Doi:10.3969/j.issn.1003-5060.2013.08.009

基于并行抽样的海量数据关联挖掘算法

宛 婉, 周国祥
(合肥工业大学 计算机与信息学院,安徽 合肥 230009)

摘 要:在"信息爆炸"的当今社会,海量数据对数据挖掘提出新的挑战。文章针对海量数据挖掘时所面临的内存和性能问题,提出了一种基于并行随机数据抽样的云频繁项集挖掘算法。该算法可以实现在单次扫描海量数据进行并行数据抽样的基础上,对样本数据进行并行的频繁项集挖掘。实验结果表明,通过并行随机抽样算法可以有效抽取反映数据真实情况的样本数据,并对其进行相关清理,在得到样本数据后采用文中所提的并行关联云挖掘算法能有效解决内存和性能方面的问题,为推动数据挖掘在海量数据下的发展奠定了良好基础。

关键词:云计算;并行计算;随机抽样;关联分析
中图分类号:TP391.12 **文献标志码**:A **文章编号**:1003-5060(2013)08-0933-05

Mass data association mining based on parallel random sampling

WAN Wan, ZHOU Guo-xiang
(School of Computer and Information, Hefei University of Technology, Hefei 230009, China)

Abstract: In an age of information explosion, data mining faces new challenges because of mass data. In view of the memory and performance problems faced by mass data mining, a cloud frequent itemsets mining algorithm based on parallel random sampling of data is put forward. After the parallel data sampling of the single scanning mass data, it can realize the parallel frequent itemsets mining of the sample data. The experimental results prove that the parallel random sampling algorithm can effectively extract data which can reflect the real data sets, and carry on the related cleaning. After obtaining the sample data, the memory and performance problems can be effectively solved by this parrallel association cloud mining algorithm. This data mining algorithm lays a good foundation for promoting the development of mass data mining.

Key words: cloud computing; parallel computing; random sampling; correlation analysis

0 引 言

数据挖掘是从大量的、不完全的、有噪声的、模糊的、随机的数据提取隐含其中的、未知的、具有潜在利用价值的信息和知识的过程。关联规则挖掘是数据挖掘的一个重要分支,它通过收集数据库中大量的事物记录进行分析,查找其中有趣的关联关系,是数据挖掘的一个重要功能。但是传统的关联算法如(Apriori 算法[1])在处理海量数据时,会对内存带来极大的压力。

本文对传统 Apriori 算法的优缺点进行详细分析,并提出了一种在海量数据并行随机抽样后,在数据抽样样本中采用 MapReduce 实现的并行频繁项集发现算法,其可以有效地减小对内存的压力。实验结果证明,该算法能有效提高频繁项集发现算法的运算效率,减少了内存的压力,极大

收稿日期:2012-12-03;**修回日期**:2013-06-07
基金项目:国家自然科学基金重点资助项目(60633060)
作者简介:宛 婉(1988—),女,安徽芜湖人,合肥工业大学硕士生;
周国祥(1956—),男,安徽合肥人,合肥工业大学教授,硕士生导师.

地扩展了该关联算法处理海量数据的能力。

1 传统频繁集算法的局限

最经典的 Apriori 算法[2]主要是基于单调性理论,即:如果一个集合的子集不是频繁项集,则该集合也不可能是频繁项集。基于该思路,对所有不同的集合大小 k,都要对整个数据集分别进行一遍扫描处理,如果不存在某个大小的频繁项集,则不可能存在更大的频繁项集,因此,扫描过程结束。其主要流程如图 1 所示,其中,C_k 为大小为 k 的候选项集集合,即必须通过计算来确定真正频繁的项集组成的集合;L_k 为大小为 k 的真正频繁项集集合。

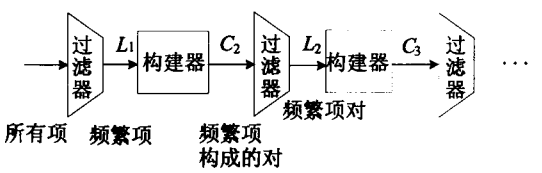

图 1 Apriori 算法过程示意图

上述每次扫描过程均在内存中维护很多不同的计数值。假定单元素频繁项的总数为 n,在 Apriori 的 C_2 扫描过程中,需要空间来存储 C_n^2 个项对计数。如果每个计数需要 4 字节,则共需要 $2n^2$ 个字节。如果单台机器为 2 GB,即 2^{31} 字节内存,则要求 $n \leqslant 2^{15}$,即 $n \leqslant 33\ 000$。如果 n 超过该值,则在单台机器上运行传统频繁集算法会出现内存抖动现象,即在磁盘和内存之间反复传输数据,从而运行速度可能比直接在内存中运行慢几个数量级。

另外,Apriori 算法利用单调性原则[3]进行剪枝,虽然已经大大减小了候选项集的个数,但在数据量大、支持度阈值较小的情况下,仍然会产生大量的频繁项集,进一步对内存提出了更高的挑战。Apriori 算法是逐层扫描进行计数,在数据量较大时,扫描的代价将会相当大[4]。传统的 Apriori 算法在空间(内存大小)和时间(扫描时间)上的消耗都非常大。

2 传统 Apriori 算法的改进

针对 Apriori 算法的局限,已经有多名学者对 Apriori 算法进行了改进,主要思想在于控制候选集的规模或减少数据集扫描次数。几种优秀的改进算法如下:杂凑表技术[5]、减少数据集的方法[6]、Partition 算法[7]、抽样方法[8]、动态模式计数法[9]、TreeProjection 方法[10]和 FP-Growth 算法[11]。

对于数据集非常大、支持度阈值相对较小的情况,上述算法存在各种各样的问题。如 FP-tree 算法,如果生成的条件 FP-Tree 非常茂盛(最坏情况下为 1 棵前缀树),则该算法产生大量的子问题。

为了进一步提高算法效率,有不少学者从并行化的角度来改进 Apriori 算法。其中比较成熟的有 3 种[12]:

(1) Count Distribution 算法,简称 CD 算法。其主要聚集于最小化节点间的通信。但其在实际应用中,有可能因为计算中的 I/O 读写过度频繁,从而引起整体的性能下降。

(2) Data Distribution 算法,简称 DD 算法。其先将候选项集分发到各个节点上,而获取到购物篮的统计计数。该算法需要进行大量的通信,并且等待其他节点同步信息的时间事先不可确定,所以该算法性能较差。

(3) Candidate Distribution 算法,简称 CaD 算法。其综合了 DD 算法和 CD 算法,以弥补其各自的不足。但是该算法实现复杂,实际应用时效果不好。

由上述分析可知,以上几种并行频繁项集算法同样不能很好地满足实际应用。在面临海量数据时,当前的频繁项集挖掘算法的局限主要表现在以下 3 个方面:

(1) 由于数据量过大,很难得知具体数据集记录条数,进而无法确定计数值阈值来产生合适数目的频繁项集。

(2) Apriori 要多次扫描海量数据集来产生频繁 k 项集,其代价是非常大的。

(3) 由频繁 k 项集来产生候选 $k+1$ 项集,运算量大,需要内存大,耗时多。

针对以上几点,本文提出了一种基于 MapReduce 编程模型的随机抽样的并行关联挖掘算法。

3 MapReduce 编程模型

MapReduce 编程模型[13]采用"分布治之"思想,把对大规模数据集的操作,分发给 1 个主节点管理下的各分节点共同完成,然后能整合各分节点的中间结果,得到最终结果,如图 2 所示。它有 2 个重要函数,可以由用户编写,即 map 和 reduce。map 负责把任务分解成多个任务,reduce

负责把各个任务处理的结果汇总起来。至于在并行编程中的其他各种复杂问题,如分布式存储、工作调度、负载均衡、容错处理及网络通信等,均由MapReduce 框架负责处理,可以不用程序员操心。

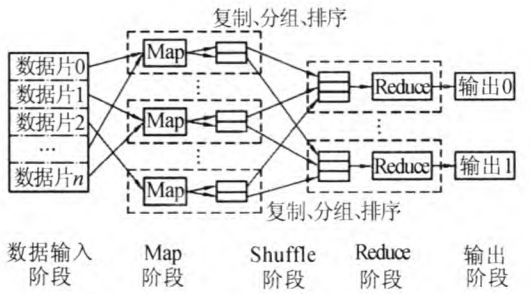

图 2　MapReduce 处理大数据集的过程

4　并行随机抽样算法

在传统频繁项集算法中,处理海量数据时,因为内存问题无法容纳数据和提供对某规模项集进行计数所需空间,要计算所有频繁项集不可避免需要进行 k 次扫描。因此可利用数据抽样样本而非数据全集来进行计算。但在现实生活中,往往很难预知所处理的海量数据所包含的记录数目(1 条记录相当于 1 个购物篮),而数据量过大时,扫描 1 次代码代价非常高。所以本文提出一种在未知数据记录数目时,单次扫描即可实现随机抽样[14]并且得到样本数据的记录数目的并行算法,有效地解决了上述局限问题,保证了后续关联挖掘算法的支持度计算的正确性。设计思想为:先保存前 k 个元素(k 为样本记录数目),从第 $k+1$ 个元素开始,以 $1/i$ ($i=k+1, k+2,\cdots,N$) 的概率选中第 i 个元素,并随机替换掉一个已保存的记录,则遍历 1 次得到 k 个元素,可以保证完全随机选取。

证明　当 $n=k$ 时,由前 k 个数放入蓄水池可知,每个样本的取出概率均相等,即 $k/k=1$。设当前样本号为 n,其每个取出样本概率均相等,即 k/n,要证明这种情况对于 $n+1$ 也成立。

由于以 $k/(n+1)$ 来决定是否把 $n+1$ 放入蓄水池,对于 $n+1$ 其出现在蓄水池中的概率为 $k/(n+1)$,对于前 n 个元素中的任意元素 m ($k+1\leqslant m\leqslant n$),其出现在蓄水池中的概率为:

$$\frac{k}{m}\left[\left(\frac{k}{m+1}\frac{k-1}{k}+\frac{m+1-k}{m+1}\right)\times\cdots\times\left(\frac{k}{n+1}\frac{k-1}{k}+\frac{n+1-k}{n+1}\right)\right]=$$

$$\frac{k}{m}\left(\frac{m}{m+1}\frac{m+1}{m+2}\cdots\frac{n}{n+1}\right)=$$

$$\frac{k}{m}\frac{m}{n+1}=\frac{k}{n+1}。$$

可见,对于 $n+1$ 每个样本取出概率也相等,即为 $k/(n+1)$。证毕。

要实现该并行抽样算法,只需在 MapReduce 框架中编写 mapper 即可。在 map 函数中,用户定义 1 个数组保存选中的 k 个元素,待扫描完所有元素后,在析构函数中将数组中的数据写到磁盘中。

5　云关联算法

运用上述并行抽样算法,在所得样本中挖掘频繁项集。本文提出了一种能运用 MapReduce 编程框架实现的并行频繁项集算法,在很大程度上提高了挖掘效率。

首先将从并行抽样算法中得到的结果存放在分布式文件系统中,此处采用 Hadoop 的 HDFS[15]。利用 MapReduce 框架,可以实现对样本的自动分块,即 HDFS 的 block(默认为 64 M)做为 1 个 map 任务的输入,这样可以实现多个 map 并行运行。map 函数的输入为 1 条记录,输出为{key:项集,value:1}。由于在 map 中会产生大量的{项集,1}的 key-value 对,从而会导致和 reduce 任务的通信负载过重。

为了进一步提高效率,在 map 过程中添加 combine 过程,其主要是先对每个 map 任务进行相同 key 的 value 值的合并,以减少通信负载。在 reduce 中对这些键值对进行规约,输出支持度大于最小支持度阈值的项集,即频繁项集。再由频繁 k 项集生成候选 $k+1$ 项集,同样采用 MapReduce。

其主要思想为:由于此频繁 k-项集是上一步 MapReduce 的输出,已经是去重有序的。本文将 k-项集的前 $k-1$ 项称为主模式,第 k 项称为从模式。在 MapReduce 过程中实现将主模式相同项集合并,进而得到 $k+1$ 项集。具体实现步骤如下:频繁 k-项集作为 map 任务的输入,map 任务输出(前 $k-1$ 项集,第 k 项集)的键值对,最后 reduce 任务进行规约,输出为{key = 主模式,value=2 个从模式}。

对 reduce 输出结果进行处理,得到 $k+1$ 项集。最后将其作为下一次 MapReduce 的 map 任务的 Distributed Cache[16] 和样本数据一起输入

到 map 中继续计算频繁 $k+1$-项集。其流程如图 3 所示。

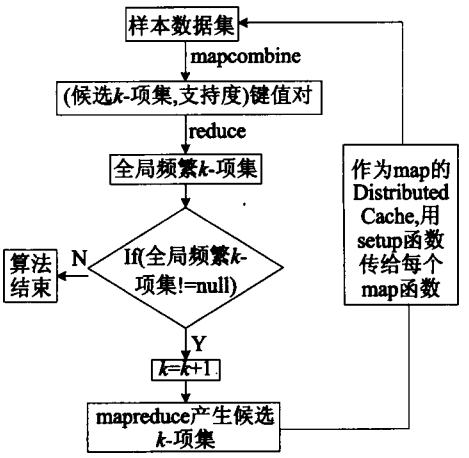

图 3 并行云关联算法流程

图 3 是一个迭代实现的过程,其结束条件为得到的频繁 k-项集为空[17],则根据频繁项集的单调性,不可再有更大的频繁项集。其伪代码如下:

```
while (频繁 k 项集 ! =null){
    //first mapreduce to get frequent k-items
    map 函数
    输入:样本数据和前一次产生的候选 k 项集(k>2)
    输出:{k 项集,支持度计数}
    map(){
        for each 项集 in value
            key=项集 value=1
            emit(key,value)
        }
    reduce 函数
    输入:{k 项集,支持度计数}
    输出:{k 项集,全局支持度计数}
    reduce(){
        sum=0
        for each value in values
          sum=+1
        if sum> minsup
          emit(key,sum);
    }//combine 同 reduce()函数除去判断语句
    //second mapreduce to produce k+1 candidate itemsets
    map 函数
    输入:{k 项集,全局支持度计数}(前一次 mapreduce 的输出)
    输出:{前 k-1 项集,第 k 项集}
    map(){
        string[] fields=value. toString(). split("\t")
        k=value. toString(). split(")\t"). length
        for i=0,i <k-1;i++
            key+=fields[i]
            value=fields[k-1]
            emit(key,value)
        }
    reduce 函数
    输入:{前 k-1 项集,第 k 项集}
    输出:{前 k-1 项集,list(第 k 项集)}
    reduce(){
        String sum=null
        for each value in values sum=+value
        emit(key,two of sum);//从 sum 中按排序取 2 个项集组合,此处有可能有多个 emit。
}
```

6 实现与分析

本文中所有的实验均在 Hadoop 平台上运行。其中实现时分别由 2、4、6、8、10 台机器组成集群。这些机器均为 AMD4234 双路 12 核构成,内存为 16 G。Hadoop 版本为 hadoop-0.20.2-cdh3u3,Java 版本是 1.6.0_31。每台机器之间用千兆以太网通过交换机连接。

实验数据是收集的应用数据,维度为 30 维,其大小约为 150 G。其中有些记录为不完全记录,即为脏数据。首先应用并行随机抽样对其进行清洗和抽样,提取其中要挖掘的一维数据。设 $k=100\ 000$(上述并行随机算法中的参数),则总共有 465 个 map task。整个抽样过程花费时间仅为9 419 ms,约为 2 min。在该样本数据上分别运行改进的并行关联云挖掘算法和传统的串行 Apriori 算法,得出其加速比。加速比是同一个任务在单处理器系统和并行处理器系统中运行消耗的时间的比率,用来衡量并行系统或程序并行化的性能和效果。

采用不同数目的结点进行测试,其结果如图 4 所示。

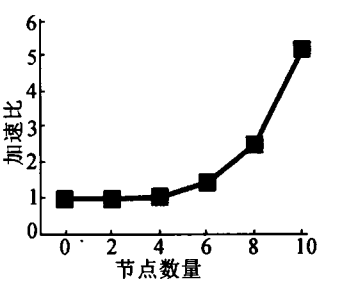

图 4 云关联算法加速比

图 4 结果表明,随着节点数量的增加,其运行效率成正比上升,这充分说明了并行关联云挖掘的算法非常适合进行海量数据的挖掘。

为了验证在样本空间所发现的频繁项集的正确性，本文对比在完整数据集上发现的频繁项集，给出其频繁结果的比例，见表 1 所列。

表 1　样本频繁项集占完整数据频繁项集比例

样本条数/10^4	样本 1	样本 2	样本 3	样本 4
0.5	0.31	0.28	0.24	0.27
1	0.52	0.53	0.50	0.56
5	0.60	0.54	0.67	0.62
10	0.65	0.68	0.71	0.63
15	0.69	0.64	0.69	0.70
20	0.78	0.71	0.79	0.75
50	0.86	0.89	0.85	0.87

为了保证其实现的准确性，分别对每种样本的大小，进行了 4 次随机抽样并频繁挖掘。由表 1 知，随着样本数的增加，其频繁项集的覆盖率正比例上升。可见，本文所提的云关联算法可以很好地适应那些挖掘实时要求高、挖掘精确性要求次之的应用。

7　结束语

本文提出了在 Hadoop 上运用 MapReduce 并行框架实现的单次扫描的随机抽样算法和利用抽样结果和 MapReduce 框架进行频繁项集发现的并行云关联挖掘算法。实验证明了该算法的高效性和正确性。由于是在样本数据集上进行关联挖掘，所以不可能保证会产生整个数据集上频繁的所有项集，也不能保证只产生整个数据集上的频繁项集。后续的工作将对这方面进行优化，以实现高效去除伪反例和伪正例[18]。

[参　考　文　献]

[1] Agrawal R, Imielinski T, Swami A. Ming associations between sets of items in massive databases[C]//Proc of the ACM-SIGMOD 1993 International Conference on Management of Data, 1993: 207－216.

[2] Agrawal R, Srikant R. Fast algorithms for mining association rules[C]//Int Conf on Very Large Databases, 1994: 487－499.

[3] Tan P N, Steinbach M, Kumar V. 数据挖掘导论[M]. 范明，范宏建，译. 北京：人民邮电出版社，2012: 176－180.

[4] Fang M, Shivakumar N, Garcia-Molina H, et al. Computing iceberg queries efficiently[C]//Int Conf on Very Large Databases, 1998: 299－310.

[5] Park J S, Chen M S, Yu P S. Efficient parallel data mining of association rules[C]//4th International Conference on Information and Knowledge Management, 1995: 233－235.

[6] Han Jiawei, Kamber M. 数据挖掘概念与技术[M]. 范　明，孟小峰，译. 北京：机械工业出版社，2001: 149－175.

[7] Savasere A, Omiecinski E, Navathe S B. An efficient algorithm for mining association rules in large database[C]//Int Conf on Very Large Databases, 1995: 432－444.

[8] Toivonen H. Sampling large database for association rules [C]//Int Conf on Very Large Databases, 1996: 134－145.

[9] Brin S, Motwani R, UIlman J K, et al. Dynamic itemset counting and implication rules for market basket data[C]//ACM SIGMOD International Conference on the Management of Data, 1997: 255－264.

[10] Agarwal R C, Aggarwal C C, Prasad V V V. A tree projection algorithm for generation of frequent itemsets[J]. Journal of Parallel and Distributed Computing, 2001, 61(3): 350－371.

[11] 李志云，周国祥，张丽萍. 一种挖掘大型数据库的关联规则新算法[J]. 合肥工业大学学报：自然科学版，2010, 33(2): 204－207.

[12] 赵　虎. 云计算环境下的关联数据挖掘算法实现[D]. 成都：电子科技大学，2011.

[13] Dean J, Ghemawat S. Mapreduce: simplified data processing on large clust[J]. Communications of the ACM, 2008, 51(1): 107－113.

[14] 高纳德. 计算机程序设计艺术 [M]. 北京：国防工业出版社，2007: 657.

[15] Apache Software Foundation. Hadoop streaming [EB/OL]. [2011-12-23]. http://hadoop.apache.org/common/docs/r0.15.2//streaming.html.

[16] 李建江，崔　健，王　聃，等. MapReduce 并行编程模型研究综述[J]. 电子学报，2011, 39 (11): 2635－2642.

[17] 秦如新，陈　静，冯一宁. 一种新的关联规则抽样算法[J]. 中国农业大学学报，2007, 12(3): 85－88.

[18] Rajaraman A, Jeffrey D U. Mining of massive datasets [M]. Cambridege: Cambridge University Press, 2012: 167.

（责任编辑　闫杏丽）