基于 CUDA 的汇流分析并行算法的研究与实现 *

赵向辉[1,2], 苗 青[1,2], 付忠良[1,2], 苏 畅[1,2], 李 昕[1,2]
(1. 中国科学院 成都计算机应用研究所, 成都 610041; 2. 中国科学院 研究生院, 北京 100049)

摘 要: 针对基于数字高程模型(DEM)生成流域等流时线的快速运算问题,提出了一种基于统一设备计算架构(CUDA)平台同时可发挥图形处理器(GPU)并行运算特性的汇流分析的快速并行算法。采用改进后的归并排序算法进行数据排序及新的内存分配策略和改进的并行算法进行汇流分析。用该并行算法和 CPU 上的串行算法, 对生成基于 DEM 的等流时线运算时间和矩阵乘法运算时间进行分析验证。实验结果表明,基于 CUDA 的汇流分析并行算法能提高系统的计算效率,具有较好的效果。

关键词: 并行计算; 图形处理器; 统一设备计算架构; 汇流分析; 数字高程模型
中图分类号: TP391; TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2010)07-2445-03
doi:10.3969/j.issn.1001-3695.2010.07.011

Research and realization in parallel algorithm of confluence analysis based on CUDA

ZHAO Xiang-hui[1,2], MIAO Qing[1,2], FU Zhong-liang[1,2], SU Chang[1,2], LI Xin[1,2]
(1. *Chengdu Institute of Computer Applications, Chinese Academy of Sciences, Chengdu* 610041, *China*; 2. *Graduate School, Chinese Academy of Sciences, Beijing* 100049, *China*)

Abstract: Aiming at the fast parallel computing of generating isochrones of watersheds that based on digital elevation model (DEM), this paper proposed a fast parallel algorithm of confluence analysis based on compute unified device architecture (CUDA) platform that could use parallel computing of graphic processing unit (GPU). Carried data sorting out by using the improved merge sorting algorithm, adopted the new memory allocation strategy, carried confluence analysis out by using the improved parallel computing algorithm. Using the presented parallel algorithm and the serial algorithm based on CPU to analyze and verify the time consumed when generating isochrones of watersheds based on DEM and executing matrix multiplication. The experiments results illustrate that this parallel algorithm of confluence analysis based on CUDA can improve the computational efficiency of the system and have a better effect.

Key words: parallel computing; graphic processing unit (GPU); compute unified device architecture (CUDA); confluence analysis; digital elevation model (DEM)

CUDA 是 NVIDIA 公司提供的 GPU 用于通用计算的开发环境,是一个全新的软硬件架构,可以将 GPU 视为一个并行数据计算的设备,对进行的计算进行分配和管理。图形显示卡,尤其是高端的显卡,现在几乎是单机必配的重要硬件。其核心部件 GPU 采用 SIMD 体系结构,大量的计算部件使得它很适合处理高数据量的并行运算[1~3]。基于 CUDA 平台,利用 GPU 强大的并行处理能力来加速通用科学计算及普通应用程序是非常好的应用趋势,已经成功应用到诸如金融风险管理、媒体图像以及科学研究[4]等领域。

本文结合科研项目"基于 3S 技术的自然灾害智能预测平台的研发与应用"中相关的关键信息技术来开展基于 CUDA 的汇流分析并行算法的研究与实现。在自然灾害的山洪汇流分析中,项目中要绘制的等流时线是基于 DEM 高程图绘制的[5,6]。若采用传统的基于 CPU 的串行算法,高分辨率的 DEM 图在带来更好的计算效果的同时,也大大地增加了计算量,使得计算时间大幅度增加,降低了计算效率。例如作为四川省试点地区的彭州甘溪沟流域,其流域面积 11.1 km^2,沟道长度为 6.83 km,若使用 80 m 作为单位栅格边长来划分流域,则流域可被划分为 1 805 个有效栅格单位,若采用高精度的划分方法,以 20 m 作为栅格边长来划分流域,则流域被划分为 28 900 个有效栅格,在能获得更高精度 DEM 图的情况下,栅格的数量会更大。这样,在获得更精确结果的同时,大大牺牲了运算效率。传统的方法难以满足高数据量的计算需求,本文算法的计算核心是矩阵相乘计算,需要花费大量的时间,GPU 在计算矩阵相乘时,比 CPU 要快得多,因此可巧妙地利用基于 CUDA 平台的 GPU 强大的并行处理能力来加速计算速度[7,8]。本文提出了一种基于 CUDA 平台可发挥 GPU 并行运算特性的快速并行算法。

1 CUDA 并行计算的引入

CUDA 表示统一计算设备架构,使 GPU 能够解决复杂的计算问题。它是一个并行编程模型和一个软件编程环境,它主

收稿日期: 2009-11-26; **修回日期**: 2010-01-04 **基金项目**: 四川省科技支撑计划基金资助项目(2008SZ0100,2009SZ0214)

作者简介:赵向辉(1982-),男,河南长葛人,博士研究生,主要研究方向为机器学习、数据挖掘、模式识别、图像分析等(xianghuizhao@hotmail.com);苗青(1982-),男,四川成都人,博士研究生,主要研究方向为图形图像处理、机器学习、模式识别等;付忠良(1967-),男,重庆人,研究员,博导,主要研究方向为机器学习、机器视觉、模式识别、图形图像处理等;苏畅(1983-),男,安徽淮南人,助理研究员,硕士,主要研究方向为机器学习、虚拟仿真、图形图像处理等;李昕(1985-),男,陕西汉中人,硕士研究生,主要研究方向为机器学习、虚拟仿真、图形图像处理等.

要是为了帮助广大的程序员来更好地开发平滑扩展的并行程序。当前主流 GPU 产品在浮点运算性能上都超过了 1 TFlops[9]，而 Intel 公司最新的四核处理器 Core2 i7 也仅达到 70 GFlops[10]。CUDA 模型中，GPU 相当于 CPU 的协处理器，它能并行执行非常多的线程，处理大规模的并行运算。整体来说，GPU 芯片类似于流处理器，适合一次进行大量相同的工作；CPU 则比较有弹性，能同时处理变化较多的工作[3]。图 1 表明，GPU 具备了超高的计算能力，在多个处理核心和高存储器带宽的配合下，最新的 GPU 成为了图形和非图形处理的超强工具。CUDA 的软件堆栈[11]采取如图 2 所示的多层组成方式，以此实现高性能计算。它允许定义一种叫 kernel 的函数扩展，当一个 kernel 函数被调用时，会有 N 个不同的 CUDA 线程在 GPU 中并行地执行相同的程序，并行执行的线程数量用扩展的“〈〈〈 〉〉〉”来指定。例如 FunctionOnGPU〈〈〈4，256〉〉〉(…)，该函数一旦成功调用，便共有 4 × 256 = 1 024 个独立的并行线程来执行该函数的内容。其中 4 为块(block)的数量，256 为每块内的线程(thread)的数量。

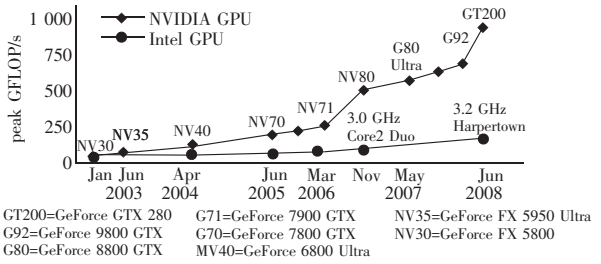

图 1　CPU 和 GPU 的每秒浮点运算次数

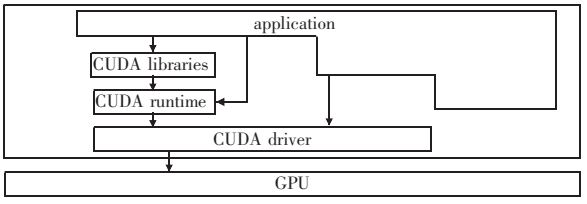

图 2　CUDA 软件堆栈

2　基于 DEM 的流域等流时线绘制算法

为了解决山洪灾害预测问题，通过建立山洪形成过程的数学模型，实现对山洪预测的数值模拟。等流时线是一种经典的流域汇流曲线[6]，根据对基于数字高程模型的流域等流时线绘制算法的描述[12]，在整个计算和绘制的过程中，主要经过五个大步骤的计算：a)对 DEM 图作填洼处理；b)根据高程对栅格进行排序；c)判断栅格水流方向；d)计算单个栅格汇流时间；e)计算流域汇流时间。其中：步骤 a)填洼处理可以由 GIS 软件，本文使用 Supermap Deskpro2008 软件来完成；步骤 b)对栅格按高程排序，一般排序算法的复杂度为 $o(n \log 2^n)$，对于大量元素的排序，较为耗费系统计算资源；步骤 c)判断栅格水流方向，需要与栅格周围的八个栅格进行比较；步骤 d)计算单个栅格汇流时间；步骤 e)从 DEM 值最小的栅格开始计算流域汇流时间。若采用传统的基于 CPU 的串行算法，这些运算都需要对栅格进行多次的遍历，在栅格划分较多的高精度 DEM 图上，就会成为整个算法的瓶颈。

观察分析以上几步可发现，在栅格运算的过程中，排序可以使用优化的并行排序算法来提高效率，步骤 c)和 d)的运算，由于其栅格之间的相关性很低，计算水流方向时，只需要考虑与之相关的八个栅格；计算单个栅格的汇流时间时几乎是独立运算。这两步运算符合 SIMD(single instruction multiple data，单指令多数据流)指令系统，宜于利用并行运算将其实现。按照规格网格法，把系统所需的 DEM 图表示成高程矩阵，在计算机中以二维矩阵的方式存放。流域外的空白地区的高程值被设为 -999。所以对 DEM 图的操作，实际上就是对高程矩阵的运算。CPU 的流式指令体系结构在提高矩阵运算速度上有着很大的限制。GPU(图形处理器)采用单指令多数据流体系结构，大量的计算部件使得它很适合处理矩阵运算。因此，本文将以上几步的算法进行优化，使其适合并行计算，并基于 CUDA 平台利用 GPU 的通用计算功能，使用改进的并行算法在单机上进行并行实现，最大化利用电脑硬件的计算能力。

3　基于 CUDA 汇流分析的并行算法

3.1　算法实现流程

本文使用改进后的归并排序算法对栅格进行排序。将第 2 章中介绍的相关算法进行优化，使其适合并行计算。该算法在 GPU 上的具体实现(以四川彭州甘溪沟流域的 DEM 图为例来对上述算法并行部分进行验证)，主要流程如下：

a)将 DEM 转换为高程矩阵。

b)使用改进后的归并排序算法，按照高程值对栅格进行排序。对高程矩阵按照数据量将数据分块并排序，将结果送至共享内存并分配空间，同步线程后，对奇偶块进行 Merge 运算。

c)计算栅格水流方向和经流时间。

d)绘制等流时线进行汇流展示。

3.2　算法及其应用分析

3.2.1　按照高程值对栅格进行排序

加利福尼亚州大学的 Satish 等人[13]提出了一种在 CUDA 上进行归并排序的算法，并提出通过预先抽取数据采样，对待排序序列根据采样值进行分块来提高归并算法中 Merge 运算的效率。同时，采样以 256 个待排元素为间隔，来保证划分出的每个子块最多含有 256 个元素。这样使在 Merge 运算时划分出的子块能够快速地在 GPU 上基于其共享内存来运算。

在计算等流时线前需要对栅格进行排序，以获得高程值最低的栅格作为整个流域汇流的出口。在排序算法中，由于多路归并排序，将数据分组排序后再合并，适合并行计算，在此处使用改进后的归并排序。首先将 DEM 对应的高程矩阵看做一维数组，然后按照数据量将数据分成 k 个块，每块分别送进 CUDA 的一个 block，使用 bitonic 对其进行排序，最后将结果送至共享内存中，同步线程后，对奇偶块进行 merge 运算。从国内外已有的研究来看，排序运算元素之间并不能达到高的独立运算，元素之间仍有一定的相关性，并行时加速比不是很高。因此，并行排序运算虽然有一定的效率提高，但并不是整个运算加速的重点。

3.2.2　计算栅格水流方向和经流时间

假设高程矩阵是一个 $m \times n$ 的矩阵，根据汇流分析的相关知识，求取栅格水流方向，只需计算该栅格与相邻的八个栅格的距离权落差，在计算单个栅格的流经时间时，需要使用公式：

$$t_i = nL/[K_v(S_i)^b] \tag{1}$$

式中各个参数的意义请参见文献[12]，只有参数 S_i 与该栅格的流向栅格有关，这两个运算之间的相关性很高，可以在一个

运算过程中完成。在这两个运算过程中,栅格与栅格之间的相关度较低,可以方便地实现高并行运算。

1)高程矩阵的分块和 CUDA 中内存的分配

该算法的整个并行计算过程中,尽量降低内存带宽的使用是提高性能的重要方式。而该算法的特点决定了即使是并行计算,在计算过程中仍可能存在重复读取的现象。例如,分块时把栅格 A 分入 block(0,0)中运算,但是栅格 A 的相邻栅格 B 可能在分块时被分入 block(0,1)中计算,这样 A 和 B 栅格就会在运算的过程中被重复地读取。为提高并行算法在 GPU 上的效率,如何分块尤为重要。为了防止在计算过程中对内存的反复读取,所以分块时,将需要参与计算的元素全部分入一个 block 中。例如需要 block(0,1)计算 $k\times k$ 个栅格的流向,就以这 $k\times k$ 的矩阵为中心,向四周相邻的方向各多取一个元素,即取以这个矩阵为中心的 $(k+2)\times(k+2)$ 阶矩阵。这样在计算该 $k\times k$ 个栅格时将需要的所有元素完全加载到存取速度很快的共享内存中来,在计算的过程中,该部分栅格的运算就不需要再存取任何外部的内存。

针对绘制等流时线的运算,需要创建三个与运算栅格相同大小的辅助矩阵,即流向矩阵、坡度矩阵和单位栅格流经时间矩阵。如果想取得更高的运算速度,最好能将这三个矩阵都存放在存取速度较快的共享内存中。同时,在 CUDA 平台上,一个线程块最大只支持 512 个线程,而且根据内存节距对齐的原则,GPU 的内存控制器从某个固定倍数的地址开始读取才会有最高的效率[14]。根据其支持的线程数,最好的办法就是将高程矩阵分解为 14×14 的小矩阵,这样,每次共享内存中就会加载 16×16 阶的矩阵,并在共享内存创建三个 14×14 阶的辅助矩阵。为了方便进行线程块的计算,针对高程矩阵,可为每个线程块分配 14×14 个线程,再建立 $(m/14)\times(n/14)$ 个线程块。这就充分利用了 GPU 多线程的特点,从而大大地提高运算速度。当然,分解高程矩阵时,有可能小矩阵块是边沿矩阵块,这就需要在分解时对小矩阵块作出是否是边界元素的判断,如果是边界元素,就需要用 -999 作为值填充在空白的相邻栅格的位置,创建出 16×16 阶的矩阵。

由于参加运算的高程矩阵的阶很可能不是 16 的倍数,为了使 GPU 更有效率地工作,在开始运算前为分解的分块小矩阵开辟内存空间时,就直接把内存大小配置成 16 的倍数,并在复制矩阵到显卡内存之前将其清零。这充分利用了 GPU 的内存读取特点,且符合 GPU 内存读取高效率的原则。使用 CUDA 提供的 cudaMallocPitch()函数就可以满足该要求,用于提高共享内存的访问速度[15]。

分配空间的伪代码如下:

```
#define BLOCK_SIZE 16
……
int Block_num = ((n + (BLOCK_SIZE - 1)-1) / (BLOCK_SIZE-1)) * (BLOCK_SIZE-1);
cudaMallocPitch((void **) &demc, &pitch_dem, sizeof(float) * Block_num, Block_num);
cudaMallocPitch((void **) &dirc &pitch_dir, sizeof(float) * Block_num, Block_num);
cudaMallocPitch((void **) &gradc, &pitch_grad, sizeof(float) * Block_num, Block_num);
cudaMallocPitch((void **) &timec, &pitch_time, sizeof(float) * Block_num, Block_num);
cudaMemset(demc, 0, pitch_dem * Block_num);
```

2)运算核函数

有了上述的高程矩阵分块方法和内存分配机制,真正需要在 GPU 上执行的核函数的伪代码如下:

```
__global__ static void genDLSX(const float * dem, size_t lddem, const float * dir, size_t ldir, float * grad, size_t ldgrad, float * time, size_t ldtime, int n)
{  分配共享内存空间
DEM[BLOCK_SIZE][BLOCK_SIZE];
将该线程需计算的分块矩阵的对应行列载入共享内存对应的位置;
__synthreads();
//线程同步,确保各个线程需要数据完全装载在共享内存中;
for(int i = 1;i < BLOCK_SIZE-2; i ++)
//只计算中间的 14×14 阶矩阵
{
计算出水流方向表,依据栅格水流方向,将其写入 dir 矩阵中;
根据水流方向,计算出水流方向的长度和坡度的比值,放入矩阵 grad 中;
依据式(1),计算出单个栅格的汇流时间,存入 time 矩阵;
}
__synthreads();//线程同步,确保以上各个线程计算均完成;
将结果写入 DIR 和 TIME 矩阵对应的位置;
}
```

4　实验分析

实验平台为 2.60 GHz 的 Intel E5300 双核处理器,2 GB 内存,Windows 7×64 操作系统。GPU 为 NVIDIA GeForce 9600GT,显存 256 MB,使用 2.2 版本的 CUDA toolkit 及对应 SDK。编程环境为 Visual Studio 2008。

考虑到传统的计算方法难以满足高数据量的计算需求,本文算法的计算核心是矩阵相乘计算,需要花费大量的时间,GPU 在计算矩阵相乘时,比 CPU 要快得多。因此,本文采用改进后的归并排序算法进行数据排序及新的内存分配策略和改进的并行算法进行汇流分析,可巧妙地利用基于 CUDA 平台的 GPU 强大的并行处理能力来加速计算速度。

实验以四川彭州甘溪沟流域的 DEM 图为例,对上述算法并行部分进行验证(参与运算的数据类型均定义为 float 类型)。彭州甘溪沟流域 DEM 图采用 16 位 GRID 数据集,划分为2 511×2 550 个栅格,计算机中表达为二维矩阵形式。其中最大值为 4 810.000 0,最小值为 0.000 0,空白边界值为 -9 999。实验结果如表 1 所示。由于数据量仅达到百万级的运算量,且其中包含加速效果不明显的排序运算,速度的提升不是很明显,仅达到 4 倍多。借助 GPU 强大的并行计算能力,如果数据量进一步增多,将会大大提高运算速度。实验运算过程中的方阵相乘使用 CUDA 运算的实验数据如表 2 所示。若矩阵阶数较小,由于 GPU 运算必须经过 PCI-E 总线传输及显存分配等必要耗时,加速比不明显;若矩阵阶数较大,如在本实验条件下,矩阵阶数达到 450,加速比就开始增大。实验结果表明,基于 CUDA 的并行算法比采用 CPU 上的串行算法具有更好的计算能力,能提高系统计算效率。

表 1　生成基于 DEM 的等流时线的运算时间比较

ms

CPU 上串行算法	基于 CUDA 的并行算法
3 207	650

表 2　矩阵乘法运算时间比较

ms

矩阵阶数	CPU 上串行算法	基于 CUDA 并行算法	加速比	矩阵阶数	CPU 上串行算法	基于 CUDA 并行算法	加速比
10	<5	41	无	700	657	102	X6.44
100	<10	43	无	1000	1880	232	X8.10
450	84	79	X1.06	2000	15308	1386	X11.04

(下转第 2451 页)

细精度地刻画了信念的形成。

在 B_{SoET} 系统中讨论 R_e 和 R_t 关系时,本文主要讨论了它们的必然算子,即$\boxdot$和$\Box$。对于$\boxdot$和$\Box$的对偶算子$\diamondsuit$和$\Diamond$在本文中并没有讨论,不讨论其的主要原因在于$\diamondsuit$和$\Diamond$算子不是信念形成的关键,同时也对愿望和意图不起关键作用。因此,在下一步工作中的研究重点在于,如何将 R_e 扩充为动作和动态关系,如将算子$\boxdot$扩充为$[\alpha]$或$[\alpha]^n$,又如何进一步在子结构演算中丰富 R_t 关系,使其进一步具有线性、序列性、非分支性和有穷间隔性等性质。同时,还可以通过添加相应的表示将来状态的算子"■",由相关领域的研究人员形成相应的愿望、意图和 BDI 模型,并最后付诸领域应用。

参考文献:

[1] 史忠植. 智能主体及其应用[M]. 北京:科学出版社,2000:12-22.

[2] MOORE R C. A formal theory of knowledge and action[M]//Formal Theories of the Commonsense World. [S. l.]:Ablex Publishing Corperation,1985:319-358.

[3] COHEN P R,LEVESQUE H J. Intention is choice with commitment [J]. Artificial Intelligence,1990,42(2-3):213-261.

[4] RAO A S,GEORGEFF M P. Deliberation and intentions,Technical Notes 10[R]. [S. l.]:Australian Artificial Intelligence Institute,1991.

[5] JIAO Wen-ping,SHI Zhong-zhi. Formalizing agent's attitudes with polyadic π-calculus[C]//Proc of the 4th Workshop on Practical Reasoning and Rationality. Stockholm:[s. n.],1999:21-27.

[6] 胡山立,石纯一. Agent 意图的双子集语义改进模型[J]. 软件学报,2006,17(3):396-402.

[7] HU Shan-li,SHI Chun-yi. An improved twin-subset semantic model for intention of agent[J]. Journal of Software,2006,17(3):396-402.

[8] KONOLIGE K,POLLACK M E. A representationalist theory of intention[C]//Proc of IJCAI'93. 1993:390-395.

[9] SINGH M P. Multiagent systems:a theoretical framework for intentions,know-how,and communications[C]//Lecture Notes in Artificial Intelligence. [S. l.]:Springer,1994.

[10] NAIR V C P. On extending BDI logics[D]. Queensland:Griffith University,2003.

[11] RAFAEL H B,MEHDI D,JÜRGEN D,*et al*. Multi-agent programming:languages,platforms and applications[M]. Berlin:[s. n.],2005.

[12] RAFAEL H B,MICHAEL F,WILLEM V,*et al*. Verifying multi-agent programs by model checking[J]. Journal of Autonomous Agents and Multi-Agent Systems,2006,12(2):239-256.

[13] RAFAEL H B,JOMI F H,MICHAEL W. Programming multi-agent systems in AgentSpeak using Jason[M]//[S. l.]:Wiley,2007.

[14] RESTALL G. An introduction to substructural logics[M]. Routledge,Tokyo:Mathematical Society of Japan,2000.

[15] ONO H. Proof-theoretic methods in nonclassical logics[R]. 1998:207-254.

[16] 刘冬宁. 时态信息处理中若干问题的逻辑公理化研究[R]. 广州:中山大学,2009.

[17] CAMILO T. The BDI model of agency and BDI logics[R]. 2005.

[18] BULLING N. Modal logics for games,time,and beliefs[D]. [S. l.]:Clausthal University of Technology,2006.

(上接第 2447 页)

5　结束语

本文算法能提高系统的计算效率,具有较好的效果。实验表明,对相同的数据,通过比较分析生成基于 DEM 的等流时线的运算时间和矩阵乘法的运算时间,GPU 比 CPU 的计算处理速度要快很多。基于 CUDA 平台可在一台主机上运行多个 GPU,通过 CPU 的线程来管理多个 GPU,或建立 GPU 集群,使其计算性能得到更大的提升。可以将基于 CUDA 的汇流分析的并行算法的设计思想应用于涉及到需求转换后能应用快速并行计算的各种项目系统中,以提高系统的计算效率。本文的算法设计思想和相关技术解决方法,为相关功能的实现提出了新的思路和解决办法,将对构建模型化、定量化、直观现代的新型自然灾害监测预警分析系统等提供重要的借鉴与参考。

参考文献:

[1] 吴恩华. 图形处理器用于通用计算技术、现状及其挑战[J]. 软件学报,2004,15(10):1493-1540.

[2] 张舒,褚艳利. GPU 高性能运算之 CUDA[M]. 北京:中国水利水电出版社,2009.

[3] GOTTLIED A,HWANG K,SAHNI S. Special issue on general-purpose processing using graphics processing units[J]. Journal of Parallel and Distributed Computing,2008,68(10):1305-1402.

[4] LESSIG C. An implementation of the MRRR algorithm on a data parallel coprocessor[R]. Toronto: University of Toronto, 2008.

[5] 詹道江,叶守泽. 工程水文学[M]. 北京:中国水利水电出版社,2000:96.

[6] 熊立华,彭定志. 基于高程模型的等流时线推求与应用[J]. 武汉大学学报:工学版,2003,36(3):1-3.

[7] KRISHNAN M,NIEPLOCHA J. SRUMMA:a matrix multiplication algorithm suitable for clusters and scalable shared memory systems [C]//Proc of the 18th International Parallel and Distributed Processing Symposium. Washington DC:IEEE Computer Society,2004.

[8] NVIDIA. CUDA programming guide[Z]. 2009.

[9] NVIDIA official site[EB/OL]. (2009). http://www.nvidia.com/.

[10] Intel official site[EB/OL]. (2009). http://www.intel.com/.

[11] NVIDIA. NVIDA CUDA compute unified device architecture reference manual[EB/OL]. [2008-08-20]. http://www.nvidia.com/object/cuda/cuda_develop.html.

[12] 杜尚海,史超,王晶晶. 基于数字高程模型的小流域等流时线绘制方法及应用[J]. 中国环境管理,2006,6(论文专辑):44-45.

[13] SATISH N,HARRIS M,GARLAND M. Designing efficient sorting algorithms for manycore GPUs[EB/OL]. [2008-09-15]. http://mgarland.org/files/papers/nvr-2008-001.pdf.

[14] LEFOHN A,KNISS J,OWENS J. Implementing efficient parallel data structures on GPUs[C]//Proc of GPU gems 2:Techniques for High Performance Graphics and General Purpose Computation. Boston,MA:Addison-Wesley,2005:512-545.

[15] 苏畅,付忠良,谭雨辰. 一种在 GPU 上高精度大型矩阵快速运算的实现[J]. 计算机应用,2009,29(4):1177-1179.