集群环境下的并行聚类算法之研究*)

周 兵[1] 冯中慧[2] 王和兴[1]
(东北大学秦皇岛分校计算机工程系 秦皇岛 066004)[1]
(西安交通大学电子与信息工程学院 西安 710049)[2]

摘 要 本文的目的就是通过理论分析和试验,探讨集群环境下并行聚类算法的设计思想。作为一种低成本、通用并行系统,集群系统的通讯能力相对于节点的计算能力是一个瓶颈。所以本文提出,在集群环境下设计并行聚类算法时,应采用数据并行的思想。本文首先从理论上,对采用数据并行思想后影响加速比的因素和通讯策略的选择进行了分析,然后实现了一个新的并行聚类算法——PARC算法。通过PARC算法的实验,证明了理论分析的正确性,并且表明并行聚类算法可以得到良好的聚类质量。本文的研究结果可以为以后设计更好的数据并行聚类算法提供一定的理论依据。

关键词 数据挖掘,聚类算法,并行计算,集群,通讯,数据并行

The Study of Parallel Clustering Algorithm for Cluster System

ZHOU Bing[1] FENG Zhong-Hui[2] WANG He-Xing[1]
(Department of Computer Science & Engineering, Northeastern University at Qinhuangdao, Qinhuangdao 066004)[1]
(School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049)[2]

Abstract The goal of this paper is to discuss the designing ideal of a parallel clustering algorithm for Cluster System. As a kind of low-cost all-purpose parallel computing system, the bottleneck of Cluster System is the communication ability compared to the computing ability. This paper proposes the idea of adopting data parallelism when designing a parallel clustering algorithm for Cluster System. Firstly, this paper theoretically analyzes the influence of the speedup and the choice of communication scheme. And then develops a parallel clustering algorithm named PARC. The experiments prove the correctness of the theoretical analysis and show that parallel clustering algorithm could obtain good quality of clustering as linear clustering algorithm. This theoretical analysis will help us to design better parallel clustering algorithm based on Cluster System.

Keywords Data mining, Clustering algorithm, Parallel computing, Cluster system, Communication, Data parallelism

1 引言

集群系统[1,2]是近年来出现的一种新型并行计算环境。它是利用网络,将一组高性能工作站或PC机,按照某种结构连接起来,在一定专用软件统一调度、协调处理下,实现高效并行处理的系统。从结构和结点间的通信方式来看,它属于分布存储系统,主要利用消息传递方式实现各主机之间的通信,由建立在一般操作系统之上的并行编程环境完成系统的资源管理及相互协作。对程序员和用户来说,集群系统是一个整体的并行系统,具有成本低、使用方便、扩展性好等优点,在许多领域得到了广泛的应用和研究。

聚类,又称无指导的学习,是一个将数据库中的数据划分成具有一定意义的子类,使得不同子类中的数据尽可能相异,而同一子类中的数据尽可能相同的过程。由于聚类技术无需任何应用领域知识就能发现数据中隐含的关系和模式,受到了数据挖掘研究人员的广泛重视,并被看作是数据挖掘的主要任务之一。聚类在模式识别、图像处理等领域有着广泛的应用。但因为其处理对象多为海量数据库和高维数据类型,对时间和空间复杂性的要求很高,所以设计并行的聚类算法,以提供更好的计算能力是十分必要的。

因此,如何在集群环境下,设计并行聚类算法,是一个值得关注的研究方向。而目前,在国内外,在这一领域的研究才刚刚起步。本文的目的就是通过理论分析和试验,探讨集群环境下并行聚类算法的设计思想,以及影响加速比和聚类质量的因素。

本文的其余部分的安排如下:在第2节介绍了前期的一些并行聚类算法。第3节详细讲述了本文对并行聚类算法的加速比和通讯策略的研究。第4节是的试验结果。最后是结论。

2 相关的工作

串行聚类算法在统计和数据库领域得到了大量的研究和应用,早期的有k-means,k-mediods,以后又有面向大规模数据库系统的BIRCH[3]算法,处理非数值属性聚类的CACTUS[4]算法、处理空间数据的STING[5]算法、子空间聚类算法ENCLUS[6]等。

*)本课题得到东北大学985工程,科技创新平台,"基于内容的并行图像检索技术的研究"和"无线移动自组网络安全技术研究"以及河北省教育厅2006年科研计划(Z2006303)资助。**周 兵** 讲师,博士,主要研究方向为数据挖掘及基于内容的检索系统等;**冯中慧** 讲师,博士,主要研究方向为数据挖掘、中间件技术和并行计算;**王和兴** 硕士,讲师,主要研究方向为并行计算、网络安全。

同时许多相关的工作对并行聚类算法也做了很有意义的探讨和研究。但有些早期的工作无法处理大规模高维数据库。如采用分割聚类[7,8]和层次聚类方法[9,10]的并行聚类算法，要求处理数据一次性读入内存。这对大规模数据库来说，是不可能的。

还有些并行聚类算法是基于共享存储系统等低通讯延迟系统的，如文[10,11]是基于PRAM(Parallel Random Access Machine)模型的。这些并行聚类算法在设计算法时没有把通讯代价作为重点来考虑。因而这些算法往往采用计算并行的设计思想，把全局聚类模式分布在各个计算节点中，以便可以并行地进行聚类模式匹配。这一过程中，匹配过程的启动及结果的收集都需要通讯。也就是说，处理的每一步至少需要两次通讯。作为一种低成本、通用并行系统集群系统的通讯能力相对于节点的计算能力，是一个瓶颈。因为集群系统的通讯机制一般是MPI、PVM等，基于TCP/IP和UDP通讯协议的消息传递机制，所以通讯延迟非常大，一般达到毫秒级[2]，因而这些采用计算并行的并行聚类算法不适用于集群环境。

在文[12]中，提出了一种结合数据并行和任务并行的并行聚类算法。该算法需要多次读取数据，而且由于采用任务并行，其通讯次数仍然较多，仍然不适合在集群环境使用。

本文提出，在集群环境下设计并行聚类算法时，应采用数据并行的思想，即把整个数据空间按照节点数分成若干子空间，每个节点拥有一个自己的局部聚类模式，然后对每个子空间独立地进行聚类，最后各自输出聚类结果。这样，一方面可以并行处理数据的读取，同时通讯次数可以减少为零。当然，在聚类过程中，也可以进行一定量的通讯，交换各个节点的聚类信息，以改善聚类质量。但这些通讯本身并非必不可少的，通讯次数可以人为控制在一个合适的范围内。

本文首先从理论上对采用数据并行思想后，影响聚类算法的聚类质量和加速比的因素进行了分析。然后根据理论分析，实现了一个集群环境下的并行聚类算法。本文的试验较好地验证了理论分析的正确性。本文的研究可以为以后设计性能更好的数据并行聚类算法提供一定的理论依据。

3 数据并行的思想

集群环境下的并行聚类算法应采用数据并行的思想，即把整个数据空间分成若干子空间，对每个子空间使用相同的聚类方法，独立地进行聚类，最后输出各自的聚类结果。为了保证能对每个子空间独立地进行聚类，需要以下两个条件：(1)是要保证聚类算法同数据特性的无关性，即在聚类过程中，不要求输入某些特定的数据；输入任何的数据都可以保证聚类过程正常地进行，否则各个节点就有可能必须通过通讯来从其它节点获得必要的数据。(2)是减少聚类算法同全局聚类结构的相关性。因为要获得全局聚类结构，必须通过通讯，从其它节点收集信息，所以可以让每个节点维护一份自己的局部的聚类模式。如果能满足这两个条件，各个节点就可以独立地完成聚类过程，并行聚类算法的通讯的代价就会很小，甚至可以忽略不计。

3.1 加速比的影响因素

在本文，一个聚类过程分为三步：第一步是读取数据；第二步是搜索合适的聚类；第三步是合并聚类，即把数据加入聚类后对聚类模式所作的修改。一般认为，除了数据量，影响读取数据过程的主要是硬盘的读写速度；影响搜索聚类过程的主要是聚类的表达方式、聚类个数及聚类索引结构；而影响合并聚类过程的因素很多，根据不同的算法，差异很大，是一个变数。

本文用公式：

$$\cos t_s = A(n) + S(n) + M(n) \tag{3.1.1}$$

来描述串行聚类算法的执行时间；采用数据并行思想后，如果均匀分配子空间，并行聚类的执行时间可以用公式：

$$\cos t_p = A(n/p) + S(n/p) + M(n/p) + E(n/p) \tag{3.1.2}$$

来描述，其中 A 为读取(Access)数据的时间，S 为搜索(Search)合适的聚类的时间，M 为合并(Merge)聚类的时间，E 为交换(Exchange)聚类信息的时间，n 是数据量，p 是计算节点的个数。

因为 S 同聚类个数 K 一般有如下关系：对于线性搜索法有 $S(n)=nKSTEP$，$STEP$ 为进行一次比较的时间，对非线性搜索法，如索引树，有 $S(n)=nSTEP\log K$，所以有以下关系式：

$$A(n/p)=A(n)/p, S(n/p)=S(n)/P \tag{3.1.3}$$

由以上公式和加速比的定义，有

$$Speedup=\frac{p}{1+\frac{pM(n/p)-M(n)}{\cos t_s}+\frac{pE(n/p)}{\cos t_s}} \tag{3.1.4}$$

如果通讯代价 $E(n/p)$ 相比于串行计算时间 $\cos t_s$ 很小(采用数据并行后，可以把通讯次数控制在很小的范围内，而且一般情况下，串行程序的运行时间是相当大的)，则有

$$Speedup=\frac{p}{1+\frac{pM(n/p)-M(n)}{\cos t_s}} \tag{3.1.5}$$

可见，并行聚类算法的加速比除了受节点个数的影响，还受到合并聚类过程的影响。这里有三种情况：第一种，如果 $M(n/p)=M(n)/p$，则 $Speedup=p$，算法具有线性的加速比，且等于节点数；第二种情况，如果 $M(n/p)\ll M(n)/p$，则 $Speedup>p$，那么算法具有超线性的加速比；第三种，如果 $M(n/p)\approx M(n)$，则有

$$Speedup\approx\frac{p}{1+\frac{(p-1)M(n)}{\cos t_s}} \tag{3.1.6}$$

这时加速比取决于 M 在整个聚类过程中的比例，如果比例很大，则 $Speedup\approx 1$。

从以上分析可以看出，对数据并行聚类算法来说，在不考虑通讯的影响下，合并聚类的方法对整个聚类算法的加速比的影响是巨大的。在设计并行聚类算法时，如何使合并聚类的时间随着节点数的线性增加而线性甚至超线性减少，是获得高加速比的重点。

3.2 通讯策略的选择

为了在各个节点间交换聚类结果，需要一定量的通讯。一般来说，可以采用的通讯策略有两种，同步通讯策略和异步通讯策略。那么在设计并行聚类算法时，如何选取通讯策略呢？通讯策略同并行聚类算法之间有没有内在的联系呢？下面的分析将回答以上两个问题。

同步通讯策略是指各个节点在完成各自的计算后，同时开始通讯过程。采用的方法一般是折半通讯方法，即以两个节点为一组，一个是信息收集节点，一个是信息发送节点。在一次通讯中，所有的信息发送节点同时向各自的信息收集节点发送信息。当完成一次通讯后，把信息收集节点再分组，再通讯，直到信息收集节点变成一个时，结束通讯。通讯次数为

$\log_2 p$(p 是节点个数)。为了保证各个节点能同时开始通讯，要把数据集划分为相同大小的子集。

异步通讯策略是指各个节点不是同时开始通讯的，哪个先完成计算，哪个开始通讯。这种方法一般要指定一个专门的接收节点用于收集信息，其它节点向它发送信息。为了尽量使通讯过程同计算过程能同时进行，要把数据集划分为大小不同的子集。因为如果把子集划分为相同的大小，那么各节点就会基本同时完成聚类计算，同时开始向接收节点发送信息。但因为接收节点一次只能响应一个通讯请求，结果势必造成其它节点处于空闲等待状态，浪费计算资源。把子集划分为不同的大小，节点的聚类过程就会形成时间差。每个节点依次完成聚类和通讯，而不会导致通讯的冲突。同时当某一节点开始通讯时，其它节点仍然处于计算状态，使通讯过程同聚类计算过程能达到最大的重合，不会浪费计算资源。这种通讯一次只能处理一个节点的信息交换，所以要 $p-1$ 次通讯才能完成所有信息交换。

图 1 是这两种通讯策略的示意图。两种通讯策略各有优缺点。从通讯效率的角度看，同步通讯的通讯次数大大小于异步通讯，效率高。但从通讯与计算的重叠度来看，异步通讯的通讯过程和计算过程的重叠度较大。那么那种通讯策略更好一些呢？下面从代价公式入手，加以分析。

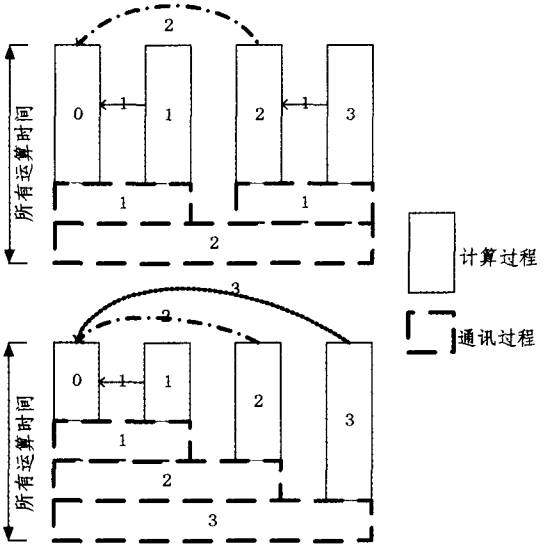


图 1　两种通讯策略的示意图

采用同步通讯的代价公式是

$$Cost_s = C(n/p) + E\log_2 p \qquad (3.2.1)$$

采用异步通讯的代价公式是

$$Cost_a = C(m) + E \qquad (3.2.2)$$

其中 C 是聚类算法的时间代价；E 是一次通讯的时间代价。本文认为对同步通讯和异步通讯来说，E 是相同的；n 是总数据量，p 是节点个数，m 是异步通讯时，最大子集的数据量。

对异步通讯，假设在最理想的情况下，当各个子空间以比例 α 递增时，通讯过程同计算过程达到最大的重合度，也就是说，处理 α 个数据的时间等于一次通讯的时间，即假设等式 $E = C(\alpha)$(3.2.3)成立，同时各个子集的数据量有如下关系：

$$c + c + (c+\alpha) + c(1+2\alpha) + \cdots + (c+(p-2)\alpha) = n \qquad (3.2.4)$$

参见图 1，其中 c 为最小子集的数据量。由公式(3.2.4)可以得到

$$c = \frac{2n-(p-1)(p-2)\alpha}{2p} \qquad (3.2.5)$$

那么最大子集的数据量有

$$m = c + (p-2)\alpha = \frac{n}{p} + \frac{(p-2)(p+1)\alpha}{2p} \qquad (3.2.6)$$

由公式(3.2.1)，(3.2.2)，(3.2.3)和(3.2.6)，同步通讯和异步通讯的代价公式转变为

$$Cost_s = C(n/p) + C(\alpha)\log_2 p \qquad (3.2.7)$$

和

$$Cost_a = C\left(\frac{n}{p} + \frac{(p-2)(p+1)\alpha}{2p}\right) + C(\alpha) \qquad (3.2.8)$$

可见，两种通讯策略的时间代价同聚类程序 C 的特性有很大关系，这里可以分两种情况加以考虑。

第一种情况，如果聚类程序 C 具有线性的时间特性，即有 $C(s+t) = C(s) + C(t)$ (3.2.9)

则有 $Cost_a = C\frac{n}{p} + \frac{(p+2)(p-1)}{2p}C(\alpha)$ (3.2.10)

对比公式(3.2.7)，可以得到同步通讯优于异步通讯的结论，而且随着节点数的增加，同步通讯的优势更加明显。

第二种情况，聚类程序 C 具有收敛的时间特性，即 $C(s+t) < C(s) + C(t)$。特别是如果存在一个阈值 N，当数据量超过 N 时，聚类算法的运行时间基本不变，即有 $C(N+\Delta t) \approx C(N)$，这样可以得到如下公式：

$$Cost_a = C\left(\frac{n}{p} + \frac{(p-2)(p+1)\alpha}{2p}\right) + C(\alpha) \approx C\left(\frac{n}{p}\right) + C(\alpha) \qquad (3.2.11)$$

其中 n/p 大于 N。那么很明显，采用异步通讯策略要好于同步通讯策略。

根据以上分析，可以得到这样一个结论：在设计并行聚类算法时，选择什么样的通讯策略，可以根据聚类程序的特性来判断。如果聚类程序有较明显的线性的时间特性，可以选择同步通讯方式。而如果聚类程序有较强的时间收敛特性，可以选择异步通讯。或将两种方式结合起来，当数据量较小，收敛特性不明显时，用同步通讯；当数据量较大，收敛特性明显时，用异步通讯。

3.3　并行算法的实现

PARC 算法是在串行聚类算法——CLAP 算法[13]的基础上实现的。PARC 算法首先将整个数据空间按照集群节点个数，分成若干子空间；然后对每个子空间使用 CLAP 算法进行聚类。CLAP 算法是一种两阶段的聚类算法，它包括样本聚类和整体聚类阶段。为了提高聚类精度，在样本聚类阶段结束后，各个节点进行一次通讯，来交换初始聚类结构，得到一个全局的初始聚类结构。然后，再完成整体聚类。最后，各个节点再进行一次通讯，来交换各自的聚类结构，得到一个全局的聚类结果。PARC 算法的整体运行过程如下：

Input: dataset D

Input: p, the nodes number of a cluster system, then the nodes is named as N_1, N_2, …, N_p

Input: CLAP(), the clustering method to be applied

Step 1: For each node reads subset D_i to local storage, $1 \leqslant i \leqslant p$

Do Sample-Clustering Phase of CLAP() in parallel

Output local cluster pattern c_i, and the minimal similarity of two clusters s_i

Step 2: Do communication to exchange c_i and s_i

Output an initial global cluster pattern G_{init}

Step3: Do Entire-clustering Phase of CLAP() in parallel

Output local cluster pattern c_i, and the minimal similarity of two clusters s_i

Step 4: Do communication to exchange c_i and s_i

Output the final global cluster pattern G_{final}

4 实验结果

本算法用C语言和MPI消息通讯库实现。集群环境是时实并行集群服务器PCC-8S。它由西安交通大学系统所研制,共8个节点,每个节点的处理器为Pentium II 400,内存256M,硬盘4G,系统软件有RedHat 7.2,MPICH和MYSQL。本文使用两种数据集进行了实验。第一个是1990 US Census Data,来自美国商业部人口调查局的网站:http://www.census.gov/DES/www/des.html。该数据集有2,458,285条记录,每个记录有68个属性。另一个数据集是一些2维图片数据。采用这种2维图片数据是因为聚类结果比较直观。

4.1 通讯策略的对比

首先在4节点和8节点的情况下,使用第一个数据集对同步和异步通讯策略进行了对比试验,结果如表1和表2所示。

表1 8节点的试验结果

数据量	通讯时间(s)			全部运行时间(s)		
	同步	异步		步	异步	
		相等划分	最佳划分		相等划分	最佳划分
5%	11	15	18	17	22	21
20%	32	40	48	74	80	72
50%	111	99	114	302	289	271
100%	389	276	294	1145	1055	987

表2 4个节点的试验结果

数据量	通讯时间(s)			全部运行时间(s)		
	同步	异步		步	异步	
		相等划分	最佳划分		相等划分	最佳划分
5%	23	20	25	64	62	61
20%	95	69	72	286	259	254
50%	363	229	235	1121	987	972
100%	1767	1600	1636	10082	9873	9504

通过对试验结果的对比分析,发现随着数据量的增加,不论是等分划分还是不等划分,异步通讯表现出更好的效果。

从通讯时间看,随着数据量的增加,异步通讯时间从大于同步通讯时间变为小于同步通讯时间。这是因为,随着数据量的增加,各个节点的差异表现得更为明显,所以很难同时进入通讯状态,同步通讯等待时间大大增加。

从全部运行时间看,采用异步通讯的运行时间基本都是小于采用同步通讯的。这是因为采用异步通讯后,整体运行时间取决于数据量最大的节点,但由于PARC算法具有收敛性,数据量最大的节点的计算时间并没有增加多少,因此整体的运行时间反而小。而同步通讯必须等到速度最慢的节点完成计算后,才能进入通讯,所以同步通讯的整体时间性能总是最差的。

另外还可以发现,异步通讯在最佳划分情况下的通讯时间要比相等划分大,但全部运行时间反而小,这正是因为在最佳划分情况下通讯时间与计算时间的重叠度大,通讯时间的增加不会导致全部运行时间的增加。

此外还可以发现,在8节点情况下,异步通讯在数据量为50%处,通讯时间才小于同步通讯时间。而在4节点情况下,在5%处,异步通讯时间就已经小于同步通讯时间了,而且两者差异也大一些。这是因为在8节点情况下,异步通讯的通讯次数多,通讯时间就会长一些。可以想象当节点数较多时,通讯次数对通讯性能的影响将占主流,这时异步通讯的效果也许会比同步通讯效果差。但由于条件限制,本文没有进一步的试验。

4.2 加速比的测定

接下来使用第一个数据集,在采用异步通讯策略的情况下,对使用不同节点个数时的加速比进行了测定,结果如图2所示。

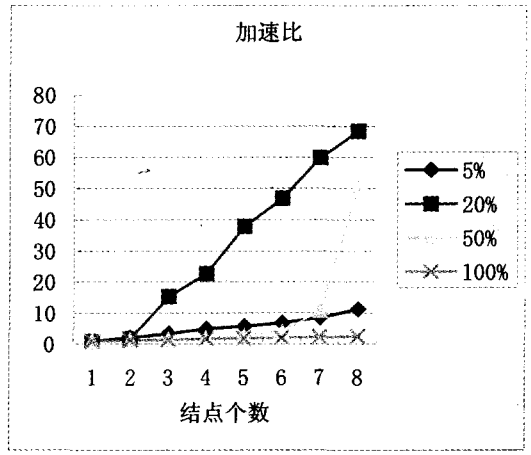


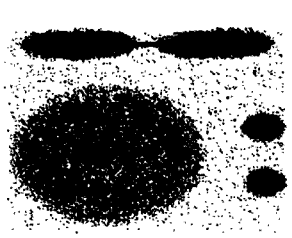
图2 节点数同加速比的关系图

从图2可以看到,当数据量为5%时,加速比基本是线性增加,数值接近节点数;当数据量为20%时,加速比是超线性增加;当数据量为50%时,在节点数较小时,加速比基本相同且接近1,但在节点较大时,呈现超线性增加;当数据量为100%时,加速比接近1。这一现象的产生是由PARC算法的聚类过程的特性决定的,并且符合第3节的理论分析。PARC算法在合并聚类时,要不断重新构造聚类索引树,这是一个较费时的迭代过程。当数据量为5%时,无论使用多少节点都没有重构过程,合并聚类时间同数据量基本成正比,即满足$M(n/p)\approx M(n)/p$,加速比约为p。当数据量为20%时,随着节点数的增加,重构次数大大减少,有$M(n/p)\ll M(n)/p$,所以呈现超线性增加态势。在数据量为50%时,在节点较少时,重构次数减少缓慢,加速比较小,而节点数较大(有7或8个节点)时,重构次数很少,因而加速比增加迅速。当数据量为100%时,随着节点数的增加,重构次数变化不大而且重构次数都较大,所以$M(n/p)\approx M(n)$且合并聚类的时间占整个聚类时间较大,所以加速比趋近于1。通过以上分析,表明试验结果很好地验证了理论分析。同时表明,当数据量较大时,必须进一步增加节点数,PARC算法才能获得较高的加速比,所以还要进一步改进合并聚类的方法。

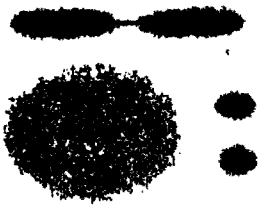
4.3 聚类质量的测试

第二个数据集是3个二维的图片数据,用它们对BIRCH算法和PARC算法的聚类质量进行了对比测定。图片1有四个聚类,其顶部是一个哑铃状的聚类,下方是一个椭圆形聚类。椭圆形聚类旁边是两个距离较近的小聚类。图片2有两

个距离很近的聚类及噪音，且这两个聚类的密度分布是不均匀的。图片3有6个大小、形状都不同的聚类，同时还有一条干扰曲线。结果如图3所示，从左到右依次为：原图，BIRCH算法的结果，PARC算法的结果。从图3中可以看到，并行聚类算法同串行算法都可以正确地发现聚类分布模式。同时，由于并行聚类算法的节点多，内存资源多，聚类的精度相应地也较高，这可以比较清楚地从第三个图片中看到。

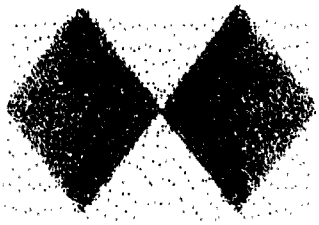


图片1

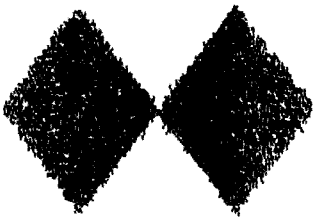


图片1的BIRCH结果

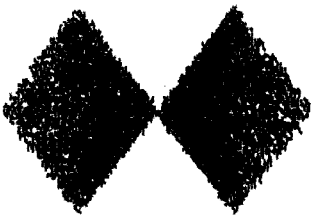
图片1的PARC结果



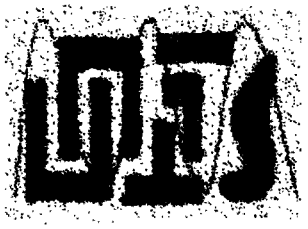
图片2



图片2的BIRCH结果



图片2的PARC结果



图片3

图片3的BIRCH结果

图片3的PARC结果

图3 BIRCH算法和PARC算法的聚类结果对比

结论 本文针对集群环境通讯代价较大的特点，提出了在设计并行聚类算法时，应通过采用数据并行，来减少通讯的思想，并通过理论分析，得出在不考虑通讯代价的情况下，影响加速比的主要因素是聚类合并过程。在选择通讯策略时，根据聚类算法的时间特性来决定采用何种通讯策略。对具有线性时间特性的聚类算法，可以选择同步通讯方式。对具有收敛性时间特性的聚类算法，应该选择异步通讯。最后实现了一个并行聚类算法PARC，证明了理论分析的正确性，并且得到了良好的聚类质量。本文为以后设计性能更好的并行聚类算法提供了理论依据。

下一步的工作是：①重点研究合并聚类的策略，以获得较好的加速比。②目前高维数据的聚类分析的一个难点是如何挖掘蕴含在子空间中的聚类模式。我们将研究适用于集群系统的并行的子空间聚类算法。

参考文献

1 Atiquzzaman M, Srimani P K. Parallel computing on clusters of workstations. Guest editorial, Parallel Computing, 2000, 26: 175～177

2 Warschko T M, Blum J M, Tichy W F. ParaStation: Efficient parallel computing by clustering workstations: Design and evaluation. Journal of Systems Architecture, 1998, 44: 241～260

3 Zhang T, Ramakrishnan R, Livny M. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Canada, 1996. 103～114

4 Ganti V, Gehrke J, Ramakrishnan R. CACTUS-Clustering Categorical Data Using Sumnaries. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge discovery and data mining, San Diego, California, United States, 1999. 73～83

5 Wang W, Yang J, Muntz R. STING: A Statistical Information Grid Approach to Spatial Data Mining. In: Proceedings of the 23rd VLDB Conference, Athens, Greece, 1997. 186～195

6 Cheng Chun-Huang, Fu A Wai-chee, Zhang Yi. Entropy-based Subspace Clustering for Mining Numerical Data. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, USA, 1999. 84～93

7 Ranka S, Sahni S. Clustering on a hypercube multicompute. IEEE Trans Parallel Distributed Syst, 1991, 2(2): 129～137

8 Li X, Fang Z. Parallel clustering algorithms. Parallel Computing, 1989, 11: 275～290

9 Ni L, Jain A. A VLSI systolic architecture for pattern clustering. IEEE Trans Pattern Anal Machine Intell., 1985, 7(1): 80～89

10 Rasmussen E M, Willett P. Efficiency of hierarchical agglomerative clustering using ICL distributed array processor. J Doc, 1989, 45(1): 1～24

11 Olson C F. Parallel algorithms for hierarchical clustering. Parallel Computing, 1995, 21(8): 1313～1325

12 Boutsinas B, Gnardellis T. On distributing the clustering process. Pattern Recognition Letters, 2002.23: 999～1008

13 周兵，沈钧毅，彭勤科. 基于随机抽样和聚类特征的聚类算法. 西安交通大学学报，2003，37(12)：1234～1237