ISSN 1004-9037,CODEN SCYCE4
Journal of Data Acquisition and Processing Vol. 31,No. 1,Jan. 2016,pp. 223－230
DOI:10.16337/j.1004-9037.2016.01.025

http://sjcj.nuaa.edu.cn
E-mail:sjcj@nuaa.edu.cn
Tel/Fax:＋86-025-84892742

基于负载均衡和冗余剪枝的并行 FP-Growth 算法

刘祥哲[1,2]　刘培玉[1,2]　任　敏[1,3]　伊　静[1,4]　高　钊[1,2]

(1.山东师范大学信息科学与工程学院,济南,250014;2.山东省分布式计算机软件新技术重点实验室,济南,250014;3.山东财经大学数学与数量经济学院,济南,250014; 4.山东建筑大学计算机科学与技术学院,济南,250101)

摘　要: 针对现有的并行 FP-Growth 算法在数据并行分组时存在数据冗余和负载不均的问题,提出了基于负载估算和冗余剪枝的优化算法。首先,在采用高频策略分组时,引入节点任务估算方法,把每个分组中最大模式树的最长路径和支持度作为该分组的估计值,将估计值远大于其他节点的分组进行分割,平均到其他分组中,并且对不同分组中重复的列表元素进行截断,去除冗余数据。实验表明,本文提出的算法能够有效防止并行化的数据倾斜,减少数据冗余,在时间和空间复杂度上要低于以前的并行化 FP-Growth 算法。

关键词: 关联规则;MapReduce;冗余剪枝;FP-Growth 算法

中图分类号: TP311.13　　**文献标志码:** A

Parallel FP-Growth Algorithm Based on Load Balancing and Redundancy Pruning

Liu Xiangzhe[1,2], Liu Peiyu[1,2], Ren Min[1,3], Yi Jing[1,4], Gao Zhao[1,2]

(1. School of Information Science and Engineering, Shandong Normal University, Jinan, 250014, China; 2. Shandong Provincial Key Laboratory for Distributed Computer Software Novel Technology, Jinan, 250014, China; 3. School of Mathematics and Quantitative Economics, Shandong University of Finance and Economics, Jinan, 250014, China; 4. College of Computer Science and Technology, Shandong Jiazhu University, Jinan, 250101, China)

Abstract: Focusing on the data redundancy and load-unbalancing problems in the existing parallel FP-Growth algorithm, an improved algorithm for redundancy pruning and load balancing is proposed. Firstly, the improved algorithm introduces group task estimation method when using high-frequency strategies group. The longest path in the maximum pattern tree and the highest frequency are used as estimation. The group task will be averaged to others again when the group estimated is much larger than the value of other group. Then, the repetitive elements are removed in the list of different groups. Experimental results show that the improved algorithm avoides the data skew in the MapReduce and it is superior to the original one due to its high execution efficiency and speedup.

Key words: association rules; MapReduce; redundancy pruning; FP-Growth algorithm

---

基金项目: 国家自然科学基金(61373148)资助项目;国家社科基金(12BXW040)资助项目;教育部人文社科基金(14YJC860042)资助项目;山东省自然科学基金(ZR2012FM038,ZR2014FL010)资助项目;山东省优秀中青年科学家奖励基金(BS2013DX033)资助项目; 山东省社科规划项目(2012BXWJ01)资助项目;山东省高等学校科技计划(J15LN02, J15LN22)资助项目。
收稿日期: 2014-09-28;**修订日期:** 2015-10-22

引 言

随着大数据时代的到来,云计算技术越来越成熟,许多经典的数据挖掘算法进行了并行化改进,其中关联规则算法的并行化研究是其中热点之一,频繁项挖掘算法作为关联规则挖掘的基础与核心,在频繁模式挖掘方面取得的进展会对其他数据挖掘任务产生重要的影响。在实际应用中,Aprior 算法和 FP-Growth[1]算法较为常见。早在 1993 年,文献[2]就提出了 Aprior 算法,优点是思路比较直接,应用起来简单;缺点是需要多次扫描,其时空复杂度较高。以后有很多研究者提出了改进算法,如直接哈希剪枝的 DHP 算法[3],基于垂直数据挖掘的 ECLAT 算法和基于树形结构的 FP-Growth 算法[4],然而,面对大数据量时,需要重复扫描的 Aprior 算法内存消耗成指数型增长,ECLAT 算法不断生成的频繁二项集需全部放在内存之中,FP-Growth[5]算法生成的 FP-tree 非常大时也无法直接放入内存,文献[6]证明频繁项集挖掘是非确定性多项式(Non-deterministic polynomial, NP)难的完全问题,单服务器的挖掘已经远远不能满足需求,对传统算法的并行化改进成为流行趋势。文献[7]基于 MapReduce 模型在 Hadoop 平台上实现了并行的 Aprior 算法,2008 年文献[8]提出了并行 FP-Growth 算法,2014 年文献[9]实现了 MRECLAT 算法。然而现有的并行化算法存在数据冗余和负载倾斜问题,为保证子节点挖掘数据的完备性,每个节点都有相同事务项的完整备份,这必然存在一定的数据冗余,而且现有的平均划分数据集的负载均衡策略存在高频节点挖掘任务较严重的缺陷。

本文针对并行化算法中存在冗余问题,采用高低频分的负载均衡策略,对高频分组的事务项进行低频剪枝处理,按分界时的频率计数为界,去除事务中低频部分,减少数据冗余,而且在负载均衡处理上引入节点负载估算方法,将负载估算值较大的节点再进行分组,递归执行改进算法。通过实验证明本文方法能够有效减少并行化挖掘算法的执行时间。

1 FP-Growth 算法

1.1 基本概念

事物形数据库 DB,整个数据库是由许多事务 T 组成,每条事务又由 1 个或 n 个数据项 I 组成。事务集合 $T=\{T_1, T_2, T_3, \cdots, T_m\}$,项目集合 $I=\{I_1, I_2, I_3, \cdots, I_n\}$。

定义 1 支持度。项目集 $A \subseteq I$,若 $A \subseteq T$,则事务 T 支持 A,A 的支持度记为数据库 DB 中包含 A 的事务集合的百分比。最小支持度:预先设定的一个百分比,区分频繁项与非频繁项。

定义 2 频繁项集列表 F_List。存储数据库中所有大于最小支持度的项目集合和其支持度计数。

定义 3 极大频繁项集。极大频繁项集 $I_{极}=\{I_1, I_2, I_3, \cdots, I_n\}$ 是指该项集不存在大于最小支持度的直接超集。极大频繁项集的所有可能子集为 2^n-1 个。

定义 4 条件前缀。设通过 FP-tree 算法构建 FP 树,树中节点 I_n 到根节点路径上的所有父节点集合叫做条件前缀。所有 I_n 条件前缀集合叫条件模式基。所有条件模式基构造的 FP-tree 叫作条件模式树。

1.2 算法描述

FP-Growth 算法过程简述如下。

(1)扫描待挖掘数据集的所有事务支持度列表,然后降序排列每个事务中的项,剔除那些低于最小支持度阈值的项,这样就得到频繁 1 项集支持度列表 F_List。根据 FP-tree 算法[10],构造 FP-tree 和索引表 Header Table。

(2)对上一步构造的 FP-tree 进行挖掘。遍历表头项中的每一项,对于各项都执行以下操作:设 P 为 FP-tree 的叶子节点,从 FP-Tree 中找到所有 P 节点,并向上遍历它的祖先节点,得其路径,对于每一

条路径上的节点,其父节点的计数都设置为 P 的计数,得到 P 节点的条件模式基(Conditional pattern base, CPB),此时的后缀模式是(CPB|P),把上面的(CPB)当作 P 的子事务数据库,返回构造条件模式树 FP-tree,递归迭代运行。然后对其他节点按倒叙进行迭代挖掘,最后得频繁 K-项集。

FP-Growth 算法在内存中构造数据结构 FP-Tree 来表示输入的数据集,然后根据支持度对 FP-Tree 进行挖掘就可以得到频繁项集。虽然 FP-Growth 算法性能较高,但仍不能满足其对大数据挖掘的应用。FP-Growth 由于是树形数据结构,并行化处理不像 Apriori 表结构一般的容易划分,如果 FP-Growth 能够真正的独立进行并行化,那么就需要这些数据分区必须能够互相独立,也就是这些分区针对某一部分项目来说是完备的,在分组方式上本文采用 FP-close 中的分组方法[11]。但是如果数据库频繁闭项集分组中其估计值有负载过重的节点,需要对负载过重的节点再次进行任务分割,根据其估计值进行负载分配[12]。

2 改进的并行 FP-Growth 算法

2.1 相关概念

定义 5 频繁项本地分组。频繁 1 项集列表 F_List 中事务的所有元素都按频率降序排列,按数据项计数的频率分成相应的几个分组,每个赋予唯一的 id 形成一个 G_List。

定义 6 最大模式树。本地频繁项集列表 G_List 的首次投影 FP-tree 为本节点的最大模式树,所有叶子节点中前缀最多的路径为该模式树的最长路径。

定义 7 分组节点负载估计值。将本地上最大模式树中最长路径 n 和该路径上所有节点的频率计数 m 相加作为节点负载估计值,本地负载估计值 load=$n+m$。

推理 若分发到低频分组 group2 的事务也将会被分发的到高频分组 group1 中时,group1 删除包含在 group1⋂group2 的数据不会破坏本节点内频繁项挖掘的完整性。

假设 F_list = {(f:4),(c:4),(a:3),(b:3), (m:3), (p:3)},这里把它分割 2 个组 G_list={ {group1:(f:4), (c:4), (a:3) }和{group2: (b:3), (m:3), (p:3) } },所有的事务里的数据项都是按照 F_list 的支持度降序排列,那么 group 的分布情况如下。哈希表 H 存储 Key-Value 对的形式如表 1 所示。

表 1 Hash 表

Tab. 1 Hash table

Key	value
f	group1
c	group1
a	group1
b	group2
m	group2
p	group2

group1 和 group2 的分界线是遍历到 a 项时哈希表中出现的频率,group1 里所有的 item 支持度都大于等于 3,group2 所有 item 支持度都小于或等于 3。group2 的 FP 挖掘事务 T1 中的极大频繁项集,包含所有 item 序列,比如{b:3, bm:2, pm:2, fb:3,…}等。但这个极大频繁项集中,包含{b, m, p}的元素的子集支持度都不会超过 3,因为在 F_List 中,这 3 个元素出现的次数都不大于 3。在 group1 中挖掘事务 T1 中剩余元素的序列,比如{f:4,fc:3,ca:3,…},在 F_List 统计时,这个序列里元素的支持度都不小于 3。在 group1 做挖掘时删除了包含{b, m, p}的 item 组合。这个推论成立的前提是所有 F-List 里的元素排列都必须按一致的顺序,这个顺序的意义在于,所有的 transaction 都是遵循一致的先后顺序来进行截断,从而分配到不同的 group 中。本质上,可以删掉事务中频率低的一些元素的原因是 group1⋂group2 的集合已经包含在先分发到 group2 的事务里,所以 group1 里不需要再包含 group1⋂group2 的数据。

2.2 算法改进

基于数据冗余剪枝和负载均衡优化改进的并行 FP-Growth 算法思想如下:

(1)数据冗余剪枝,去除数据冗余。第 1 次扫描数据集,将每条事务中的数据项按支持度降序排列

的频繁 1-项集列表 F_List = {I1:count1, I2:count2, I3:count3,…},其中 count1> count2 > count3>…,然后将 F_List 中的条目按上文所述方法分成几个 Group,给每个 Group 分配 1 个 ID 并将此组的数据项放入 G_List。第 2 遍扫描数据库中的每一条事务,如果这条事务中包含 1 条 G_List 中的 Item,那么这条事务就被添加到该 Group 对应的模块中去。这种分组方法使得同一条事务可能会被重复分到不同的 G_List 里面,进而导致分组数据中存在大量冗余,本文将事务在分配前进行处理,采用高低频分组的负载均衡策略,对不同分组中重复的列表元素进行截断,删除高频分组表中低频部分的元素,降低数据冗余。

(2)负载均衡优化。首先引入负载任务估算方法,把分组节点中条件模式树最长路径和该路径上的频率计数作为该节点负载的估计值,统计所有分组节点任务的估计值,然后将所有任务按估计值进行平均分组,避免大部分数据任务分配到一个分组节点中出现数据倾斜的情况,可有效解决高频复杂的事务型数据库某些节点负载过重的问题。

2.3 算法描述

算法的步骤如下所示。

步骤 1 数据分割。把数据库平均分割发送到各个节点上,每个节点上的部分数据称之为一个 shard。

步骤 2 并行统计。收集每个 share 的计数,计算 F_list 中每个数据项的支持度,得到全局数据项的支持度列表,根据给定的最小支持度,删除计数小于最小支持度的数据项,最后将所有数据项和对应的支持度存储在 F_List 中。

步骤 3 事务分组和冗余剪枝。分组方法是将包含该事务的 G_List 分配到 Group 对应的模块中去,每个组都被分配一个 group_id。

步骤 4 负载均衡优化并行 FP-Growth,这是并行化算法最重要的一步。(1) Mapper 过程。先读入 G-List,将所有 group_id 相同的的 item 集合读入到同一个节点上,形成针对本节点所有 item 的完备数据集,对完备的数据集进行数据冗余剪枝,输出剪枝后的事务集,记为 Key-Value 对,key 保存 group_id,Value 保存事务集,然后对所有分组节点的负载任务进行估计;统计所有分组节点估计值,将任务进行重新分配。(2) Reduce 过程。操作 Mapper 输出的数据集,对于每一个节点的 Reduce 建立一个本地局部的 FP-tree,在本地运行 FP-Growth 算法。此步骤的伪代码如下:

Procedure: Mapper (key, value=T_i)
输入:所有事务中的项
过程:Mapper 过程是将每个事物中项分配到分组节点中
输出:每个节点中的本地 Hash 头表和本地链表
Output〈Hash Num, a[0]+…+a[j]〉=Output〈group_id,a[0]+…+a[j]〉;
end
Procedure: Reducer (key=gid, value=DB_{gid})
输入:本地 G_List
过程:将本地 FP-Growth 挖掘结果暂存为局部频繁项集
输出:局部频繁集的临时数据 Temporary results

步骤 5 将所有结果聚集。各个节点将自己挖掘出来的局部频繁项集结果发送到其他节点,每个节点把接收到的局部频繁闭项集保存到一个堆(Heap, HP)中,然后遍历堆 HP 判断所有局部频繁项集

的全局性,删除非全局的频繁项集,输出全局频繁项集。此步骤的伪代码如下。

Procedure: Mapper (key, value=v+supp(v))
输入:Temporary results
过程:自己挖掘出来的局部频繁项集结果发送到其他节点
End
Procedure: Reducer (key=a_i, value=S(v+sup(v)))
输入:各个节点都接收其他节的局部频繁项集
过程:每个节点把接收到的局部频繁闭项集保存到一个堆(HP)中,遍历堆(HP)生成全局频繁项集。
输出:全局频繁项集
end

2.4 算法分析

改进的 FP-Growth 算法基于 MapReduce 模型将原始的数据库细化为几个模块,这些模块将被分配到各个不同的节点,在每个节点上运行本地的 FP-Growth 算法,这样以并行的方式来挖掘数据库中的频繁项,把用时最多的节点挖掘时间作为并行算法总的挖掘时间。又因为循环函数的时间复杂度 $O(f(n))$要小于递归函数的时间复杂度 $O(f(n)\log_2 n)$[13](当 $f(n)$的递归次数大于 2 时,$\log_2 n>1$),本算法中的负载均衡算法用时最多的步骤要数负载估算的步骤,但其最大模式树的深度遍历的时间复杂度要远小于所有条件模式树的递归查找所有条件模式基的规模,所以节点负载估计值的引入不会增加本节点内的算法复杂度,但是引入节点任务估算方法后,对每个分组的负载进行估计,将负载远大于其他节点的异常分组,再进行一次并行化算法调用,均分负载过重节点的任务量就可以有效减少并行化算法的某节点负责过重的问题,可有效降低并行化算法关键节点的最大运行时间;而且冗余剪枝算法删除了高频分组中的低频数据,在项数较多的事务中能够有效减少条件模式基和条件子树的数目,有效降低挖掘算法最费时环节的规模,可以从空间复杂度上降低内存消耗[14]。基于上述两点改进,本算法可有效提高并行化 FP-Growth 算法的时空效率。

本文是对 Mahout 开源项目中的并行化 FP-Growth 算法进行了改进,在 Map 阶段减少了分组内的数据冗余,但算法不会删除分布式计算框架中的作为备份的拷贝数据,算法对冗余数据的删除不会影响到算法在分布式框架中的稳定性。首先对冗余数据的判断条件是,分组开始时确定支持度的值,根据支持度划分分组高频和低频分组,高频分组不需要挖掘低频分组部分,同样在低频分组,不用挖掘高频部分,在分组节点中暂时删除的数据会在另一节点中存在,其次算法去冗余的地方是在高频分组中减掉低频部分的内容,但被减掉的内容会存在在低频分组中,最后通过堆递归将所有频繁项聚合到一起,不会造成数据丢失[15]。

3 实验结果与分析

本文采用 6 台 PC 机构成的 Hadoop 集群(1 个 Master 节点,5 个 Slave 节点),单个 PC 机的配置:处理器 Intel®Core(TM) CPU i3-3220,主频为 2.60 GHz,操作系统为 CentOS6.5,软件版本:Hadoop 为 1.2.1;Mathout 为 0.8;开发环境为 Myeclipse10.0,算法用 JAVA 语言实现。

3.1 实验步骤

实验用数据集来自 IBM Almaden Quest 研究组提供的频繁项集挖掘数据集 T10I4D100K (.dat) T40I10D100K (.dat)[16]和 Accidents 数据集。表 2 为 3 个数据集的特征,Dataset 列表示 3 个标准数据集,Items 表示每个数据集内所有数据项的种类,Trans 表示每个数据集中包含的事务的总条数,Avg 表

示数据集中每条事务中包含数据项的平均数。Accidents 数据集的事务中包含数据项的平均数明显比其他两个数据集要多,但数据项的种类(Items)没有其他两个数据集的多。

表 2 数据集特征

Tab. 2 DataSet features

Dataset	# Items	# Trans	Avg|*T*|
Accidents	468	340 183	33.8
T10I4D100K	870	100 000	10.1
T40I10D100K	990	100 000	24.9

本文实验采用 Aprior 和 FP-Growth,Eclat 算法的并行改进算法进行对比实验。实验中对长事务型数据集(每条事务中包含的数据项较多)进行频繁项挖掘时,Aprior 算法重复扫描数据集次数非常多,构造的候选项集较多,需要占用大量内存,挖掘时间都比较长,数据量稍微增大,挖掘时间和需要占用的内存成指数型增长,常出现溢出。因此本实验未将并行的 Aprior 算法在实验图中进行显示,只是将改进的并行 FP-Growth 算法和之前的并行 FP-Growth 算法及 MREclat 算法进行了显示。

3.2 改进的内存对比

改进的内存方面实验结果如图 1 所示。本文改进算法标记为 SPFP-growth 算法,未改进的并行 FP-Growth 算法标记为 PFP-Growth,并行 Eclat 算法标记为 MReclet,4 种算法的内存资源消耗的对比结果,在内存资源消耗较高的 Accidents 数据集上 SPFP-Growth 算法的挖掘曲线位于图中最低端,在相同支持度的条件下 SPFP-growth 算法的内存消耗最小,这是因为在进行频繁项挖掘时,SPFP-growth 算法进行了冗余剪枝,减小了 FP-tree 占用内存的规模。

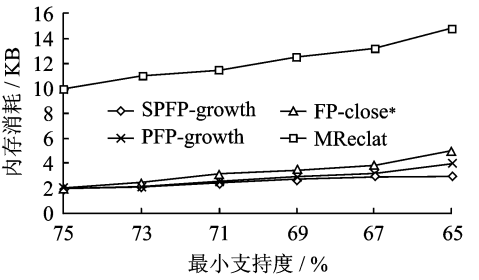

图 1 在 Accidents 数据集上的内存消耗对比

Fig. 1 Memory usage comparison for Accidents dataset

3.3 改进的时间对比

表 3 所示是本文的 SPFP-Growth 算法在不同支持度情况下不同节点上的运行时间。当支持度由 1%增至 3%时,挖掘时间减少,这是由于支持度增大,候选的频繁项减少,所需挖掘 FP-tree 就减小,而且在频繁项分布不均的 T10I4D100K. dat 数据集上进行分节点挖掘时每个节点上运行时间比较平均,不会出现数据倾斜的情况。这是因为在并行化挖掘时,本文进行分布式节点的任务估计,再进行任务平均,该方法即缩短算法的整体挖掘时间,而且有效利用分布式其他节点的资源,实验结果证明本文算法在降低内存消耗和解决数据倾斜现象有较好的表现。

表 3 在 T10I4D100K. dat 数据集上的运行时间

Tab. 3 Run time for T10I4D100K. dat ms

支持度/%	1 号节点	2 号节点	3 号节点	4 号节点	5 号节点
3.0	43	41	35	28	44
2.5	130	117	114	95	140
2.0	254	210	206	236	268
1.5	451	406	370	449	410
1.0	1 084	950	764	864	1 120

改进的并行化 FP-Growth 算法在数据集 T40I10D100K. dat 进行实验,设置最小支持度为 1%。实

验的分组节点与执行时间之间的关系如图 2 所示。图 2 中表示每当增加一个节点时,频繁项挖掘算法挖掘时间减少,图中曲线斜率变化速度是先增后减,说明并行节点的加速比是先增后减。这是因为分布式节点数增加时,挖掘任务被平均分配,算法挖掘时间会减少,但是当节点数量增加,节点间的通信开销也会增加,当通信开销增加到一定程度时,反而会延长挖掘时间。

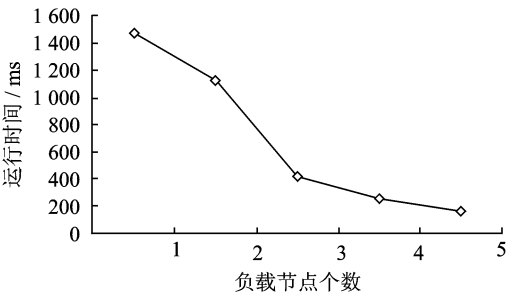

图 2 算法运行时间与负载节点数之间的关系

Fig. 2 Relationship between algorithm running time and load nodal point number

图 3,4 是 4 种改进的并行算法在两个标准数据集上的挖掘时间对比,符合支持度越大,频繁项越少,并行的频繁项挖掘算法运行时间开销越小,本文的改进算法在数据项较多的长事务型数据集上有更好的表现。这是因为在长事务型数据集上,删除分配到不同分组的相同事务中的低频交集元素更多,去数据冗余效果更明显,运行时间更短,优化效果更好。

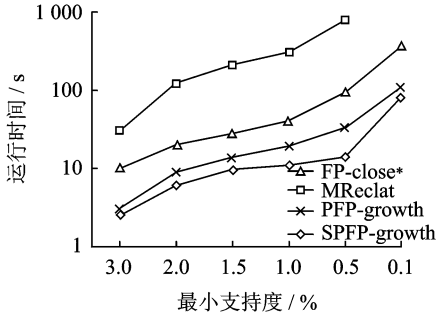

图 3 在 Accidents.dat 上的运行时间

Fig. 3 Running time in Accidents.dat

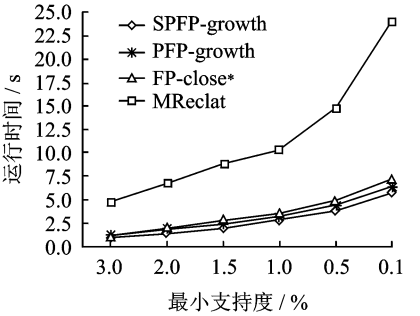

图 4 在 T40I10D100K.dat 上的运行时间

Fig. 4 Running time in T40I10D100K.dat

4 结束语

基于负载均衡和 G_list 优化剪枝的并行 FP-Growth 算法利用 F_List 中的事务必须按照统一的顺序排列这一特点,在并行化分组时按频率高低分组策略进行负载均衡调整,同时在第 2 次 MapReduce 操作前舍去一些事务中的冗余数据,减少空间复杂度,从而降低算法的资源开销,并且改进并行算法的负载均策略,引入前缀路径长度和最高频率为估计值,将负载过重的节点通过多次频分分割,将挖掘任务进一步分割,递归调用优化的并行 FP-Growth 算法。最后聚合频繁项集结果,实验结果表明,本算法在时间和空间复杂度上要低于先前的并行化 FP-Growth 算法,而且在长事务型数据库上的优化效果更明显。

参考文献:

[1] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation[C]// ACM SIGMOD Record. [S.l.]: ACM, 2000,29(2):1-12.

[2] Agrawal R, Srikant R. Fast algorithms for mining association rules[J]. Proc 20th Int Conf Very Large Data Bases (VLDB), 1994,23(3):21-30.

[3] Park J S, Chen M S, Yu P S. An effective hash based algorithm for mining association rules[M]. SIGMOD Record: ACM, 1995:175-186.

[4] 滕翠, 梁川. 三种频繁模式挖掘算法的分析与比较[J]. 电脑知识与技术, 2010,6(23):6416-6417.

Teng Cui, Liang Chuan. Analysis and comparison of three kinds of frequent pattern mining algorithms [J]. Computer Knowledge and Technology, 2010,6(23):6416-6417.

[5] Han J, Pei J, Yin Y, et al. Mining frequent patterns without candidate generation: A frequent-pattern tree approach[J]. Data Mining and Knowledge Discovery, 2004,8(1):53-87.

[6] Gunopulos D, Khardon R, Mannila H, et al. Discovering all most specific sentences[J]. ACM Transactions on Database Systems (TODS), 2003,28(2):140-174.

[7] Yu H, Wen J, Wang H, et al. An improved Apriori algorithm based on the Boolean matrix and Hadoop[J]. Procedia Engineering, 2011, 15: 1827-1831.

[8] Li H, Wang Y, Zhang D, et al. Pfp: Parallel FP-Growth for query recommendation[C]// Proceedings of the 2008 ACM Conference on Recommender Systems. [S. l.]: ACM, 2008:107-114.

[9] 章志刚，吉根林，唐梦梦. 并行挖掘频繁项目集新算法——MREclat[J]. 计算机应用，2014,34(8):2175-2178.
Zhang Zhigang, Ji Genlin, Tang Mengmeng. MREclat: New algorithm for parallel mining frequent itemsets[J]. Journal of Computer Applications, 2014,34(8):2175-2178.

[10] 陈慧萍，王建东，叶飞跃. MAXFP-Miner:利用 FP-tree 快速挖掘最大频繁项集[J]. 控制与决策，2005,20(8):887-891.
Chen Huiping, Wang Jiandong, Ye Feiyue. MAXFP-Miner: Mining maximal frequent itemsets efficiently by using FP-tree [J]. Journal of Control and Decision, 2005,20(8):887-891.

[11] 谈克林，孙志挥. 一种 FP 树的并行挖掘算法[J]. 计算机工程与应用，2006,42(13):155-157.
Tan Kelin, Sun Zhihui. An algorithm of mining FP-tree in parallel[J]. Computer Engineering and Applications, 2006, 42 (13): 155-157.

[12] 曾志勇，杨呈智，陶冶. 负载均衡的 FP-Growth 并行算法研究[J]. 计算机工程与应用，2010,46(4):125-126.
Zeng Zhiyong, Yang Chengzhi, Tao Ye. Research of load balance FP-Growth algorithm in parallel[J]. Computer Engineering and Applications, 2010,46(4):125-126.

[13] 王红梅，应红霞，季绍红. 递归函数时间复杂度的分析[J]. 东北师大学报：自然科学版，2001(4): 111-113.
Wang Hongmei, Ying Hongxia, Ji Shaohong. Analysis of recursive function's time complexity[J]. Journal of Northeast Normal University, 2001(4): 111-113.

[14] 郭磊，侯维刚. 云数据中心网络中的先应式碎片整理算法[J]. 数据采集与处理，2015,30(3):519-527.
Guo Lei, Hou Weigang. Provident resource defragmentation algorithm for cloud data center network[J]. Journal of Data Acquisition and Processing, 2015,30(3):519-527.

[15] 亓峰，唐晓璇，邢宁哲，等. 未来大数据环境下的配用电通信网虚拟网络架构及应用[J]. 数据采集与处理，2015,30(3):511-518.
Qi Feng, Tang Xiaoxuan, Xing Ningzhe, et al. Virtual network architecture and application for smart distribution grid in future large data environment[J]. Journal of Data Acquisition and Processing, 2015,30(3):511-518.

[16] Frequent itemset mining implementations repository. Frequent itemset mining dataset repository: FIMI dataset[EB /OL]. http://fimi. cs. Helsinki, 2013-04-20.

作者简介：

刘祥哲(1987-)，男，硕士研究生，研究方向：数据挖掘与处理，E-mail: 744435403@qq. com。

刘培玉(1960-)，男，教授，博士生导师，研究方向：数据挖掘与处理和信息安全等。

任敏(1979-)，女，博士研究生，研究方向：网络安全数据挖掘与处理等。

伊静(1990-)，女，讲师，研究方向：网络安全和模式挖掘。

高钊(1990-)，男，硕士研究生，研究方向：网络安全取证、数据分析与处理等。