基于结构化 P2P 系统的层次聚类算法①

Hierarchical Clustering Algorithm Based on CAN

徐芳虹 (桂林空军学院 教育技术中心 广西 桂林 541003)
帅剑平 刘天鹏 (桂林电子科技大学 计算机与控制学院 广西 桂林 541004)

摘 要： 利用结构化 CAN 系统中数据严格按照规律分布的特点，将系统坐标空间与聚类数据空间重叠，使相邻数据存储在同一(或相邻)节点上并设计了层次聚类 SOC(Structure Overlay Cluster)，使 CAN 系统数据聚类达到与集中式聚类算法相同的结果。通过分析 SOC 算法的聚类过程得到在数据维数一定的情况下，算法的时间复杂度是 O(N)，即与节点数量成正比,并通过仿真实验得到证明。

关键字： 结构化 坐标空间 聚类数据空间 层次聚类 时间复杂度

1 引言

由于 P2P(Peer-to-Peer)技术的流行，使 P2P 结构的网络上储存了大量的数据。P2P 网络中的这些节点同时充当这资源的拥有者和需求者。如果对这样的一个含有大量数据的系统应用数据挖掘(Data Mining)技术一定会得到许多有用的信息。当前数据挖掘技术和 P2P 技术的结合点大多在于利用数据挖掘技术设计更为合理的资源发现算法或路由算法。传统的分布式数据挖掘(Distribute Data Mining)多是基于 C/S 结构，需要将数据集中到中心站点上，而多处理机的并行数据挖掘算法大多需要处理机之间的消息广播，它们并不能适用于 P2P 这种没有中心的网络系统。

2 相关工作

2.1 结构化拓扑系统 CAN

结构化拓扑是纯粹的 P2P 模型，是针对集中式目录结构模型存在单点失效和安全问题，洪泛请求模型存在消息泛滥而提出了结构化、有址型的分布式系统。结构化拓扑采用 DHT 技术将使每个内容关键字和一个唯一的存储位置相对应。近年来，人们将分布式哈希表(DHT)算法引入大规模的 P2P 系统来提高资源定位查询效率和系统可扩展性。

CAN[1]系统是有 TI 设计的一种结构化 P2P 拓扑系统。它提供了一个可无限扩充的空间网络结构，通过空间坐标和空间分割获得有效的节点路由。CAN 是一种基于几何分割的拓扑结构，在多维笛卡尔空间上实现 DHT 抽象，坐标空间被分成很多个超矩形，每个矩形称为一个区域，相对应于一个节点，区域的边界标志了节点的属性。

2.2 P2P 对数据挖掘的新要求

参考文献［6］提出了一种在 CAN 网络上实现的基于密度的异步层次聚类算法，并且要求被聚类数据的空间与 CAN 的路由空间一致 ;它首先是让每个节点对本地的数据进行聚类,然后按照 CAN 划分路由空间的逆过程合并聚类，最后得到全局的聚类结果。

虽然并行聚类算法和 C\S 系统的聚类算法不能直接应用到 P2P 系统中，但是它们都为设计 P2P 系统的聚类算法提供了许多有益的思路。在 P2P 系统中的数据挖掘算法应该是不依赖中心节点处理数据，不依赖洪泛方式传播消息，并且能够得到近似集中式数据挖掘算法的结果的“ 轻量级 “ 算法。

在并行聚类算法中，需要把整个聚类任务划分为许多子任务，由每个参加运算的节点独立完成，然后通过节点间交互信息共同完成整个聚类过程；而在 C\S 系统的聚类算法中，为了节约网络带宽，每个 Client 节点用局部聚类特征信息代替原始数据集合，

① 收稿时间:2008-07-17

Server 节点利用这些从 Client 节点接收的特征信息完成全局聚类。

3 结构化拓扑层次聚类算法

在这一节中，我们将设计一种基于结构化拓扑系统 CAN 的层次聚类算法 SOC(Structure Overlay Clustering)。SOC 算法包括 3 个部分组成：

(1) 层次化结构：在结构化 P2P 系统的拓扑基础上，设计一个虚拟的层次结构用于扩张聚类和合并聚类结果。

(2) 聚类边缘扩张规则：用于决定单个节点上的聚类在什么情况需要扩张到相邻的节点上。

(3) 聚类结果合并规则：确定如何合并两个聚类，并确定聚类的编号。

3.1 层次化结构

通过研究 CAN 系统的拓扑结构，我们设计了一种可以用于聚类合并的层次结构 PVtree（Peer Virtual Tree，节点虚拟树）。以二维的 CAN 系统为例，我们可以将 CAN 的拓扑结构转化为一种二进制编码的树型结构，如图 1 所示:

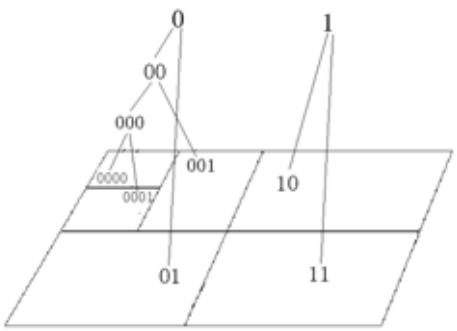

图 1　PVTree 结构示意图

树根节点代表整个空间（由于系统的坐标空间和系统内容的数据空间是一致的，后面用空间同时代表这两者），并且将每个子区域都用二进制编码表示。编码的规则是，设原始区域的编码是 X，则将区域 X 一分为二以后得到的两个子区域的编码分别是以原始区域编码为前缀加上 0 和 1 的后缀。原始区域有 0 和 1 两个子区域组成。

定义 1 如果区域 A 是由区域 B 分裂得到的，那么区域 A 叫做区域 B 的子区域，区域 B 是区域 A 的父区域。如果区域 A 是由区域 B 分裂第一次就得到的，那么区域 A 就叫做区域 B 的直接子区域，区域 B 是区域 A 的直接父区域。

定义 2 如果区域 A 和区域 B 的编码只有最后一位不同，则称区域 A 和区域 B 为兄弟区域。

在 PVTree 中所有的 Peer 都是叶子节点，并且只要将所有的叶子节点代表的区域与他们的兄弟区域合并就能够得到他们的直接父区域，这些区域在 CAN 系统中并不存在，但是通过子区域编号保存，用于接收子区域聚类的结果。

3.2 聚类边缘扩张规则

SOC 算法开始前需要在每个节点做局部聚类，将节点上的数据划分为几个初始簇。初始的局部聚类算法可以选择任何一种集中式聚类算法，在本文中使用的是相似度阈值为 ε 的 k-NN 算法。

定义 3 当数据距离区域所有边界均大于 ε 时，称这个数据为内部数据；当数据距离区域某个边界小于 ε 时，称这个数据为边沿数据。如图 2 所示:

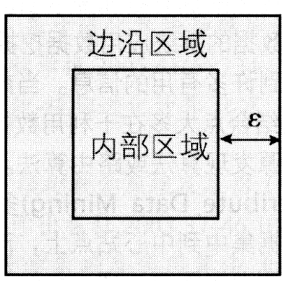

图 2　边沿区域示意图

引理 1 设 D 表示全体数据对象集合，D_i 表示区域 i 上的数据对象集合，D_{i_in} 表示区域 i 上的内部数据对象；对于区域 i 上的一个簇 C，如果对于 $\forall p \in C : p \in D_{i\_in}$ 则簇 C 是只属于区域 i 的聚类，即簇 C 是不可以扩张的。

引理 2 设 D_{i_out} 表示区域 i 上的边沿数据对象；对于区域 i 上的一个簇 C，C 是可以扩张的必要条件是 $\exists p \in C$，有 $p \in D_{i\_out}$。

引理 3 在采用平均链接度量时，设 D_{i_out} 表示区域 i 上的边沿数据，c 表示聚类 C 的质心，C 是可以扩张的必要条件是 $c \in D_{i\_out}$。

通过引理 3 我们只需要计算所有聚类的中心，并

且在属于边沿数据的中心里寻找距离满足相似度 ε 要求的数据，如果存在，那么这两个中心代表的簇就是可以合并的簇。

3.3 聚类结果合并规则

聚类结果合并规则主要讨论如何将各个区域产生的簇最终合并成为全局聚类结果。在讨论如何合并局部的簇之前，我们需要确定什么样的区域是可以合并的。

首先，能够合并的区域一定是相邻的区域，不相邻的区域一定不能直接合并，需要通过中间区域间接合并；其次，只要是相邻区域的数据，在满足聚类边缘扩张规则的情况下就是可以合并的，合并的结果与合并区域的顺序无关。如图 3 所示的数据分布分区情况，C_1、C_2、C_3 分别是区域 A、B_1、B_2 中的三个簇，p_1、p_2、p_3 分别是簇 C_1、C_2、C_3 的代表点。

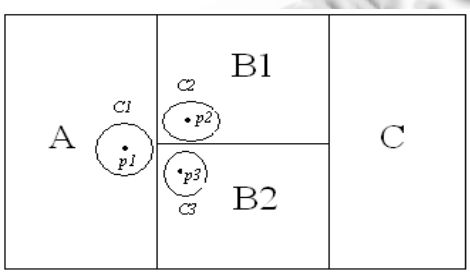

图 3　区域合并示意图

如图 3 所示，由于区域 A 和 C 不直接相邻，所以区域 A 和 C 不能直接合并，如果 A 和 C 上有可以合并的簇，那么这个簇一定可以通过 B1、B2 上的簇间接合并。

我们可以得到一些性质：

性质 1 两个区域的合并操作∪只在两个相邻区域间才有意义。

性质 2 两个区域的合并操作∪符合交换率律和结合律。

聚类合并规则中一个重要内容就是聚类的命名。为了使聚类在全局使用统一的名字，SOC 算法采用自底向上的命名原则，按照节点虚拟树结构，按照树生成结构的逆顺序逐步合并兄弟节点，通过合并产生的新簇由“较年长”的簇命名，参加合并的另一个区域向它的子区域发送改名的消息。所谓“较年长”是指由较多次合并操作生成的簇，这样做的目的是为了减少系统发送的改名消息。

兄弟节点合并后成生的虚拟父节点的信息记录在哪个节点上有两种选择策略：

策略 1 将这些信息存放在已经记录较多信息的节点（类似上一段中说的“较年长”的节点），这样做也是为了减少系统节点间传递的信息数；

策略 2 将信息存放在编号的中含有偶数个 1 且没有存放过虚拟信息的子节点上，这样做是为了平衡系统中各个节点的存储负载。

算法　1　结构拓扑层次聚类算法

输入：本地数据及兄弟节点的信息

输出：本地聚类的结果

步骤：

(1) 本地聚类，产生本地的初始簇；

(2) 通过自己的虚拟编号计算自己的兄弟节点，并且计算根据引理 2 确定可以扩张的簇；

(3) 合并满足 3.3 节中的簇合并条件的簇，并根据策略 1 或策略 2 的规则存储区域合并后生成的虚拟父节点的信息存储；

(4) 簇命名与虚拟父节点的那个节点向自己子节点广播簇改名消息；

(5) 存储虚拟父节点信息的节点如果不是树根节点则返回步骤(2)；

(6) 任何节点收到父节点的簇改名消息都向子结点发送簇改名消息。

图 4　结构拓扑层次聚类算法

图 4 所示是结构化拓扑层次聚类算法的基本流程，每个节点在完成初始簇的划分后便开始与自己的兄弟节点合并，在算法的步骤（2）中采用不同的虚拟父节点存放信息会使每个节点的工作量有所不同。当采用策略 1 时，由于可能存在越是“年长”的节点在吸收“年轻”节点信息后便的更加“年长”，导致一部分节点要一直参加合并计算并且保存虚拟父节点的信息，但是这种策略保证了有簇名修改时需要通知较少的节点；当采用策略 2 时，算法有意识的将父节点信息和合并计算均摊到系统中的所有节点上，但是这种均摊没有考虑到簇名修改时会有多少节点需要被通知。当算法结束时所有节点上簇的名字在全局是统一的。

4　实验结果

实验数据采用随机分布在 10000*10000 的二维区域上点，节点编号也是随机分布在相同数据区域上。实验中我们采用平均链接作为簇相似度的度量。

根据性质 1 和性质 2 我们知道 SOC 算法可以达到与集中式聚类算法同样的效果，所以在实验中我们主要关注算法对网络的负载。如图 5 所示是在数据量一定(10000 个随机数据)的情况下不同节点数量的结果，说明构拓扑层次聚类算法对网络的负载主要与系统中节点的数量有关，图 6 所示的是相同节点数（400 个节点）而不同数据量的结果，说明构拓扑层次聚类算法对网络的负载与数据量的多少没有直接关系。

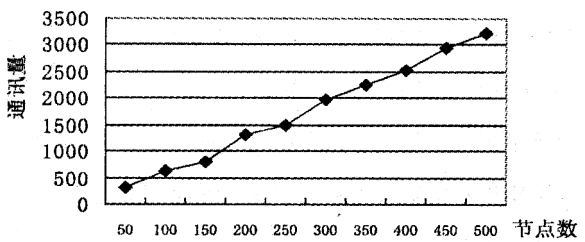

图 5　数据量一定的情况下不同节点数的网络负载

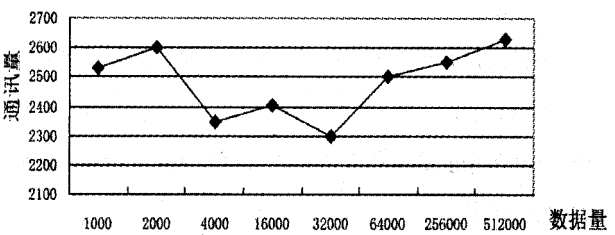

图 6　节点数一定的情况下不同数据量的网络负载

当数据维数不同时，拓扑层次聚类算法对网络的负载也会不同，当有 10000 个随机数据和 400 节点时，通讯量随数据维数变化如图 7 所示。

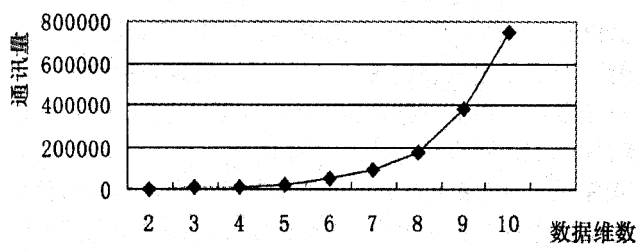

图 7　数据维数对网络负载的影响

通过分析拓扑层次聚类算法的聚类生成过程我们可以计算算法的复杂度。聚类的操作时间复杂度就是聚类/簇合并操作+簇名改变操作=$(2^{(k-1)}+1)*N+(k*N-1)/(k-1)-N=O(2^kN)$。也就是说在数据维数一定的情况下，算法的时间复杂度是 O(N)，也就是说与节点数量成正比。

5　结论

SOC 算法利用结构化 CAN 系统数据严格按照规律分布到节点的特点，将系统坐标空间与聚类数据空间重叠，使相邻数据落在同一(或相邻)节点上。通过系统坐标空间划分的逆过程层次聚类，使 CAN 系统中的聚类能够达与集中式聚类相同的结果。最后通过理论分析计算出算法的复杂度并通过仿真实验验证。

参考文献

1 Sylvia R,Paul F,Mark H,Richard K,Scott S.A scalable content-addressable network . proceedings of the 2001 conference on Applications,technologies,architectures,and protocols for computer communications.California, United States 2001:161–172.

2 DHILLON I S, MODHA D S. A data-clustering algorithm on distributed memory multiprocessor. Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems, London: Springer-Verlag,1999.

3 FORMAN G, ZHANG. B. Distributed data clustering can be efficient and exact. ACM SIGKDD Explorations Newsletter, 2000,2(2):34–38.

4 JOHNSONEL, KARGUPTA H.Collective,hierarchical clustering from distributed, heterogeneous data. Large-Scalle Parallel KDD System, London: Springer-Verlag ,1999.

5 BANDYOPADHYAY S, GIANELLA C, MAULIK U, et al. Clustering Distributed Data Streams in Peer-to-Peer Environment.Information Science Journal, 2005, 176(14):1952–1985.

6 Lee Mei, Lee Guanling, Lee Wang Chien. PENS: An Algorithm for Density-Based Clustering in Peer-to-Peer Systems.Proceedings of the 1st international conference on Scalable information systems, New York: ACM Press, 2006.