

基于二叉树的并行频繁项集挖掘算法

陈 静, 郑 彦

(南京邮电大学 计算机学院, 江苏 南京 210003)

摘 要: 大数据时代的到来,使得人们对数据的处理速度、利用率等方面的要求变得更高。在频繁项集挖掘方面, Count Distribution 算法和 Data Distribution 算法是比较经典的并行频繁项集挖掘算法,由于挖掘过程中需要较大的存储空间和通信开销,挖掘效率并不十分理想。文中提出了一种基于二叉树的并行频繁项集挖掘算法,利用了 MapReduce 的并行性,先通过遍历二叉树的方法找出数据库中固定大小的所有子集,然后统计每个子集的出现次数,再与事先设定好的一个固定阈值进行比较,超过阈值的子集即为所求的频繁项集。通过对实验结果进行对比分析表明,提出的算法只需要一次 MapReduce 过程即可完成挖掘,充分利用了集群的并行性,不需要使用迭代的方式进行挖掘,性能上明显优于 CD 和 DD 算法,也就是说,该算法具有较高的挖掘效率。

关键词: 频繁项集挖掘; MapReduce; 并行计算; 二叉树

中图分类号: TP311

文献标识码: A

文章编号: 1673-629X(2015)10-0080-04

doi:10.3969/j.issn.1673-629X.2015.10.017

Parallel Algorithm of Frequent Itemset Mining Based on Binary-tree

CHEN Jing, ZHENG Yan

(College of Computer, Nanjing University of Posts and Telecommunications,
Nanjing 210003, China)

Abstract: Along with the advent of the era of big data, people have higher requirements in the speed of data processing and the utilization of data. In the aspect of mining frequent itemset, the algorithms of Count Distribution and Data Distribution are classical parallel algorithms for mining frequent itemset, because large storage space and communication overhead are needed in the process of mining, the mining efficiency is not very ideal. A parallel algorithm of frequent itemset mining based on the binary-tree is proposed in this paper, it takes advantage of the parallelism of MapReduce. Firstly, find out all subsets of fixed size in the database by using the method of traversing the binary-tree. Secondly, count occurrence numbers of each subset, and compare with a fixed threshold which is set in advance. If the occurrence number of a subset is more than the threshold value, the subset is the frequent itemset which is requested. The study of the comparison and analysis of the experimental results show that the proposed algorithm needs only one process of MapReduce to complete the mining work, it makes full use of the parallelism of the cluster. It does not need to use iterative way for mining frequent itemset, and the performance is superior to the CD and DD algorithms, in other words, it has higher mining efficiency.

Key words: frequent itemset mining; MapReduce; parallel computing; binary-tree

0 引 言

频繁项集挖掘是许多数据挖掘问题中都比较重要的一个步骤,比如关联规则、序列模式、闭模式、最大化模式还有许多其他重要的数据挖掘任务^[1]。1993年, Agrawal 等首次在大数据库中提出频繁项集的挖掘问题^[2],之后国内外学者也一直在研究。其中比较经典的 Apriori 算法^[3]通过不断地构造候选集、筛选候选集

来挖掘频繁项集,需要多次扫描原始数据,当原始数据较大时,磁盘 I/O 次数会太多,效率比较低。近年来,有很多专家学者对于频繁项集的并行计算进行了大量的研究,其中较有影响的包括 R. Agrawal 等提出的类 Apriori 算法的并行算法 Count Distribution (CD) 和 Data Distribution (DD)^[4]。但通过实验对比可以发现 CD 算法中候选集哈希树存储空间较大,而 DD 算

收稿日期:2015-01-29

修回日期:2015-05-04

网络出版时间:2015-09-23

基金项目:国家“973”重点基础研究发展计划项目(2006AA01Z201)

作者简介:陈 静(1990-),女,硕士研究生,研究方向为数据仓库与决策支持系统;郑 彦,教授,硕士生导师,研究方向为数据挖掘、信息安全。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20150923.1505.038.html>

法需要传递大量的数据,通信开销较大。

同时,由于现在正处于大数据时代,每天需要处理的数据源源不断,因此几种经典的频繁项集算法其效率已不能很好地提供服务。因此,文中在这样的背景下,通过学习和研究了几种并行频繁项集挖掘算法^[5-6],将 MapReduce 并行计算思想和频繁项集挖掘算法相结合^[7],提出了一种基于二叉树的并行频繁项集挖掘算法模型。实验结果表明,提出的算法不需要使用迭代的方式,只需一次 MapReduce 过程即可完成频繁项集的挖掘,充分利用了集群并行化的思想提高了挖掘效率。

1 算法概述

频繁项集挖掘用于挖掘一些经常一起出现的项集合,这个项集合就是频繁项集。通过挖掘频繁项集,可以进行关联推荐,即当在一个事务中出现频繁项集的其中一个项时,可以将该频繁项集中的其他项作为推荐^[8-9]。比较经典的例子就是购物篮分析中啤酒和尿布的关联推荐。

本节将介绍一些关于频繁项集的基础知识以及经典的频繁项集挖掘算法。

1.1 相关定义

定义 1(支持度与置信度):假设有一个包含 m 项的集合 $I = \{I_1, I_2, \dots, I_m\}$, 一个表示购物交易的事务数据库 $D = \{T_1, T_2, \dots, T_n\}$, 其中事务 T_i 是 I 的非空子集。关联规则是形如 $A \Rightarrow B$ 的蕴含式,其中 A, B 是 I 的子集,且 A 与 B 的交集为空集。 A 和 B 是否具有关联性是由某一规则 $A \Rightarrow B$ 的支持度(support)和置信度(confidence)来衡量的。关联规则 $A \Rightarrow B$ 的支持度是指 D 中事务包含 $A \cup B$ 的百分比,置信度是指包含 A 的事务中同时包含 B 的百分比,即条件概率 $P(B|A)$, 可表示如下:

$$\text{sup}(A \Rightarrow B) = P(A \cup B)$$

$$\text{conf}(A \Rightarrow B) = P(B|A)$$

若 $\text{sup}(A \Rightarrow B)$ 和 $\text{conf}(A \Rightarrow B)$ 均高于某个事先设定的阈值,则可以认为 A 和 B 具有关联性。

定义 2(频繁 k -项集):设 $I = \{I_1, I_2, \dots, I_m\}$ 为包含 m 项的集合,其中 $I_j(j = 1, 2, \dots, m)$ 表示一个项。如果 X 包含在 I 中,则集合 X 被称为项集。若项集 X 中包含的项的个数为 k ,则 X 被称为 k -项集。项集 X 的支持度表示为 $\text{sup}(X)$,它的值与概率 $P(X)$ 的值相等,可以用 D 中包含 X 的事务数占所有事务数的百分比来求得。对于给定的事务数据库 D 和最小支持度阈值 λ ,如果 $\text{sup}(X) \geq \lambda$,则项集 X 被称为频繁项集,此时若频繁项集 X 中包含的项的个数为 k ,则 X 被称为频繁 k -项集。

1.2 Apriori 算法

Apriori 算法通过多轮迭代的方法来逐步挖掘频繁项集,是频繁项集挖掘中的经典算法。在第一轮迭代中,计算事务数据库中每一项的支持度,并找出所有的频繁项。之后的每次迭代中将前一轮生成的频繁 k -项集作为本轮迭代的种子项集来生成候选 $(k+1)$ -项集,然后计算每个候选 $(k+1)$ -项集在事务数据库中的实际支持度,这样就可以找出所有的频繁 $(k+1)$ -项集,并将此作为下一轮迭代的种子项集。这样不停地迭代直至不能产生新的频繁项集为止。

根据上述算法过程的描述可知,该算法是要对全部项穷尽所有可能的组合,并计算每种组合的实际支持度,在真正执行时可能造成时间和空间复杂度都较大,下面总结了两个 Apriori 算法的缺点:

(1)由频繁 k -项集进行自连接生成的候选 $(k+1)$ -项集数量巨大;

(2)在验证候选 $(k+1)$ -项集的时候需要对整个数据库进行扫描,非常耗时。

1.3 Count Distribution 算法

CD 算法是一种并行频繁项集挖掘算法,使用了一种在其他空闲处理器上进行冗余的并行计算来避免节点间通讯的原则,主要用于最小化节点间的通讯^[10]。

CD 算法在执行时只有一步与传统的 Apriori 算法不同,因为该算法是多个处理器并行计算,所以在计算得到每个候选 $(k+1)$ -项集在事务数据库中的实际支持度后,各处理器需要交换彼此的实际支持度计数来得到全局支持度计数。由于每个处理器上的候选项集是相同的,所以实际上要做的就是各处理器将其实际支持度计数按顺序放入一个计数数组,再执行一个并行的向量汇总即可。

该算法利用交换支持度计数而不用交换数据元组来减少通讯开销,但是它没有有效地利用内存空间,在挖掘过程中需要较大的存储空间来存放候选集哈希树。实验结果表明,该算法比较适用于不同项较少并且支持度较高的数据集。

1.4 Data Distribution 算法

DD 算法也是一种并行的频繁项集挖掘算法,较 CD 算法有所改进,当处理器数量增加时 DD 算法可以充分有效地利用系统内存空间^[11]。该算法第一遍的挖掘与 CD 算法类似,但其后的过程中会通过分割候选项集哈希树来减小哈希树所占的内存空间,但这样也会带来另一个问题,就是分割过程中因为要传输大量的数据,因而大大增加了通信量。并且随着算法循环次数的增加,候选集哈希树会变得很小,但是通信量并没有减小。实验结果表明,该算法仅适用于具有很快通讯速度的机器。

2 MapReduce 并行计算

随着计算机及信息的快速更新发展,行业应用系统的规模迅速扩大,其产生的数据呈爆炸性增长,传统算法的处理效率越来越不能满足实际需求。在这样的背景下,Google 公司发明设计了 MapReduce, MapReduce 是一种编程模型,用于大规模数据集(大于 1 TB)的并行运算。简单来说,MapReduce 是一个基于集群的高性能并行计算平台,它是并行计算与运行相结合的软件框架,是一个并行程序设计模型与方法,它的出现实现了对大数据的自动并行化计算,并为程序员隐藏了系统层的细节^[12]。

MapReduce 并行计算框架是一个并行化程序执行系统,它提供了一个包含 Map 和 Reduce 两阶段的并行处理模型和过程,提供一个并行化编程模型和接口。MapReduce 就是任务的分解与结果的汇总。首先由 Map 将一个任务分解成为多个任务,然后由 Reduce 将分解后的多任务处理结果汇总起来,得到最终的分析结果^[13]。在分布式系统中,可以将机器集群看作硬件资源池,并行的任务被拆分后,交由空闲机器资源去处理,可以明显地提高计算效率,并且由于资源无关性,扩展计算集群得到了最好的设计保证。这样的功能划分使得 MapReduce 非常适合在大量计算机组成的分布式并行环境里进行数据处理。

MapReduce 以键值对数据输入方式来处理数据,并能自动完成数据的划分和调度管理。在程序执行时,MapReduce 并行计算框架将负责调度和分配计算资源,划分和输入输出数据,调度程序的执行,监控程序的执行状态,并负责程序执行时各计算节点的同步以及中间结果的收集整理^[14]。MapReduce 通过把对数据集的大规模操作分发给网络上的每个节点实现可靠性。每个节点会周期性地把自己完成的工作和状态的更新反馈回来,如果一个节点保持沉默超过一个预设的时间间隔,主节点会记录下这个节点的状态为死亡,并把分配给这个节点的数据发到别的节点。每个操作使用命名文件的不可分割操作以确保不会发生并行线程间的冲突,当文件被改名的时候,系统可能会把它们复制到任务名以外的另一个名字上去。

3 基于二叉树的频繁项集挖掘算法

文中提出的基于二叉树的并行频繁项集挖掘算法是在 MapReduce 并行化设计的思想上提出的,先通过 Map 阶段求出所有满足条件的候选 k -项集($k=1,2,\dots$),然后通过 Reduce 阶段将所求得的候选 k -项集进行统计,汇总得到最终的频繁 k -项集。由前面的频繁项集定义可知,某个频繁项集一定是数据库中某个事务的子集,所以在计算频繁项集时不一定需要知道整

个项集,只要根据事务数据库本身就能得到频繁项集。文中算法的基本思想是通过遍历二叉树快速找到候选子集(这与文献[15]中使用的模式指导树有所不同),再将候选子集的出现次数与阈值比较来完成对频繁项集的挖掘。

3.1 算法基本思路分析

为了方便描述算法思想,先假设数据库记为 D ,事务记为 T ,项集大小记为 k ,并设定一个阈值记为 S ,算法的基本思路如下:

- (1)扫描数据库 D ,将 D 中的每个事务 T 中所有大小为 k 的子集求出来;
- (2)统计输出所有子集的个数;
- (3)若某子集的个数超过了 S ,则可以认为该子集是频繁项集;
- (4)重复步骤(3),将上述所有满足频繁项集条件的子集输出即可。

从上述算法思路可以看出,找出所有大小为 k 的子集是该算法的关键,也是该算法的创新之处,下一节将介绍求取子集的具体算法。

3.2 遍历二叉树求取子集

设某个集合有 n 个元素,现要求该集合中大小为 k 的所有子集。在文中提出的算法中用一个高度为 n 的二叉树表示一个子集,二叉树中某一层的节点表示集合中对应位置的项,并且规定二叉树中某个节点的左孩子节点都用“0”表示,右孩子节点用“1”表示。假设初始状态的二叉树如图 1 所示,前 k 层都只有右孩子“1”,后 $n-k$ 层都只有左孩子“0”;

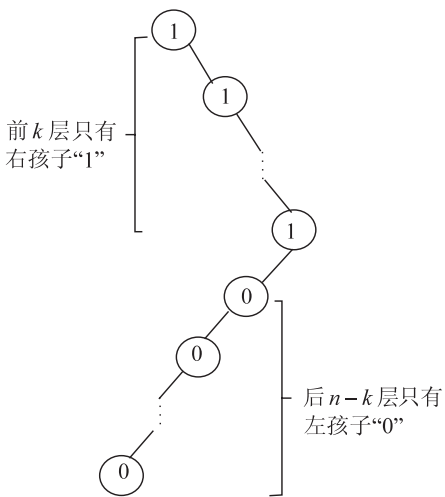


图 1 初始状态的二叉树

终止状态的二叉树如图 2 所示。前 $n-k$ 层都只有左孩子“0”,后 k 层都只有右孩子“1”;其他状态的二叉树都是中间转化的二叉树,每次得到一个状态的二叉树都需要将对应子集加入到结果集中。如果某一层有节点“1”,说明集合中对应位置的项属于该子集,反之不属于。

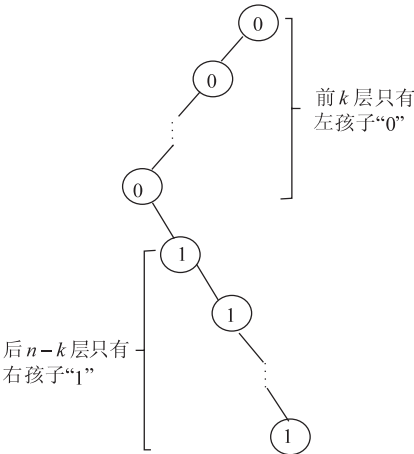


图 2 终止状态的二叉树

根据上述从初始二叉树到终止二叉树中节点的变化,可以发现其转化过程就是使“1”节点不断与下层“0”节点交换实现的。每次移动都有某一层的“1”节点与下一层的“0”节点进行交换,直至最下面一层的节点为“1”,此时便要重新计算需要从哪一层开始交换节点。具体算法过程如下:

- (1)在初始二叉树中寻找第一次出现某“1”节点有左孩子“0”的位置,若有就将“1”节点所在层定为第 i 层,转第二步,否则结束;
- (2)检查此时二叉树最下面一层的节点是否为“0”,若是则说明此轮交换还没结束,并且将第 i 层的“1”节点与第 $i+1$ 层的“0”节点进行交换,注意调整二叉树的树形,转第一步,若不是“0”则转第三步;
- (3)若此时二叉树最下面一层的节点为“1”,说明此轮交换已结束,必须重新计算交换的初始位置。具体计算方法如下:

- ①将此时二叉树中第 i 层(第一次出现某“1”节点有左孩子“0”的位置时“1”节点所在层为第 i 层)的“1”与其下面一层的“0”进行交换,并且 $i++$;
- ②将第 $i+1$ 层的“0”节点与最后一层的“1”节点交换,并且 $i++$,转第一步。

按上述算法执行,将得到一个最终的结果集,该结果集中的每一项都是要求的大小为 k 的候选子集。再根据这些候选子集在数据库中出现的次数与事先设定的阈值进行比较,就可以得到频繁 k -项集,至此挖掘结束。

4 实验结果与分析

综上可知,Apriori、CD、DD 算法都是通过不断地迭代来找到频繁项集的,而文中提出的算法(以下简称 FIMB 算法)并没有使用迭代的方式,并且 Hadoop MapReduce 在迭代处理上的性能较差,所以为了测试 FIMB 算法的性能,就将该算法与 CD 和 DD 算法进行

比较,并对 FIMB 算法的性能进行评估。实验中用到的测试数据集是通过数据生成器随机生成的,其特征如表 1 所示。

表 1 四个数据集的参数特征

数据集	事务数	项目数	事务平均长度
Data1	100 000	800	50
Data2	50 000	600	35
Data3	25 000	300	20
Data4	12 500	100	15

实验主要从两方面来进行比较。一方面,在相同的支持度和不同大小的数据集下比较两种算法的运行效率,实验时设最小支持度阈值 $\text{min_sup}=0.3$,所得实验结果用 MATLAB 绘制成折线图如图 3 所示。

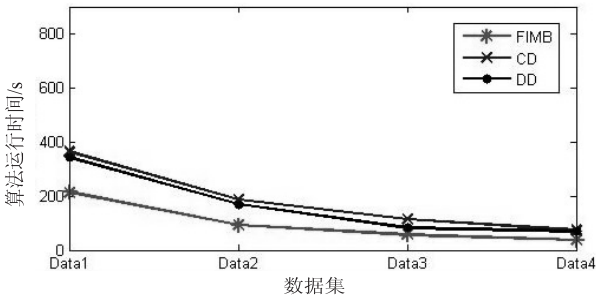


图 3 同一支持度、不同数据集下两种算法的运行时间结果

另一方面,在不同大小的支持度和相同的数据集下比较两种算法的运行效率,实验时使用 Data3 数据集进行实验,所得实验结果用 MATLAB 绘制成折线图如图 4 所示。

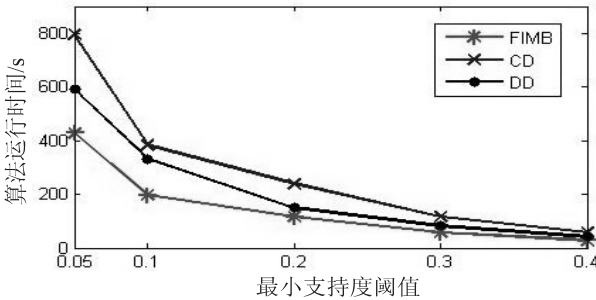


图 4 同一数据集、不同支持度下两种算法的运行时间结果

从上述实验结果可以看出,文中提出的基于二叉树的并行频繁项集挖掘算法在性能上优于 CD 和 DD 两种并行算法,提高了挖掘效率。

5 结束语

文中提出了一种基于二叉树的并行频繁项集挖掘算法,通过使用二叉树这种数据结构来求取候选子集,与 MapReduce 并行化计算的思想相结合,只需要一次
(下转第 87 页)

4 结束语

文中提出了一种基于分区的梯度投影算法。该算法对于图像灰度值变化不明显时,能有效提高算法精度。并且当被摄背景中存在运动物体时,通过分区的方法可以有效降低运动物体的干扰,从而提高精度。然而文中算法也存在一定的局限性,对于存在旋转运动时,该算法稳像效果不佳。

参考文献:

[1] 赵红颖,金 宏,熊经武. 电子稳像技术概述[J]. 光学精密工程,2001,9(4):353-359.

[2] Vella F, Castorina A, Mancuso M, et al. Digital image stabilization by adaptive block motion vectors filtering [J]. IEEE Transactions on Consumer Electronics, 2002, 48(3):796-801.

[3] 张 怡. 基于块匹配的电子图像稳定[D]. 哈尔滨:哈尔滨工程大学,2004.

[4] 王 畅,冯 驰. 基于块匹配的电子图像稳定算法[J]. 咸阳师范学院学报,2006,21(4):36-38.

[5] 张旭光,张媛媛,王春艳. 几种块匹配运动估计算法的比较[J]. 今日科苑,2007(18):251-251.

[6] Ko Sung-Jea, Lee Sung-Hee, Jeon Seung-Won, et al. Digital

image stabilizing algorithms based on bit-plane matching[J]. IEEE Transactions on Consumer Electronics, 1998, 44(3):617-622.

[7] 吕高杰,车 宏,赵 龙,等. 一种鲁棒性强的电子稳像方法[J]. 电光与控制,2010,17(1):57-60.

[8] 龚卫国,王小立,李正浩. 一种特征匹配的高精度电子稳像方法[J]. 计算机应用研究,2010,27(7):2751-2753.

[9] 韩绍坤,赵跃进,刘明奇. 电子稳像技术及其发展[J]. 光学技术,2001,27(1):71-73.

[10] 余 博,郭 雷,赵天云. 基于对数极坐标变换的灰度投影稳像算法[J]. 计算机应用,2008,28(12):3126-3128.

[11] Erturk S. Digital image stabilization with sub-image phase correlation based global motion estimation [J]. IEEE Trans on Consumer Electronics, 2003, 49(4):1320-1325.

[12] Ko S, Lee S H, Jeon S, et al. Fast digital image stabilizer based on gray-coded bit-plane matching [J]. IEEE Trans on Consumer Electronics, 1999, 45(3):598-603.

[13] 孙 辉. 快速灰度投影算法及其在电子稳像中的应用[J]. 光学精密工程,2007,15(3):412-416.

[14] 张国栋,王明泉,郭 栋. 基于灰度投影算法的实时电子稳像研究[J]. 微电子学与计算机,2010,27(10):53-56.

[15] 汪小勇. 基于灰度投影的实时电子稳像算法研究[D]. 杭州:浙江大学,2006.

(上接第 83 页)

MapReduce 过程即可完成挖掘,充分利用了集群的并行性,不需要进行迭代计算,在性能上明显优于 CD 和 DD 两种并行频繁项集挖掘算法,提高了挖掘效率。但是在实验中也发现,在 Map 阶段并行求取候选子集的时候,时间和空间耗费相对较大,这是以后需要进一步改进优化的地方。

参考文献:

[1] 刘 洁,杨路明,毛伊敏,等. 改进的数据流频繁闭项集挖掘算法[J]. 计算机工程,2011,37(9):75-77.

[2] Agrawal R, Imieliński T, Swami A. Mining association rules between sets of items in large databases[C]//Proceedings of ACM SIGMOD record. [s. l.]:ACM,1993:207-216.

[3] 颜跃进,李舟军,陈火旺. 频繁项集挖掘算法[J]. 计算机科学,2004,31(3):112-114.

[4] Agrawal R, Shafer J C. Parallel mining of association rules [J]. IEEE Transactions on Knowledge and Data Engineering, 1996, 8(6):962-969.

[5] 金 桃,何艳珊,宋伟国,等. 一种简单有效的并行化频繁项集挖掘算法[J]. 微计算机信息,2010(18):147-149.

[6] 王永恒,杨树强,贾 焰. 海量文本数据库中的高效并行频繁项集挖掘方法[J]. 计算机工程与科学,2007,29(9):110

-113.

[7] 戎 翔,李玲娟. 基于 MapReduce 的频繁项集挖掘方法[J]. 西安邮电学院学报,2011,16(4):37-39.

[8] Kotsiantis S, Kanellopoulos D. Association rules mining: a recent overview[J]. GESTS International Transactions on Computer Science and Engineering, 2006, 32(1):71-82.

[9] Han E H, Karypis G, Kumar V. Scalable parallel data mining for association rules[J]. IEEE Transactions on Knowledge and Data Engineering, 2000, 12(3):337-352.

[10] 王丹阳,田卫东,胡学钢. 一种有效的并行频繁项集挖掘算法[J]. 计算机应用研究,2008,25(11):3332-3334.

[11] 张 净,王惠文. 一种高效的并行频繁项集挖掘算法[J]. 计算机工程,2008,34(11):55-57.


[12] 涂子沛. 《大数据》正在到来的数据革命[J]. 求贤,2012(12):60-61.

[13] 李成华,张新访,金 海,等. MapReduce:新型的分布式并行计算编程模型[J]. 计算机工程与科学,2011,33(3):129-135.

[14] Dean J, Ghemawat S. MapReduce: a flexible data processing tool[J]. Communications of the ACM, 2010, 53(1):72-77.

[15] 张大为,黄 丹,嵇 敏,等. 利用模式指导树的并行频繁项集挖掘方法[J]. 计算机工程与应用,2010,46(22):147-150.

基于二叉树的并行频繁项集挖掘算法

作者: [陈静](#), [郑彦](#), [CHEN Jing](#), [ZHENG Yan](#)
作者单位: [南京邮电大学 计算机学院, 江苏 南京, 210003](#)
刊名: [计算机技术与发展](#) 
英文刊名: [Computer Technology and Development](#)
年, 卷(期): 2015(10)

引用本文格式: [陈静](#), [郑彦](#), [CHEN Jing](#), [ZHENG Yan](#) [基于二叉树的并行频繁项集挖掘算法](#)[期刊论文]-[计算机技术与发展](#) 2015(10)