基于 Hadoop 的关联规则挖掘算法研究
——以 Apriori 算法为例

刘木林,朱庆华
(南京大学 信息管理学院,江苏 南京 210023)

摘 要:为了解决传统关联规则挖掘算法在挖掘效率、算法扩展性等方面无法适应大数据挖掘需求的问题,以经典的关联规则挖掘算法—Apriori 算法为例,首先基于 Hadoop 平台和 MapReduce 编程模型,实现算法的并行化。在此基础上,基于事务缩减的思想对算法进行优化,进一步提高算法的挖掘效率。搭建 Hadoop 集群环境,对算法的挖掘结果和挖掘效率进行实验。通过并行挖掘结果验证、串行版与并行版效率对比、挖掘时间与节点数目的变化关系、挖掘时间与数据量的变化关系 4 组实验,结果表明:文中实现的 Apriori 算法不仅能够准确挖掘频繁项集,而且比传统串行算法具有更高的挖掘性能和可扩展性。该算法能够更好地适应大数据集的挖掘要求,能够实现从大规模数据集中高效挖掘频繁项集和关联规则。

关键词:数据挖掘;关联规则;Hadoop;Apriori
中图分类号:TP393 **文献标识码**:A **文章编号**:1673-629X(2016)07-0001-05
doi:10.3969/j.issn.1673-629X.2016.07.001

Research on Association Rules Mining Algorithm Based on Hadoop
—Taking Apriori as an Example

LIU Mu-lin,ZHU Qing-hua
(School of Information Management,Nanjing University,Nanjing 210023,China)

Abstract:In order to solve the problem that the traditional association rules mining algorithm has been unable to meet the mining needs of large amount of data in the aspect of efficiency and scalability,take Apriori as an example,the algorithm is realized in the parallelization based on Hadoop framework and MapReduce model. On the basis,it is improved using the transaction reduce method for further enhancement of the algorithm's mining efficiency. The experiment,which consists of verification of parallel mining results,comparison on efficiency between serials and parallel,variable relationship between mining time and node number and between mining time and data amounts,is carried out in the mining results and efficiency by Hadoop clustering. Experiments show that the paralleled Apriori algorithm implemented is able to accurately mine frequent item sets,with a better performance and scalability. It can be better to meet the requirements of big data mining and efficiently mine frequent item sets and association rules from large dataset.

Key words:data mining;association rules;Hadoop;Apriori

0 引 言

关联规则是指存在于两个或多个变量之间的一类重要的能被发现的规律性,这种规律性往往对实际的生产生活有着重要的指导作用,因此关联规则挖掘一直都是数据挖掘的一个重要方面。随着大数据时代的来临,传统的关联规则挖掘算法已难以满足大数据量的挖掘需求。Hadoop 平台是在大数据背景下诞生的分布式计算平台。Hadoop 的工作方式是并行的,通过并行提高处理效率。因此将传统挖掘算法的思想基于 Hadoop 平台实现,使得传统关联规则算法能够适应大规模数据集的处理要求,同时借助 Hadoop 平台在分布式处理方面的优势,获得处理性能方面的提升无疑是很有意义的,而这也正是文中研究的目的所在。

1 关联规则算法研究现状

1993 年,Agrawal 提出关联规则挖掘后,关联规则

收稿日期:2015-08-13 **修回日期**:2015-11-18 **网络出版时间**:2016-06-22
基金项目:国家自科基金面上项目(71473114)
作者简介:刘木林(1991-),男,硕士研究生,通讯作者,研究方向为互联网用户行为分析;朱庆华,教授,博士生导师,研究方向为网络信息资源管理、信息用户行为、信息政策分析、决策咨询服务等。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20160621.1701.010.html

的挖掘算法就成为了研究的一个热点,相继出现了很多研究成果。其中最著名的包括 Agrawal 提出的 Apriori 算法[1],这是布尔型关联规则挖掘的基础算法,此外 Han 等提出的改进算法—FP-Growth 算法[2]。随着数据量的不断积累,串行算法越来越难以满足需要,因此 Agrawal 等又在 1996 年创造性地提出了并行的挖掘算法[3]。而 Hadoop 平台诞生后,也有研究将关联规则算法通过 Hadoop 平台进行移植和改进。

表 1 总结了目前关联规则算法研究的现状。

表 1　关联规则算法研究现状

类别	算法描述
经典串行算法	Apriori 算法[1]:一种限制候选集的逐层迭代算法,在两个地方存在性能瓶颈,一是需要多次扫描数据库;二是虽然可以限制候选项集,但是当数据量较大时,候选集的数量依然会非常庞大,特别是候选 2 项集
	FP-Growth 算法[2]:不产生候选项集的挖掘算法
	Eclat 算法[4]:一种基于垂直数据格式的算法
串行算法的改进	DHP 算法[5]、Partition 算法[6]、Sample 算法[7]、基于矩阵或压缩矩阵的 Apriori[8-10]、用支持度二维表的方法对 FP-Growth 算法进行改进[11]、对 Eclat 算法进行改进[12-13]
并行算法	(CD 算法、DD 算法和 CaD 算法)[3]、MLFPT 算法[14](基于 FP-Growth 算法的并行化)、PDM 算法[15](基于 DHP)、FDM 算法[16]
基于 Hadoop 的研究	已有的基于 Hadoop 的关联规则算法研究大都基于 Apriori 算法和 FP-Growth 算法。基于 Apriori 的算法[17-24]主要利用两种思想:第一种是将数据集分发到各节点,节点并行挖掘局部频繁项集,然后再汇总得到全局的频繁项集。第二种则是通过迭代 MapReduce 的方式挖掘频繁项集。此外,还有学者研究了 FP-Growth 算法在 Hadoop 平台上的并行化[25-27]

从文献调研中发现,基于 Hadoop 平台实现关联规则挖掘仍然主要基于 Apriori 和 FP-Growth 这两种基础算法,在实现并行化的基础上,再采用优化策略对算法进行性能优化。但是已有的研究在对算法进行实验时,大都只专注于提高挖掘性能,并不注重对挖掘结果的验证和比较。基于这种情况,文中重点关注 Apriori 算法,首先将算法在 Hadoop 平台上进行实现,同时利用事务缩减思想对算法进行改进,在保证挖掘结果正确性的前提下,提高算法的挖掘性能和可扩展性。

2　基于 Hadoop 的 Apriori 算法的实现与改进

2.1　Hadoop 平台简介

Hadoop 是基于主从结构的架构,集群中主要分为 master 节点和 slave 节点。整个平台最核心的两个部分是 HDFS(Hadoop Distributed File System)和 MapReduce。

HDFS 是 GFS(Google File System)的开源实现,从架构上看是一个主从体系结构,以流式数据访问模式来存储超大文件[28]。它的高容错性、高可扩展性、高吞吐率与高获得性等特点为大数据提供了可靠的存储。而 MapReduce 则大大降低了并行编程的门槛,并行编程中的工作调度、负载均衡以及容错处理等问题均由 MapReduce 框架负责。使用 MapReduce 框架编写分布式并行程序时,只需要实现框架下的 map( )和 reduce( )函数即可。

2.2　Apriori 算法的 Hadoop 实现

Apriori 的 Hadoop 实现一般有两种思路,笔者这里是通过迭代 MapReduce 的方法来实现,因为 HDFS 默认的数据分块大小是 64 MB,如果采取另一种思路来实现,则意味着每个节点都需要单独处理 64 MB 的数据量,同时可能会产生数量庞大的局部频繁项集。虽然 Hadoop 允许自定义文件块的大小,但是这种更改会对 Hadoop 的可扩展性、伸缩性以及并行性造成影响,并不值得提倡。实现后的程序流程图见图 1。

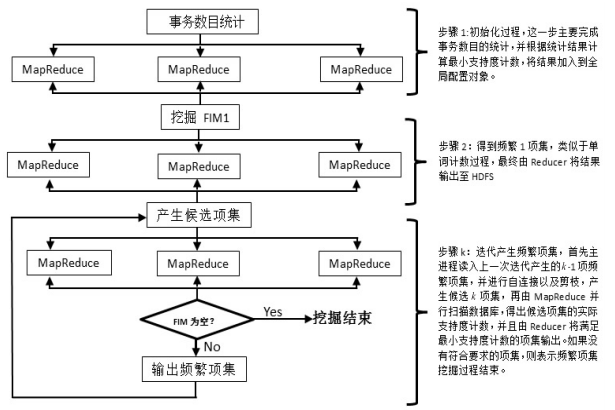

图 1　并行后的程序流程图

下面对程序实现的关键细节进行说明。

(1)初始化过程。

挖掘频繁项集首先需要确定实际的支持度计数,这要根据事务总数和输入的支持度百分比来确定,但实验中是以文件代替数据库来存储事务集,因此需要通过一个单独的 Job 来完成事务总数的统计,统计完成后再计算出最小支持度计数,并放入全局配置对象中。

(2)挖掘频繁 1 项集。

挖掘频繁 1 项集的过程类似于单词计数过程,不同的是需要 Reducer 判断每个条目的计数是否满足最小支持度,并将满足最小支持度的条目输出,这个过程同样需要主进程新建一个 Job。

(3)迭代挖掘频繁 k 项集。

这一步是整个算法实现的核心。首先需要读入上一步得出的频繁 1 项集,进行连接后产生候选 2 项集,由于候选 2 项集的产生无法进行剪枝,因此只需要进

行连接步即可,然后再进行候选 2 项集的支持度计数与筛选。随后 Mapper 从全局配置对象中获取候选 2 项集的路径,然后将文件中的条目读入放到内存中。随后在 map 函数中按行读入每一条事务,并将包含于事务中的候选 2 项集输出至 Combiner 中。Combiner 先对本地相同 key 值的条目进行汇总,将本地汇总结果输出至 Reducer 中,Reducer 对全部的结果进行最终汇总,并且判断条目的支持度计数是否满足最小支持度计数,如果满足,则将条目及其支持度输出至 HDFS。主进程在 Job 完成之后,到对应的上一次频繁项集的输出路径中读取 $k-1$ 项频繁项集,并且判断 $k-1$ 项频繁项集是否为空,如果是,则结束挖掘,如果否,则对读入的 $k-1$ 项频繁项集进行自连接,并利用先验性质进行剪枝,随后开始一个 Job 对候选频繁项集进行支持度计数,并且不断迭代这个过程,直到产生的频繁项集为空。

2.3　基于事务缩减的算法改进策略

上节叙述了 Apriori 算法在 Hadoop 平台上实现的基本思路,在这一思路中将数据库的扫描过程实现了并行化,而数据库扫描是 Apriori 算法的主要瓶颈之一。在基本实现的基础上,还可以对算法的实现进行改进。表 1 中分析了 Apriori 算法的两处性能瓶颈,对于问题二,文中算法在主程序产生候选项集的过程中应用了先验剪枝,对于候选项集的数量产生了限制作用。而对于问题一,文中进一步采用事务缩减的思想来减少数据库事务的扫描次数。事务缩减思想同样基于频繁项集的一种性质即:不包含任何 $k-1$ 项频繁集的事务不可能包含 k 项频繁集,因此在数据库扫描过程中可以将这些事务进行标记,从而减少需要扫描的事务数目,提高挖掘效率。而文中利用了与此相似的另外一种性质即:不包含任何候选 k 项集的事务不可能包含任何 k 项频繁集。

基于事务缩减的算法改进策略需要解决的第一个问题就是如何唯一地标识每一条事务记录。在 HDFS 中,每个文件都会以 64 MB 的块为单位进行存储,每个块都有一个唯一的 URL。此外,在 MapReduce 执行过程中,每个 Mapper 都需要单独处理一个 split(split 与 HDFS 中的 block 是相对应的),采用按行读入事务记录的方式时,key 值为该行记录在文件中的偏移字节数,对于该记录而言,此 key 值可以作为其在该 split 中的唯一标识。这样,由 split 的 URL 加该事务记录的 key 值便可以将其唯一地标识出来。按照该策略,改进的重点就在 Mapper 的执行逻辑中。即 Mapper 首先需要获取 split 的 URL,存入 Mapper 中的一个成员变量。同时根据 split 的 URL,根据约定的路径找到存储其剔除列表的文件,并将剔除列表读入一个 HashSet 中。map 函数对候选项集计数时,如果发现该条事务不包含任何候选项集,则将其加入最新的剔除列表。最后在 Mapper 的 cleanup 函数中将新的剔除列表附加到剔除文件中,以供下一次扫描时使用。

笔者在测试中发现,采用事务缩减进行改进后,在挖掘频繁 3 项集时可以剔除约 5% 的事务,4 项集时可以剔除约 10% 的事务,5 项集时可以剔除约 17% 的事务,6 项集时可以剔除约 25% 的事务。因此,随着挖掘的不断进行,剔除的事务量会不断增多,挖掘效率的提升也更加明显。

3　实　验

3.1　Hadoop 分布式环境搭建

为了进行实验,笔者搭建了一个小型的集群环境,包括 1 个 master 节点和 2 个 slave 节点(节点计算机配置如表 2 所示),namenode、jobtracker 均位于 master 节点,datanode、tasktracker 均位于 slave 节点,3 个节点统一使用 JDK1.7.0_45 版本,Hadoop 版本则为 1.2.1。

表 2　集群节点配置情况

节点类型	CPU	内存/GB	操作系统	硬盘大小/GB
master	双核,3 GHz	2	Ubuntu 13.04	320
slave1	双核,1.3 GHz	2	Ubuntu 13.04	320
slave2	双核,2 GHz	1.5	Ubuntu 13.04	200

3.2　实验结果分析

实验采用的数据为 FIMI Repository(Frequent Itemset Mining Implementations Repository,该网站提供大量 IEEE 国际数据挖掘大会上关于频繁项集挖掘方面的数据集、论文、实验结果等资料)提供的一份 webdocs 事务数据集,该数据是由意大利国家研究理事会的 Claudio Lucchese 等通过 Web 爬虫爬取了 170 万 Web 文档后,通过初步清理(取出 html 标签)和分词,并通过词干提取算法,获得了出现在文档中的所有单词的事务集(同一个文档中一个单词只计 1 次,并将相应的单词转换为数字 id 来表示)。文件大小为 1.4 GB,包含 169 万条事务集,其中最长的事务集约包含 7 万个条目。为了方便实验的进行,在原文件的基础上,将文件分割成 50 MB,100 MB,150 MB,200 MB,250 MB,300 MB,500 MB,750 MB,1 GB,1.4 GB 等分块。另外,笔者利用 Java 语言实现了单机串行版的 Apriori 算法(以下简称串行版),并将串行版与并行算法的效率进行对比,其中串行版程序都运行在 slave1 节点上,实验共分为 3 组进行,每次实验都运行 3 次取平均值。

(1)并行挖掘结果验证。

挖掘结果的验证主要通过串行程序与并行程序挖掘结果的对比来展示,如果并行程序的挖掘结果与串

行程序的一致,则说明笔者实现的并行算法是可靠的,反之则说明并行算法的设计与实现存在问题,无法得出正确的挖掘结果。

表 3 显示了 150 M 文件的挖掘结果;表 4 显示了 250 M 文件的挖掘结果(FIM1 代表频繁项集 1 项集,其他的依此类推)。

表 3　150 M 文件挖掘结果

算法	FIM1	FIM2	FIM3	FIM4	FIM5
串行算法	26 项	61 项	55 项	27 项	4 项
并行算法	26 项	61 项	55 项	27 项	4 项

表 4　250 M 文件挖掘结果

算法	FIM1	FIM2	FIM3	FIM4	FIM5
串行算法	23 项	61 项	55 项	27 项	4 项
并行算法	23 项	61 项	55 项	27 项	4 项

从表中可以看出,串行算法与并行算法挖掘出的频繁项集的数目是一致的,另外,笔者对比了从频繁 1 项集到频繁 5 项集的具体挖掘结果,均完全一致。因此,文中提出的并行挖掘算法是可靠的,能够准确挖掘出满足最小支持度的频繁项集。

(2)串行版与并行版效率对比。

分别利用串行版程序与并行版程序对大小为 50 MB,100 MB,150 MB,200 MB,250 MB,300 MB,350 MB,400 MB,450 MB,500 MB 的数据进行挖掘,最小支持度设为 0.25,实验结果见图 2。

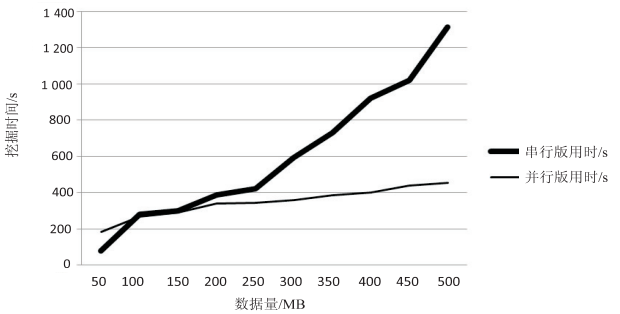

图 2　串行版与并行版效率对比

从图中可以看出,在数据量较小时,并行算法由于在工作调度等方面的开销,并没有体现出挖掘效率的优势。而随着数据量的不断积累,并行算法的优势逐渐体现出来,挖掘时间也大大少于串行算法。更重要的是,串行算法在挖掘 500 MB 以上的数据量时,内存不足等方面的限制会导致运行失败,除非继续改进单机的配置,否则无法继续挖掘更大的数据,而并行算法则不存在这样的问题。

(3)挖掘时间与节点数目的变化关系。

笔者搭建的集群共有 3 台计算机,其中配置较好的一台即作为 NameNode、JobTracker,也作为 DataNode、TaskTracker。分别将集群调整为 1 个节点、2 个节点、3 个节点,并对 300 M 的数据进行挖掘,设最小支持度为 0.25,实验结果见图 3。

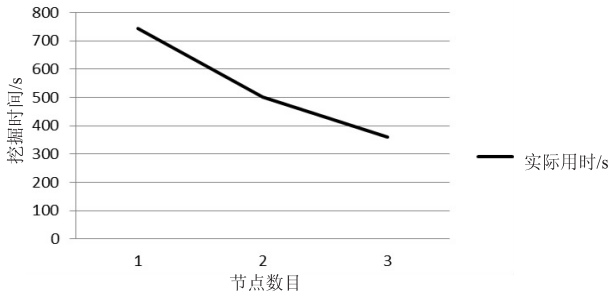

图 3　挖掘时间与节点数目的变化关系

从图中可以看出,计算节点的增大能够明显提高挖掘效率,这也是分布式计算可扩展性方面的最大优势之一,即通过节点的灵活配置,可以很轻松地应对大数据的处理。

(4)挖掘时间与数据量的变化关系。

采用并行版程序分别挖掘大小为 100 MB,200 MB,300 MB,400 MB,500 MB,600 MB,700 MB,800 MB,900 MB,1 000 MB,1 100 MB,1 200 MB,1 300 MB,1 400 MB 的数据集,设置最小支持度为 0.25,观察随着数据量的增加挖掘时间的变化情况,实验结果如图 4 所示。

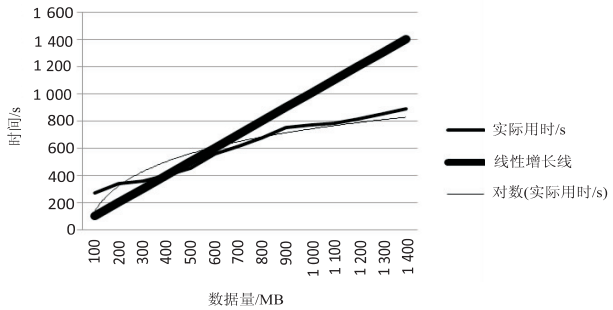

图 4　挖掘时间与数据量的变化关系

从图中可以看出,随着挖掘数据量的不断增长,挖掘时间的增长速度低于线性增长速度,并且接近于对数增长速度,而图 3 中的普通串行算法的挖掘时间会因数据量的增加而迅速增长,说明文中算法对于数据量的增长有着更好的适应性。如果结合计算节点的适当扩展,完全能够适应更大数据量的挖掘要求。

4　结束语

文中通过 Hadoop 平台实现了 Apriori 算法的并行化,通过事务集的并行扫描大大提高了算法的执行效率,同时为了减少数据库的扫描消耗,运用事务缩减思想优化算法实现,进一步提高了算法效率。经过一系列的实验表明,文中实现的并行 Apriori 算法在保证挖掘结果准确的前提下,比普通串行挖掘具有更少的时间消耗,能够更快速地挖掘出频繁项集,同时从实验中看出,并行算法对数据量的不断增长有着更好的适应

能力,对于大文件也有着很好的挖掘性能。此外,实验结果还表明,计算节点的增加同样能够提高挖掘效率,这也是分布式集群系统的最大威力所在。综合来看,文中的研究能够为大数据条件下关联规则的高效挖掘提供借鉴意义。当然,目前也还存在一些不足,比如最小支持度的变化对算法性能的影响比较明显,特别在频繁 2 项集的挖掘上,因为先验剪枝无法对候选 2 项集的产生进行限制,同时文中提出的事务缩减思想同样也无法提高频繁 2 项集的挖掘效率。因此,下一步的研究重点主要是如何利用哈希散列的方式来限制候选 2 项集的产生,进一步提高算法的效率。

参考文献:

[1] Agrawal R, Srikant R. Fast algorithms for mining association rules[C]//Proceedings of the 20th VLDB conference. Santiago, Chile: [s. n.], 1994: 487-499.

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

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

[4] Zaki M J. Scalable algorithms for association mining[J]. IEEE Transactions on Knowledge and Data Engineering, 2000, 12(3): 372-390.

[5] Park J S, Chen M S, Yu P S. An effective hash-based algorithm for mining association rules[J]. ACM SIGMOD Record, 1995, 24(2): 175-186.

[6] Sarasere A, Omiecinsky E, Navathe S. An efficient algorithm for mining association rules in large databases[C]//Proc of 21st international conference on very large databases. Zurich, Switzerland: [s. n.], 1995.

[7] Toivonen H. Sampling large databases for association rules [C]//Proc of conference on very large data bases. [s. l.]: [s. n.], 1999: 134-145.

[8] 孙逢啸,倪世宏,谢　川. 一种基于矩阵的 Apriori 改进算法[J]. 计算机仿真,2013,30(8):245-249.

[9] 罗　丹,李陶深. 一种基于压缩矩阵的 Apriori 算法改进研究[J]. 计算机科学,2013,40(12):75-80.

[10] 高海洋,沈　强,张轩溢,等. 一种基于数据压缩的 Apriori 算法[J]. 计算机工程与应用,2013,49(14):117-120.

[11] 杨　云,罗艳霞. FP-Growth 算法的改进[J]. 计算机工程与设计,2010,31(7):1506-1509.

[12] 张玉芳,熊忠阳,耿晓斐,等. Eclat 算法的分析及改进[J]. 计算机工程,2010,36(23):28-30.

[13] 冯培恩,刘　屿,邱清盈,等. 提高 Eclat 算法效率的策略[J]. 浙江大学学报:工学版,2013,47(2):223-230.

[14] Zaïane O R, El-Hajj M, Lu P. Fast parallel association rule mining without candidacy generation[C]//Proceedings IEEE international conference on data mining. [s. l.]: IEEE, 2001: 665-668.

[15] Park J S, Chen M S, Yu P. Efficient parallel data mining for association rules[C]//Pissinou N, Silberschatz A, Park E K, et al. Proceedings of the fourth international conference on information and knowledge management. New York, NY, USA: ACM, 1995: 31-36.

[16] Cheung D W, Han J, Ng V T, et al. A fast distributed algorithm for mining association rules[C]//Proc of fourth international conference on parallel and distributed information systems. [s. l.]: IEEE, 1996: 31-42.

[17] Yang X Y, Liu Z, Fu Y. MapReduce as a programming model for association rules algorithm on Hadoop[C]//Proc of 3rd international conference on information sciences and interaction sciences. [s. l.]: IEEE, 2010: 99-102.

[18] Li N, Zeng L, He Q, et al. Parallel implementation of Apriori algorithm based on MapReduce[C]//Proc of 13th ACIS international conference on software engineering, artificial intelligence, networking and parallel & distributed computing. [s. l.]: IEEE, 2012: 236-241.

[19] Lin M Y, Lee P Y, Hsueh S C. Apriori-based frequent itemset mining algorithms on MapReduce[C]//Proceedings of the 6th international conference on ubiquitous information management and communication. [s. l.]: ACM, 2012.

[20] Li L, Zhang M. The strategy of mining association rule based on cloud computing[C]//Proc of international conference on business computing and global informatization. [s. l.]: IEEE, 2011: 475-478.

[21] Woo J. Apriori-Map/Reduce algorithm[C]//Proc of the 2012 international conference on parallel and distributed processing techniques and applications. Las Vegas: [s. n.], 2012.

[22] 章志刚,吉根林. 基于迭代式 MapReduce 的 Apriori 算法设计与实现[J]. 华中科技大学学报:自然科学版,2012(S1):9-12.

[23] 黄立勤,柳燕煌. 基于 MapReduce 并行的 Apriori 算法改进研究[J]. 福州大学学报:自然科学版,2011,39(5):680-685.

[24] 范燕燕. 基于 MapReduce 的分布式关联规则挖掘算法研究[D]. 哈尔滨:哈尔滨工程大学,2013.

[25] Yong W, Zhe Z, Fang W. A parallel algorithm of association rules based on cloud computing[C]//Proc of 8th international ICST conference on communications and networking in China. [s. l.]: IEEE, 2013: 415-419.

[26] Zhou L, Zhong Z, Chang J, et al. Balanced parallel FP-growth with MapReduce[C]//Proc of IEEE youth conference on information computing and telecommunications. Beijing: IEEE, 2010: 243-246.

[27] 周诗慧. 基于 Hadoop 的改进的并行 Fp-Growth 算法[D]. 济南:山东大学,2013.

[28] White T. Hadoop: the definitive guide[M]. 3rd ed. USA: O´Reillv Media, 2012.