分布式数据挖掘计算过程
——DDCP算法研究

方英武*[1,2] 张广鹏[1] 吴德伟[2] 黄玉美[1] 赵修斌[2] 王 轶[2]

(1. 西安理工大学机械与精密仪器工程学院 西安 710048; 2. 空军工程大学电讯工程学院 西安 710077)

【摘要】提出了一种关联规则挖掘大项集生成的并行和分布式处理的计算框架的算法，该算法以大规模事务数据库为基础，将数据有效地分片后作分布或者并行处理，通过节点之间的通信降低了节点间传输的数据量。通过算法实例验证了算法的正确性和可行性，可以在分布式或者并行环境里实现高效的数据挖掘。

关 键 词 数据挖掘; 关联规则; 大项集; 数据库

中图分类号 TP311.5 文献标识码 A

Research on Distributive Datamining Calculating Process
——DDCP Algorithm

Fang Yingwu[1,2] Zhang Guangpeng[1] Wu Dewei[2] Huang Yumei[1] Zhao Xiubing[2] Wang Yi[2]

(1. College of Mechanical and Precision Tool Engineering, Xi' an Univ. of Technology Xi' an 710048;

2. The Telecommunication Engineering Institure, Air Force Engineering Univ. Xi' an 710077)

Abstract This article proposed a algorithm of the calculate architecture used for the association rule and this algorithm based on the data partition, fully uses the merits and specialties, at the same time uses controller to assign transactions randomly to resolve the data skew in the database. The algorithm is used for the example and shows the correctness and feasibility. It can be used for distribute database and most applicable for distribute calculation.

Key words datamining; association rule; large itemset; database

目前数据挖掘的算法很多[1]，但这些算法都是针对特定的问题和应用领域，在有些方面是高效的，但都存在或多或少的缺陷。主要问题在于每一种方法都是对具体的计算方法的研究，在改善以往算法时却牺牲了一些以往算法的优点。大规模数据库的关联规则挖掘算法的效率瓶颈是大项集的生成过程，这个过程相当耗时，故所有的算法都针对这一点进行了研究和分析，提出了各种不同技术的算法，其目的是尽量减少数据库的扫描次数。本文通过深入分析以往算法的优缺点[2]，提出了一种关联规则挖掘大项集生成的并行和分布式处理的计算框架的算法—分布式数据挖掘计算过程(Distributive Datamining Calculating Process，DDCP)算法。旨在能够提供一个灵活的和可扩展的计算平台，利用现在相对廉价的单机进行网络计算，充分挖掘网络计算的优势。

1 分布式计算框架

1.1 问题描述

关联规则挖掘问题是在分析零售业事务数据库时提出的，现在的发展已经超出了原来的应用范围，其

2002年7月10日收稿

* 男 30岁 博士研究生 主要从事计算力学与自动控制方面的研究

深度和广度都有很大提高，但关联规则的形式化描述有其通用意义，本文即采取这种形式化的描述方法。

关联规则的发现可以分解为两个子问题：

1) 找出存在于事务数据库中的所有大项集。项集I的支持度support(I)≥minsup，则称X为大项集或者称为频繁项集；

2) 利用大项集生成关联规则。对每个大项集A，若$B\subset A$，$B\neq\varnothing$，且support(A)/support(B)≥minconf，则有关联规则：$B\Rightarrow(A-B)$。

问题2)较容易解决，已有成熟的生成算法，问题1)的解决影响大规模数据库的检索，所以效率和准确性是问题的关键，讨论和算法都是集中在不牺牲精度的前提下提高大项集生成效率上，本文算法和体系结构将基于Apriori和Partition 对这一问题进行分析。

1.2 分布式计算框架

分布式计算框架利用了Partition数据库分片的思想，但是各个部分具体的算法不是固定的，在不同的部分使用不同的算法，该计算框架可以应用在并行和分布式的环境里。其计算框架如图1所示。图中：DU表示分布单元(Distribute Control and Management Unit)，ICMU表示信息控制管理单元(Information Control and Management Unit)，TRDU表示事务读取分发单元(Transaction Read and Distribute Unit)，TDB表示事务数据库(Transaction Database)，C_1～C_n表示数据传道(Data Transfer Channel)，P_1～P_n表示处理器(紧耦合)或树(Local Set Enumerate Tree)，GT表示全局集合枚举树(Global Set Enumerate Tree)或其他数据结构。

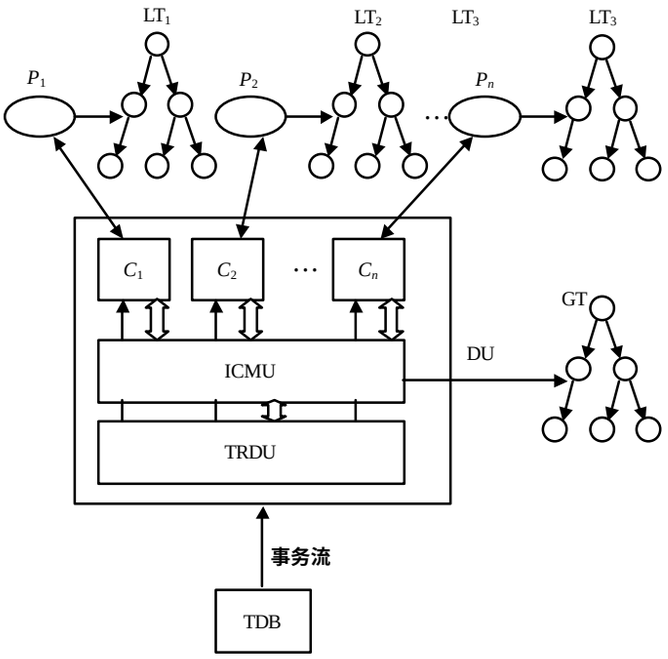


图1　分布式计算框架图

1.3 分布式计算流程

分布式计算框架从概念上来讲是一种层次计算方法，将整个数据挖掘大项集的生成算法中涉及的各个相对独立的计算过程分离出来，利用单独的模块进行计算，因为各个模块之间没有紧密的耦合现象，相互之间只是事务数据的传递和少量的指令流。

该计算框架的核心是分布单元DU，其具体的流程可以分四步进行：

1) TRDU首先根据处理器或者可利用的分布式单机的数目创建数据传输通道C_1～C_n，负责各种初始化工作；

2) 启动函数Decide_Receive_Node()，根据所使用的解决数据偏度策略的不同初始化变量，顺序读取数据库中的事务块，将每一个事务分配到不同的处理器，做到负载均衡和解决数据偏度；

3) P_i各自处理自己的事务，如果所有事务可以放入内存，则选择高效的算法生成本地大项集；否则将事务缓存到本地磁盘，在所有事务从TRDU接受完毕后，生成最后的本地大项集；

4) ICMU负责和各个节点之间的数据通信，同时维护全局枚举树GT。各节点在处理过程中或者处理完成后都可以和ICMU通信，这取决于不同的实现策略。

2 DDCP算法

2.1 DDCP算法步骤

基于分布式计算框架，本文提出了DDCP算法来实现框架的计算思想，算法分为控制节点和分节点两个部分。

控制节点的算法包括三个阶段：

1) TRDU的初始化。进行必要的全局信息收集和相应变量的初始化，同时通知各个节点全局的信息；

2) TRDU的事务分布。决定读入的每一个事务应该分配给哪一个节点进行处理，纪录每一个节点得到的事务总数，根据在初始化阶段得到的各个节点可以分配的事务数目控制是否在下一个事务的分配决策中包含该节点；

3) ICMU的事务处理。根据TRDU读取数据的状态决定事务处理所处的阶段，当得知所有的事务已经被TRDU读取结束后，各个节点就得到了它们应该处理的全部事务，因此可以得到本地的大项集，此时ICMU处于等待状态，每当一个节点完成后就通知ICMU，同时将本地的大项集传递给ICMU，ICMU动态的合并所有的本地大项集，最终输出全局大项集。

分节点的算法也包括三个阶段：

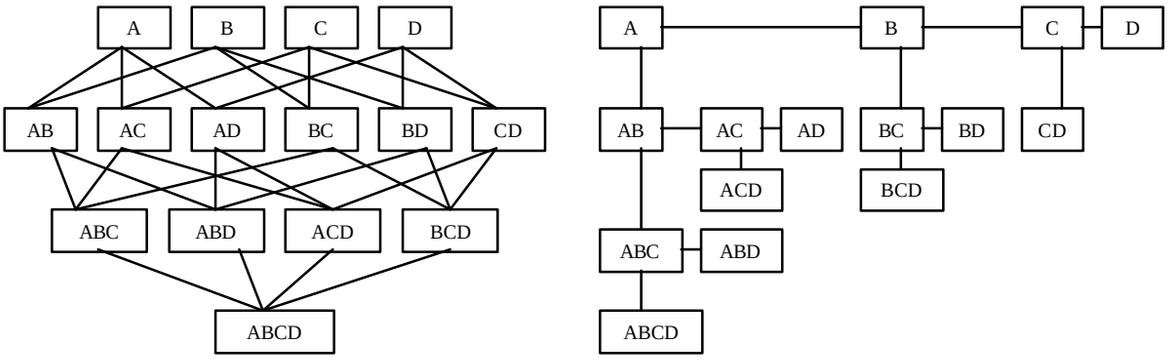
1) 从TRDU得到全局的分布信息，初始化自身的变量，包括将自身节点同分配的通道相绑定；

2) 连续接受中央节点传递的事务，同时负责清除通道的数据为下一个事务的接收做准备。分节点根据是否可以将全部事务放入内存执行来决定是否放入内存处理，如果不可以放入内存执行则利用动态的事务处理或者缓存到本地的磁盘；

3) 在得到了TRDU发送的明确的事务分发结束信号后，如果不适用动态的集合枚举树生成方法则开始处理所有得到的事务，处理结束后将得到的大项集传递给中央节点的ICMU单元。

2.2 有序集合枚举树的动态生成

有序集合枚举树是一种集合枚举方法，集合中的所有元素是按字母排序的。集合{ABCD}形成的格和集合枚举树如图2所示，集合枚举树同格相比降低了图的复杂度，更有利于图的遍历和裁剪。



格　　有序集合枚举树

图2　四阶集合枚举树和格

集合枚举数同格相比虽然具有相同的节点数目，但是节点之间有很明确的顺序关系，如果对集合枚举数进行先序遍历就可以得到项集所有子集的按照字母顺序从小到大的列表，同时该枚举数是一棵平衡的二叉树，所以有效地降低了数据高度。对于具有n个项的项集I，树的高度就是n层，定位一个k-项集最多需要遍历k层。根据枚举树的特点可以得到：父节点都是其子节点的子集；定位一个项集所经过的节点包含了所有排在该项集之前的该项集的所有子集；一个节点代表的项集是该节点所有子节点的公共部分。因为一般项集I中包含的项很多，所以构造这样的一棵完全的有序集合枚举树是不可能的，因此这样的一棵集合枚举树可以遵循一定的原则根据需要动态的创建。对于每一个事务中的项集在这样虚拟的枚举树中定位的过程将会经过在该项集前面的所有该项集的子集，树中节点的支持度每当有事务扫描就加1，这样就可以在事务的扫描中收集尽可能多的信息，最终用一棵不完全的有序树代替整个数据库的事务，减小扫描整个事务数据库的开销[3]。从树中每一个节点的支持度的角度来看，这个支持度是所有包含该节点而且排在该节点后面的数据库中事务项集的个数，因此树构造完成后得到真正支持度的过程就是找出排在该节点前面的包含该节点的数据库事务项集的个数。

设节点集合为N，构造树过程中得到的支持度表示为$\mathrm{support}^{+}(N)$，需要计算的剩余的支持度为

support$^-$(N)，则：

$$\text{support}^+(N)=\sum num(t)(t\in D，\text{itemset}(t)\supseteq N，\text{itemset}(t)\geq N)$$

$$\text{support}^-(N)=\sum num(t)\ (t\in D，\text{itemset}(t)\supseteq N，\text{itemset}(t)<N)；$$

$$\text{support}(N)=\text{support}^+(N)+\text{support}^-(N)$$

式中　$num(t)$表示数据库中t的个数，itemset(t)表示事务t的项集。

由于树是动态生成的，不可能对每一个事务产生所有的子集。所以当事务项集已经在树中存在时，简单的增加这个项集的支持度；当事务项集在树中不存在时，根据该项集与当前项集的关系决定具体的行为，这种情况将会插入代表该项集的新节点；如果两个项集的前k个项完全相同，说明它们有公共的子集，即父节点，则创建此k-项集的节点作为这两个父节点。算法在扫描完数据库时只能得到support$^+$(N)，因此后续的步骤就是计算support$^-$(N)。

3　算法实例

根据DDCP算法，下面给出算法实现的一个实例。算法测试所用数据根据文献[4]所给出的算法生成程序，该数据生成程序使用Apriori算法，算法中利用的数据结构主要包括：

1) 事务数据库的纪录结构(客户号、事务号、事务项数、项列表)；

2) 动态树的节点结构(项数目、项集支持度、项列表、父指针、子指针、左兄指针、右兄弟指针)；

3) 大项集文件结构(事务数、项集标志、总数、项集列表、……、项集标志、总数、项集列表、……)。

下面根据一个简单的事务数据库来描述算法的实际实现情况。

3.1　动态集合枚举树的生成

假设事务分配单元将一个纪录分配到一个节点P，P在接收节点的同时将纪录类型转换为一个树的节点类型，然后将该节点插入到已经生成的枚举树中。若其接收的所有纪录转换后成为以下的节点列表，如表1所示，则最终生成的树结构如图3所示。

表1　节点列表

项数目	5	4	4	3	5	4	3	6	4	3	4
支持度	0	0	0	0	0	0	0	0	0	0	0
项列表	ACDF	BCEL	ADHL	BCH	CDHKL	CDKL	HKL	AEFGKL	EGHK	AEF	ADHK

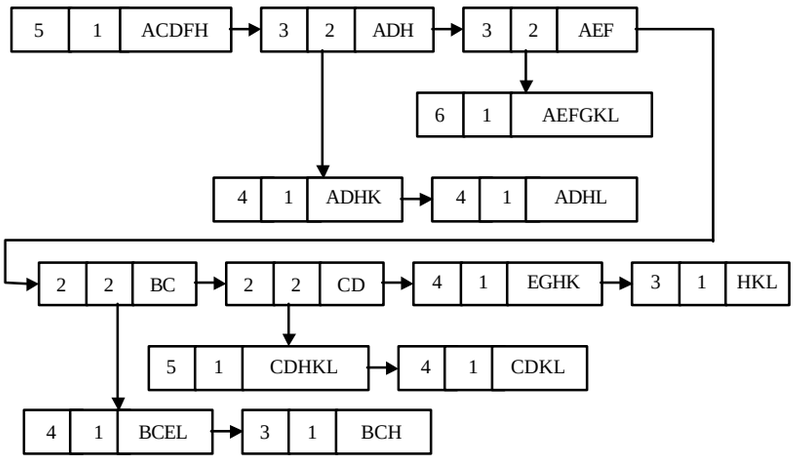


图3　树结构图

3.2　项集支持度的计算

由算法的描述，在节点得到每一个事务的过程中可以生成大–1项集，然后利用Apriori的候选项集生成算法得到候选大项集，再利用生成的枚举树得到项集的总支持度。如对于项集{AE}，当到达节点{AEF}后发现，{AE}⊂{AEF}，所以不用再搜索{AEF}的子树了，因为根据生成算法，该节点已经包含了所有其超集

的信息，此时得到{AE}的支持度为2，然后继续搜索其右子树，最终得到项集{AE}的总支持度为2。对项集{CD}，到达{ACDFH}时可知{CD}⊂{ACDFH}，所以{CD}的支持度为1，然后搜索{ACDFH}的右子树，到达{CD}，发现{CD}={CD}，于是{CD}的支持度为1+2=3，因为已经得到了相同的节点，所以停止搜索，最终{CD}的总支持度是3。

3.3 算法性能分析

算法实现分为两个相互独立的阶段。在事务分配阶段，由TRDU控制从数据库中读取事务流在度取得过程中可以直接得到大–1项集，因此这个过程同数据库中事务的数目是成线性关系的，即若事务数据库中事务数为n，则事务分配TRDU的时间复杂度为$O(n)$，空间复杂度为常量$O(1)$。然后是各个节点的本地处理过程，其中最重要的就是有序集合枚举树的动态生成和遍历的过程。假设总共有k个处理节点，因为事务是平均分配到各节点，因此每个事务得到的节点数为n/k，在假设网络带宽足够的前提下，认为各个节点都在相同的时间段内得到了所有本地事务。集合枚举树是一棵二叉数，因此一次二叉树的遍历时间复杂度为$O(n/k)$[4]，其空间复杂度即为遍历过程中间的最大容量，即树的深度。因为该二叉树不是平衡二叉树，所以最坏的情况下空间复杂度为$O(n)$，最好情况即为平衡二叉树，空间复杂度为$O([\log_2(n/k)]+1)$。二叉树的遍历分为创建和后来的搜索，即计算项集的支持度。在创建过程中需要遍历树n/k次，计算项集支持度时遍历的次数和由Apriori的候选大项集生成程序及最小支持度有关，假设所有的候选大项集个数为p，则整个算法的过程需要遍历二叉树$p+(n/k)$次，在各个节点生成本地大项集算法的时间复杂度是$O\{n/k[p+(n/k)]\}$。因此算法性能在假定各个子处理节点具有相同的处理能力并且忽略网络带宽限制时可得到：总时间复杂度为$O\{n+n/k[p+(n/k)]\}$，其中p是一个变数，它和数据库中事务的性质及所选择的最小支持度有关。在主节点即TRDU驻留的节点空间复杂度为一常量，在各个处理分解点在最坏的情况下空间复杂度为$O(n)$，最好的情况下为$O([\log_2(n/k)]+1)$。

4 结 论

DDCP算法利用了本地节点动态有序集合枚举树生成方法来代替数据库，节省了本地空间的占用，保证了在事务顺序流过本地节点后不间断地生成枚举树，处理过的事务可以抛弃掉，所有事务的信息都保存在了动态生成的树中，而这棵树的存储是远小于整个事务数目，因此可在内存中进行计算，从而可以明显提高效率。算例研究表明了算法的正确性和可行性。

参 考 文 献

1 胡 侃，夏绍玮. 基于大型数据仓库的数据采掘：研究综述[J]. 软件学报，1998，(1)：53-63

2 May R A. Mining association rules between sets of items in large database[C]. Proc. ACM SIGMOD int'l conf. Management of data, Washington DC. 1993，207-216

3 IBM Almaden Research Center. Quest Synthetic Data Generation Code[EB/OL]. http://www.almaden.ibm.com/cs/quest/syndata.html，1996

4 Goulbourne G，Coenen F，Leng P. Algorithms for computing association rules using a partial-support tree[J]. Knowledge-Based Systems，2000，(13)：141-149

编 辑 漆 蓉