云环境下基于二进制编码聚类的并行频繁项集挖掘算法

刘 博，李 云，张晓斌，徐 杰
（扬州大学 信息工程学院，江苏 扬州 225127）

摘 要：本文提出了一种云环境下基于二进制编码的并行频繁项集挖掘算法，利用一种特殊的二进制编码的依赖度计量方法对原始数据集合进行编码转换及依赖度聚类，然后将数据集分布部署在云环境中，并采用共享多头表的FP-Growth并行改进算法挖掘频繁项集.实验表明，对于大规模数据集来说，本文算法可以取得良好的性能.

关键词：云计算；二进制编码；聚类；并行；频繁项集
中图分类号：TP18 **文献标识码**：A **文章编号**：1000－7180(2012)11－0062－04

A Parallel Frequent Itemsets Mining Algorithm Based on Binary Coding and Clustering under Cloud Environment

LIU Bo, LI Yun, ZHANG Xiao-bin, XU Jie
(College of Information Engineering, Yangzhou University, Yangzhou 225127, China)

Abstract: This paper proposes a parallel frequent itemsets mining algorithm based on binary coding under cloud environment. A special binary coding dependency calculating method is adopted to transfer the raw data and cluster based on dependency, then the data is distributed deployed in cloud environment and the parallel improved algorithm of FP-Growth based on shared multi-head table is used to mine frequent item sets. Experiments show that the algorithm performed nicely with large scale of data sets.

Key words: cloud computing; binary coding; clustering; parallel; frequent itemsets

1 引言

在如今这个数据量庞大的信息时代，频繁项集挖掘(FIM)是有效地找到人们感兴趣数据的方法之一.最早提出的Apriori算法[1]与之后提出的FP-Growth算法[2]取得了良好的效果，但对于更大规模的数据量，传统算法在内存中的存储方式使之成为了大规模数据挖掘的瓶颈.虽然MLFTPT算法[3]、PFP-tree算法[4]都大大的提高了执行效率，但都忍受巨大的通信开销.云计算可以在不同地域的异构平台上运行海量数据计算.PFP算法[5]虽然把全局挖掘任务分割成相互独立的子任务并映射到MapReduce下得到接近线性的加速比，但当数据量非常庞大的时候各个计算节点间有很大的数据冗余，执行效率显著降低.

本文提出一种基于Map-Reduce框架的并行FP-Growth算法来进行频繁项集挖掘，将大规模事务数据分布到不同的独立计算节点上，把挖掘任务映射(map)到相应MapReduce上执行，但在大规模数据集下，为了减少数据冗余，提出一种基于二进制的特殊依赖关系编码规则对数据集进行依赖度计量，然后利用一种矩阵聚类的算法对数据集进行划分，最后分布到各个计算节点中，采用一种在多计算节点上共享多头表的并行频繁挖掘算法有效完整的挖掘出所有的频繁项集.

2 相关知识

假设$I=\{i_1,i_2,\cdots,i_n\}$是一个项集集合，$X\subseteq I$

收稿日期：2012－05－30；**修回日期**：2012－06－28
基金项目：国家自然科学基金(61070047,61070133)；江苏省自然科学基金(BK2010134)；江苏省教育厅自然科学基金(11KJD520011)

表示 X 是 I 的一个子集,长度为 k 的项集称作 k 项集.D 为一个事务数据库,D 中含有的 X 的数目为项集 X 的支持度 $\sup(X)$,如果 X 在数据库中的支持度不小于一个给定的支持度阈值 minsup,则称 X 为频繁的项集,否则称 X 为非频繁.

FP-tree 是一棵频繁模式树,具有高度压缩的数据结构,树中的每一个节点表示一个项集,项集的元素组成是从根节点到此节点的路径上的所有节点构成,树节点的 count 计数表示此项集在数据集中的统计数之和,图 1 是 minsup=3 的相应的 FP-tree 结构.

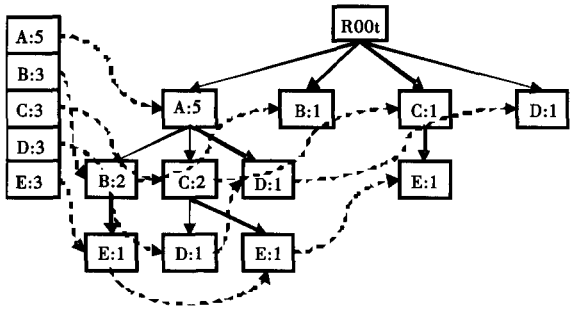


图 1　FP-Tree(minsup=3)

3　云环境下基于二进制编码的并行频繁项集挖掘算法

本文提出的云环境下基于二进制编码的并行频繁项集挖掘算法 BCC-FIM(Frequent Itemset Mining Algorithm based on Binary Coding in Cloud)首先根据一种特殊的二进制编码的依赖度计量方法对原始数据集合进行编码转换及依赖度聚类,然后根据聚类结果把数据集分布部署到云环境中多节点上并利用一种共享多头表的 FP-Growth 并行改进算法独立完整地挖掘频繁项集.可以有效避免因数据冗余而造成的存储空间的浪费以及由于全局挖掘递归次数的增加而造成的时间浪费,从而无遗漏的并行独立挖掘出所有的频繁项集.

3.1　数据集编码及聚类

3.1.1　项集编码规则

定义 1　频繁一项集的降序序列为 $Array$,设序列 $Array$ 的长度为 m,对于 $\forall T_i \in DB$,用 m 位二进制码 B_i 表示 T_i,若序列 $Array$ 中某一项包含在 T_i 中,则在 B_i 中相应的用 1 表示,其余位补 0,即

$$T_i = b_1 b_2 \cdots b_m, \text{其中 } b_j = \begin{cases} 1, & a_j \in T_i \\ 0, & a_j \notin T_i \end{cases}$$

$j \in [1,m]$, $a_j \in Array$

算法 1 是数据集进行二进制编码过程的伪码表示.

算法 1　Procedure: BinaryEncoding(DB, Array)
输入:原始数据库 DB,频繁项序列 Array
输出:数据集编码 BDB-List

```
1:Define B_i, BDB-List
2:Foreach T_i in DB do
3:   Clear B_i
4:   Foreach a_j in T_i do
5:      Set the corresponding bit in the B_i to 1
6:      Binaryencode(a_j)⇒B(a_j)
7:      Change B_i: B_i=B_i∪B(a_j)
8:      Set B_i to BDB-list: BDB-list=B_i, BDB-list++
9:      Clear the T_i from DB
10:  End
11:End
```

3.1.2　编码数据集的依赖度矩阵 DM

定义 2　设 T 为事务集合,T_i、T_j 为 T 的子集.若 T_i、T_j 的二进制编码 B_i、B_j 不为 0 且交集不为空,则记 T_i 和 T_j 的依赖度为 $\text{dependency}_{ij} = B_i \cap B_j$.

定义 3　设矩阵 OM 为事务间的依赖度构成的原始矩阵,n 为所有二进制编码非 0 的事务集合的数量,若 $TM(x,y)(x,y<n)$ 为 OM 中第 x 行列与第 y 行列转置生成的过渡矩阵,则全局参数 $GM(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{n} TM(x,y)_{i,j}(TM(x,y)_{i,j-1} + TM(x,y)_{i,j+1})$ 为全局依赖度度量 $\text{dependency}_{gb}(GM(x,y))$.

利用 BEA 算法[6]对矩阵 OM 进行依赖度聚类转换,若 $GM(x_1,y_1) > GM(x_2,y_2)$,则全局依赖度 $\text{dependency}_{gb}(GM(x_1,y_1)) > \text{dependency}_{gb}(GM(x_2,y_2))$

3.1.3　数据集聚类及部署

定义 4　设 DM 为具有最大 GM 值的依赖度矩阵,矩阵 DM 中第 i 行第 j 列的值 DM_{ij} 为原始矩阵 OM 变换后相应事务间的依赖度,则分类参数为:

$$PM_p = \sum_{i=1}^{p}\sum_{j=1}^{p} DM_{ij} * \sum_{i=p+1}^{n}\sum_{j=p+1}^{n} DM_{ij} - \left(\sum_{i=1}^{p}\sum_{j=p+1}^{n} DM_{ij}\right)^2, \quad p < n.$$

若 $PM_{p_1} > PM_{p_2}$,则

$$| \text{dependency}_{gb}\left(\sum_{i=1}^{p_1}\sum_{j=1}^{p_1} DM_{ij}\right) -$$

$$\text{dependency}_{gb}\left(\sum_{i=p_1+1}^{n}\sum_{j=p_1+1}^{n}DM_{ij}\right)|>$$

$$|\ \text{dependency}_{gb}\left(\sum_{i=1}^{p_2}\sum_{j=1}^{p_2}DM_{ij}\right)-$$

$$\text{dependency}_{gb}\left(\sum_{i=p_2+1}^{n}\sum_{j=p_2+1}^{n}DM_{ij}\right)|$$

用一种递归的二分算法(BinaryParting)(算法2)来求最优解,其中 S_i 为事务 T_i 的大小,CS_j 为计算节点 CN_j 的存储容量.

算法 2 Procedure: BinaryParting(DM,$CS[]$)
输入:依赖度矩阵 DM,节点负载能力 $CS[]$
输出:各节点上分配的数据集键值对$<CS_j,T_i>$

1. 把矩阵 DM 的行列元素二分成两类$\{T_1,T_2\cdots T_p\}$(矩阵 DML)与$\{T_{p+1},T_{p+2}\cdots T_n\}$(矩阵 DMR),使得分类参数 PM 最大化,即类$\{T_1,T_2\cdots T_p\}$与类$\{T_{p+1},T_{p+2}\cdots T_n\}$之间依赖度最小,类内依赖度最大,选取 p 的值从 1 到 $n-1$,最后得到 p 的值使得 PM 最大.

2. 分别计算 DML 和 DMR 的数据量大小 $S_{DML}=\sum_{i=1}^{p}S_i$,

$S_{DMR}=\sum_{i=p+1}^{n}S_i$.

3. 如果 DML 或 DMR 的数据量能够放到某一个计算节点中($S_{DML\parallel DMR}<CS_j$),就把相应的事务数据分配到相应的计算节点,并且更新计算节点 $CS_j=CS_j-S_{DML\parallel DMR}$,执行步骤 4.

否则如果 $S_{DML\parallel DMR}>\forall CS_j$,将 $DML\parallel DMR$ 看成 DM 跳到第一步递归执行.

4. 记录$<CS_j,T_i>$,其中 $T_i\in(DML\parallel DMR)$.

3.2 云环境下的并行频繁项集挖掘算法

在云节点中利用基于共享多头表的无数据传输的 FP-Growth 并行改进算法(BCC-FIM)对数据进行频繁挖掘,如算法 3 所示. 与传统 FP 树不同的是本地 FP 树(MHTFP-tree)在云环境中的 Master 节点上共享一张全局多头表 MHT,包含了所有处理节点上的分片数据集的项集计数.

定义 5 (膨胀支持度)设 $C=\{C_1\cup C_2\cup\cdots\cup C_n\}$是一个模式集合,若 $C_j,j\in[1,n]$是 C 的一个子模式,则 $Sup(C)\leqslant\sum_i\min\{Sup_i(C_1),Sup_i(C_2),\cdots,Sup_i(C_n)\}$,其中 i 为计算节点的数量.

对于模式 C 的膨胀支持度计数 EGS,有 $Sup_{gs}(C)=\sum_i\min\{Sup_i(C_1),Sup_i(C_2),\cdots,Sup_i(C_n)\}$. 递归挖掘模式 C 的频繁模式时,综合模式 C 在本地处理机上的前缀树 $PT_{i\{C\}}$ 与全局多头表 MHT 来创建本地头表 $HT_{i\{C\}}$,然后筛除 $Sup_{gs}(C)$ 小于支持度阈值的项,最后在前缀树 $PT_{i\{C\}}$ 中剪枝掉非频繁的项生成本地候选模式树 $CT_{i\{C\}}$,此时 $CT_{i\{C\}}$ 中已经筛除了大多数的非频繁项,利用编码依赖度的计量方法使得各个处理节点上的数据集合有最大化的模式长度,使得频繁挖掘时由节点中的短模式而引起的频繁度膨胀造成的影响最小,在执行效率上有了极大提高. 图 2 是以图 1 中 P1 节点的数据集为例,minsup=3 的节点频繁项集挖掘过程.

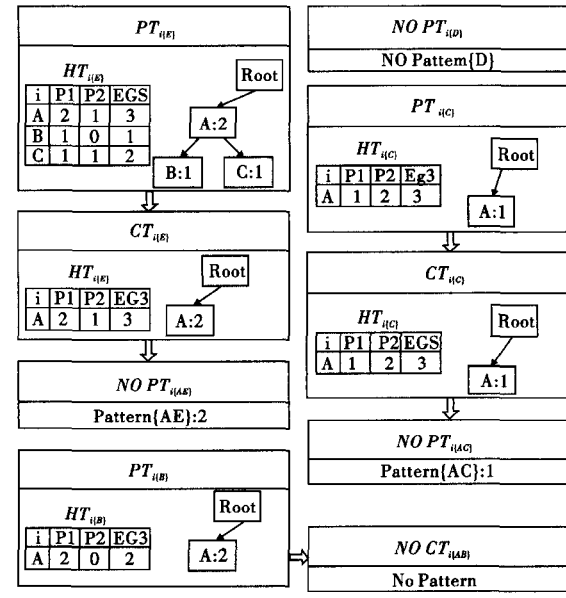
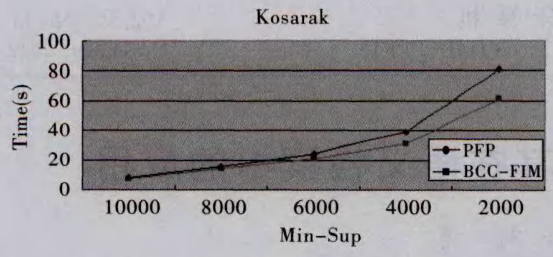


图 2 P1 节点的频繁模式挖掘流程图

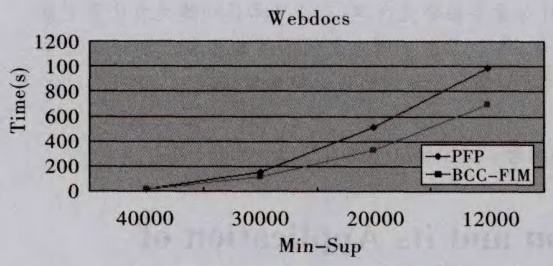
4 实验验证及分析

图 3 分别为 Kosarak 数据集和 Webdocs 数据集在不同的支持度阈值下的执行时间,由于本文编码规则的独特性,类内的数据在构造本地 FP-Tree 时具有最大限度的相同前缀性,类间数据在原始的全局 FP-Tree 中为没有交叉的枝,因此把各个类分布到不同的云计算节点进行挖掘时构造的本地 FP-Tree 相互独立,极大减少了多处理节点间的数据冗余以及因冗余数据而造成的存储空间的浪费,并且减少了频繁挖掘时多处理节点中因冗余数据而造成的额外递归迭代次数从而减少了整体的执行时间,大大提高了多计算节点间的并行性.

图 4 说明随着处理机个数的增加,BCC-FIM 算法产生的膨胀频繁项个数增多,但是由于聚类算法的特殊性,曲线趋于平缓,对比可以看出,本文提出的 BCC-FIM 算法和频繁模式的长度无关.



(a)Kosarak 在不同支持度下的执行时间



(b)Webdocs 在不同支持度下的执行时间

图 3　Kosarak 与 Webdocs 比较

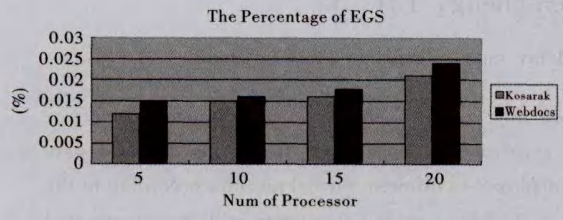


图 4　不同节点数目下膨胀频繁项百分比

5　结束语

本文提出一种基于二进制编码的依赖度计量方式对数据集合进行相似度聚类，并且根据一种启发式分割算法对数据集合进行分片后传递给多个计算节点并行执行，为了减少多节点间的通讯开销，本文采用一种在云计算中 Master 节点上共享一张全局多头表，在本地处理机按照模式增长理论构造本地头表的方法对候选模式进行过滤，统计膨胀频繁项集. 在实验中可以发现，尽管在多个处理机上分布的数据集进行挖掘的时候会产生膨胀的非频繁模式，但是其在全局结果中的比例是非常小的，用二进制编码方式定义依赖度指标，最大限度的把相同前缀的数据集分片到一起，弥补了 BCC-FIM 算法的缺点，在云环境下并行挖掘频繁项集时取得了良好的性能.

参考文献：

[1] Agrawal Rakesh, Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases [C]// Proceedings of 20th International Conference on Very Large Data Bases, 1994: 487-499.

[2] Jiawei Han, Jian Pei, Yiwen Yin. Mining frequent patterns without candidate generation[C]// Proceedings of the ACM SIGMOD International Conference on Management of Data, 2000,29(2): 1-12.

[3] Zaïane O R, El-Hajj M, Lu P. Fast parallel association rule mining without candidacy generation[C]// Proceedings of IEEE International Conference on Data Mining, 2001: 665-668.

[4] Javed A, Khokhar A. Frequent Pattern Mining on Message Passing Multiprocessor Systems[J]. Distributed and Parallel Databases, 2004,16(3): 321-334.

[5] Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang, and Edward Y. Chang. PFP: Parallel FP-Growth for Query Recommendation[C]// Proceedings of the 2008 ACM Conference on Recommender Systems, 2008: 107-114.

[6] McCormick W T, Sehweitzer P J. Problem decomposition and data reorganization by a clustering technique [J]. Operations Research,1972,20(5): 993-1009.

作者简介：

刘　博　男，(1987－)，硕士研究生. 研究方向为数据挖掘、云计算.

李　云　男，(1965－)，博士，教授. 研究方向为数据挖掘、并行分部处理.

张晓斌　男，(1986－)，博士. 研究方向为分部式处理与高性能计算.

徐　杰　男，(1989－)，硕士研究生. 研究方向为数据挖掘与高性能计算.