

# 基于垂直频繁模式树带有负载均衡的分布关联规则挖掘算法

冯 勇, 尹洁娜, 徐红艳\*

(辽宁大学 信息学院, 沈阳 110036)

(\* 通信作者电子邮箱 fyxuhy@163.com)

**摘 要:** 大数据时代, 开展面向海量、分布数据的知识发现研究成为学界和业界关注的热点, 而负载均衡问题是开发分布式挖掘算法必须考虑的重要因素之一。为此, 提出了一种基于垂直频繁模式树带有负载均衡的分布关联规则挖掘算法, 算法采用垂直频繁模式树存储项及其关联而无需对局部挖掘结果进行合并, 减少了通信量, 简化了处理流程。同时所提出的算法采用混合体系结构即中心站点按照局部站点的处理能力分配任务, 实现了负载均衡, 提升了算法的性能。实验结果表明所提算法切实可行并具有较高效率。

**关键词:** 关联规则挖掘; 分布式; 垂直频繁模式; 负载均衡; 序列化

**中图分类号:** TP311.1 **文献标志码:** A

## Distributed rules mining algorithm with load balance based on vertical FP-tree

FENG Yong, YIN Jiena, XU Hongyan\*

(School of Information, Liaoning University, Shenyang Liaoning 110036, China)

**Abstract:** In mass data era, the research on knowledge discovery of massive and distributed data has become the hot spot in both academic field and industry. The problem of load balance is one of the important factors that must be considered in developing a distributed mining algorithm. Therefore, a distributed association rules mining algorithm with load balance based on vertical FP-tree (VFP-LBDM) was proposed in this paper. Vertical frequent pattern tree was used in this algorithm to store items and their associations, and there was no need to combine the local mining results. Therefore, the communication cost was reduced and the processing procedure was also simplified. At the same time, the algorithm used the hybrid architecture in which the central site assigned tasks according to the processing capacity of each local site. It realized the load balance and improved the performance of the algorithm. The experiment shows that the algorithm given in this paper is feasible and has higher efficiency.

**Key words:** association rules mining; distribution; Vertical Frequent Pattern (VFP); load balance; serialization

## 0 引言

大数据时代, 企业数据量激增且分布存储于不同站点, 这些数据蕴含着巨大的商业价值, 企业亟需利用数据挖掘工具从中获取竞争优势。分布关联规则挖掘因其过程简捷、结果易于理解等优势, 已成为应用最为广泛的挖掘方法<sup>[1-2]</sup>, 其适用性改进与完善一直为学界与业界所关注。

目前, 主流分布关联规则挖掘算法大多是为适应分布式数据环境对经典 Apriori 算法<sup>[3]</sup> 和频繁模式增长 (Frequent Pattern growth, FP-growth) 算法<sup>[4]</sup> 的改进, 如: Tseng 等<sup>[5]</sup> 提出将大型数据库按照相关性进行分块后再分配给各站点进行挖掘, 只在任务分配和结果返回时产生通信; 何波<sup>[6]</sup> 提出的基于频繁模式树的分布式关联规则挖掘算法 (Distributed algorithm for Mining Association Rules based on FP-tree, DMARF) 采用顶部和底部剪枝策略, 有效压缩了候选项集, 降低了通信量。以上算法在合并局部结果时都存在通信量开销较大的弊端, 有学者采用垂直频繁模式树结构对此问题予以解决, 如: Chen 等<sup>[7]</sup> 提出的基于 FP-growth 高性能并行算法

(High-performance Parallel algorithm based on FP-growth, HPFP) 通过使用垂直频繁模式树存储数据, 该算法避免了局部结果的合并, 但在进行并行处理时中心站点负载过大。在此基础上, 徐杰等<sup>[8]</sup> 提出的基于垂直 FP 树的并行频繁项集挖掘 (Parallel Frequent Itemsets Mining Algorithm Based on Vertical FP-tree, DVFP) 算法同时采用数据并行和任务并行策略来减少站点间的通信量, 但该算法仅对任务进行简单划分, 易导致负载不均衡。刘慧玲<sup>[9]</sup> 提出的一种基于投影树频繁模式 (Light Partial Support, LPS) 挖掘算法通过消除垂直频繁模式树节点的父亲-孩子指针, 降低算法的空间复杂度, 同时按各垂直频繁模式树的大小进行顺序和逆序交替的任务分配, 实现了一定程度的负载均衡, 但该算法对局部站点的处理能力欠缺考虑。

在现今物联网、云计算背景下开展分布关联规则挖掘研究需要考虑各类资源的综合利用。故此, 本文对基于垂直频繁模式树的分布关联规则挖掘中的负载均衡问题开展研究, 首先介绍了相关理论, 然后给出了基于垂直频繁模式树带有负载均衡的分布关联规则挖掘算法 (Distributed Rules Mining

收稿日期: 2013-08-13; 修回日期: 2013-10-18。

基金项目: 教育部人文社会科学研究青年基金资助项目 (12YJCZH048); 辽宁“百千万人才工程”培养经费资助项目 (2011921033)。

作者简介: 冯勇 (1973 -), 男, 辽宁沈阳人, 副教授, 博士, 主要研究方向: 数据挖掘、商务智能; 尹洁娜 (1988 -), 女, 吉林榆树人, 硕士研究生, 主要研究方向: 分布式关联规则挖掘; 徐红艳 (1972 -), 女, 辽宁丹东人, 副教授, 硕士, 主要研究方向: Web 挖掘、数据管理。

Algorithm with Load Balance based on Vertical Frequent Pattern tree, VFP-LBDM)。算法在各局部站点生成垂直频繁模式树,再由中心站点按照局部站点处理能力进行任务分配,局部站点接受任务后进行并行挖掘并共享挖掘结果;最后通过实验证明了 VFP-LBDM 具有更好的挖掘性能。

## 1 相关理论

### 1.1 分布式关联规则基本概念

**定义1** 频繁项集。设  $I = \{I_1, I_2, \dots, I_m\}$  是项的集合,  $D$  为事务数据库, 对于项集  $S$  且  $S \subseteq I$ ,  $S$  在  $D$  中的支持数是指  $D$  中包含  $S$  的事务项个数, 记为  $S.count$ ;  $S$  在  $D$  中的支持度是指  $S$  在  $D$  中出现频率, 记为  $S.sup$ 。如果  $S$  的支持度不小于给定的最小支持度阈值  $minsup$ , 则  $S$  为  $D$  中的频繁项集。

**定义2** 全局频繁项集。设  $D_i$  为站点  $P_i$  上的局部事务数据库, 且  $D = D_1 \cup D_2 \cup \dots \cup D_n$ , 在局部数据库中频繁的项集为局部频繁项集, 在  $D$  中频繁的项集为全局频繁项集。

**定义3** 最大上界。设  $I_i$  在站点  $P_j$  上为非频繁项, 则  $I_i$  的局部支持数  $I_i.count$  的最大上界为  $S_j.count - 1$ 。

**定理1** 若  $X$  为全局频繁模式, 则必存在站点  $P_i (1 \leq i \leq n)$ , 使得  $X$  在站点  $P_i$  为局部频繁模式。

**证明** 若不存在站点  $P_i$ , 使得  $X$  及其所有的非空子集在站点  $P_i$  上为局部频繁模式, 则  $X$  在任意站点的支持数  $X.count_i < minsup \times D_i$ , 有  $\sum_{i=1}^n X.count_i < minsup \times D$ , 这与  $X$  为全局频繁模式相矛盾。证毕

**推论1** 设  $D_i$  上的局部频繁  $k$  项集为  $L_k^i$ , 全局频繁项集为  $L_k$ , 则  $L_k \subseteq L_k^1 \cup L_k^2 \cup \dots \cup L_k^n$ 。

### 1.2 垂直 FP 树

**定义4** 垂直频繁模式树 (Vertical Frequent Pattern tree, VFP-tree) 是一种压缩存储数据的技术, 树的根节点不再为 “null” 而是为项 “ $I_i$ ”, 用  $VFP(I_i)$  来表示以频繁项  $I_i$  为根节点的垂直频繁模式树。

**定义5** 垂直频繁模式树林 (Vertical Frequent Pattern forest, VFP-forest) 即为垂直频繁模式树的集合, 由多棵垂直频繁模式树组成。

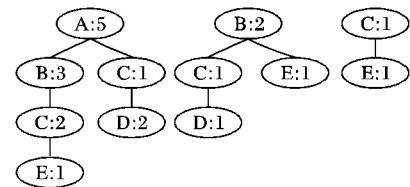
VFP-tree 通过将根节点定义为相应项, 使每棵树包含该项的全部信息, 因此各 VFP-tree 可开展并行挖掘。下面通过示例介绍 VFP-tree 的构造过程。给定一个事务数据库, 如表 1 所示, 设最小支持数为  $minsup\_count = 8$ , 可得到频繁 1 项集  $\{A:15, B:12, C:12, D:9, E:9\}$ 。设  $P_0$  为中心站点用于决策,  $P_1, P_2, P_3$  为局部站点, 其中事务  $T_1 \sim T_8$  存储于站点  $P_1$ , 事务  $T_9 \sim T_{16}$  存储于站点  $P_2$ , 事务  $T_{17} \sim T_{24}$  存储于站点  $P_3$ 。

VFP-tree 的构造包括两个部分: 1) 生成初始 VFP-tree。局部站点按照从中心站点计数统计的候选项支持数从大到小的顺序建立头表并生成局部 FP-tree, 将局部 FP-tree 的根节点 (null) 去掉, 生成由若干以项为根的 VFP-tree 组成的 VFP-forest,  $VFP_k(I_i)$  表示在站点  $k$  上以  $I_i$  为根的 VFP-tree。图 1 (a) 为站点  $P_1$  上生成的初始 VFP 树。2) 初始 VFP-tree 合并。合并的目的是使每棵 VFP-tree 包含该项的全部信息。本文在进行 VFP-tree 合并时, 按照 VFP-tree 根节点在头表中的顺序

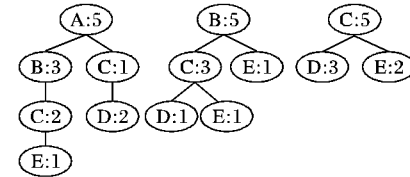
由上至下依次进行合并 (只对后继树中根节点为该树根节点的子节点进行合并)。运用这种合并方法, 每棵树只需要扫描一次。使用  $VFP(A)$  对  $VFP(B)$ 、 $VFP(C)$  进行更新, 再使用更新后的  $VFP(B)$  对  $VFP(C)$  进行更新, 直至全部更新完毕。合并的结果如图 1(b) 所示。

表1 事务数据库  $D$

TID	ITEMS	TID	ITEMS	TID	ITEMS
1	A, B, F	9	D, G	17	A, B, C, G
2	A, D	10	A, D	18	A, G
3	B, E, G	11	A, B, C	19	A, E, G
4	A, C, D	12	A, B, C, E, F	20	B, D
5	B, C, D, F, G	13	A, B, C, D	21	D, E
6	A, B, C, E, F	14	C, F	22	A, D, G
7	A, B, C	15	C, E	23	A, B, C
8	C, E	16	A, D, E	24	B, E, F



(a) 生成初始VFP-tree



(b) 初始VFP-tree合并

图1 站点  $P_1$  上 VFP-tree 的构造

## 2 VFP-LBDM 算法

为减少各站点间的通信量, Cheung 等<sup>[10]</sup> 提出了快速分布关联规则挖掘 (Fast Distributed Mining of association rules, FDM) 算法, 该算法中各局部站点之间只交换局部大项集的计数, 大大减少了传递的数据量。本文对 FDM 算法加以改进, 采用混合体系结构来完成站点间的数据通信, 即设置中心站点的同时允许各局部站点之间进行通信。中心站点负责决策的生成及任务的分配, 局部站点接受分配的任务开展局部挖掘, 并通过相互通信传递局部 VFP-tree, 该结构的优势在于: 可以避免中心站点成为瓶颈, 并且算法的并行化处理使其具有更好的可扩展性。VFP-LBDM 算法首先采用 VFP-tree 存储频繁项, 削减了局部结果合并环节, 同时在挖掘前统计各站点的处理能力; 然后按各站点处理能力生成决策表进行任务分配, 实现负载均衡; 最后, 局部站点按照决策表将序列化的 VFP-tree 发送给处理站点, 因此局部站点包含了相关项的全部信息, 则局部生成的规则即为全局规则。下面分别介绍算法的流程、算法的形式化描述及算法示例。

### 2.1 VFP-LBDM 算法流程

VFP-LBDM 算法由中心站点决策部分和局部站点挖掘部分构成, 具体算法流程如图 2 所示, 流程中核心环节的功能介绍如下。

#### 1) 计算站点处理能力。

为使分布式挖掘负载均衡, 中心站点在任务分配时需充

分考虑各局部站点的处理能力以及通信开销。鉴于 FP 算法的内存局限性<sup>[11]</sup>及对分布式环境下通信开销的考虑,本文采用式(1)计算局部站点的处理能力:

$$H = \alpha L_b + \beta L_m \quad (1)$$

其中: $\alpha, \beta$ 为权重系数,且 $\alpha + \beta = 1$ ;  $L_b$ 为网络带宽空闲率,其值为局部站点空闲网络带宽与全部站点中网络带宽最大值的比率;  $L_m$ 为内存空闲率,其值为局部站点空闲内存与全部站点中内存最大值的比率。

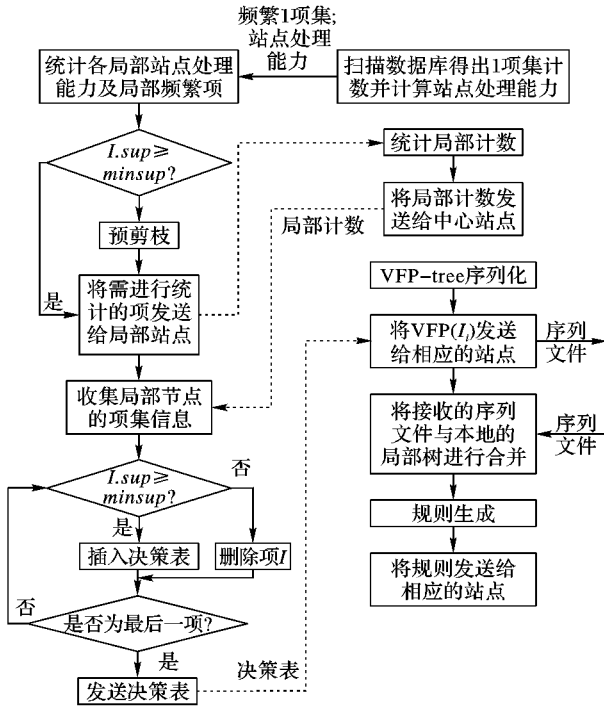


图2 VFP-LBDM 算法流程

## 2) 预剪枝。

中心站点在收到局部站点发送的频繁 1 项集后进行预剪枝,假设项  $I_i$  在非频繁站点  $S_j$  上的局部支持数均取该项局部支持数的最大上界,即  $I_i \cdot count = S_j \cdot count - 1$ 。如果按照最大上界计算的全局支持数仍小于全局最小支持数,说明该项一定为非频繁项,则不再收集该项的支持数信息,从而有效减少通信量。

## 3) 生成决策表。

中心站点根据各站点的处理能力形成决策表,首先将全局频繁项和各站点处理能力均按降序排序,再将各频繁项按序依次分配给相应的站点进行并行挖掘。决策表由频繁项、处理站点构成。

## 4) VFP-tree 序列化。

为压缩通信成本,本文对文献[8]所给的序列化方式加以改进实现站点间的数据传递。文献[8]所给序列化方式首先采用深度优先遍历整棵树,每个节点的输出为:项以及该项所对应的计数。如果该节点有子节点则输出“—”,没有子节点但有兄弟节点则输出“1”,如果也没有兄弟节点则输出“+”,并从该节点的上一层继续扫描。由于该序列化方式无法解决连续回溯问题,本文进行了如下修改:如果回溯时该节点没有兄弟节点,则输出“\*”。例如将图 1(b)的  $VFP_1(A)$  进行序列化为“A5—B3—C1—E1 + \* C2—D2 +”。序列化的结果存入到序列文件中,处理站点只需要对收到的序列文

件进行扫描来更新相应 VFP-tree。采用这种方法可以压缩树传递过程中的通信量,而且易于对局部树进行更新来生成全局 VFP-tree。

## 5) 规则生成。

对局部站点的  $VFP(I_i)$  树进行挖掘时,只生成由根节点项出发的相关规则。由于某一局部站点包含与  $I_i$  相关的全部信息,所以在规则生成的过程中,只需要对该局部站点上的数据进行挖掘即可得到与  $I_i$  相关的全部规则,在该站点上获得的即为全局规则,而不需要各局部站点间进行数据传递,从而减少了通信量。同时为了避免各站点产生冗余规则,本文在生成规则时只生成由根节点项出发的规则。各局部站点将生成的规则按照决策表发送给相应站点,使各站点包含所处理项的全部信息。虽然生成的规则需要进行共享,但是由于规则的共享只发生在与该规则所涉及项相关的站点间,所以通信量相对较小。

## 2.2 VFP-LBDM 算法描述

### 1) 局部站点挖掘部分。

输入 本地数据集  $D_i$ , 最小支持度  $S$ ;

输出 规则。

```

 $H_i = \alpha L_b + \beta L_m$ ; // 计算本地处理能力
Send  $\langle P_i, H_i \rangle$  to  $P_0$ ; // 将本站点的处理能力发送到中心站点
 $T_i = \text{get\_local\_count}(D_i)$  // 计算本站点各项计数
For all  $X$  in  $T_i$  do
    If  $X \cdot count_i > S \times D_i$  then // 判断各项是否频繁
        Insert  $\langle X, X \cdot count_i \rangle$  into  $LL_i$ ;
        // 将频繁项加入到局部频繁 1 项集  $LL_i$ 
send  $LL_i$  to  $P_0$ ; // 将  $LL_i$  发送至中心站点
Receive GC from  $P_0$ 

```

```

// 接收中心站点的计数请求, GC 为全局候选 1 项集
For all  $X$  in GC do
    Insert  $\langle X, X \cdot count_i \rangle$  into  $GC_i$ ;
Send  $GC_i$  to  $P_0$ ;
Receive dec_table from  $P_0$  // 接收中心站点的决策表
For  $j = 1$  to dec_table.length do // 根据决策表中项生成 VFP-tree
    create  $VFP(I_j)$ ; // 生成 VFP-tree
    File_VFP( $I_j$ ) = serial_vfp( $VFP(I_j)$ );
    // 将 VFP-tree 序列化为二进制文件
send File_VFP( $I_j$ ) to the  $P_k$ ;
    // 根据决策表将序列化后文件发送给相应站点
release  $VFP(I_j)$ ; // 对占用的空间进行释放
receive File_VFP( $I_k$ ) from  $P_k$ ;
    // 接收其他站点发送来的局部 VFP-tree

```

```

update  $VFP(I_k)$ ; // 更新 VFP-tree
rule_set = mining  $VFP(I_i)$ ; // 生成规则
if  $I_i$  in  $rule_i$  then send  $rule_i$  to  $P_k$  // 将相关规则发送给相应的站点

```

### 2) 中心站点决策部分。

输入 局部频繁 1 项集、站点处理能力;

输出 决策表。

```

For  $j = 1, 2, \dots, n$  do
    接收所有站点处理能力  $\langle P_j, H_j \rangle$ , 插入到处理能力统计表  $L\_map$  中;
     $L\_map = \text{Asc}(L\_map)$ ; // 降序排列处理能力
    For  $i = 1, 2, \dots, n$  do
        接收各站点的局部频繁 1 项集, 合并至全局候选项集 GC;
        For all  $X$  in GC do
            If  $X \cdot count < S \times D$  then // 预剪枝

```

```

If g_upper_bound(X) < S * D then //求支持数的最大上界
Delete X from GC;
Send GC to all P; // 全局候选项集 GC 广播到各局部站点
For j = 1, 2, ..., n do
接收各站点的局部候选项集并更新全局候选项集;
For all X in GC do
If X.count < S * D then
Delete X from GC; //将所有非频繁项删除,生成全局频繁项集
GC = ASC(GC); //将全局频繁集按降序排序
For j = 1 to GC.length do //将降序的 GF 中的每一项进行任务分配
将 GC 中的项依次分配给 L_map 中的站点,生成决策表 dec_table;
Send dec_table to all sites

```

### 2.3 VFP-LBDM 算法示例

为更好地说明 VFP-LBDM 算法的工作机理,下面对 1.1 节给出的例子加以分析,设  $\alpha = \beta = 0.5$ 。首先,各站点按照式(1)计算出本地的处理能力,发送给站点  $P_0$  用于决策的生成。其次,局部站点统计出本地局部支持数,得出局部的频繁 1 项集,即  $L_1^1 = \{A:5, B:5, C:5, D:3, E:3, F:3\}$ ,  $L_1^2 = \{A:5, B:3, C:5, D:4, E:3\}$ ,  $L_1^3 = \{A:5, B:4, D:3, E:3, G:4\}$ 。各局部站点将该局部频繁 1 项集信息发送给  $P_0$ , 站点  $P_0$  对各局部支持数进行累计,即  $L_1 = \{A:15, B:12, C:10, D:10, E:9, F:3, G:4\}$ , 其中 A、B、C、D、E 为全局频繁项直接进行全局频繁数的统计。对于 F、G 项,计算支持数的最大上界分别为  $F.maxcount = 3 + (3 - 1) \times 2 = 7 < minsup\_count$ ,  $G.maxcount = 4 + (3 - 1) \times 2 = 8 = minsup\_count$ , 其中 F 一定不会全局频繁,则不需要进行支持数的统计。而由于 G 的支持数最大上界等于全局支持数阈值,需要向  $P_1$ 、 $P_2$  站点请求 G 的支持数,经统计 G 的支持数小于全局频繁支持数阈值,因此 G 为非频繁项。最终生成的决策表如表 2 所示。将决策表发送给各局部站点,各局部站点将相应信息发送给处理站点,即  $P_1$ 、 $P_2$  将 VFP(A)、VFP(D) 发送给  $P_3$ ,  $P_2$ 、 $P_3$  将 VFP(B) 和 VFP(E) 发送给  $P_1$ , 而  $P_1$ 、 $P_3$  将 VFP(C) 发送给  $P_2$ 。

表 2 决策表

频繁项	处理站点	频繁项	处理站点
A	$P_3$	D	$P_3$
B	$P_1$	E	$P_1$
C	$P_2$		

最后,各站点使用从其他站点收到的序列文件对本地的 VFP 树进行更新。以项 A 为例,在  $P_2$  及  $P_3$  上生成的 VFP(A) 如图 3 所示,将  $P_1$  上的 VFP(A) 序列化为“A5—B3—C1—E1 + \* C2—D2 +”及  $P_2$  上的 VFP(A) 序列化为“A5—B3—C1|E1 + D2|E1 +”传送到  $P_3$ ,  $P_3$  对自身 VFP(A) 进行更新,使 VFP(A) 包含关于项 A 的全部信息。生成全局 VFP(A),如图 4(a)所示,对全局 VFP(A) 进行剪枝,将小于阈值的节点删除,得到图 4(b)。此时,由于 VFP(A) 包含了关于项 A 的全部全局频繁信息,可以直接对 VFP(A) 进行挖掘,而无需其他站点向其传递信息。再进行频繁模式的挖掘,生成规则时只生成包含根节点所对应项的规则,假设置信度为 0.8,则生成的规则为  $B \rightarrow A, A \wedge B \rightarrow C, A \wedge C \rightarrow B, B \wedge C \rightarrow A$ ,

生成的规则即为全局规则,并将  $B \rightarrow A, A \wedge B \rightarrow C, A \wedge C \rightarrow B, B \wedge C \rightarrow A$  发送至  $P_1$ ; 将  $A \wedge B \rightarrow C, A \wedge C \rightarrow B, B \wedge C \rightarrow A$  发送至  $P_2$ , 从而实现规则共享。

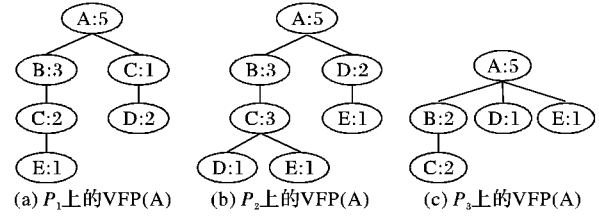


图 3 各站点上的 VFP(A)

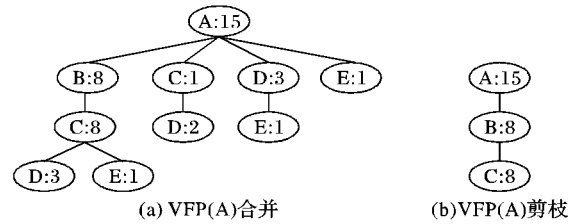


图 4 全局 VFP(A)

### 3 实验分析

实验在 4 台 CPU 主频为 2.39 GHz, 内存为 1.82 GB 的计算机组成的局域网环境下进行, 将其中一台计算机设为中心站点, 其他三台计算机设为局部站点。采用的数据集为 T40I10D100k, 该数据集包含 3960456 条事务, 数据集可以在 <http://fimi.ua.ac.be/data/> 获得, 实验中将数据集等量划分成三个部分, 分别存储于三个局部站点, 另设  $\alpha = \beta = 0.5$ 。

1) 不同数据量下运行时间的对比。局部站点数据量大小分别为 1000、5000、10000、50000、75000、100000 条事物时算法 VFP-LBDM 与一种低通信分布式挖掘 (Low Traffic Distributed Mining, LTDM) 算法<sup>[12]</sup>、DMARF<sup>[6]</sup> 算法在支持度 0.1 下运行时间的统计结果, 如图 5 所示。VFP-LBDM 算法在数据规模较小的情况下运行时间较长, 这是因为 VFP-LBDM 算法需要为每个频繁项建立 VFP-tree。随着数据规模的增大, 由于 VFP-LBDM 算法能够按照各站点的处理能力均衡分配任务, 使各局部站点资源得到充分利用, 运行时间增长相对缓慢。

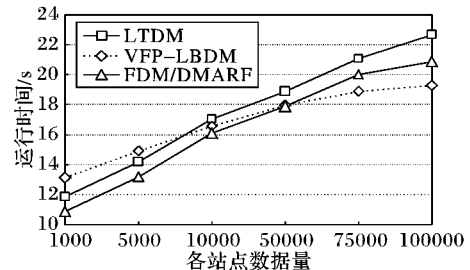


图 5 不同数据量下运行时间的对比

2) 不同支持度下运行时间的对比。VFP-LBDM 算法与 LTDM 算法、DMARF 算法在数据集 T40I10D100k 上执行时间的对比如图 6 所示。VFP-LBDM 算法在数据规模较大的情况下, 初期建立 VFP-tree 的时间相对算法总运行时间可忽略不计, 而负载均衡使算法在运行时间上具有明显的优势。当支持度较小时, 由于 VFP-LBDM 算法要为每个频繁项建立相应的 VFP-tree, 运行时间相对较长。

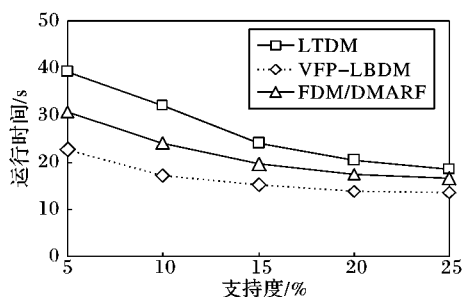


图6 不同支持度下运行时间的对比

通过实验分析可知,VFP-LBDM 算法在大规模数据集和较高支持度情况下运用更为有效。面向当今互联网应用广为普及的大数据时代,VFP-LBDM 算法具有良好的应用前景。

#### 4 结语

本文所给 VFP-LBDM 算法使用 VFP-tree 存储项及其关联,避免了局部挖掘结果合并所引发的通信和时间上的开销;同时该算法采用混合体系结构,并按照各站点的处理能力分配挖掘任务,使负载更为均衡。经实验证明所给算法在大数据集和较低支持度下具有良好的性能。

#### 参考文献:

- [1] GUO J F. Research on algorithms for mining association rules in distributed database [D]. Harbin: Harbin Engineering University, 2008. (郭俊凤. 分布式关联规则挖掘算法研究[D]. 哈尔滨: 哈尔滨工程大学, 2008.)
- [2] NI D. The study and implementation of several distributed algorithms for mining association rules [D]. Hangzhou: Zhejiang Gongshang University, 2009. (倪栋君. 分布式关联规则挖掘若干算法研究与实现[D]. 杭州: 浙江工商大学, 2009.)
- [3] AGRAWAL R, IMIELINSKI T, SWAMI A. Mining association rules between sets of items in large databases[C]// Proceedings of the 1993 ACM SIGMOD Conference on Management of Data. New York: ACM, 1993: 207-216.
- [4] HAN J W, PEI J, YIN Y. Mining frequent patterns without candidate generation[C]// Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. New York: ACM, 2000: 1-12.
- [5] TSENG F S C, KUO Y H, HUANG Y M. Toward boosting distributed association rule mining by data de-clustering[J]. Information Sciences, 2010, 180(11): 4263-4289.
- [6] HE B. Distributed algorithm for mining association rules based on FP-tree[J]. Control and Decision, 2012, 27(4): 618-622. (何波. 基于频繁模式树的分布式关联规则挖掘算法[J]. 控制与决策, 2012, 27(4): 618-622.)
- [7] CHEN X Y, HE Y S, CHEN P F. HPFP-Miner: A novel parallel frequent itemset mining algorithm[C]// Proceedings of the 5th International Conference on Natural Computation. Washington, DC: IEEE Computer Society, 2009: 139-143.
- [8] XU J, LI Y, LIU B, et al. A parallel frequent itemsets mining algorithm based on vertical FP-tree[J]. Computer & Digital Engineering, 2012, 40(10): 12-15. (徐杰, 李云, 刘博, 等. 基于垂直FP树的并行频繁项集挖掘[J]. 计算机与数字工程, 2012, 40(10): 12-15.)
- [9] LIU H. Design of frequent pattern mining algorithm LPS-miner and research on parallel formulations[D]. Lanzhou: Lanzhou University, 2009. (刘慧玲. 频繁模式挖掘算法 LPS-Miner 及其并行模式研究[D]. 兰州: 兰州大学, 2009.)
- [10] CHEUNG D W L, HAN J, NG V T, et al. A fast distributed algorithm for mining association rules[C]// Proceedings of the 4th International Conference on Parallel and Distributed Systems. Washington, DC: IEEE Computer Society, 1996: 31-42.
- [11] CHEN M, LI H. FP-growth parallel algorithm in cluster system[J]. Computer Engineering, 2009, 35(20): 71-72, 75. (陈敏, 李徽翡. 集群系统中的 FP-Growth 并行算法[J]. 计算机工程, 2009, 35(20): 71-72, 75.)
- [12] CAO W. Distributed association rules mining algorithm[J]. Computer Systems & Applications, 2012, 21(8): 218-221. (曹文梁. 一种分布式数据库关联规则挖掘算法[J]. 计算机系统应用, 2012, 21(8): 218-221.)
- [6] CHOPRA D, LEE S M, SU H H. On edge-balance index sets of wheels[J]. International Journal of Contemporary Mathematical Sciences, 2010, 5(53): 2605-2620.
- [7] CHOU C C, GALLARDI M, KONG M, et al. On edge-balance index of L-product of cycles with stars, part 1[J]. Journal of Combinatorial Mathematics and Combinatorial Computing, 2011, 78: 195-211.
- [8] LU J, ZHENG Y G. On the edge-balance index sets of  $B(n)$ [J]. Proceedings of the Jangjeon Mathematical Society, 2009, 12(1): 37-44.
- [9] ZHENG Y G, LU J, LEE S M, et al. On the perfect index sets of the Chain-Sum graphs of the first kind of  $K_4-e$ [C]// Proceedings of the 2nd International Conference on Intelligent Computation Technology and Automation. Piscataway: IEEE, 2009: 586-589.
- [10] WANG Y, ZHENG Y G, ADIGA C, et al. On the edge-balance index sets of  $N$  cycles three nested graph ( $N = 0, 1, 2 \pmod{6}$ ) [J]. Advanced Studied in Contemporary Mathematics, 2011, 21(1): 85-93.
- [11] YAO J, ZHENG Y G. On the quick construction of all edge-balance index sets of the graph  $C_n \times P_5$  [C]// Proceedings of the 2011 International Conference on Consumer Electronics, Communications and Networks. Piscataway: IEEE, 2011: 4227-4230.
- [12] ZHENG Y, YAO J. On the quick construction of all edge-balance index sets of the graph  $C_n \times P_{11}$  [C]// Proceedings of the 2011 International Conference on Consumer Electronics, Communications and Networks. Piscataway: IEEE, 2011: 4231-4234.
- [13] ZHENG Y, YAO J. Edge-balance index sets of nested graph with unlimited paths and equal circles(1) [J]. Journal of Shanghai Jiaotong University, 2013, 47(7): 1160-1163. (郑玉歌, 姚景景. 无限路等圈嵌套图边-平衡指数集的完全确定(1)[J]. 上海交通大学学报: 自然科学版, 2013, 47(7): 1160-1163.)

(上接第395页)