网格下最大频繁项集挖掘算法的实现

荣秋生,颜君彪
(湖南文理学院 计算机科学与技术系,湖南 常德 415000)

摘 要:随着网格和数据挖掘技术的发展,提出了网格平台下最大频繁项集数据挖掘算法,采用数据库的垂直表示和基于前缀关系的等价划分,以等价类长度的指数函数作为等价类的权值,减少剪枝对负载的影响,合理划分等价类,在动态负载平衡情况下使处理机异步计算,大大提高算法的执行效率。实验证明设计的算法有较好的可扩展性,其性能明显优于其他相关算法。

关键词:网格;最大;频繁项集;等价类;数据挖掘
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2007)01-0098-03

Implementation of Maximal Frequent Itemset Data Mining Based on Grid

RONG Qiu-sheng, YAN Jun-biao
(Department of Computer Science and Technology, Hunan University of Arts and Sciences, Changde 415000, China)

Abstract: As the development of grid and data mining, advance an algorithm of maximal frequent itemset data mining based on grid. With a vertical database layout scheme and a prefix-based equivalence classes, the algorithm is done by a complete inclusive relation between the equivalence classes of gene itemsets, these techniques eliminate the need of synchronization. The experimental results demonstrate the superb efficiency of the approach in comparison with other relative methods.

Key words: grid; maximal; frequent itemset; equivalence class; data mining

0 引 言

从因特网中挖掘出用户需要的信息是非常困难的。在网络环境下,基于关键字的搜索引擎能解决一些问题,但在网格环境下,资源更加丰富多样,建立基于网格环境下的数据挖掘,研究网格数据挖掘算法能有效地帮助用户进行信息的查找与检索。笔者等提出 G_MMFI(Grid Mining Maximal Frequent Itemset),即网格下最大频繁项集数据挖掘算法的研究与实现。采用数据库的垂直表示和基于前缀关系的等价划分,以等价类长度的指数函数作为等价类的权值,在动态负载平衡情况下使处理机异步计算。

1 相关数据挖掘服务模式

1.1 网格平台下数据挖掘服务模式

所设计的网格平台下数据挖掘服务模式包括全局控制器、网格服务器、网格中的资源和由控制器分配的子任务(如图1所示)。

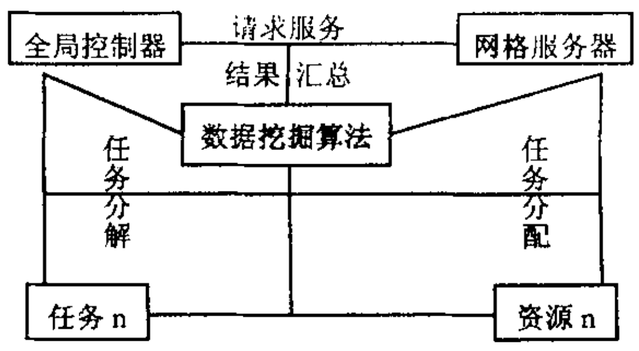

图1 网格平台下数据挖掘模式图

当启动数据挖掘任务时,控制程序将任务分解成若干子任务,通过网格服务器搜索到空闲资源分配运行——若干子任务并行处理,再将局部结果汇总并释放资源。为达到网格中算法处理的高效率,采用动态分配策略,由控制程序搜索空闲处理机,收到请求的处理机给予响应,并分配到一定数量的事务进行处理,处理程序随机向另外的空闲处理机发出请求,共同调配以合理完成任务,处理完任务的处理机向控制程序返回结果和响应请求。

1.2 传统频繁数据挖掘服务模式

CD算法是 Apriori 算法的一个简单并行实现[1],它要求每个处理机都保持整个候选项集,当候选集大到单个处理机内存无法容纳时,就必须进行磁盘读写操作;DD 算法的处理机上保存互不相交的候选集的子集来充分利用内存,它要求每一次迭代中扫描整个数据库及其划分,从

收稿日期:2006-04-21
基金项目:湖南省教育厅重点项目(04A037);湖南文理学院教研资助项目
作者简介:荣秋生(1973-),男,湖南常德人,讲师,硕士,主要研究方向:数据挖掘、网格资源管理;颜君彪,副教授,硕士,主要研究方向:中间件、网格安全。

而带来极大的通信开销;CaD 算法在第 n 次迭代中需对候选集进行重新划分,处理机之间的通信依赖于剪枝信息,它要求每遍迭代都扫描数据库,导致了极大的 I/O 代价。

Zaki 提出的基于数据库采用垂直表示形式的并行算法(Par - Eclat, Par - MaxEclat, Par - Clique, Par - MaxClique)[2],利用相似的并行化策略和不同的等价类划分技术,分别采取自底向上和混合搜索策略,但剪枝存在冗余节点,划分数据库存在无用的记录复制,从而不能得到好的负载平衡。

因此设计的算法要求一个好的并行算法,达到有效划分数据,尽可能减少处理机之间的依赖关系,使各处理机能独立地进行处理;达到负载平衡,最大化算法并行的效率;减少处理机之间的通信和同步,好的并行算法允许各节点异步操作。尽可能在这些因素中找到一个平衡点,来追求最好的效果。

2　频繁项集挖掘算法的实现

2.1　频繁项集相关定义

定义 1　项集,项的集合,含 k 个项的集合称为 k - 项集。

定义2　项集支持数,事务集 D 中包含项集 X 的事务数称为项集 X 的支持数,形式化表示为:support_count$(X) = |\{T \mid T \in D, X \subseteq T\}|$。

定义3　项集支持度,项集 X 的支持度是指项集在事务集 D 中出现的概率。记 support$(X) = P(X)$ = support_count$(X)/|D|$。

定义 4　频繁项集,支持度大于或等于用户指定的最小支持度阈值的项集称为频繁项集。项集也称为模式,频繁项集则称为频繁模式。频繁项集挖掘就是找出事务集 D 中所有支持度大于等于最小支持度阈值的项集。如果项集 X 是频繁的,那么它的所有非空子集都是频繁的;如果项集 X 是非频繁的,那么它的所有超集都是非频繁的。

笔者等设计的算法目标是通过指定需要产生的频繁项集的数量 N 而不是最小支持度阈值来控制挖掘过程产生的频繁项集的规模。

定义 5　N 个最频繁项集,将所有项集按支持度降序排序,令 S 是排在第 N 位项集的支持度,N 个最频繁项集由所有支持度大于等于 S 的项集组成,表示为公式(1)。

$$\text{Top}N = \{X \mid \text{support}(X) \geqslant S\} \tag{1}$$

这里需要注意的是,N 个最频繁项集的个数可能大于 N。因为支持度等于 s 的项集可能不只一个,如果第 $N+1, N+2, \cdots, N+m$ 个项集的支持度也是 S,那么 N 个最频繁项集则由 $N+m$ 个频繁项集组成。将所有 k - 项集按照支持度降序排序,令 S_k 是排在第 N 位的 k - 项集的支持度。N 个最频繁 k - 项集 NL_k 是所有支持度大于等于 S_k 的 k - 项集,表示为公式(2)。

$$NL_k = \{X \mid \text{support}(X) \geqslant S_k \text{ 且 } |X| = k\} \tag{2}$$

使用全局统一的当前最小支持度阈值,在每轮频繁项集产生过程中根据当前 N 个最频繁项集的最小支持度动态地调整该阈值。由于调整后的阈值大于等于原来的阈值,使得每次都是在更高的支持度阈值条件下挖掘剩余的频繁项集。如果在挖掘过程中最小支持度阈值被动态调整,那么根据当前最小支持度阈值产生的频繁项集称为当前频繁项集[3]。

2.2　频繁项集挖掘算法的实现

最典型的频繁挖掘算法是先分别求出所有 N 个最频繁 k_ 项集($k = 1, 2, \cdots, m$) 共 mN 个;为了减少搜索空间,N 个最频繁 $(k+1)$_ 项集的候选项集利用 Apriori 算法的连接步连接 N 个最频繁 k_ 项集得到[4],候选项集按支持度降序排序,第 N 个候选项集的支持度为 S_{k+1};如果 $S_{k+1} \geqslant S_k$,支持度大于等于 S_{k+1} 的候选($(k+1)$_ 项集就是 N 个最频繁 $(k+1)$_ 项集);如果 $S_{k+1} < S_k$,则回溯到 k 项集,以 S_{k+1} 为最小支持度阈值产生频繁 k_ 项集,再用新产生的频繁 k_ 项集构造新的候选 $(k+1)$_ 项集,进而得到新的最小支持度 S_{k+1} 和 N 个最频繁 $(k+1)$_ 项集。算法生成的候选项集存在以下两方面的不足:

(1) 该方法需要找出所有的 N 个最频繁 k_ 项集,如果频繁项集的最大长度为 m,那么共有 $N * m$ 个频繁项集成为 Top$N$ 的候选项集;

(2) 算法没有使用全局统一的最小支持度阈值,从而导致频繁 $(k+1)$_ 项集的最小支持度阈值 S_{k+1} 小于 S_k 时,必须回溯重新挖掘最小支持度为 S_{k+1} 的频繁 k_ 项集,降低了挖掘效率。

采用数据库的垂直表示和基于前缀关系的等价划分,以等价类长度的指数函数作为等价类的权值,减少剪枝对负载的影响,所设计的 G_ MMFI 包括初始化、异步计算和后处理三个阶段。其中初始化得到各频繁项的支持度 support(x),再划分等价类,并选择性地复制数据库记录,为各处理机异步执行作准备;各处理机异步计算处理等价类获得计算结果;最后汇总各处理机的结果。

首先划分等价类,然后将它们分配到各处理机上,分配的原则是使负载不平衡最小化,使为复制数据库记录而需要的处理机之间的通信最小化。采用了基于等价类长度的指数函数 $2^{|[X]|}$ 的负载度量方案,采取基于因子项集完全包含的任务划分策略和基于等价类按需分配的数据划分策略来维持较好的负载平衡和较小的通信量,等价类分配之后,根据各处理机上所分配的等价类的因子项集来决定数据库记录的选择性复制。

异步计算阶段,在初始化阶段结束的时候,有关的数据库记录都已在各自处理机的本地磁盘上。因此,网格中的处理机可以各自独立地执行,计算其本地最大频繁项集的集合。各等价类依次处理,剪枝在各处理机上独立进行。处理机之间无需同步和交换信息。在这一阶段,各处理机只需扫描本地数据库划分一遍。从而节省了大量的 I/O 和同步开销。

后处理阶段,由于网格中处理机得到的是本地最大频繁项集的集合,其中,会有一些本地最大频繁项集为另一些局部最大频繁项集所包含。在后处理过程中,将剔除这些有包含关系的局部结果,得到全局的最大频繁项集的集合。

G_MMFI——网格下最频繁项集数据挖掘算法过程描述为:

```
Procedure G_MMFI;
{
Initialize s = minsup;//初始化当前最小支持度阈值,缺省时设为某一较小的数
   /* 自底向上生成当前候选 k_项集 */
   S1 = support(X);//最小支持度为 S1
   P1 = NL1;//以 N 个最频繁 1 - 项集 NL1 为当前频繁 1 - 项集 P1
   For(X=1;X<=N;X++)
   {if support(Top(X))>S1
     {s=max(s,S1);//调整当前最小支持度
     Replace(P1);}//调整当前频繁项集
   }
   连接 P 中所有 1 - 项集生成候选 2 - 项集 C2;
   最小支持度阈值挖掘频繁 2 - 项集 NL2;
   L=link(P,L2);//合并排序
   If(|L|>=N)
   {s=s2;//用 L 中第 N 个项集的支持度替换原来的 s
   Replace(P2);//以 L 中支持度大于等于 s 的项集更新当前频繁项集 P2
   }
}
```

依次挖掘当前支持度阈值为 s 的频繁 3 - 项集,频繁 4 - 项集,…,频繁 k - 项集,直到没有任何频繁项集产生,每轮产生频繁 k - 项集后,调整当前最小支持度 s 和当前频繁项集,使下一轮的频繁项集挖掘在更高的支持度阈值下进行,从而有效地修剪不必要的候选项集,提高挖掘算法的效率。

3 实验数据分析

实验数据采用合成数据库 T10.I4.D2084K,T15.I4.D1471K 和 T20.I6.D1137K[5],其中,T 表示平均交易长度,I 表示平均最大频繁项集的长度,D 表示数据库中交易的数目,合成数据库由 Agrawal 提出的合成数据产生器产生,设定潜在的频繁项集的个数为 2 000,数据库的项数为 1 000,取支持度阈值 $\varepsilon = |D| * 0.25\%$,等价类的分配及访问节点数和剪枝损失情况以及不同的等价类分配方案下数据库冗余情况如表 1 所示。

根据上述数据,利用我们设计的网格平台数据挖掘模式,以 $2^{|[X]|}$ 为权值,采用数据库的垂直表示和基于前缀关系的等价划分,以等价类长度的指数函数作为等价类的权值,减少剪枝对负载的影响,合理划分等价类,实验效果如图 2 所示。

表 1 实验数据的等价类情况

等价类分配方案	处理机	等价类序列	访问节点	剪枝	数据库冗余记录数
按 $\|[X]\|$ 分配	P_1	1.2.3.7	26	2	6
	P_2	4.5.6	14	2	6
按 $2^{\|[X]\|}$ 分配	P_1	1	21	3	7
	P_2	4.5.2.3.6.7	21	3	6
按 $2^{\|[X]\|}$ 及包含关系分配	P_1	1.2.3	23	1	6
	P_2	4.5.6.7	17	1	4

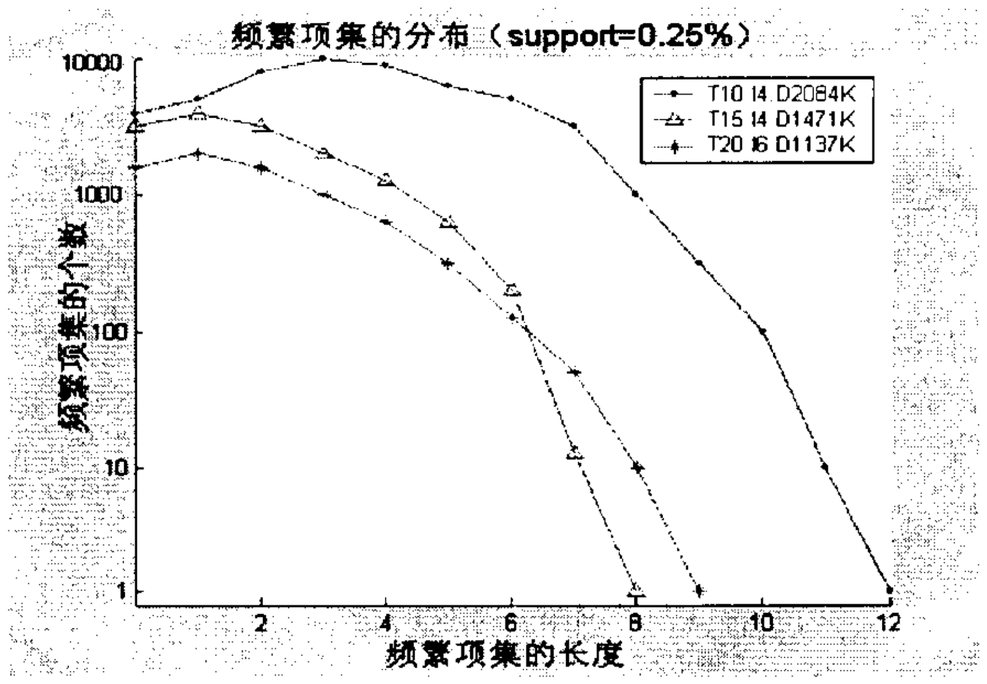

图 2 实验数据库中项集的分布图

4 小 结

笔者等提出 G_MMFI——网格下最大频繁项集数据挖掘算法,在初始化阶段之后即开始作异步计算,利用因子项集的完全包含关系在处理机之间贪心分配等价类,极大地发挥了剪枝效率,减少了数据库记录的冗余复制。采用以 $2^{|[X]|}$ 为权值,客观地度量各处理机的负载,得到了更合理的负载平衡,在动态负载平衡情况下使处理机异步计算,大大提高算法的执行效率。实验表明所设计的算法有较好的可扩展性,其性能明显优于其他相关算法。

参考文献:

[1] Han Eui - Hong, Karypis G, Kumar V. Scalable Parallel Data Mining for Association Rules[C]//in: Peckham J. Proceedings of ACM SIGMOD International Conference on Management of Data. Tucson, Arizona, USA: ACM Press, 1997:277 - 288.

[2] Zaki M J. Scalable algorithms for association mining[J]. IEEE Transactions on Knowledge and Data Engineering, 2000, 12 (3):372 - 390.

[3] 宋海声. 快速开采最大频繁项目集[J]. 计算机应用研究, 2004(3):45 - 46.

[4] 牛玉广,邓 亮. 一种改进的关联规则混合挖掘算法[J]. 微机发展,2005,15(11):141 - 143.

[5] Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules in Large Databases[C]//in: Bocca J B, Jarke M, Zaniolo Z. Proceedings of the 20th International Conference on Very Large Data Bases. Santiago, Chile: Morgan Kaufmann, 1994: 487 - 499.