doi:10.11887/j.cn.201504011 http://journal.nudt.edu.cn

GPU上高光谱快速ICA降维并行算法*

方民权,周海芳,张卫民,申小龙
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:高光谱影像降维快速独立成分分析过程包含大规模矩阵运算和大量迭代计算。通过分析算法热点,设计协方差矩阵计算、白化处理、ICA迭代和IC变换等关键热点的图像处理单元映射方案,提出并实现一种G-FastICA并行算法,并基于GPU架构研究算法优化策略。实验结果显示:在处理高光谱影像降维时,CPU/GPU异构系统能获得比CPU更高效的性能,G-FastICA算法比串行最高可获得72倍加速比,比16核CPU并行处理快4~6.5倍。

关键词:图像处理单元;高光谱影像降维;快速独立成分分析;并行算法;性能优化
中图分类号:TP391 **文献标志码**:A **文章编号**:1001-2486(2015)04-065-06

A parallel algorithm of FastICA dimensionality reduction for hyperspectral image on GPU

FANG Minquan, *ZHOU Haifang*, *ZHANG Weimin*, *SHEN Xiaolong*
(College of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: Fast independent component analysis dimensionality reduction for hyperspectral image needs a large amount of matrix and iterative computation. By analyzing hotspots of the fast independent component analysis algorithm, such as covariance matrix calculation, white processing, ICA iteration and IC transformation, a GPU-oriented mapping scheme and the optimization strategy based on GPU-oriented algorithm on memory accessing and computation-communication overlapping were proposed. The performance impact of thread-block size was also investigated. Experimental results show that better performance was obtained when dealing with the hyperspectral image dimensionality reduction problem: the GPU-oriented fast independent component analysis algorithm can reach a speedup of 72 times than the sequential code on CPU, and it runs 4~6.5 times faster than the case when using a 16-core CPU.

Key words: graphic processing unit; hyperspectral image dimensionality reduction; fast independent component analysis; parallel algorithm; performance optimization

高光谱遥感技术广泛应用于军事、农业、环境科学、地质、海洋学等领域,这些领域迫切要求及时处理[1]。但高光谱影像数据有波段多、数据量大、相关性强、冗余多等特点,直接处理将导致维数灾难、空空间现象等严重的计算问题[2-4]。因此,国内外专家学者处理高光谱影像数据过程常常包括降维步骤。降维通过一定的映射,将高维影像数据转换成低维影像数据,并保持信息量基本不变。

高光谱影像降维方法可分为线性和非线性两大类:线性降维有主成分分析(Principal Component Analysis, PCA)[5]、独立成分分析(Independent Component Analysis, ICA)[6]等方法;非线性降维包括基于核的方法和基于流行学习的方法,如等距映射法(ISOmetric MAPping, ISOMAP)[7]、局部线性嵌入(Locally Linear Embedding, LLE)[8]等。独立成分分析是应用最广泛的降维方法之一,其降维过程包含了大规模矩阵计算和大量迭代运算。由于高光谱影像波段多、数据量大等特征,降维处理过程复杂且耗时,传统的串行降维已无法满足各行业的需求,因此并行降维的研究已迫在眉睫。

自2007年Nvidia正式推出统一计算设备架构(Compute Unified Device Architecture, CUDA)以来,图像处理单元在通用计算领域发展迅猛。CPU/GPU的异构系统可编程性好、性价比高、功耗比低,已成为高性能计算领域的黑马。搭载这种异构系统的天河1A和泰坦分别在2010年和

* 收稿日期:2014-09-28
基金项目:国家自然科学基金资助项目(61272146,41375113,41305101)
作者简介:方民权(1989—),男,浙江东阳人,博士研究生,E-mail:fmq@hpc6.com;
周海芳(通信作者),女,教授,博士,硕士生导师,E-mail:haifang_zhou@163.net

2012 年夺得 TOP500 榜首[9]。CPU/GPU 异构系统的快速发展为高光谱影像降维的实时处理提供了可能。

高光谱影像并行处理在传统并行系统中已有成熟的应用，David 等[10]基于异构讯息传递接口(Message Passing Interface，MPI)在网络集群系统研究了高光谱影像处理的技术；Javier 等[11]提出基于神经网络的高光谱影像并行分类算法。在 CPU/GPU 异构系统上，Sanchez 等[12]对高光谱解混进行了 GPU 移植；Yang 等[13]利用多 GPU 实现了高光谱影像快速波段选择；Rui 等[14]在 GPU 上实现 FastICA 算法，加速 55 倍；Jacquelyne 等[15]用 OpenCL 在 GPU 上实现 Infomax ICA，最高加速 56 倍。但这些研究较少涉及基于 GPU 研究高光谱影像降维。本文研究基于负熵最大的 FastICA 降维算法[16-17]，针对算法热点设计 GPU 映射方案，最终提出并实现一种 G-FastICA 并行算法。

1 FastICA 算法与热点分析

1.1 基于负熵最大的 FastICA

FastICA 算法有基于负熵最大、基于似然最大、基于峭度 3 种形式，本文主要研究基于负熵最大的 FastICA 算法。该算法是盲源降维方法，以负熵最大为搜寻方向，实现独立源的顺序提取，采用定点迭代的优化方法，收敛更加快速、稳健[16-17]。

对于高光谱影像数据 $\boldsymbol{X}(W\times H\times B$，宽 W、高 H、波段 $B)$，$\boldsymbol{X}$ 可视为 B 行 S 列($S=W\times H$)的二维矩阵，矩阵的一行表示一个波段的高光谱影像数据。通过独立成分分析方法降维，可以获得 m 个 IC 图像，其中 $m<B$，从而达到降维目的。具体算法[15-16]如下：

step1. 计算 $\boldsymbol{X}$ 的协方差矩阵；

step2. 计算协方差矩阵的特征值及特征向量，并根据阈值 T 选取最终 IC 图像数量 m；

step3. 计算白化矩阵 $\boldsymbol{M}=\boldsymbol{D}^{-1/2}\boldsymbol{V}^{\mathrm{T}}$，其中 $\boldsymbol{D}$ 为 $\boldsymbol{X}$ 的特征值矩阵，$\boldsymbol{V}$ 为对应的特征向量矩阵；

step4. 高光谱影像数据白化处理 $\boldsymbol{Z}=\boldsymbol{MX}$，$\boldsymbol{X}$ 要减去均值(即零均值处理)，$\boldsymbol{M}$ 为白化矩阵，$\boldsymbol{Z}$ 为白化处理结果矩阵；

step5. 迭代开始：

①初始化 $\boldsymbol{W}_i$，$\boldsymbol{W}_i$ 为变换矩阵 $\boldsymbol{W}$ 的一列；

②$\boldsymbol{W}_i=E\{\boldsymbol{Z}g(\boldsymbol{W}_i^{\mathrm{T}}\boldsymbol{Z})\}-E\{g'(\boldsymbol{W}_i^{\mathrm{T}}\boldsymbol{Z})\}\boldsymbol{W}$；

其中 g 为非线性函数，经测试，取 $g(y)=y^3$ 可更稳定、更快地达到 ICA 迭代收敛。

③正交化 $\boldsymbol{W}_i=\boldsymbol{W}_i-\sum(\boldsymbol{W}_i^{\mathrm{T}}\boldsymbol{W}_j)\boldsymbol{W}_j$；

④归一化 $\boldsymbol{W}_i=\boldsymbol{W}_i/\|\boldsymbol{W}_i\|$；

⑤判断是否收敛，不收敛则返回到步骤②，若收敛，如果所有的 $\boldsymbol{W}_i$ 已经计算结束(即 $i=m$)，则退出，否则令 $i=i+1$，返回到步骤①计算 $\boldsymbol{W}_i$；

step6. 计算独立成分 $\boldsymbol{Y}=\boldsymbol{WZ}$；

1.2 加速热点分析

实现串行高光谱影像 FastICA 降维算法，对宽 781、高 6955、波段数 224 的高光谱影像降维，测试并统计各步骤占总计算时间的比率(见表 1)。表中数据显示，协方差矩阵计算(COV)、白化处理(white process)和 ICA 迭代(ICA iteration)过程比重较大，共占总计算时间的 99%，是并行研究的关键热点。此外特征值计算取决于波段数 B，本文数据波段数均为 224，因此特征值特征向量的计算属于不变串行分量，无须考虑并行；IC 变换(IC transformation)、数据 I/O 与高光谱影像数据的宽、高、波段数及其信息特征相关，作为次要热点考虑其并行化。

表 1 FastICA 时间分布

Tab. 1 Time distribution of FastICA

步骤	时间/s	百分比/%
input	2.61	0.36
COV	218.08	29.93
eigenvalue	0.33	0.05
white process	135.83	18.64
ICA iteration	368.17	50.53
IC transformation	2.56	0.35
output	1.07	0.15

2 FastICA 算法并行化研究

2.1 协方差矩阵计算的 GPU 映射

协方差矩阵是对称阵，仅需计算其下三角元素，再将值填入对应的上三角阵。针对该过程，设计了一种映射方案：GPU 上的每个线程块完成协方差矩阵中一个协方差的计算任务，如图 1 所示。启动 $B\times(B+1)/2$ 个 block，每个 block 中的 thread 数量可根据 occupancy 选取最佳的 thread 数。其思想是将协方差矩阵中单个协方差的计算任务映射到对应的 block 中，由一个 block 完成一个协方差元素的计算。一种实现策略是：每次启动维度为 <<< i, blockDim >>> (i 为循环变量)的

kernel 函数,循环 B 次完成所有协方差矩阵元素的计算任务。

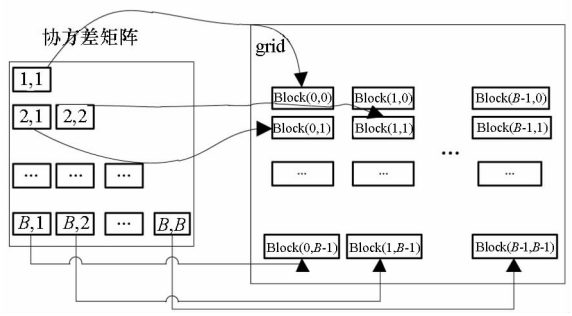

图 1　协方差矩阵的 GPU 映射方案

Fig. 1　GPU mapping scheme of covariance matrix

2.2　设备端并行白化处理

GPU 是细粒度并行协处理器,在白化处理中寻找细粒度的并行计算任务,再将这些细粒度计算任务通过 kernel 函数映射到 GPU 核上执行,这是设备端并行白化处理的关键。

在白化处理($\boldsymbol{Z}=\boldsymbol{MX}$)中,$\boldsymbol{M}$ 是 $B\times m$ 的矩阵,$\boldsymbol{X}$ 的尺寸为 $S\times B$,结果矩阵 $\boldsymbol{Z}$ 大小为 $S\times m$。结果矩阵每个元素的计算是独立的,且计算任务量类似,是长为 B 的向量内积。高光谱影像的波段数 B 一般为几十到上百不等,即向量内积的运算量有限,是细粒度任务,故视结果矩阵的单个元素的计算为单元任务,将其映射到 GPU 的 thread 上。整个结果矩阵共有 $S\times m$ 个单元任务,需要 $S\times m$个线程参与计算;但高光谱影像的宽 W 和高 H 较大(约数百到上千),故 $S=W\times H$ 较大,$S\times m$ 的数量级更大,超过 GPU 一次可启动的最大线程数量,无法直接运算。此时,采用循环计算的方法,即启动适当数量的 block 和 thread,多次循环完成计算任务。对于线程块 blockIdx 中的线程 threadIdx,需计算索引为 $i\times$(gridDim × blockDim) + blockIdx × blockDim + threadIdx 的任务(i 为循环变量),直到索引超过 $S\times m$。该方案中,gridDim 和 blockDim 不受算法自身约束,可根据 occupancy 自由设定,以最大化 GPU 设备占用率,从而获得最佳性能。

2.3　ICA 迭代的 GPU 移植

ICA 迭代过程中,受数据依赖限制,不同迭代无法同时运算,只能在单次迭代中开发并行。分析迭代中各个步骤的计算量,初始化、正交化、归一化和判断操作的计算量很小,可不必考虑,主要的计算量集中在步骤②中。分解该步骤,抽象得到图 2 所示的 ICA 迭代宏观模型。

$\boldsymbol{W}_i^{\mathrm{T}}\boldsymbol{Z}$ 模型类似于白化处理,在 GPU 任务映射时,可借鉴白化处理的映射方案,即启动 <<< gridDim, blockDim >>> 的核函数计算。$\boldsymbol{Z}g(\boldsymbol{W}_i^{\mathrm{T}}\boldsymbol{Z})$的特点是向量内积的向量元素多,单次计算量大,计算次数少,故将一个向量内积运算视为大任务,通过核函数 <<< gridDim,blockDim >>> 映射到 GPU,共需循环 m 次。线程 threadIdx 循环处理索引为((blockIdx × blockDim + threadIdx) + gridDim × blockDim × i)的元素乘加运算,共获得 gridDim × blockDim 个中间结果;后进行 block 内归约,得到 gridDim 个中间结果;由于 GPU 中 block 间无法同步,还需要一个 <<< 1,gridDim >>> 的核函数来完成归约计算,最终得到长向量内积。

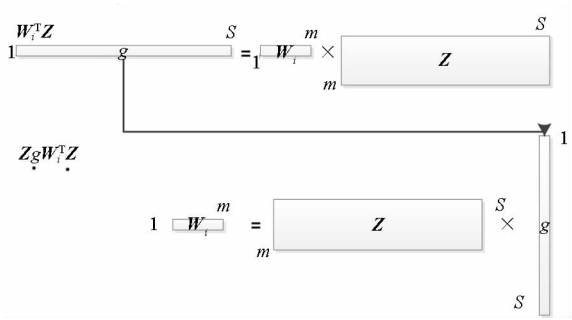

图 2　ICA 迭代关键步骤模型

Fig. 2　Model of key step for ICA iteration

2.4　G-FastICA 算法

IC 变换与白化处理的运算模型类似,因此将前文设计的白化处理并行策略套用到 IC 变换中。高光谱影像数据 I/O 的并行化依赖于并行文件系统或缓存机制,利用多线程技术,每个线程创建独立的文件指针,通过合理的数据分割,各个线程将指针指向各自的目标位置,再进行输入输出操作。

针对单 GPU,根据前面提出的协方差矩阵计算、白化处理、ICA 迭代、IC 变换等热点的 GPU 映射并行方案,用其替代串行算法的对应步骤,可实现 CPU 与 GPU 协同计算高光谱影像 FastICA 降维过程,称之为 G-FastICA 算法。

3　GPU 性能优化

基于 GPU 的计算通信重叠优化:协方差矩阵计算,存在着计算通信重叠的可能性。由图 1 可知,第 i 行计算使用高光谱影像数据前 i 个波段的数据。传输完第 1 个波段数据后,协方差矩阵第 1 行计算和第 2 波段数据传输可同时进行,此时计算与通信能够重叠。采用多流的方法实现 GPU 计算与 CPU/GPU 通信的并发执行。得益于计算通信重叠,几乎能掩盖全部高光谱影像数据传输时间。

GPU 存储优化:GPU 提供了层次分明的存储单元系统,按离核距离由近及远分别是寄存器、共享存储器、全局存储器,可以显示这些存储单元的使用。离核距离近的延迟低,访存效率高,因此应尽量使用离核近的存储单元。本文通过下面两项技术对各并行热点 GPU 进行优化(GPU 映射设计时已保证访存对齐):①共享存储访问取代全局存储访问;②采用寄存器存储数量较少的中间变量。通过这两个步骤修改,各热点均能获得部分性能提升,效果见表 2(优化手段 0 表示无优化,1 采取 GPU 存储优化手段 1,2 在 1 基础上采用 GPU 存储优化手段 2,3 在 2 基础上增加计算通信重叠优化)。表中数据显示了本文优化效果。

表 2 GPU 存储优化技术效果

Fig. 2 Effects of storage optimizing method on GPU ms

优化手段	协方差	白化处理	每次迭代	IC 变换
串行 O2	218 081.57	135 831.59	787.24	2562.71
0	16 009.95	775.09	10.18	154.83
1	8019.15	767.07	7.84	204.40
2	6180.91	447.17	6.68	113.22
3	4070.48	496.55	6.33	113.64

最佳维度设计:GPU 是众核架构,细粒度并行,对于线程数量非常敏感,因此需要合理设计线程维度来保证 GPU 性能发挥。GPU 上的线程数量可以是任意的,全部测试是不可能的,需要一套理论指导线程数量的选择。本文设计了一种线程维度选择方法,步骤如下:

1)计算 kernel 的 GPU 设备占用率(occupancy);

2)选择占用率最高的几个线程数量进行实验测试,择优选取。

GPU 设备占用率是发挥 GPU 性能的一项重要指标,主要受限于活动的 warp 数目,而活动的 warp 数目与使用的寄存器、共享存储器有关。通过统计 kernel 函数使用的寄存器数量和共享存储器容量,可以计算出可活动 warp 的数量,再根据 warp 数量与线程数量关系图,可以初步选定几个占用率高的线程数量。针对上述选取的线程数量逐一进行实验测试,对比性能,可得到最佳线程维度。

比如在协方差矩阵计算中,每个 thread 需要 18 个寄存器,每个 block 需要 3072byte 的共享存储,对 GPU 的最大活动 warp 数量进行分析,通过计算可知,寄存器和共享存储器的数量都不会限制 warp 数量;此时,warp 数量受最大 warp 数量限制,与线程数量有关,其关系见图 3。选择线程数量 128,256,512,1024 时,活动 warp 数量达到最大 64,occupancy 达到 100%。再对这 4 个线程维度进行实验测试,统计并对比结果数据,获得最佳线程维度。

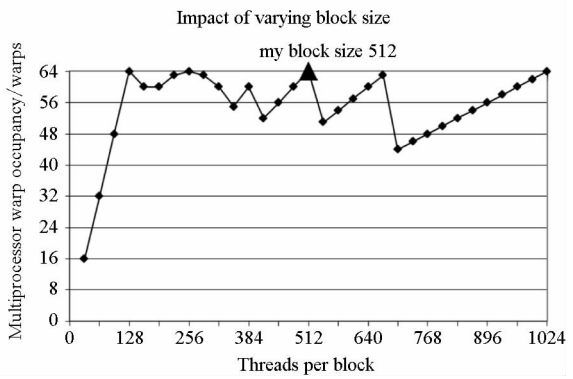

图 3 线程数量对 warp 数量的影响(协方差)

Fig. 3 Effect between threads and warps(COV)

计算所有 GPU 并行热点 occupancy,选择线程数量使得活动 warp 数量最大化,测试并统计其结果,见表 3。表中数据显示,当线程数量为 1024 时,白化处理可以获得最高性能;其他热点在线程数量为 512 时性能最佳。

表 3 线程数量对性能的影响

Fig. 3 Performance effects of threads number ms

线程数量	协方差	白化处理	每次迭代	IC 变换
128	5924.22	634.07	7.68	132.97
256	4723.96	702.25	6.83	115.34
512	4070.48	496.55	6.33	113.64
1024	4240.85	453.14	8.41	121.64

4 实验结果

4.1 实验平台与测试数据

本文用统一计算设备架构(Compute Unifed Device Architecture,CUDA)技术实现上述 GPU 并行映射方案和优化策略。其中涉及的矩阵、向量运算均手动实现,未使用 cublas 库,主要有以下几个方面的考虑:1)空间复杂度,cublas 库要求数据类型是 float,而高光谱影像数据类型为 unsigned char,如需转换将增加 3 倍存储空间,可能超过 GPU 全局存储容量,导致无法处理;2)除开 unsigned char 到 float 类型的转换开销,GPU 或集成众核(Many Integrated Core,MIC)等协处理器访

问 unsigned char 类型的延迟比 float 类型小很多;3)cublas 库函数任务无法拆分,必须先通信后计算,将导致无法使用计算通信重叠的优化策略;4)由于高光谱数据的特点,矩阵运算不规整,用通用方法优化的 cublas 未必能获得更好的效果。

实验采用当前主流的 GPU 微型超算平台,搭载 2 个 8 核 Intel Xeon E5 - 2650 CPU 和 nVidia Tesla K20 GPU。软件平台采用 Linux CentOS 6.2 操作系统、GCC 4.4.6 编译器、CUDA5.0 工具包。

本文采用表 4 罗列的 5 组机载可见光成像光谱仪(Airborne Visible Infrared Imaging Spectrometer, AVIRIS)高光谱影像数据。数据已经过预处理,把 16 位 int 型光谱信息转换为 unsigned char 类型的像素信息。

表 4　高光谱影像数据详细信息

Tab. 4　Information of hyperspectral image data

序号	宽	高	波段
1	614	512	224
2	614	1087	224
3	753	1924	224
4	781	6955	224
5	1562	6955	224

关于 ICA 迭代次数的约定:由于高光谱影像数据和 ICA 迭代过程中取得的随机数不同,ICA 迭代次数也将有所差异,不同的迭代次数导致时间消耗不可比。为了较准确地对比算法性能,本文做出如下约定:相同的数据取固定的迭代次数。通过多次实验取平均值的方法,确定各组数据的迭代次数(见表 5)。

表 5　高光谱数据迭代次数

Tab. 5　Iterations of hyperspectral data

数据	1	2	3	4	5
迭代次数	209	333	859	468	486

4.2　两种测评基准

本文从最优串行和 CPU 满负载并行两个方面对算法性能进行测评。

1)最优串行程序。表 6 分别罗列了开启优化开关 O0,O1,O2,O3 时,处理高光谱影像数据 1 的时间。优化开关为 O2 时,串行程序执行时间最短,作为与并行算法对比的标准。

表 6　串行程序优化执行时间表

Tab. 6　Execution time table of serial program optimization

优化开关	O0	O1	O2	O3
时间/s	92.00	37.16	33.55	33.82

本文通过对比线程级和进程级并行程序的性能,采取性能较好的 CPU 并行方案作为 CPU 满负载并行标准。本文分别采取 OpenMP 多线程并行和 MPI 多进程并行,实现这两种并行 FastICA 降维算法,测试性能并比较,见表 7。对比表中数据,MPI 并行程序能获得更好的性能,因此本文以 MPI 并行 FastICA 程序作为 CPU 满负载并行程序。

表 7　CPU 并行程序加速效果

Tab. 7　Speed-up of CPU parallel programs

数据	串行时间/s	OMP 时间/s	加速比	MPI 时间/s	加速比
1	33.55	2.89	11.6	3.78	8.9
2	112.16	10.38	10.8	10.34	10.8
3	379.34	46.28	8.2	38.46	9.9
4	728.65	70.37	10.4	65.47	11.1
5	1438.16	152.20	9.4	100.01	14.4

4.3　GPU 对 CPU 的性能优势

统计 G - FastICA 高光谱影像降维算法中各步骤的耗时信息,对比最佳串行程序(见图 4)和 MPI 并行程序(见图 5)。图 4 数据显示,本文设计的各热点 GPU 映射方案能获得比串行快几十倍的加速比,其中协方差矩阵计算加速近 50 倍,白化处理加速 300 倍左右,ICA 迭代过程加速 42 ~ 109 倍不等,IC 变换可加速 19 ~ 41 倍。图 5 显示 GPU 相对 CPU 在并行计算上更有优势,本文设计的 GPU 映射方案比 CPU 并行版本(MPI)效率更高,其中协方差矩阵计算 GPU 比 CPU 快 4 ~ 6 倍,白化处理加速 6 ~ 8 倍,ICA 迭代中 GPU 比

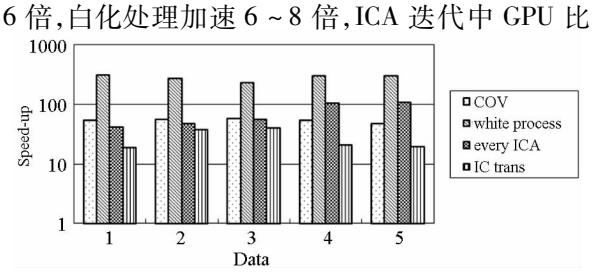

图 4　各热点 GPU 与串行加速比

Fig. 4　Speed-up between GPU and serial

CPU 快 8 ~ 11 倍,IC 变换由于计算量较小,GPU 和 CPU 性能差距不大。

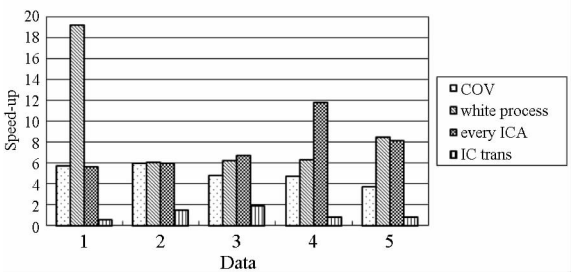

图 5 各热点 GPU 与 CPU 并行加速比

Fig. 5 Speed-up between GPU and CPU

图 6 统计了 G - FastICA 算法对最佳串行程序、MPI 并行程序总时间的加速比。图中数据显示,G - FastICA 算法比串行 FastICA 算法加速 36 ~ 72 倍不等;G - FastICA 算法比 MPI 并行算法加速 4.1 ~ 6.5 倍不等。这说明在高光谱影像降维 FastICA 处理时,CPU/GPU 异构系统能比 CPU 并行系统获得更好的性能。

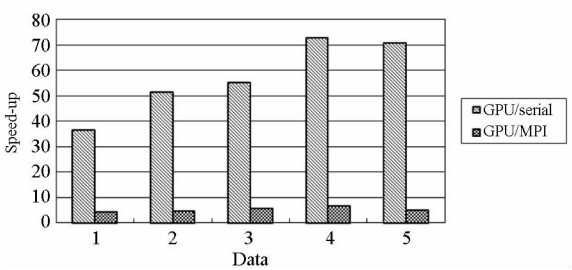

图 6 G - FastICA 与串行和 MPI 并行的加速比

Fig. 6 Speed-up between G-FastICA and serial or MPI

5 结论

本文研究了高光谱影像降维快速独立成分分析方法,提出和实现了基于 GPU 的 G - FastICA 并行算法,并基于 GPU 架构对算法进行优化研究。实验结果显示了 GPU 比 CPU 具有更好的计算优势,G - FastICA 并行算法最高可加速 72 倍。

通过在 CPU/GPU 异构系统上研究快速独立成分分析降维方法,获得了良好的性能,探讨了 GPU 加速高光谱影像降维的可行性,切实加速了高光谱影像降维过程。本文算法可进一步扩展至 GPU 集群来提高性能。

参考文献(References)

[1] Green R O, Eastwood M L, Sarture C M, et al. Imaging spectroscopy and the airborne visible/infrared imaging spectrometer(AVIRIS)[J]. Remote Sensing Environ, 1998, 65(3): 227 - 248.

[2] Green A A, Berman M, Switzer P, et al. A transformation for ordering multispectral data in terms of image quality with implications for noise removal[J]. IEEE Transaction on Geoscience and Remote Sensing, 2000, 26(1): 65 - 74.

[3] Kaarna A, Zemcik P, Kälviäinen H, et al. Compression of multispectral remote sensing images using clustering and spectral reduction[J]. IEEE Transaction on Science Remote Sensing, 2000, 38(2): 1073 - 1082.

[4] Scott D, Thompson J. Probability density estimation in higher dimensions[C]// Proceedings of the 15th Symposium on the Interface Computer Science and Statistics, 1983: 173 - 179.

[5] Jolliffe I T. Principal componnent analysis [M]. USA: Sprinner-Verlan, 1986.

[6] Hyvärinen A, Oja E. Independent component analysis: algorithm and applications [J]. Neural Networks, 2000, 13(4 - 5): 411 - 430.

[7] Tenenbaum J B, De Silva V, Langford J C. A global geometric framework for nonlinear dimensionality reduction[J]. Science, 2000, 290(5500): 2319 - 2323.

[8] Roweis S T, Saul L K. Nonlinear dimensionality reduction by locally linear embedding[J]. Science, 2000, 290(5500): 2323 - 2326.

[9] TOP500. TOP500 supercomputer sites[EB/OL]. (2012 - 11) [2014 - 09 - 26] http://www.top500.org.

[10] Valencia D, Lastovetsky A, O'Flynn M, et al. Parallel processing of remotely sensed hyperspectral images on heterogeneous networks of workstations using heteroMPI[J]. International Journal of High Performance Computing Applications, 2008, 22(4): 386 - 407.

[11] Plaza J, Plaza A, Pérez R, et al. Parallel classification of hyperspectral images using neural networks[J]. Computational Intelligence for Remote Sensing Studies in Computational Intelligence, 2008, 133: 193 - 216.

[12] Sánchez S, Ramalho R, Sousa L, et al. Real-time implementation of remotely sensed hyperspectral image unmixing on GPUs[J]. Journal of Real-time Image Processing, 2012, 32(1): 518 - 522.

[13] Yang H, Du Q, Chen G S. Unsupervised hyperspectral band selection using graphics processing units[J]. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2011, 4(3): 660 - 668.

[14] Ramalho R, Tomás P, Sousa L. Efficient independent component analysis on a GPU[C]//Proceeding of the 10th IEEE International Conference on Computer and Information Technology, 2010: 1128 - 1133.

[15] Forgette J, Wachowiak-Smolíková R, Wachowiak M. Implementing independent component analysis in general-purpose GPU architectures[C]//Proceedings of the International Conference on Digital Information Processing and Communications 2011: 233 - 243.

[16] Hyvrinen A. Fast and robust fixed-point algorithms for independent component analysis[J]. IEEE Transactions on Neural Networks, 1999, 10(3): 626 - 634.

[17] Hyvrinen A. The fixed-point algorithm and maximum likelihood estimation for independent component analysis[J]. Neural Process Lett, 1999, 10(1): 1 - 5.