一种输入感知的雷达回波快速聚类实现

周 伟[1,2] 安 虹[1] 刘 谷[1] 李小强[1] 吴石磊[1]
(中国科学技术大学计算机科学与技术学院 合肥 230027)[1] (陆军军官学院计算机教研室 合肥 230031)[2]

摘 要 聚类算法作为数据挖掘中的经典算法,在雷达回波的数据分析中经常被采用。然而对于规模和维度都较大的输入数据集,算法十分耗时。很多研究虽然对聚类算法进行了 GPU 平台的并行和优化的工作,但都忽略了输入数据集对优化的影响。因此,提出了在 GPU/CUDA 平台上的一种新颖的雷达快速聚类实现。该实现通过运行时的方式对输入的回波数据进行观察,以获取数据的分布信息,用以指导聚类计算在 GPU 上执行时的线程块调度。而该运行时模块本身的开销非常小。实验表明,引入这种输入感知的运行时调度支持后,大大削减了 GPU 的计算负载,获得了相对于一般策略的 CUDA 实现的 20%~40%的性能提升,加强了算法的实时性能。

关键词 聚类算法,实时性,输入感知,图形处理器,统一计算设备架构
中图法分类号 TP391 **文献标识码** A

Input-aware Runtime Scheduling Support for Fast Clustering of Radar Reflectivity Data on GPUs

ZHOU Wei[1,2] AN Hong[1] LIU Gu[1] LI Xiao-qiang[1] WU Shi-lei[1]
(School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)[1]
(Computer Teaching and Researching Section, PLA Army Officer Academy, Hefei 230031, China)[2]

Abstract As a classic algorithm in data mining, the clustering algorithm is often adopted in analysis of radar reflectivity data. However, it is time-consuming while facing dataset of large scale and high dimension. Recently, several studies have been conducted to make effort in parallelization or optimization of the clustering algorithm on GPUs. Although these studies have shown promising results, one important factor—program inputs—in the optimization is ignored in optimization. We took the program inputs in consider as a factor for optimization of the clustering algorithm on GPUs. By observing the distribution feature of the input radar reflectivity data, we found that the ability to adapt to inputs is important for our application to achieve the best performance on GPUs. The results shows that our approach can gain a 20%~40% performance increment, compared to previous parallel code on GPUs, which makes it satisfies the requirement of real-time application well.

Keywords Clustering algorithm, Real-time, Input aware, GPU, CUDA

1 引言

在气象分析中,经常需要对雷达回波数据进行处理。聚类方法往往作为对雷达数据处理的基本工具用以描绘风暴的结构[1]。而雷达回波的大量数据和快速的更新对数据处理的实时性提出了较高的要求。

Lakshmanan 等[2]对几种天气图像的分割算法进行了对比,并提出了一种采用 K-means 聚类的嵌套分割算法[3]。然而,K-means 算法的执行非常耗时,并随着聚类空间的规模及维度的增加而线性增长。在我们的气象应用中,对于大量回波数据,由于数据规模大、聚类空间的特征向量维度高,使得聚类的迭代过程的计算量非常大,耗时严重。这限制了其在桌面/笔记本平台上的应用。

近两三年,采用 CPU/GPU 的异构计算已经逐步发展为主流的高性能计算平台。2010 年 11 月发布的全球 top500 的超级计算机中,最快的 4 个中有 3 个是采用的 GPU 机群[8]。同时,GPU 的廉价和高性能也正将超级计算带给大众。目前,对桌面与笔记本平台中配置 GPU 已经再平常不过了。而 CUDA 编程模型的出现,更是催热了"桌面高性能"的广泛应用。正是基于平台的移动性和性能价格的考虑,我们采用 GPU/CUDA 作为应用的硬件计算平台。

聚类算法内在的并行性非常适合于 SIMD(Single Instruction Multiple Data)的形式进行表达。CUDA 编程模型所采用的 SIMT(Single Instruction Multiple Threads)本质就是一种更为灵活的 SIMD 的编程模型。目前采用 CUDA 对聚类方法进行加速的工作也得到了关注[5-7]。然而,所有这些

到稿日期:2012-02-20 返修日期:2012-06-20 本文受国家自然科学基金重点项目(60633040),国家自然科学基金项目(60970023),国家 973 计划项目(2011CB302501),国家 863 计划项目(2009AA01Z106),国家科技重大专项项目(2009ZX01036-001-002,2011ZX01028-001-002-3)资助。

周 伟(1982—),男,硕士生,讲师,主要研究方向为并行计算、流处理器体系结构,E-mail:greatzv@mail.ustc.edu.cn;安 虹(1963—),女,博士,教授,主要研究方向为新型微处理器体系结构、并行计算机系统结构、高性能计算应用。

优化方法都忽略了一个重要因素，即输入数据集对优化性能的影响。

我们观察到在雷达回波聚类的应用中，可以根据回波数据数值的区域分布特点，指导和调整 GPU 的线程块调度，从而减少 GPU 的计算负载。基于此，我们提出了一种输入感知的运行时调度支持机制，用以雷达回波快速聚类的实现。该实现能根据输入数据集的情况，自适应地调整计算任务，从而削减 GPU 的负载，缩短计算耗时。运行时模块本身带来的开销非常小。实验结果表明，上述方法相对原来的 GPU 并行代码，可以获得 20%～40%的性能提高，大大提高了算法的实时性能。

本文第 2 节介绍聚类的串行化算法和聚类算法在 GPU 平台上并行化的相关工作；第 3 节介绍这种输入感知的雷达回波快速聚类实现的具体内容；第 4 节给出实验与分析；最后进行了总结。

2 应用背景和相关工作

2.1 K-means 聚类算法

聚类算法作为数据挖掘的经典算法，被广泛地用于天气过程分析领域，Lakshmanan 提出了一种嵌套的天气图像分割方法[3]，其实现就采用了 K-means 算法对雷达反射率数据进行聚类。

应用中，我们将雷达回波数据映射到一个二维的笛卡尔平面，形成一个大型的二维数组。接下来计算出平面数组中每个点对应的纹理特征向量[4]，以此作为聚类的特征空间：$R=\{r_1,r_2,\cdots,r_n\}$。其中，n 为空间的元素个数，每个元素 r_i 的维度设为 d。K-means 算法将该特征空间 R 分割成 K 个类 $S=\{S_1,S_2,\cdots,S_k\}$。每个类中各个点对应的向量值具备最大的相似度。这里的相似度通过一个考虑了气象意义的价值函数[4]来确定，它可以用来表征临近区域的天气过程的相似性。为了讨论方便，对串行的算法简化表述如下：

算法 1 串行的 K-means 回波聚类算法

Step0 指定 K 值，将空间 R 依据反射率的最大值和最小值均等地划分成 K 个初始类 S_i，并计算出每个类的初始的中心点 C_i。

Step1 样本点分类：对聚类空间中的每个样本点 r_j，计算其与每个类的中心点 C_i 间的相似度，即根据气象意义所定义的价值函数。然后将该点重新分配给相似度最大的那个中心点所对应的类中。

Step2 更新聚类的中心点：重新统计每个类 S_i 的所属样本点，重新计算每个类 S_i 的中心点 C_i 的值；

Step3 判断指定的收敛准则是否满足，或每个类 S_i 是否不再变化。如果不满足，跳转到 Step1。如果满足，算法中止。

算法的主要耗时在于 Step1 与 Step2 的迭代过程，主要的计算量则体现在迭代的第一个步骤 Step1 样本点分类过程，它以计算相似度函数为基本操作，则算法复杂度为 $O(nk)$；迭代的第二个步骤 Step2 更新聚类中心点，其复杂度为 $O(n+k)$，而且只是以简单的加法操作为主，计算量并不大。另外，由于雷达回波聚类中，n 的值往往在 1M～10M，而 $k<50$，即 $n\gg k$，因此，每次迭代 $t=O(nk+n+k)\approx O(nk)$。所以，可以发现主要耗时在迭代的第一个阶段，即样本点分类过程。通常一般的 CUDA 并行化策略，就是将 Step1 在 GPU 上计算，而 Step2 交由 CPU 来完成。

2.2 基于 CUDA 的聚类算法的相关工作

目前，有很多研究对聚类算法在 GPU/CUDA 平台上的并行化做了大量优化工作[5-7]，其方法主要通过存储系统和通信同步等方面进行优化。如利用 GPU 丰富的存储体系、GPU 同步和通信等。

文献[5]提出了 CUDA 的并行化的基本方法，得到了较高的加速比和满意的性能，但其处理的输入数据集是规模很小的一维向量。

文献[7]在均值漂移算法中对 K-means 聚类进行了 CUDA 并行化，通过 GPU 单线程来完成聚类的第二个阶段更新聚类中心点的操作，这种方法减少了 CPU 与 GPU 之间的数据通信开销。

文献[6]在肝脏图像分割中使用了聚类算法，为了获取更高的性能，他们通过忽略线程间的同步来进行聚类中心点的更新，将误差控制在允许范围内，但这是以牺牲精度为代价的。

以上这些优化工作都忽略了一个重要因素可能带来的性能提升，即程序的输入数据集。本文通过观察应用中的输入数据集-雷达反射率，提出了一种输入感知的雷达回波快速聚类的实现。这种新颖的方法能够根据输入的雷达回波数值的区域分布特点，给出线程块在 GPU 上执行的指导信息，从而减少 GPU 的计算负载，进一步挖掘算法的实时性能。

3 一种输入感知的雷达回波快速聚类实现

本节先介绍程序的输入数据集(雷达回波)的数值分布特点。在 3.2 小节中，引入一种输入感知的运行时调度支持机制，并给出其设计和实现。3.3 小节中，讨论了该方法所带来的开销。

3.1 雷达回波聚类的输入数据集特点

这一节介绍雷达回波聚类的输入数据集的数值分布特点。首先来看一组常见的雷达回波情况，如图 1 所示。

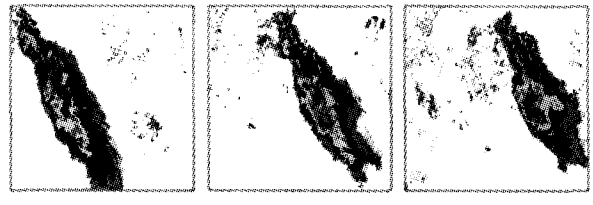
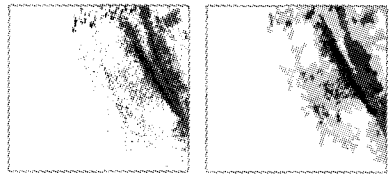


图 1 一个风暴在同一区域不同时刻的情况

图 1 所示是经过处理的一个风暴区域在不同时间点的回波数值情况。其中，不同颜色表示着不同大小值的反射率区域。白色区域是反射率低于一个很小的阈值(即接近于 0dBZ)的区域，其在气象分析中可以忽略，这里我们称之为无效区域。



(a) 聚类前的回波情况　　(b)聚类后的结果

图 2 聚类前后的回波情况

这里需要指出的是，反射率为 0dBZ 的点未必一定就是无效区域的点。图 2 是同一区域聚类前后的情况。可以看到，图 2(a)中云团边缘上的区域含有大量反射率为 0dBZ 的点和非零点混杂在一起。但聚类后，这些反射率为 0dBZ 的点和非零点一起形成了反射率值较低的区域，即图 2(b)中云团最外围一层的区域。

在大量的雷达回波实例中，我们可以观察到，风暴所占区域仅占整个观察区域的一部分，整个输入数据集中往往同时包含大量的无效区域。然而，我们不得不把整个观察区域作为聚类的输入数据，因为输入数据集中的风暴位置可能出现在观察区域的任意位置，所以无法预知输入数据的内容。这就使得大量的数值为 0dBZ 的无效区域参与了聚类过程的计算，造成了计算资源的浪费，这种计算在算法层面是无意义的。

因此，削减这些无意义的计算负载的直观想法决定能否使用一个运行时模块，动态地“感知”输入数据集的分布特点，并以此来指导聚类在 GPU 的计算过程。事实上，当 GPU 进行计算时，CPU 往往是空闲的，因此我们可以利用闲置的 CPU 来完成这个任务。

3.2 一种输入感知的雷达回波快速聚类实现

这一节给出一种输入感知的雷达回波快速聚类的实现方法。该方法利用空闲的 CPU 对输入的雷达回波数据的分布特征进行处理和“感知”，从而获取聚类空间数据的基本分布信息。接下来，GPU 进入执行阶段时，根据这些分布信息来调整 GPU 上的聚类计算过程，以削减 GPU 上不必要的计算负载。方法的框架如图 3 所示。

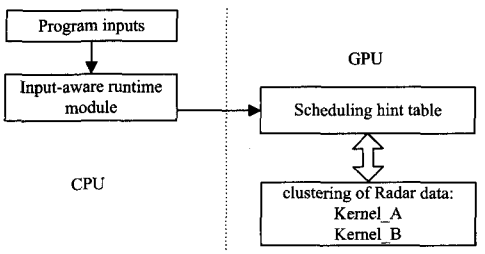


图 3 聚类前后的回波情况

我们所提出的方法包括一个输入感知的运行时模块、两个 GPU 上执行的核函数、一个存储输入数据分布特点的数据结构 scheduling hint table(命名为调度指导表)。

3.2.1 CUDA 并行化聚类程序的线程划分

对输入数据集(雷达回波数值矩阵)进行大小均等的块状划分，每个矩阵块的大小与 Block 线程块的大小一致，块状划分后的输入数据集的维度和尺寸就定义为 Grid 的维度和尺寸。这样，在处理器指派时，可以让每个划分的矩阵块与 Block 线程块对应起来。矩阵块中每个点对应的数值矩阵中的数据由 Block 线程块中相应线程进行计算处理。

这样，输入的雷达回波数值矩阵和 CUDA 编程模型的 Grid 二维网格之间建立了非常契合的直接映射关系。

3.2.2 输入感知模块

由于输入数据集是程序执行时才能获得的，因此，输入感知模块以运行时的方式执行。我们用 CPU 来执行该模块。

运行时模块首先对划分后的每个输入数据的矩阵块进行一次扫描。扫描过程即为简单的比较和累加，求出该小块区域有多少个非 0dBZ 的点；当该矩阵块中的反射率为非 0dBZ 的点的总数低于一个很小的阈值时，我们就认为矩阵块为无效区域，可以不予计算，并将其记录在 scheduling hint table 中，在 GPU 执行时供其查询，如图 4 所示。

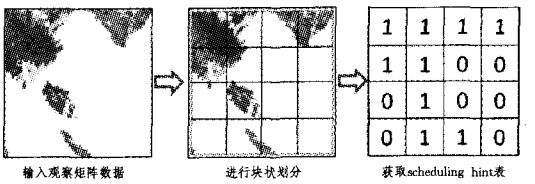


图 4 scheduling hint 表获取

scheduling hint table 为一个二维矩阵，其维度和 Grid 的维度要保持一致。表中每个值对应一个 block 块。1 代表该矩阵块中的点要进行聚类计算；0 代表该区域的反射率为非 0dBZ 的点小于一个很小的阈值，是一个无效区域。图 4 中 scheduling hint table 是一个 4 * 4 大小的简化的表达，实际的规模为 128 * 128。

完成扫描后，将 scheduling hint table 传入 GPU 的显存中，并作为参数传给核函数，我们可以将其放入常数存储器，通过缓存机制加速其访问速度。

3.2.3 聚类过程核函数设计

除了输入感知模块和调度指导表，还需要修改 GPU 的 kernel 函数，让 GPU 上执行的聚类过程能感知到无效区域，并对相应的线程块不进行样本点的分类计算。这就是我们将输入数据集的块状划分和 Grid 线程划分结构完全对应起来的好处。

下面给出一般策略的聚类的 CUDA 并行化程序的伪代码。

算法 2 一般策略的 CUDA 并行聚类算法

```
//R 表示聚类的输入数据集
//C 表示聚类中心点相关数据
Initial(R,C);
while(not stable){
  gpu_clustering<<<…>>>(R,C);
  …… ……
  updataCentriods(R,C);
  …… …… }
```

其中，gpu_clustering<<<…>>>(*R*,*C*)为样本点分类过程，它由 CUDA 并行化后在 GPU 上执行；updataCentriods(*R*,*C*)为更新聚类中心点过程，它通常在 CPU 上执行。

引入输入感知的运行时调度机制后，需要对聚类过程的核函数 gpu_clustering 进行修改。修改后的核函数在执行聚类操作之前，应该首先通过内置变量 BlockID 查询调度指导表，以确定所计算的点是否是无效区域内的点，如果是，则放弃样本点分类的计算。下面给出修改后的核函数 gpu_clustering_input_aware 的伪代码，见代码 1。

代码 1 引入调度指导机制的核函数

```
//R 表示聚类的输入数据集
//C 表示聚类中心点相关数据
```

```
//H 表示 scheduling hint table 的内容
gpu_clustering_input_aware<<<…>>>(H,R,C)
{ int flag;
  flag=querySchedulinghint(BlockId);
  if( flag==0)
    gpu_clustering<<<……>>>(R,C);
  else
    ;//空操作
}
```

代码 1 中，核函数内调用的核函数 gpu_clustering 在 G200 架构中，将其编译为内联方式的核函数。在 fermi 架构中，则是以函数指针的方式调用。

我们将算法 2 中的 gpu_clustering 函数替换成含有上述功能的 gpu_clustering_input_aware 函数，就得到了引入输入感知的运行时调度机制算法，我们确定其为算法 3。

在 GPU 的执行的过程中，block 线程块执行的实体是 SM(Stream Multiprocessor 流多处理器)，调度则是以 block 划分后的 warp 线程束为单位。在 gpu_clustering_input_aware 核函数执行时，通过联合访存的优化，访问 scheduling hint table 的同一个 warp 内的一组线程只需要一个访存事务就能获得 flag 值。另外，在这种机制下，显然代码 1 中的同一组 warp 的线程会走向 if 语句的同一个分支。因此，相对于原来的 kernel，新的 kernel 给 GPU 所带来的开销并不大。

通过这种调度指导方式，图 4scheduling hint table 中 0 所对应的回波数值的矩阵块的 GPU 计算负载被动态地削减掉了。

3.2.4 输入感知的快速雷达回波聚类实现总体流程

为了极端情况下不损失算法 1 原始的效率，CPU 在调用 GPU 进行聚类前，先统计 scheduling hint table 中 0 的总数。如果 0 值的总数小于一个阈值，即说明获得的性能好处非常少，甚至有可能被 GPU 所带来的开销抵消掉。那么这种情况，就调用原始的一般策略的 CUDA 聚类算法 2 进行计算。这里的阈值由 GPU 和 CPU 的运算能力决定，是个实验的值，我们设为 m。图 5 给出了总体的流程图。

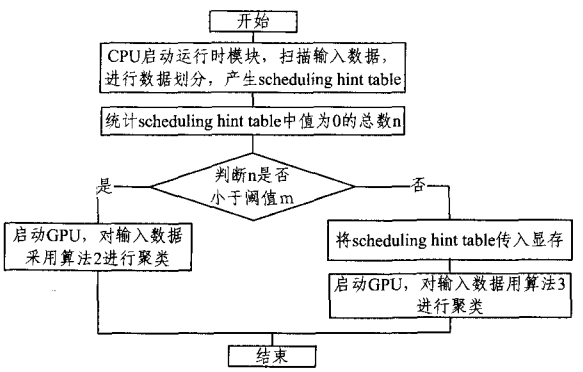


图 5 总体流程图

3.3 引入输入感知的调度机制所带来的开销

引入这种输入感知机制所带来的开销有两部分，一是模块在 GPU 启动前，CPU 上执行的运行时模块所耗费的时间；二是将 scheduling hint table 拷贝到 GPU 显存和算法 3 中 GPU 访问该表并判断是否需要执行样本点分类计算所耗费的时间。

前者的耗时，在 1M～10M 规模的矩阵扫描并获得 scheduling hint table 的时间是毫秒级，相对于几秒到几十秒的聚类计算过程，可以忽略不计；后者的耗时，由 3.2.3 中的分析可以得知，其所带来的开销也远远小于聚类计算的耗时。第 4 节的实验分析也可以证实这一点。

4 实验与分析

实验中，主机端使用 CPU 为 Intel Core 2 Quad Q9400，GPU 芯片为 Geforce GTS250。GTS250 是 GT92 架构，含有 16 个流多处理器(stream Multiprocessor)，每个流多处理器含有 8 个流处理器(Stream processor)。每个流多处理器的主频为 1.73GHz，显存带宽在峰值为 70GB/sec。根据应用的需求，聚类对象的规模为 1M 个点的规模，维度为 10，而 k 值小于 50。

(1)耗时分析

输入的回波数值为某地区两天内不同时刻所产生的数据：CREF_20100718_110，CREF_20100718_120，CREF_20100718_130，CREF_20100718_140，CREF_20090513_210。分别用 I1，I2，I3，I4，I5 表示。

从表 1 中可以看到，串行聚类过程的主要耗时是算法 1 的 Step1 样本点分类计算。采用一般策略的 CUDA 并行化后，算法 1 的 Step1 在 GPU 上执行(即 gpu_clustering 函数)，耗时大大减少，从而整个算法的耗时也大大减少。但并行化后，样本点分类计算(gpu_clusteirng 函数)仍然是主要的耗时过程。

表 1 串行聚类及一般策略的 CUDA 并行聚类耗时

输入实例	串行算法 1 耗时(秒)	gpu_clustering (秒)	算法 2 耗时 (秒)
I1	124.331	4.421	6.481
I2	122.912	4.389	6.712
I3	122.931	4.132	6.423
I4	124.281	4.212	6.322
I5	123.566	4.476	6.783

我们的输入感知模块优化的就是在 GPU 上执行的样本点分类计算过程，即 gpu_clusteirng 核函数。

(2)性能提升

引入了输入感知的运行时模块后，即将算法 2 中的核函数 gpu_clustering 由核函数 gpu_clustering_input_aware 代替。根据输入数据集情况的不同，可以获得不同的性能提升。我们用 scheduling hint 表中 0 值的比例来表达应用中的输入数据集中无效区域的比例。性能提升如表 2 所列。

表 2 引入输入感知机制所获取的性能分析

输入实例	gpu_clustering (秒)	gpu_clustering_ input_aware(秒)	GPU 执行部分性能提升	hint 表中 0 值比例
I1	4.421	2.782	37.07%	41%
I2	4.389	2.921	33.45%	38%
I3	4.132	2.841	31.24%	36%
I4	4.212	2.696	35.99%	37%
I5	4.476	1.517	66.11%	72%

从表 2 中可以看到，对算法 2 引入输入感知机制后，性能根据输入内容的不同，获得了 30%～60%的性能提升。由于

聚类算法中还包含了更新中心点等操作，而我们的输入感知模块仅仅作用于样本点分类过程，因此从全局耗时上看，性能优化为 20%～46%。

当聚类的聚类中心点发生变化时，传统的 GPU 聚类方法的加速比会受到影响，但这种输入感知的调度支持机制由于是在输入数据全局进行 GPU 负载的削减，因此仍然能有效提高其性能。图 6 是在 3 款 GPU 芯片下测得的传统方法的 GPU 聚类的加速比，图 7 是引入这种输入感知的调度支持机制后的加速比情况。

需要指出的是，为了实时跟踪气象过程（如风暴等）的运动，我们所输入的数据总是将其缩放到一个合适的尺度，以获得完整的风暴形状，因此无效区域往往是大面积存在的。

另外，由于 gpu_clustering_input_aware 本身也具有一定的开销，并且体现在每个 warp 上，因此从表 2 中可以看到，性能的提升总是要小于 hint 表中 0 值的比例，但所获得的性能仍然远远高于开销。

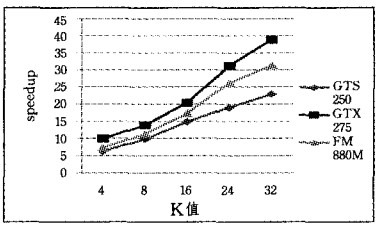


图 6　传统的 GPU 聚类在不同 K 值时的加速比

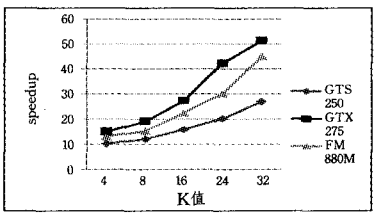


图 7　引入输入感知机制后的 GPU 聚类的加速比

结束语　本文提出一种输入感知的雷达回波快速聚类方法实现，给出了这种机制所需要的运行时模块和 GPU 代码中所需要的设计。该方案能够根据输入数据集的分布信息，削减 GPU 的计算负载。相对于一般策略的 CUDA 并行化方法，其在 GPU 上获得了 30%～60%的性能提升，带来了全局算法 20%～40%的性能提升，进一步挖掘了算法的实时性能。

下一步的工作，我们将继续寻找输入数据集对 GPU 并行化算法的影响应用场合，找出通过输入数据集优化并行程序的共性方法，以提高相关算法的实时性。

参 考 文 献

[1] Yang H P, Zhang J, Langston C. Synchronization of Radar Observations with Multi-Scale Storm Tracking[J]. Advances in atmospheric sciences, 2009, 26

[2] Lakshmanan V, Rabin R, DeBrunner V. Multiscale storm identification and forecast[J]. Atmospheric Research, 2003, 67/68: 367-380

[3] Lakshmanan V, Rabin R, DeBrunner V. Segmenting Radar Reflectivity Data using Texture[C]//Proc of the 30th International Conference on Radar Meteorology. Zurich, Switzerland: [s. n.], 2001

[4] Wang G L. The Development on a Multiscale Identifying Algorithm for Heavy Rainfall and Methods of Extracting Evolvement Information[D]. Nanjing University of Information Science and Technology, 2007

[5] Farivar R, Rebolledo D, Chan E, et al. A parallel implementation of K-means clustering on GPUs[C]//Proc of the 2008 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2008). 2008: 340-345

[6] Walters J P, Balu V, Kompalli S, et al. Evaluating the use of GPUs in liver image segmentation and HMMER database searches [C]//Proc of the 2009 IEEE International Parallel and Distributed Processing Symposium (IPDPS2009). IEEE Computer Society, 2009

[7] 陈加，吴晓军，蔡荣. GPU 并行加速的均值偏移算法[J]. 计算机辅助设计与图形学学报，2010(3)

[8] http://www.top500.org/

(上接第 263 页)

破解难度，适用于数据量大、存在大量连续同色区域的医学图像加密。

参 考 文 献

[1] 徐杰，杨娣洁，隆克平. 基于时滞混沌系统的带密钥 Hash 函数的设计与分析[J]. 电子科技大学学报，2011，40(3)：451-455

[2] 郑皓洲，胡进峰，徐威，等. 一类新型超混沌系统的非线性反馈同步研究[J]. 电子与信息学报，2011，33(4)：844-848

[3] Zhu Zhi-liang, Zhang Wei, Wong K-W, et al. A chaos-based symmetric image encryption scheme using a bit-level permutation [J]. Information Sciences, 2011, 181: 1171-1186

[4] 王重英. 基于 Logistic 的混沌加/解密图像算法研究[J]. 计算机应用技术，2009，18：123-127

[5] Zhang Jun, Li Jin-ping, Wang Lu-qian. A New Compound Chaos Encryption Algorithm for Digital Images[C]//2010 International Forum on Information Technology and Applications (IFITA 2010). Kunming, China: IEEE Computer Society, 2010: 277-279

[6] Wang Yong, Wong K-W, Liao Xiao-feng, et al. A new chaos-based fast image encryption algorithm[J]. Applied Soft Computing, 2011, 11: 514-522

[7] 黎全，赵凯，邓正才，等. 对一种混沌加密图像方法的破译研究[J]. 国防科技大学学报，2007，29(3)：45-49

[8] Xu Yang. Design of streams encryption key generator based on Chaos Theory[C]//2009 Pacific-Asia Conference on Knowledge Engineering and Software Engineering (KESE 2009). Shenzhen, China, IEEE Computer Society, 2009: 213-216

[9] 朱从旭，黄大足，郭迎. 结合多混沌映射和输出反馈的图像加密算法[J]. 武汉大学学报：信息科学版，2010，35(5)：528-531