Mach数6，6°攻角钝锥边界层转捩的直接数值模拟，得到了转捩过程中的时空演化流场。针对这一算例，GPU程序OpenCFD-SCU与CPU程序OpenCFD-SC相比，实现了60倍的加速效果（单GPU卡对单CPU核心），大大加速了DNS计算过程。【结论】未来，相信会有更多高超声速湍流模拟选择在GPU上开展。

关键词：高超声速湍流；直接数值模拟；GPU

Direct Numerical Simulation of Hypersonic Turbulence Based on CPU/GPU Heterogeneous System Architecture

Dang Guanlin[1,4], Liu Shiwei[3,4], Hu Xiaodong[2], Zhang Jian[2], Li Xinliang[1,4]*

1. Institute of Mechanics,Chinese Academy of Sciences, Beijing 100190, China
2. Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China
3. Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China
4. University of Chinese Academy of Sciences, Beijing 100049, China

Abstract: [**Objective**] The direct numerical simulation (DNS) of hypersonic turbulence requires great many grids points and time steps. Therefore, the amount of calculation is very large. Excessively long time of calculation is an important reason that DNS cannot be applied in real applications. In

基金项目：中国科学院战略先导科技专项（XDC01040100）；国家重点研发项目 (2016YFA0405300)
*通讯作者: 李新亮 (E-mail:lixl@imech.ac.cn)

order to accelerate the calculation, design of a high-performance computational fluid mechanics program OpenCFD-SCU under the CPU/GPU Heterogeneous System Architecture (HSA) is introduced in this paper. [**Method**] This program is based on the CPU [fortran] code OpenCFD-SC which is a high-precision finite difference solver developed by the authors. OpenCFD-SCU has the same program framework as OpenCFD-SC, and the computing part of the GPU program is programmed by CUDA to ensure that all arithmetic operations are completed on the GPU. [**Results**] In a same DNS task, the GPU version of OpenCFD-SCU is 60 times faster than the CPU version of OpenCFD-SC. The computing power of GPU is much higher than that of CPU. Using GPU can effectively accelerate the calculation, which is the future trend of DNS programs for hypersonic turbulence. [**Conclusion**] In the future, we believe that more and more hypersonic turbulence simulation can be moved to the GPU.

Keywords: hypersonic turbulence; DNS; GPU

引言

湍流是常见的自然现象，自然界的流动多以湍流形式呈现。人们对湍流的系统性研究始于上世纪 20 年代。经过百年的发展，现今人们对一些湍流现象已经可以较好地进行描述和预测，但过往大多数湍流研究结论都还是经验性的，对湍流的基础性理解依然欠缺。2000 年，美国克雷（Clay Mathematics Institute，CMI）数学研究所悬赏了七个重要的数学难题，N-S 方程（Navier-Stokes 方程，流动控制方程）的求解位列第六。深入研究湍流，可以加强人们对随机性和确定性的认识。

在工程方面，受航空航天等领域需求的引导，高超声速湍流（一般指来流马赫数大于 5）成为当前流体力学研究的热点。飞行器表面流体从层流转捩到湍流后，摩阻和热流变为原来的数倍。理解湍流与转捩的机理，对解决运载火箭 \ 导弹热防护、大型飞机减阻降噪、吸气式高超声速飞行器发动机燃烧室湍流混合等问题有重要意义。

直接数值模拟（Direct Numerical Simulation, DNS）是研究湍流的重要手段。DNS 是使用足够密的网格直接离散求解 N-S 方程，可以识别最小尺度的湍流涡，能够得到湍流与转捩过程时空间演化的全部信息，是目前最为可靠的湍流计算手段。精细的 DNS 不仅可以帮助人们对湍流机理开展研究，也能为飞行器等工程装备设计提供参考与评估，甚至可以发现实验中难以观察到的现象，扩展人类认知的边界。高超声速湍流 DNS 开展较晚，近十余年才发展起来，主要原因之一是湍流 DNS 对空间网格规模与时间推进步数的要求极高，计算量极大，根据 Lesieur M 的推算，在积分尺度为 L 的实际问题中，对应雷诺数为 $Re_{L,}$ 则 DNS 所需的空间网格数和时间步数至少要达到 $O(Re_L^{9/4})$ 和 $O(Re_L^{1/2})$ [1]。美国 Boeing 公司的 Edward N. Tinoco 在 2009 年估计，以当时的计算机发展速度，到 2080 年才有可能对民航客机全机进行 DNS[2]。

在过往对高超声速湍流的 DNS 中，往往通过在大规模 CPU 集群上做并行计算来实现，节点间采用 MPI（Message Passing Interface）、ZeroMQ（0MQ）、Hadoop 等方式进行数据通讯。然而，多核架构的 CPU 的计算能力现今已被众核架构的 GPU 甩在身后，图 1、图 2 分别展示了 2003 年以来 Nvidia GPU 与 Intel CPU 的单、双精度浮点运算峰值性能和访存带宽的对比，可以看出，GPU 的峰值性能与访存带宽如今都已经达到 CPU 的近百倍。表 1 展示了 2019 年 11 月公布的超级计算 TOP500 榜单前十位，其中一半超算采用了异构加速器的模式，说明利用 GPU 等加速器进行加速计算的异构模式已经成为高性能计算发展的主流。

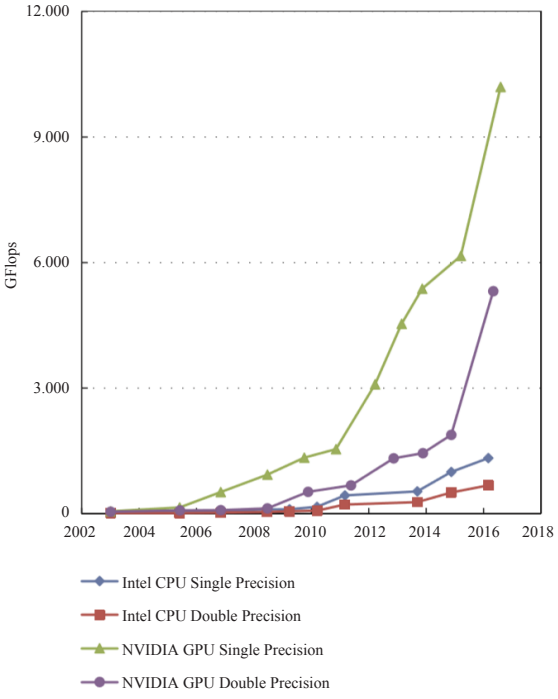


图 1 CPU 与 GPU 浮点运算能力 [16]

Fig.1 Floating-point operations per second for CPU and GPU[16]

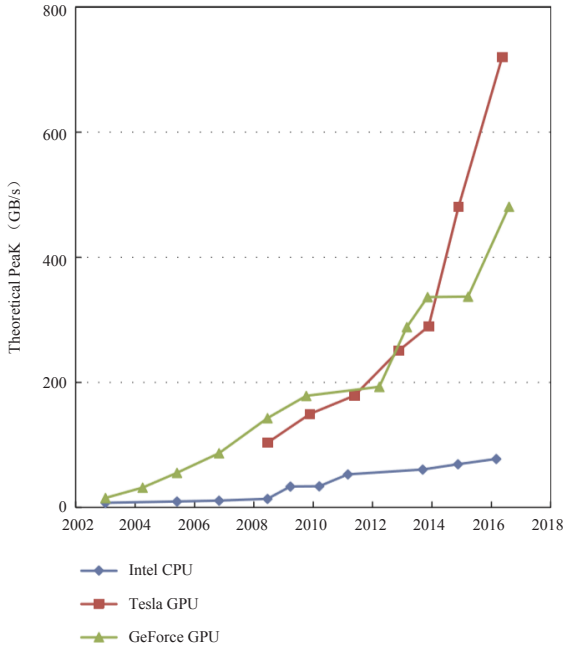


图 2 CPU 与 GPU 的带宽 [16]

Fig.2 Memory bandwidth for the CPU and GPU[16]

表 1 超级计算 TOP500 榜单中前十名 [17]

Table 1 Top 10 in the TOP500 Supercomputer list[17]

Rank	Machine	Architecture
1	Summit	异构加速器
2	Sreera	
3	神威太湖之光	异构众核
4	TH-2	异构加速器
5	Frontera	同构众核
6	Piz Daint	异构加速器
7	Trinity	同构众核
8	ABCI	异构加速器
9	SuperMUC-NG	同构众核
10	Lassen	异构加速器

利用 GPU 求解流体力学问题前人已经开展过一些工作，2003 年 Takashi Amada 在 GPU 上实现了基于平滑分子动力学（Smoothed Particle Hydrody-namics）的粒子流动模拟 [3]，同年，J.Kruger 利用 GPU 求解了二维不可压 N-S 方程 [4]。2007 年，T.Brandvik 完成了二维 Euler 方程在 GPU 上的求解 [5]，随后他将求解器扩展到了三维 [6]。2010 年，D.Jacobsen 利用 MPI-CUDA 在 128 个 GPU 上并行求解了不可压 N-S 方程，但弱扩展性只有 17%[7-8]。随后他将 GPU 并行数量扩展到 256，三维并行效率依然没有提高 [9]。2012 年，Khajeh–Saeed 使用 192 块 GPU 对均匀各项同性湍流进行了 DNS[10]。2014 年，Wang 利用 GPU 对壁湍流分别进行了 LES 和 DNS，他的工作是基于求解 LBE（Lattice Boltzmann Equation）[11]。2015 年 V.Emelyanov 开发了基于 GPU 的三维可压缩的欧拉方程与 N-S 方程有限体积法求解器，获得 20 到 50 倍的加速 [12]，之后他们对亚、跨、超声速翼型外流场利用 GPU 开展了数值模拟，最高马赫数为 1.6[13]。2018 年，Lai 等人利用 CPU/GPU 异构系统架构下的可压缩 NS 方程求解器，对高超声速的双椭球绕流进行了计算，马赫数为 8.02[14]。随后，他们又开展了对最大马赫数为 10.02 的高超声速三维航天飞机模型的数值模拟，在 4 块 GPU 上取得了比单块 CPU 最大 147 倍的加速效果 [15]。

本文利用 GPU 对高超声速湍流问题进行 DNS，最大扩展到 64 块 GPU 卡，证明 GPU 可以用于高超声速湍流 DNS，并能大大缩短计算时间。

1 控制方程与数值方法

1.1 控制方程

本文使用的有限差分法求解器基于三维可压缩 N-S 方程。形式如下：

$$\frac{\partial U}{\partial t}+\frac{\partial F_1(U)}{\partial x}+\frac{\partial F_2(U)}{\partial y}+\frac{\partial F_3(U)}{\partial z}=\frac{\partial G_1}{\partial x}+\frac{\partial G_2}{\partial y}+\frac{\partial G_3}{\partial z} \quad (1)$$

其中 U 为守恒变量，包含质量密度、动量密度与能量密度，F_1，F_2，F_3 为三个方向的无粘通量，G_1，G_2，G_3 为三个方向的粘性通量。

其中粘性通量中的粘性应力由牛顿应力方程获得。总能 E 由内能与动能组成。为使方程封闭补充，另需补充理想气体状态方程。

计算基于无量纲的 N-S 方程。

在计算时，利用三维 Jacobian 变换，将物理空间的网格变换到直角坐标系下的计算空间后进行计算。

1.2 空间离散格式

为减少无粘项的混淆误差，保证计算稳定，无粘项先使用 Steger-Warming 分裂 [18]，后利用 7 阶 Weno-SYMBO[19] 格式离散。Weno-SYMBO 格式相比于传统 Weno 格式，在保持高精度的同时，具有更小的数值耗散，更利于计算湍流问题。

粘性项使用 6 阶中心格式进行离散。

1.3 时间推进格式

湍流问题具有极强的非定常性，为能够分辨出湍流结构实时变化，时间步长不能取得太大。显式格式计算量小，物理问题所必须的较小时间步长正好保证了计算稳定性，非常适合湍流计算。此处使用 TVD 型的三步三阶 Runge-Kutta 方法。

1.4 滤波

高超声速问题由于流场动能非常接近于总能，计算如果产生非物理波很可能使动能超过总能，进而出现负温度导致的计算发散，为增加计算稳定性，在计算过程中，每经过 10 步计算进行一次滤波，以滤除流场中高频非物理波。守恒型滤波色散误差小，对计算结果影响较小。这里采用守恒型的捕捉激波滤波 [19]。

2 Opencfd-SCU 程序流程与 GPU 程序特点

CPU/GPU 异构系统架构（HSA）下的计算流体力学程序 OpenCFD-SCU（OpenCFD Scientific Computing - CUDA）以作者前期开发的高精度有限差分求解器 OpenCFD-SC[20] 为基础，两者使用相同的程序框架，程序框架如图 3。

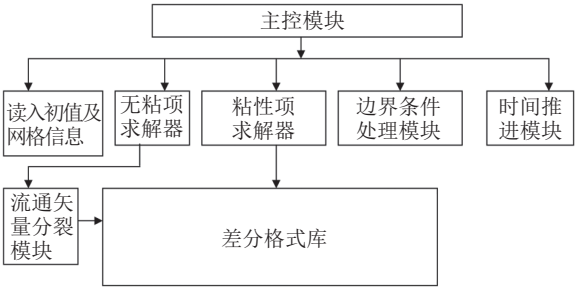


图 3 OpenCFD-SC 计算流程

Fig.3 The program hierarchy of OpenCFD-SC

程序先读入控制文件中的参数信息，并通过 MPI_bcast 广播到各个进程，之后，各个进程读取初值与网格，并使用 Memcpy3D 将初值与网格信息传递到 GPU 的全局内存（Global Memory），而后，利用 GPU 计算 Jocabian 系数。程序初始化完成，正式进入计算环节。

程序的所有计算都在 GPU 上进行，为了避免过多的 CPU-GPU 间数据传输影响程序性能，数据在传递到 GPU 上之后，全程储存在 GPU 的全局内存中，CPU-GPU 间数据通讯只在需要进行数据交换时启动。

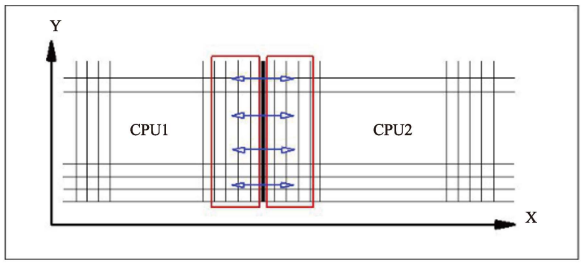


图 4 区域分割示意图

Fig.4 Sketch map of region segmentation

为最大化减少数据交换量，以提高并行效率，我们只对差分需要的边界上 3-4 层数据进行交换。如图 4 所示，以二维为例，只有 MPI 进程之间差分计算涉及的重叠部分网格点上的数据需要进行数据交换。交换时。需要交换的数据要先从 GPU 传递到 CPU，然后，CPU 之间通过 MPI 进行数据交换，最后，CPU 将交换得到的数据再传至 GPU，GPU 继续接下来的计算。CPU 上的 MPI 分割采用三维剖分。目前为了保证计算的稳定，采用阻塞式通讯。

程序中所有数据都是双精度。

3 Opencfd-SCU 程序优化

GPU 程序编程思想与 CPU 程序不同，GPU 编程实际是控制线程索引与数据地址指针的关系，从而操作成大量线程同时对数据进行计算。GPU 上的资源类型较复杂，在 GPU 上进行程序优化，主要以最大化利用 GPU 资源为思想。以下是我们做的一些工作。

3.1 数据重排列

在进行 CPU-GPU 间数据交换时，我们在 GPU 上将所有需要交换的数据重排列为连续一维数组，再通过 Memcpy 传递至 CPU。

之所以进行重排列操作，是因为需要传输的数据是数据块边界上的数据，只是完整数据块中靠外的几层，他们在全局内存中的存储并不连续，而 GPU 较高的访存带宽是通过高访存位宽实现的，只有对在内存中连续排列的大数据块，GPU 高访存的特性才能充分发挥，在计算时，可以通过对内存进行对齐与合并，再控制指针访问内存中的数据，从而降低数据内存不连续带来的影响。然而，在使用 Memcpy 函数拷贝数据时，函数默认的访存与传输方式难以控制，不连续的数据对 Memcpy 函数效率影响明显，直接影响了 CPU-GPU 间数据传递速度。在我们的测试中，重排列的操作可以明显提升 Memcpy 速度。

3.2 对块的大小进行控制

GPU 的线程层次由线程格、线程块、线程（Grid，Block，Thread）三层组成，每层又可以构成一个 x，y，z 方向的三维结构。

而 GPU 程序运行时，可以理解为数据被划分为许许多多的数据单元，由流式多处理器（Streaming Multiprocessor，SM）控制线程对各个数据单元进行操作。SM 是 GPU 上重要的共享资源，每个 GPU 上 SM 数量固定，Nvidia Tesla V100 为 80 个，最大化 SM 占用率，才能最大化 GPU 性能。

SM 通过控制线程束（Warp）来操作线程的行为，线程束是线程调度的最小单位。实际上，GPU 对每个 SM 可容纳的 Block 数、每个 SM 可容纳 Warp 数、每个 SM 可容纳 Thread 数都有限制，每个 Warp 所包含的 Thread 数量也是固定的。以 Volta 架构的 Nvidia Tesla V100 为例，其计算能力（Compute Capability）为 7.0，对应每个 SM 最多可容纳 32 个 Block、64 个 Warp、2048 个 Thread，每个 Warp 包含的 Thread 数量为 32。也就意味着，对 Tesla V100 而言，Block 的大小必须是 32 的整数倍，且为了使 SM 控制的 Thread 数最大，占用率最高，Block 最小尺寸应大于 64（SM 可容纳的最大 Threads 数 /SM 可容纳最大 Block 数）。

与此同时，Block 的大小又受限于 GPU 上其他的共享资源，例如：共享内存（Shared Memory），寄存器（Register）。以 Nvidia Tesla V100 GPU 为例，GPU 上共有 65536 个大小为 32bit 的 Register，为所有线程共享。若将所有寄存器分配给一个 SM，并在一个 SM 控制 2048 个 Thread 的情况下，则每个 Thread 中使用的 Register 用量最多不超过 32。如果 Thread 中

使用的 Register 超过 32，CUDA 会强制减少 Block 数，从而导致 SM 利用率的下降，此时，应尽可能减少 Register 使用量，如果 Register 用量不能减少，也应适当调整 Block 的大小，使 Block 大小适应 SM 可启动的最大线程规模。此外，Shared Memory 是 Block 内的共享资源，如果对 Shared Memory 的申请超过最大限额，GPU 会从 Global Memory 中寻找替代资源，导致核函数执行速度下降，此时应减小 Block 的大小。

在我们的测试中也发现过大过小的 Block 都会影响性能。表 2 列举了在我们的测试中对 6 阶中心差分格式的核函数 dx0 设置不同 Block 大小时核函数的执行时间，结果发现，对 dx0 而言，最合适的 Block 大小为 32*2*2。不同核函数资源需求量不同，Block 的最优尺寸可能也不同，应视情况而定。

表 2 线程块大小与函数运算耗时

Table 2 Thread block size and the time of function operation

Block大小	时间（ms）
32*1*1	821.47
32*2*1	449.34
32*2*2	349.12
32*4*2	349.37
32*4*4	371.04

3.3 对寄存器、共享内存、常量内存的使用

GPU 上的内存层次分为多层。尽可能利用好各种类型的内存资源，如寄存器（Register）、共享内存（Shared Memory）、常量内存（Constant Memory），有利于提高程序性能。

上面已经说到，过多使用寄存器，会导致 SM 利用率下降，线程并发数量减少，影响程序性能。在我们的程序中，我们令核函数内临时变量尽可能复用，以减少 Register 的开销，此外也采用将寄存器用量过多的核函数进行拆分的方式减少单个核函数的寄存器用量。

Shared Memory 是 GPU 上可编程缓存，与 1 级缓存有同样高的访存速度，与 Global Memory 相比，延迟低 20 到 30 倍，带宽大约高 10 倍。它可以在核函数外声明，并在核函数内调用，Shared Memory 在 Block 内是共享的。Nvidia Tesla V100 GPU 具有 98 304 字节大小的 Shared Memory，我们在各个核函数中采用一个名称相同的一维向量作为中间变量，并在 Shared Memory 上为其开辟内存。这样可以提高访存速度。在我们的程序中，例如 7 阶 WENO-SYMBO 格式进行数值通量的计算时，先从 Global Memory 获取流场数据，加权计算得到通量放在 Shared Memory 中再进行最后一步的差分，这样的操作可以使核函数的性能提升 20%。

Constant Memory 是 GPU 上一个高速只读内存，将程序始终不变的常量存放于 Constant Memeoy 中，也可以提高访问速度，而且能使寄存器用量减少。以 7 阶 WENO-SYMBO 格式为例，有接近 60 个常量系数，如果放到寄存器中，势必会导致核函数开销过大进而并发数减少，将这些系数放到 Constant Memory 中是保障核函数性能的关键，在我们的测试中，网格量为 256^3 时利用 Constant Memory 可以用 32*4*4 的 Block 规模来运行程序，而未使用 Constant Memory 时程序最大只能以 32*4*2 的 Block 规模运行，并发数和性能都有一倍的差距。

3.4 访存的合并与对齐

GPU 在线程操作时以线程束为基本单位，具有一定并发性，而在访存时，GPU 也是并发操作的，它利用高访存位宽获取了高的访存带宽。在 V100 上，访问 Global Memroy 时有两种方式：（1）通过 1 级缓存一次访问 128 字节的数据；（2）通过 2 级缓存一次访问 32 字节的数据。GPU 默认的访存方式是 2 级缓存访问。在此情况下，一次访问操作会访问 4 个双精度数据，为了不浪费访存带宽，最大化提高带宽利用率。在访问时，我们让每一个线程束访问首地址为 32 字节（128 字节）整数倍的位置，并获

取连续 32 字节整数倍的数据。

在我们的程序中，以 6 阶中心差分格式核函数 dx0 为例，为了使访存能够合并，我们在一个 kernel 中执行两次 reinterpret (double 4)，读入 8 个点的数据，在一个 kernel 上进行 2 个点的差分运算。执行一次 reinterpret (double 4) 读入的数据长度为 32 字节，确保每次获取的都是连续的 32 字节数据，此时访存是合并的。在经过这样的优化后，dx0 执行时间从未优化前的 349ms 缩减到了 147ms，核函数运行速度快了一倍以上。

4 高超声速湍流 DNS

钝锥是火箭、导弹等外形的典型头部特征，对钝锥开展 DNS 具有重要工程意义。我们通过异构程序使用最多 64 块 GPU 对 6° 攻角，1mm 头半径的小钝头锥开展了 DNS，有攻角钝锥边界层流动如图 5。

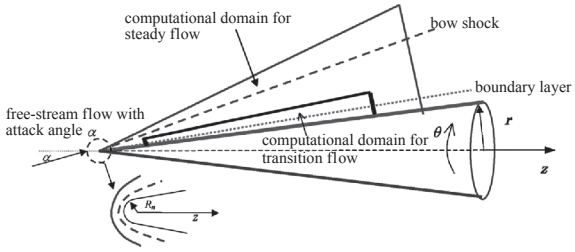


图 5 有攻角钝锥边界层流动 [21]

Fig.5 Schematic of boundary layer flow over a blunt cone with angle of attack[21]

在进行 DNS 模拟之前，我们使用粗网格，利用有限体积程序 OpenCFD-EC[22] 获取定常的层流流场，利用该流场插值出用于 DNS 的初值与边界条件。

有限体积法的层流计算时，无粘项使用 Van Leer 分裂与 MUSCL 限制器，时间推进采用 LU-SGS 方法。

为促进转捩，模拟钝锥表面粗糙单元，在钝锥头部 90mm 到 100mm 位置添加吹吸气扰动，扰动幅值为来流速度的千分之一。

4.1 网格与流场设置

参数设置如表 3。来流马赫数为 6，来流攻角 6°，初始壁温为 294K，来流温度为 79K。钝锥半锥角为 7°，头半径为 1mm。

表 3 流动参数设置

Table 3 Flow parameter Setting

Re_n	Ma_∞	α(AOA)	T_W	T_∞	half cone angle
10000	6	6°	294K	79K	7°

网格规模为 1600*1200*120，其中流线网格数 1600，周向网格数 1200，法向网格数 120，总网格 2.3 亿，流向网格在头部进行加密，周向采用非均匀网格在迎风面对网格进行加密。壁面第一层网格为 0.01mm。流向与周向网格见图 6、图 7。

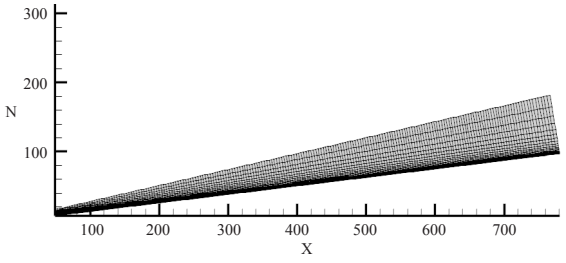


图 6 钝锥表面流向网格

Fig.6 The flow direction's grids of blunt cone

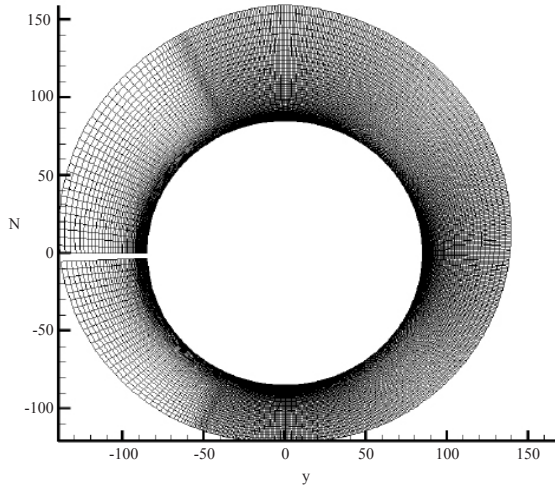


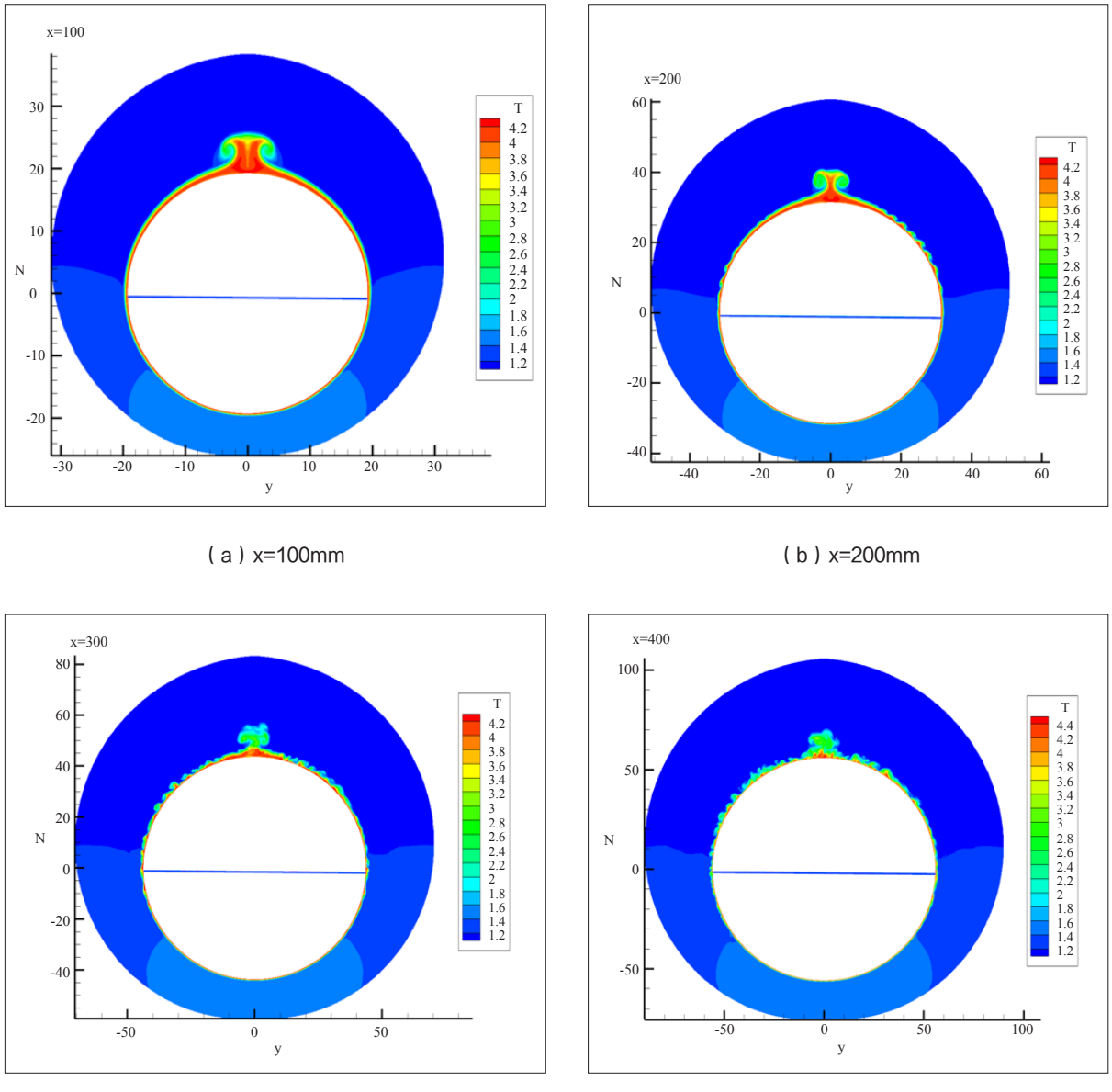
图 7 钝锥表面周向网格

Fig.7 The circumferential direction's grids of blunt cone

4.2 计算结果

图 8 为距离头部 100 mm、200 mm、300 mm、400mm 位置，钝锥表面的热流分布。可以看出钝锥在 x=100 mm 位置，表面卷起蘑菇状涡，随后在周向表面出现波纹状结构，在 x=300 mm 位置已经开始转捩，x=400 mm 位置流场已经是充分发展的非均匀湍流。

图 9 是我们绘制的从不同方向观察的钝锥表面摩阻分布。可以看到，由于来流的冲击作用，钝锥下表面迎风面的摩阻明显高于上表面背风面的摩阻。且迎风面摩阻分布较均匀，被风面摩阻分布呈条带状结构。条带状的摩阻结构证明流动已经进入湍流。

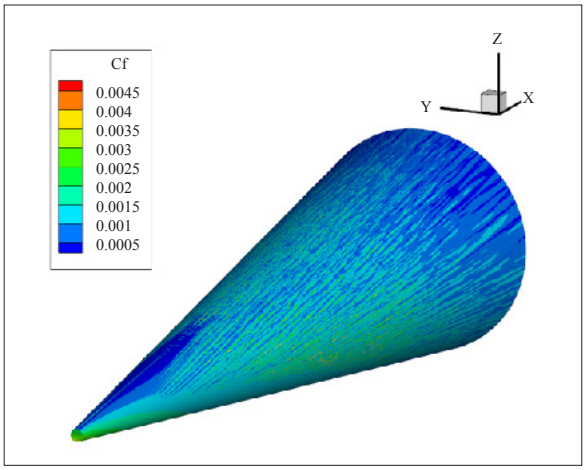


(a) x=100mm　　(b) x=200mm

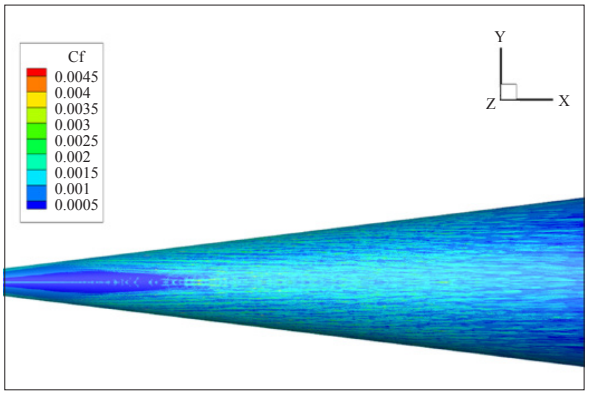
(c) x=300mm　　(d) x=400mm

图 8 钝锥不同位置温度分布

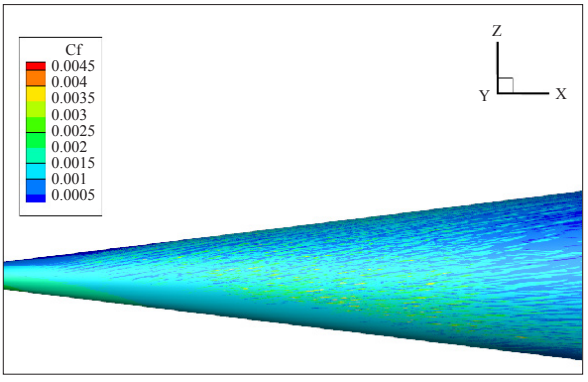
Fig.8 Heat flow distribution in different positions of blunt cone



（a）三维



（b）z 方向



（c）y 方向

图 9 钝锥表面摩阻分布

Fig.9 Surface friction distribution of blunt cone

4.3 计算时间

在使用 GPU 计算之前，我们先使用 CPU 端的程序 OpenCFD-SC 对这一算例计算，使用 1024 个 CPU 核心，CPU 核心的频率为 2.5GHz，系统内存频率为 2666MHz，单核峰值性能 0.01 TFlops/S。使用此 CPU 计算每迭代一时间步耗时 1.498s（100 次迭代取平均）。

而后，我们通过异构程序 OpenCFD-SCU 在 GPU 上对相同算例进行计算，我们使用的 GPU 频率为 1455MHz，显存带宽 900GB/S，双精度浮点性能 7TFlops/S。我们分别使用 16、32、48、64 块 GPU 卡对其进行计算，表 4 列出了在这 4 种规模下，计算每迭代一时间步所耗时间（100 次迭代取平均）。

表 4 不同 GPU 数量，每步迭代耗时

Table 4 Wall time per iteration using different numbers of GPU

	GPU卡数	每步计算时间(S/秒)
case 1	16	1.591
case 2	32	0.745
case 3	48	0.551
case 4	64	0.395

可以看出，在使用 16 块 GPU 卡对钝锥算例进行计算时，每步迭代时间已经接近使用 1024 个 CPU 核心进行计算的时间。程序在单块 GPU 上取得了比单个 CPU 核心快 60 倍的加速效果。使用更多 GPU 卡，单 GPU 比单 CPU 加速效果甚至更明显。图 10、图 11 中展示了在 GPU 上程序的加速比与并行效率，可以看出有超线性的情况，这可能是此算例对 16 个 GPU 而言负载过大，但也可以看出，OpenCFD-SCU 程序整体扩展性很好，在最大 64 块 GPU 卡上，并行效率基本没有变化。

在整个算例的计算中，我们迭代了 10 万步，使用 64 块 GPU 只用 11 个小时左右就得出了结果，而以往对相同类型算例的计算，使用 200 余个主频 2.5GHz 的 CPU 核心，往往需要花费一周时间。如果是对更大规模的算例，可能需要数月时间，利用 GPU 能将耗时缩短一个数量级。

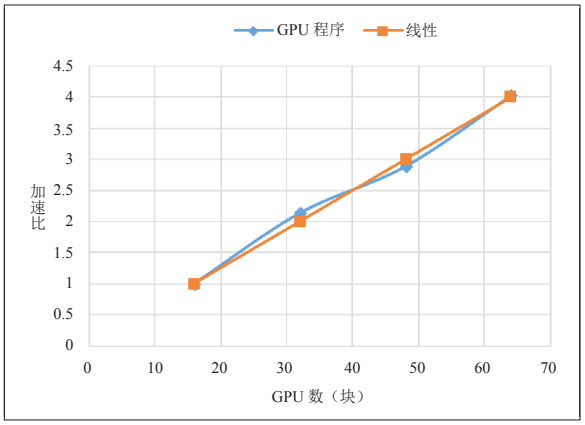


图 10 GPU 程序加速比

Fig.10 GPU program speedup

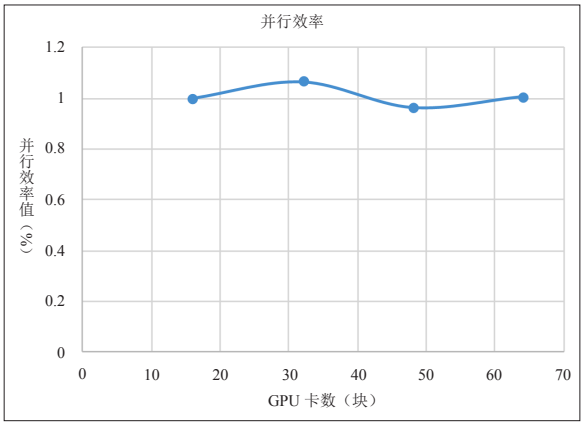


图 11 GPU 程序并行效率

Fig.11 Program parallel efficiency

5 小结

本文介绍了 GPU 在高超声速湍流计算上的应用。描述了 GPU 程序优化相关技术，并使用 CPU/GPU 异构程序对来流马赫数为 6 的 6° 攻角小钝头锥开展了 DNS。

结果表明，GPU 可以胜任高超声速湍流直接数值模拟，并能取得了较好的加速效果，程序在单 GPU 上比单 CPU 核加速了 60 倍，大大缩短了模拟时间。可以想象，在未来会有更多高超声速湍流模拟问题在 GPU 上开展。

目前，OpenCFD-SCU 的功能模块尚不够丰富，核函数还有优化空间，未来将继续添加功能以适应更多不同的算例，还将对核函数进行深度优化，以提升程序性能。此外，在程序结构上，目前仍采用单线程阻塞式 MPI 通讯，为加速整体性能，未来将对程序结构做调整，完成 CPU 上计算与 MPI 通讯重叠，在 GPU 方面，也将使用更多流（Stream）操作来提升 GPU 并发性，并使 GPU 上的计算与主机、设备间内存拷贝进行重叠，进而提升整体性能。

利益冲突声明

所有作者声明不存在利益冲突关系。

参考文献

[1] Lesieur M, MetaisO,Comte P. Large-eddy simulations of turbulence[M]. Cambridge University Press, 2005.

[2] http://www.casc.org/meetings/09sept/Edward_ Tinoco.ppt.

[3] Takashi Amada, et al.. Particle-Based Fluid Simulation on GPU. http://chihara.naist.jp/people/2003/takasi-a/gp2/.

[4] Krüger J, Westermann R. Linear algebra operators for GPU implementation ofnumerical algorithms [J]. ACM Trans. Graph. 2003, 22 (3): 908–916.

[5] Brandvik T, Pullan G. Acceleration of a two-dimensional Euler flow solver usingcommodity graphics hardware [J]. Proceedings of the Institution of MechanicalEngineers, Part C: Journal of Mechanical Engineering Science. 2007, 221 (12): 1745–1748.

[6] Brandvik T, Pullan G. Acceleration of a 3D Euler Solver Using Commodity Graph-ics Hardware [M]. Aerospace Sciences Meetings. Ameri-can Institute of Aeronautics and Astronautics, 2008.

[7] Jacobsen D, Thibault J, Senocak I. An MPI-CUDA Implementation for MassivelyParallel Incompressible Flow Computations on Multi-GPU Clusters [M]. Aerospace Sciences Meetings. American Institute of Aeronautics and Astronautics, 2010.

[8] Jacobsen D, Senocak I. Scalability of Incompressible Flow

Computations onMulti-GPU Clusters Using Dual-Level and Tri-Level Parallelism [M] .Aerospace Sciences Meetings. American Institute of Aeronau-tics and Astronautics, 2011.

[9] Jacobsen D A, Senocak I. Multi-level parallelism for incompressible flow compu-tations on GPU clusters [J]. Parallel Computing,2013, 39 (1): 1–20.

[10] Khajeh-Saeed Ali, J. Blair Perot. Direct numerical simulation of turbulence using GPU accelerated supercomputers[J]. Journal of Computational Physics, 2013, 235: 241-257.

[11] X.Wang,Y.Shangguan,N.Onodera,H.Kobayashi,andT. Aoki. Direct numerical simulation and large eddy simulation on a turbulent wall-bounded flow using lattice Boltzmann method andmultiple GPUs[J].Mathematical Problems in Engineering,vol. 2014.

[12] Emelyanov, Vladislav N., A. G. Karpenko, K. N. Volkov. Development of advanced computational fluid dynamics tools and their application to simulation of internal turbulent flows[J]. Progress in Flight Physics–Volume 7,2015,7: 247-268.

[13] Emelyanov, Vladislav, Anton Karpenko, Konstantin Volkov. Development and acceleration of unstructured mesh-based CFD solver[J]. Progress in Flight Physics–Volume 9, 2017, 9: 387-408.

[14] Lai, Jianqi, Hua Li, Zhengyu Tian. CPU/GPU Heterogeneous Parallel CFD Solver and Optimizations[C]. In Proceedings of the 2018 International Conference on Service Robotics Technologies. ACM, 2018: 88-92.

[15] LAI, Jianqi, et al..A Multi-GPU Parallel Algorithm in Hypersonic Flow Computations[J]. Mathematical Problems in Engineering, 2019.

[16] https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html.

[17] https://www.top500.org/.

[18] Joseph L Steger, R.F Warming. Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods[J]. Journal of Computational Physics, 40(2): 263-293.

[19] M.P. Martín, E.M. Taylor, M. Wu,et al.. A bandwidth-optimized WENO scheme for the effective direct numerical simulation of compressible turbulence[J]. Journal of Computational Physics, 220(1): 270-289.

[20] 李新亮.OpenCFD-SC用户手册.http://pan.baidu.com/s/1slfC5Yl.

[21] Li X, Fu D,Ma Y . Direct numerical simulation of hypersonic boundary layer transition over a blunt cone with a small angle of attack[J]. Physics of Fluids, 2010, 22(2): 025105.

[22] 李新亮.OpenCFD-EC理论手册.http://pan.baidu.com/s/1slfC5Yl.

收稿日期：2019 年 12 月 6 日

党冠麟，中国科学院力学研究所，在读博士研究生。主要研究方向为可压缩湍流与转捩、直接数值模拟、高性能计算。

本文承担工作为程序实现、优化与应用测试。

Dang Guanlin is a PhD candidate in Institute of Mechanics, Chinese Academy of Sciences. His research interests include compressible turbulence and transition, direct numerical simulation, and high-performance computing.

In this paper he undertakes the following tasks: code implementation, optimization, and application testing.

E-mail：dangguanlin@imech.ac.cn

刘世伟，中国科学院数学与系统科学研究院，计算数学与科学工程计算研究所在读博士研究生，主要研究方向为计算流体力学高精度格式、大涡模拟，以及 OpenCFD 在 GPU 上的异构并行程序移植与优化。

本文承担工作为框架代码实现、程序优化与测试。

Liu Shiwei is currently pursuing the PhD degree with

the Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences. His research interests include high-order schemes of computational fluid dynamics, large eddy simulation, as well as porting and optimization of heterogeneous parallel programs of OpenCFD on CUDA platform.
In this paper he undertakes the following tasks: code implementations of the whole framework，program optimization and testing.
E-mail：liusw@lsec.cc.ac.cn

胡晓东，中国科学院计算机网络信息中心，博士，工程师，主要研究方向为并行计算、计算流体力学。
本文承担工作为程序优化、编程指导。
Hu Xiaodong is an engineer in Supercomputing Center of Computer Network Information Center, Chinese Academy of Sciences. He received master's degree in fluid dynamics from Northwestern Polytechnical University in 2011, and the PhD degree in computer software and theory form University of Chinese Academy of Sciences in 2019. His current research interests include computational fluid dynamics and high-performance computing.
In this paper he undertakes the following tasks: program optimization and guidance of programing.
E-mail：huxd@sccas.cn

张鉴，中国科学院计算机网络信息中心，博士，研究员，主要研究方向为科学计算、高性能计算和计算机可视化。
本文承担工作为程序整体结构设计、编程指导。
Zhang Jian received the BS degree in computational mathematics from Peking University, PR China, in 1995, and the PhD degree in applied mathematics from the University of Minnesota in May 2005. He is a professor in Computer Network Information Center, Chinese Academy of Sciences. From June 2005 to June 2009, he was a postdoc in the Pennsylvania State University working on scientific computing and modeling. His current research interests include scientific computing, high-performance computing, and scientific visualization.
In this paper he undertakes the following tasks: code design and execution director of programing.
E-mail：zhangjian@sccas.cn

李新亮，中国科学院力学研究所高温气体动力学国家重点实验室，中国科学院大学岗位教授，研究员，理学博士，博士生导师。任中国空气动力学会物理力学专业委员会副主任委员，计算物理学会常务理事，Computers & Fluids 杂志编委，《空气动力学报》及《计算物理》杂志编委等职。
目前主要从事计算流体力学及湍流研究。构造了优化保单调、加权群速度控制格式等高精度激波捕捉格式，并在此基础上开发一套开源的高精度计算流体力学软件 OpenCFD；进行了一系列典型超声速、高超声速可压缩湍流的直接数值模拟，并在此基础上对湍流机理、模型及控制进行了深入探索。
本文承担工作为程序结构设计与开发、DNS 研究指导。
Li Xinliang, is the professor in laboratory of high-temperature gas dynamics, institute of mechanics, Chinese academy of sciences (CAS) and University of Chinese academy of sciences (UCAS). Prof. Li got his PhD in institute of mechanics, CAS in 2000, and then worked as a postdoctoral researcher in Tsinghua University. Dr. Li works in institute of mechanics, CAS since 2002, and his is major in CFD and DNS of turbulent flows.
In this paper he undertakes the following tasks: code design and research guidance of the DNS.
E-mail：lixl@imech.ac.cn

引文格式：党冠麟,刘世伟,胡晓东,张鉴,李新亮. 基于CPU/GPU异构系统架构的高超声速湍流直接数值模拟研究[J].数据与计算发展前沿,2020,2(1):105-116. DOI:10.11871/jfdc.issn.2096-742X.2020.01.009.PID:21.86101.2/jfdc.2096-742X.2020.01.009.
Dang Guanlin, Liu Shiwei, Hu Xiaodong, Zhang Jian, Li Xinliang . Direct Numerical Simulation of Hypersonic Turbulence Based on CPU/GPU Heterogeneous System Architecture [J].Frontiers of Data & Coputing,2020,2(1):105-116.DOI:10.11871/jfdc.issn.2096-742X.2020.01.009.PID:21.86101.2/jfdc.2096-742X.2020.01.009.