基于向量的并行关联规则挖掘算法①

A Parallel Algorithm for Mining Association Rules Based on Vector

何中胜　(常州工学院 计算机信息工程学院 江苏 常州 213002)

摘　要：向量处理是计算机的优势，将事务数据库转换为向量矩阵，并对其进行关联规则挖掘，不仅可以充分发挥计算机的优势，而且提高挖掘的效率。基于此思想提出了一种在基于无共享体系结构的计算机上高效并行关联规则算法，详细描述了该算法的思想及实现步骤。最后实现该算法并与 CD 算法进行了比较，实验结果证明该算法效率较 CD 算法高，且具有很好的可扩充性，同时保证了挖掘结果的完整。

关键词：向量处理 关联规则 并行挖掘 可扩性

1　引言

现实世界中事情的发生多多少少有一些关联。一件事情的发生，很可能会引起另外一件事情的发生。即这两件事情很多时候很大程度上会一起发生的。那么人们通过发现这个关联的规则，可以由一件事情的发生，来推测另外一件事情的发生，从而能更好地了解和掌握事物的发展、动向等等。这就是数据挖掘技术中，挖掘关联规则的基本意义。目前关联规则的应用主要包括顾客购物分析、目录设计、商品广告、邮寄分析、销售仓储规划及网络故障诊断分析等[1-3]。

经典的关联规则挖掘算法主要有 Agrawal 等提出的基于 Apriori 算法的频集方法,以及基于 FP 树的模式增长的无候选集生成的挖掘算法等，但随着数据容量的增加，在单机上进行挖掘的效率很低，为了提高关联规则的挖掘效率,研究人员又提出了并行挖掘算法[4],主要包括:Agrawal 等人提出的 CD (CountDistri2bution)、CaD (CandidateDistribution)、DD (DataDistribution)算法和 Park 等人提出的 PDM 算法。Chueng 等人提出了 DMA 和 DM 算法(这两种算法尽管是基于分布式数据库的,但同样适应于并行挖掘)。这些算法虽然具有速度快、容易实现、要求各计算机间同步次数较少等优点,但仍然存在着可扩性较差、通讯量大、候选项集多、规则合成难度高等缺点。本文针对上述问题提出了一种在微机集群上实现的挖掘关联规则的高效并行算法(简称 PMARBV 算法)。

2　关联规则挖掘形式化描述

2.1 顺序关联规则描述

顺序关联规则的采掘问题可形式化描述如下[4,5]:设 I= {i1，i2,…，im}是由 m 个不同的项目组成的集合。给定一个事务数据库 D，其中的每一个事务 T 是 I 中一组项目的集合，即 T⊆I，T 有一个唯一的标识符 TID。若项集 X⊆I 且 X⊆T，则事务 T 包含项集 X。一条关联规则就是形如 X⇒Y 的蕴涵式，其中 X⊆I,Y⊆ I,X∩Y=∅。关联规则 X⇒Y 成立的条件是:①它具有支持度 s,即事务数据库 D 中至少有 s%的事务包含 X ∪Y;②它具有置信度 c，即在事务数据库 D 中包含 X 的事务至少有 c%同时也包含 Y。顺序关联规则的挖掘问题就是在事务数据库 D 中找出具有用户给定的最小支持度 minsup 和最小置信度 minconf 的关联规则。

顺序采掘关联规则问题可以分解为以下两个子问题[4,5]:

(1) 找出存在于事务数据库中的所有频繁项集。若项集 X 的支持度 support(X)不小于用户给定的最小支持度 minsup,则称 X 为频繁项集 Frequent itemset).

(2) 利用频繁项集生成关联规则。对于每个频繁

① 基金项目:江苏省教育厅基金项目(04kjd520010)
收稿时间:2008-09-03

项集 A ,若 B⊂A ,B≠∅,且 support(A)/ support(B) ≥minconf，则有关联规则 B⇒(A-B)。

对于第二个子问题Agrawal等人已给出了较好的解决办法[5]，关联规则挖掘算法的性能主要集中在第一个子问题上，大部分的算法主要集中在如何高效地发现频繁项集上。

2.2 并行关联规则问题描述

现今已有多种发现频繁项集的串行算法，通常的串行算法都是首先生成候选项目集，然后计算它们的支持度从而生成频繁项目集。这样，频繁项目集生成所需的时间和空间开销往往很大，而且大规模的数据库一般都很大(按 GB 乃至 TB 计)，而随着高性能计算机的出现，采用并行数据挖掘算法不仅可行而且能提高效率。因此，并行数据挖掘是一个重要的研究方向[6]。

并行挖掘关联规则问题可形式化地描述如下[7]：设 Pl，P2 ，…，Pn 为 N 台基于无共享体系结构的计算机，即它们之间除了通过网络传递信息外，其它资源(如硬盘、内存等)全部是独立的。DBi (i=l，2，…，n)是存储于计算机 Pi 硬 盘上的分事务数据库，其中的事务有 Di 条，则总的事务数据库为 $DB=\bigcup_{i=1}^{n} DBi$ ，总的事务条数为 $D=\sum_{i=1}^{n} Di$ 。并行挖掘关联规则问题就是如何通过 N 台计算机同时工作，计算机 Pi(i=1，2，…， n)只处理自己的私有数据库 DBi(i=l，2，…，n)，各台计算机间仅通过网络传递有限的信息，最终在整个事务数据库 DB 中挖掘出关联规则。

并行挖掘关联规则问题也可以像顺序挖掘关联规则问题一样分解为两个子问题。目前已经提出的并行采掘关联规则的算法有 Agrawal 等人提出的 CD(CountDistribution)、CaD(CandidateDistribution)、DD(Data distribution)，Park 等人提出的 PDM 等算法，算法 DMA 虽然是基于分布式数据库的挖掘算法，但也可适用于并行挖掘。 这些并行算法都有局限性，主要的局限性在于：它们都重复扫描驻留在磁盘上的数据库分区，这就导致了大量 I / 0 开 销。此外，这些算法因涉及到在每次迭代过程中产生大量候选项集并且需要对它们进行计数或远程数据库分区的交换，导致了大量的计算、通信和同步性的开销。基于此，就需要更有效、实用的算法来解决这类问题。文[8]利用矩阵理论中上三角矩阵的良好性质,通过数据库约简、投影等操作，在 N 台机器上开展并行挖掘，从而提高挖掘算法的效率和可扩性。但该算法存在最大的不足就是舍弃了许多潜在的频繁项集，本文受此启发，得出了基于向量的并行挖掘算法(PMARBV 算法)。

3 基于向量的并行挖掘算法(PMARBV算法)

3.1 PMRRBV 算法思想

设一事务数据库 DB 共有 M 个项目，含有 N 条事务，通常结构如表 1 所示：

表 1　含有 5 条事务、8 个项目的事务所

TID	Items
T01	fgh
T02	h
T03	defgh
T04	cdefgh
T05	abcdefgh

现将其转换为如图 1 所示的 5×8 布尔矩阵，并通过整理为上三角形矩阵：

$$A=\begin{array}{c} \begin{matrix} a & b & c & d & e & f & g & h \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

图 1　数据库 DB 对应的布尔矩阵 A

其中，A[i,j]为 1 表示事务 Ti 包含项目 j，为 0 表示不包含项目 j。

实际上向量处理是计算机的优势，将事务数据库转换为向量矩阵，可以充分发挥计算机的优势。利用如上的上三角矩阵不仅能够有效地发现基于单属性关联规则,而且对多属性关联规则也非常奏效。如从矩阵 A 中可以方便地计算出：

sup(f ⇒ e)= 0. 6,conf(f⇒e)=0.75；

sup(g⇒e)= 0. 6,conf(g⇒e)=0.75；

sup(h⇒e)= 0. 6,conf(h⇒e)=0.6；

sup(d⇒e)= 0. 6,conf(d⇒e)=1.0；

如果设定 minsup= 0. 5,minconf=0.7，可以得到如下的关联规则:

f ⇒e； g⇒e； d⇒e

从上面例子可看出对于项集支持度的计算完全可转换成对矩阵 A 的相应列进行布尔运算，本算法正是将上述原理应用到集群式计算机中，设有计算机 n 台，将原来的大规模数据库 DB 分割成 p 个规模较小的数

据库 DBi(i=1,2…n)，每个 DBi 中的事务个数约为|DB|/n。并分配到相应的计算机 Pi 上。然后在每台计算机上对各自的数据库 DBi 进行关联规则挖掘，先挖掘出局部频繁项集，再经过通讯产生全局频繁项集，最后产生全局的关联规则。

3.2 PMARBV 算法算法步骤

整个算法分为三个阶段，阶段一进行数据库的预处理：数据整理及属性约简。阶段二进行关联规则挖掘：各计算机对各自的局部数据库进行挖掘，产生局部频繁项集，经过通讯产生最终的全局的频繁项集；阶段三根据频繁项集生成关联规则。下面对各阶段的内容进行详细说明。

第一阶段进行数据库的预处理，包括数据库整理和属性约简，最终形成如图 2 所示的形式的上三角矩阵。设事务数据库 DB 共有 M 个项目，事务个数记为|DB|，局部数据库 DBi 的事务个数记为|DBi|。

步骤 1：将整个数据库 DB 进行水平分割成 n 个局部数据库 DBi，n 为计算机的个数。考虑到各计算机的负载均衡，直观上看每个 DBi 的事务个数为|DB|/n 较合理。这样原数据库被分割成 n 个规模为[|DB|/n]的局部数据库 DBi(i=1,2,…n)。

步骤 2：将各个局部数据库 DBi 传送到计算机 Pi(i=1,2,…n)。

步骤 3：在每台计算机 Pi 上，对局部数据库 DBi 进行如下处理：将 DBi 转换成|DBi|×M 的布尔矩阵 A，计算 $Si=\sum_{j=1}^{M}|aij|$ ，其中 aij 是数据库 DBi 中第 i 个元组第 j 个属性值，|aij|是对 aij 取绝对值运算，M 同上。若 Si 为 0，表明该行的信息量为 0，丢弃该行；若 Si 不为 0，则依 Si 值按降序对矩阵 A 自上而下进行排列，产生新数据库 DBi’，对应的新的布尔矩阵记为 A’；对 A’，计算 $Tj=\sum_{i=1}^{N}|a'ij|$ ，其中 N 为|DBi’|。若 Tj 为 0，表明列的信息量为 0，丢弃该列； 若不为 0，则依 Tj 值按升序对矩阵 A’自左向右进行排列，产生新的数据库 DBi*，对应的新的布尔矩阵为 A*，这一步的目的是产生上三角矩阵。

下面进行第二阶段的工作，即各计算机 Pi 对各自的局部数据库(经过整理约简后的 DBi*，对应的布尔矩阵为 A*)进行挖掘，产生局部频繁项集，经过通讯产生最终的全局的频繁项集，具体步骤如下：

步骤 1：在计算机 Pi 上，对于布尔矩阵 A*，计算，且恰为第 j 个项的局部支持度，即为 Ij.supi，得出候选局部频繁 1 项集的集合，记为 CLFi1。各个计算机之间广播交换 CLFi1 及局部支持度以生成全局支持度，根据最小支持度 minsup，以生成全局频繁 1 项集 GF1。

步骤 2：在计算机 Pi 上，仅对 GF1 中的频繁 1 项，考察两个频繁 1 项的组合即 2 项集，如 Ij 和 Ik 为频繁 1 项，则考察 2 项集｛Ij，Ik｝(不妨记为 X2)的支持度，且 $X_2.supi=\sum_{i=1}^{N}aij*aik$ ，其中 N 为|DBi*|。因此，得出候选局部频繁 2 项集的集合，记为 CLFi2。各个计算机交换 CLFi2，再根据最小支持度 minsup 以生成全局频繁 2 项集 GF2。依此类推，各计算机 Pi 对全局频繁 K-1 项集 GFk-1 采用文[1]中的连接函数产生候选局部频繁 k 项集的集合，记为 CLFik。对每个候选 k 项集，不妨设为{Ij1,Ij2,…,Ijk},计算它的支持度为 $\sum_{i=1}^{N}\prod_{j=j1}^{jk}a_{ij}$ ，然后各计算机之间通过通讯广播各候选频繁 K 项集的支持度以生成各自的全局支持度，再根据最小支持度 minsup 生成全局频繁 K 项集 GFk。

步骤 3：生成全局频繁项集 $GF=\bigcup_{i=1}^{n}GFi$ 。

最后进入第三阶段，根据全局频繁项集 GF，产生关联规则，这一步的方法已较成熟，这里不再赘述。

在第二阶段中为保证频繁项集的不丢失，采取了计算机间交换候选局部频繁项集的支持数，看起来通讯量较大，因此在实现过程中计算机间采取主从式，各从结点将各基项的支持度传送到主结点上，由主结点进行统一收集并计算出各项集的全局支持数，再采取广播方式传送到各从结点上，再由各从结点进行计算出全局频繁 K 项集。

4 实验结果分析与性能评价

根据对以上算法步骤的分析与研究，发现 PMARBV 算法将传统频繁项集发现的算法关键转移到对数据库的整理与项目约简上，而对各 K 项集的支持度的计算显得尤为简便，无需对数据库进行多趟扫描，同时候选项目集的生成及支持度计算非常简便，其时间复杂性为 O(N)，整体时间复杂性约为 O(M3N)，M 为项目数，N 为 DB 的记录条数。

根据以上算法思想，采用 C+ + 和 MPI 库实现了 PMARBV 算法，并在 4 台联想计算机(P41.7G，256M 内存)上进行实验，采用的数据库是来自某电梯公司的电梯维护维修数据，约 10 万条。同时在单机上对同样的数据库采用 Apriori 算法进行实验，结果发现并

行算法与串行算法所产生的频繁项集是相同的，这证明了 PMARBV 算法结果的正确性。同时对相同数据库采用 CD 算法(一种基于 Apriori 算法的并行关联规则算法)也在相同的 4 台机器上进行实验，发现所产生的频繁项集相同，但是 PMARBV 算法的运行时间效率比 CD 算法少，而且当支持度 minsup 参数逐渐变小时，PMARBV 算法和 CD 算法运行时间变化曲线都相应增大，但是明显 PMARBV 算法比 CD 算法增加得慢，这说明 PMARBV 算法比 CD 算法效率高。主要原因还是在于 PMARBV 算法充分利用向量计算的优势，它只扫描事务数据库一次且直接通过布尔运算计算项集的支持度，比经典的 Apriori 算法占用内存空间小，执行速度快。

参考文献

1 Agrawal R, Srikant R. Fast algorithm formining associationrules. Proc of 20th International Conference Management of Very Large Database. Santiago, Chile:[s.n.], 1994:487－499.

2 铁治欣,陈奇,俞瑞钊.关联规则采掘综述.计算机应用研究,2000,32(1):125.

3 李新征.一种新的高效 Apriori 算法.微计算机信息,2006,22(3):193－194.

4 铁治欣,陈奇,俞瑞钊.采掘关联规则的高效并行算法.计算机研究与发展,1999,36(8):948－953.

5 Hwang K, Xu Z. Scalable Parallel Computing: Technol-ogy, Architecture, Programming. NY: WCBMcGraw2Hill, 1998:293－297.

6 Cheung DW. Efficient mining of association rules in dis2tributed databases. IEEE Transactions on Knowledge and Data Engineering, 1996,8(6):910－921.

7 何中胜,刘宗田. 一种无候选集产生的并行关联规则挖掘算法.计算机工程与应用, 2004,24:163－165.

8 杨欣斌,孙京诰,陈霁威,等.一种高效并行关联规则挖掘新算法.华东理工大学学报, 2003,29(3):295－298.