频繁闭合项目集的并行挖掘算法研究

缪裕青
(桂林电子工业学院计算机系　桂林541004)

摘　要　频繁项目集挖掘因其在数据挖掘领域中的基础地位和广泛应用备受学术界和产业界的关注，用挖掘频繁闭合项目集代替挖掘频繁项目集是近年来提出的一个重要策略。不同于以往提出的挖掘所有频繁项目集的并行算法，本文针对频繁闭合项目集的特性及并行挖掘的特点，给出了共享存储器模型上(Shared Memory)基于频繁模式树(FP-tree)的挖掘频繁闭合项目集的并行算法(FCIPM)思想，提出了频繁闭合项目集直接判断法，性能分析表明所提技术对算法的性能提高起到了关键作用。

关键词　数据挖掘，频繁项目集，频繁闭合项目集，频繁模式树，并行算法

A Parallel Algorithm for Mining Frequent Closed Itemsets

MIAO Yu-Qing
(Department of Computer Science and Technology, Guilin University of Electronic Technology, Guilin 541004)

Abstract　Frequent itemsets mining has been studied extensively in data mining research. Because the huge number of frequent itemsets is found, an interesting alternative has been proposed recently, instead of mining the complete set of frequent itemsets, only find frequent closed itemsets, which has the same power as mining the complete set of frequent itemsets, but it will substantially reduce the number of frequent itemsets.

In this paper, we propose an efficient parallel algorithm, FCIPM (Frequent Closed Itemsets Parallel Mining), based on shared memory for mining frequent closed itemsets, with the development of three techniques: (1)A FP-tree based bottom-up no tree-projection method, (2)itemsets pruning, (3)frequent closed itemsets checking directly. Our performance study shows the advantage of these techniques and that the FCIPM may has good performance in terms of runtime, memory usage and scalability.

Keywords　Data mining, Frequent itemsets, Frequent closed itemsets, FP-tree, Parallel algorithm

1　引言

频繁项目集挖掘是数据挖掘领域中一个非常重要的研究课题，自从问题被提出[1]，就受到学术界和产业界的广泛关注，目前仍是国内外的一个研究热点。一个最著名的挖掘频繁项目集的算法是 Apriori 算法[2]，其后大部分算法均是这一算法的变体，可称为类 Apriori 算法。这类算法的基本思想是基于"产生-测试"的迭代思想，算法的特点是简单，计算代价与扫描数据库次数有关，而扫描数据库的次数又与频繁项目集长度有关，因此，在频繁模式较短的情况下，这类算法可获得较好的性能。但当频繁模式较长(如长度为30～50)时，这类算法的性能则极大地下降。最近，Han 等人提出了一个完全不同于 Apriori 的、基于频繁模式树(FP-tree)的频繁模式增长(FP-growth)算法[3]，该算法只需要扫描数据库两次，而且与频繁模式的长度没有关系，因而在频繁模式较短和较长的情况下，均可获得较好的性能。

直接挖掘数量巨大的频繁项目集会带来高昂的计算代价，对许多用户而言也并非对每一个频繁项目集均感兴趣。因而近年，由 Pasquier 等人提出了一个非常有趣的方案，用挖掘频繁闭合项目集代替挖掘频繁项目集[4]，其中关键的一点是所有频繁闭合项目集包含了所有频繁项目集的完整信息，由所有频繁闭合项目集可直接产生所有频繁项目集及其支持度，但数量上前者比后者减少了许多，最多可达几个数量级，大大减少了计算代价。因此，挖掘频繁闭合项目集的研究已受到特别关注，提出的算法有 A-close[4]，CLOSET+[5]等。

尽管如此，挖掘频繁闭合项目集仍是一个计算密集型问题，利用并行计算机强大的计算、存储和输入输出能力是解决该类问题的一个有效途径。挖掘频繁项目集的并行算法已提出了许多，如 CD、DD[6]、IDD[7]、MLFPT[8]等，但这些并行算法均是挖掘所有频繁项目集的，如何针对频繁闭合项目集的特点，设计高效的并行挖掘算法是一个有待研究的问题。

本文正是研究共享存储器模型上频繁闭合项目集的并行挖掘问题。文中分析了该问题的特点，针对这些特点提出了基于频繁模式树(FP-树)自底向上挖掘策略的频繁闭合项目集直接判断法，并通过综合运用 MLFPT、CLOSET+等算法策略的优势，给出了共享存储器模型上基于 FP-树的频繁闭合项目集并行挖掘算法 FCIPM 思想。本文第2节给出了问题的描述，第3节针对问题的特点，提出了基于 FP-树自底向上挖掘策略的频繁闭合项目集直接判断法，并给出了 FCIPM 算法思想，第4节研究了算法的性能，最后给出了结论。

2　问题描述

2.1　频繁项目集

设 $I=\{i_1,i_2,\cdots,i_m\}$ 为项目(item)集合，I 的一个非空子

缪裕青　副教授。

集称为一个项目集(itemset),含有 k 个项目的项目集称为 k-项目集。TDB 是一个交易数据库,其中每个交易 T 表示成一个二元组 (Tid, X),Tid 为交易标识符,X 为 I 中项目组成的集合,即 $X \subset I$。假如某个项目集 $Y \subseteq X$,则称交易 T 包含项目集 Y。包含项目集 Y 的交易数在整个交易数据库中的比例称为项目集 Y 在 TDB 中的支持度(support),用 $sup(Y)$ 来表示。

给定一个最小支持度阈值 min_sup(minimum support threshold),假如一个项目集的支持度大于等于 min_sup,则称该项目集为频繁项目集(frequent itemset)或频繁模式(frequent pattern)。

2.2 频繁闭合项目集挖掘

定义1 假如项目集 X 是频繁项目集,且不存在任何真超集 Y,$Y \supset X$,使得 $sup(Y)=sup(X)$,则称 X 为频繁闭合项目集(frequent closed itemset)。

分析上述定义可知,一个频繁闭合项目集要么不存在真超集,要么只存在支持度小于它的真超集。

引理1 包含频繁项目集 X 且与 X 有相同支持度的频繁闭合项目集有且仅有一个。

证明:对于任一频繁项目集 X,总能找到一个包含 X 的最大频繁项目集 Y,$Y \supseteq X$,使得 $sup(Y)=sup(X)$,且不存在任何频繁项目集 Z,$Z \supset Y$,或对于任何频繁项目集 Z,$Z \supset Y$,$sup(Z)<sup(Y)$,根据定义可知,Y 是包含 X 的频繁闭合项目集。

设 Y 是一频繁闭合项目集,$Y \supset X$,且 $sup(Y)=sup(X)$;设存在另一频繁闭合项目集 Z,$Z \supset X$,且 $sup(Z)=sup(X)$。$\because Y \cup Z \supseteq Y$,$sup(Y \cup Z)=sup(X)=sup(Y)$;$Y \cup Z \supseteq Z$,$sup(Y \cup Z)=sup(X)=sup(Z)$;又 Y 和 Z 都是频繁闭合项目集。$\therefore Y \cup Z=Y$ 且 $Y \cup Z=Z$。因此 $Z=Y$。证毕。

频繁闭合项目集挖掘问题可描述为从大型交易数据库中找出所有频繁闭合项目集。

3 基于 FP-树的频繁闭合项目集并行挖掘算法

3.1 基于 FP-树的频繁闭合项目集判断

在求频繁闭合项目集时,关键的一个技术是判断一个频繁项目集是否是频繁闭合项目集。许多方法采取的是子集判断法[4],即将这个频繁项目集逐个与已发现的频繁闭合项目集比较,若是某个频繁闭合项目集的子集且支持度相等,则该频繁项目集就不是频繁闭合项目集,否则该频繁项目集就是频繁闭合项目集。显然,当频繁闭合项目集个数较多时,这种逐一比较匹配的代价是昂贵的,而且其很强的串行性也不适合并行处理。

由 Han 等人提出的挖掘频繁项目集的频繁模式增长算法,其基本思想是先将交易数据库的信息高度压缩存储在一棵频繁模式树(简称 FP-树)上,其后在 FP-树上递归挖掘频繁项目集,而无需再扫描原交易数据库。该算法的最大特点是 FP-树保存了交易数据库的完整信息,频繁闭合项目集的判断可充分利用这些信息。

一个交易数据库的 FP-树是由各交易中的频繁项目序列构成的一棵前缀树[5],其构造过程如下:首先扫描 TDB,计算每个项目的支持度,去掉不满足支持度阈值的非频繁项目,余下的频繁项目按支持度非递增次序存入频繁项目表(f-list);第二次扫描 TDB,每个交易按 f-list 表中项目次序排序所包含的频繁项目,然后插入到 FP-树中。每次插入都从 FP-树的根结点 root 开始,首先判断交易的第一个项目是否是根结点的孩子,若是直接在该孩子结点的出现次数上加1,然后再以这个孩子结点作为子树的根结点,继续交易下一个项目的判断;若不是根结点的孩子,则将这个项目作为根结点的新孩子结点链到根结点上,同时将新孩子结点的出现次数置为1,再以这个新孩子结点作为子树的根结点,继续交易的下一个项目判断,递归执行这个过程,直到 TDB 中的每个交易都插入到 FP-树为止。由此可见,交易数据库中的所有交易信息都压缩保存到 FP-树中,离根越近的项目,支持度越高。在挖掘过程中,为了快速找到每一个相同的频繁项目,在 FP-树中还建立了相同项目链表,链表中相同的项目结点通过边指针(side-link pointer)链起,每个链表的头指针保存到头指针表(header table)中。

设 TDB 是一个交易数据库,将 TDB 中所有频繁项目及其支持度按支持度非递增排序构成一张表,称为全域频繁项目表,用 Gf-list 表示。

设 X 是一个频繁项目集,在 FP-树上,以 X 为后缀的所有路径中,与 X 同时出现的次数(支持度)满足最小出现次数(最小支持度)的项目称为 X 的局域频繁项目,与 X 同时出现的支持度(次数)定义为该局域频繁项目的域支持度。将 X 的所有局域频繁项目及其域支持度按全域频繁项目表次序排序构成的表称为 X 的局域频繁项目表,用 X'f-list 表示。

由上述定义可以很容易推出以下引理。

引理2 设 X 是一个频繁项目集,Y 是 X 的局域频繁项目,则 $Y \cup X$ 为频繁项目集,其支持度为 Y 的域支持度。

引理3 设 X 是一个频繁项目集,在 FP-树上,若从 X 至根结点的所有路径中不存在任何与 X 同时出现的项目,且在 X 的所有后代结点中也不存在任何与 X 同时出现的项目,则 X 为频繁闭合项目集。

证明:设 X 是一个频繁项目集,在 FP-树上,从 X 至根结点的所有路径中不存在任何与 X 同时出现的项目,说明从 X 向上扩展(向根方向)不存在任何频繁项目集 Y,$Y \supset X$,且 $sup(Y)=sup(X)$。在 X 的所有后代结点中不存在任何与 X 同时出现的项目,说明从 X 向下扩展(向叶方向)也不存在任何频繁项目集 Z,$Z \supset X$,且 $sup(Z)=sup(X)$,这表明不存在任何与 X 有相同支持度的真超集,根据定义1可知 X 就是频繁闭合项目集。证毕。

3.2 基于 FP-树的自底向上分治策略

文[5]提出了两种在 FP-树上挖掘频繁闭合项目集的分治策略:自底向上的物理树投影方法(Bottom-up physical tree-projection)与自上而下的伪树投影方法(Top-down pseudo tree-projection)。针对我们研究的并行挖掘频繁闭合项目集问题,先分析一下这两种策略。自底向上树投影方法尽管物理上构造局域 FP-树后可方便局域挖掘,但建局域树不仅带来时间的额外开销,更主要的是每递归一次建立一棵局域树,递归越深,局域树越多,占的空间就越多,这在多个处理器共享存储器模型(SM)上是不可行的。自上而下伪树投影方法尽管物理上没有构造局域 FP-树,但随着逻辑局域 FP-树的变化需不断调整树结点的边指针,这在多个处理器共享存储器模型上可能会带来许多写冲突。所以在共享存储器模型上简单地使用这两种方法都是不可行的。

实际上,挖掘过程中所需要的所有信息都保存在最初的全域 FP-树中,在递归过程中,只需针对局域频繁项目保存必要的信息,就可通过挖掘全域 FP-树得到所要的结果。因此,

在后面提出的 FCIPM 算法中采取不需要任何树投影直接挖掘全域 FP-树的方法。至于是采取自底向上还是自上而下策略，通过观察与分析发现，在 FP-树中，支持度越高的频繁项目越靠近树根，在自上而下挖掘中，由树根往树叶走，项目的支持度是由大变小，随着项目集长度的增加，其支持度也趋向减小，由引理1可知，在求出包含某频繁项目较长的频繁闭合项目集(支持度小)时，包含该频繁项目较短的频繁闭合项目集(支持度大)也已求出，因而后面任务的工作量相对前面而言要小得多。相比之下，自底向上策略中，由树叶往树根走，项目的支持度是由小变大，在求出包含某频繁项目较长的频繁闭合项目集(支持度小)时，包含该频繁项目较短的频繁闭合项目集(支持度大)并没有求出，因而前后任务相对较均匀。鉴于此，在 FCIPM 算法中为了各处理器的负载平衡采取了自底向上分治策略。

3.3 FP-树的并行构造

将上述基于 FP-树的自底向上分治方法在共享存储器模型(SM)上进行并行化并不是平凡的，其中关键的一点是如何构建 FP-树，使得各处理器既负载平衡又尽量无读写冲突，且便于后面挖掘处理。

在挖掘过程中，多台处理器共享一棵全局 FP-树可以很方便地得到全局信息，但多台处理器共建一棵全局 FP-树，在 SM 模型上由于大量写冲突显然不可行。若由各处理器各建一棵局部 FP-树，在挖掘过程中为了避免将全局频繁而局部非频繁的项目误删除(伪否定现象)，就必须得到全局的信息，如何能既解决写冲突同时又避免伪否定现象的发生？Za¨ane 等人提出的算法 MLFPT[8]采取的是在并行构造多棵局部 FP-树的同时，在一张全局头指针表(header table)中为每一棵局部 FP-树的项目链表都分配一个头指针，这样巧妙地避免了写冲突与伪否定现象的发生。

3.4 FCIPM 算法

算法 FCIPM 分为两个阶段：

第一阶段，并行构造 FP-树。这部分与 MLFPT 算法基本相同，首先将交易数据库划分成多个局部数据库分配给各处理器并行扫描，计算出所有项目的支持度，最后得到全域频繁项目表 Gf-list；然后各处理器再根据 Gf-list 扫描各局部数据库，并行构造各局部 FP-树，同时建立一个全局头指针表(header table)。

第二阶段，并行挖掘频繁闭合项目集。从全域频繁项目表开始，按照自底向上分治策略并行挖掘以各频繁项目结尾的频繁闭合项目集，在基于 FP-树自底向上递归挖掘过程中，逐级建立各局域频繁项目表，根据引理2逐步合并频繁项目集至无法再合并为止，再根据引理3找出频繁闭合项目集，最后根据引理1以及 CLOSET＋算法中修剪技术进行项目修剪以减少搜索空间。当全域频繁项目表中所有频繁项目均已被挖掘整个算法结束。

该算法采取的是基于 FP-树的自底向上分治策略，在文[3]中，已证明该策略可找到包含任一频繁项目的所有频繁模式，依据上述引理2，3则可保证找到包含任一频繁项目的所有频繁闭合项目集。

4 性能分析

FCIPM 算法采取了各处理器并行扫描各局部数据库，并行构造各局部 FP-树，并行挖掘以各频繁项目结尾的频繁闭合项目集，并行判断各频繁闭合项目集，这样既提高了并行性，又避免了在 SM 模型上可能造成的写冲突，特别是频繁闭合项目集直接判断法，不但克服了子集判断法的强串行性，而且直接利用了 FP-树存储的完整信息，节省了额外的空间开销；此外，除了建立最初对应全局数据库的各局部 FP-树，在递归挖掘过程中，只建立各局域频繁项目表，不再建立其对应的各局域 FP-树，这样也可节省大量空间。但这些方法也会带来一些额外的时间开销。若在递归挖掘过程中，每次都建立各局域 FP-树，尽管可以加快挖掘速度，但随着递归层次的增强，各处理器所建立的局域 FP-树所占空间很快会超出共享存储器的存储限度，从而造成系统崩溃。所以，FCIPM 算法采取的是用时间换取空间的一种折衷办法，而且 FCIPM 算法具有良好的可伸缩性。

结论 频繁项目集挖掘问题已得到广泛研究，提出了许多算法，其中基于频繁模式树(FP-树)的频繁模式增长算法是近年提出的一个著名算法，它对频繁长模式和短模式挖掘均获得较好性能。用挖掘频繁闭合项目集代替挖掘频繁项目集是近两年提出的一个新策略，频繁闭合项目集与频繁项目集有相同的表达能力，但在数量上却小许多，从而可大大减少计算代价。由于频繁项目集挖掘是一个计算和数据密集型问题，利用并行计算机的强大计算、存储和输入输出能力一直是吸引频繁项目集挖掘研究者的一个方向，提出了不少并行算法，但都是针对挖掘频繁项目集的。

本文是研究频繁闭合项目集的并行挖掘问题，分析了频繁闭合项目集的特点与频繁项目集的关系，针对并行挖掘特点提出了基于 FP-树的频繁闭合项目集直接判断法，通过综合运用 MLFPT 、CLOSET＋等算法策略的优势，给出了共享存储器模型上基于 FP-树的频繁闭合项目集并行挖掘算法 FCIPM，性能研究表明，FCIPM 算法具有良好的可伸缩性，在时间和空间上均获得较好的性能。

参 考 文 献

1 Agrawal R, et al. Mining association rules between sets of items in large databases. In: Proc. 1993 ACM SIGMOD Int'l Conf. Management of Data, Washington, D. C. , May 1993

2 Agrawal R, et al. Fast algorithms for mining association rules. In: Proc. 1994 VLDB Int'l Conf. , Santiago, Chile, Sept. 1994

3 Han Jiawei, et al. Mining frequent patterns without candidate generation. In: Proc. ACM SIGMOD Int'l Conf. On Management of Data, Dallas, Texas, May 2000

4 Pasquier N, et al. Discovering frequent closed itemsets for association rules. In: Proc. Int'l Conf. On Database Theory, Jan. 1999

5 Wang Jianyong, et al. CLOSET ＋: searching for the best strategies for mining frequent closed itemsets. In: Proc. Ninth ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, Washington, DC, Aug. 2003

6 Agrawal R, et al. Parallel mining of association rules. IEEE Trans. on Knowledge and Data Eng. , Dec. 1996. 8

7 Han Eui-Hong, et al. Scalable parallel data mining for association rules. IEEE Trans. on Knowledge and Data Eng. , May/June 2000, 12

8 Za¨ane O R, et al. Fast parallel association rule mining without candidacy generation. In: Proc. IEEE Int'l Conf. On Data Mining, San Jose, CA USA, Nov. 2001