

基于 MapReduce 的并行加权 FIUT 算法

张 航, 张 欣, 张平康, 李 琪

(贵州大学 大数据与信息工程学院, 贵州 贵阳 550025)

摘 要: 针对目前大数据环境下, 传统频繁项集挖掘算法效率低下的问题, 在 MapReduce 框架的基础上, 结合加权模型提出了一种并行加权频繁项集挖掘算法 PWFUT (Parallel Weighted Frequent Itemset Ultrametric Tree). 该算法将候选项映射到哈希表中以划分支持度, 同时避免构建条件模式和实现压缩存储. 最后, 对 PWFUT 算法在 Hadoop 平台进行了测试与分析, 实验结果表明所提出的算法具有较好的运行效率和扩展性.

关键词: 大数据; 关联规则; MapReduce; 加权模型; FIUT

中图分类号: TP301.6

文献标识码: A

文章编号: 1000-7180(2018)07-0041-04

Parallel Weighted FIUT Algorithm Based on MapReduce

ZHANG Hang, ZHANG Xin, ZHANG Ping-kang, Li Qi

(College of Big Data and Information Engineering, Guizhou University, Guiyang 550025, China)

Abstract: Aiming at the the inefficiency of traditional frequent itemsets mining algorithm in view of the big data environment, a solution to this problem parallel weighted mining of frequent itemsets using PWFUT (Parallel Weighted Frequent Itemset Ultrametric Tree) algorithm is implemented on MapReduce framework. Support is counted by mapping the items from the candidate list into the buckets which is divided according to support known as Hash table structure, also to avoid building conditional patterns and to achieve compressed storage. Finally, the Algorithm is verified and analyzed on Hadoop platform. According to the compared experiment results, it shows that the proposed algorithm has high efficiency and good scalability.

Key words: big data; association rule; MapReduce; weighted model; FIUT

1 引言

关联规则作为数据挖掘 KDD 研究的重要方面之一, 主要用于对大量数据中项集之间进行相关分析, 是数据挖掘领域重要的研究方向之一^[1]. 在目前大数据的实际应用中, 单机平台面临海量数据在存储能力、处理能力和挖掘上都面临显而易见的瓶颈. 而通过并行化的方式, 不仅能有效解决内存不足的问题, 还能极大地提升挖掘效率. 另一方面, 现实中数据各项属性的重要程度往往不尽相同, 按各项属性权重相同进行数据分析, 其结果可能有所偏差. 因此通过在关联规则挖掘时引入加权模型, 可以更好

地提升数据使用价值满足, 决策者的侧重要求. 综上, 将并行处理的思想与增加权重的方法进行结合利用具有重大的现实意义.

近年来, 并行、分布式的关联规则挖掘算法由于大数据的兴起而越来越成为学界、实务界的关注热点. 结合 MapReduce 编程模型, 研究者们从减少 MapReduce 任务数、候选项集数量、并行节点之间的通信量等多个角度对并行化算法效率进行了提升^[2]. 朱晓峰等基于 MapReduce 框架提出了一种 MRFUP 算法^[3], Li 等人提出了一种并行 FP-Growth 算法^[4], 分别地实现了 FUP 算法和 FP-

收稿日期: 2017-10-20; **修回日期:** 2017-11-17

基金项目: 国家国际科技合作专项项目(2014DFA00670); 贵州省研究生教育教学改革重点课题(黔教研合 JG 字[2016]15); 贵州省科技厅工业攻关项目(黔科合 GY 字[2010]3056)

Growth 算法在 MapReduce 计算框架上的移植. 杨勇等人提出基于 MapReduce 的并行增量更新算法, 通过引入分块索引提高挖掘效率, 同时采用负载均衡策略保证集群节点之间的负载均衡^[5].

针对目前将加权关联规则移植到 MapReduce 平台实现并行挖掘并进行效率优化提升的问题, 本文设计并实现了一种基于 MapReduce 计算框架的并行加权 FIUT 算法. 该算法通过对传统 FIUT 算法进行并行化加权改进, 将其移植到 MapReduce 计算框架上, 使其通过三个 MapReduce 任务完成分布式环境下对加权 FIU-tree 的构建和挖掘, 提高了并行计算效率. 文章最后, 通过实验证明了该算法的有效性和可扩展性.

2 相关概念

2.1 FIUT 算法

FIUT 算法是由 Yuh-Jiuan Tsay, Tain-Jung Hsu 等于 2009 年提出的一种用于在数据库中的频繁项集挖掘的算法^[6]. FIUT 算法基本类似于 FP-Growth 算法, 都是通过两次扫描数据库以得到频繁项集, 但其所有的频繁项的挖掘与 FP-Growth 算法通过递归挖掘不同, 其主要思想及步骤可分为三个步骤, 第一步通过两轮扫描数据库获取生成频繁 1-项集和 k -itemsets, 其中 k 代表每件事务中剩余的频繁项的数量; 第二步则是利用分解 h -itemsets 到 k -itemsets 中, 其中 $k+1 < h < M$, 合并上原始的 k -itemsets 以建立 k -FIU-tree; (3) 最后, FIUT 通过检查所得的 k -FIU-tree 上的叶子节点的计数挖掘所有的频繁项 k -项集.

2.2 MapReduce 编程模型

MapReduce 是 Apache 旗下的用于处理和产生大数据集的并行计算编程模型. 其主要原理是通过分治的思想将数据分成不同分区, 把每一个作业分解成为更小的任务并行处理各个分区. 其利用一个输入的键值对集合, 通过一系列处理来产生一个输出的键值对集合. 它将对数据集的操作分解为 Map 任务和 Reduce 任务, 用户可以通过重写 mapper 和 reducer 这两个函数分发给集群上的各个节点来对数据进行处理. Mapper 函数输入 $\langle \text{key}_M^{\text{In}}, \text{value}_M^{\text{In}} \rangle$, 输出 $\langle \text{key}_M^{\text{OUT}}, \text{value}_M^{\text{OUT}} \rangle$, reducer 函数输入 $\langle \text{key}_R^{\text{In}}, \text{value}_R^{\text{In}} \rangle$, 输出 $\langle \text{key}_R^{\text{OUT}}, \text{value}_R^{\text{OUT}} \rangle$, 其中, $\text{key}_M^{\text{OUT}}$ 需要转换为 key_R^{In} , 而 $\text{value}_R^{\text{In}}$ 是

保存 $\text{value}_M^{\text{OUT}}$ 的集合^[7].

3 基于 MapReduce 的改进 FIUT 算法

设 $\text{DB} = \{T_1, T_2, T_3, \dots, T_n\}$ 为事务数据库, 其项目集合 $I = \{i_1, i_2, i_3, \dots, i_n\}$, 每一项事务都是 I 的子集, $W = \{w_1, w_2, w_3, \dots, w_n\}$ 为 I 的权重集合用于表示项目的重要程度, 每一项 i_j 对应的权重设为 W_j . 最小加权支持度设为 $W_{\min, \sup}$, 即当 $W_{\min}(A) \geq W_{\min, \sup}$ 时, 项集 A 为加权频繁项目集.

本文对所提出的改进 PWFIUT 算法思想是: 使用一个新增的哈希表, 将候选项集中的项按支持度映射到哈希表中, 相同项则计数加一, 不同则插入新项; 同时, 通过引入额外的控制以限制候选项集的生成, 对支持度小于最小支持度阈值的项丢弃其事务属性, 甚至丢弃整个事务, 来达到逐步修剪事务数据库的目的.

PWFIUT 算法的实现包含三个 MapReduce 任务, 如图 1 所示, 第一个 MapReduce 任务负责生成频繁 1-项集, 详细描述见 3.1 节; 第二个 MapReduce 任务负责生成 k -itemsets, 详细描述见 3.2 节; 最后一个 MapReduce 任务负责构造 k -FIU-tree 并挖掘所有频繁 k -项集, 详细描述见 3.3 节.

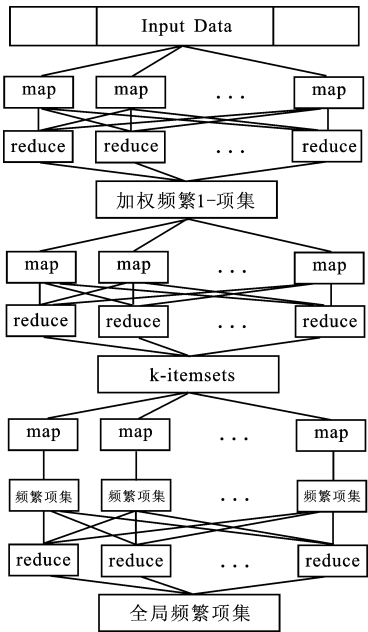


图 1 PWFIUT 算法框架

3.1 生成频繁 1-项集

将事务数据库划分为多个输入分片文件储存在集群 HDFS 分布式文件系统中, 每个 mapper 函数

读取其分片中的事务数据,并将数据按照键值对 $\langle \text{key}, \text{value} \rangle$ 的形式进行存储,每个键值对的数据对应 $\langle \text{offset}, \text{record} \rangle$ 的形式.之后 mapper 函数计算事务项频率并产生局部 1-项集,具有相同 key 值的数据经过 shuffle 分类合并到指定的 reducer 中,这一过程使用 combiner 以减少传输到 reducer 中的数据量,然后在 reducer 中生成全局 1-项集.最后根据事务权重表计算加权支持度,筛选得到全局加权频繁 1-项集,将这些加权频繁项集及其计数结果储存在本地文件系统中.

3.2 生成 k-itemsets

通过 mapper 函数对事务数据库进行第二轮扫描,同时结合 3.1 节所产生的加权频繁 1-项集修剪其中非频繁项事务,合并其本身项目集和修剪后项目集. mapper 所产生数据依然经过 combiner 处理 shuffle 分类合并到相应的 reducer 中, reducer 进行合并计数操作后,得到形式为 $\langle \text{itemnumber}, \text{count} \rangle$ 的键值对,按照其词序对频繁项进行排序并储存在本地文件系统中.

3.3 构造并挖掘 k-FIU-tree

第三个 MapReduce 任务是本算法的核心部分,它负责 WFIUT 的构建和挖掘.首先每一个 mapper 负责将 3.2 节所产生 k-项集分解成长度为 2 至 k-1 项集,这样可以使使用多个 mapper 并行处理,然后将 shuffle 阶段项目个数设置为 mapper 阶段输出中间键值对的键,其具有相同长度的项集发送至同一个 reducer 中进行合并.最后构建 k-PWFIU-tree 并通过检查叶子节点计数对其进行挖掘,挖掘完成后释放 k-PWFIU-tree.

4 实验结果分析

4.1 实验数据与运行环境

本文实验采用 Frequent Itemset Mining Dataset repository 上所提供的 accidents. dat. gz 作为实验数据集^[8],该数据集包含了 340.184 条交通事故记录,共 572 个不同的属性值表示在数据集中.实验中使用一个随机生成函数给每一个属性一个权重,权重分布在 1 至 10 之间.

本文实验采用的 MapReduce 并行计算环境是由五个节点组成的主从式 Hadoop 集群,其中一个节点作为 NameNode 和 JobTracker,其他四个节点

作为 DataNode 和 TaskTracker.每个节点均保持相同的硬件配置和软件配置,其中 CPU 采用 Intel Core i5 处理器,主频 2.5 GHz,计算机内存大小为 4 GByte,硬盘大小为 160 GB,操作系统采用 Centos7 版本,Hadoop 采用 2.6.0 版本,JDK 采用 jdk-7u79-linux-x64 版本,SSH 采用 OpenSSH 5.8 版本,集群使用默认的配置参数.

4.2 算法性能分析

为了测试 WPFIUT 算法在分布式并行计算环境下的性能,本文设置了一组对比试验,比较了本文提出算法和 PWARM 算法^[9].两种算法都是基于 JAVA 进行实现,其中最小加权支持度设置为 0.1.实验对这两种算法进行了比较分析,结果如图 2 所示.

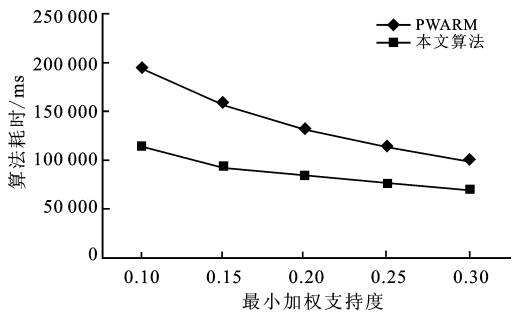


图 2 最小加权支持度对算法影响

由图 2 可以看出,在 Hadoop 计算环境下,随着节点数目的增多,本文算法所消耗的时间相比于 PWARM 算法更少.同时,随着最小加权支持度的不断增大,满足频繁项集条件的事务越来越少,由此所需处理这些事务的时间也相应减小.本文算法所消耗的时间开销增幅较为稳定,保持了较好稳定的性能.因此,实验结果证明了本文算法在 Hadoop 环境下能很好地完成数据挖掘任务.

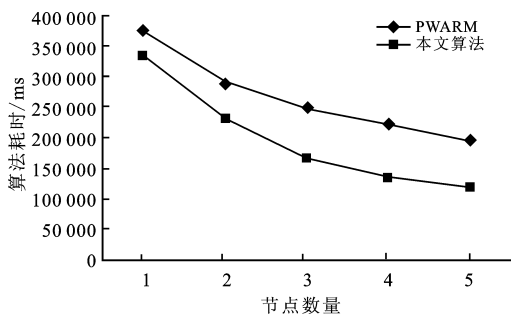


图 3 不同节点数量对算法影响

由图 3 可以看出,在面对规模较大的数据集,在

支持度阈值等条件相同的情况下,随着数据节点数目的增加,算法执行时间逐渐降低,说明算法具有较好的并行性.同时,虽然每个节点处理的项目集数量逐渐减少,但是节点之间 mapper 和 reducer 之间的通信开销逐渐增加,因此加速比性能减缓.实验结果证明了本文算法在 Hadoop 环境下具有较好的可扩展性.

5 结束语

本文以 FIU-tree 算法为基础,设计并实现了一种基于 MapReduce 编程模型的并行加权关联规则挖掘算法.本文算法结合加权计算模型,解决了数据库中数据项重要性不同对挖掘结果的影响;同时结合 MapReduce 平台,通过三个 MR 任务的调用,解决了算法的并行化问题,有效提升了挖掘效率.实验结果表明,本文算法能高效运行于 Hadoop 平台,完成快速扫描和挖掘数据,具有较好的可扩展性和加速比.

参考文献:

[1] 梁吉业,冯晨娇,宋鹏. 大数据相关分析综述[J]. 计算机学报, 2016,38(1):1-18.

[2] 肖文,胡娟,周晓峰. 基于 MapReduce 计算模型的并行关联规则挖掘算法研究综述[J]. 计算机应用研究, 2018,34(1):1-13.

[3] 朱晓峰,李玲娟,徐小龙,等. 基于 MapReduce 的关

联规则增量更新算法[J]. 计算机技术与发展, 2012, 22(4):115-118.

[4] Li H, Wang Y, Zhang D. Parallel fp-growth for query recommendation [C]// Proc of ACM Conference on Recommender Systems. Como, Italy,2008:107-114.

[5] 杨勇,高松松. 基于 MapReduce 的关联规则并行增量更新算法[J]. 重庆邮电大学学报(自然科学版), 2014, 26(5):670-678.

[6] Tsay Y J, Hsu T J, Yu J R. FIUT: A new method for mining frequent itemsets[J]. Information Sciences, 2009, 179(11):1724-1737.

[7] 宋杰,孙宗哲,毛克明,等. MapReduce 大数据处理平台与算法研究进展[J]. 软件学报, 2017, 28(3): 514-543.

[8] Frequent Itemset Mining Dataset repository [EB/OL]. [2012-10-21]. <http://fimi.ua.ac.be/data/>

[9] 李翔,刘韶涛. FP-Growth 的并行加权关联规则挖掘算法[J]. 华侨大学学报(自然版), 2014, 31(5): 523-527.

作者简介:

张 航 男,(1991-),硕士研究生.研究方向为云计算与大数据.

张 欣(通讯作者) 男,(1976-),博士,副教授.研究方向为下一代无线通信及应用等. E-mail:2225400115@qq.com.

张平康 男,(1993-),硕士研究生.研究方向为图像处理.

李 琪 男,(1990-),硕士研究生.研究方向为并行化计算.