

基于 GPU 并行算法的水动力数学模型建立及其效率分析

赵旭东¹, 梁书秀^{*1}, 孙昭晨¹, 刘忠波^{1,2}, 韩松林¹, 任喜峰¹

(1. 大连理工大学 海岸和近海工程国家重点实验室, 辽宁 大连 116024;

2. 大连海事大学 交通管理学院, 辽宁 大连 116026)

摘要: 应用非结构化网格建立水动力模型目前已经得到了广泛的应用. 针对在网格数过多, 且无集群机情况下难以快速获得计算结果这一问题, 基于 GPU 的高性能计算技术, 在 CUDA 开发平台下设计并行算法, 建立非结构化网格的二维水动力模型. 与利用 GTX460 显卡和集群机的计算效率对比表明, 在保持计算精度的前提下, 速度提升了一个量级, 且随着网格数的持续递增, 可以保持较高的加速比增幅, 比较适合应用于大范围海域的水动力模型的数值计算.

关键词: GPU; 非结构化网格; 水动力模型

中图分类号: TP391.75

文献标识码: A

doi: 10.7511/dllgxb201402008

0 引言

海洋水动力模型的发展, 一方面要求实现不同动力因素的耦合, 如目前风-浪-潮耦合模型的需求, 这将导致计算区域加大; 另一方面需要对所关心区域不同尺度物理现象的精确模拟, 这要求局部网格较小. 这两方面的要求使采用非结构化网格模拟计算大范围海域的水动力模型成为一种趋势. 但是网格密度和计算区域的增大将导致网格数量变得十分庞大, 在没有集群机或在集群机计算条件受限情况下, 单机计算时间过长, 难以在较短时间内获得计算结果, 在一定程度上限制了非结构化网格水动力模型在工程领域上的应用.

GPU(graphic processing unit, 图形处理器)与 CPU(central processing unit, 中央处理器)的计算能力的差别主要源于二者的计算架构不同, 一般的处理器由控制单元、逻辑单元和存储单元 3 个部分组成. 在 CPU 中, 控制单元和存储单元占的份额较大, 而 GPU 中拥有较多的逻辑运算单元, 因此相对于 CPU, GPU 拥有更强的浮点运算能力. 以本文使用的 GTX460 显卡为例, 其拥有 336 个流处理器, 核心频率为 700 MHz, 显存容量为 1 GB, 浮点运算能力达到 0.961 TFLOPS.

早期的 GPU 仅仅用来完成图形处理工作,

但随着技术的进步, 对 GPU 的可编程性增强, 其能力已经超出了原先设计的功能, 尤其是近年来, 图形处理器硬件水平得到高速发展^[1], 更多的开发者将其强大的并行计算能力应用到通用计算领域. 目前 GPU 已经应用于 n-body 模拟、医学 CT 图像重建、计算流体力学、分子动力学模拟、金融计算和油/气藏模拟等方面. 2003 年 Li 等^[2]利用 API 环境实现了格子玻尔兹曼方法在 GPU 上的计算. 2011 年 Aaron 把 GPU 应用在利用大涡模拟解决基于结构化网格的湍流问题上, 选用的 CPU 是 AMD Phenom 2.6 GHz 四核处理器, 在 Linux x64 操作系统上运行, 最多将计算速度提高了 15 倍左右. 2011 年朱小松^[3]实现了基于 GPU 并行加速的粒子法解决液体晃荡问题, 加速能力最多提高到 26 倍.

总的来说, 目前基于 GPU 的流体计算, 多采用粒子法求解拉格朗日型的 N-S 方程, 而对于求解欧拉型 N-S 方程, 尤其是非结构网格下, 利用 GPU 解决流体动力学问题的研究很少. 针对以上问题, 本文采用 CUDA 编程模型, 基于 GPU 架构并行算法建立非结构化网格水动力模型, 并对所建模型的计算效率与集群机的性能进行对比分析.

CUDA 平台使用带有关键词扩展的标准 C 开

收稿日期: 2013-10-15; 修回日期: 2014-01-08.

基金项目: 国家自然科学基金资助项目(51221961); 创新研究群体科学基金资助项目(51221961).

作者简介: 赵旭东(1986-), 男, 博士生, E-mail: zhaoxvdong@dlut.edu.cn; 梁书秀*(1972-), 女, 副教授, E-mail: sxliang@dlut.edu.cn.

发语言,可以通过类 C 语言编写在显卡芯片上执行的程序. CUDA 编程模型中,将 CPU 作为主机 (Host),将 GPU 作为协处理器或设备 (Device)^[4],二者共同协作完成整个程序.其中,Host 端负责执行串行部分的程序,Device 端负责并行部分,Device 端的程序又称为“kernel”.GPU 执行时的最小单位是 thread,CUDA 架构可以拥有海量的 thread,非常适合大规模数据并行计算^[5].

GPU 并行计算相对于传统的 CPU 有以下几个优势:成本相对较低,一个普通计算机的价格,计算性能达到了 TFLOPS,相当于一个高性能计算集群系统^[6];具有可扩展性,可以多个 GPU 协同并行计算,支持高效线程,使得应用程序可以提供比现有硬件执行资源更高的并行度^[7];节点之间可以共享存储器数据,显存带宽是内存的 10 倍左右,可以高速传递数据;有众多的核心,完全可以使用胖节点式的并行算法,而 CPU 的大型的并行,都是在集群层面的,而不是在单机上的;CUDA 这一类的 GPU 并行编程语言,并行层面在线程级别,相对于 CPU 要方便许多.

1 非结构化网格水动力模型

1.1 数学模型

从 N-S 方程出发,假设海水为不可压缩黏性流体,密度恒定,以连续性方程和动量方程作为二维水动力数值模型的控制方程:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad (1)$$

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} - f v = -g \frac{\partial p}{\partial x} - \frac{(\tau_{sx} + \tau_{bx})}{\rho} + \nu_t \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \quad (2)$$

$$\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + f u = -g \frac{\partial p}{\partial y} - \frac{(\tau_{sy} + \tau_{by})}{\rho} + \nu_t \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \quad (3)$$

当水表面风速 $|W| \in (11, 25)$ 时,自由表面剪切力

$$(\tau_{sx}, \tau_{sy}) = (0.49 + 0.065 |W|) \times 10^{-3} (W_x, W_y) \quad (4)$$

底部剪切力

$$(\tau_{bx}, \tau_{by}) = C_d \sqrt{u^2 + v^2} (u, v) \quad (5)$$

$$C_d = \frac{g}{C^2} = \frac{g}{(M\eta^{1/6})^2} \quad (6)$$

固边界条件

$$\partial u / \partial n = 0 \quad (7)$$

其中 (u, v) 为水平面流速, (τ_{sx}, τ_{sy}) 为表面风应力, (τ_{bx}, τ_{by}) 为底部剪切应力, f 为科氏力系数, η 为水面高程, M 为曼宁系数, C 为谢才系数. 根据 Smagorinsky 涡黏参数法^[8], 水平黏性项可以由下式近似表示:

$$\nu_t \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \approx \frac{\partial}{\partial x} \left(2A_m H \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left[A_m H \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \right] \quad (8)$$

$$\nu_t \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \approx \frac{\partial}{\partial y} \left(2A_m H \frac{\partial v}{\partial y} \right) + \frac{\partial}{\partial x} \left[A_m H \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \right] \quad (9)$$

$$A_m = 0.5 C \Omega^a \sqrt{\left(\frac{\partial u}{\partial x} \right)^2 + 0.5 \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2} \quad (10)$$

1.2 空间离散方法

本文使用的模型选用非结构化三角形网格,采用有限体积法进行数值求解,在求解过程中将整个数值计算区域划分为相互不重叠的非结构化三角形网格,如图 1 所示.

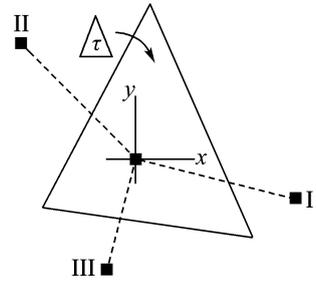


图 1 数值求解离散化非结构化三角形网格示意图
Fig. 1 Illustration of numerically solving the discretized unstructured triangular grid

三角形单元 τ 中的变量为线性分布,满足二阶空间精度,如式(11)~(15)所示:

$$\bar{R}(x, y) = \bar{R}_\tau + ax + by \quad (11)$$

$$a = \left[\sum_{i=1}^3 (\bar{R}_i - \bar{R}_\tau) x_i \sum_{i=1}^3 (y_i)^2 - \sum_{i=1}^3 (\bar{R}_i - \bar{R}_\tau) y_i \sum_{i=1}^3 x_i y_i \right] / \Delta \quad (12)$$

$$b = \left[\sum_{i=1}^3 (\bar{R}_i - \bar{R}_\tau) y_i \sum_{i=1}^3 (x_i)^2 - \sum_{i=1}^3 (\bar{R}_i - \bar{R}_\tau) x_i \sum_{i=1}^3 x_i y_i \right] / \Delta \quad (13)$$

$$\Delta = \begin{vmatrix} \sum_{i=1}^3 (x_i)^2 & \sum_{i=1}^3 x_i y_i \\ \sum_{i=1}^3 x_i y_i & \sum_{i=1}^3 (y_i)^2 \end{vmatrix} \quad (14)$$

$$\Delta = (x_1 y_2 - x_2 y_1)^2 + (x_1 y_3 - x_3 y_1)^2 + (x_2 y_3 - x_3 y_2)^2 \quad (15)$$

1.3 时间离散方法

在本文的数学模型中对连续性方程(1)求面积积分,通过修正过的四阶 R-K 时间积分进行求解运算,如式(16)~(18)所示:

$$\zeta_j^0 = \zeta_j^n;$$

$$R_\zeta^0 = R_\zeta^n = \sum_{m=1}^{NT(j)} [(\Delta x_{2m-1} \bar{v}_m^n - \Delta y_{2m-1} \bar{u}_m^n) D_{2m-1}^n + (\Delta x_{2m} \bar{v}_m^n - \Delta y_{2m} \bar{u}_m^n) D_{2m}^n] \quad (16)$$

$$\zeta_j^k = \zeta_j^0 - \alpha^k \frac{\Delta t R_\zeta^{k-1}}{\Omega_j^\zeta};$$

$$\zeta_j^{n+1} = \zeta_j^1, \bar{u}_i^0 = \bar{u}_i^n, \bar{v}_i^0 = \bar{v}_i^n, \bar{R}_u^0 = \bar{R}_u^n, \bar{R}_v^0 = \bar{R}_v^n \quad (17)$$

$$\bar{u}_i^k = \bar{u}_i^0 - \alpha^k \frac{\Delta t \bar{R}_u^0}{\Omega_i^u D_i};$$

$$\bar{v}_i^k = \bar{v}_i^0 - \alpha^k \frac{\Delta t \bar{R}_v^0}{\Omega_i^v D_i}; \bar{u}_i^{n+1} = \bar{u}_i^1; \bar{v}_i^{n+1} = \bar{v}_i^1 \quad (18)$$

2 基于 GPU 并行算法的实现及优化

目前 CUDA 的应用程序大多为基于 C 语言进行开发的,程序实现相对容易.而现在广泛使用的海洋模型如 POM、ECOM、FVCOM 等均为使用 FORTRAN 语言进行开发的,虽然可以在原有的模型中调用 CUDA 计算核心,但是并行计算部分的设计,与重新建立模型的工作量相差不大,计算效率也略低于 C 程序,因此本文以 C 语言为开发语言,在 CUDA 平台下建立基于 GPU 并行计算的水动力模型.

在基于 CPU 并行计算水动力模型中,由于 CPU 核心的数量远小于网格节点个数,传统的方法是利用美国密西根大学 Karypis 等编写的用于图的分区和稀疏矩阵排序的串行包——METIS 库,将计算区域分为 N 个子区域,分配给 N 个 CPU 进行计算,把子区域的初始流场信息、几何信息分别存储到对应的 CPU 内存中,在每一个 CPU 中启动计算进程,由主进程调度各 CPU 的计算^[9]. 在这一模式下,计算速度受到了多个方面因素的影响,包括 CPU 的计算速度、内存带宽、在分区的边界

处须保持 CPU 间数据交换过程等,因此并行计算的效率并不随 CPU 核心的个数呈线性增加.

由于 GPU 的并行计算架构与 CPU 的并行计算架构有较大的区别,在并行化的实现过程中也有很大的不同. GPU 中的计算核心称为流处理器(SP),以本文中使用的 GTX460 显卡为例,包含了 330 个流处理器. GPU 通过同时往每个流处理器上并发执行线程来实现加速计算.若像传统的并行方式一样利用 METIS 库将计算区域分块计算,一方面如果分块过多会导致计算过程过于复杂,另一方面并行程序会因为每个分块之间进行数据交换而导致并行计算效率的下降,因此在 GPU 并行程序设计中尽可能多地分配线程,每个线程执行对一个网格或网格节点的计算.

2.1 GPU 并行算法的实现

在 GPU 上实现并行算法需要以下几个步骤:

(1)执行 GPU 并行计算前检测设备,根据不同的硬件设备,设置执行环境,并在 CPU 上完成程序的前处理.

(2)在显卡上分配存储空间,用 cudaMemcpy 函数将在文件中读取的初始数据发送到显存中.

(3)将串行计算程序中需要每次对网格循环计算的程序段用对应的 GPU 并行子程序实现,在每个并行子程序中设定在 GPU 并行计算中的 Grid 和 Block 的维度.

(4)输出计算结果,用 cudaMemcpy 函数将显存中的数据返回到内存中,再输出到结果文件.

2.2 对 GPU 并行算法的优化

在 GPU 上实现并行计算,核心技术是算法的并行化和程序优化.在本文中按照计算模型的特点进行了适当的优化.

在 GPU 并行程序中处理的对象是空间离散后的每个网格以及每个网格节点,这样就避免了由于分区不均衡引起的计算效率降低,同时还节省了分区边界处数据交换的时间.因此在原串行程序中的每次对网格数组或节点数组进行操作中,将其每次循环需要计算的内容对应地分配到 GPU 的各个线程中,以海量的线程个数隐藏了数据访问延迟的时间.

由于在本文中使用的非结构化网格,无法按照矩阵方式分配在每个线程中计算的网格,故在 GPU 并行程序的线程设计中将 Block 设置为一维,并且每个 Block 中包含 64 个线程,即两个线程束.则整体的 Block 的个数为 $N/64+1$ (其中

N 为计算单元个数), 这样就将 GPU 划分为 $(N/64+1) \times 64$ 个线程, 这种线程分配方式最大限度地降低了分支流程和空载线程的个数, 实现了 GPU 核心的最高利用效率. 根据式(1)~(16), 每个 GPU 的线程计算一个网格或网格节点的数据, 将所需要的数据和需要计算的结果作为 kernel 函数的参数, 并在所有节点的数据计算完成后, 将计算结果返回到显存的数组中.

为了减少显存和内存之间数据传输速度的限制, 在整个计算过程中, 完成初始化后将变量传递到显存中, 在迭代计算过程中, 所有数据均在显存中进行交换, 直至需要将中间结果输出, 再将显存中的数据返回到内存的数组中, 这样减少了由于数据交换引起的时间损耗.

3 模型验证

数值实验的模型水深 $D = 10$ m, 长 $L =$

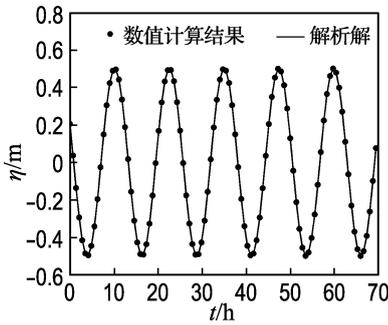
3.7 km, 宽 $B = 1.5$ km, 整个流场的右侧为开边界, 有作用振幅为 0.5 m 的 M2 分潮. 为了避免受到边界影响, 在该数值实验结果对比中, 验证点选择整个流场的中间点, 距离左侧边界 1.85 km^[10].

根据 Ippen 的理论^[11], 对于此种水域情况, 有如下解析解:

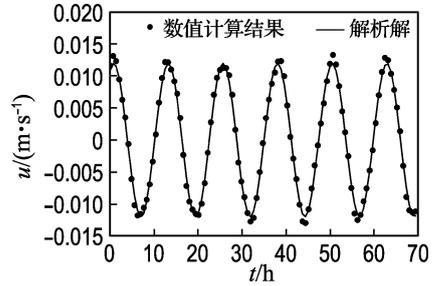
$$\eta = A \cos(\omega t); u = \frac{A \omega x}{h} \sin(\omega t); \omega = \frac{2\pi}{T} \quad (19)$$

其中 A 为振幅, ω 为角频率, x 为该点距离左侧边界的距离, h 为水深, T 为潮周期, η 为潮波面升高, u 为纵向流速.

在数值模型中将计算区域剖分成 1 620 个三角形网格, 验证点潮位和纵向流速计算结果历时曲线如图 2 所示. 从图中可以看出, 数值计算结果与解析解吻合度较好.



(a) 潮位对比



(b) 纵向流速

图 2 验证点潮位及流速历时曲线计算结果对比

Fig. 2 The contrast of elevation and current speed duration curve result in the verified points

4 GPU 算法的并行效率分析

为了更好地说明 GPU 在水动力模型并行计算过程中起到的作用, 数值模型实验采用在上一章模型验证中的计算水域, 并设计了 4 种不同网格密度的工况算例, 分别使用 CPU 单线程 (Intel Core™ i7 930 2.8 GHz)、集群机 (使用 30 核心、24 核心、20 核心, 最大计算节点数为 12, CPU 型号为 Intel Xeon E5430, 主频 2.66 GHz), 以及 GTX460 显卡 (330 个流处理器, 核心频率 700 MHz) 进行计算, 并对计算耗用时间进行对比分析. 在集群机计算过程中发现, 在工况 1~3 中, 网格数不多时, 总的计算核心数保持不变, 集群机节点数以及每个节点使用核心数变化对计算时间影

响不大, 计算所需时间基本不变; 在工况 4 中, 网格数超过 20 万, 受集群机节点数以及每个节点使用核心数影响明显, 不同节点的分配方式得到的加速比 R 如图 3 所示, 其中加速比为单线程计算时间与集群机计算使用时间的比值, 横坐标为节点数 \times 每个节点使用核心数.

由图 3 可以看出, 在核心总数相同的条件下, 使用的节点数越多, 计算效率越高, 相应的加速比就越高. 因此集群机的计算效率受计算核心总数, 以及使用节点个数的影响比较明显.

采用集群机的最优化的节点分配方案, 并与 GPU (Nvidia GTX460) 并行计算的加速能力进行对比如表 1 和图 4 所示.

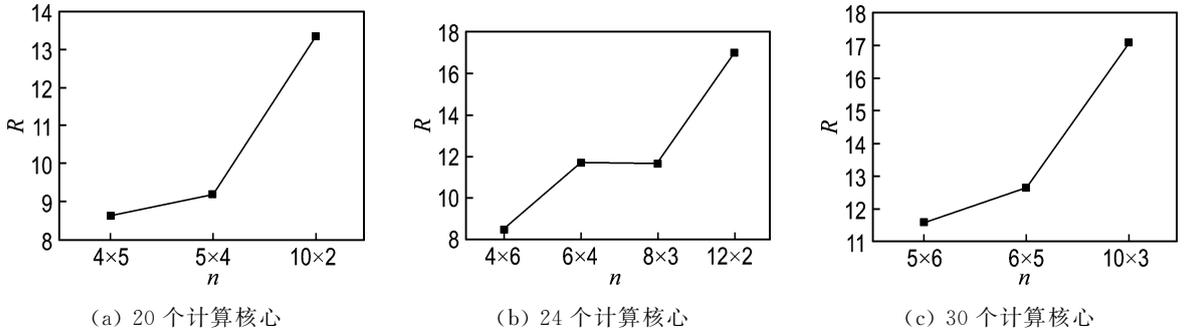


图3 集群机在不同节点分配条件下计算工况4得到的加速比

Fig. 3 The speedup ratio in different distributions of compute nodes when using cluster to compute Case 4

表1 集群机和GPU不同网格数条件下所用时间

Tab. 1 The time of cluster and GPU cost in different number of grids

工况	网格数/个	CPU	集群机(30核)	集群机(24核)	集群机(20核)	GPU
		计算时间/s	计算时间/s	计算时间/s	计算时间/s	计算时间/s
1	1 620	727	205	215	205	475
2	14 668	5 520	596	568	620	1 091
3	58 974	28 080	1 494	1 807	2 091	3 383
4	236 176	139 560	8 849	8 175	10 453	13 470

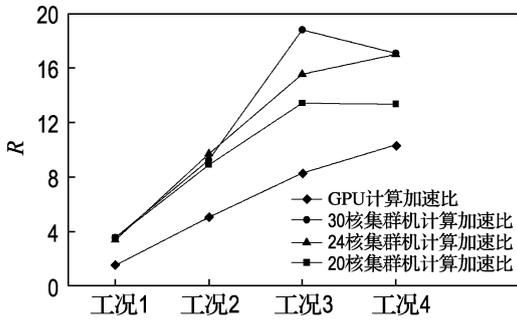


图4 不同网格数条件下并行加速能力对比

Fig. 4 The contrast of speedup capacity at different number of grids

表1中结果为分别使用单线程CPU串行算法、3种不同计算核心数的MPI并行算法,以及GPU并行算法,计算4种不同网格数下算例所使用时间.图4为表1中各并行算法所对应的加速比.在本文中最多只使用了12个计算节点,由图4可以看出,随着网格个数的增加,GPU并行计算的加速比能够实现相对稳定的提高.计算网格数在工况1~3时,加速比提高的速度低于集群机,网格数在工况3~4时,由于节点使用个数的限制,使用30核和20核的加速比略有下降,GPU的加速比仍稳定地提高,其加速比提高的速度基本与24核集群机的平行.由加速比随网格数变化曲线的趋势可以看出,若网格数继续增加,集群机的加速能力会受到节点分配方案和等待其他分区计算完成时间的影响而有所降低,而GPU

加速能力仍会相对稳定地提高.

5 结论

(1)对不同计算核心分配方案的MPI并行计算与单CPU串行计算性能进行对比分析,得出结论:当网格节点数在工况1~3(即网格数较少)时,不同的节点分配方案对集群机的加速能力影响不大;在网格数量较多时,集群机加速能力受计算节点分配方式影响显著.

(2)通过对不同计算核心数的MPI并行算法以及GPU并行算法的加速比的对比得出:使用GPU并行算法,能够在保证较高计算精度的基础上,大幅提高方程求解的速度,尤其是随着网格数的增加,基于GPU求解方程的计算加速比得到显著的提高.当网格个数增加到20万个以上时,GPU的计算能力已经接近使用20核集群机的计算能力,加速比提高了10倍以上,并且加速比的变化率已经超过20核和30核的集群机,并行加速效果十分显著.若考虑其他方面的计算成本,如集群机节点的利用率、能耗等方面,使用GPU进行并行计算的优势更加明显.由此可见,GPU在大范围海域的水动力模型的数值计算方面具有较好的发展前景.

参考文献:

[1] NVIDIA. NVIDIA CUDA C programming guide

- [EB/OL]. [2013-08-19]. <https://www.nvidia.com>.
- [2] LI Wei, WEI Xiao-ming, Kaufman A. Implementing lattice Boltzmann computation on graphics hardware [J]. **Visual Computer**, 2003, **19**(7-8):444-456.
- [3] 朱小松. 粒子法的并行加速及在液体晃荡研究中的应用 [D]. 大连:大连理工大学, 2011.
ZHU Xiao-song. Parallel acceleration of particle method and its application on the study of liquid sloshing [D]. Dalian: Dalian University of Technology, 2011. (in Chinese)
- [4] 张舒, 褚艳丽. GPU 高性能运算之 CUDA [M]. 北京:中国水利水电出版社, 2009.
ZHANG Shu, CHU Yan-li. **GPU High Performance Computing: CUDA** [M]. Beijing: China Water Power Press, 2009. (in Chinese)
- [5] 王英俊, 王启富, 王钢, 等. CUDA 架构下的三维弹性静力学边界元并行计算 [J]. 计算机辅助设计与图形学学报, 2012, **24**(1):112-119.
WANG Ying-jun, WANG Qi-fu, WANG Gang, *et al.* CUDA based parallel computation of BEM for 3D elastostatics problems [J]. **Journal of Computer-Aided Design & Computer Graphics**, 2012, **24**(1): 112-119. (in Chinese)
- [6] 多相复杂系统国家重点实验室. 基于 GPU 的多尺度离散模拟并行计算 [M]. 北京:科学出版社, 2009.
State Key Laboratory of Multiphase Complex Systems. **Multi-scale Discrete Simulation Parallel Computing Based on GPU** [M]. Beijing: Science Press, 2009. (in Chinese)
- [7] Kirk D B, Hwu Wen-mei W. **Programming Massively Parallel Processors** [M]. Beijing: Tsinghua University Press, 2010.
- [8] 陈长胜. 海洋生态系统动力学与模型 [M]. 北京:高等教育出版社, 2004.
CHEN Chang-sheng. **Ocean Ecosystem Dynamics and the Model** [M]. Beijing: Higher Education Press, 2004. (in Chinese)
- [9] 王敏, 王明, 杨明. 小浪底水库三维数学模型并行计算研究 [J]. 人民黄河, 2012, **34**(5):25-27.
WANG Min, WANG Ming, YANG Ming. Parallel computing research of Xiaolangdi Reservoir three-dimensional mathematical model [J]. **Yellow River**, 2012, **34**(5):25-27. (in Chinese)
- [10] 张瑞瑾. 三维潮流数值模型及其应用 [D]. 大连:大连理工大学, 2002.
ZHANG Rui-jin. Establish and the application of a three dimensional numerical tidal model [D]. Dalian: Dalian University of Technology, 2002. (in Chinese)
- [11] Ippen A T. **Estuary and Coastline Hydrodynamics** [M]. New York: McGraw-Hill Book Co, 1966.

Foundation and analysis of computational efficiency for hydrodynamic model based on GPU parallel algorithm

ZHAO Xu-dong¹, LIANG Shu-xiu^{*1}, SUN Zhao-chen¹,
LIU Zhong-bo^{1,2}, HAN Song-lin¹, REN Xi-feng¹

(1. State Key Laboratory of Coastal and Offshore Engineering, Dalian University of Technology, Dalian 116024, China;
2. Transportation Management College, Dalian Maritime University, Dalian 116026, China)

Abstract: Unstructured grid has been widely used to establish hydrodynamic models. To rapidly get the calculation results without a cluster when the number of computational grids is too large, a high-performance computing technology which is based on GPU (graphic processing unit), is adopted to design a parallel algorithm and establish a 2D unstructured grid hydrodynamic model on CUDA (compute unified device architecture) development platform. Through the comparisons to the computational efficiency of a cluster and the Graphic Card of GTX460, the advantages of GPU method are confirmed: the speedup ratio can reach more than 10 times and maintain a high growth as the increase of the number of computational grids. It is suitable for the numerical simulation of large-domain hydrodynamic models.

Key words: graphic processing unit (GPU); unstructured grid; hydrodynamic model