Robust 的分布式 k 中心聚类算法的研究与实现

陶 冶,曾志勇
TAO Ye,ZENG Zhi-yong

云南财经大学 信息学院,昆明 650221
School of Information,Yunnan University of Finance and Economics,Kunming 650221,China
E-mail:taoye_yn@126.com

TAO Ye,ZENG Zhi-yong.Robust distributed k-mediods clustering algorithm.Computer Engineering and Applications, 2009,45(32):122-125.

Abstract: Parallel is very important in data mining.This paper proposes a distributed k-mediods clustering algorithm by analyzing how to get satisfactory clustering information from local information.Its quality is equivalent to serial PAM algorithm but its calculation performance is higher.The paper gives still the way that improves the efficiency of parallel PAM algorithm by reducing the cost of communication,analyzes the performance of the algorithm and gives the result of experiment.This explains that the algorithm is effective and reliable.

Key words: clustering;Partitioning Around Mediods(PAM) algorithm;parallel;Message Passing Interface(MPI)

摘 要:并行处理的研究在数据挖掘中是十分必要的。在理论分析的基础上,提出在对经典串行 PAM 算法进行并行时应如何从局部聚类信息生成完备的全局聚类信息,据此提出了算法 DPAM,在提高计算性能的同时,使聚类质量等价于相应串行 PAM 算法。为提高并行算法的执行效率,还介绍了如何减小计算结点间通信的代价。最后对提出的算法进行性能分析和实验,说明该算法是高效可行的。

关键词:聚类;围绕中心点的划分(PAM)算法;并行;消息传递接口(MPI)
DOI:10.3778/j.issn.1002-8331.2009.32.039 **文章编号**:1002-8331(2009)32-0122-04 **文献标识码**:A **中图分类号**:TP311

1 概述

数据聚类分析是数据挖掘研究领域中一个非常活跃的研究课题,它在数据挖掘、统计学、机器学习、空间数据库技术、生物学和市场营销等研究领域都有应用。随着数据挖掘中数据量的高速增长,数据挖掘处理海量数据的能力成了一个不可忽视的问题,以 Gigabytes、甚至 Terabytes 为单位的数据集是很普通的[1]。这时即使算法的复杂性是线性增长的,对于空间和时间的要求也是巨大的, 而并行算法是解决这一问题的有效途径,因此,对数据挖掘中的传统串行算法进行并行化的研究,是十分必要的。

2 相关工作

目前,有关聚类算法的并行研究还不多,在划分聚类方面,已有的一些研究主要是对 k-means,如文献[1-3]。k-means 对离群点是敏感的,因为一个具有很大的极端值的对象可能显著地扭曲数据的分布[4]。为了解决这个问题,可以不采用簇中的平均值作为参照点, 而是用簇中位置最靠近中心的数据对象,即中心点作为参照点,这就是 k 中心点算法。它的基本思路是:首先为每个簇任意选择一个对象作为中心点,剩余的对象根据其与各簇中心点的距离分配给最近的一个簇。然后反复地用非中心点对象来代替中心点对象,以改进聚类的质量。这个划分方法仍然是基于最小化所有对象与其参照点之间的相异度之和的原则来执行的[5]。但是,当存在噪声和离群点时,k 中心点方法比 k 均值鲁棒。这是因为中心点不像均值那样容易受离群点或其他极端值的影响[4]。PAM(Partitioning Around Medoids,围绕中心点的划分)是一个经典的 k 中心点算法[4-5],这里将对其进行并行设计。

3 PAM 串行算法

PAM 希望用簇中位置最中心的对象作为代表对象来代表该簇,其余的每个对象聚类到与其最相似的代表对象所在的簇中。这样,就把 n 个对象的集合划分为 k 个簇。这里,其他对象也被称为非代表对象。

PAM 算法进行划分的原则是最小化所有非代表对象与其对应的代表对象之间的相异度之和。即,使用绝对误差标准,其定义如下:

基金项目:云南省自然科学基金(the Natural Science Foundation of Yunnan Province of China under Grant No.2007G079M);云南省教育厅科学研究基金项目(the Scientific Research Project of Department of Education of Yunnan Province of China under Grant No.6Y0159D)。
作者简介:陶冶(1969-),男,讲师,主要研究领域为数据挖掘,并行计算;曾志勇(1974-),男,博士研究生,副教授,主要研究领域为数据挖掘,并行计算。
收稿日期:2009-04-24 **修回日期**:2009-06-16

表 1　PAM 的串行算法(SPAM)和并行算法(DPAM)

PAM 串行算法(SPAM)	PAM 并行算法(DPAM)
输入:k,结果簇的个数;D,包含 n 个对象的数据集合	输入:k,结果簇的个数;D,包含 n 个对象的数据集合
输出:k 个簇的集合	输出:k 个簇的集合
(1)从 D 中读取数据,并从中随机选择 k 个对象作为初始的代表对象;	(1)进程 0 从 D 中读取数据,并从中随机选择 k 个对象作为初始的代表对象
(2)	(2)进程 0 将所有数据和 k 个代表对象广播到所有进程;
(3)do {	(3)do {
(4)　计算每个对象与 k 个中心点之间的距离,并找出离该对象距离最近的中心点和次近的中心点	(4)　各进程计算每个对象与 k 个中心点之间的距离,并找出离该对象距离最近的中心点和次近的中心点
(5)　for i=1 to k	(5)　for i=1 to k{
(6)　　for m=1 to n	(6)　　for m=1 to n{
(7)　　$C[i][m]$=0	(7)　　$C[i][m]$=0
(8)　　if O_m 是非代表对象{	(8)　　if O_m 是非代表对象{
(9)　　for j=1 to n{	(9)　　for j=1 to n/p{
(10)　　计算 C_{jim}	(10)　　计算 C_{jim}
(11)　　$C[i][m]$+=C_{jim}	(11)　　$C[i][m]$+=C_{jim}
(12)　　}	(12)　　}
(13)　}	(13)　}
(14)　}	(14)　}
(15)}	(15)}
(16)	(16) MPI_Reduce(C,$cost$,MPI_SUM,0)
(17) 在数组 C 中找出最小值 min	(17) 进程 0 在数组 $cost$ 中找出最小值 min
(18) if　min<0{	(18) if　min<0{
(19)	(19)　进程 0 将 min,m,i 广播出去
(20)　用 O_m 替换 O_i,成为一个新的代表对象	(20)　各进程用 O_m 替换 O_i,成为一个新的代表对象
(21)}	(21)}
(22)}while (min<0)	(22)}while (min<0)
(23)形成 k 个簇的集合,算法结束	(23)形成 k 个簇的集合,算法结束

$$E=\sum_{i=1}^{k}\sum_{p\in C_i}|p-o_i| \tag{1}$$

其中,E 是数据集中所有对象的绝对误差之和;p 是空间中的点,表示簇 C_i 中一个给定对象;o_i 是簇 C_i 中的代表对象。

为了找到最合适的代表对象,该算法首先随机选择 k 个对象作为 k 个簇的代表对象,然后,反复地选择簇的更好的代表对象,即,用所有非代表对象 O_m 来代替所有代表对象 O_i,O_i 将被某个可以使误差值减少最多的非代表对象所取代,成为下次迭代的新的中心点。这个迭代一直进行到当前的每个代表对象都成为它的簇的实际中心点。

当某个非代表对象 O_m 替代某个代表对象 O_i 后,对于数据集中的每一个样本 O_j,如果把其绝对误差的差视为代价,O_j 的代价可以分下面的四种情况来考虑。先说明各符号代表的含义:O_i 和 O_s 是两个中心点,$i\neq s$,O_m 将替换 O_i 作为新的中心点,$d(j,i)$是 O_j 到 O_i 的距离,$d(j,s)$是 O_j 到 O_s 的距离,$d(j,m)$是 O_j 到 O_m 的距离,C_{jim} 表示 O_j 在 O_i 被 O_m 代替后产生的代价。

情况 1　O_j 当前隶属于 O_i 所代表的簇。当 O_i 被 O_m 代替后,O_j 离 O_s 最近,那么 O_j 将被重新分配给 O_s。这时,

$$C_{jim}=d(j,s)-d(j,i) \tag{2}$$

情况 2　O_j 当前隶属于 O_i 所代表的簇。当 O_i 被 O_m 代替后,O_j 离 O_m 最近,那么 O_j 被重新分配给 O_m。这时,

$$C_{jim}=d(j,m)-d(j,i) \tag{3}$$

情况 3　O_j 当前隶属于 O_s 所代表的簇。如果 O_i 被 O_m 代替后,O_j 依然离 O_s 最近,那么 O_j 的隶属不发生变化。这时,

$$C_{jim}=0 \tag{4}$$

情况 4　O_j 当前隶属于 O_s 所代表的簇。如果 O_i 被 O_m 代替后,O_j 离 O_s 最近,那么 O_j 被重新分配给 O_m。这时,

$$C_{jim}=d(j,m)-d(j,s) \tag{5}$$

O_m 替代 O_i 的总代价就是所有对象所产生的代价之和,为:

$$TC_{im}=\sum_{j=1}^{n}C_{jim} \tag{6}$$

如果总代价是负的,表明绝对误差将会减小,此时,可以用 O_m 替代 O_i。如果总代价是正的,则当前的中心点 O_i 被认为是可接受的,不被替代。

表 1 中给出 PAM 的串行算法(SPAM)和并行算法(DPAM),SPAM 中故意加了一些空行,这是为了方便读者对比两个算法。

下面来分析算法 SPAM 的效率。把每一个加法、乘法或比较看作是一个浮点运算(floating point operation,flop),它所用的时间记为 T_{flop}。如果不考虑 I/O,只考虑算法的核心部分,即步骤(3)~(19),表 2 列出在每次迭代中重要步骤中所需的运算时间。表中的 n 是数据集中样本的数量,d 是聚类所用到的属性的数量,k 是簇的数量。

表 2　SPAM 算法的重要步骤所需的运算时间

重要步骤	运算时间
4	$nkdT_{flop}$
5~15	$k(n-k)ndT_{flop}$
17	nT_{flop}

从表 2 可得算法 SPAM 在每次迭代中总的执行时间为:

$$T_s=(nkd+k(n-k)nd+n)T_{flop} \tag{7}$$

因为 n 通常远大于 k,可以把算法 SPAM 每次迭代时间复杂度看作 $O(kdn^2)$。可见,当 n 非常大时,算法 SPAM 的时间复杂度不容乐观。降低算法 SPAM 的时间复杂度最直观的办法就是减少参加聚类的对象数量 n。但减少对象的数量会影响聚类的质量。

4 消息传递接口(MPI)

现在,找到通过网络连接的多台完整的计算机是很方便的(这里,“完整”的含义是指具有独立的 CPU 和内存),它们通过传递消息协同工作,称为消息传递多处理机,如图 1 所示。要使这些处理机协同工作必须要有相应的软件,学术界和产业界的合作伙伴联合研发了一种标准,称为 MPI(Message Passing Interface,消息传递接口)[6]。

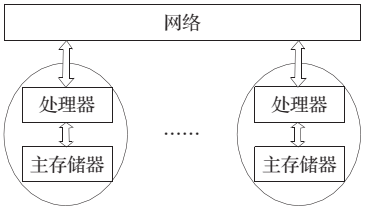

图 1 消息传递多处理机模型

MPI 定义了一组具有可移植性的编程接口。1994 年发布了第一个版本,到 1997 年发布了第二个版本。随着高性能计算技术的普及,尤其是集群系统的普及,MPI 标准如今已经成为事实上的消息传递并行编程标准,也是最为流行的并行编程接口。基于 MPI 的并行程序可以分为单程序多数据(Single Program Multiple Data,SPMD)和多程序多数据(MPMD)两种形式。SPMD 使用一个程序来处理多个不同的数据集以达到并行的目的。相应的,MPMD 则使用不同的程序处理多个数据集。SPMD 是 MPI 程序中最常用的并行模型[7]。该文正是基于 MPI 中的 SPMD 计算模型讨论 PAM 串行算法的并行化问题。假设有 p 个处理机参加运算,可将 n 个数据对象平均分配到每个处理机,这样,既达到了减少对象数量的目的,又不影响聚类的质量,很好地解决了 PAM 串行算法遇到的矛盾。

5 PAM 并行算法

在设计算法的时候,特别要注意并行聚类算法必须能够从局部聚类信息生成完备的全局聚类信息,这样聚类质量才能等价或优于相应串行算法。在此前提下,要尽量减小计算结点间通信代价,提高算法的执行效率[8]。从表 2 和式(7)可以看到,SPAM 算法中最费时的地方是计算某个非代表对象替代某个代表对象的总代价,时间复杂性达到了 $O(kdn^2)$。

定理 1 在消息传递多处理机模型中,PAM 算法里一个代表对象被一个非代表对象所代替的总代价等于各处理机中分代价之和。

证明 若有 p 个进程,n 个样本,每个进程中有 $n_1,n_2,\cdots,n_p$ 个样本。由式(6),

$$TC_{im}=\sum_{j=1}^{n}C_{jim}=C_{1ih}+C_{2ih}+\cdots+C_{nih}$$

如果把 1~n_1 个代价记为 $C_{1ih}^1C_{2ih}^1\cdots C_{n_1ih}^1$,$n_1+1$~$n_2$ 个代价记为 $C_{1ih}^2C_{2ih}^2\cdots C_{n_2ih}^2$,$\cdots$,$n_{p-1}+1$~$n_p$ 个代价记为 $C_{1ih}^pC_{2ih}^p\cdots C_{n_pih}^p$,则

$$TC_{im}=C_{1ih}^1+C_{2ih}^1+\cdots+C_{n_1ih}^1+C_{1ih}^2+C_{2ih}^2+\cdots+C_{n_2ih}^2+\cdots+C_{1ih}^p+C_{2ih}^p+\cdots+C_{n_pih}^p=\sum_{j=1}^{n_1}C_{jim}^1+\sum_{j=1}^{n_2}C_{jim}^2+\cdots+\sum_{j=1}^{n_p}C_{jim}^p=\sum_{q=1}^{p}\sum_{j=1}^{n_q}C_{jim}^q \quad (8)$$

得证。

设在消息传递多处理机模型中,每个处理机将处理 n/p 个数据,根据定理 1,将 SPAM 的步骤(9)改为算法 DPAM 中的步骤(9),即让各个处理机分别计算分代价,这样,步骤(5)~(15)的时间复杂度可以降到 $O(n^2/p)$,从而提高了计算性能。然后在步骤(16)将所有分代价相加得到总代价。此处用到了一个 MPI 函数 MPI_Reduce,它是将所有进程(处理机中运行的 DPAM 程序)中的 $C[i][m]$相加,并将结果放到进程 0 的 $cost[i][m]$。在 MPI 中,消息传递是相当费时的,每一次消息传递都包括一个启动时间和一个传递时间。为了提高通信性能,应尽量减少通信的次数。所以在一次迭代中,只使用一次 MPI_Reduce 传递(kn)个浮点数,这样做大大节省了通信的时间。此外还尽量使用组通信,如第(19)行用到了 3 次广播,它是从进程 0 将数据广播到所有进程。

6 DPAM 的性能分析

并行计算所用时间由通信和计算两部分组成。和对 SPAM 的分析一样,不考虑 I/O,只考虑算法核心部分。表 3 和表 4 里符号的含义同表 2。

(1)通信时间

由表 3 得,DPAM 中每次迭代通信所需的时间为:

$T_{comm}=knT_{reduce}+3T_{bcast}$

表 3 中,T_{reduce} 是归约一个数据所用的时间,T_{bcast} 是广播一个数据所用的时间,对于特定的环境,它是常数,所以可以把 DPAM 的通信时间复杂度看作 $O(kdn)$。

表 3 DPAM 算法的重要步骤所需的通信时间

重要步骤	通信时间
16	knT_{reduce}
19	$3T_{bcast}$

(2)计算时间

由表 4 得,DPAM 中每次迭代计算所需的时间为:

表 4 DPAM 算法的重要步骤所需的运算时间

重要步骤	运算时间 4
4	$nkdT_{flop}$
12~15	$nkd(n-k)/pT_{flop}$
17	nT_{flop}

$$T_{comp}=(nkd+\frac{nkd(n-k)}{p}+n)T_{flop}$$

因为 n 通常远大于 k,可以把 DPAM 的计算时间复杂度看作 $O(kdn^2/p)$。

(3)总的执行时间[6]

$T_p=T_{comm}+T_{comp}$

(4)加速比系数[6]

$$\text{加速比系数}=\frac{T_s}{T_p}=\frac{(nkd+nkd(n-k)+n)T_{flop}}{(nkd+\frac{nkd(n-k)}{p}+n)T_{flop}+knT_{reduce}+3T_{bcast}}\approx\frac{1}{\frac{1}{p}+\frac{T_{reduce}}{(n-k)dT_{flop}}}(\text{因为 } n \text{ 很大}) \quad (9)$$

T_{reduce} 的取值与 p 和 n,以及一些随机因素有关,目前尚不能准确估计。在处理机较少的情况下,$\frac{T_{reduce}}{(n-k)dT_{flop}}$的值较小,可以忽略不计,此时加速比为 p。当处理机较多时,$\frac{T_{reduce}}{(n-k)dT_{flop}}$的值会变大,此时不能忽略。

(5)计算通信比[6]

$$\text{计算通信比}=\frac{T_{comp}}{T_{comm}}=\frac{(nkd+\frac{nkd(n-k)}{p}+n)T_{flop}}{knT_{reduce}+3T_{bcast}}\approx\frac{(n-k)dT_{flop}}{pT_{reduce}} \tag{10}$$

随着结点 p 的增加，$\frac{T_{reduce}}{(n-k)dT_{flop}}$ 也将变大，则计算通信比会减少。

7　实验与分析

实验所用的数据参照 http://archive.ics.uci.edu/ml/上的 Iris 数据集，随机生成样本数分别为 1 500、3 000、4 500、6 000、7 500、9 000 的 6 个数据集。实验所用硬件环境为：2.0 GHz CPU，128 MB 内存，软件环境为 Ubuntu8.10，MPICH1.2.7。实验分别在 1 台、2 台、4 台、8 台、16 台微机上进行，结果如表 5 所示，表中记录的数据为每次迭代所用的平均计算时间和平均通信时间，单位为秒。

表 5　实验数据　　s

数据集		1 台用时	2 台用时	4 台用时	8 台用时	16 台用时
1 500 个样本	计算时间	2.278 513	1.157 894	0.571 594	0.285 818	0.147 657
	通信时间	0.000 080	0.012 129	0.024 046	0.070 786	0.040 100
3 000 个样本	计算时间	9.072 008	4.550 058	2.288 711	1.142 505	0.578 148
	通信时间	0.000 129	0.062 976	0.089 703	0.266 304	0.150 956
4 500 个样本	计算时间	20.464 164	10.204 476	5.073 604	2.563 323	1.317 334
	通信时间	0.000 195	0.209 089	0.227 269	0.650 901	0.758 935
6 000 个样本	计算时间	37.075 372	18.122 808	9.032 487	4.528 234	2.273 607
	通信时间	0.000 259	0.739 126	0.217 017	1.150 326	0.874 691
7 500 个样本	计算时间	57.055 378	28.448 526	14.516 666	7.063 041	3.710 173
	通信时间	0.000 317	1.076 802	0.021 238	1.736 136	1.226 850
9 000 个样本	计算时间	81.946 264	41.552 474	20.913 737	10.407 632	5.084 857
	通信时间	0.000 376	0.027 493	0.507 209	2.398 684	1.643 052

通过表 5 中的数据可得图 2。图 2 以样本个数为横坐标，计算时间为纵坐标。从上到下依次为 1、2、4、8、16 台处理机的结果。从图 2 可以直观地看到，对同一个数据集，计算时间约为串行时间的 1/p，达到了 DPAM 的设计要求。

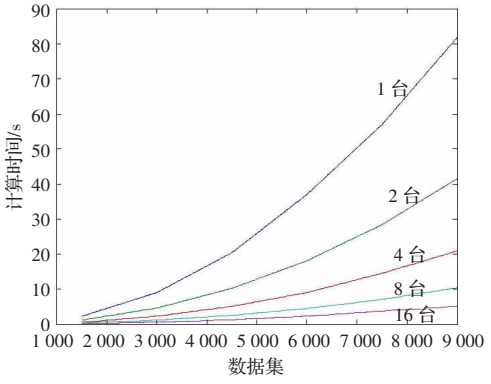

图 2　DPAM 的在不同数据集下的计算时间

如果用 1 台处理机的（计算时间−通信时间）除以多台处理机的（计算时间+通信时间），可得一次迭代的加速比。如表 6 所示。

通过表 6 中的数据可得图 3。图中各条曲线分别代表不同数据集的加速比，因其靠得太近，不便加上数据集的说明，不过在此只是做一般性的说明，不影响分析。从图中可以看到，结点

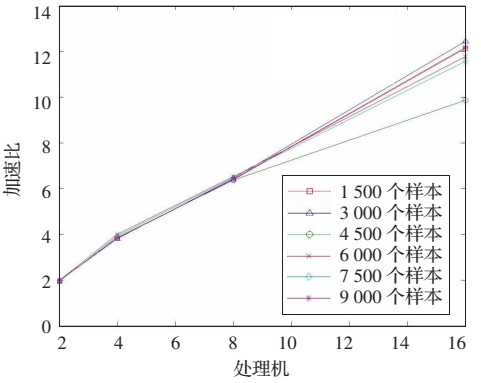

图 3　不同数据集的加速比

小于 8 的时候，每个数据集的加速比很接近，当结点大于 8 后，各个加速比的区别就分出来了。从表 6 中的数据还可以看到，对不同数据量的情况，2 台和 4 台处理机的加速比约为处理机的数量，8 台和 16 台处理机的加速比明显小于处理机的数量。这符合前面对式(9)的分析。

表 6　加速比

数据集	2 台的加速比	4 台的加速比	8 台的加速比	16 台的加速比
1 500 个样本	1.947 340	3.825 185	6.389 253	12.135 010
3 000 个样本	1.966 575	3.814 256	6.439 396	12.442 503
4 500 个样本	1.965 126	3.860 490	6.366 690	9.856 126
6 000 个样本	1.965 605	4.008 335	6.528 964	11.776 240
7 500 个样本	1.932 411	3.924 573	6.484 136	11.556 572
9 000 个样本	1.970 802	3.825 503	6.398 865	12.179 994

8　小结

对 PAM 聚类算法的并行进行了理论上的研究，并予以实现。从加速比系数和计算通信比来看，该算法是值得并行的。实验结果也证明了这一点。从表 5 的实验结果发现通信时间不稳定，如何准确度量通信时间是下一步工作的重点。

参考文献：

[1] Dhillon I S，Modha D S.A data-clustering algorithm on distributed memory multiprocessors[EB/OL].(1999).http://www.cs.rpi.edu/~zaki/WKDD99/dhillon.ps.gz.

[2] Kantabutra S，Couch A L.Parallel k-means clustering algorithm on nows[J].NECTEC Technical Journal，1999，1(1)：243-247.

[3] Datta S，Giannella C，Kargupta H.K-means clustering over a large，dynamic network[C]//2006 SIAM Conf on Data Mining，Bethesda，MD，2006.

[4] Han Jiawei，Kamber M.Data mining：Concepts and techniques[M].北京：机械工业出版社，2007：264-265.

[5] 毛国君，段立娟，王实，等.数据挖掘原理与算法[M].北京：清华大学出版社，2007：175-177.

[6] Wilkinson B，Allen M.Parallel programming：Techniques and applications using networked workstations and parallel computers[M].Beijing：Higher Education Press，2006：24-25，62-63.

[7] 多核系列教材编写组.多核程序设计[M].北京：清华大学出版社，2007：201-204.

[8] 倪巍伟，陈耿，孙志挥.一种基于数据垂直划分的分布式密度聚类算法[J].计算机研究与发展，2007，44(9)：1612-1617.