一种基于 GPU 和内存映射文件的高分辨率遥感图像快速处理方法

马秀丹, 崔宾阁, 钟　勇, 张永辉, 费　东

(山东科技大学 计算机科学与工程学院, 山东 青岛 266590)

摘要: 高分辨率遥感图像处理经常面临程序执行时间过长和内存空间不足的问题, 虽然并行计算技术可以提高遥感图像的处理速度, 但是无法降低算法占用的巨大内存空间。为了解决这一问题, 本文提出了一种利用 CUDA 和内存映射文件的高分辨率遥感图像快速处理方法, 并以 K-Means 算法为例进行了实现。其中, CUDA 技术可以有效利用 GPU 强大的并行计算能力, 而内存映射文件技术降低了磁盘 I/O 速度较慢对算法性能的影响。实验结果表明, 本文方法比传统 K-Means 聚类算法计算速度提高了 30 倍左右, 内存使用量降低了 90%以上。

关键词: 内存映射文件; CUDA; 优化; 遥感图像; 聚类

中图分类号: TP391.41　　**文献标识码**: A　　**文章编号**: 1000-3096(2018)01-0139-08

DOI: 10.11759/hykx20171011025

随着卫星遥感技术的快速发展, 遥感图像的数据量越来越大。比如, SPOT-5 卫星图像存储了 5.76×10^8 个像素, WorldView-2 卫星图像存储了 2.62×10^8 个像素。如此大的数据量给遥感数据处理带来极大的压力和挑战, 主要表现在程序运行时间过长、计算机内存不足。当前计算机 CPU 主频和内存容量增长比较缓慢, 因此依靠硬件性能的提高解决遥感图像处理时间过长和内存不足的问题比较困难。如何提高遥感图像处理的时间效率和空间效率, 已成为遥感图像处理领域面临的最紧迫的问题之一。

为了提高遥感图像的处理速度, 各种并行计算方法被引入遥感图像处理过程中, 包括基于 MPI 的遥感图像快速处理方法[1], 它采用 MPI 标准并行编程环境和基于集群的多机处理系统对遥感图像进行信息提取; 基于多线程的遥感图像快速处理方法[2], 其中主线程负责对图像进行分块, 创建子线程, 具体由子线程完成各个分块的处理。在单核处理器中, 这种方法一般能够节省 30%的处理时间; 基于 GPU 与 CPU 异构模式的遥感图像快速处理方法[3-5], 在传统计算机系统中加入 GPU 作为加速部件并配合 CPU 共同承担计算任务的新型系统, 相比于传统的单纯以 CPU 作为计算部件的同构计算系统, 异构系统优势明显; 基于 GPU 的遥感图像快速处理方法[6-9], 与 CPU 只有少数几个核(多核)相比, GPU 几百个核(众核)具有更大的并行计算潜力, 可用于处理计算密度高、逻辑分支简单的大规模数据并行任务。GPU 技术是一种显卡的并行计算技术, 它以大量线程实现面向吞吐量的数据并行计算, 相对于其他技术而言, GPU 技术更适合像素数目庞大、逻辑分支简单、迭代次数多的遥感图像处理算法。

为了解决大数据量遥感图像处理内存不足的问题, 最基本的想法就是将图像划分为若干块, 每次只处理一块的图像数据, 这种方法可以显著降低内存的使用量。然而, 由于需要频繁进行 I/O 操作, 对于需要若干次迭代收敛的算法, 图像处理速度将会下降几倍甚至数十倍。解决这个问题的一种方法是采用分布式计算技术[10-12], 每个计算节点处理的分块大小应当小于其内存容量, 避免了迭代过程中多次读写磁盘; 另一种方法是采用内存映射文件技术[13-14]。内存映射文件建立了从磁盘中文件到特定内存区域的映射关系, 通过这种映射关系可以显著提升文件访

收稿日期: 2017-10-11; 修回日期: 2017-12-29

基金项目: 国家自然科学基金项目(41406200); 山东省自然科学基金(ZR2014DQ030)

[Foundation: This work was co-supported by National Natural Science Foundation of China, No.41406200; Shandong Province NaturalScience Foundation of China, No.ZR2014DQ030]

作者简介: 马秀丹(1993-), 女, 山东聊城人, 硕士研究生, 主要从事遥感图像处理等方面的研究, 电话: 17863951225, E-mail: lanhaimo@l63.com; 崔宾阁(1979-), 通信作者, 男, 山东莱阳人, 副教授, 主要从事遥感图像处理, 高光谱遥感, 智能信息处理, 高性能计算等方面的研究, 电话: 13969680498, Email: cuibinge@qq.com

问速度。

虽然集群、网格和分布式系统都能够提高遥感图像处理算法的时空效率，但是它们需要特定的计算基础设施，不仅代价昂贵而且环境配置和使用过程复杂。本文研究了普通计算机环境下的遥感图像处理算法优化方法，其基本思想是利用 GPU 并行计算技术提高遥感图像的处理速度，采用图像分块处理方法降低内存需求，利用内存映射文件技术提高图像文件访问速度，最终实现在不增加任何计算资源的情况下提高遥感图像处理算法的时空效率。

1 K-Means 算法、CUDA 和内存映射文件

在介绍本文的方法之前，首先回顾一下 K-Means 聚类算法、CUDA 平台和内存映射文件的相关知识。

1.1 K-Means 聚类算法

K-Means 聚类算法是应用较广的一种非监督分类算法，其基本思想是将所有样本聚类为 k 个簇，每个簇内具有较高的相似度。给定训练样本 $X=\{x^{(1)},x^{(2)},\cdots,x^{(n)}\}$，每个 $x^{(i)}\in R^p$，K-Means 聚类算法可以描述如下:

(1) 随机选取 k 个聚类中心: $\mu_1,\mu_2,\cdots,\mu_k\in R^p$

(2) 重复下面过程直至收敛: {

1) 对于每一个样本 i，计算其应该属于的类:

$$c^{(i)}:=\arg\min_j\left\|x^{(i)}-\mu_j\right\|^2 \quad i=1,2,\cdots,n \quad j=1,2,\cdots,k \tag{1}$$

2) 对于每一个类 j，重新计算该类的聚类中心

$$\mu^j:=\frac{\sum_{i=1}^{n}1\cdot\{c^{(i)}=j\}\cdot x^{(i)}}{\sum_{i=1}^{n}1\cdot\{c^{(i)}=j\}} \quad j=1,2,\cdots,k \tag{2}$$

}

上式中，$c^{(i)}$ 代表样本 i 与 k 个类中距离最近的那个类，$c^{(i)}$ 的值是 1 到 k 中的一个。如果 $c^{(i)}=j$，则 $\{c^{(i)}=j\}$ 的值为 1；如果 $c^{(i)}\neq j$，则 $\{c^{(i)}=j\}$ 的值为 0。公式(2)中的分母表示第 j 类中样本的数量，分子表示第 j 类中所有样本向量的总和，因此除法计算的结果是第 j 类的中心向量。K-Means 聚类算法的优点是简单、快速，对于处理大数据集是相对可伸缩和高效率；缺点是聚类结果对初始值敏感，易受噪声和孤立点的影响。K-Means 聚类算法的时间复杂度是 O(nkt)，其中 n 为样本数目，k 为聚类中心数目，t 为迭代次数，通常 $k\ll n$ 且 $t\ll n$；空间复杂度是 $O((n+k)\cdot p)$，其中 n 为样本数目，k 为聚类中心数目，$k\ll n$，p 为样本维数。

1.2 CUDA 平台

CUDA(Compute Unified Device Architecture，统一计算架构)是 NVIDIA(英伟达)公司推出的一种并行计算架构[15-17]。该架构通过利用 GPU(Graphic Processing Unit，图形处理器)的处理能力，可大幅提升计算性能。用户可使用 NVIDIA 的 GeForce 8 及以后的 GPU 进行计算，亦可以利用 GPU 作为 C-编译器的开发环境。以 GeForce 8800 GTX 为例，其核心拥有 128 个处理器。利用 CUDA 技术可以将那些内处理器串通起来，成为线程处理器解决数据密集的计算。各个内处理器能够交换、同步和共享数据。利用 NVIDIA 的 C-编译器，就能利用这些功能。

从硬件角度讲，GPU 是由多个 SM(Streaming Multiprocessor，流多处理器)组成，每个 SM 包含有多个 SP(Streaming Processor，流处理器)，SP 是最基本的处理单元。从软件角度讲，CUDA 是 SIMT(单指令多线程)架构，grid(线程网格)和 block(线程块)是 thread(线程)的组织形式。最小的逻辑单位是一个 thread，若干个 thread(128~512)组成一个 block，若干个 block 组成一个 grid 对应一个 kernel(函数)。一个 SM 负责多个 block 的计算任务(一般为 8 个)，每个 SP 在任一时刻只负责一个 thread。SM 中有共享内存、寄存器和 L1 缓存，因此 block 内可以共享公共内存，每个 thread 拥有自己的局部内存。warp(线程束)是 SM 调度和执行的基本单位，SIMT 架构使得同一个 warp 里的线程根据不同的数据执行相同的指令。SM 一次只能运算一个 block 里的一个 warp，如果 warp 中有线程的数据没有取到，那么 SM 调度下一个 warp 进行运算。

1.3 内存映射文件

内存映射文件(Memory Mapping File，MMF)是由一个文件到一块内存的映射。Win32 提供了允许应用程序把文件映射到一个进程中的函数 (Create-FileMapping)。内存映射文件技术提供了一种独特的内存管理特征，它允许应用程序与通过指针访问动态内存相同的方式访问磁盘上的文件。因此，可以在进程地址空间中将磁盘上的文件部分或全部映射到特定的地址范围，然后通过指针就可以访问内存映射文件中的内容。一旦某个文件被映射，就可以访问

它，就像整个文件已经被加载到内存中一样，从而可以不必对文件执行I/O操作，这对于大数据文件来说存取的效率极高。

内存映射文件技术比磁盘I/O操作快得多。应用程序可以像读写内存数据那样直接访问或修改映射后的内存文件，而不需要像普通I/O操作那样从文件开头进行寻址。内存映射文件的另外一个好处是“延迟加载”，当文件大小远大于内存容量时，将整个文件加载到内存需要操作系统反复将文件读入内存和写磁盘，这会引起严重的“抖动”。内存映射文件每次只需要加载较小的页面大小区域，因此基于内存映射文件的程序能够处理任意大的数据文件。

2 本文方法

为了解决遥感图像处理内存不足的问题，本文采用了图像分块的思想，即将遥感图像划分为若干块，每次只处理一个分块。然而，分块处理通常需要图像数据在内存和硬盘之间频繁交换。相对于内存访问来说，磁盘读写速度非常慢，从而导致遥感图像处理时间比较长。本文将基于CUDA并行计算技术、图像分块处理方法和内存映射文件技术3种手段相结合，克服内存空间不足和磁盘访问速度慢的瓶颈，提高遥感图像处理算法的时空效率。

2.1 基于CUDA的遥感图像K-Means聚类算法的并行化

GPU拥有众多的核(SP)，非常适合遥感图像的并行处理。由于SP在任一时刻只能执行一个线程，处理一个像素，因此需要对遥感图像进行分块，每次GPU处理一块图像数据。分块大小通常设置为256×256或512×512，即一个grid包含256/512个block，一个block包含256/512个thread，每个thread处理一个像素。太大或太小的分块都不利于充分发挥GPU的计算性能。CUDA能够将计算任务映射为大量的可以并行执行的线程，并由硬件动态调度和执行这些线程。因此，算法只需要将图像划分为若干块，然后将每个块发送给GPU处理，不需要关注其内部的线程调度和执行问题。

对于遥感图像的K-Means聚类算法，假设相对于图像大小内存空间足够大，基于CUDA平台的并行化处理算法可以描述如下：

(1) 使用RasterIO函数将遥感图像读入到内存中；

(2) 随机选取k个聚类中心为$\mu_1,\mu_2,\cdots,\mu_k\in R^p$；

(3) 将遥感图像划分为若干块，每一块的大小等于(最后一行或一列可能小于)256×256/512×512；

(4) 重复下面过程直至收敛：{

1) 对于每一个分块，将其数据从内存拷贝到设备端；

2) 将每一个类的聚类中心从内存拷贝到设备端；

3) 利用公式(1)在设备端对分块图像做并行计算；

4) 将分块计算结果(即样本的类别标记)从设备端拷贝到内存；

5) 处理下一个分块，直到所有分块处理完成为止；

6) 对于每一个类，利用公式(2)重新计算该类的聚类中心；

}

基于CUDA的K-Means聚类算法的时间复杂度是$O((2np/C+\alpha nk/m)\cdot t)$，其中，$n$为样本数目，$p$为样本维数，$C$为从内存到设备端的数据拷贝速度，因为需要双向拷贝，所以乘以2；α为CPU与GPU单核速度比值，k为聚类中心数目，t为迭代次数，m为流处理器数目。基于CUDA的K-Means聚类算法的空间复杂度是$O((n+k)\cdot p)$。

2.2 基于CUDA和MMF遥感图像K-Means聚类算法优化

由于操作系统和其他程序运行占用了较多的内存空间，遥感图像处理程序可以利用的内存空间通常只有几百兆字节，处理大小不足百兆字节的遥感图像。为了能够处理大数据量的遥感图像，必须对图像进行分块处理。分块处理方法需要将各个分块依次读入到内存中进行计算，然后再将中间计算结果写到磁盘中。对于需要若干次迭代的遥感图像处理算法(如K-Means聚类算法)，频繁地读写磁盘文件将会大大增加程序的执行时间。

为了解决这个问题，本文首先对遥感图像数据进行重新组织和存储，然后利用内存映射文件技术提高图像分块的读写速度。传统的遥感图像数据组织格式有三种类型：BSQ、BIP和BIL。BSQ(Band SeQuential Format，波段顺序格式)：每行数据后面紧接着同一波段的下一行数据；BIP(Band Interleaved by Pixel format，波段按像元交叉格式)：按顺序存储第一个像元的所有波段，接着是第二个像元的所有波段，然后是第三个像元的所有波段，交叉存储直到像元总数为止；BIL(Band Interleaved by Line Format，波段按行交叉格式)：先存储第一行的第一个波段，

接着是第一行的第二个波段，然后是第一行的第三个波段，交叉存储直到最后一行最后一个波段为止。

BSQ 格式适合于对单个波段某个空间范围内的数据存取，BIP 格式适合于像元光谱数据的存取，BIL 格式适合于存取某些行的像元光谱数据。如果需要读取遥感图像某个分块的所有波段数据，BSQ、BIP 和 BIL 格式都不是最优方案，因为分块各个波段的数据被分散存储在磁盘的不同区域，无法利用 MMF 技术进行存取。为此，本文提出第四种存储格式 BIT(Band Interleaved by Tile Format，波段按瓦片交叉格式)：先存储第一个瓦片的第一个波段，接着是第一个瓦片的第二个波段，然后是第一个瓦片的第三个波段，交叉存储直到最后一个瓦片的最后一个波段为止。每个瓦片对应于遥感图像的一个分块。这种存储格式使同一个瓦片的所有波段连续存储，因而可以利用 MMF 技术通过内存指针连续读写瓦片的各个波段数据。

基于 CUDA 和 MMF 的 K-Means 聚类算法具体实现描述如下：

(1) 将遥感图像按行列切分成 M 个瓦片(Tile)，其中每个瓦片的大小为 B(512×512 或其他尺寸)；

(2) 建立图像文件 tempImage 保存原始图像中的每个瓦片，建立类别文件 tempClass 保存图像中每个像元的类别；此操作目的是建立内存映射文件；

(3) For i = 1 to M Do Begin

1) 使用 RasterIO 函数将第 i 个瓦片读入到内存中；

2) 将第 i 个瓦片写入到图像文件 tempImage 中；

End

(4) 随机选取 k 个聚类中心为：$\mu_1, \mu_2 \cdots, \mu_k \in R^p$；

(5) 打开图像文件 tempImage 并映射到内存指针 ptrImage，打开类别文件 tempClass 并映射到内存指针 ptrClass；

(6) 重复下面过程直至收敛：{

For i = 1 to M Do Begin

1) 使用内存指针 ptrImage 读取第 i 个瓦片的图像数据，并拷贝到设备端；

2) 将每一个类的聚类中心从内存拷贝到设备端；

3) 利用公式(1)在设备端对瓦片数据做并行计算；

4) 将分块计算结果(即样本的类别标记)从设备端拷贝到内存；

5) 使用内存指针 ptrClass 将类别信息写到类别文件 tempClass 中；

End

对于每一个类 j，重新计算该类的聚类中心

$$\mu^j := \frac{\sum_{r=1}^{M}\sum_{s=1}^{B} 1\cdot\{c^{(r,s)}=j\}\cdot x^{(r,s)}}{\sum_{r=1}^{M}\sum_{s=1}^{B} 1\cdot\{c^{(r,s)}=j\}} \quad j=1,2,\cdots,k \qquad (3)$$

其中，$x^{(r,s)}$ 表示通过内存指针 ptrImage 读取的第 r 个瓦片第 s 个像元的光谱向量，$c^{(r,s)}$ 表示通过内存指针 ptrClass 读取的第 r 个瓦片第 s 个像元的类别，而 μ^j 表示第 j 类的聚类中心(光谱向量均值)。

}

基于 CUDA 和 MMF 的 K-Means 聚类算法时间复杂度是 $O(2np/D+(2np/C+\alpha nk/m)\cdot t)$，其中：$n$ 为样本数目，p 为样本维数，C 为从内存到设备端的数据拷贝速度，D 为磁盘读写速度，$D \ll C$；α 为 CPU 与 GPU 单核速度比值，k 为聚类中心数目，t 为迭代次数，m 为流处理器数目。算法空间复杂度是 $O((B+k)\cdot p)$，其中 B 为瓦片大小，通常 $B \ll n$，k 为聚类中心数目，p 为样本维数。

在处理大数据量遥感图像时，如果仅仅采用分块方法而没有使用 MMF 技术，那么每一轮迭代中所有遥感图像分块都要读写磁盘各一次，这种情况下算法的时间复杂度是 $O((2np/D+2np/C+\alpha nk/m)\cdot t)$，$D \ll C$；算法的空间复杂度是 $O((B+k)\cdot p)$，$B \ll n$。

2.3 复杂度对比与分析

本文已讨论了 4 种实现策略下的 K-Means 聚类算法。第 1 种策略基于 CPU 和内存实现遥感图像的聚类，简称“CPU+内存”；第 2 种策略基于 CUDA 和内存实现，简称“CUDA+内存”；第 3 种策略基于 CUDA 和内存映射文件实现，简称“CUDA+ MMF”，第 4 种策略基于 CUDA 和图像分块方法实现，简称“CUDA+分块”。表 1 中包含了各种实现策略的算法复杂度。

关于算法的时间复杂度，由于 GPU 通常拥有数百个核，即 m 值很大，理论上时间复杂度可以达到原有算法的 1/m，但在实现时由于 CUDA 复杂的执行模型、CPU 与 GPU 时钟频率的差异以及数据在内存和设备端双向拷贝花费的时间等因素影响，其实际时间复杂度一般高于理论时间复杂度。“CUDA+MMF”策略只比“CUDA+内存”策略增加了一次图像读取和写入磁盘的时间，而“CUDA+分块”策略在每一轮迭代都需要增加一次图像读取和写入磁盘的时间，因此其时间复杂度大大提高。

表 1　四种实现策略的算法复杂度

Tab. 1　Algorithm complexity of four kinds of implementation strategies

实现策略名称	算法时间复杂度	算法空间复杂度
CPU+内存	$O(nkt)$	$O((n+k)\cdot p)$
CUDA+内存	$O((2np/C+\alpha nk/m)\cdot t)$	$O((n+k)\cdot p)$
CUDA+MMF	$O(2np/D+(2np/C+\alpha nk/m)\cdot t)$	$O((B+k)\cdot p)$　$B \ll n$
CUDA+分块	$O((2np/D+2np/C+\alpha nk/m)\cdot t)$	$O((B+k)\cdot p)$　$B \ll n$

关于算法的空间复杂度，“CUDA+MMF”和“CUDA+分块”策略的空间复杂度是相同的。由于 B 远小于 n，而 k 可以忽略不计，因此后两种实现策略的空间复杂度远远低于前两种实现策略。

3　实验与分析

本文中采用的实验数据是广州省雷州湾的 SPOT-5 多光谱遥感图像，图像尺寸 6 000×6 000 像素，大小 137MB，包含绿、红、近红外、短波红外四个波段，其假彩色图像如图 1 所示。

图 1　广州省雷州湾 SPOT-5 多光谱遥感图像

Fig. 1　The SPOT-5 multi-spectral remote sensing image of Guangzhou Leizhou Bay

计算机的硬件环境配置如下：CPU(Intel Core 2 Duo Processor E6550, 4M Cache, 2.33 GHz)；显卡(NVIDIA GeForce GTX 650 Ti, 768 个流处理器, GPU 核心时钟 324MHz，内存时钟 162MHz，显存 1.0GB)；内存 2.0GB。软件环境配置如下：操作系统 Windows XP；运行平台 CUDA 5.0；开发环境 Visual Studio 2010；编程语言 C++。

K-Means 聚类算法设定的收敛条件是连续两次迭代得到的聚类中心完全相同或很相似，可以采用绝对距离、欧氏距离、马氏距离、曼哈顿距离、夹角余弦[18]距离度量公式进行相似性评价。本文采用了夹角余弦法，因为它的计算结果不依赖于图像的数据类型和取值范围，鲁棒性较好。当对应聚类中心的夹角余弦值都大于某个给定的阈值(如 0.999 9)时，算法终止。

为了检验 K-Means 聚类算法在不同实现策略下的性能，本文使用了一幅裁切后的小图像(600×600 像素)和整景图像(6 000×6 000 像素)进行了实验。由于 K-Means 聚类算法的初始聚类中心是随机选择的，所以每次算法执行需要的迭代次数都太不相同。为此本文做了 12 组对比实验，每一组实验随机选择 15 个初始聚类中心，然后执行各种 K-Means 聚类算法的实现策略。实验结果如表 2 中所示。

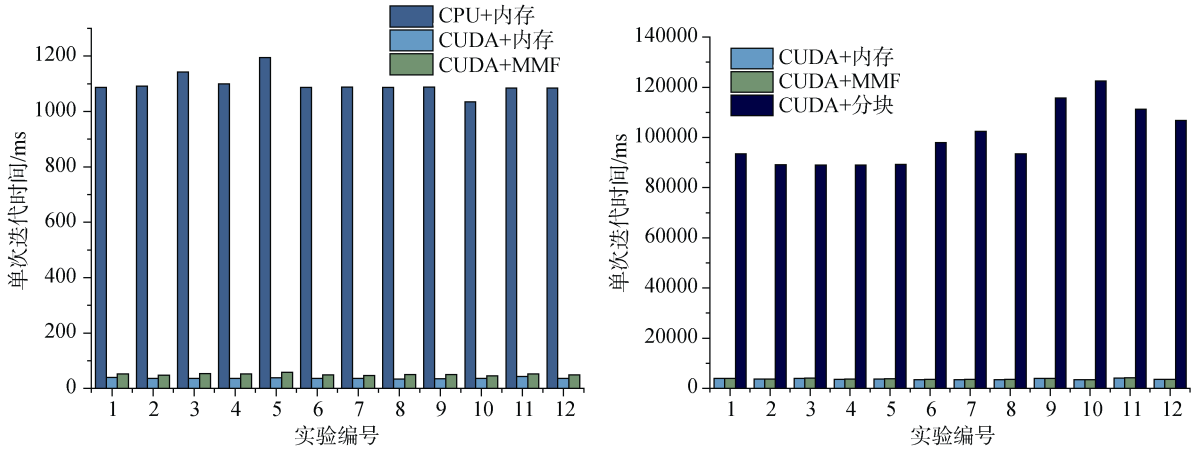
表 2　K-Means 聚类算法四种实现策略的执行时间(ms)

Tab. 2　Execution time of four kinds of implementation strategies for K-Means clustering algorithm (ms)

实验编号	迭代次数	图像大小(600×600 像素)			图像大小(6000×6000 像素)		
		CPU+内存	CUDA+内存	CUDA+MMF	CUDA+内存	CUDA+MMF	CUDA+分块
1	8	8 698	421	323	30 992	31 615	747 595
2	24	26 184	1 152	854	87 108	87 249	2 137 526
3	11	12 563	593	392	44 124	44 961	979 234
4	13	14 281	687	463	46 852	47 863	1 157 276
5	9	10 750	520	350	33 568	34 159	803 184
6	21	22 837	1 031	754	71 854	73 472	2 056 384
7	22	23 939	1 034	812	75 628	76 912	2 252 206
8	33	35 881	1 652	1 134	114 345	116 883	3 084 543
9	16	17 402	797	560	62 984	63 416	1 851 616
10	30	31 058	1 391	1 078	101 220	103 530	3 674 140
11	7	7 589	361	298	28 618	29 275	778 928
12	26	28 210	1 264	934	91 724	92 726	2 777 424

根据 K-Means 聚类算法的执行时间和迭代次数，可以求出算法单次迭代需要花费的时间。对于 600×600 像素的图像，“ CPU+内存 ”、“ CUDA+内存 ”和“ CUDA+MMF ”三种策略单次迭代花费的时间如图 2(a)所示。可以看出，“ CUDA+内存 ”和“ CUDA+MMF ”策略花费的时间远远小于“ CPU+内存 ”策略花费的时间，只有后者的 3%左右，其中“ CUDA+MMF ”策略花费的时间略高于“ CUDA+内存 ”策略花费的时间。

对于6000×6000像素的SPOT-5整景图像，“ CUDA+内存 ”、“ CUDA+MMF ”和“ CUDA+分块 ”三种策略单次迭代花费的时间如图 2(b)所示。可以看出“ CUDA+内存 ”和“ CUDA+MMF ”策略花费的时间基本相同，并且远远小于“ CUDA+分块 ”策略花费的时间，也只有后者的 3%左右。由此可见 MMF 技术确实能够极大地提高 K-Means 聚类算法的时间效率。



(a) 600×600 像素 SPOT-5 遥感图像　　(b)6000×6000 像素 SPOT-5 遥感图像

图 2　K-Means 聚类算法在各种实现策略中单次迭代花费的时间

Fig. 2　The time of single iteration ofK-Means clustering algorithm in a variety of strategies

图 3 中显示了“ CUDA+内存 ”、“ CUDA+MMF ”和“ CUDA+分块 ”3 种策略占用的内存数量。对于 600×600 像素的小图像，“ CUDA+MMF ”和“ CUDA+分块 ”策略占用的内存略小于“ CUDA+内存 ”策略占用的内存；对于 3 000×3 000 像素的中等大小图像，“ CUDA+MMF ”策略和“ CUDA+分块 ”策略占用的内存明显小于“ CUDA+内存 ”策略占用的内存，只有后者的 25%左右；对于 6 000×6 000 像素的大图像，“ CUDA+MMF ”和“ CUDA+分块 ”策略占用的内存远小于“ CUDA+内存 ”策略占用的内存，只有后者的 10%左右。由此可见，遥感图像的尺寸越大，使用分块处理方法的“ CUDA+MMF ”和“ CUDA+分块 ”策略对内存数量的节省越明显。

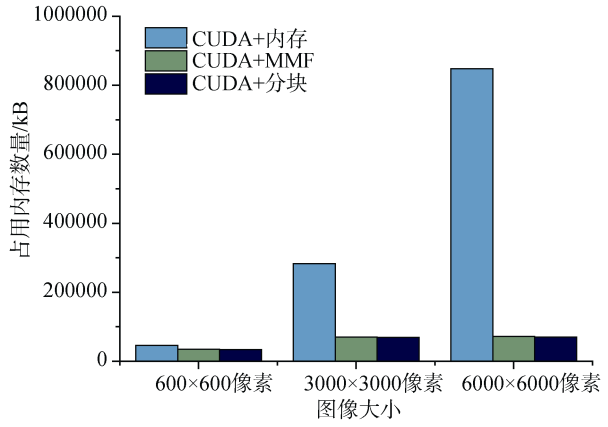


图 3　K-Means 聚类算法在各种实现策略中占用的内存数量

Fig. 3　The occupied memoryof K-Means clustering algorithm in a variety of strategies

从图 3 中还可以看出，使用“ CUDA+内存 ”策略对 6 000×6 000 像素的 137MB 的 SPOT-5 多光谱图像进行处理时，占用的内存数量已经超过 800MB。如果需要处理更大数据量的遥感图像，可能会出现内存不足的问题。使用“ CUDA+MMF ”或“ CUDA+分块 ”策略，当图像数据量从 34MB(3000×3000 像素)增到 137MB 时，算法占用的内存数量基本保持在 70MB 左右不变，因此这两种策略都能够处理特大数据量的遥感图像。

通过上面的讨论可以看出，“ CUDA+内存 ”策略运行速度最快，但是占用内存数量较大；“ CUDA+ 分块 ”策略占用内存数量较小，但是运行速度较慢；“ CUDA+MMF ”策略运行速度与“ CUDA+内存 ”

策略相近，占用内存数量与“CUDA+分块”策略相同，显然它是最优的实现策略。“CPU+内存”策略不仅运行速度较慢，而且占用内存数量较大，通常只适用于处理较小的遥感图像。

本文还研究了瓦片大小对“CUDA+MMF”实现策略执行时间的影响，总共做了 5 组对比试验。每一组实验中瓦片的大小分别设定为 128×128、256×256、512×512 和 1024×1024 四种尺寸，占用的内存数量如表 3 所示。可以看出，随着瓦片尺寸变大，“CUDA+MMF”策略的执行时间呈下降趋势，但是下降幅度很小，通常情况下瓦片大小设定为 512×512 像素或 256×256 像素即可。

表 3　各种瓦片大小下 CUDA+MMF 策略的执行时间(ms)
Tab. 3　The execution time of CUDA + MMF strategy in various tile sizes(ms)

瓦片大小	迭代次数				
	8	12	20	25	30
128×128	32 094	46 409	76 076	96 016	115 026
256×256	31 678	45 423	70 422	88 110	105 632
512×512	31 500	43 422	69 698	87 157	103 578
1024×1024	30 561	42 907	68 282	83 437	102 813

4　结论

本文提出了一种结合 CUDA 并行计算技术、分块处理方法和内存映射文件技术的遥感图像处理算法优化方法。内存映射文件技术的引入不仅保留了分块处理方法内存占用少的优点，而且图像文件访问速度与直接内存访问几乎相同。CUDA 并行计算技术可以在不增加计算资源的条件下极大地提高遥感图像的处理速度。本文介绍了 K-Means 聚类算法的“CUDA+内存”和“CUDA+MMF”两种优化策略的实现方法，并且分析了各种实现策略的时间复杂度和空间复杂度。实验结果表明，与传统 K-Means 聚类算法相比，“CUDA+MMF”实现策略能够提高计算速度 30 多倍，而占用的计算机内存数量仅为原有算法的 10%左右。此外，图像分块时瓦片的大小对算法性能影响不大。

参考文献:

[1] 申焕, 石晓春, 邱宏华. 基于 MPI 的海量遥感影像并行处理技术探析[J].全球定位系统, 2012, 37(6), 73-76.
ShenHuan, Shi Xiaochun, QiuHonghua. Study on parallel processing technology of massive remote sensing Iamge Based on MPI[J]. GNSS World of China, 2012, 37(6): 73-76.

[2] Kika A, Greca S. Multithreading Image Processing in Single-core and Multi-core CPU using Java[J]. International journal of advanced computer science & applications, 2013, 4(9): 165-169.

[3] Torti E, Danese G, Leporati F, et al. A hybrid CPU–GPU real-time hyperspectral unmixing chain[J]. IEEE Journal of Selected Topics in Applied Earth Observations & Remote Sensing, 2016, 9(2): 945-951.

[4] 闸旋. CPU+GPU 单机异构环境下遥感数据并行处理技术研究[D]. 郑州: 解放军信息工程大学, 2013.
Zhaxuan. Research on parallel processing technologies of remote sensing data based on CPU+GPU under single-computer[D]. Zhengzhou: The PLA Information Engineering University.

[5] 汤媛媛, 周海芳, 方民权, 等. 基于 CPU/GPU 异构模式的高光谱遥感影像数据处理研究与实现[J]. 计算机科学, 2016, 43(2): 47-50.
Tang Yuanyuan, Zhou haifang, Fang minquan, et al. Hyperspectral remote sensing image data processing research and realization based on CPU/GPU heterogeneous model[J]. Computer Science, 2016, 43(2): 47-50.

[6] Bernabe S, Lopez S, Plaza A, et al. GPU Implementation of an automatic target detection and classification algorithm for hyperspectral image analysis[J]. IEEE Geoscience & Remote Sensing Letters, 2012, 10(2): 221-225.

[7] Wu Z, Wang Q, Plaza A, et al. Parallel Implementation of sparse representation classifiers for hyperspectral imagery on GPUs[J]. IEEE Journal of Selected Topics in Applied Earth Observations & Remote Sensing, 2015, 8(6): 2912-2925.

[8] 汤媛媛, 周海芳, 方民权, 等. 基于 GPU 的高光谱遥感影像数据处理[J]. 信息安全与技术, 2015(4): 46-51.
Tang Yuanyuan, Zhou haifang, Fang minquan, et al. Hyperspectral remote sensing image data processing on GPU[J]. Information Security and Technology, 2015(4): 46-51.

[9] 王化喆, 魏先勇, 等. 基于 GPU 的遥感图像前期处理算法研究与应用[J]. 现代电子技术, 2016(3): 47-50.

Wang Huazhe, WeiXianyong, et al. Research and application of GPU-based preprocessing algorithms for remote sensing image[J]. Modern Electronics Technique, 2016(3): 47-50.

[10] 葛澎. 分布式计算技术概述[J]. 微电子学与计算机, 2012, 29(5): 201-204.
Ge Peng. A Brief overview on distributed computing technology[J]. Microelectronics & Computer, 2012, 29(5): 201-204.

[11] 刘小利, 徐攀登, 朱国宾, 等. 结合 MapReduce 和 HBase 的遥感图像并行分布式查询[J]. 地理与地理信息科学, 2014, 30(5): 26-28.
Liu Xiaoli, Xu Pandeng, Zhu Guobin, et al. Parallel and distributed retrieval of remote sensing image using hbase and map reduce[J]. Geography and Geo-Information Science, 2014, 30(5): 26-28.

[12] Olasz A, Kristóf D, Thai B N, et al. Processing big remote sensing data for fast flood detection in a distributed computing environment[J]. 2017, XLII-4/ W2: 137-138.

[13] 刘平, 贾林林. 内存映射技术在大数据实时存储中的应用[J]. 河南科技, 2017(5): 39-41.
LIN Ping, Jia Linlin. Application of memory mapping technology in large data storage[J]. Journal of Henan Science and Technology, 2017(5): 39-41.

[14] 胡伟忠, 刘南, 刘仁义. 基于内存映射文件技术的海量影像数据快速读取方法[J]. 计算机应用研究, 2005, 22(2): 111-112.
Hu Weizhong, Liu Nan, Liu Renyi. Quick-read means for large scale images based on storage mapping file technique[J]. Application Research of computers, 2005, 22(2): 111-112.

[15] Jeong I K, Hong M G, Hahn K S, et al. Performance study of satellite image processing on graphics processors unit using CUDA[J]. Journal of International Criminal Justice, 2012, 28(6): 683-691.

[16] 张琳. 基于CUDA架构的高性能图像处理程序设计[D]. 成都: 电子科技大学, 2014.
Zhang Lin. CUDA Architecture-based high-performance image processing program design[D]. Chengdu: University of Electronic Science and Technology, 2014.

[17] 郭忠明. 基于 CUDA 的并行图像处理性能优化[D]. 大连: 大连理工大学, 2012.
Guo Zhongming. The Performance optimization of parallel image processing based on CUDA[D]. Dalian: Dalian University of Technology, 2012.

[18] Gang Qian, Shamik Sural, Yuelong Gu, et al. Similarity between euclidean and cosine angle distance for nearest neighbor queries[C]//ACM Symposium on Applied Computing, 2004: 1232-1237.

A fast processing method for high-resolution remote sensing image based on GPU and memory mapping file

MA Xiu-dan, CUI Bin-ge, ZHONG Yong, ZHANG Yong-hui, FEI Dong

(College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China)

Received: Oct. 11, 2017
Key words: Memory mapping file; CUDA; Optimization; Remote sensing image; Clustering

Abstract: High-resolution remote sensing image processing programs often encounter the problems of lengthy execution time and insufficient memory space because of the limitations of computer processor frequency and memory capacity. Parallel computing can enhance remote sensing image processing speed, but is unable to reduce the requirements for memory space. In this paper, we present an approach to simultaneously optimize the time and space efficiencies for image processing programs. This has been previously implemented for the K-Means algorithm. CUDA can effectively take advantage of GPU powerful parallel computing capabilities, and memory mapping file can offset the impact of a slow disk I/O performance on algorithms. Experimental results show that our method is about 30 times faster than the traditional K-Means clustering algorithm, and the memory usage decreased by more than 90%.

(本文编辑: 康亦兼)