

基于 GPU-CPU 流水线的雷达回波快速聚类

周 伟^{1,2}, 施 宁², 王 健², 汪群山²

(1 中国科学技术大学 计算机科学与技术学院, 安徽 合肥 230027;

2 中国人民解放军陆军军官学院 计算机教研室, 安徽 合肥 230031)

摘 要: 提出了基于 GPU-CPU 流水线的雷达回波快速聚类方法. 该方法利用 GPU 与 CPU 异步执行的特征, 将聚类的各步骤组织成流水线, 大大的挖掘了聚类全过程的并行性. 实验表明, 引入这种 GPU-CPU 流水线机制后, 该方法比一般策略的基于 GPU 的并行聚类算法性能有 38% 的提升, 而相对于传统的 CPU 上的串行程序, 获得了 47x 的加速比, 满足了气象实时分析应用中的实时性要求.

关键词: GPU; CUDA; 聚类算法; 流水线

中图分类号: TP391

文献标识码: A

文章编号: 1000-7180(2012)04-0071-05

Fast Clustering of Radar Reflectivity Data Using GPU-CPU Pipeline Scheme

ZHOU Wei^{1,2}, SHI Ning², WANG Jian², WANG Qun-shan²

(1 School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China;

2 Computer Teaching and Research Room, PLA Army Officer Academy, Hefei 230031, China)

Abstract: In our meteorological application, the clustering algorithm was adopted for analysis and processing of radar reflectivity data. While facing problems of large scale of dataset and high dimension of feature vector, the clustering algorithm is too time-consuming to satisfy the real-time constraint in our applications. This paper proposes a parallelized clustering algorithm using GPU-CPU pipeline scheme to solve this problem. In our method, we utilized the feature of asynchronous execution between GPU and CPU, and organized the process of clustering into pipeline-style, with which we can largely exploit the parallelism in algorithm. The experimental results show that our GPU-CPU pipeline based parallelized clustering algorithm outperform normally parallelized clustering algorithm using CUDA without GPU-CPU pipeline by 38%. Compared to the serial code on CPU, our approach can achieve a 47x performance improvement, which makes it satisfy the requirements of real-time applications.

Key words: GPU; CUDA; image segmentation; clustering algorithm

1 引言

在气象应用中, 我们采用聚类的方法对雷达回波数据进行基本的分析和处理, 从而研究降雨、风暴等天气过程特点^[1-2].

在应用中, 对于大量的回波数据(单站雷达每隔 5~6 分钟就产生一批体扫数据, 并要求实时处理)进行聚类, 受限于数据规模大、聚类空间的特征向量维度高, 使得聚类在每次迭代过程中的计算量非常

大, 耗时严重. 这大大限制了其在具有一定实时性要求的领域中的应用. 目前, 对于单帧规模在 1024×1024 点的数据, 在主流的 CPU 平台上进行聚类, 耗时达到了 140 s 左右, 远远无法满足业务软件在几秒内响应的实时性要求.

近些年来, GPU 在通用目的领域的并行计算能力不断提高. 以往需要昂贵的计算集群才能获得理想加速效果的并行代码, 现在由一个 GPU 就可以完成. 它相当于几十甚至上百个 CPU 核的计算能

力. 在 GPU 上能获得较高加速的应用特征是: (1) 计算密集型的程序, 即具有较高的计算访存比; (2) 数据级并行的程序; (3) 算法本身能够较好的调整负载均衡.

由于聚类方法的固有特点, 采用 GPU 并行化往往可以获得较好的加速, 能大大降低算法的运行时间. 目前这方面的工作也得到了关注^[3-6]. 然而, 对聚类过程的 CUDA 并行化, 多数只是仅仅并行化了聚类的第一阶段的样本分类步骤, 而将更新聚类中心点放到 CPU 上, 使两个阶段串行的执行^[3-4, 6]; 个别工作绕开了第二个阶段的中心点更新的同步问题, 但是以牺牲精度为代价^[5].

本文提出了基于 GPU-CPU 流水线的回波快速聚类方法.

2 背景和相关工作

2.1 K-means 聚类的串行算法

在 K-means 聚类应用中, 我们从回波数据中计算出每个点对应的纹理特征向量, 并构成了聚类的特征空间: $R = \{r_1, r_2, \dots, r_n\}$. 其中, 特征空间的维度为 d ; K-means 将聚类空间 R 分割成 k 个类 $S = \{S_1, S_2, \dots, S_k\}$. 每个类中的各个点间具备最大的相似度, 这里的相似度函数通过一个气象意义所定义的一个价值函数^[1], 表征特定天气过程的相似性. 串行算法如下:

算法 1 串行的 k -means 聚类算法

Step0: 确定 k 值, 将 R 初始划分为 k 个类 S_i , 并计算各个类的初始中心点 C_i .

Step1: 样本点分类: 依次计算聚类空间中每个样本点 r_j 与各个类的中心点 C_i 的相似度, 并根据最大的相似度对相应点 r_j 进行重新划分所属的类 S_i .

Step2: 更新聚类中心点 C_i : 计算新生成的各个类 S_i 中所有样本点的均值, 并作为新的中心点.

Step3: 判断指定的熟练准则是否得到满足, 或每个聚类是否不再变化. 如不满足, 则重复前面两步 Step1 和 Step2, 直到满足为止.

可以看出, 算法主要耗时在迭代的过程. 每次迭代的第一个阶段 Step1. 样本点分类的复杂度是 $O(nk)$, 第二个步骤 Step2. 更新聚类中心点的复杂度是 $O(n+k)$. 在本应用中, $n \approx 1M, k < 50$, 即 $n \gg k$, 因此, 该算法的每次迭代的时间是 $t = O(nk + n + k) \approx O(nk)$. 可以看出, 算法在并行化之前, 主要耗在于第一个阶段.

2.2 CUDA 加速聚类过程的相关工作

聚类方法在 CUDA 平台上的并行化已被广泛关注, 文献[3-6] 都涉及到了聚类问题的 CUDA 加速.

目前, 大多数 CUDA 并行化聚类的方法, 都是仅仅并行化聚类迭代的第一个阶段^[3-4, 6]. 这使得采用 CUDA 并行化后的代码, 第一个阶段与第二个阶段之间仍然是串行执行的. 为了获取更高的加速比, 以满足应用中的实时性要求, 我们提出了基于 GPU-CPU 流水线机制的快速回波聚类方法, 使得聚类迭代的两个阶段的计算能够以流水线的方式并行的执行, 极大的挖掘了算法的并行性.

3 基于 GPU-CPU 流水线的回波快速聚类方法

3.1 基于 CUDA 的 K-means 聚类

对于采用 CUDA 并行化聚类的过程, 其迭代的第一阶段由于具备典型的数据级并行的特点, 适合于映射到 SIMT 模型在 GPU 上执行; 而第二个阶段的聚类中心点更新, 由于需要所有线程访问一个求和的值, 而计算量很小, 属于访存和通信受限的计算过程, 而 GPU 的 Block 间缺少全局的同步特性, 因此该阶段的计算更适合于拥有较大可读写 cache 的 CPU 上执行.

通常, 在 K-means 聚类的 K 值较小时, 我们将聚类空间中的每个样本点对应到一个线程, 负责该点的分类, 即执行算法 1 的 Step1. 由于 GPU/CUDA 编程模型缺少 Block 块间有效的全局大同步, 因此每次迭代间的同步, 是通过在循环体内多次调用 GPU 的 kernel 函数实现的. 而对于算法 1 的 Step2 步骤, 我们将数据传回主机端, 将其在 CPU 上计算.

整个算法的伪代码如算法 2 所示, 它包括一个循环, 迭代的进行样本分类计算 `gpu_clustering()` 和聚类中心点更新计算 `updateCentroids()`, 直到满足终止条件而稳定时, 算法执行结束.

算法 2 基于 GPU 的并行聚类算法伪码

```
//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);
    ...
    updateCentroids(R,C);
    ...
}
```

算法在 GPU 与 CPU 上的执行过程如图 1 所示, $S1$ 表示聚类第一个阶段的样本划分操作 `gpu_clustering()`, $S2$ 表示聚类第二个阶段的中心点更新操作 `updateCentroids()`, R 表示输入的数据集, $S1^i(R)$ 与 $S2^i(R)$ 的上标 i 表示循环的第 i 次迭代. 图中的箭头表示各个执行阶段间的数据依赖关系. 执行时必须满足箭头标示的顺序一致性.

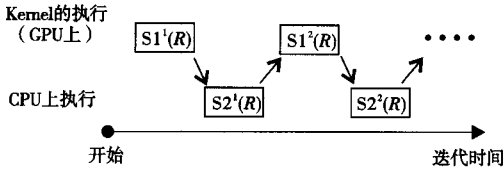


图1 基于 GPU 的并行聚类算法执行过程

在这种一般策略的 CUDA 并行化方法中, 聚类迭代的第一个阶段, 由 GPU 以数据级并行的方式执行, 而第二个阶段的更新聚类中心点由 CPU 完成. 由于第二个阶段的更新聚类中心依赖于第一个阶段的结果, 因此两个阶段只能串行的执行的, 如图 1, 这时 GPU 的计算资源在 CPU 执行时处于闲置状态, 大大降低了 GPU 的计算通量.

对于 k -means 串行算法, 其时间为 $t(n) = O(nkt + (n+k)t)$, 其中, t 为迭代次数, 在本应用中, $n \gg t; n \gg k$; 由此可见, 第一个阶段的耗时远远大于第二个阶段. 采用 CUDA 并行化后, 则其时间为 $t(n) = O(nkt/p + (n+k)t)$, 其中 p 为计算单元的数量. 在并行化后, 第二个阶段的计算时间 $O((n+k)t)$ 成为进一步加速的新的瓶颈, 在 4.1 节中的耗时分析部分, 我们用实验来说明这一点.

由此, 我们引入 GPU-CPU 流水线机制, 通过将 Step2 和 Step3 组织成流水线, 通过流水线将更新聚类中心的计算时间进行隐藏. 理想情况下, 可以将整个过程的执行时间从 $t = O(nkt/p + (n+k)t)$ 降低到近似于 $t'(n) = O(nkt/p)$.

3.2 基于 GPU-CPU 流水线机制的回波快速聚类

基于 GPU-CPU 流水线机制的回波快速聚类方法, 将聚类算法划分为几个阶段并分别映射到 GPU 与 CPU 上, 试图通过将 GPU 核函数部分的计算过程与在 CPU 线程部分的计算过程重叠起来, 以隐藏算法一个阶段的执行时间.

当聚类算法执行过程变换成 GPU-CPU 流水线执行时, 首先要满足和串行的算法 1 一样的顺序一致性.

在应用中, 由于存在一个接一个的输入数据集

要通过聚类程序逐一处理, 我们通过交叉的处理不相关的输入数据集来解决数据依赖问题. 如图 2 所示, 细箭头表示数据集 A 处理过程中的依赖关系, 宽箭头表示数据集 B 处理过程中的依赖关系. 我们通过在流水线中放入不同数据集 A 与 B, 使得提高数据通量的同时, 仍然保证了原来的数据间的依赖关系.

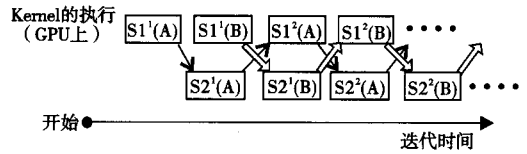


图2 基于 GPU-CPU 流水线的聚类算法执行过程

算法 3 给出了实现这种机制下的 GPU 端源程序, 它包括一个循环, 其中每次迭代调用 GPU 核函数 `gpu_clustering()` 来对两个数据集逐一处理, 这种调用是异步的, CPU 将继续执行 `threadMake()`, 来创建后续的线程. 其中, 变量 `st1` 和 `st2` 分别用来保证对数据集 A 和 B 各步骤间依赖关系的正确性. (当 `gpu_clustering()` 执行完毕时, 要对 `st1` 或 `st2` 置 2, 当 `updateCentroids()` 执行完毕时, 要对 `st` 或 `st2` 置 1) 这样, 就形成了图 2 的流水线执行过程, 使得整个算法的各个部分的计算时间可以相互掩盖.

关于流水线执行机制和顺序一致性保证我们总结如下:

(1) 对于各自数据集 (A 或 B) 每次迭代间的 GPU 核函数执行的顺序一致性, 通过创建各自的流 `Stream_A` 与 `Stream_B` 来保证.

(2) 设备端核函数的调用是异步的, 而主机端的 `threadMake()` 函数通过系统 API 实现, 用来创建函数线程, 也是异步形式调用, 以保证各阶段计算能流水执行.

(3) 同一个数据集得不同阶段之间的顺序一致性, 通过变量 `st1` 或 `st2` 保证;

由于应用中, 两个阶段的耗时都在秒一级, 而 CPU 内存到 GPU 显存的数据传输, 按照 PCI-E 的传输速度 5 GB/s, 传输数据规模在 10M 规模, 耗时在微秒一级, 远远小于算法两个阶段的执行时间, 可以忽略不计, 在算法的伪代码中略去了相关代码. 实时上, 如果算法执行时间变短, 和传输在同一数量级时, 我们也将它视作是流水线的一个节拍, 通过 GPU-CPU 流水线机制将其隐藏起来.

算法 3 基于 GPU-CPU 流水线的并行聚类算法伪码

```

// A,B 表示两次聚类的两个输入数据集;
// CA,CB 表示聚类中心点相关数据;
// Stream_A,Stream_B 是数据流;
// s1,s2 用来控制对 A,B 处理的顺序一致性;
Initial(A,CA);
Initial(B,CB);
Boolean st1=st2=1;
while(not stable){
    if(st1==1){
        gpu_clustering<<<...Stream_A>>>
        (A,CA,&st1);
    }
    if(st2==1){
        gpu_clustering<<<...Stream_B>>>
        (B,CB,&st2);
        ...
    }
    if(st1==2)
        threadMake(updataCentriods(A,CA,&st1));
    if(st2==2)
        threadMake(updataCentriods(B,CB,&st2));
    ...
}

```

3.3 关于 GPU-CPU 流水线的控制

3.2 节中有一个假设,即 CPU 的计算能力足够,使得聚类的样本分类在 GPU 上的执行时间与 CPU 上聚类中心点更新的执行时间大致是相当的。

当聚类的两个阶段的计算时间差别很大时,就需要考虑流水线深度的问题。如图 3 所示,在 GPU-CPU 流水线机制中值 Δ ,决定了 CPU 上执行聚类中心更新到下一次迭代的样本分类核函数可以使用该聚类中心值的时间距离,我们称之为流水线深度。流水线深度的控制,可以参考聚类迭代的样本分类的核函数执行时间与聚类中心点更新的执行时间的比值取整。如图 3 所示,假设聚类中心点更新在 CPU 上的执行时间是样本分类核函数 `gpu_clustering()` 执行时间的近 2 倍,那么流水线深度应该为 2。

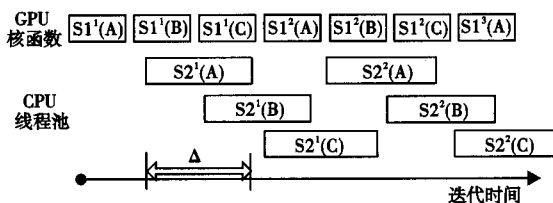


图 3 基于 GPU-CPU 流水线的聚类算法执行过程 2

流水线深度取决于两个阶段的计算时间,因此在硬件平台上,其取决于 GPU 的计算性能和 CPU 的计算性能,这是平台相关的。因此,流水深度即取决于聚类算法两个过程的计算量,也取决于具体的 GPU 与 CPU 硬件平台的计算能力。

本应用中,另外一个制约流水线深度的因素是显存大小。对于数据集规模较大的计算,流水线需要将几个不同的输入集都同时放入显存。本应用,一组输入数据(包括雷达回波值、特征向量值)的大小在几十兆字节(Size<100 MB)。因此对于 512 MB 显存来说,流水线深度最好在 5 以内。

4 实验与分析

实验平台采用 Intel Core 2 Quad Q9400 四核处理器和 Geforce GTX275 的图形处理器。GTX275 含 30 个流多处理器,每个流多处理器的主频在 1.40 GHz,显存的峰值带宽在 127 GB/s。实验使用的 CUDA 版本 2.3。实验输入的聚类对象为雷达回波数值点和抽取的特征向量。根据应用的需求,聚类对象的规模在百万个点的规模($1M * n$),而 k 值小于 50;

(1) 耗时分析

输入的回波数值为某地区一天内不同时间所产生的数据: CREF_20100920_010, CREF_20100920_010, CREF_20100920_030。

表 1 所示为串行的聚类程序,在 CPU 可以看出,并行化前,第一个阶段是耗时的热点部分。而在一般策略的 CUDA 并行化后,第二阶段则成为一个耗时热点部分,如表 2 所示。

表 1 串行聚类算法两个阶段的时间比例

输入实例	第一阶段/s	第二阶段/s	第二阶段比例/%
CREF_20100920_010	122.612	2.031	1.63
CREF_20100920_020	124.716	1.827	1.44
CREF_20100920_030	123.458	1.993	1.59

表 2 一般策略的 CUDA 并行化后的情况

输入实例	第一阶段/s	第二阶段/s	第二阶段比例/%
CREF_20100920_010	2.303	2.031	46.86
CREF_20100920_020	2.933	1.827	38.38
CREF_20100920_030	2.004	1.993	49.86

(2)加速比比较

图4中给出了基于GPU-CPU流水线的回波快速聚类的加速比.由于近乎隐藏了第二阶段的计算过程,因此相对于一般策略的CUDA并行化方法,在流水线理想情况下,能获得38%~50%的性能提升.而相对于串行的算法,可以获得高达47~49x的加速比.图中可以看到,加速比随着输入规模的扩大,基本保持不变.

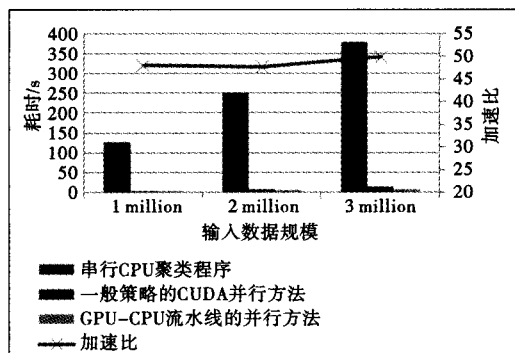


图4 基于GPU-CPU流水线的回波快速聚类加速比

5 结束语

本文提出了基于GPU-CPU流水线的雷达回波快速聚类方法,给出了一种通过流水线隐藏算法部分计算过程的方法.该方案相对于普通的CUDA并行方法提高了40%,相对于串行算法则获得了37倍的加速比,满足了应用中实时处理的需求.

下一步的工作,我们将深入研究这种GPU-CPU流水线机制的特点,找出其所适用的应用场景,以提高其他算法并行化的效率.

参考文献:

- [1] Lakshmanan V, Rabin R, DeBrunner V. Multiscale storm identification and forecast[J]. Atmospheric Research, 2003(7): 67-68.
- [2] Lakshmanan V, Rabin R, DeBrunner V. Segmenting radar reflectivity data using texture[C]//Proc. of the 30th International Conference on Radar Meteorology. Zurich, Switzerland, 2001.
- [3] 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). USA: Las Vegas, 2008: 340-345.
- [4] Miao Q, Fu Z L, Zhao X H. A new approach for color character extraction based on parallel clustering[C]//Proc. of 2009 International Conference on Computational Intelligence and Software Engineering. China: Wuhan, IEEE Computer Society, 2009.
- [5] Walters J P, Balu V, Kompalli S, et al. Evaluating the use of GPUs in liver image segmentation and HM-MER database searches[C]//Proc. of the 2009 IEEE International Parallel and Distributed Processing Symposium(IPDPS2009). Germany, Weimer, IEEE Computer Society, 2009.
- [6] 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).

作者简介:

周伟男, (1982—), 硕士研究生. 研究方向为流处理器体系结构、并行计算.

施宁 讲师. 研究方向为图像处理与模式识别.

(上接第70页)

- [5] High pin count rapid relay-based ESD & latch-up test system for evaluating advanced IC devices[R]. USA One Lowell Research Center. Lowell, MA 01852.
- [6] Sudhir Wokhlu, Erika Beskar, Raul Rousselin, et al. Low cost automatic mixed-signal board test using[C]//International Test Conference. USA California, 2007.
- [7] 雷鑑铭, 邹雪城, 邹望辉, 等. 应用于LCD控制器的动态抖动算法及帧速率控制技术[J]. 微电子学与计算机, 2004, 21(5): 22-24.

- [8] 汪翼, 沈海斌, 樊俊锋. 优化帧频控制的LCD控制器IP的设计与验证[J]. 计算机工程, 2006, 32(16): 235-236.

作者简介:

张乐男, (1986—), 硕士研究生. 研究方向为新型存储器设计.