

# 基于图形处理器的划分聚类算法效率研究<sup>\*</sup>

李琳<sup>1</sup>, 李肯立<sup>2</sup>

(1. 衡阳师范学院 计算机科学技术系, 湖南 衡阳 421008; 2. 湖南大学 计算机与通信学院, 长沙 410082)

**摘要:** 提出一种利用 GPU(图形处理器)和 CPU 的协同计算模式来提高划分聚类算法 enhanced\_K-means 的计算效率。利用 GPU 多个子素处理器可以并行计算的特性,将算法中比较耗时的欧氏距离计算与比较、中心点改变后簇中没有发生变化的点集合判断步骤由 GPU 执行,算法其余步骤由 CPU 执行,使聚类效率得到显著提高。在配有 Pentium 4 3.4 GHz CPU 和 NVIDIA GeForce7800GT 显卡的硬件环境下经过实验测试,证明其运算速度比完全采用 CPU 计算速度要快。这种改进的划分聚类算法适合在数据流环境下对大量数据进行实时高效聚类操作。

**关键词:** 聚类分析; 图形处理器; 通用计算; 划分聚类

中图分类号: TP301.6 文献标志码: A 文章编号: 1001-3695(2009)04-1276-03

## Research of efficiency of partitioning clustering algorithm based on graphics processing unit

LI Lin<sup>1</sup>, LI Ken-li<sup>2</sup>

(1. Dept. of Computer & Science & Technology, Hengyang Normal University, Hengyang Hunan 421008, China; 2. College of Computer & Communication, Hunan University, Changsha 410082, China)

**Abstract:** This paper proposed a mode with CPU + GPU co-processing to improve the efficiency of enhanced\_K-means algorithm. By the characterization that the parallel computing could be finished by the multiple fragment processor, the step that the calculation and comparison of Euclidean distance, the judgment on the point aggregation in the clustering that has no difference after the central point was changed, both of which would spent much time, were finished by GPU, while other steps were finished by CPU. Therefore the clustering efficiency was improved greatly. Some experiments conducted in a PC with Pentium 4 3.4GHz AMD 643500 + CPU and NVIDIA GeForce7800GT graphic card demonstrate that the presented algorithm is faster than the previous CPU-based algorithms, thus the improved partitioning clustering algorithm is applicable for the clustering data stream that requiring for high speed processing and high quality clustering results.

**Key words:** clustering analysis; graphics processing units(GPU); general purpose computation; partitioning clustering algorithm

## 0 引言

随着传感器和计算机网络技术的发展,在数据流环境下研究数据聚类方法,除了需要考虑聚类质量以外,聚类处理速度往往也是数据流挖掘算法的一个重要指标。由于 GPU 为图形渲染设计<sup>[1]</sup>,超长流水线及并行计算和可编程性的出现,使其运算速度越来越快。鉴于其在一些非图形绘制方面通用计算<sup>[2]</sup>的广泛应用:如代数计算、流体模拟、数据库操作、频谱变换和滤波等,本文提出了一种使用 GPU(图形处理器)和 CPU 协同处理聚类算法的模式,以使聚类算法效率显著提高。本文以 enhanced\_K-means 算法为例,利用 GPU 多个子素处理器可以进行并行计算的特性,将算法中处理比较耗时的距离计算与比较、每次参与循环计算的点集合判断步骤由 GPU 实现,同时考虑 CPU 与 GPU 之间的较小总线带宽,将 CPU 与 GPU 之间的数据传输最小化。

## 1 数据流聚类算法分析

数据流聚类问题<sup>[35]</sup>概括来说是将连续产生的、没有边界的源源不断到达的大量数据元素所组成的序列划分为若干组,使得组内对象的相似性尽可能地高,而组间对象的相似性尽可能地低。基于划分的方法<sup>[6]</sup>是将数据集  $D$  采用一个划分准则(如距离),划分为  $k$  个子集(簇, cluster),使同一个簇中的对象是相似的,不同簇中的对象是不相似的,使相应准则最优。典型的划分方法包括 K-平均算法、K-中心点算法等。

### 1.1 Enhanced\_K-means 算法

传统的 K-means 算法每次执行循环时均要计算每个数据点和所有  $k$  个中心点的距离,这对于大容量的数据库来说,占用很大的执行时间。而改进的 enhanced\_K-means<sup>[7]</sup>先保留上次循环得到的所有数据点到最近簇的距离和标号,在下步循环中在计算每个数据点到对应的新簇心的距离后,若新距离比原来数据点到原簇心的距离小或者相等,则说明这个数据点还在

收稿日期: 2008-05-29; 修回日期: 2008-08-26 基金项目: 2008 湖南高等学校科学研究资助项目(08C173); 衡阳师范学院青年科学基金资助项目(07A29); 衡阳师范学院教学研究资助项目(A267)

作者简介: 李琳(1974-),女,讲师,硕士,主要研究方向为数据挖掘、并行计算(li0502@sina.com); 李肯立(1971-),男,湖南娄底人,CCF 高级会员,教授,博士,主要研究方向为并行处理、DNA 计算。

原簇,没有发生变化,则下次循环时不需要计算这个点到其他  $k - 1$  个中心点的距离;否则这些远离中心点的数据点就会被重新计算到其他簇中心的距离。因为在一次操作中有一定量数据点还在原簇,则意味着这部分点不参加与所有中心点的比较计算操作,从而节省了总的距离计算时间。

### 1.2 Enhanced\_K-means 算法描述

```

Procedure Distance( )[7] /* 用数组 clusterid[ i ] 和 pointdis[ i ] 分别记录每个数据点到最近簇的距离和标号 */
for i = 1 to n
  for j = 1 to k
    { d2( xi, mj );
  /* 计算当前点 xi 到所有中心 mj 的欧氏平方距离,找出 xi 的最近的点 mj */
  mj = mj + xi;
  nj = nj + 1;
  MSE = MSE + d2( xi, mj )
  clusterid[ i ] = 到最近中心点的标号;
  pointdis[ i ] = 到最近点的欧氏距离; }
for j = 1 to k
  mj = mj/nj;
  Procedure Distance_new( )[7]
  for i = 1 to n
    { d2( xi, clusterid[ i ] )
  //计算点 xi 到对应新中心点的欧氏距离平方;
  if ( d2( xi, clusterid[ i ] ) < = pointdis[ i ] )
  //点 mj 仍然在原来的簇中;
  else
  for j = 1 to k
    d2( xi, mj );
  /* 计算点 xi 到所有中心 mj 的欧氏距离平方; 找出 xi 的最近的点 mj */
  mj = mj + xi;
  nj = nj + 1;
  MSE = MSE + d2( xi, mj )
  clusterid[ i ] = 到最近的中心点的标号;
  pointdis[ i ] = 到最近点的欧氏距离; }
for j = 1 to k
  mj = mj/nj;

```

这两个算法先执行 procedure distance( ),再执行 procedure distance\_new( )。Enhanced\_K-means 算法时间复杂度是  $O(nk)$ <sup>[7]</sup>。其中:  $k$  表示中心点的个数;  $n$  表示数据点个数。比传统算法的时间复杂度  $O(nkl)$  显然少了一个表示循环次数的因子  $l$ ,因此计算效率有一定提高。

## 2 基于 GPU 的 enhanced\_K-means 算法实现

由于 GPU 的并行计算功能主要是通过多个渲染管道(它由多个并行处理单元组成的,在 GeFore7800GTX 中,并行处理单元的个数多达 24 个)和  $(r, g, b, \alpha)$  四个颜色通道同时计算来体现的,顶点程序的多个渲染管道意味着一个时钟周期可以并行处理多个顶点。并行计算的实质<sup>[8]</sup>是对纹理的每个像素并行执行同一运算,该运算对像素的各颜色通道值进行计算并将结果存储到纹理。因此本文将并行处理的优异性能用在 enhanced\_K-means 算法中,通过 GPU 和 CPU 协同工作处理模式,使聚类效率优化。

### 2.1 GPU 和 CPU 协同计算模式

如算法 procedure distance( ) 和 procedure distance\_new( ) 分析所述, enhanced\_K-means 算法主要计算步骤有: a) 初始化; b) 点间欧式距离计算与比较; c) 新中心点计算; d) 中心点改变后簇中没有发生变化的点集合判断; e) 聚类结束条件判

定。计算步骤分配如图 1 所示。其中欧式距离计算与比较是影响时间复杂度的关键计算步骤,交由 GPU 完成;而因为中心点改变后簇中没有发生变化的点集合判断,是以距离计算与比较为基础完成的,也由 GPU 完成,但是考虑到 GPU 处理累加运算不是很适合,而且在 GPU 计算时, CPU 应该同时并行计算,所以生成中心点的计算交由 CPU 完成;同时初始化及聚类结束条件判定也由 CPU 完成。

当类标号计算结果从 GPU 中取回后,在 CPU 中将标号相同的数据点累加并生成新的中心点,然后由 CPU 传回给 GPU。这里 GPU 主要向 CPU 传输标号。为节省传输开销,采用 GL\_BYTE 模式传输标号,这一模式有 8 bit 限制,即值最高可取为 256,当数值结果大于 256 时,算法将增加部分通信代价,执行过程中使 CPU 和 GPU 之间的通信开销尽量降到最低是必要的。

### 2.2 计算方式变化

传统基于 CPU 计算模式的 K-means 聚类算法每次计算从某个数据点到  $k$  个中心点的距离并比较这  $k$  个距离值。从而获得离该数据点最近的中心点标号值和距离值,而在本文基于 GPU + CPU 模式的 K-means 聚类算法中,因为要对参加聚类的数据组织成纹理矩阵,对纹理中的每个数据点并行进行某个操作运算,所以要调整计算顺序<sup>[9]</sup>,将原有的计算顺序改为所有的数据点到某个中心点间的计算,这种计算方式可以更好地利用 GPU 组成纹理形式并行执行。

### 2.3 数据存储转换

利用 GPU 计算,首先要把这些数据组织成合适的纹理格式<sup>[8,9]</sup>,再进行矩阵计算。因为按每 4 维属性构成一个数据点矩阵,即一个纹理,当数据点维度  $d$  高于 4 维时,  $d$  维数据即按 4 维划分成不同的纹理,而每个纹理中的数据点是相对应的(图 2)。对各个纹理矩阵进行计算,并累计各距离矩阵,从而得到最终的距离矩阵。当前 GPU6800 最多可同时支持八个纹理单元,也就是说,在一遍计算过程中,数据点被允许计算的维数是 32 维,若超过 32 维,则采用多遍渲染的方法进行计算,并将各遍渲染结果进行累加,从而获得最终结果。例如: GPU 可同时支持  $s$  个纹理单元,则对  $d$  维数据点距离计算的渲染次数为  $\lceil d/(4s) \rceil$ 。另一方面,随着下一代 GPU 发展,其可同时支持的纹理单元个数会越来越多。假设数据块中有  $M$  个  $d$  维数据点  $P_i (1 \leq i \leq M)$  和  $k$  个中心点  $C_i (1 \leq i \leq K)$ 。首先,  $P_i$  被组织成图 2 中的矩阵(1)形式(称为数据点矩阵  $N$ )。其中  $A$  表示矩阵的列数;  $R$  表示矩阵的行数,  $N = A \times R$ 。矩阵中的元素  $a[m][n]$  对应于数据点  $P_{(m-1) \times A + n}, 1 \leq m \leq R, 1 \leq n \leq A$ , 一般  $A$  或  $R$  取  $N$ 。之后经过对纹理的渲染操作,即进行每个数据点与某个中心点  $c_i$  的距离矩阵(如图 3 矩阵(2)的  $dis(p_i, c_i)$ ) 计算后,将结果存储在深度缓存中。

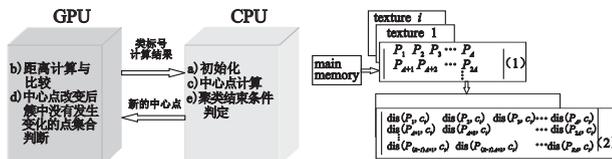


图1 Enhanced\_K-means 计算步骤分配

图2 等于大于4维数据转换计算

### 2.4 数据计算比较

数据计算比较是通过硬件完成<sup>[9]</sup>的,如图 2 中距离矩阵

(2)。计算所有数据点与某个中心点的距离,用深度缓存保存各个数据点到中心点间的距离,每次距离矩阵都将计算完毕的结果直接存入深度缓存中,然后每次均利用硬件进行深度测试;第一次距离计算后将结果直接放入深度缓存,随后计算出来的子素深度值如果小于深度缓存中对应的深度值,则更新深度缓存,否则保持不变;若中心点有  $k$  个,这个过程要进行  $k$  次计算,最终在深度缓存中保留每个数据点到最近的中心点的距离,从而实现距离矩阵之间距离的比较。因为模板缓存也与各像素点一一对应,用模板缓存中的模板值记录对应数据点到最近中心点的标号值。首先将模板值初始化为 1(即假设所有点是离第一个中心点最近),然后模板测试设置为通过状态,在依次计算其他距离矩阵  $D_{i+1}(1 \leq i < k)$  后,若  $D_i$  中新产生的子素通过深度测试,则更新对应像素的深度值,同时将对应的模板值更新为  $i+1$ 。当距离矩阵计算  $k$  次后,模板缓存区保存了与各个数据点最近的中心点标号。GPU 支持可编程的子素处理器,在子素处理器中运行的程序称为子素程序,子素程序可以对纹理进行检索并对子素执行数学计算,本文利用子素程序进行距离计算。

### 2.5 计算过程概述

算法 procedure distance\_new() 首先由 GPU 完成,在经过  $k$  个距离矩阵计算后得到计算结果,即数据点到最近中心点标号保存在模板缓存中,而数据点到最近中心点的距离保存在深度缓存中。将数据点到最近中心点标号送到 CPU 中, CPU 将所有标号相同的数据点累积求平均值,得到新簇心,再将新簇心回传给 GPU。此时在图 2 的纹理矩阵(2)中,因为经过初次处理后,数据分为  $k$  个子集合,各子集合中的标号是相同的,将这些标号相同的  $k$  个子集合与对应的  $k$  个新簇心  $c_i(1 < i < k)$  在一个纹理矩阵中分别求对应数据点到新簇心的欧氏距离。

距离解出后,进行深度测试和模板测试,并在帧缓存中保留此像素。样经过纹理矩阵的计算和测试后最终保留在帧缓存中的就应该为中心点改变后远离原来簇中心的点集合,因此完成了中心点改变后簇中没有发生变化的点集合判断。

这些保留在帧缓存中的簇中心点改变后远离原来簇中心的点集合组成新的纹理矩阵,在下次纹理渲染时,就只要对这个纹理矩阵进行上述的  $k$  次矩阵计算过程即可。此过程中 CPU 通过向 GPU 输入新的中心点来控制聚类计算中的每一次循环。

## 3 实验分析

### 3.1 实验环境

实验在一台配有 Pentium 4 3.4 GHz CPU 和 NVIDIA GeForce7800GT 显卡的方正 PC 上进行,显卡 256 MB,主存 1 GB,使用操作系统为 Windows XP Professional SP2。基于 GPU 的聚类算法采用 OpenGL API 加以实现,基于 CPU 的聚类算法用 VC++6.0 实现;GPU 与 CPU 之间的纹理和数据传送采用 PCI-Express 16x 作为总线接口。实验数据采用 (<http://mllearn.ics.uci.edu/MLSummary.html>) UCI machine learning repository 中的数据 pen-based recognition of handwritten digits。这个数据集有 600 条记录,将数据集的大小取在 10 KB~16 MB,簇的个数为 8128,维度是 420,数据点的各维属性为 32 位的单精度浮点数。在由 CPU 完全进行运算和 CPU 与 GPU 协同模式运算的两种模式下比较算法执行的效率。

### 3.2 测试结果

效率评价采用算法执行时间进行度量。算法的总代价为数据 I/O 传输时间 transtime 和聚类计算时间 clustime。基于 GPU + CPU 模式的算法执行时间 clustime 包括在 GPU 与 CPU 之间传送数据的时间与在 GPU 中的计算时间(用 GPU-K 表示基于 GPU + CPU 模式的 enhanced\_K-means 算法,用 CPU-K 表示基于 CPU 的 enhanced\_K-means 算法,以下同此)。

图 3 所示是 GPU-K 和 CPU-K 算法的聚类计算时间 transtime 的比较。GPU-K 执行时间大概是 CPU-K 执行时间的 45%。这是因为子素处理器的并行计算能力,现代 GPU 通常具有多个子素处理器,在本文进行实验的 NVIDIA GeForce 6800 中有 16 个子素处理器,通过 GPU 中硬件优化的向量指令进行距离计算,从而进一步加速了聚类算法中距离计算操作。

实验还比较了基于 GPU + CPU 模式和基于 CPU 模式的 enhanced\_K-means 算法的总执行时间代价,如图 4 所示。GPU-K 执行时间约是 CPU-K 执行时间的 65%,即 GPU-K 执行效率提高了 35%,与图 3 相比略有增加。这是因为在总的计算时间中考虑了 I/O 传输时间,这部分执行时间占总时间的一定比例,而在聚类计算分析中没有包括 I/O 传输时间分析,所以会出现这种情况。

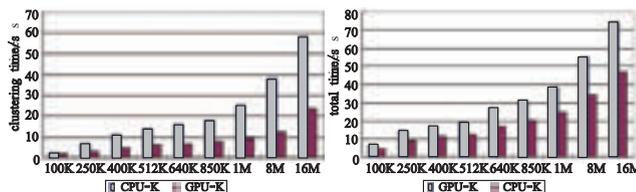


图3 GPU-K和CPU-K聚类计算时间比较

图4 GPU-K和CPU-K总的执行时间比较

## 4 结束语

采用 GPU 计算 enhanced\_K-means 算法中距离计算与比较步骤,经过测试,结果表明在具有相同聚类质量的情况下,基于 GPU 和 CPU 协同处理模式下的聚类算法效率比完全在 CPU 上执行的聚类算法的效率显然要快。

这种用图形处理器进行运算处理的方法在有源源不断的大量数据流到达的情况下,可以满足聚类处理对速度实时性的要求。

### 参考文献:

- [1] OWENS J D. The GeForce 6 series GPU architecture [K]//GPU Gems 2: streaming architectures and technology trends, 2005:453-474
- [2] 吴恩华. 图形处理器用于通用计算的技术、现状及其挑战[J]. 软件学报, 2004, 15(10): 1495-1507.
- [3] 曹锋. 数据流聚类分析算法[D]: 上海: 复旦大学, 2006: 15-20.
- [4] 蒋盛益, 李庆华, 李新. 数据流挖掘算法研究综述[J]. 计算机工程与设计, 2005, 26(5): 1130-1134.
- [5] AGALWAL C C, HAN J, WANG J, et al. A framework for clustering evolving data stream[C]//Proc of VLDB. 2003: 81-92.
- [6] HAN Jia-wei, KAMBER M. 数据挖掘概念与技术[M]. 范明, 孟小峰, 译. 北京: 机械工业出版社, 2005.
- [7] FAHIM A M, SALEM A M, TORKEY F, et al. An efficient enhanced K-means clustering algorithm[J]. Journal of Zhejiang University Science A, 2006, 7(10): 1626-1633.
- [8] TAKIZAWA H, KOBAYASHI H. Hierarchical parallel processing of large scale data clustering on a PC cluster with GPU co-processing [J]. J Supercomput, 2006, 36(10): 219-234.
- [9] 曹锋. 数据流快速聚类[J]. 软件学报, 2007, 18(2): 291-302.