异构平台下格子 Boltzmann 方法实现及性能分析

张丹丹 徐 莹 徐 磊
(上海超级计算中心 上海 201203)

摘 要 对CPU+GPU异构平台下的多种并行编程模式进行了研究,并针对格子 Boltzmann 方法实现了 CUDA,MPI+CUDA,MPI+OpenMP+CUDA 多级并行算法。结果表明,算法具有较好的加速性能;提出的根据计算量比例参数调节 CPU 和 GPU 之间负载均衡的方法,对于在异构平台上实现多级并行处理及资源的有效利用具有一定的参考和应用价值。

关键词 异构平台,GPU,格子 Boltzmann,并行
中图法分类号 TP31 **文献标识码** A

Performance Analysis and Implementation of Lattice Boltzmann Methods on Heterogeneous Platform

ZHANG Dan-dan XU Ying XU Lei
(Shanghai Supercomputer Center, Shanghai 201203, China)

Abstract This work investigated the implementation of lattice Boltzmann method(LBM) with CUDA on the CPU+GPU heterogeneous platform with multiple parallel programming models, MPI+CUDA and MPI+OpenMP+CUDA, which shows good performance speedup. A method used to justify the computational load ratio was proposed in this paper to balance the computational time on CPU and GPUs, which provides insightful information about the multi-level parallelization and efficient usage of different computational resource on the CPU+GPU heterogeneous platform.

Keywords Heterogeneous platform, GPU, LBM, Parallel

在高性能计算应用时代,大规模计算流体力学越来越依赖于计算科学和计算技术的发展。集群环境下多节点或单节点多核并行 CPU 计算已广泛用于高性能计算(HPC)应用。而自 1993 年开始,GPU 的性能便以每年 2.8 倍的速度高速增长,GPU 性能的迅速提升及 HPC 相关技术的发展给 HPC 领域带来了新的机遇和挑战。随着 NVIDIA CUDA[1](Compute Unified Device Architecture)的发布,GPU 在浮点运算能力、带宽和更好的可编程性等方面的优势,已经被应用于通用计算领域,即 GPGPU(General-Purpose Graphics Processing Unit)。2011 年 6 月的世界超级计算机 TOP500[2] 中,使用加速部件的有 19 台系统,比上一届的 5 台提高了约 3 倍。

HPC 领域中许多科学计算应用被移植到 GPU 上,并获得了更高性能,但很少有大规模的计算流体力学(CFD)应用,原因在于其大多难以满足内存访问需求少、计算密集性强的特点。当前,已经有 CFD 中的应用移植到 GPU 平台上,并获得了 10~100 倍的加速性能[3-6]。

格子 Boltzmann 算法,因其计算简单,适合处理复杂边界和易于并行,现已成功地应用在许多流体问题的模拟和建模方面[7]。由于 GPU 拥有更多的计算单元,而格子 Boltzmann 方程只在边界点上存在弱相关性,非常适合于 GPU 擅长的单指令多线程 SIMT(Single-Instruction Multiple-Thread)并行。目前,LBM 在单 GPU/多 GPU 平台上已获得高性能,D3Q19 LBM 模型 Li et al. 在单 GPU 上获得了 15.3 的加速比和 3.8 MLUPS(Mega Lattice-Updates Per Second)[8],Fan et al. 在多 GPU 上获得了 21.4 倍的加速[9]。

当前,GPU 设备在 HPC 领域的应用往往与 CPU 结合作为集群环境的计算节点。GPU 的加入,使得系统单节点的计算能力得到大幅度提升,但往往这时会出现一个问题,那就是 CPU 资源得不到充分利用。如何既能发挥 GPU 的计算优势,又使得 CPU 资源合理利用,成为大规模并行急待解决的问题。本文以 LBM D3Q19 为例,在多路、多核、多 GPU 异构平台下通过多种并行编程模式对应用实现多级并行,并对异构平台下计算资源的有效利用问题进行了探索。

本文第 2 节介绍格子 Boltzmann 方法;第 3 节介绍 CPU+GPU异构平台下多层次并行实现;最后为结束语。

1 格子 Boltzmann 方法

格子 Boltzmann 算法是一种模拟流体流动的数值方法,它不再基于连续介质假设,而是把流体看成许多只有质量没有体积的微粒组成,这些微粒可以向空间的若干方向任意流动。它的主要思想就是以简单规则的微观粒子运动代替复杂多变的宏观现象。粒子在每个时间步的运动由两个子步构成,即:

(1)碰撞(collision):当多个粒子到达同一网格结点时,它

到稿日期:2011-07-21 返修日期:2011-10-12 本文受国家高技术研究发展计划项目(2009AA012201),上海市科委科研计划项目课题新型计算结构与应用产业技术创新战略联盟(08dz1501600),上海浦东人才计划项目(10PJ1430600)资助。
张丹丹(1978-),女,硕士,工程师,主要研究领域为高性能计算、系统测评及分析、GPU 计算,E-mail:ddzhang@ssc.net.cn;徐 莹 女,博士,副高,主要研究方向为计算机流体力学、GPU 计算等;徐 磊(1982-),男,主要研究方向为并行计算、高性能系统评测、GPU 应用移植及优化。

们按碰撞规则相互作用并改变各自的速度方向；

(2)流动(evolution)：每个粒子按其速度方向移动到最近的网格结点。

1992年，Y H Qian(钱跃竑)等人提出格子 Boltzmann 算法的单松弛模型，即格子 BGK(Lattice Bhatnagar-Gross-Krook，LBGK)模型[10]。目前，LBGK 模型是格子 Boltzmann 算法中最主要的，也是应用最广泛的模型。LBGK 模型抛弃了 Fermi-Dirac 分布，采用 Maxwell 分布，使得各向同性、Galilean 不变性和压力独立于流速等问题都得到满足。Qian 等将格子 Boltzmann 方程中的碰撞项完全线性化，用单松弛时间逼近碰撞项，其对应的格子 Boltzmann 演化方程为：

$$f_\alpha(\mathbf{x}+\mathbf{e}_\alpha,t+1)=f_\alpha(\mathbf{x},t)+\omega(f_\alpha^{eq}(\mathbf{x},t)-f_\alpha(\mathbf{x},t)) \quad (1)$$

式中，ω 为松弛因子，为了达到良好的稳定性，必须在 0 到 2 之间，小于 1 为亚松弛，大于 1 为超松弛；f_α 为单粒子分布函数，表示在 t 时间上，x 格点上沿 $\mathbf{e}_\alpha$ 运动有 1 个粒子的概率；$\mathbf{e}_\alpha$ 称为粒子运动方向或离散速度方向；f_α^{eq} 为平衡态分布函数，是流体动力学量(如密度，动量等)的函数。

将上式转化一下，得到：

$$f_\alpha(\mathbf{x}+\mathbf{e}_\alpha,t+1)-f_\alpha(\mathbf{x},t)=\omega(f_\alpha^{eq}(\mathbf{x},t)-f_\alpha(\mathbf{x},t)) \quad (2)$$

方程右侧即为碰撞项，也称 BGK 碰撞项(由 Bhatnager. Gross. Krook 提出)。方程左侧为流动步。由此方程可以看出，碰撞过程只与本节点有关，为局部性计算过程；流动步也仅与相邻的网格点之间进行数据交换。

构造 LBGK 模型的关键是选择恰当的平衡态分布函数 f_α^{eq}。一旦选定了离散速度 $\mathbf{e}_\alpha$，只须确定出权系数 t_p 就可得到平衡态分布函数。DnQb 系列模型的分布函数为：

$$f_\alpha^{eq}(\mathbf{x},t)=t_p\rho\left\{1+\frac{\mathbf{e}_\alpha\cdot\mathbf{u}}{c_s^2}+\frac{(e_\alpha\cdot\mathbf{u})^2}{2c_s^4}-\frac{\mathbf{u}^2}{2c_s^2}\right\} \quad (3)$$

式中，c_s 为声速，t_p 对应粒子速度的权系数，$\mathbf{u}$ 为宏观速度，ρ 为宏观密度。

为了保证得到正确的宏观方程，在选择这些权系数时，应当使 f_α^{eq} 满足质量守恒、动量守恒和各向同性等约束，即：

$$\begin{cases}\sum_i f_i(\mathbf{x},t)=\sum_i f_i^{eq}(\mathbf{x},t)=\rho(\mathbf{x},t)\\ \sum_i c_i f_i(\mathbf{x},t)=\sum_i c_i f_i^{eq}(\mathbf{x},t)=\rho(\mathbf{x},t)\mathbf{u}(\mathbf{x},t)\\ \sum_i c_{i\alpha}c_{i\beta}f_i^{eq}(x,t)=p\delta_{\alpha\beta}+\rho u_\alpha u_\beta\end{cases} \quad (4)$$

式中，$p=c_s^2\rho$。

DnQb 系列模型的粘性系数为：

$$\nu=\frac{c_s^2}{2}\left(\frac{2}{\omega}-1\right) \quad (5)$$

对于本文所选用的 D3Q19 模型，其网格结构如图 1 所示。权系数选取如下：

$$t_0=\frac{12}{36},t_1=\frac{2}{36},t_2=\frac{1}{36},c_s^2=\frac{1}{3}$$

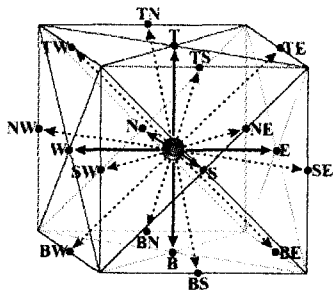


图 1 D3Q19 模型的网格结构

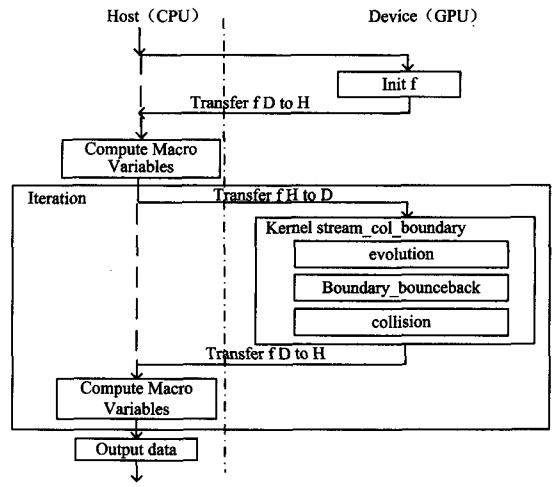
2 CPU+GPU 异构平台下 LBM 多级并行实现

2.1 CPU+GPU 多级并行实现

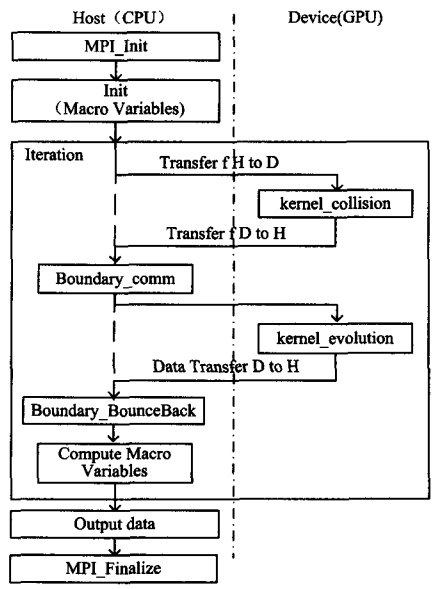
CPU+GPU 异构平台可分为单计算节点和集群系统两种。在高性能计算领域，GPU 一般作为加速部件存在于计算节点中。在这样的异构集群系统中实现并行计算，可经过 3 种不同粒度的并行加速：GPU 硬件并行、线程级并行和节点级并行。3 种层次的并行分别用到的并行计算技术为 CUDA 或 OpenCL，OpenMP，MPI。CPU+GPU 混合编程模型支持 3 种编程方式：OpenMP + CUDA，MPI + CUDA，Hybrid MPI + OpenMP+CUDA。MPI 是一种消息传递型并行通信的程序设计标准，其定义了一组消息传递函数，用来实现节点级并行。OpenMP 是支持共享存储并行编程的工业标准，其提供了一种与平台无关的编译制导指令、编译指示符、函数调用和环境变量，用来实现节点内线程级多 GPU 之间或多核之间的并行。

2.2 LBM D3Q19 多级并行实现

LBM 的显著特点之一就是算法简单，粒子的运动只需要碰撞和流动两种简单运算便可描述，其中碰撞过程是一种单一时间松弛过程，粒子以单一的松弛时间趋于平衡态，碰撞过程完全局部化。同时，整个算法是简单的显示迭代过程，涉及到的通信仅仅发生在边界。



(a)CUDA 实现流程



(b)MPI+CUDA 实现流程

图 2 LBM D3Q19 CUDA 及 MPI+CUDA 实现流程

在各种流动情况中，顶盖驱动方腔流以其简单的几何形状和良好的漩涡结构，一直作为测试和评估计算复杂流体问题的数值方法的基准(Benchmark)，而且方腔流很容易转化到其他复杂流体，如前、后台阶流等。这里，为了能更好地衡量性能，文中以三维顶盖驱动方腔流为例，采用 D3Q19 模型进行实现，固壁上使用无滑移完全反弹边界条件，具体采用了单 GPU CUDA 实现，MPI+CUDA 实现，MPI+OpenMP+CUDA 实现。

单 GPU CUDA 实现将碰撞、移动及边界处理均放到 GPU 上完成，程序流程如图 2(a)所示。

D3Q19-LBM CUDA 实现中迭代过程由若干个 CUDA Threads 并发完成，从图 2 中可以看出，在 GPU 进行计算的同时，CPU 处于长时间的等待之中，如图中的虚线部分。且要充分利用多 GPU 的优势需实现多 GPU 直接的通信，这里采用 MPI 进行实现。假设同时运行 np 个 MPI 进程，每个进程的 ID 号定义为 myid；GPU 设备的数量为 DevNum，GPU 设备的 id 号对应 MPI 进程定义为 myid mod DevNum，假设 DevNum=2，其对应关系如图 3 所示。

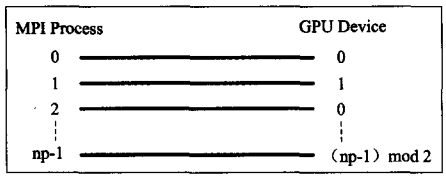


图 3　DevNum=2 时 MPI 进程和 GPU 设备之间的对应关系

集群环境中可通过 MPI+CUDA 实现跨节点多 GPU 并行，多路、多核 CPU 资源的利用则采用在节点内部的 Hybrid MPI+OpenMP+CUDA 编程模式，如图 4 所示。由于 LBM 算法的特殊性，碰撞过程完全局部化，在采用 Hybrid 实现时，引入参数 $fraction(0 \leqslant fraction \leqslant 1)$，一方面保证了应用性能，另一方面对 CPU 和 GPU 中碰撞和移动计算进行负载均衡处理，使得 CPU 和 GPU 都得到充分利用。$fraction$ 定义如下。

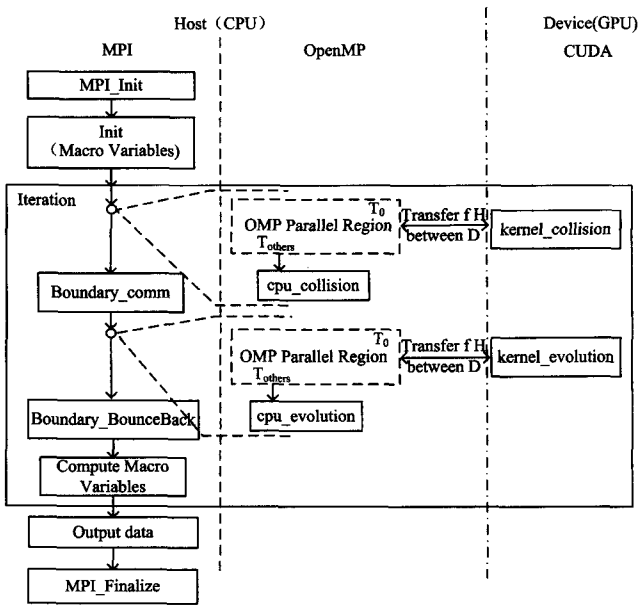


图 4　Hybrid MPI+OpenMP+CUDA LBM D3Q19 实现流程

$$fraction=\begin{cases}0, & \text{所有计算在 CPU 上执行}\\ 1, & \text{所有计算在 GPU 上执行}\\ \text{其他}, & \text{CPU 和 GPU 各承担一部分计算}\end{cases}$$

对碰撞 collision 和移动 evolution 过程进行处理时，使用 OpenMP 的线程 0 调用 GPU 设备，其他线程调用 CPU 上执行的碰撞和移动过程，CPU 和 GPU 执行计算的比例由参数 $fraction$ 决定。一个 MPI 进程对应若干 OpenMP 线程，线程 0(图中 T_0)对应一个 GPU Device，其他线程(T_{others})调用 CPU 端程序。图 5 给出了 MPI 进程 P_i 和 OpenMP 线程 T_i 及设备 Dev_i 之间的隐射关系。

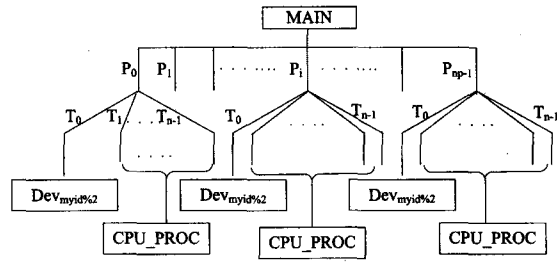


图 5　Hybrid 编程模式下进程、线程与设备之间的隐射关系

2.3　测试结果及分析

本文测试平台单计算节点采用的是曙光 A950r-F 服务器。节点配置为 8 路四核的 AMD Opteron 1.9GHz HE 和 128GB 内存，并通过 PCI-E x16 连接了 2 块 NVIDIA S2050。操作系统为 SLES11 sp1，CUDA Driver、CUDA Toolkit 和 CUDA SDK 均采用最新 4.0 版本。

为了验证程序的正确性，采用 D3Q19 模拟三维后台阶流，并与 Armaly 等人[11]的实验结果进行了比较，结果证明，它与实验值吻合较好，如图 6 所示。

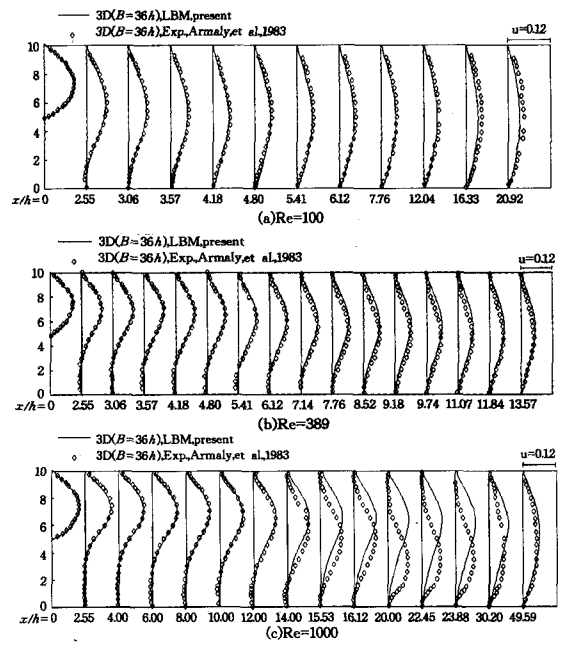


图 6　三维后台阶流与实验值槽道对称面上的速度剖面

Wang Xian[11]对 D3Q19-LBM 模拟顶盖驱动方腔流 96*96*96 个网格点，在 NVIDIA Geforce GTX280 平台下实现了 87 倍的加速和 270MFLUPS(Million fluid lattice cell updates per second)(Xeon E5420 2.5GHz)。本文实现了约 100 倍的加速和 220MFLUPS。

表 1 和表 2 中分别给出了 MPI+CUDA 及 Hybrid MPI+OpenMP+CUDA 实现时计算、通信及 Host 和 Device 之间数

(下转第 303 页)

[16] 王炜,乔林,杨广文,等. 二维片上网络局部均匀随机通信性能分析[J]. 计算机研究与发展,2010,47(3):532-540

[17] Denneau M M. The Yorktown simulation engine [C]// Proceedings of the 19th Design Automation Conference. Piscataway, NJ:IEEE,1982:55-59

[18] Broomell G,Heath J. An integrated-circuit crossbar switch system design[C]//Proceedings of the 4th International Conference on Distributed Computing Systems(ICDCS 1984). Los Alamitos,CA:IEEE Computer Society,1984:278-287

[19] Faraj A, Yuan Xin. Communication characteristics in the NAS parallel benchmarks[C]// Proceedings of International Conference on Parallel and Distributed Computing Systems(PDCS 02). Cambridge:IASTED/ACTA Press,2002:724-729

[20] Vetter J F,Mueller F. Communication characteristics of large-scale scientific applications for contemporary cluster architectures[J]. Journal of Parallel and Distributed Computing,2003,63(9):853-865

[21] Tutsch D, Lüdtke D. Chip multiprocessor traffic models providing consistent multicast and spatial distributions[J]. SIMULATION,2008,84(2/3):61-74

[22] Pande P P,Grecu C,Ivanov A,et al. Design,Synthesis,and Test of Networks on Chips[J]. IEEE Design & Test of Computers,2005,22(5):404-413

[23] Jin Y,Kim E J,Yum K H. A domain-specific on-chip network design for large scale cache Systems[C]// Proceedings of the 13st International Conference on High-performance Computer Architecture(HPCA 2007). Los Alamitos,CA:IEEE Computer Society,2007:318-327

(上接第 298 页)

据传输所占的比例。从数据可以看出,在使用 Hybrid 方式之后很大程度上提高了计算所占的比例,同时也减少了因为 GPU 和 CPU 之间数据传输所占用的时间。

表 1 D3Q19-LBM MPI+CUDA 实现的计算与通信所占比例

	64	96	128	192	240
comp(%)	8.64%	10.19%	11.23%	11.49%	11.29%
mpi(%)	2.53%	1.85%	1.48%	1.13%	1.01%
memcpy(%)	88.83%	87.95%	87.29%	87.38%	87.70%

表 2 D3Q19-LBM Hybrid 实现的计算与通信所占比例

	64	96	128	192	240
comp(%)	46.68%	52.05%	52.56%	53.92%	52.41%
mpi(%)	3.37%	3.21%	2.08%	1.74%	1.47%
memcpy(%)	38.59%	36.42%	36.14%	34.78%	34.32%

MPI+CUDA 实现中绝大多数时间用在了 Device 和 GPU 直接的数据传输,性能上相比 MPI 的 D3Q19-LBM 实现提升了 2 倍,而 Hybrid 实现则提升了约 5 倍。这里,Hybrid 运行时,在单计算节点采用的是 2 个 MPI 进程,分别对应 2 个 GPU Device,每个进程设置的 OpenMP 线程数为 4,重复利用 4 核的计算性能。

参数 *fraction* 的数值也会影响到 Hybrid 的程序性能。从表 3 可以看出,对于碰撞过程来说,在 GPU 上的计算比例大更能获得高性能,但增加 GPU 上计算的同时,也会增加 GPU 和 CPU 之间的数据传输量和数据传输时间。移动过程由于涉及到边界上的通信过程,因此需要更多地利用到 CPU,以减少与 CPU 或 GPU 之间的数据传输,更好地提高性能。

表 3 fraction 对性能的影响

	collision_time%	evolution_time%
fraction=0.25	38.72%	35.16%
fraction=0.35	40.06%	39.94%
fraction=0.5	38.08%	45.56%
fraction=0.75	41.75%	43.97%

结束语 本文对 D3Q19-LBM 在异构平台上采用了多层次并行模式进行实现,考查了在 CPU+GPU 平台上应用的性能表现,并初步探索了 CPU 和 GPU 直接的负载均衡问题。从试验结果来看,Hybrid MPI+OpenMP+CUDA 能很大程度上提高应用性能,改善计算和通信时间比;在 GPU 和 CPU 之间可以通过计算量的配比参数来调节负载,并进一步优化性能。

异构平台上应用的实现及性能表现与应用的特点紧密相关。应用的计算及通信特征决定了其并行性,LBM 是 CFD 中通信模式比较简单的计算方法,CFD 在异构平台下的性能研究还有待进一步深入。文中对 D3Q19 的 MPI+CUDA 及 Hybrid 实现的性能还有待进一步的优化。

参 考 文 献

[1] nVIDIA. http://www.nvidia.com/object/cuda_home_new.html

[2] TOP 500 List. http://www.top500.org/lists/2011/06

[3] Stone J E, et al. Accelerating molecular modeling applications with graphics processors[J]. Journal of Computational Chemistry,2007,28(16):2618-2640

[4] Ogaa S,et al. GPU computing for 2-dimensional incompressible-flow simulation based on mulit-grid method[J]. Transactions of JSCES,Paper No. 20090021,2009

[5] HaraDa T,et al. Smoothed particle hydrodynamics on GPUs[C]// Proceeding of the Spring Conference on Computer Graphics. 2007:235-241

[6] Rossinelli D, et al. GPU accelerated simulation of bluff body flows using vortex particle methods[J]. Journal of Computational Physics,2010,229:3316-3333

[7] 郭照立,等. 格子 Boltzmann 方法的原理及应用[M]. 北京:科学出版社,2009

[8] Li W,et al. GPU-based Flow Simulation with Complex Boundaries[M]. GPU Gems2,Adison-Wesley,2005:747-764

[9] Fan Z,et al. GPU cluster for high performance computing[C]// SC'04:Proceedings of the 2004 ACM/IEEE Conference on Supercomputing. IEEE Computer Society,Washington,DC,USA,2004:47

[10] Qian Y,Succi S,Orszag S. Recent advances in lattice Boltzmann computing[J]. Ann. Rev. Comp. Phys. ,1995,3:195-242

[11] Armaly B,Durst F,Rereira J,et al. Experimental and theoretical investigation of backward facing step[J]. J. Fluid Mech. ,1983,127:473-496

[12] Wang Xian, et al. Multi-GPU performance of incompressible flow computation by lattice Boltzmann method on GPU cluster [Z]. Parallel Comput,2011