doi:10.11887/j.cn.201503013 http://journal.nudt.edu.cn

应用多GPU的可压缩湍流并行计算*

曹文斌,李 桦,谢文佳,张 冉
(国防科技大学 航天科学与工程学院,湖南 长沙 410073)

摘 要:利用CUDA Fortran语言发展了基于图形处理器(GPU)的计算流体力学可压缩湍流求解器。该求解器基于结构网格有限体积法,空间离散采用AUSMPW+格式,湍流模型为k-ω SST两方程模型,采用MPI实现并行计算。针对最新的GPU架构,讨论了通量计算的优化方法及GPU计算与PCIe数据传输、MPI通信重叠的多GPU并行算法。进行了超声速进气道及空天飞机等算例的数值模拟以验证GPU在大网格量情况下的加速性能。计算结果表明:相对于Intel Xeon E5-2670 CPU单一核心的计算时间,单块NVIDIA GTX Titan Black GPU可获得107~125倍的加速比。利用四块GPU实现了复杂外形1.34亿网格的快速计算,并行效率为91.6%。

关键词:CUDA;图形处理器;湍流;并行计算;计算流体力学
中图分类号:V211.3 **文献标志码**:A **文章编号**:1001-2486(2015)03-078-06

Parallel computation of compressible turbulence using multi-GPU clusters

CAO Wenbin, LI Hua, XIE Wenjia, ZHANG Ran
(College of Aerospace Science and Engineering, National University of Defense Technology, Changsha 410073, China)

Abstract: Based on CUDA Fortran for compressible turbulence simulations, a finite volume computational fluid dynamics solver on the GPU (Graphical Processing Unit) was developed. The solver was implemented with an AUSMPW+ scheme for the spatial dispersion, the k-ω SST model for turbulence model, and MPI communication for parallel computing. Some optimization strategies for fluxes computation and multi-GPU parallel algorithms for overlap of PCIe data transfer and MPI communication with GPU computation have been discussed for the latest generation GPU architecture. Several test cases, such as a supersonic inlet and a space shuttle were chosen to demonstrate the acceleration performance of GPU on large-scale grid size. Results show that when using a NVIDIA GTX Titan Black GPU, the computational expense can be reduced by 107~125 times than using a single core of an Intel Xeon E5-2670 CPU. Fast computing for a complex configuration with 0.134 billion grid sizes has been achieved by using 4 GPUs and the parallel efficiency is 91.6%.

Key words: CUDA; graphical processing unit; turbulence; parallel computing; computational fluid dynamics

随着硬件性能的提高及编程技术的改进,图形处理器(Graphical Processing Unit, GPU)加速器在高性能计算领域逐渐得到广泛的应用。在最新公布的超级计算机Top500名单中共有62套系统采用了加速器/协处理器,其中采用GPU加速器有46套,而在最新的Green500名单中前10位的超级计算机均采用了GPU加速器,由此可见GPU加速器在高性能计算中的优势。在通用计算中NVIDA GPU具有低成本、低功耗、高性能及易编程等优点,非常适合计算密集型应用。发布的CUDA并行编程架构[1]使开发人员能够在不了解图形学知识的条件下进行高级语言环境下的GPU编程以高效快速地解决许多复杂计算问题。目前在流体力学领域,已有大量的数值模拟方法针对GPU进行了改造,这其中有计算流体力学(Computational Fluid Dynamics, CFD)、格子玻尔兹曼、直接模拟蒙特卡洛、分子模拟等方法。

湍流一直是CFD领域的难点与热点,近年来,国内外已有许多学者将GPU计算应用到湍流的数值模拟当中,取得了良好的加速效果。Aaron[2]针对不可压缩湍流问题,基于Tesla C1060 GPU发展了大涡模拟求解器并给出了实现多GPU并行计算的方法。Cao与Xu等[3]应用高精度求解器在天河-1A系统上实现了多GPU与

* **收稿日期**:2014-10-07
基金项目:国家自然科学基金资助项目(91016010,91216117)
作者简介:曹文斌(1985—),男,湖南常宁人,博士研究生,E-mail:caowenbin08@163.com;
李 桦(通信作者),男,教授,博士,博士生导师,E-mail:lihuakd@tom.com;

多 CPU 的协同并行计算,讨论了提高异构系统并行效率的方法。Lin 等[4]基于 GeForce 560Ti GPU 发展了多块结构网格湍流求解器,利用 MPI 加 OpenMP 实现了多 GPU 的并行计算。Ali 等[5]在 Tesla S1070 多 GPU 集群上实现了湍流的直接数值模拟,分析了利用零拷贝技术与固定内存技术对提高系统效率的影响。可以发现,在现有的计算流体力学应用中,大多数研究针对的是计算能力较低的设备。而最新的设备(计算能力 3.5)更新了硬件架构,例如引入了 Hyper - Q,提升了显存带宽,增加了可用的寄存器个数等,这就需要研究人员根据最新架构的功能与特点对算法做出相应的调整与改进,以发挥设备的最佳性能,实现更加复杂更大计算规模任务的模拟。研究发展了基于 CUDA Fortran 的三维定常可压缩湍流求解器。

1　控制方程

当不考虑外部加热和彻体力的影响时,三维曲线坐标系下守恒形式的 Navier-Stokes 方程如下:

$$\frac{1}{J}\frac{\partial Q}{\partial t}+\frac{\partial(F-Fv)}{\partial \xi}+\frac{\partial(G-Gv)}{\partial \eta}+\frac{\partial(H-Hv)}{\partial \zeta}=0 \tag{1}$$

此处 Q 为守恒变量;J 代表曲线坐标系(ξ,η,ζ)与笛卡尔坐标系(x,y,z)之间的雅克比转换行列式;F,G,H 为曲线坐标系三个方向的无黏通量;Fv,Gv,Hv 代表黏性通量。考虑 $k-\omega$ SST 湍流模型时,除了添加两个包含源项的湍流方程之外,方程(1)的形式不变,此时黏性系数 μ 由层流黏性系数 μ_L 及涡黏性系数 μ_T 两部分组成,相应的 μ/pr 关系式如下:

$$\begin{cases}\mu=\mu_L+\mu_T\\ \dfrac{\mu}{pr}=\dfrac{\mu_L}{pr_L}+\dfrac{\mu_T}{pr_T}\end{cases} \tag{2}$$

式中,μ_L 通过 Sutherland 公式得到,μ_T 由湍流模型给出,pr_L 为层流普朗特数,对于空气可取 0.72,pr_T 为湍流普朗特数,一般取 0.9。当不考虑湍流模型时,涡黏性系数为 0。湍流模型的具体公式及参数设置见文献[6]。

对控制方程(1)运用基于单元中心型的有限体积法求解,无黏通量采用 AUSMPW + 格式[7],通过 MUSCL 方法选用 Vanleer 平均限制器对原始变量进行空间重构达到二阶精度,黏性通量采用高斯积分公式进行求解,时间推进为单步显式方法。

2　GPU 算法优化方法

由于 GPU 程序优化与硬件的体系结构及软件环境密切相关,因此本节首先给出计算平台配置情况。计算软件环境如下:CPU 代码用 Fortran 语言编写,GPU 代码为 CUDA Fortran 语言;两个版本的代码均由 PGI Fortran 13.10 编译器[8]加 OpenMPI 库编译,优化参数设置使两种设备的性能达到最佳;GPU 计算使用单精度。硬件配置如下:每个节点配置一块 CPU 与一块 GPU 及 64G 内存,节点间通信采用链路汇聚的千兆以太网络;运行的 CPU 为 Intel Xeon E5 - 2670@2.6 GHz,超线程关闭;测试使用的 GPU 为 6G 显存的 NVIDA GTX TITAN Black。

2.1　单 GPU 算法优化

CFD 迭代求解过程主要包括边界条件更新、通量计算、时间步长与源项计算及流场更新四个步骤。其中通量计算是决定求解器整体运行速度的主要因素,通常是优化的重点。不管是无黏通量还是黏性通量,在 GPU 上它们的计算都是首先从全局内存中输入网格单元的几何量和原始变量;其次通过空间格式得到单元界面通量;最后将通量或通量差结果输出至全局内存中。因此从优化方面考虑,可以将通量计算方法分为两类算法:输入优化(算法 1)与输出优化(算法 2)。为了分析对比,将不进行优化的算法定义为全局内存方法。

算法 1 的具体流程如下所示。为了节省全局内存开销,界面通量数组 f 被多次重复使用。

算法 1　输入优化

Alg. 1　Input optimization

内核 1:读入数据,计算 $F_{i-1/2}$,存入全局数组 f 中
内核 2:读入数据,计算 $Fv_{i-1/2}$,更新 $f=f-Fv_{i-1/2}$
内核 3:计算 $f_{i+1/2}-f_{i-1/2}$,存入全局数组 RHS 中
内核 4:读入数据,计算 $G_{j-1/2}$,存入全局数组 f 中
内核 5:读入数据,计算 $Gv_{j-1/2}$,更新 $f=f-Gv_{j-1/2}$
内核 6:更新 $RHS=RHS+f_{j+1/2}-f_{j-1/2}$
内核 7:读入数据,计算 $H_{k-1/2}$,存入全局数组 f 中
内核 8:读入数据,计算 $Hv_{k-1/2}$,更新 $f=f-Hv_{k-1/2}$
内核 9:更新 $RHS=RHS+f_{k+1/2}-f_{k-1/2}$

算法 1 中每个内核在读取数据阶段,均使用了下面介绍的优化方法。由于每个线程都需要用到相邻线程的信息,例如在计算 ξ 方向无黏通量

$F_{i-1/2}$时涉及 $i-2$, $i-1$, i, $i+1$ 共 4 个网格单元的原始变量值,若应用全局内存方法(将数据从全局内存直接读入寄存器)读取这些信息,将会产生大量的重复访问。为了提高带宽利用率降低访问延迟,常用的办法就是使用共享内存来缓解对全局内存的重复访问,具体使用方法见文献[9]。文献[10]给出了无黏通量计算的一种更高效的方法,即使用寄存器移位技术达到减少重复访问的目的,同时还给出了黏性通量的优化方法。本文求解器在算法 1 中采用文献[10]的方法进行优化。图 1 给了大小为 192^3 的网格在不同 GPU 架构下的通量计算时间对比,与全局内存方法相比,算法 1 在计算能力 2.0 的 Tesla C2050 上获得了 25.1% 的性能提升,而在计算能力 3.5 的 GTX TITAN Black 上仅有 8.3% 的提升。可以看出,对于重复数据的读取,在显存带宽有了显著提升并引入了 L2 缓存的计算能力 3.5 的设备上,使用优化加速技术并不能取得明显的加速效果。

算法 2 对结果输出做优化,具体的流程如下

算法 2 输出优化

Alg. 2 Output optimization

内核 1:读入数据,计算 $F_{i-1/2}$,存入共享内存,
计算 $F_{i+1/2}-F_{i-1/2}$,存入 *RHS*

内核 2:读入数据,计算 $Fv_{i-1/2}$,存入共享内存
计算 $Fv_{i+1/2}-Fv_{i-1/2}$,存入 *RHS*

内核 3:读入数据,计算 $G_{j-1/2}$,存入共享内存
计算 $G_{i+1/2}-G_{i-1/2}$,存入 *RHS*

内核 4:读入数据,计算 $Gv_{j-1/2}$,存入共享内存
计算 $Gv_{i+1/2}-Gv_{i-1/2}$,存入 *RHS*

内核 5:读入数据,计算 $H_{k-1/2}$,存入共享内存
计算 $H_{i+1/2}-H_{i-1/2}$,存入 *RHS*

内核 6:读入数据,计算 $Hv_{k-1/2}$,存入共享内存
计算 $Hv_{i+1/2}-Hv_{i-1/2}$,存入 *RHS*

可以看出,算法 2 在输出阶段利用共享内存消除了中间变量 f 的使用,这样既节省了全局内存消耗又避免了对全局变量的额外读写操作,同时所有内核不存在数据依赖性可并发执行。因为存储于网格中心的右端项 *RHS* 是通过两个界面通量做差值得到,所以每个线程块中计算得到的 *RHS* 个数少于线程块中的线程数,这样在每个线程块中除了需要引入新的分支语句,还需要在相邻线程块的边界处进行冗余计算。分支语句的增多与冗余计算的存在使得算法 1 中的输入优化技术在算法 2 中不但不能发挥应有的加速效果而且会引起性能的下降。因此算法 2 在读入数据阶段不进行优化,即应用全局内存方法读取数据。由图 1 可以看出,算法 2 非常适合计算能力 3.5 的 GTX TITAN Black,与全局内存方法相比,获得了 18% 的速度提升。

除了上述优化方法外,一些常规的优化方法在本文中均有应用,如全局变量按数组结构体方式储存,尽量避免束内分支,尽可能减少中间变量以降低寄存器使用量,减少 PCIe 数据传输,采用树形规约算法执行残值加和操作。所有计算配置的线程块大小为 $32\times8\times1$。

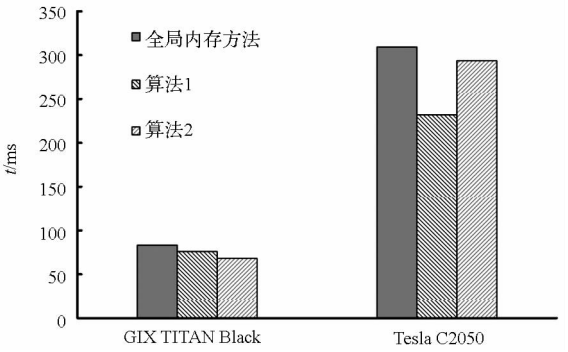


图 1 不同算法计算时间对比

Fig. 1 Computing time of different algorithms

2.2 多 GPU 并行算法

在求解器进行基于区域分解的并行计算过程中,相邻网格区域在边界处需要对原始变量进行数据交换。当多区网格位于同一块 GPU 中,此时为内部边界,可以在 GPU 中实现直接数据交换(求解器支持一个 MPI 进程处理多区结构网格);当相邻网格区域位于不同 GPU 时,需要先将一块 GPU 的边界数据通过 PCIe 总线传输至主机端,然后调用 MPI 进行交换,最后再通过 PCIe 传输至另一块 GPU。虽然已有 GPUDirect 技术实现 GPU 间的直接通信,但是现有系统的通信设备并不具有该功能,而升级设备成本又很高,因此提高现有系统的并行计算效率具有很大的实用价值。计算能力 3.5 的 GPU 引入了一个名为 Hyper-Q 的重要功能,它允许 32 个独立的 CUDA 流队列同时工作。利用该功能,可以实现 GPU 计算与边界数据交换的高度重叠。具体实现过程如图 2 所示。

首先,创建 N 个 CUDA 流($N\leqslant32$)。流 1 与流 2 分别用于所有区域的 ζ 方向内点无黏通量计算与内部边界更新等内核的启动。流 3 至流 N 负责所有需要进行数据交换的边界处理,这其中包括 GPU 上数据整理(将边界数据整理至预先分配好的连续储存空间中),设备端至主机端的异步传输(Device to Host, D2H),主机端至设备端

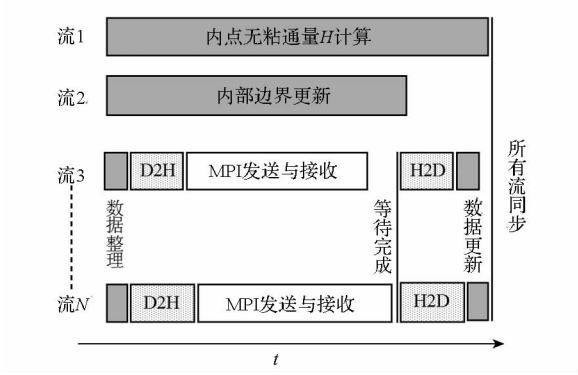


图 2　多 GPU 并行算法

Fig. 2　Multi-GPU parallel algorithm

的异步传输(Host to Device, H2D)及 GPU 边界数据更新。其次,所有流启动相应的内核执行,流 3 至流 N 的每个流在启动 MPI 非阻塞发送之前应用 CUDA 流查询函数确保 D2H 传输已完成,在启动 H2D 传输之前需等待 MPI 非阻塞接受完成。由于 Hyper-Q 特性允许流与流之间的操作互不阻塞,所以流 1 与流 2 对应的多个内核从一开始就与数据整理内核同时启动,内核的执行与 PCIe 传输及 MPI 数据交换同步进行,多个 D2H 或 H2D 传输可以同时存在, MPI 非阻塞通信与 D2H 传输在不同流之间重叠。最后调用流同步完成所有流的内核执行。图中流 1 虽然只包含了 ζ 方向的内点无黏通量计算,若它的计算时间小于数据交换所需要的时间,可以将 η 方向的相应计算加入流队列中,以实现 GPU 计算与边界数据交换的高度重叠。

3　算例分析

3.1　超声速进气道

Reinartz 等[11]对混合压缩进气道进行了大量的数值与实验研究,获得了清晰的流场结构与进气道壁面压力分布。图 3 所示为扩张段以前的进气道构型,上压缩面的压缩角 δ_2 为 21.5°,下楔面的倾斜角 δ_3 为 9.5°,扩张段的扩张角 δ_4 为 5°,隔离段的长度 l 为 79.3mm,高度 h 为 15mm,进气道总长为 400mm,宽为 52mm。

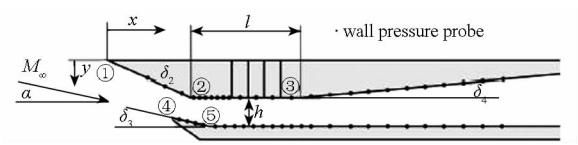


图 3　进气道模型示意图

Fig. 3　Schematic of the inlet model

进气道流动计算条件为:攻角 α 为 -10°,马赫数 M_∞ 为 2.41,总压 540kPa,总温 305K。由于实验时间较长,可认为壁面已达到热传导平衡状态,因此计算时壁面采用绝热条件假设。采用单区结构网格对进气道流场区域进行离散。为了测定 GPU 在不同网格量下的性能表现,设置了多套网格,其中最粗的网格由 256 × 48 × 64(流向 × 法向 × 展向)个网格点组成,在此基础上将前两个方向的网格点数量分别加密得到其余的网格,最密的网格其网格点个数达 5033 万(2048 × 384 × 64)。各套网格均在壁面处进行了加密。

图 4　计算得到的密度梯度图

Fig. 4　Density gradient magnitude of the computation

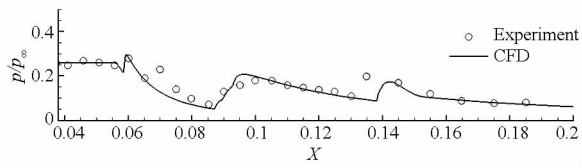


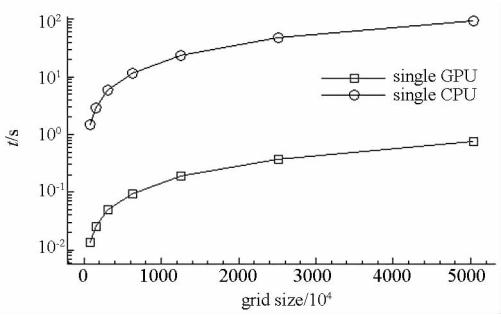
图 5　下表面压力分布

Fig. 5　Lower surface pressure distribution

图 4 给出了最密网格下湍流模型计算所得到的流场对称面密度梯度云图,可以发现数值计算结果清晰地捕获了进气道内复杂的波系结构。图 5 给出了进气道下壁面无量纲压力分布计算与实验结果的对比。从图中结果可以看出湍流模型的计算值与实验值在整体分布规律上都比较符合,这表明本文所发展的 GPU 求解器对可压缩湍流定常问题的模拟是可行的。

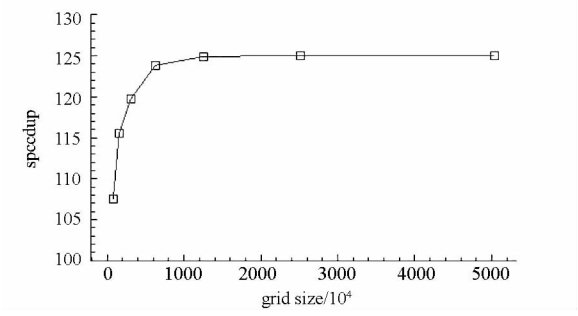
图 6 给出了不同网格下单块 GPU 与单个 CPU 核心的计算时间对比及相应的加速比。图中所示的时间为每迭代步所消耗的平均时间,可以看出 GPU 计算具有明显的速度优势,相对于单个 CPU 核心而言,随着网格量的增加,GPU 的加速比由最初的 107 倍逐渐增大至 125 倍。由加速比曲线可发现,小网格量下的加速比明显低于大网格量的情况,即 GPU 非常适合处理大网格量的计算任务。同时,随着网格量的增加,加速比趋于平缓,这是由于 GPU 中用于并行计算的 CUDA 核心数及可并发执行的线程块数是有限的,当 CUDA 核心的利用率达到饱和时一些线程将会采用串行方式执行,此时加速比将不再增加而趋于稳定。对于不同架构的设备,这个饱和值是不同的,可以看出,当前使用的 GPU 负载饱和值为 500

万以上的网格量。



(a) 计算时间

(a) Computing time



(b) 加速比

(b) Speedup

图 6 不同网格的计算时间与加速比

Fig. 6 Computing time and speedup for different grid sizes

并行效率是并行计算必须关注的性能指标。受显存容量的限制,单块 GPU 所能处理的最大网格量是有限的(对于定常湍流问题,6GB 显存至多可处理 5000 万网格),考虑到网格规模大小对 GPU 的性能发挥影响较大,为了真实反映同步与通信开销对并行效率的影响,本文定义并行效率如下:对于某个计算任务,假设它在多块 GPU 上并行处理的时间为 t_p,而对每个 GPU 来说都负载一定数量的网格(单个或多个分区),设所有 GPU 单独处理相应的负载所需时间的平均值为 t_s,则并行效率为 t_s 与 t_p 之比。上述的 GPU 单独处理既不考虑节点间通信又不考虑节点内通信。对于本算例,每块 GPU 的负载网格量都相等,且网格拓扑结构简单无节点内通信。表 1 给出了 4 块 GPU 在不同网格量下的计算时间与并行效率,可以看出,较小网格量下 GPU 的并行效率不高,随着网格量的增加 GPU 的并行效率逐渐提高。这是因为随着网格量的增加,节点间的通信时间在总的时间中所占比重逐渐减小所以并行效率得到提高。由表中数据可知,应用本文所发展的计算与通信的重叠算法,可使并行效率得到较大提高,相对于无重叠的并行计算,四套网格下内点通量计算与数据传输及通信的重叠可使并行效率提高 10% 以上。

表 1 计算时间与并行效率

Tab. 1 Computing time and parallel efficiency

网格大小	平均时间	无重叠并行时间/效率	有重叠并行时间/效率
768 × 128 × 64	24.8	40.2 / 61.7%	34.6 / 71.7%
1024 × 192 × 64	49.0	67.3 / 72.8%	57.4 / 85.3%
1536 × 256 × 64	94.9	121.0 / 78.4%	106.0 / 89.5%
2048 × 384 × 64	187.6	225.0 / 83.4%	198.6 / 94.5%

注:时间单位为 ms/迭代步。

3.2 超声速进气道

针对文献[12]所述的空天飞机模型,本文计算了马赫数为 4.94,攻角为 −5°的实验状态,对应的雷诺数为 5.26×10^7 m。空天飞机的长度为 0.29m,文献提供了高精度的实验结果。计算方法如前所述,按定常湍流状态考虑,壁面采用绝热条件假设。

图 7 所示为空天飞机的表面及对称面网格分布,网格总量为 1.34 亿,共有 26 个分区,壁面第一层网格高度为 1×10^{-6} m。采用 4 块 GPU 并行计算,每块 GPU 负载网格量为 3350 万左右。图 8 给出空天飞机的压力及流线分布,因外形复杂,流场中不可避免地会出现激波与激波干扰及流动分离等现象。

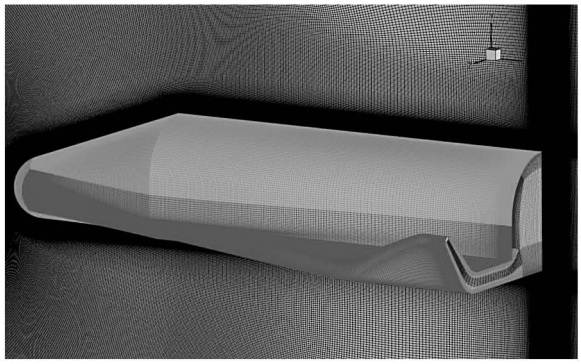


图 7 空天飞机计算网格

Fig. 7 Computational grid for the space shuttle

图 9 给出了机身对称面中心线上表面的压力分布,可以看出,计算结果与实验结果吻合较好,能够反映压力的变化趋势,这说明了求解器对复杂外形模拟具有很好的适应性。

4 块 GPU 并行计算时,每迭代步所消耗的时间为 0.667 秒,每块 GPU 单独处理的平均时间为 0.611 秒,计算得到的并行效率为 91.6%,与超声速进气道算例的最高并行效率相比有所降低。这

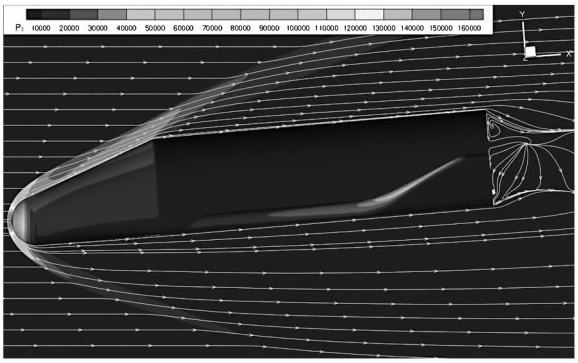


图 8　空天飞机压力云图及空间流线分布

Fig. 8　Pressure contours and streamlines distribution of the space shuttle

主要是因为单块 GPU 负载的网格分区过多(尾流场包含 18 个分区,分配在两个 GPU 上)引起同步时间消耗增多。此外,负载不平衡也造成了一定的影响,虽然 4 个 GPU 所负载的网格量基本相等,但是内部边界及物理边界条件(如壁面条件、远场条件等)更新个数并不相等,分区过多导致了节点内的边界更新操作过多从而影响了整体并行效率。

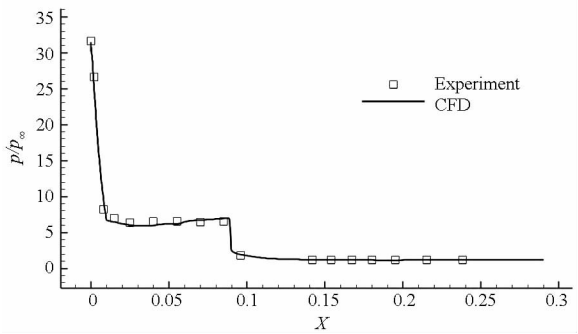


图 9　空天飞机对称面上表面的壁面压力分布

Fig. 9　Pressure coefficient distribution along the upper surface in the symmetry plane

4　结论

本文建立的基于 GPU 的可压缩湍流求解器,可以快速准确地对超声速流动中的湍流问题进行求解。为了达到良好的加速效果,根据最新的 GPU 架构特点对 GPU 程序的算法结构进行了调整与优化,采用两个超声速流动算例验证了 GPU 求解器的准确性与适应性,实现了多 GPU 上复杂外形上亿量级网格的快速计算,在最新的 GTX TITAN Black GPU 上取得了可观的加速比与并行效率。根据研究结果,得出以下结论:

1)不同架构的 GPU 对应不同的优化算法,对于重复数据的读取,在计算能力 3.5 的设备上,使用共享内存或寄存器移位等优化加速技术并不能取得明显的加速效果。

2)利用 Hyper－Q 特性,可以实现 GPU 计算与 PCIe 数据传输、MPI 通信的高度重叠,能有效掩盖边界交换所带来的时间延迟。

3)从 GPU 的性能发挥及并行效率方面考虑,GPU 负载的网格量应越大越好。对于本文使用的 GTX TITAN Black GPU,最佳的负载网格量应在 500 万以上。

参考文献(References)

[1] NVIDIA Corporation. NVIDIA CUDA C Programming Guide Version6. 5 [EB/OL]. [2014 － 08 － 01]. http://docs.nvidia.com/cuda/pdf/ CUDA-C-Programming-Guide.pdf.

[2] Shinn A F. Large eddy simulations of turbulent flows on graphics processing units: application to film cooling flows[D]. USA: University of Illinois at Urbana Champaign, 2011.

[3] Cao W, Xu C F, Wang Z H, et al. CPU/GPU computing for a multi-block structured grid based high-order flow solver on a large heterogeneous system[J]. CLUSTER COMPUT, 2014, 17(2):255－270.

[4] Fu L, Gao Z H, Xu K, et al. A multi-block viscous flow solver based on GPU parallel methodology [J]. Computers & Fluids, 2014, 95:19－39.

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

[6] Menter F R. Influence of freestream values on k－ω turbulence model prediction [J]. AIAA Journal, 1993, 30(6): 1657－1659.

[7] Kim K H, Kim C, Rho O H. Methods for the accurate computations of hypersonic flows I. AUSMPW + scheme[J]. Journal of Computational Physics, 2001, 174(1): 38－80.

[8] The Portland Group. CUDA Fortran Programming Guide and Reference Release 2013, version 13.10[EB/OL]. [2013]. http://www.pgroup.com/ resources/docs.htm.

[9] Ruetsch G, Fatica M. CUDA fortran for scientists and engineers: best practices for efficient CUDA fortran programming[M]. Holland:Elsevier, 2013.

[10] Salvadore F, Bernardini M, Botti M. GPU accelerated flow solver for direct numerical simulation of turbulent flows[J]. Journal of Computational Physics, 2013, 235:129－142.

[11] Reinartz B U, Herrmann C D, Ballmann J, et al. Aerodynamic performance analysis of a hypersonic inlet isolator using computation and experiment[J]. Journal of Propulsion and Power, 2003, 19(5): 868－875.

[12] 李素循. 典型外形高超声速流动特性[M]. 北京:国防工业出版社, 2007.
LI Suxun. Hypersonic flow characteristics around typical configuration [M]. Beijing: National Defense Industry Press, 2007. (in Chinese)