基于 MapReduce 的增量数据挖掘研究

廖宝魁[1],孙隽枫[2]
(1.贵州大学 计算机科学与信息学院,贵州 贵阳 550025;
2.贵州大学 管理学院,贵州 贵阳 550025)

摘 要:频繁项集挖掘是数据挖掘过程中的重要部分,传统数据挖掘算法中常用 Apriori 算法和 FP 增长算法来挖掘频繁项集。在实际应用中,传统算法往往不能用于频繁更新的数据库,采用 IMBT 数据结构能从不断更新的数据库中挖掘频繁项集,但是这将导致存储空间不足和运行效率低下的问题。基于 MapReduce 的增量数据挖掘能够有效解决这些问题,通过对比基于 MapReduce 的增量数据挖掘和传统增量数据挖掘的运行时间可以证明,基于 Mapeduce 的增量数据挖掘更高效。

关键词:增量数据挖掘;MapReduce;增量挖掘二叉树;频繁项集
中图分类号:TP3 **文献标识码**:A **文章编号**:1674-7720(2014)01-0067-04

Research of incremental data mining based on MapReduce

Liao Baokui[1], Sun Junfeng[2]
(1.College of Computer Science and Information, Guizhou University, Guiyang 550025, China;
2.College of Mangement, Guizhou University, Guiyang 550025, China)

Abstract: Frequent itemset mining is an important part of data mining. Apriori and FP-tree are often used to mine frequent itemsets in traditional data mining algorithms. In practical situation, the traditional algorithms often cannot be used in the database which updates frequently. IMBT data structure is used to mine frequent itemsets from a continuously updated database, but this will lead to lack of storage space and the low efficiency. Incremental data mining based on MapReduce can solve these problems, To compare the running time of incremental data mining based on MapReduce and traditional incremental data mining can demonstrate the incremental data mining based on MapReduce is more efficient.

Key words: incremental data mining; MapReduce; IMBT; frequent itemset

目前,数据挖掘[1]在计算机领域正飞速发展。数据库系统领域的发展主要在数据收集、数据库创建、数据管理、数据分析、数据挖掘、数据仓库等方面。在数据挖掘中,挖掘关联规则是相当重要的部分。该部分的主要研究集中在挖掘算法上。国内外对频繁项集挖掘算法一直都有着很深的研究,例如:Apriori 算法[1],FP 增长算法[1-2]。

但是,现实应用中新的事务会不断地录入数据库,这使得许多挖掘算法不能处理动态的数据。数据库随机变动,这些算法不能有效应对数据的增添、删除等操作,这使得增量数据挖掘[3-5]变得尤为重要。

1 增量数据挖掘的必要性

在现实应用中,事务数据库常处于动态更新状态,这需要对传统关联规则挖掘算法做进一步改进,因此出现了一些新的关联规则。一些传统的批量挖掘算法通过反复扫描数据库来发现是否有新的事物添加到数据库中,但是这样做需要大量的运算时间。

实际应用中,由于新的事务不停地添加到数据库中,原先产生的频繁项集将被淘汰掉,基于新的数据库会产生新的频繁项集。增量挖掘算法能有效避免这样的问题。增量数据挖掘以之前挖掘的结果为基础,利用新增的事务来进行增量挖掘。

2 增量挖掘发展现状

2.1 基于 IMBT 的增量挖掘

为了能更好地利用现成的挖掘结果,采用了一种新的树形结构来代替 FP 树。该结构叫做增量挖掘二叉树(IMBT)[2],在事务添加到数据库中或从数据库中删除

后，它能给出每个项集的支持度计数。与之前的在数据库更新后通过反复扫描数据库得出的频繁项集支持度计数相比，该算法一次只处理一条事务并且记录数据集结构中可能的频繁项集，节约了大量的时间。

2.2 IMBT 构建过程

本文用一个数据集来说明 IMBT 的形成过程。假设一条事物为 I={a，b，c，d}，IMBT 的树形结构如图 1 所示。

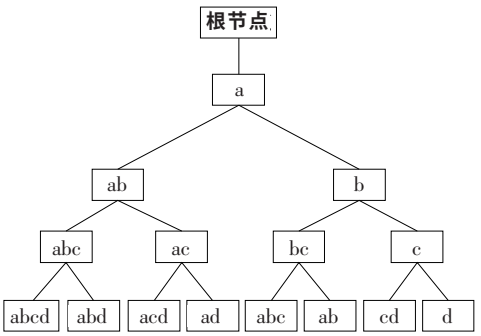

图 1 IMBT 树结构

根节点为一个空节点，用来指向树的第一个节点，第一个节点存储了发现的第一个最小的项集：{a}。之下的左右子节点存储了事务中包含的其余项。在最后一条事务处理完之后，IMBT 树构造完成。

2.3 向数据库中增加事务

首先给出源数据库 DB，如表 1 所示。

表 1 源数据库

ID	事务
1	b，c，d

用上面的方法构造 IMBT。另外一个数据库 DB1 存储了加到 DB 中的事务，如表 2。

表 2 向源数据库中添加的数据

ID	事务
1	a，d，e，f
2	a，e，e

DB1 中的第一条事物中有 4 个项(a，d，e，f)。需要插入新的节点并且为这个项集增加支持度计数。最后得出的 IMBT 如图 3 所示。

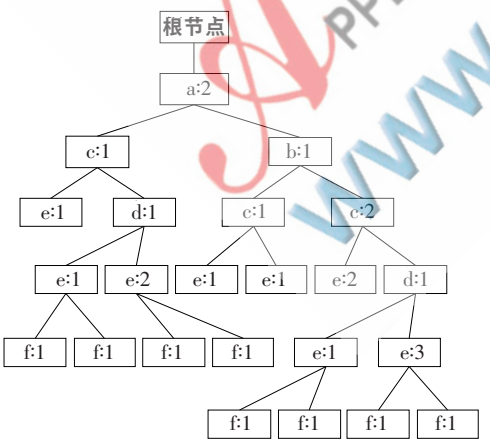

图 3 表 2 向表 1 添加数据后的 IMBT 树

2.4 在数据库更新后挖掘频繁项集

给定一个项集 X，如果 X 在事务数据库中出现的频率大于或等于预设的最小支持度，X 则称为频繁项集。如果项集 X 不是频繁项集，它的左半部分的子项集也不是频繁的，该算法会停止处理左半部分的子项集，这样可以提高挖掘效率。构建好 IMBT 树后，需要遍历该树来找出满足最小支持度阈值的频繁项集，挖掘结果被保存在一张表中以便将来使用。由于 IMBT 树的构建不需要支持度阈值，所以可以在数据库更新后以任何支持度阈值挖掘频繁项集。

该方法采用 IMBT 树结构，重用从源数据库中挖掘出来的结果挖掘新增事务，使得性能有大幅度提升。但是该方法仍面临内存空间不够的问题。随着程序运行，IMBT 树将会逐渐扩大，这使得内存空间容纳不下 IMBT 树，运行效率也将大大降低。采用并行机制来改进现有的串行挖掘算法将在性能上有很大地飞跃。

3 问题解决方案

3.1 并行算法

在并行计算[6-7]中，数据会被分发到不同的计算机节点中去，并行过程中每台计算机对不同的数据块执行相同的任务。

由于真实环境中的数据库通常非常大，把整个数据库保存在每个节点计算机上将造成存储空间过多的浪费。将数据库拆分则能成功地将子数据库分发在不同的计算机节点上。由于每台计算机都保存子数据库，节约了大量的存储空间。

3.2 并行算法的优势

随着数据的增加，当数据量超过了 GB 的时候，串行数据挖掘算法将很难在短时间内给出挖掘结果。而且单台电脑没有足够的内存来容纳全部的数据。在并行条件下，由于聚集了多台计算机的存储空间和处理能力，因此并行算法能很好地解决运行效率低下，存储空间不足的问题。

3.3 MapReduce 工作流程

要顺利实现并行算法需用 MapReduce 框架[8]。MapReduce 是由谷歌开发的标准软件架构，主要用于处理大数据操作任务。该架构由 Map 和 Reduce 组成。

当有数据输入时，输入数据被分割成块发送到各个节点计算机上，被分配了任务的节点计算机读取并处理收到的输入数据块。Map 函数处理完数据后输出中间数据：键值对，输出的中间键值对暂时缓冲到内存，这些内存中的的键值对将会写入到本地硬盘，然后将中间键值对在本地硬盘的位置信息发送给主节点计算机，主节点计算机负责向执行 Reduce 函数的计算机发送位置信息，这些计算机通过位置信息远程从运行 Map 函数的计算机硬盘上读取中间键值对，并将中间键值对按键分类，拥有相同键的值都分在一起，由 Reduce 函数处理后输出最终结果[8]。图 4 给出了其工作流程图。

4 提出系统

4.1 系统思路

针对上述内存空间和运行效率的问题，提出了一种

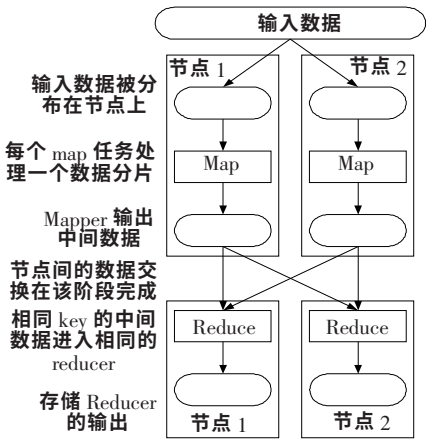

图 4 MapReduce 工作流程图

并行构建 IMBT 树挖掘频繁项集的方法。该方法主要完成两项工作：并行构建 IMBT 树及频繁项集计数。由于单台计算机内存和处理器能力有限，该算法不适用于单台电脑上运行。为了让算法的性能更高，就需要尽量减少计算机之间数据的传输并且避免过多的处理过程。

4.2 系统设计

首先将输入文件分为若干独立文件块。然后并行处理输入的每一个文件块。由于该方法需要用到基本数据结构 IMBT 进行增量挖掘，不需要为 IMBT 树定义最小支持度阈值。当本地 IMBT 树在各个节点计算机中生成后，每个项集将会在独立的节点计算机中进行支持度计数。然后将生成的局部频繁项集结合起来，在全局数据库中生成一个全局的频繁项集。最后，由用户定义一个最小支持度阈值，并将其用于全局频繁项集从而计算出真正的频繁项集。MapReduce 框架的工作模式能很好地实现该方法，该方法的设计流程图如图 5 所示。

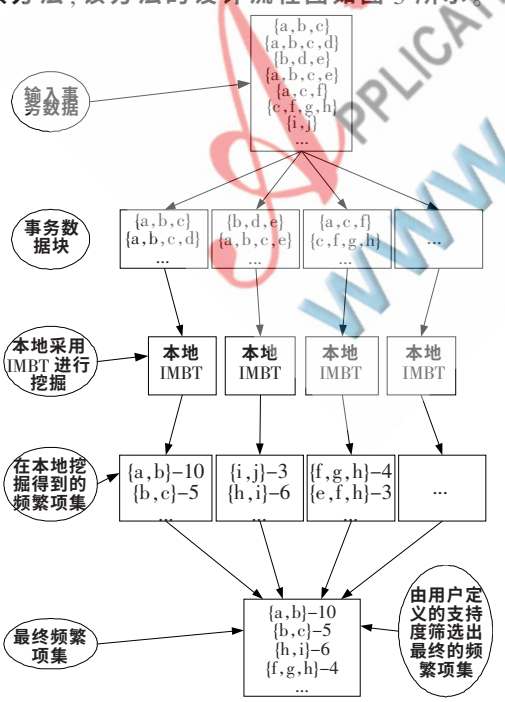

图 5 系统流程图

5 性能仿真与结果分析

为了对比基于 MapReduce 的增量数据挖掘和传统增量数据挖掘的运行效率，实验选取一个拥有 85 643 条事务的数据库，数据库中每条事务的项目数平均为 7 个，项目总共有 1 300 种，实验用计算机总共 3 台，配置均为双核 CPU AMD Athlon(tm)64 X2 Dual Core Processor 4000+，内存为 2 GB，安装 Ubuntu10.10 与 Window XP 双系统，其中传统 IMBT 挖掘算法在单台电脑上用 XP 系统运行，基于 MapReduce 的 IMBT 在 3 台电脑上用 Ubuntu10.10 运行，其中 1 台计算机配置为 namenode，另外 2 台配置为 datanode。由于是实验，所以没有配置 second namenode。

在数据挖掘进行之前，数据库中预存有 30 000 条事务，在基于 MapReduce 的 IMBT 算法中，这 30 000 条事务被平均分配到 3 台电脑上，实验开始后不断地向数据库录入事务数，两种算法均取支持度阈值为 800。图 6 给出在不断向数据库中添加事务时，两种算法的耗时对比。

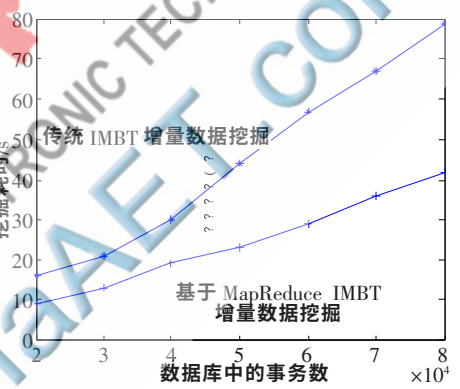

图 6 基于 MapReduce 的 IMBT 算法和传统 IMBT 算法对增量数据库的挖掘耗时对比

从图 6 中可以看出，基于 MapReduce 的 IMBT 算法的运行效率几乎比传统 IMBT 算法快一倍，图中的运行时间并非完全线性增长，这是由于数据库中每条事务的项目种类和项目数量不一致导致的。理论上基于 MapReduce 的 IMBT 算法采用 2 台计算机同时进行挖掘任务，效率应该快一倍，图中结果并未达到一倍是因为整个 MapReduce 过程需要频繁传递信息，namenode 需要一定的响应时间，导致实际效率与理论效率存在一定误差。但基于 MapReduce 的增量数据挖掘算法在运行效率上比传统数据挖掘算法仍然有了质的提升。

传统数据挖掘技术：Apriori、FP 树等算法，虽然都能有效地找出频繁项集，但不能适用于真实环境下动态的数据。所以出现了增量数据挖掘，本文给出了一种基于 IMBT 结构的增量数据挖掘算法，该算法能够在新事务添加到数据库或从数据库中删除后有效地列举出每一个项集的支持度计数。由于在树的构建过程中不需要预设最小支持度阈值，该算法允许用户以任何支持度阈值挖掘频繁项集。结合之前从数据库中挖掘出来的结果，

该算法能够挖掘更新后的数据库，效率上有很大的提升。但是IMBT树在单台计算机中运行时，该算法面临存储空间不足的问题，随着算法的进行，IMBT树逐渐扩展，会造成内存溢出，效率降低。

为此提出了一种新的方法，该方法采用MapReduce框架，将数据库分为若干子数据库然后发向多个节点计算机，由于计算机集群聚集了多台计算机的存储能力和计算能力，在存储空间上可以动态的增加，并且能够并行处理数据，从而解决了运行效率和存储空间的问题，因此该方法比传统的非并行增量算法更高效。

参考文献

[1] 范明，孟小峰.数据挖掘概念与技术[M].北京：机械工业出版社，2012.

[2] 蒋翠清，胡俊妍. 基于FP-tree的最大频繁项集挖掘算法[J].合肥工业大学学报(自然科学版)，2010，33(9)：387-1391.

[3] HONG T P，WANG C Y，TAO Y H. A new incremental data mining algorithm using pre-large itemsets[J]. Intelligent Data Analysis，2001，5(2)：111-129.

[4] HONG T P，LIN C W，WU Y L. Incrementally fast updated frequent pattern trees [J]. Expert Systems With Applications，2008，34(4)：2424- 2435.

[5] YANG C H，YANG D L. IMBT-a binary Tree for Efficient Support Counting of Incremental Data Mining[J]. 2009 International Conference on Computational Science & Engineering，2009，1(1)：324-329.

[6] 刘鹏.云计算[M].北京：电子工业出版社，2011.

[7] 高岚岚.云计算与网格计算的深入比较研究[J].海峡科学，2009(2)：56-57.

[8] LAM C，韩冀中.Hadoop实战[M].北京：人民邮电出版社，2012.

(收稿日期：2013-10-11)

作者简介：

廖宝魁，男，1988年生，硕士研究生，主要研究方向：网络系统与信息安全。

孙隽枫，男，1988年生，硕士研究生，主要研究方向：数据挖掘。