基于 GPU 的大规模栅格数据分块并行处理方法

崔树林[1,2],张 旭[2],张树清[3],张 军[1]
(1. 中山大学 信息科学与技术学院,广东 广州 510006;
2. 吉林大学珠海学院,广东 珠海 519041;
3. 中国科学院 东北地理与农业生态研究所,吉林 长春 130012)

摘 要:数学形态学运算是栅格数据处理的重要方法,具有较高的计算复杂度、并行度等特点,较容易发挥 GPU 众核高度并行执行的优势,以提高其计算效率。然而,有限的 GPU 全局存储器限制了其在大规模数据中的应用。文中在分析现有栅格数据并行方法的基础上,基于通用并行计算架构 CUDA,设计一种适应大规模数据的分块处理方法。文中以经典的膨胀算法为例对分块处理方法进行测试。实验结果表明:与传统的 CPU 串行处理方法相比,该方法可以显著提高数据处理速度。

关键词:GPU;分块;膨胀;栅格数据
中图分类号:P208 **文献标识码**:A **文章编号**:1673-629X(2015)03-0019-04
doi:10.3969/j.issn.1673-629X.2015.03.005

Parallel Processing Method for Large Scale Raster Data Blocking Based on GPU

CUI Shu-lin[1,2],ZHANG Xu[2],ZHANG Shu-qing[3],ZHANG Jun[1]
(1. School of Information Science and Technology,Sun Yat-sen University,
Guangzhou 510006,China;
2. Zhuhai College of Jilin University,Zhuhai 519041,China;
3. Northeast Institute of Geography and Agroecology,Chinese Academy of Sciences,
Changchun 130012,China)

Abstract:Mathematical morphology operations are important methods in the field of raster data processing,with high degree of computational complexity and parallelism. GPU-based hypercore parallel computing method can significantly improve the calculation speed. However,GPU's global memory limits its application in large scale data. Present a block-based method for large scale data,based on a general purpose parallel computing architecture,after analyzing the present parallel method for raster data. The new method is specifically tested with the classical dilation algorithm. Experimental results show that the calculation speed of the new method is faster than that of traditional sequence algorithm based on CPU.

Key words:GPU;block-based;dilation;raster data

0 引 言

数学形态学是一种利用数学工具进行图像空间结构分析的理论[1]。数学形态学方法更多地考虑了图像的结构特征,具有最大限度保留原始影像信息和消除噪声的优势[2]。数学形态学的应用几乎遍及计算机图形图像处理的所有方面,包括图像滤波、图像分割分类等[3]。在实际应用中,需要在规定时间内对巨大数据量进行多次重复计算,因此研究数学形态学的并行运算方法具有重要意义。

高性能计算机集群是常用的并行处理技术[4]。例如,文献[5]研究了基于数学形态学的高分辨率遥感

收稿日期:2014-03-31 **修回日期**:2014-07-01 **网络出版时间**:2015-01-20
基金项目:中国科学院重点部署项目(KZZD-EW-07,KZZD-EW-07-02);国家自然科学基金(面上项目)(41271196)
作者简介:崔树林(1980-),男,博士,研究方向为三维空间数据建模、并行空间分析;张树清,研究员,CCF 会员,研究方向为 GIS 理论与算法、GIS 数据模型、数据结构、几何计算、空间数据挖掘、GIS 应用;张 军,教授,研究方向为智能计算理论及应用、高性能计算、云计算、大数据处理、传感器网络。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20150120.2155.008.html

图像道路提取方法,并提出基于 MPI 编程模式和 MPI +OPENMP 编程模式的并行算法;文献[6]研究了基于 MPI 的改进中值滤波的并行算法等。然而,这种基于集群的并行方法具有开发难度大、硬件要求高等特点,很难满足普通用户的需要。

2006 年 11 月,英伟达(NVIDIA)公布了业界的第一个 DirectX 10 GPU(GeForce 8800GTX),即第一个基于 NVIDIA 的 CUDA 架构构建的 GPU。CUDA 的全称是统一计算设备框架(Compute Unified Device Architecture)。传统的图形处理架构将计算资源划分为顶点着色器和像素着色器。CUDA 架构的独特之处在于,包含了一个统一的着色器流水线,使得执行通用计算的程序能够在芯片上的每个数学逻辑单元(Arithmetic Logic Unit,ALU)进行排列。这些 ALU 都可以使用一个裁剪后的指令集执行通用计算,且具有 IEEE 单精度浮点数学运算功能。除此之外,GPU 上的执行单元不仅能任意地读写内存,而且能够访问有软件管理的缓存[7]。实际应用中,一方面,CUDA 能够利用 GPU 的并行计算引擎比 CPU 更高效地解决许多复杂计算任务[8];另一方面,开发 GPU 资源的程序员无需经过专门的图形 API 接口程序培训,就可以轻松地从普通程序开发转移到 GPU 并行计算的程序开发上[9]。

近年来基于 GPU 的数学形态学并行计算得到了研究人员的广泛关注,并取得了大量成果。例如,文献[3]的实验结果表明,基于 GPU 并行数学形态学运算速度可以达到几个数量级的提高。文献[10]进一步研究了有效利用 GPU 存储器分层结构,提高计算效率的方法。但是,相关研究并未考虑 GPU 端存储空间有限,面向大规模栅格数据导致 GPU 端加载数据失败这一现象。基于 GPU 端可以一次加载所有数据的假设,限制了此类方法的实际应用。因此,文中提出一种基于 GPU 的大规模栅格数据分块并行处理方法。

膨胀是一种最基本的数学形态学运算,通过它可以生成许多其他的形态学运算。膨胀的方式和程度由一个称为结构元素的集合控制。特别是结构元素较大时,导致图像处理的速度过低[11]。因此,文中以经典的膨胀算法为例,对大规模栅格数据分块并行处理方法的性能进行测试。实验结果表明,该方法不但可以有效提高运行速度,而且可以处理 GPU 端存储空间无法装载的大型栅格地理数据。

1 膨 胀

膨胀定义为集合运算。A 被 B 膨胀,记为 $A \oplus B$,定义为 $A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \varnothing\}$,其中 $\varnothing$ 为空集,B 为结构元素[12]。图 1 说明了膨胀的计算过程。图 1(a)显示了包含一个矩形对象的简单二值图像;图 1(b)是一个结构元素,在此例中它是一个菱形。计算时,结构元素用 0 和 1 的矩阵表示。另外,结构元素的原点用方框标明。图 1(c)为输出图像。

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 0 0 0
0 0 0 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

(a)原图像 A

0 1 0
1 [1] 1
0 1 0

(b)结构元素 B

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 0 0 0
0 0 1 1 1 1 1 1 0 0
0 0 1 1 1 1 1 1 0 0
0 0 0 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

(c)输出图像 $A \oplus B$

图 1 膨胀示例

一方面,经典的膨胀算法中,结构元素每一个像素与其所对应的图像上的像素按照膨胀要求进行逐一计算,时间复杂度为 $O(n^2)$,可见膨胀操作很耗时。另一方面,基于 GPU 的并行计算方法可以大大提高计算效率,但多数方法并未考虑 GPU 端存储器容量的限制,所以无法处理大规模栅格数据。

2 大规模栅格数据分块并行处理框架

2.1 CUDA 编程模型

(1)主机与设备。

CUDA 编程模型是以 C 语言为基础扩展而来的。从整体上,CUDA 编程模型分为两部分:主机(host)和设备(device)。主机是指运行在 CPU 上的部分;设备是指运行在 GPU 上的部分。一个系统可以包括一个主机和若干个设备。CPU 主要负责逻辑性强的事务处理和串行计算;而 GPU 则执行高度线程化的并行处理任务。

(2)kernel 函数。

运行在 GPU 上的 CUDA 并行计算函数称为 kernel(内核函数)。一个 kernel 函数是整个 CUDA 程序中一个可以被并行执行的步骤,并不是一个完整的程序。

kernel 调用时会被 N 个 CUDA 线程执行 N 次,即每个线程仅执行 1 次。

内核函数只能在主机端代码中调用,且必须通过__global__函数类型限定符定义。另外,在调用时必须利用一种全新的<<<... >>>执行配置语法指定内核的线程数,即声明内核函数的执行参数。实际上,每个执行 kernel 的线程拥有一个唯一的线程 ID,并通过内置的 threadIDx 变量在内核中访问。

(3)线程层次结构。

在 CUDA 编程模型下,GPU 执行的最小单位是线程。kernel 以线程网格(Grid)的形式组织。每个线程网格由若干个线程块(Block)组成,而每个线程块又由若干个线程(Thread)组成。实际上,Grid 仅用来表示一系列可以被并行执行的 Block 的集合,kernel 是以 Block 为单位执行的。各 Block 并行执行,没有执行顺序,且它们之间无通信。

线程块内的所有线程必须存在于同一个处理器核心中,且共享该核心有限的存储器资源。从而,一个块内的线程数目应当是有限的。目前,一个线程块可以包含多达 1 024 个线程。一个 kernel 可被多个同样大小的线程块执行,总的线程数是所有块内线程数之和。

(4)存储器层次结构。

CUDA 线程可在执行过程中访问多个存储器空间的数据。每个线程都有一个私有的本地存储器(Local Memory),每个线程块共用一个共享存储器(Shared Memory)。另外,所有线程都可访问相同的全局存储器(Global Memory)[13-14]。

2.2　分块并行处理框架

图 2 给出了栅格数据分块并行处理的框架。基于 GPU 的并行处理方法主要包括三个阶段:数据读入阶段①;分块并行处理阶段②~④;数据写入阶段⑤。

数据读入阶段将原始栅格数据文件读入到 host 端内存;数据写入阶段将最终结果写入到 host 端栅格数据文件。文中假设 host 端内存空间足够大,可以同时存放原始数据与处理后数据。读写栅格文件可以采用用户自己编写程序的方式,也可以采用开源库(例如 GDAL)。

文中采用开源库的方式,相关网站(http://www.gdal.org/)有详细的介绍。

文中主要处理栅格数据大小超过 GPU 端存储器容量的情况,即采用按行分块的形式,依次将栅格数据连续的若干行读入 GPU 全局存储器。分块并行处理阶段重复执行②~④三个步骤,直到所有数据块处理完成:②将数据块读入 GPU 全局存储器;③调用内核并行处理栅格数据;④将处理后数据块写入 host 端内存。为保证并行处理的正确性,实际读入 GPU 全局存储器的相邻数据块之间应存在重叠。重叠的行数与结构元素的行数 N 有关,即 $\lfloor N/2 \rfloor$。

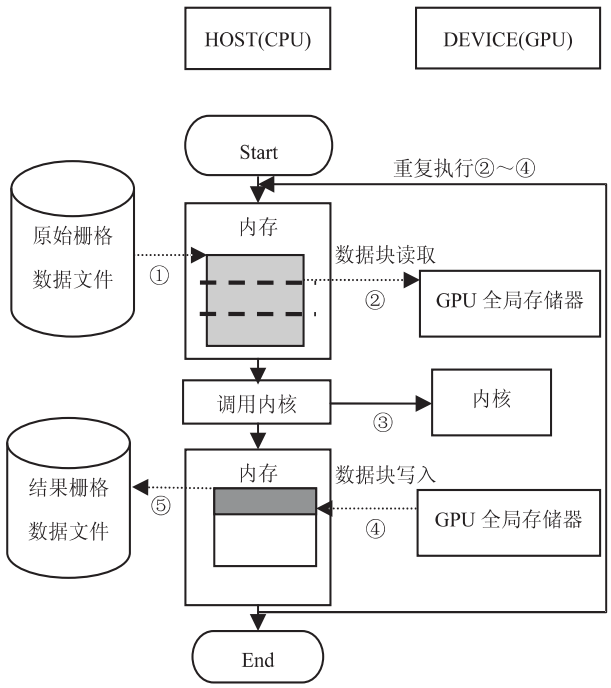

图 2　栅格数据分块并行处理框架

3　实　验

3.1　测试平台

GPU:NVIDIA NVS 4200M

CPU:Intel(R) Core(TM) i3-2350M CPU @ 2.30 GHz

OS:Fedora 17 x 86 64-bit version

CUDA:Release 4.2,V0.2.1221

3.2　实验结果

实验采用两个大小分别为 20 000 * 20 000、15 000 * 15 000 的图像进行测试,结构元素为 15 * 15 的菱形。为获得可靠的结果,采用多次测试求平均值的方法。基于 CPU 和 GPU 平台测试结果如表 1 所示,时间单位为 s。其中 100、500、1 000、2 000、3 000、4 000、5 000代表数据块的大小,即一次读入 GPU 端全局存储器的图像数据行数。"■"表示 GPU 端数据加载失败,即一次读入的数据量超过 GPU 全局存储器的实际容量。

表 1　基于 CPU 和 GPU 平台测试结果

时间/s	CPU	GPU						
		100	500	1 000	2 000	3 000	4 000	5 000
20 000 * 20 000	791.9	50.1	47.1	46.9	45.8	42.3	■	■
15 000 * 15 000	445.1	26.2	25.9	25.8	25.8	25.6	25.7	■

文中利用加速比(speedup)来分析并行计算性能:

$S(CPU,GPU) = T(CPU)/T(GPU)$

其中，$T(CPU)$ 表示在 CPU 平台下的计算时间；$T(GPU)$ 表示在 GPU 平台下的计算时间。

图 3 显示加速比与每次读入数据块的大小关系。可以看出，两个图像的加速比主要分布在 15 ~ 19 之间。与 20 000 * 20 000 图像相比，15 000 * 15 000 图像的加速比变化不是很明显，因为数据传输时间占整个程序运行时间的比重相对较小。

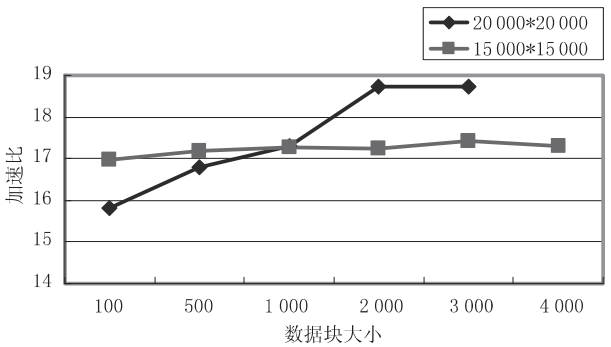

图 3 加速比与数据块大小关系

4 结束语

利用 GPU 提高计算效率，已经成为空间数据处理的重要方法。针对 GPU 端全局存储器容量有限，而在 GIS 实际应用中数据量较大，导致 GPU 端数据加载失败这一现象，文中提出一种适应大规模栅格数据的分块并行处理方法。该方法将栅格数据分块，并依次将数据块传输到 GPU 端进行并行处理，然后将计算结果拷贝到主机内存，重复顺序处理所有数据块，最终将结果写入数据文件。

文中以经典的膨胀算法为例，在一般 PC 系统上对分块并行处理方法进行测试，结果表明该方法不仅可以有效处理大规模栅格数据，而且可以极大提高计算效率。

参考文献：

[1] 尹星云，王 峻. 基于改进的彩色图像形态学膨胀和腐蚀算子设计[J]. 计算机工程与应用，2008，44(14)：172-174.

[2] 武文波，杨贵军，陈步尚. 基于数学形态学遥感影像分类后优化处理[J]. 辽宁工程技术大学学报：自然科学版，2003，22(2)：172-175.

[3] 张 聪，邢同举，罗 颖，等. 基于 GPU 的数学形态学运算并行加速研究[J]. 电子设计工程，2011，19(9)：141-143.

[4] Hawick K A，Coddington P D，James H A. Distributed frameworks and parallel algorithms for processing large-scale geographic data[J]. Parallel Computing，2003，29(10)：1297-1333.

[5] 刘 冬. 基于数学形态学的高分辨率遥感图像道路信息并行提取方法研究[D]. 长春：吉林大学，2011.

[6] 曹海燕. 数学形态学与变换域图像去噪算法及其并行化研究[D]. 成都：成都理工大学，2009.

[7] Sanders J，Kandrot E. GPU 高性能编程 CUDA 实战[M]. 北京：机械工业出版社，2011.

[8] Nvidia C. Programming guide 2.0[J]. Santa Clara，CA，USA：NVIDIA Corporation，2008.

[9] 李明彬. 基于 GPU 的图像压缩编码技术[D]. 长春：吉林大学，2009.

[10] Cui S L，Zhang S Q. An image decomposition strategy based on GPU′s memory hierarchy architecture[C]//Proc of the 5th international conference on computational and information sciences. [s. l.]：[s. n.]，2013：114-117.

[11] 杨 琨，曾立波，王殿成. 数学形态学腐蚀膨胀运算的快速算法[J]. 计算机工程与应用，2005，41(34)：54-56.

[12] 冈萨雷斯. 数字图像处理[M]. MATLAB 版. 北京：电子工业出版社，2005.

[13] Sanders J，Kandrot E. CUDA by example：an introduction to general-purpose GPU programming[M]. [s. l.]：Addison-Wesley Professional，2010.

[14] Temizel A，Halici T，Logoglu B，et al. Experiences on image and video processing with CUDA and OpenCL[J]. GPU Computing Gems，2011，2：547-567.

(上接第 18 页)

analysis[J]. Pacing and Clinical Electrophysiology，2004，27：757-763.

[6] 邹 月，陈建兵. Socket 的网络编程研究与实现[J]. 电脑编程技巧与维护，2009(8)：10-12.

[7] 谢希仁. 计算机网络[M]. 北京：电子工业出版社，2008.

[8] 房大伟，吕 双. ASP. NET 开发实战 1200 例[M]. 北京：清华大学出版社，2011.

[9] 仰燕兰，金晓雪，叶 桦. ASP. NET AJAX 框架研究及其在 Web 开发中的应用[J]. 计算机应用与软件，2011，28(6)：195-198.

[10] Microsoft Developer Network. FormView Class[EB/OL]. 2014-03-12. http://msdn. microsoft. com/zh-cn/library/system. web. ui. webcontrols. formview. aspx.

[11] Microsoft Developer Network. GridView Class[EB/OL]. 2014-03-12. http://msdn. microsoft. com/zh-cn/library/system. web. ui. webcontrols. gridview. aspx.

[12] ZedGraph Wiki. ZedGraph class library documentation[EB/OL]. 2014-03-12. http://zedgraph. sourceforge. net/documentation/default. html.

[13] 于国卿，李趁趁，赵雨森. ZegGraph 控件在水闸监测系统开发中的应用研究[J]. 南水北调与水利科技，2008，6(3)：43-45.

[14] 朱亦钢. 应用 Zedgraph 高效开发数据图表[J]. 电脑编程技巧与维护，2009(12)：59-61.

基于GPU的大规模栅格数据分块并行处理方法

作者： 崔树林， 张旭， 张树清， 张军， CUI Shu-lin， ZHANG Xu， ZHANG Shu-qing， ZHANG Jun

作者单位： 崔树林,CUI Shu-lin(中山大学 信息科学与技术学院，广东 广州 510006；吉林大学珠海学院，广东 珠海 519041)， 张旭,ZHANG Xu(吉林大学珠海学院,广东 珠海,519041)， 张树清,ZHANG Shu-qing(中国科学院 东北地理与农业生态研究所,吉林 长春,130012)， 张军,ZHANG Jun(中山大学 信息科学与技术学院,广东 广州,510006)

刊名： 计算机技术与发展 ISTIC

英文刊名： Computer Technology and Development

年，卷(期)： 2015(3)