科学通报 2010 年 第 55 卷 第 20 期: 1979 ~ 1986
论 文 www.scichina.com csb.scichina.com
《中国科学》杂志社
SCIENCE CHINA PRESS

单相流动数值模拟的 SIMPLE 算法在 GPU 上的实现

王健[①], 许明[①②], 葛蔚[①], 李静海[①]

① 中国科学院过程工程研究所, 多相复杂系统国家重点实验室, 北京 100190;
② 中国科学院研究生院, 北京 100049
E-mail: wangjian@home.ipe.ac.cn

2009-12-25 收稿, 2010-04-12 接受
国家自然科学基金(20221603, 20490201)、中国科学院知识创新工程重要方向性项目(KGCX2-YW-124)和国家重点基础研究发展计划(2009CB219906)资助

摘要 基于交错网格的SIMPLE算法, 利用CUDA(compute unified device architecture)技术进行了图形处理器(GPU)上的直接数值模拟(DNS). 将高雷诺数方腔流作为研究实例, 在 NVIDIA GTX 295 显卡的单个和 4 个 GPU 上的计算速度最高可分别达 Intel Xeon 5430 CPU 之单核的 50 和 150 倍, 与其他模拟结果的对比表明了上述计算的合理性, 并展示了采用 GPU 实现高精度大规模的湍流计算的前景.

关键词
单相流动
直接数值模拟
CUDA
GPU
并行计算

近年来, 随着计算机以及相关技术的迅速发展, 计算流体力学(computational fluid dynamics, CFD)已逐步发展成为跨及流体力学、数值计算方法以及计算机科学等多种领域的交叉学科. 除了传统的航空、气象和水利等领域, 其在化学工程这个庞大领域中的作用也与日俱增. 相较于传统的试验逐级放大方法所存在的开发周期漫长、费用高昂、优化困难等一系列缺点, CFD 方法不仅具有成本低、速度快、资料完备、可以模拟真实及理想条件等优点, 还可以再现转化过程随时间、空间的变化, 观察普通试验难以发现的细节. 此外, 借助于发展大规模并行技术、平台和先进的模拟方法, 可以较方便地通过 CFD 计算获得大型装置工业热态条件下的宝贵信息.

直接数值模拟(direct numerical simulation, DNS)是最根本的 CFD 方法. 该方法无需引入任何简化和近似, 但计算量非常可观. 在传统的基于中央处理器(CPU)的计算机上, 仅能求解 Reynolds 数较低或几何边界比较简单的湍流问题. 而 20 世纪 90 年代发展起来的图形处理器(GPU)的计算能力现已远高于 CPU, 具有性价比和计算密度高、能耗低等优势. 近期发布的 CUDA[1]可以把 GPU 直接视为数据并行计算设备而不再将计算映射到图形操作上, 并以类 C 语言的方式给开发者更大的自由来实现 GPU 算法. 这种新的软硬件架构使得如何在流动直接数值模拟中发挥 GPU 的强大计算能力成为一个重要的课题. 当前, 基于 GPU 的 DNS[2~5]还存在精度较低或数值方法过于简单而不适用于复杂流动等问题. 到目前为止, 最接近实际应用的模拟研究主要有: Brandvik 等人[6]采用同位网格的有限体积法, 分别在 AMD 和 NVIDIA 显卡上模拟了二维和三维可压缩流动; Elsen 等人[7]采用有限差分法, 在 NVIDIA 8800 GTX 上进行了 NACA 0012 机翼亚音速绕流以及高速飞行器绕流的数值模拟. 这些方法均采用分步法首先显式估算速度场, 继而通过求解一个压力 Poisson 方程来得到下一时刻的流场分布.

本文基于 CUDA 在 GPU 上进行了交替网格 SIMPLE 算法[8,9]的并行研究, 实现了高雷诺数方腔流的直接数值模拟. 该算法属于有限体积法范畴, 采用全隐式差分格式离散化控制方程, 既能确保数值解满足物理上真实的性状, 又能维持控制容积内总的质量和动量守恒. 并且交错网格较之同位网格, 既不会增加程序设计的难度, 又可有效剔除棋盘形压

英文引用格式: Wang J, Xu M, Ge W, et al. GPU accelerated direct numerical simulation with SIMPLE arithmetic for single-phase flow (in Chinese). Chinese Sci Bull (Chinese Ver), 2010, 55: 1979—1986, doi: 10.1360/972009-1202

力场和波形速度场等数值误差问题.

1 方法

1.1 GPU-DNS 算法

模拟过程中考察了如下二维不可压缩流的 N-S 方程:

$$\nabla \cdot \boldsymbol{u} = 0,$$
$$\rho \frac{\partial \boldsymbol{u}}{\partial t} + \rho \boldsymbol{u} \cdot \nabla \boldsymbol{u} = \mu \Delta \boldsymbol{u} - \nabla p, \tag{1}$$

其中 ∇ 和 Δ 分别代表梯度算子和 Laplace 算子, $\boldsymbol{u} = (u, v)$ 表示速度向量, ρ 表示密度, μ 表示黏度, p 表示压力. 在图 1 所示的交错网格系统中, 标量参数(如密度、黏度、压力等)定义在圆黑点标识的控制容积中心, 速度分量 u 和 v 分别定义在控制容积界面与 x 和 y 方向两相邻容积中心连线的交点上. 根据有限体积法将控制方程(1)离散化, 具体的离散化方程格式以及 SIMPLE 算法的实施步骤详见文献[10]. 需要特别指出的是, SIMPLE 算法虽然已经广为应用, 但全隐式的差分格式, 以及速度分量与标量参数计算网格的交错却对数值计算, 尤其是对 GPU 程序设计带来了很大难度. 因此对于二维问题, 需要使用 3 个内核函数, SOLVE_Kernel_U, SOLVE_Kernel_V 和 SOLVE_Kernel_P, 来分别求解速度分量 u, v 和修正压力以及速度分布.

由于离散化方程信息简单、相对独立、近程相关, 因此基于交错网格的 SIMPLE 算法适合于拥有大量能够并行运算且共享多级高速显存的线程处理器的 GPU 硬件结构. 针对 GPU 的特点, 将计算区域划分成若干个小区域(其大小需符合 GPU 配置), 每个线程块(block)对应一个小区域, 而每个线程(thread)对应其中一个计算格点. 计算格点的近程相关性导致两个相邻小区域共享单层网格的重叠区域.

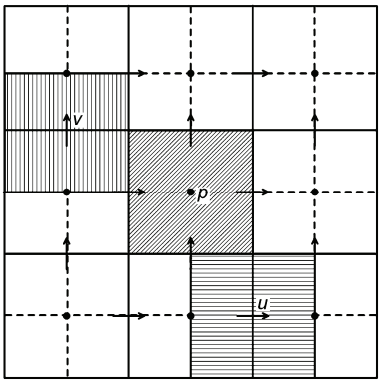

图 1 交错网格系统[10]

基于 CUDA 技术在 GPU 上进行的整个 DNS 过程可以分解成计算离散化方程系数和显式迭代求解代数方程组两个主要部分. 若仅用 GPU 处理其中一个模块, 则主存和显存间频繁的数据传输将成为其性能发挥的瓶颈. 在图 2 所示的算法流程中, 类似于 Chen 等[11]的实施方式, 我们也采用 GPU 处理所有模块, 这样 CPU 只需在开始计算前读入模拟参数和初始化信息, 并将信息由主存复制到显存, 以供 GPU 计算时调用.

为确保数值解满足物理上真实的性状以及维持控制容积内总的质量和动量守恒, 有限体积法在控制方程离散化过程中严格遵循 4 项基本法则[10], 所得代数方程组的系数矩阵是严格对角占优的, 因此采用显式的迭代方法, 如本文采用的 Jacobi 迭代法, 进行方程求解是可行的. CPU 单核计算结果已经证实, 经由 3~4 步迭代就可以获得代数方程组的收敛解. 即便是相邻子域间有重叠区域, 显式迭代法也能够保证 GPU 上格点信息的同步更新. 另外, 各计算格点对应的离散化方程:

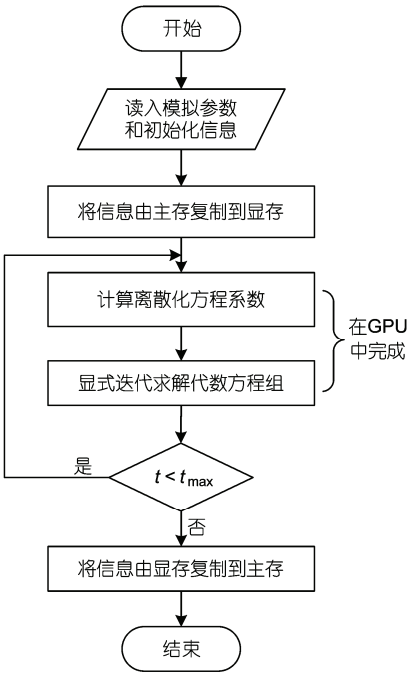

图 2 直接数值模拟 GPU 计算流程图

$$a_{\mathrm{p}}\phi_{\mathrm{p}}=\sum_{\mathrm{nb}}a_{\mathrm{nb}}\phi_{\mathrm{nb}}+b_{\mathrm{p}} \tag{2}$$

中仅涉及该格点及其相邻格点信息，此格点数与体系大小无关，三维时共 7 个，二维时共 5 个，故本算法的计算量与网格点数 n 成正比. 从这些角度看，可扩展且数据局部相关的 CFD 算法在 GPU 上的实现是可能的，从而使多尺度离散模拟的框架[12]也能适用于传统的数值计算领域.

1.2 GPU-DNS+MPI 并行计算方法

虽然单个 GPU 的计算能力已经非常强劲，但对于大规模 DNS 还远远不够. 实际应用中遇到的研究体系更加复杂，规模更大. 对此，我们应用计算能力远超主流中央处理器(CPU)的图形处理器(GPU)，扩展性较好的区域分解法和常用的消息传递接口(message passing interface, MPI)可以使 DNS 在复杂流动的大规模计算上发挥更大的作用.

如图 3 所示，并行过程中将整个计算区域划分成若干个子域，每个 GPU 对应一个子域. 考虑到显式 Jacobin 迭代算法的收敛性，每个子域分别包含双层而非单层网格宽幅的内外边界区，也就是说相邻子域共享 4 层网格的重叠区域. 在 GPU+MPI 并行模式下，流场信息的实时更新导致相邻进程间发生局部频繁的通信且传输量不大，为此我们使用 4 个内核函数，BoundaryTransX, BoundaryTransY, BoundaryBackX 和 BoundaryBackY，来完成边界信息的传递. 在图 4 所示的并行算法流程中，CUDA 和 MPI 间无相互制约，子域内所有计算均在 GPU 上完成，而进程间通过 CPU 完成通信，即为 GPU-CPU-CPU-GPU 模式.

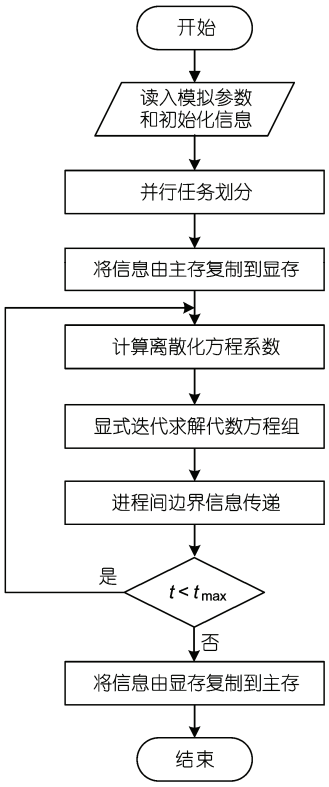

图 4 直接数值模拟并行 GPU 计算流程图

2 方腔流模拟

方腔流是流体力学领域的一个经典算例，其形式简单却可以形成复杂的流动现象，如在封闭体内可能出现角涡、纵涡、多态性、转捩、湍流等多种流体力学现象[13]. Moffatt[14]曾指出，方腔内临近底角处应存在无穷多个尺寸逐渐减小且强度逐渐减弱的角

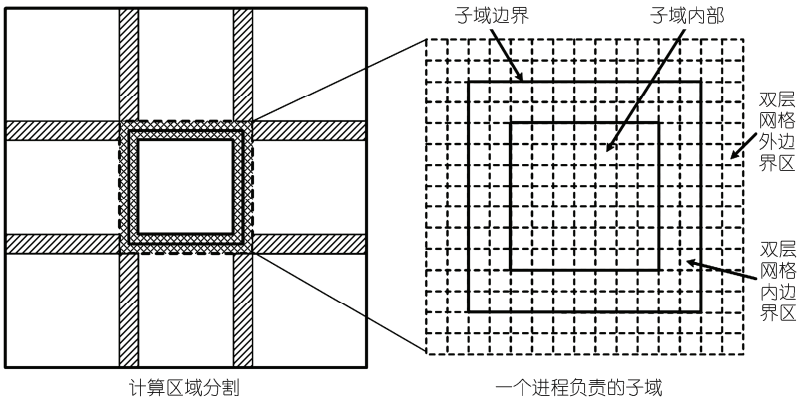

图 3 区域分解示意图

涡. 为便于讨论, 以下将右下角、左下角以及左上角的二次涡分别记为 BR_1, BR_2, …, BL_1, BL_2, …, TL_1, TL_2, …. 近年来已有很多学者对其进行了计算机模拟研究, 其中涉及到的方法包括 CFD[15], LBM[16,17]和 MD[11,18,19]等. DNS 无需引入额外的简化假设, 其结果有助于认识方腔流中各种流动现象的机理.

我们将 GPU 应用于方腔流的直接数值模拟, 所计算区域的边长大小为 1.0. 在 NVIDIA GTX 295 的单个和 4 个 GPU 上计算得到的雷诺数 $Re=10^3$ 和 $Re=10^4$ 时的流场如图 5 和图 6 所示.

上述计算结果分别与文献[13](涡-流函数法)、文献[20](同位网格的 SIMPLER 算法, CPU 并行)和文献[21](QUICK 格式, SIMPLE 算法, CPU 串行)中给出的模拟报道定性相符. 特别是, 当雷诺数升至 10^4 时(图 6(a)和 6(b)), 除左下角小涡 BL_2 之外, 方腔内的主涡和二次涡的形状、位置都能正确地获得.

需要特别指出的是, 图 5(a)和(b)底角处两个几乎不可见的二次涡, BR_2 和 BL_2, 以及图 6(a)和(b)最左下角处的二次涡, BL_2, 在上述文献中并未给出. Ghia 等人[22]给出了方腔流主涡和二次涡中心与 Reynolds 数之间的依赖关系(图 7).

从图 7 中可以看出: (1) 当 $Re=10^3$ 时确实存在两个非常靠近底角, 且几乎不可见的二次涡 BR_2 和 BL_2; (2) 当 $Re=10^4$ 时左底角处确实存在一个二次涡 BL_2; (3) 随着 Reynolds 数的增加, 主涡中心逐渐向计算区域的几何中心靠近; (4) 当 $Re\geqslant 3200$ 时, BL_1 的垂直高度将反高于 BR_1 的垂直高度, 且左上角处出现一个二次涡 TL_1.

我们的计算结果与图 7 吻合良好, 充分说明所建立的 GPU 直接数值模拟算法确实是合理有效的. 继续将雷诺数提升至 10^5, 由图 8 中所示模拟结果可以看出, 方腔内除了主涡之外, 二次涡的个数至少有 8 个之多.

3 GPU 性能的发挥

采用交错网格的 SIMPLE 算法, 图 9 中对雷诺数 $Re=10^5$ 时 Intel Xeon 5430 CPU 单核与 NVIDIA GTX

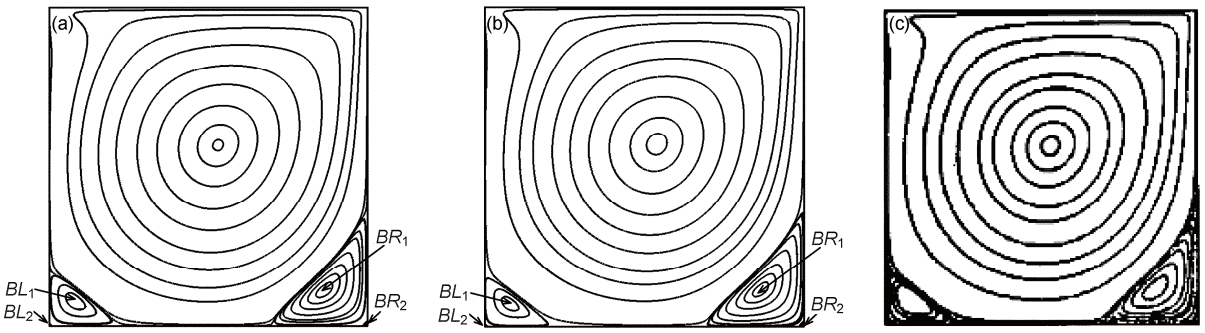

图 5 方腔流计算流线图($Re=10^3$)

(a) 单个 GPU 计算结果, n=254×254; (b) 4 个 GPU 计算结果, n=250×250; (c) 文献结果[13,20]

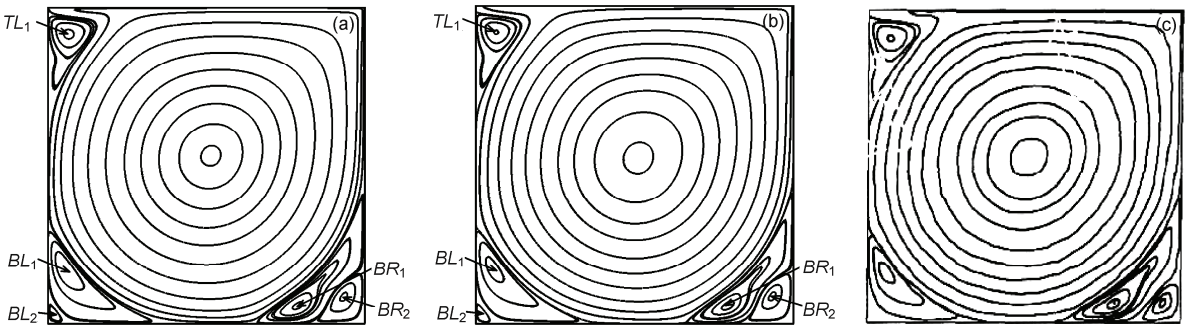

图 6 方腔流计算流线图($Re=10^4$)

(a) 单个 GPU 计算结果, n=254×254; (b) 4 个 GPU 计算结果, n=250×250; (c) 文献结果[13,21]

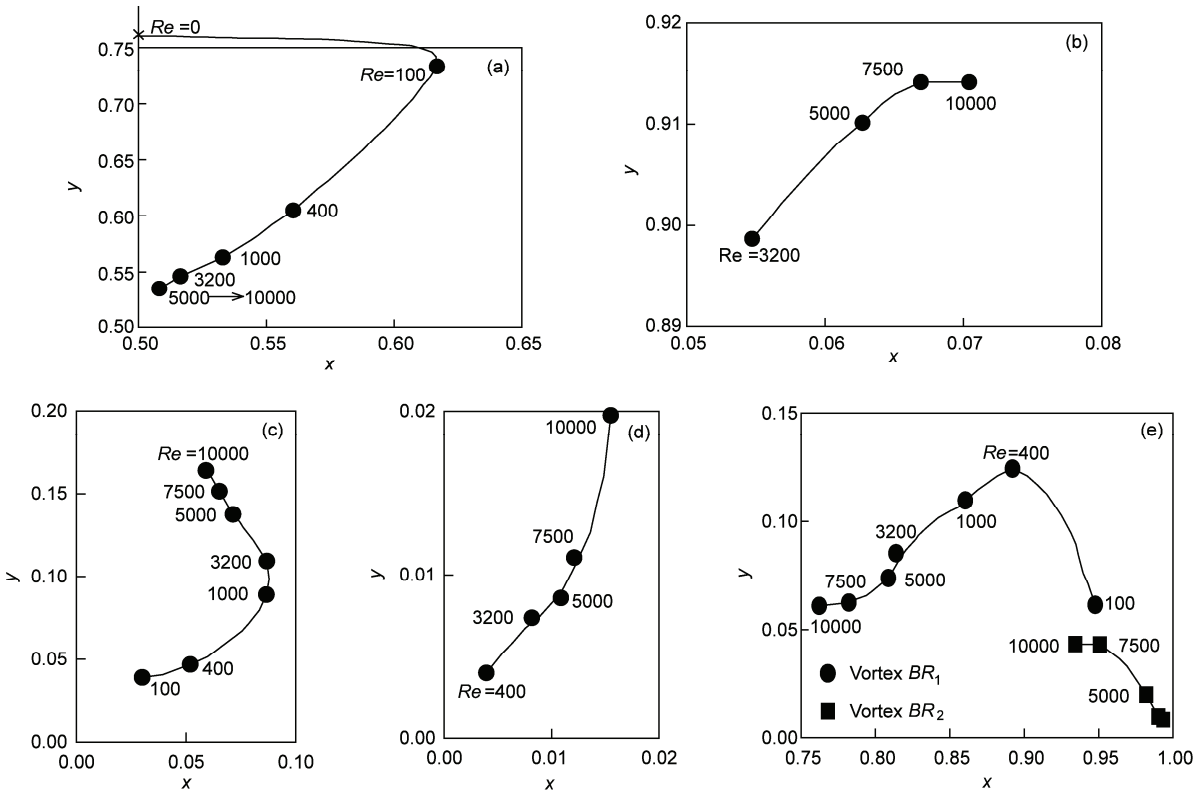

图 7　涡中心与 Reynolds 数之间的依赖关系[22]

坐标原点位于方腔左下角, x 和 y 的正方向分别对应计算区域的水平和垂直方向. (a) 主涡; (b) 二次涡 TL_1; (c) 二次涡 BL_1; (d) 二次涡 BL_2; (e) 二次涡 BR_1 和 BR_2

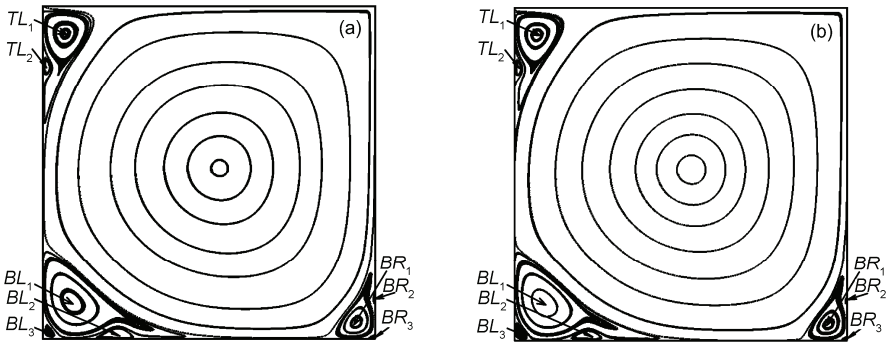

图 8　方腔流计算流线图(Re =10^5)

(a) 单个 GPU 计算结果; (b) 4 个 GPU 计算结果

295 单个和 4 个 GPU 计算不同算例的性能进行了比较. 因 GPU 特定的配置要求, 图中标定的是单 CPU 和单 GPU 的网格点数, 而多 GPU 对应的网格点数较之少 4×4 个.

CPU 计算采用 Intel icc 10.1.015 编译, 命令行选项为-fast, 其结果是我们在不同的编译器和编译选项下得到的最快的数值. 从图 9 中数据可以看出, 当网格点数较少时, 内存拷贝和结点间通信占用较多时间, 此时不能充分发挥 GPU+MPI 的计算性能; 而随着网格点数的增加, GPU 的多线程并行计算性能逐渐得到发挥. 特别是当 n=562×562 时, 相同算法发挥的单 GPU 计算能力达到 CPU 的约 53 倍之多, 而 4 个

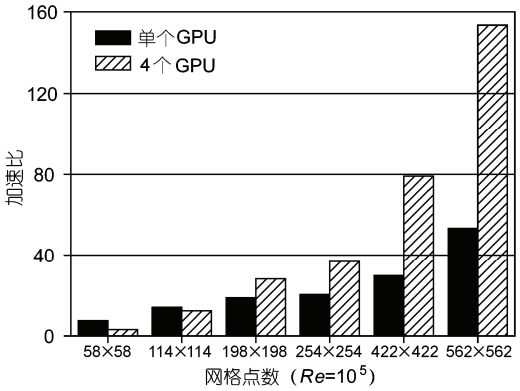

图 9 GPU 相对于 CPU 计算加速比与计算规模的关系图

GPU 可将计算能力提升至 150 多倍. 比较 CPU 与单 GPU 的计算结果发现, 除个别点(低于总网格点数的 0.5%)的相对差值低于 10^{-2} 外, 经过如此多次迭代后二者的最大相对差值($n\max\limits_{(x_i,y_j)\in\Omega}\left(\left|\phi_{i,j}^{\mathrm{C}}-\phi_{i,j}^{\mathrm{G}}\right|\right)\Big/\sum\limits_{(x_i,y_j)\in\Omega}\left|\phi_{i,j}^{\mathrm{C}}\right|$, Ω 表示计算区域; ϕ 指代所有因变量, 包括 u, v 和 p 等; 上标 C 和 G 分别标识 CPU 和 GPU; n 是网格点数)不高于 10^{-4}; 而单个 GPU 与 4 个 GPU 的计算结果几乎完全一致, 进一步说明了上述 GPU 直接数值模拟算法是精确可靠的.

4 网格与时间步长的影响

4.1 网格步长的影响

从以上各图的演化过程可以看出, 高雷诺数方腔流具有非常显著的多尺度结构特征. 首先着重考察网格点数(即网格步长)对计算结果的影响, 以雷诺数等于 10^4 为例, 图 10 中给出了单 GPU 计算流线图与网格点数 n 之间的依赖关系.

从图 6(a)和图 10 的变化过程可以看出, 在相同的模拟参数条件下随着网格点数 n 的减少, 所有二次涡, BR_1, BR_2, BL_1, BL_2, TL_1 的存在区域逐渐萎缩, 底部 4 个角涡的垂直高度也逐渐降低, 特别是当 n 降至 58×58 时, 方腔右下角处的二次涡, BR_2, 几近消失. 模拟结果充分说明, 为了复现方腔内可能存在的各种复杂结构, DNS 方法既要求计算区域大到足以包含最大尺度的涡, 又要保证计算网格小到足以分辨最小尺度的涡. 也就是说, 一旦计算网格大于某个二次涡的分辨尺度, 任何高精度算法都将无法“感知”它的存在, 这极有可能是文献中未给出方腔内完整多尺度结构的最根本原因.

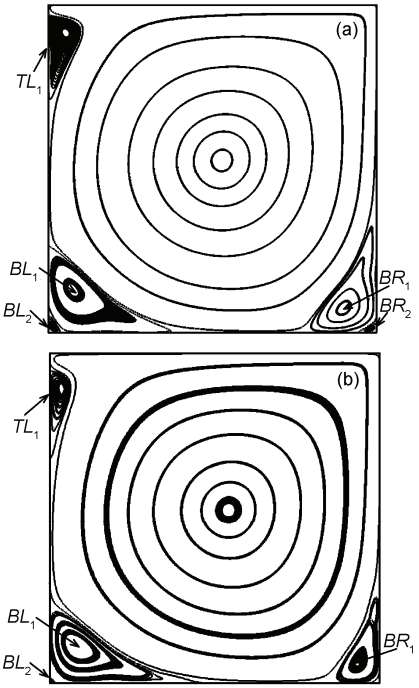

图 10 计算流线图与网格点数 n 之间的依赖关系

(a) n=114 × 114; (b) n=58 × 58

4.2 时间步长的影响

在其他操作参数不变的情况下, 我们通过扩大计算区域来增加系统的复杂程度. 在 NVIDIA GTX 295 的 4 个 GPU 上计算得到的雷诺数 Re=2 × 10^5 和 Re= 4×10^5 时的流场如图 11 所示. 从图 5(b), 图 6(b), 图 8(b)和图 11 的变化过程可以看出, 随着黏性作用的增强, 固体壁面附近的湍动现象越来越显著. 比如, 当雷诺数由 10^5 提升至 2 × 10^5 时, 图 8(b)中的两个角涡 TL_1 和 BL_1 的尾部逐渐脱落为图 11(a)中 3 个新的小涡 TL_3, BL_4 和 BL_5; 当雷诺数继续增至 4 × 10^5 时, 角涡 BR_1 的尾部脱落为图 11(b)中的小涡 BR_4, 而 TL_1 的尾部则进一步脱落为 TL_4, 并且在壁面附近衍生出许多小涡.

需要特别指出的是, 随着计算规模的不断扩大, 必须逐步减小时间步长方能得到图 11 中所示的较完整流线图. 具体的计算细节详见表 1.

1.1 节和 1.2 节中的计算结果充分说明, 网格与时间步长显著地影响着计算结果. 由于 N-S 方程形式复杂, 影响因素众多, 且在 SIMPLE 算法的压力修

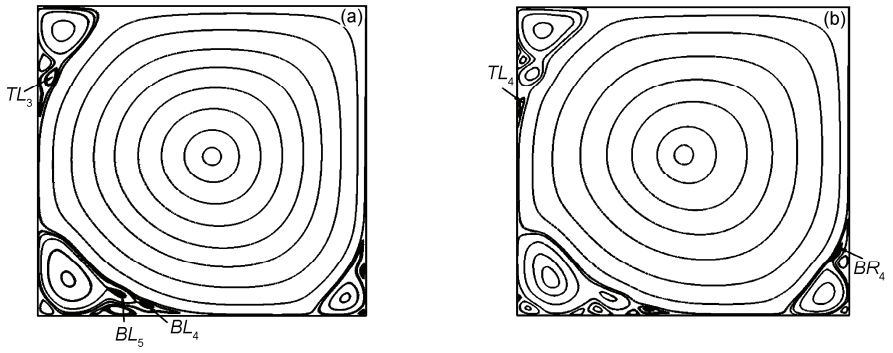

图 11　高雷诺数计算流线图

(a) $Re=2\times10^5$; (b) $Re=4\times10^5$

表 1　4 个 GPU 计算的具体细节

	计算区域	*Re*	网格点数	时间步长	迭代步数	运行时间/s
4 个 GPU	1.0×1.0	10^5	194×194	0.1	100000	31.3221
	2.0×2.0	2×10^5	418×418	0.01	1000000	682.956
	4.0×4.0	4×10^5	754×754	0.001	10000000	16895.9

正方程中忽略了 $\sum a_{\mathrm{nb}}u'_{\mathrm{nb}}$ 这一项, 因此该算法的空间与时间步长估计非常困难. 目前, 我们仅是通过试差的方法来调试程序.

5　结论

综上所述, 本文利用CUDA技术在GPU上实现了流体力学经典算例——高雷诺数方腔流的直接数值模拟, 获得了合理有效的计算结果. 分析表明, 基于交错网格 SIMPLE 算法的 GPU 程序是精确可靠的, 目前单个和 4 个 GPU 最高可分别发挥约 50 和 150 倍于主流 CPU 的计算能力. 并且高雷诺数方腔流具有显著的多尺度结构, 不仅计算网格必须小到足以分辨最小尺度的涡, 时间步长也要适当调整, 才能得到合理的计算结果. 毫无疑问, 大规模并行技术和平台的发展将大大提升复杂系统的计算能力与规模, 使得多尺度离散模拟的思想应用于传统的数值模拟领域变成现实.

致谢　感谢课题组王小伟、陈飞国、李博、徐骥等的大力帮助.

参考文献

1　NVIDIA. NVIDIA CUDA Compute Unified Device Architecture Programming Guide Version 2.0, 2008

2　Harris M J, Baxter W V, Scheuermann T, et al. Simulation of cloud dynamics on graphics hardware. In: HWWS'03. Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware. California: San Diego, 2003. 92—101

3　Krüger J, Westermann R. Linear algebra operators for GPU implementation of numerical algorithms. In: ACM Trans Graph (Proceedings of SIGGRAPH). California: San Diego, 2003. 908—916

4　Rumpf M, Strzodka R. Nonlinear diffusion in graphics hardware. In: Proceedings of EG/IEEE TCVG Symposium on Visualization VisSym'01. Berlin: Springer, 2001. 75—84

5　Bolz J, Farmer I, Grinspun E, et al. Sparse matrix solvers on the GPU: Conjugate gradients and multigrid. In: SIGGRAPH'03. California: San Diego, 2003. 917—924

6　Brandvik T, Pullan G. Acceleration of a 3D Euler solver using commodity graphics hardware. In: 46th AIAA Aerospace Sciences Meeting and Exhibit. AIAA, 2008. 1—15

7　Elsen E, LeGresley P, Darve E. Large calculation of the flow over a hypersonic vehicle using a GPU. J Comput Phys, 2008, 227: 10148—10161

8 Patankar S V, Spalding D B. A calculation procedure for heat, mass and momentum transfer in three-dimentional parabolic flows. Int J Heat Mass Transfer, 1972, 15: 1787—1806
9 Patankar S V. Numerical Heat Transfer and Fluid Flow. New York: Hemisphere Publishing Corporation, 1980
10 Patankar S V, 著, 张政, 译. 传热与流体流动的数值计算. 北京: 科学出版社, 1984. 131—157
11 Chen F G, Ge W, Li J H. Molecular dynamics simulation of complex multiphase flow on a computer cluster with GPUs. Sci China Ser B-Chem, 2009, 52: 372—380
12 李静海, 欧阳洁, 高士秋, 等. 颗粒流体复杂系统的多尺度模拟. 北京: 科学出版社, 2005
13 Shankar P N, Deshpande M D. Fluid mechanics in the driven cavity. Annu Rev Fluid Mech, 2000, 32: 93—136
14 Moffatt H K. Viscous and resistive eddies near a sharp corner. J Fluid Mech, 1964, 18: 1—18
15 Albensoeder S, Kuhlmann H C. Accurate three-dimensional lid-driven cavity flow. J Comput Phys, 2005, 206: 536—558
16 Hou S, Zou Q, Chen S, et al. Simulation of cavity flow by the lattice boltzmann method. J Comput Phys, 1995, 118: 329—347
17 Nie X, Robbins M O, Chen S. Resolving singular forces in cavity flow: Multiscale modeling from atomic to millimeter scales. Phys Rev Lett, 2006, 96: 134501
18 Qian T, Wang X P. Driven cavity flow: From molecular dynamics to continuum hydrodynamics. Multiscale Model Simul, 2005, 3: 749—763
19 Koplik J, Banavar J R. Corner flow in the sliding plate problem. Phys Fluids, 1995, 7: 3118—3125
20 肖曼玉, 欧阳洁, 李永刚. 基于区域分解的并行 SIMPLER 算法研究. 应用基础与工程科学学报, 2006, 14: 316—323
21 闫文辉, 张常贤, 陈宁宁, 等. 用 Gao-Yong 湍流方程组数值模拟高雷诺数顶盖驱动方腔流. 水科学进展, 2008, 19: 428—433
22 Ghia U, Ghia K N, Shin C T. High-Resolutions for imcompressible flow using the Navier-Stokes equations and a multigrid method. J Comput Phys, 1982, 48: 387—411

GPU accelerated direct numerical simulation with SIMPLE arithmetic for single-phase flow

WANG Jian[1], XU Ming[1,2], GE Wei[1] & LI JingHai[1]

[1] *State Key Laboratory of Multi-Phase Complex Systems, Institute of Process Engineering, Chinese Academy of Sciences, Beijing 100190, China;*
[2] *Graduate University of Chinese Academy of Sciences, Beijing 100049, China*

Based on SIMPLE arithmetic in a stageered grid system, Compute Unified Device Architecture (CUDA) was used to design and implement direct numerical simulation (DNS) on graphics processing units (GPU). High Reynolds number cavity flow was simulated on NVIDIA GTX 295, about 50 fold speedup of one GPU and 150 fold speedup of four GPUs over that of one core of the Intel Xeon 5430 CPU was achieved. The simulation results agreed with the literature data well. GPU accelerated DNS for high-accuracy and large-scale turbulent flow has a broad application prospect.

single-phase flow, direct numerical simulation, CUDA, GPU, parallel computing

doi: 10.1360/972009-1202