基于 MapReduce 并行关联挖掘的网络入侵检测

徐胜超[1],宋 娟[2],潘 欢[2]
(1. 广东财经大学华商学院 数据科学学院,广东 广州 511300;
2. 宁夏大学 宁夏沙漠信息智能感知重点实验室,宁夏 银川 750021)

摘 要:随着海量大数据的出现,关联数据挖掘算法需要新型计算模式来提高计算速度与运行效率。提出了基于 MapReduce 并行关联挖掘的网络入侵检测方法 Cloud-Apriori。Apriori 是一种基于频繁项集的关联规则数据挖掘算法,Cloud-Apriori 是经 MapReduce 云计算并行化后的新算法。Cloud-Apriori 利用开源的 Hadoop 分布式计算框架,采用 Hadoop 分布式文件系统存储海量数据;结合 MapReduce 的映射,规约操作,可以把关联挖掘的数据流和任务组成一个有向无环图,方便专业技术人员按照映射-规约的方式进行分布式计算的编程。分析了基于 MapReduce 的并行关联挖掘的模块组成与实现过程。Cloud-Apriori 利用 Kddcup 的案例数据和网络入侵检测这种大数据应用来仿真算法的效果。实验结果表明:与存在的网络入侵检测算法相比,Cloud-Apriori 在检测精度、运行时间上有很好的优势。

关键词:云计算;网络入侵检测;关联数据挖掘;映射-规约;并行化
中图分类号:TP393.093 **文献标识码**:A **文章编号**:1673-629X(2021)06-0123-06
doi:10.3969/j.issn.1673-629X.2021.06.022

Network Intrusion Detection Based on Parallel Association Mining of MapReduce

XU Sheng-chao[1], SONG Juan[2], PAN Huan[2]
(1. School of Date Science, Huashang College, Guangdong University of Finance & Economics, Guangzhou 511300, China;
2. Ningxia Key Lab of Intelligent Sensing for Desert Information, Ningxia University, Yinchuan 750021, China)

Abstract: With the emergence of massive big data, the association data mining algorithm needs a new computing mode to improve the calculation speed and operation efficiency. We propose a network intrusion detection based on MapReduce parallel association mining called Cloud-Apriori. Apriori is a data mining algorithm for association rules based on frequent item sets, and Cloud-Apriori is a new algorithm after the parallelization of MapReduce cloud computing. Cloud-Apriori uses open source Hadoop distributed computing framework and uses Hadoop distributed file system to store massive data. Combined with the mapping and protocol operation of MapReduce, the data flow and tasks of association mining can be formed into a directed acyclic graph, which is convenient for professional and technical personnel to carry out the programming of distributed computing in the way of map-protocol. The design and implementation of the data mining model based on Cloud-Apriori is described and discussed. A serial of experiments are also done using Kddcup datasets and the intrusion detection processing of big data. Experiment shows that the overall detection effect and executing times of Cloud-Apriori are significantly better than the existing intrusion detection algorithms.

Key words: cloud computing; network intrusion detection; association data mining; MapReduce; parallelization

0 引 言

随着计算机网络的发展及新型网络协议的出现,近年来各类网络入侵攻击行为持续增加,普通的数据挖掘算法已经不能满足网络入侵检测的需求。作为数据挖掘的重要分支,关联规则挖掘已经成为了研究热点[1-4]。关联规则算法中,很多都能够产生频繁项集,在这些关联挖掘产生频繁项集的过程中,连接数据库和产生侯选项集的数目非常大,这样将大大增加了关联规则算法的时间代价,降低了效率。例如经典的 Apriori 算法增强了丰富的颜色到数据的分析与处理,其目的是为了发现不同的数据项在历史交易数据库中的关系,核心功能是通过重复扫描数据项,获得频繁项

收稿日期:2020-08-11 **修回日期**:2020-12-15
基金项目:国家自然科学基金项目(青年基金)(F030203 61403219);广东财经大学华商学院校内导师制项目资助(2020HsDs04)
作者简介:徐胜超(1980-),男,硕士,讲师,研究方向为计算机网络和云计算。

集和研究项集之间的关系。目前,Apriori 算法和并行化的 Apriori 算法都得到了很大的改善[5-7]。

关联规则模型中的 Apriori 算法主要应用在大量的历史交易数据中,用来发现频繁模式和数据项之间的关联性。尽管如此,考虑到 Apriori 算法的固有的特点,当数据容量十分大的时候,它的缺点和不足体现得更加明显,此时使用 Apriori 算法来分析和处理数据将变得非常消耗时间与内存空间,效率低下[8]。所以为了快速处理大容量数据,文献[9]提出了基于 Apriori 算法的重要状态确定方法来完成网络入侵检测。文献[10]提出了改进的 Apriori 算法来完成频繁模式树的建立,将其应用到数据挖掘领域。

针对早期的低效率的传统 Apriori 算法而言,要处理网络入侵检测的海量大数据,目前比较常见的是 Hadoop 分布式计算框架。同时改进早期经典的 Apriori 算法,建立新的框架用来处理频繁项集的挖掘的并行 Apriori 算法,用来加快计算速度,节省内存空间。该文提出的基于 MapReduce 云计算的并行关联挖掘的网络入侵检测方法 Cloud-Apriori 就是基于这个思路。Cloud-Apriori 的实验通过构造 Hadoop 集群来完成。实验结果表明改进的并行 Apriori 算法具有很高的检测精度和更少的运行时间,提高了网络入侵检测效率。

1 Hadoop 框架分析

1.1 分布式框架

文中关联挖掘的网络入侵检测采用了 Hadoop 云计算平台,这样可以并行处理频繁项集,以进一步提高大数据处理的速度与效率[11]。

Hadoop 是一种开源并行大数据处理的总体支撑平台,一般基于 MapReduce 的并行算法中采用了 6 个步骤来表示任务流 Job stream 的流程情况。每个任务流都表示为一系列的 Map 任务或者 Reduce 任务,并行关联挖掘的 Apriori 算法作为一个应用程序,其包括的所有子任务组成了一个有向无环图,如图 1 所示[12]。在部署 Hadoop 的时候,主控节点部署为 JobTracker,工作机一般部署为 TaskTracker。因特网上有很多关于 Hadoop 的文献,由于篇幅原因这里不再过多累述,这 6 个具体步骤如下:

(1)对输入的大数据的样本数据集文件进行设置与分片;

(2)主节点(JobTracker)调度从属节点(TaskTracker)执行 Map 子任务;

(3)从属节点读取输入源片段;

(4)从属节点执行 Map 子任务,并将临时结果文件保存在本地;

(5)主节点调度从节点执行 Reduce 子任务,Reduce 阶段的从属节点读取 Map 子任务的输出文件;

(6)执行 Reduce 子任务,将最后的结果保存到 HDFS 分布式文件系统当中。

有了这 6 个步骤,并行关联挖掘的 Apriori 算法的编程人员就可以摆脱本身分布式计算的编程细节,可以使用高级语言在很短的时间内,完成分布式计算的编程。

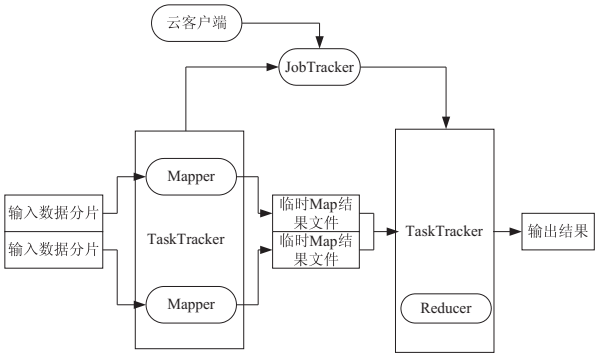

图 1 基于 MapReduce 的并行关联挖掘任务流

1.2 MapReduce 模型分析

Map 操作和 Reduce 操作是 MapReduce 并行编程模型的关键操作,所有的任务流必须经过这 2 个阶段。MapReduce 中的任务流可以通过下面的公式表示:

$$DAG = (W, E, DAG_{info}) \tag{1}$$

$$W = \{W_{name}, \{Map\}, \{Reduce\}, Param, Input, Output\} \tag{2}$$

其中,W 表示被并行处理后的任务的集合,W_{name} 表示子任务的名称;Map 和 Reduce 分别表示映射和规约操作;Param 表示任务执行所需要的参数信息;Input 和 Output 分别表示输入任务和输出任务相关的数据信息;E 表示在 DAG 图中的两个任务之间的边;DAG_{info} 表示 DAG 图中的特殊鉴别信息。

公式(2)中的 Map 操作可以更详细地表达如下:

$$Map = (M_{name}, InK, InVal, OutK, OutVal, Properties) \tag{3}$$

其中,M_{name} 表示 Map 操作的名称;InK 和 InVal 分别表示在输入 Map 操作过程中 Key-value 关键字与关键字值之间的映射对;OutK 和 OutVal 分别表示在输出 Map 操作过程中 Key-value 关键字与关键字值之间的映射对;Properties 表示在 Map 操作过程中的属性参数。公式(2)中的 Reduce 操作可以更详细地表达如下:

$$Reduce = (R_{name}, InK, InVal, OutK, OutVal, Properties) \tag{4}$$

其中,R_{name} 表示 Reduce 操作的名称,后面的参数 InK 和 InVal 等的含义都与 Map 操作中是一致的。E 表示在 DAG 图中的两个任务之间的边。E 的详细描述

如下:

$$E = (\text{Path}, \text{StartTK}, \text{EndTK}) \quad (5)$$

其中,Path 显示了数据流的传输路径,StartTK 表示当前的任务,EndTK 表示任务的子任务。

2　Cloud-Apriori 并行关联规则挖掘模型

2.1　基本模快

Cloud-Apriori 的关联规则数据挖掘如图 2 所示。

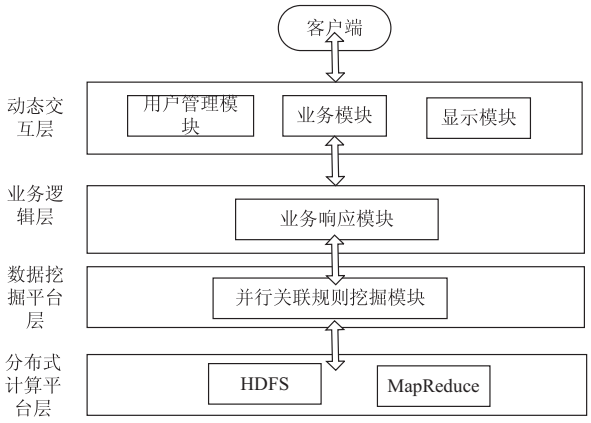

图 2　基于 Cloud-Apriori 的并行关联挖掘模型

主要包括:

(1)动态交互层:用来与客户端通过接口进行交互;

(2)业务逻辑层:用来完成针对交互层和数据挖掘平台层的所有业务处理操作的命令和控制;

(3)数据挖掘平台层:是该文基于 Hadoop 的关联规则算法的核心层,具体完成大数据的处理,包括并行数据挖掘算法,工作流模块,数据装载模块和存储模块;

(4)分布式计算的平台层:它是利用 Hadoop 框架来获得 HDFS 分布式文件的存储和 MapReduce 的工作的执行。

2.2　关联规则 Apriori 算法原理

关联规则表示两个实体在某种规则下的一些隐藏信息,关联挖掘的目的是发现这些隐藏的关系[13]。关联规则可以通过数据模型来描述,例如 X 表示关联规则的前提假设,Y 表示后续的关联规则。另外,关联规则算法有一个关于支持度(support)和置信度(confidence)的定义,如公式(6),这里 A 和 B 分别表示了不同的事件。

$$P(A \mid B) = \frac{P(AB)}{P(B)} \quad (6)$$

关于支持度和置信度的通信关系可以通过公式(7)来计算:

$$\text{conf}(Y \mid X) = P(Y \mid X) = \frac{P(XY)}{P(X)} \approx \frac{\text{The number of times } XY \text{ appears}}{\text{The number of times } X \text{ appears}} \quad (7)$$

关联挖掘的 Apriori 算法的基本思想是使用频繁项集的优先知识信息来完成迭代计算,该过程是层对层的搜索进行的[14]。

为了实现在 Apriori 算法下的关联分析,一般的关联挖掘的步骤如下:

步骤 1:设置最小的需求支持度 min_Sup 和最小的置信度 min_Conf;

步骤 2:顺序地连接和扫描数据集,确定每个数据项的支持数量,选择出在步骤 1 中的最小支持度 min_Sup 的频繁项集 1;

步骤 3:随机地结合 2 个频繁项集 1 来生成侯选的频繁项集 2,然后顺序地连接数据集和完成对侯选的频繁项集 2 的支持度的计算,最后根据步骤 2 过滤频繁项集 2;

步骤 4:重复步骤 3 直到产生了空的最高序号的频繁项集;

步骤 5:输出关联挖掘的结果,算法结束。

2.3　Cloud-Apriori 挖掘算法的设计

前面提到 MapReduce 编程模型采用了主-从(Master/Slave)模式,即主控节点上可以部署 Job Tracker,在从节点上可以部署 Task Tracker。在 DAG 的有向无环图中,为了更好地分配正确的任务流到所有的处理节点,该文采用并行 MapReduce 的任务流处理技术,具体的实现机制如图 3 所示。

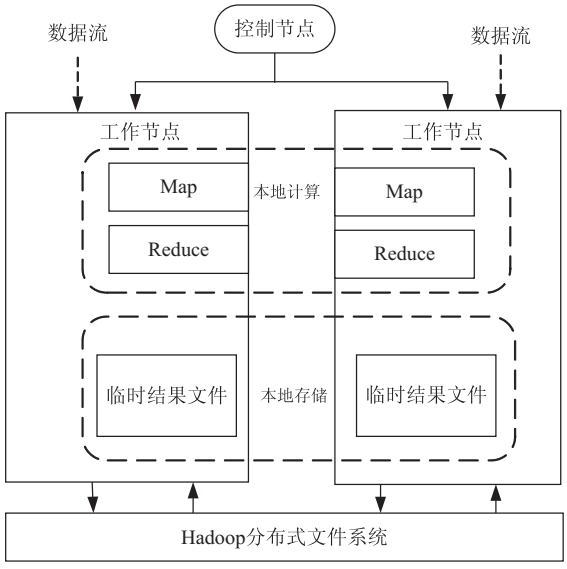

图 3　MapReduce 的并行关联挖掘工作机制

2.4　Cloud-Apriori 挖掘算法的实现

(1)产生频繁项集。

文中算法采用了自顶向下的方式,从概念的第一层开始到最后一层,利用 MapReduce 产生大量的频繁项集,包括在不同的概念层之间的层次交互。对于每一层来说,MapReduce 操作产生一个频繁集的数据项,包括特殊层的交互,这种操作将一直迭代,直到在本层

中不能发现更多的频繁项集为止。

在数据入口的处理中,主要使用了三个类:输入格式 Input format、输入分片 Input split 和记录阅读器 Record Reader。在这三个类中,输入格式又被输入数据处理一次,每个输入的分片在后面的类中处理。记录阅读器的功能是对数据块的输入分片进行分析,并将其信息对接到多个 key-value 关键字和关键值的映射对中,并发送到 Map 函数。与任务相关的关键字和关键值的映射<key,value>如表 1 所示。

表 1 关键词和关键值的映射表描述

输入/输出	映射函数	规约函数
Input key/value	key:Data offset value:key The content of the line	key:Local frequent itemsets value:min_Sup
Wine	key:Local frequent itemsets value:min_Sup	key:Global frequent itemsets value:min_Sup

(2)Map 和 Reduce 的处理。

当使用 Hadoop 来完成大数据处理的时候,在文件中的每行都被作为交易记录对待,在行中的每个入口都被一个空格符号分离开。在客户端提交了文件到 HDFS 后,文件中的数据块的默认尺寸是 64 MB,每个迭代的轮次都执行一个 MapReduce 任务。每个 Map 节点同时处理多个数据块,处理完后输出到候选集及其对应的支持度。

Reduce 操作持续地处理 Map 操作阶段的结果,所有的具有相同的键值的关键字-关键值对都将完成规约操作,最后完整的频繁项的支持结果将完成。这些并行的规约过程都采用 MapReduce 结构设计,包括 Map( )函数和 Reduce( )函数,它们都运行在 Hapdoop 平台上,加快了关联规则挖掘的计算速度。下面是 Map( )函数和 Reduce( )函数的伪代码,这些代码经过少量的修改即可在 Java 语言环境下执行。

下面的 Map_Class 是 Map 阶段的类描述代码:

```
Map_Class{
1. map(key,value){
2.    Sort=0;
3. D_is=Max_Value;
4. for(int i=1;i<k;i++){
5. Records D_is=d_is(i,pointer);
6. if(Records D_is<min D_is){
7.    Sort=i;
8. min D_is=Records D_is;
9. }
10.   }
11. produce<"Sort" , value>;
12. }
13.   }
```

下面的 Reduce_Class 是 Reduce 阶段的类描述代码:

```
Reduce_Class{
1. reduce(key,value){
2.    D1=0;
3.    Sort=k;
4.    Temps=new int[D1];
5.    for(int i=0;i<Col;i++){
6.    for(int D1=0; D1<value.length;
7.    D1++){
8. Temps[i]=value[D1][i];
9.    }
10. }
11.    for(int j=0;j<Col;j++){
12. pointer+=Temps[j];
13.    }
14. produce<key,pointer>;
15.    }
16. }
```

(3)计算复杂度分析。

如果 i 是原始数据集 D 的交易数,j 表示每个交易记录的平均长度,采用传统的 Apriori 关联挖掘算法,在 L1 层中计算时间复杂度为 $T_1 = O(i*j)$,整个算法的时间复杂度为:

$$T = O(i*j) + \sum_{k\geq 2}[O(|L_{k-1}|*|L_{k-1}|) + O(|C_k|*|L_{k-1}|) + O(i*|C_k|)] \quad (8)$$

如果采用基于 MapReduce 的并行关联挖掘的 Apriori 算法,Hadoop 集群有 a 个节点,每个节点都操作 1 个数据块,则获取频繁项集的整个时间复杂度可以表示为:

$$T = \{O(i*j) + \sum_{k\geq 2}[O(|L_{k-1}|*|L_{k-1}|) + O(|C_k|*|L_{k-1}|) + O(i*|C_k|)]\}/a \quad (9)$$

从公式(8)和公式(9)比较可以看出,从理论上讲,基于 MapReduce 的并行 Apriori 关联挖掘算法可以很好地降低执行时间。集群的物理节点个数越多,则执行时间越短,效率越高。

3 系统测试与性能分析

3.1 测试环境

测试平台的具体参数包括:8 个计算节点(Intel i7

的 3.2 GHz 主频处理器,8 GB 内存。节点之间的通讯为 Intel 82574L 芯片组主板,双网卡,带宽为 1 GB/S。

Hadoop 分布式平台的版本为 2.2,JDK 的版本为 1.8.0。其中一个节点上部署 JobTracker,另外 7 个节点上部署 TaskTracker。每个 TaskTracker 上有 1 个 reduce 工作 slot 和 2 个 map 工作 slot,每个物理主机的 IP 地址配置如表 2 所示。

表 2　Hadoop 云平台的物理主机的配置

服务节点名称	角色	节点 IP 地址	运行的进程
Hadoop0	Master	10.20.01.10	namenode/secondaryna-menode/job Tracker
Hadoop1	Slave	10.20.01.21	datanode/task Tracker
Hadoop2	Slave	10.20.01.22	datanode/task Tracker
Hadoop3	Slave	10.20.01.23	datanode/task Tracker
Hadoop4	Slave	11.30.01.31	datanode/task Tracker
Hadoop5 ~ 7	Slave	11.30.01.32 ~ 34	datanode/task Tracker

所有物理主机上都安装 Linux 操作系统,Hadoop 的主目录安装在 usr/local/hadoop,同时在 profile 文件中修改和配置好环境变量:

```
# /etc/profile
export JAVA_HOME=/usr/local/jdk
export HADOOP_HOME=/usr/local/hadoop
export PATH=.:$HADOOP_HOME/bin:$JAVA_HOME/bin:$PATH
```

3.2　实验数据

为了比较普通的关联规则挖掘 Apriori 算法和基于 MapReduce 的并行关联挖掘 Apriori 算法的性能,文中采用了 Kddcup 数据集。Kddcup 数据集是在网络入侵检测中采用的最常见的标准 Benchmark 数据集[15],它也是一个在国内外最有代表性和影响性的网络入侵检测数据集。该数据集中的记录主要被划分为两个部分:用于训练的数据集和用于测试的数据集。训练数据集具有一个具体的鉴定符,测试数据集没有鉴定符。测试数据集也包括不在训练数据集中的一些攻击类型,这样就使得系统的网络入侵检测能力更加的真实可信。

3.3　入侵检测的性能分析

(1)精确度比较。

模拟地攻击数据包,并将其保存到日志文件中,实验中一个大约有 2 000 个攻击数据包,用来测试并行关联挖掘算法的有效性。算法的预检测引擎可以丢弃那些认为是正常的数据包,减少没有必要的异常检测。

公式(10)很好地体现了网络入侵检测系统的错误检测率(error detection rate,EDR)[16]:

$$EDR = \frac{MP}{DP} * 100\% \tag{10}$$

其中,MP 表示丢弃的攻击包的数量,DP 表示网络入侵中所有攻击包的数量。文中关联规则挖掘 Cloud-Apriori 算法完成网络入侵检测的结果见表 3 和表 4(分别表示传统的 Apriori 算法和基于 Cloud-Apriori 算法并行关联挖掘的结果)。

表 3　传统的关联挖掘的网络入侵检测

编号	所有的攻击数	异常的攻击数	检测成功的数目	异常检测结果/%
1	494 201	49 420	31 809	64
2	494 201	98 840	56 817	57
3	494 201	148 260	81 028	55
4	494 201	197 680	117 257	59

表 4　基于 Cloud-Apriori 的关联挖掘的网络入侵检测

编号	所有的攻击数	异常的攻击数	检测成功的数目	异常检测结果/%
1	494 201	49 420	32 722	66
2	494 201	98 840	56 213	57
3	494 201	148 260	82 303	56
4	494 201	197 680	120 132	61

通过比较表 3 和表 4 可以看出,基于 Cloud-Apriori 算法的网络入侵检测的性能只有少量的改善,不是很明显,这是因为文中的并行关联挖掘算法只是在任务执行阶段完成了并行处理操作,所以整个算法的检测精度只有少量提高。

(2)入侵检测速度和效率的比较。

前面的实验主要针对检测精度,这部分完成传统的 Apriori 关联规则挖掘算法和基于 MapReduce 的 Apriori 并行关联挖掘的算法的时间比较,如表 5 所示。

表 5　传统和并行的网络入侵检测速度的比较

数据尺寸/TB	传统 Apriori 挖掘/s	基于 Cloud-Apriori 的挖掘/s
0.2	214	352
0.6	508	485
1.0	962	621
1.4	1 147	689
1.8	Out of memory	737
2.2	Out of memory	752

从表 5 可以看出,当需要检测的数据尺寸比较小的时候,MapReduce 的 Apriori 关联挖掘的优势还体现不出来,它的消耗时间有时比传统的 Apriori 关联挖掘还要多。尽管如此,随着数据容量的不断增加,基于 Cloud-Apriori 算法的关联挖掘逐步体现出它的优势,特别是当遇到海量大数据的时候,传统的 Apriori 关联

挖掘甚至无法完成网络入侵检测，而基于 Cloud-Apriori 算法的关联挖掘可以充分利用分布式并行计算的优点，它的执行过程不受数据容量的限制，这也是早期很多传统的软件都被淘汰的原因，不适应近年来的大数据、云计算技术的发展。

为了进一步验证和分析基于 MapReduce 的并行关联挖掘的网络入侵检测方法的执行时间，改变了物理主机的数量，图 4 显示了在相同的数据容量条件下，不同的物理主机数目的执行时间比较。

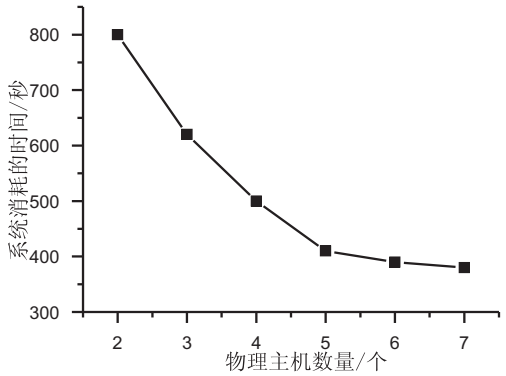

图 4 随着物理主机个数变化的执行时间比较

从图 4 可以看出，当数据容量相同的时候，物理主机从 2 增加到 7 个，执行时间平缓减少，执行效率也平稳增加。

当物理主机个数到了 6 ~ 7 个的时候，执行时间越来越平稳，表明 Cloud-Apriori 算法并行关联挖掘的算法可以很好地提高网络入侵检测的速度。它也证明 Hadoop 集群计算虽然可以提高速度，但是加速的比例是缓慢的，因为大数据处理是一种数据密集型应用，不能线性地降低处理时间。

4 结束语

该文提出了基于 MapReduce 并行关联挖掘的网络入侵检测算法，利用 Benchmark 数据集和网络入侵检测这种大数据应用对算法效果进行了仿真。结果表明，Cloud-Apriori 算法比传统的 Apriori 算法有更好的分类精度和更少的运行时间。但是，Hadoop 是一个比较低版本的开源的云平台，下一步的工作将关联挖掘 Apriori 算法移植到更高级别的云计算平台，以进一步提高关联规则挖掘算法的速度与效率。

参考文献：

[1] YE Y，TAO L，ADJEROH D，et al. A survey on malware detection using data mining techniques[J]. ACM Computing Surveys，2017，50(3)：1-40.

[2] BUCZAK A L，GUVEN E. A survey of data mining and machine learning methods for cyber security intrusion detection [J]. IEEE Communications Surveys & Tutorials，2017，18(2)：1153-1176.

[3] 宋 杨. 基于关联规则的并行优化算法研究[D]. 哈尔滨：哈尔滨工程大学，2016.

[4] 高 鹏. 云计算环境下的入侵检测算法研究[D]. 保定：河北大学，2016.

[5] JIA Y，XIA G，FAN H，et al. An improved Apriori algorithm based on association analysis[J]. Journal of Bacteriology，2012，15(15)：208-211.

[6] 杨 青，张亚文，张 琴，等. 基于 Hadoop 的多维关联规则挖掘算法研究及应用[J]. 计算机工程与科学，2019，41(12)：2127-2133.

[7] 张 玲. 基于 Hadoop 平台并行关联规则挖掘算法研究[D]. 西安：西安科技大学，2017.

[8] SAKAI H，WU M，NAKATA M. Apriori-based rule generation in incomplete information databases and non-deterministic information systems[J]. Fundamenta Informaticae，2012，130(3)：343-376.

[9] KHALILI A，SAMI A. SysDetect：a systematic approach to critical state determination for industrial intrusion detection systems using Apriori algorithm[J]. Journal of Process Control，2015，32(11)：154-160.

[10] BHANDARI A，GUPTA A，DAS D. Improvised Apriori algorithm using frequent pattern tree for real time applications in data mining[J]. Procedia Computer Science，2015，46：644-651.

[11] WEI L，ZHU H，CAO Z，et al. Security and privacy for storage and computation in cloud computing[J]. Information Sciences，2014，258(3)：371-386.

[12] CASSALES G W，CHARÃO A S，KIRSCH-PINHEIRO M，et al. Improving the performance of Apache Hadoop on pervasive environments through context-aware scheduling[J]. Journal of Ambient Intelligence & Humanized Computing，2016，7(3)：333-345.

[13] PARK S H，SYNN J，KWON O H，et al. Apriori-based text mining method for the advancement of the transportation management plan in expressway work zones[J]. Journal of Supercomputing，2017，74(3)：1-16.

[14] HE H，DU Z，ZHANG W，et al. Optimization strategy of Hadoop small file storage for big data in healthcare[J]. Journal of Supercomputing，2016，72(10)：3696-3707.

[15] SIDDIQUE K，AKHTAR Z，KHAN F A，et al. KDD cup 99 data sets：a perspective on the role of data sets in network intrusion detection research[J]. Computer，2019，52(2)：41-51.

[16] ASHFAQ R A R，WANG X Z，HUANG J Z，et al. Fuzziness based semi-supervised learning approach for intrusion detection system[J]. Information Sciences，2017，378(C)：484-497.