DOI:10.3969/j.issn.1673-4785.201305011
网络出版地址:http://www.cnki.net/kcms/detail/23.1538.TP.20140221.1804.001.html

检测僵尸网络的贝叶斯算法的 MapReduce 并行化实现

邵秀丽[1],刘一伟[2],耿梅洁[1],韩健斌[3]
(1.南开大学 计算机与控制工程学院,天津 300071; 2.北京大学 数学科学学院,北京 100871; 3.武警指挥学院 军事教育训练系,天津 300250)

摘 要:僵尸网络严重威胁互联网的安全,目前主流的僵尸网络检测方法准确性较低,针对此问题,考虑贝叶斯算法具有较高的准确性,提出了基于 Hadoop 平台的 MapReduce 机制的贝叶斯算法。该方法以主机对作为分析对象,提取 2 个主机对通信的流量特征,将这些特征作为贝叶斯分类算法的输入,通过并行化计算贝叶斯算法训练阶段的先验概率和条件概率形成贝叶斯分类器,使其学会辨认僵尸网络的流量。在检测阶段利用训练阶段形成的贝叶斯分类器和并行化计算后验概率,实现检测僵尸网络。通过实验表明,该方法检测僵尸网络是有效的,检测正确率在 90%以上,并且该方法较单机检测僵尸网络的贝叶斯算法效率有了较大的提高。

关键词:僵尸网络;检测僵尸网络;贝叶斯算法;Hadoop;MapReduce;流量
中图分类号:TP311 **文献标志码**:A **文章编号**:1673-4785(2014)01-0026-08

中文引用格式:邵秀丽,刘一伟,耿梅洁,等.检测僵尸网络的贝叶斯算法的 MapReduce 并行化实现[J]. 智能系统学报, 2014, 9(1): 26-33.
英文引用格式:SHAO Xiuli, LIU Yiwei, GENG Meijie, et al. The parallel implementation of MapReduce for the Bayesian algorithm to detect botnets[J]. CAAI Transactions on Intelligent Systems, 2014, 9(1): 26-33.

The parallel implementation of MapReduce for the Bayesian algorithm to detect botnets

SHAO Xiuli[1], LIU Yiwei[2], GENG Meijie[1], HAN Jianbin[3]
(1. College of Computer and Control Engineering, Nankai University, Tianjin 300071, China; 2. School of Mathematical Sciences, Peking University, Beijing 100871, China; 3. Department of Education and Training, Armed Police Command College, Tianjin 300250, China)

Abstract: The botnet network poses a serious threat to the Internet security, and the accuracy of the botnet detection method is low, while the Bayesian algorithm has high accuracy. This paper puts forward a Bayesian algorithm with the mechanism of MapReduce based on the Hadoop platform to achieve botnet detection. Taking the host-pairs as analysis objects, this method extracts the traffic features of communications between two hosts, takes these features as input and trains the Bayesian classifier through parallel calculations of the prior probability and condition probability on the stage of the Bayesian algorithm training to learn to recognize botnet traffic. By using the Bayesian classifier trained on the stage of the Bayesian algorithm training and parallel calculations of the posterior probability on the stage of detecting, the detection of botnets can be achieved. Experiments show that the method for detecting botnets is effective and the correct detection rate is more than 90%. The efficiency of this method is greatly improved as compared with detecting the single Bayesian algorithm of the botnets.

Keywords: botnets; botnet detection; Bayesian algorithm; Hadoop; MapReduce; flow

由于僵尸网络威胁着互联网的安全,其检测方法也随着僵尸网络的发展而发展。流行的僵尸网络检测技术一般是通过网络流量分析实现的,如通过 PageRank 算法实现检测;通过网络通信图识别;利用神经网络算法识别僵尸网络。这些僵尸网络检测方法或者需要依赖外部系统提供信息,不能独立完

收稿日期:2013-05-06. 网络出版日期:2014-02-21.
基金项目:天津市科技支撑计划资助项目(13ZCZDZGX02500, 12ZCZDZGX49600, 12ZCZDZGX46700).
通信作者:邵秀丽. E-mail: shaoxl@nankai.edu.cn.

成检测;或者由于网络流量信息量巨大,在单个服务器上完成检测的工作效率比较低[1]。此外,僵尸网络具有流量大的特征[2],因此贝叶斯分类训练阶段需要对大量的网络数据集进行训练,用单一结点来进行检测僵尸网络将会遇到计算时间和计算资源瓶颈[3]。为利用贝叶斯算法较高的准确性,基于云的 Hadoop 机制和 MapReduce 实现贝叶斯算法,该算法根据流量分析判断网络访问流量信息中是否存在僵尸行为[4]。基于 MapReduce 检测僵尸网络的贝叶斯算法,把贝叶斯算法训练阶段的先验概率、条件概率和检测阶段的后验概率的计算并行化处理。该方案把捕获的网络流量利用云环境的贝叶斯并行算法进行分析处理,最终检测出僵尸网络[5]。

1　检测僵尸网络的计算架构

图 1 给出了云环境下检测僵尸网络的架构,由被测网络环境、云环境和代理服务器层三部分构成,这三部分协同完成僵尸网络的检测[6]。每个被测网络中有若干台机器和一个核心交换机,连接一个代理服务器,代理服务器与核心交换机连接,主要负责网络流量的采集、解析、过滤并上传到云环境中。云的 Hadoop 收集并处理各个代理服务器上传的网络流量[7]。

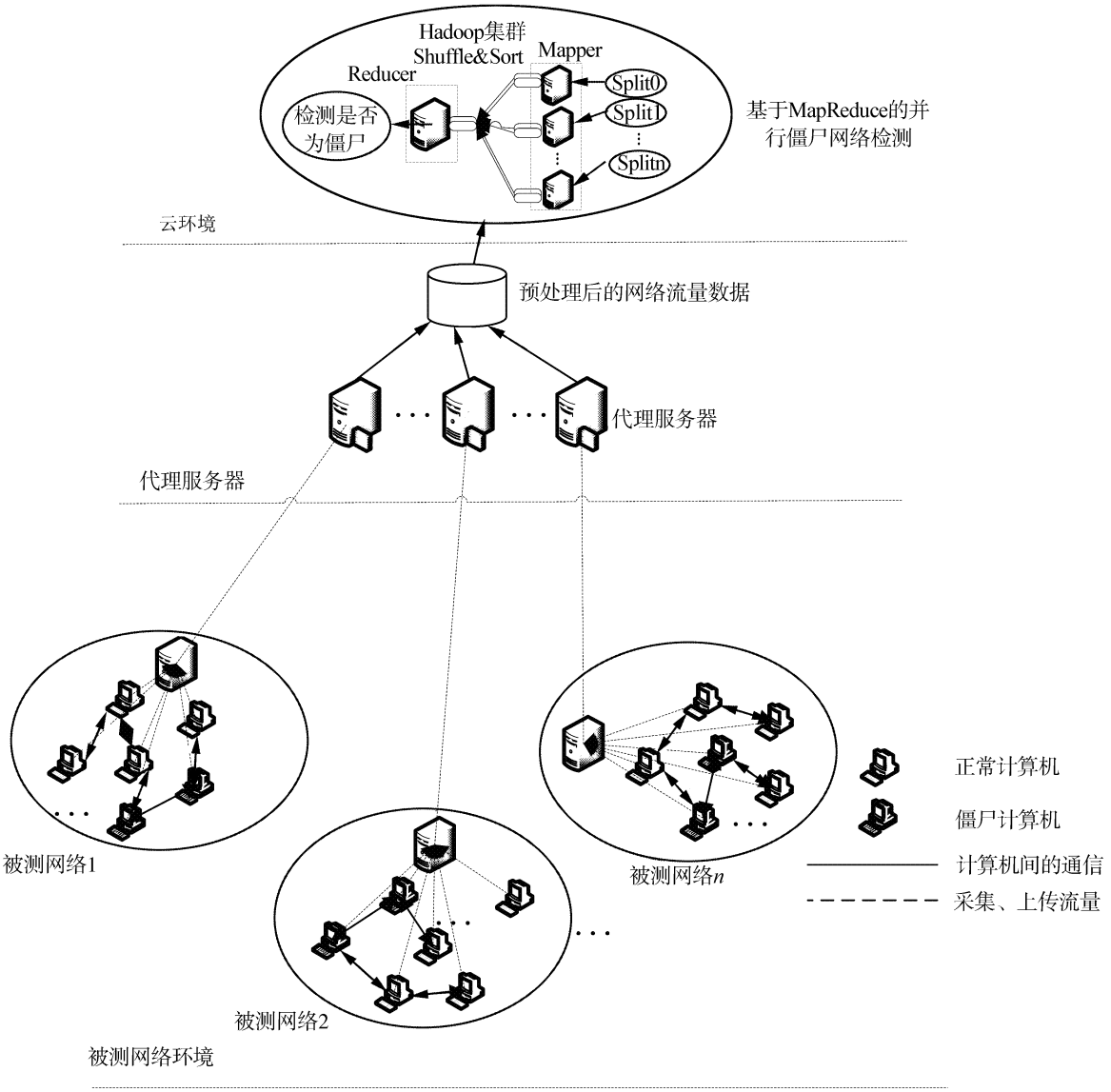

图 1　基于 MapReduce 的检测僵尸网络

Fig.1　Botnet detection based on MapReduce

代理服务器运行 tcpdump 抓包工具来抓取网络数据包,且将抓取的数据包以十六进制格式的文件存储,经过解析将文件变为可读的格式,以便程序分析处理。解析后将数据包存储在 file 中,不同协议的数据包格式不同,例如 TCP 协议的流量数据包格式为:序号|数据包到达时间|源 IP 地址|源端口号|目的 IP 地址|目的端口号|协议|数据包字节数|FIN|SYN|ACK|RST。

代理服务器将解析后的 file 上传到云的 Hadoop 中,以便用 MapReduce 并行化的贝叶斯算法进行分

析处理。file 中每一行表示一个数据包。不同被测网络的代理服务器将 file 上传到云环境的同一个指定文件夹,以便在云环境中对各个被测网络的流量集中分析处理,检测出所有被测网络中的僵尸主机[8]。

为适合 MapReduce 计算模型处理,须将 file 中的数据包进行预处理。将抓取的网络流量信息数据包处理成以行形式存储的文件[9],每行信息形式为:序号|数据包到达时间|源 IP 地址|目的 IP 地址|TCP 数据流|时间间隔平均值|时间间隔变化|数据包字节数|数据包个数平均值|持续时间平均值|类标签。其中,类标签值为 0 或者 1 来标明该条网络数据是否属于僵尸网络,本文设类标签值为 0 的网络访问为正常网络,否则为僵尸网络。

随机选择类标签值为 0 的正常网络信息行的 2/3,再随机选择类标签值为 1 的僵尸网络信息行的 2/3,这些行信息合成文件作为训练数据文件,剩余行作为检测数据文件。然后,将训练数据文件和检测数据文件分别按行分块,分块过程由 Hadoop 自动按 64 MB 大小作为一个数据块处理。

2 基于 MapReduce 的贝叶斯算法

贝叶斯算法进行 MapReduce 设计的基本思路是:贝叶斯训练阶段形成知识库,先验概率$P(b)$、$P(n)$和条件概率 $P(w_i|n)$、$P(\overline{w_i}|n)$、$P(w_i|b)$、$P(\overline{w_i}|b)(1 \leqslant i \leqslant 6)$,其中 b 表示僵尸网络,n 表示正常网络。其中 w_1 为 TCP 数据流,w_2 为时间间隔平均值,w_3 为时间间隔变化,w_4 为数据包字节数,w_5 为数据包个数平均值,w_6 为持续时间平均值。计算正常网络和僵尸网络的先验概率对应一个 MapReduce 计算过程,即 MapReduce1;对 6 个属性列既要判断是否为僵尸网络又要判断是否在阈值内,即每个属性有 4 个判断条件,因此需要求 24 个条件概率,计算这 24 个条件概率对应另一个 MapReduce 计算过程,即 MapReduce2。贝叶斯检测阶段基于由 26 个概率构成的知识库,根据贝叶斯算法公式:

$$P(b|d)=\frac{P(b)\prod_{i=1}^{n}P(w_i|b)}{P(d)}$$

$$P(n|d)=\frac{P(n)\prod_{i=1}^{n}P(w_i|n)}{P(d)}$$

计算每条网络数据的后验概率[10],进行分类并判断是否为僵尸网络。检测阶段对应一个 MapReduce 计算过程,即 MapReduce3。图 2 描述了贝叶斯算法检测僵尸网络的 MapReduce 并行化实现方法。

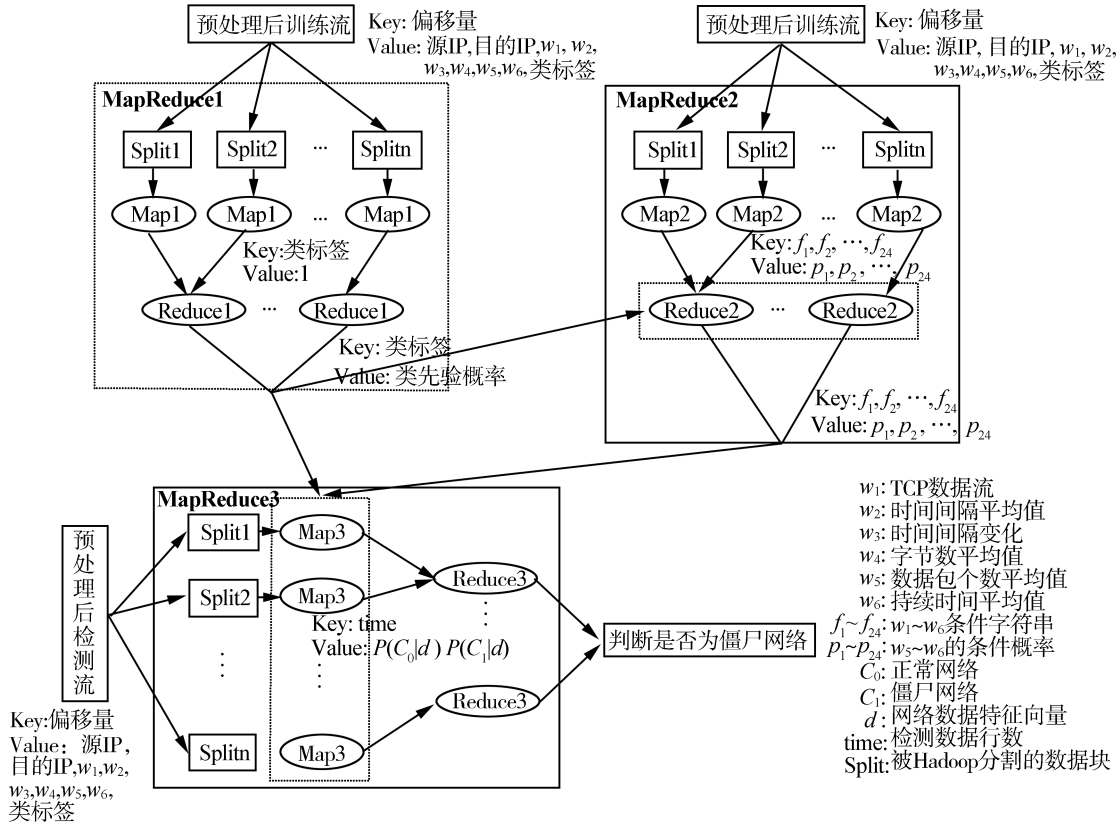

图 2 基于 MapReduce 检测僵尸网络的贝叶斯算法

Fig.2 The process of Bayesian algorithm to detect botnets based on MapReduce

2.1 MapReduce1 的设计

Map1 接收到的训练数据是被 Hadoop 处理形成的〈Key,Value〉对,形式为〈该行起始位置相对于文件起始位置的偏移量, 文本文件中的一行信息〉的信息。由于 MapReduce1 是计算贝叶斯的先验概率,只需用到 Value 的类标签属性,所以 Map1 将每行 Value 数据按空格分隔成字符串数组,取出数组最后一项,即类标签值。判断类标签值,若为 0,输出中间结果〈Key1, Value1〉对的形式为〈“正常网络”,1〉;若为 1,输出中间结果〈Key1, Value1〉对的形式为〈“僵尸网络”,1〉。并且,MapReduce 框架每执行一次 map( )说明处理一行数据,通过累加统计训练数据总行数,以成员变量 sum 存储。Map1 只是一个数据准备阶段,使 Reduce1 能在该准备数据上继续处理。Map1 过程伪代码如下。

```
输入:Object、Text。
输出:Text、IntWritable。
map(Key、Value)
{StringTokenizer itr = new StringTokenizer(value.toString());
String[ ]temp=new String[9];
While(itr.hasMoreTokens())
   {temp[i]=itr.nextToken();//属性字符串数组
      i++;
   }
Sum++;//网络数据总行数
If(temp[8].equals(0))//类标签 0 为正常网络
     Context.write(“正常网络”,1);
Else//表示为僵尸网络
     Context.write(“僵尸网络”,1);
}
```

经过 Map1 把分块的每行信息都处理成〈Key1, Value1〉形式的等待整体处理的中间文件输出,MapReduce 框架将每个 Map1 输出的中间文件的结果〈“正常网络”,1〉或〈“僵尸网络”,1〉按照 Key 值(正常网络、僵尸网络)进行分组形成新的〈Key2, Value2〉对,形式为〈类标签值,{1,1,…,1 }〉。

Reduce1 接收到的信息为〈Key2, Value2〉。Reduce1 的任务是对 Key2 相同的中间结果计数,若 Key2 值为“正常网络”,统计的 Value2 的行数为正常网络个数,并以成员变量 sum_yes 存储;若 Key2 值为“僵尸网络”,统计的 Value2 的行数为僵尸网络个数,并以成员变量 sum_no 存储。并分别用 sum_yes/sum、sum_no/sum 计算得到先验概率 $P(n)$ 和 $P(b)$,并以成员变量 sum_yes_p 和 sum_no_p 存储。Reduce1 过程伪代码如下。

```
输入:Text、IntWritable。
输出:Text、FloatWritable。
reduce(Key,Value)
{for(IntWritable val:Value)
   Sum1+=val.get();//若 Key 为“正常网络”,统计的是正常网络数据行数;否则为僵尸网络数据行数
   If(Key.equals(“正常网络”))
   {sum_yes=sum1;//存储正常网络数据行数
     sum_yes_p=sum_yes/sum;//正常网络先验概率
   }
   Else
   {sum_no=sum1;//存储僵尸网络数据行数
     sum_no_p=sum_no/sum;//僵尸网络先验概率
   }
     Context.write(key,(float)(sum1/sum));//输出先验概率
   }
```

经过 MapReduce1 的处理,形成 2 个以成员变量 sum_yes_p、sum_no_p 存储的概率,即正常网络先验概率和僵尸网络先验概率,构成知识库的一部分,供检测阶段使用。

2.2 MapReduce2 的设计

Map2 接收到的信息与 Map1 相同,是训练数据被 Hadoop 处理形成的〈Key, Value〉对,形式为〈该行起始位置相对于文件起始位置的偏移量, 文本文件中的一行信息〉的信息。MapReduce2 计算贝叶斯的条件概率,需用到 Value 的 6 个属性列及类标签值。因此 Map2 将每行 Value 数据按空格分割成字符串数组,取出数组的第 3~9 项 $w_1, w_2, \cdots, w_6$,以及类标签值。首先判断类标签值是否为“0”,然后判断各属性是否在各自阈值内。若标签值为“0”且属性值在阈值内,输出中间结果〈Key3, Value3〉对的形式为〈“$w_i|n$”,1〉;若标签值为“0”且属性值在阈值外,输出中间结果〈Key3, Value3〉对的形式为〈“$\overline{w_i}|n$”,1〉;若标签值为“1”且属性值在阈值内,输出中间结果〈Key3, Value3〉对的形式为〈“$w_i|b$”,1〉;若标签值为“1”且属性值在阈值外,输

出中间结果〈Key3, Value3〉对的形式为〈“$\bar{w}_i|b$”,1〉。Map2 过程伪代码如下。

输入:Object、Text。

输出:Text、IntWritable。

map(Key, Value)

{StringTokenizer itr = new StringTokenizer(value.toString());

String[] temp=new String[9];

While(itr.hasMoreTokens())

{temp[i]=itr.nextToken();//属性字符串数组

i++;

}

If(Float.parsefloat(temp[2])>140 && Float.parsefloat(temp[2])<150)

If(temp[8].equals(“0”))

Context.write(“$w_1|n$”,1);

Else

Context.write(“$w_1|b$”,1);

Else

If(temp[8].equals(“0”))

Context.write(“$\bar{w}_1|n$”,1);

Else

Context.write(“$\bar{w}_1|b$”,1);

//其他 5 个属性列训练过程同上。

}

经过 Map2 把分块的每行信息都处理成以〈Key3, Value3〉形式的等待整体处理的中间文件输出,MapReduce 框架将每个 Map2 输出的中间文件的结果按照 Key 值($w_i|n$、$\bar{w}_i|n$、$w_i|b$、$\bar{w}_i|b$)进行分组形成新的〈Key4, Value4〉对,形式为〈条件字符串,{1,1,…,1}〉。

Reduce2 接收到的信息为〈Key4, Value4〉。Reduce2 的任务是对 Key4 相同的中间结果计数,若 Key4 值为“$w_i|n$”,统计的 Value4 的行数 sum1 为 w_i 在阈值内且属于正常网络个数,利用 sum1/sum_yes 求得条件概率 $P(w_i|n)$,并以成员变量 wi_in_nomal 存储;若 Key4 值为“$\bar{w}_i|n$”,统计的 Value4 的行数 sum1 为 w_i 在阈值外且属于正常网络个数,利用 sum1/sum_yes 求得条件概率 $P(\bar{w}_i|n)$,并以成员变量 wi_out_nomal 存储;若 Key4 值为“$w_i|b$”,统计的 Value4 的行数 sum1 为 w_i 在阈值内且属于僵尸网络个数,利用 sum1/sum_no 求得条件概率 $P(w_i|b)$,并以成员变量 wi_in_unnomal 存储;若 Key4 值为“$\bar{w}_i|$僵尸”,统计的 Value4 的行数 sum1 为 w_i 在阈值外且属于僵尸网络个数,利用 sum1/sum_no 求得条件概率 $P(\bar{w}_i|b)$,并以成员变量 wi_out_unnomal 存储。Reduce2 过程伪代码如下。

输入:Text、IntWritable。

输出:Text、FloatWritable。

Reduce(Key、Value)

{for(IntWritable val:value)

{

Sum1+=val.get();//key 为不同的条件字符串,则统计的是满足不同条件的网络数据行数

}

//计算 Key 字符串长度 length

//截取条件字符串 Key 最后 2 个字符 msg_temp

If(msg_temp.equals(“n”))//若属于正常网络

{Float result=sum1/sum_yes;//计算条件概率

If(Key.equals(“$w_1|n$”))

w1_in_nomal=result;//成员变量存储属性列 1 在阈值内且属于正常网络条件概率

Else if(Key.equals(“$\bar{w}_i|n$”))

w1_out_normal=result;//成员变量存储属性列 1 在阈值外且属于正常网络条件概率

//其他 5 个属性列计算在阈值内、外属于正常网络的条件概率同上

}

Else

{Float result=sum1/sum_no;//计算条件概率

If(Key.equals(“$w_1|b$”))

w1_in_unnormal=result;//成员变量存储属性列 1 在阈值内且属于僵尸网络条件概率

Else if(Key.equals(“$\bar{w}_i|b$”))

w1_out_unnormal=result;// 成员变量存储属性列 1 在阈值内且属于僵尸网络条件概率

// 其他 5 个属性列计算在阈值内、外属于僵尸网络的条件概率同上

}

Context.write(Key,result);//输出条件概率

}

由于 MapReduce2 要对训练数据的 6 个属性列

进行训练,每个属性既要判断是否为僵尸网络又要判断是否在阈值内,因此每个属性有 4 个判断条件。因此,经过 MapReduce2 的处理,形成 24 个条件概率分别存储在 24 个成员变量里,这与 MapReduce1 形成的 2 个成员变量存储的先验概率共同构成完整的知识库,可用于检测僵尸网络。

2.3 MapReduce3 设计

Map3 接收到的检测数据是被 Hadoop 处理形成的〈Key,Value〉对,形式为〈该行起始位置相对于文件起始位置的偏移量, 文本文件中的一行信息〉的信息。MapReduce3 要对 6 列属性全部检测,需用到 Value 的 6 个属性列。所以 Map3 将每行 Value 数据按空格分割成字符串数组,取出数组的第 3~8 项,分别为 TCP 数据流、时间间隔平均值、时间间隔变化、数据包字节数、数据包个数平均值、持续时间平均值。判断 6 个属性列的值是否在各自阈值内,若在阈值内,分别利用存储条件概率的成员变量 wi_in_nomal、wi_in_unnormal 计算后验概率;若在阈值外,分别利用存储条件概率的成员变量 wi_out_nomal、wi_out_unnormal 计算后验概率。并将每行网络数据的正常网络后验概率 $P(n \mid d)$ 和僵尸网络后验概率 $P(b \mid d)$ 一起输出。输出结果〈Key5,Value5〉对的形式为〈数据所在行数, $P(n \mid d)$ $P(b \mid d)$ 〉。Map3 过程伪代码如下。

```
输入:Object、Text。
输出:Text、Text。
map(Key,Value)
{StringTokenizer itr = new StringTokenizer(value.toString());
String[] temp = new String[9];
While(itr.hasMoreTokens())
  {temp[i] = itr.nextToken();//属性字符串数组
    i++;
  }
  P1 = sum_yes_p; P2 = sum_no_p;
If (Float.parsefloat(temp[2]) > 140&&Float.parsefloat(temp[2])<150)
  {
    P1 = P1 * w1_in_nomal;
    P2 = P2 * w1_in_unnormal;
  }
Else if(Float.parsefloat(temp[2]) <140 || Float.parsefloat(temp[2])>150)
  {
    P1 = P1 * w1_out_normal;
    P2 = P2 * w1_out_unnormal;
  }
//其他 5 个属性列检测过程同上。
  Line++;//统计所在行数
  Context.write(Line,P1 P2);
}
```

经过 Map3 把分块的每行信息都处理成以〈Key5,Value5〉形式的等待整体处理的中间文件输出,MapReduce 框架将每个 Map3 输出的中间文件结果按照 Key 值(数据所在行数)进行分组后发送给 Reduce3。

Reduce3 接收到的信息为<Key5,Value5>。Reduce5 的任务是逐行比较网络数据的 $P(b \mid d)$ 和 $P(n \mid d)$ 的大小。若 $P(n \mid d) > P(b \mid d)$,判断该行网络数据为正常网络数据;否则为僵尸网络数据。Reduce3 伪代码如下所示。

```
输入:Text、Text。
输出:Text、Text。
reduce(Key,Value)
{StringTokenizer itr = new StringTokenizer(value.toString());
  String[] temp = new String[2];
  While(itr.hasMoreTokens())
  {temp[i] = itr.nextToken();//正常网络后验概率与僵尸网络后验概率
    i++;
  }
If(Float.parseFloat(temp[0]) > Float.parseFloat(temp[1]))//比较
  Context.write(Key,"正常网络");//判断
Else
  Context.write(Key,"僵尸网络");//判断
}
```

3 实验结果与分析

本文实验中的被测网络环境为某校园网中一个子网的流量,该子网内主机约 200 台,白天的网络流量为 150~200 Mbps。实验采集了某天数据,为测试本文提出并行化的算法性能,分别使用了 2 个不同时间段的数据集 D_1 和 D_2。D_1 解析后的文本文件 1.6 GB, TCP 数据包个数 23 631 638。D_2 解析后的

文本文件 0.8 GB,TCP 数据包个数 11 570 835。

实验一通过选取检测不同的特征向量个数,分析贝叶斯分类器的正确率。分类器由训练 D_1 数据集获得,检测率通过分类器对 D_2 测试获取。具体实验结果如图 3 所示。

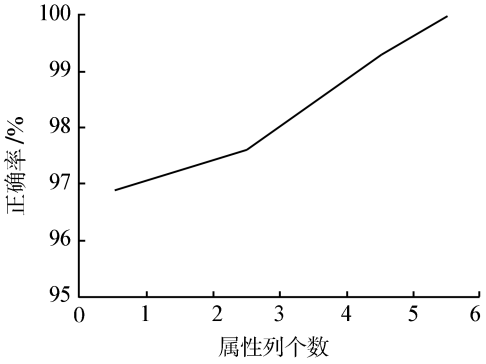

图 3 检测不同属性列个数的正确率比较

Fig.3 Comparison of correct rates among detecting different numbers of attribute columns

根据图 3 所示的实验结果,可以得到基于 MapReduce 的贝叶斯分类对于检测僵尸网络的正确率很高的结论,因此该方法是有效的。另外选取不同的属性列个数直接影响基于 MapReduce 的贝叶斯分类器的正确率,在一定程度上,检测属性列个数越多正确率越高。

实验二改变 TCP 数据流个数属性列的阈值,经过多次反复测试得到"正确率"分布如图 4 所示。

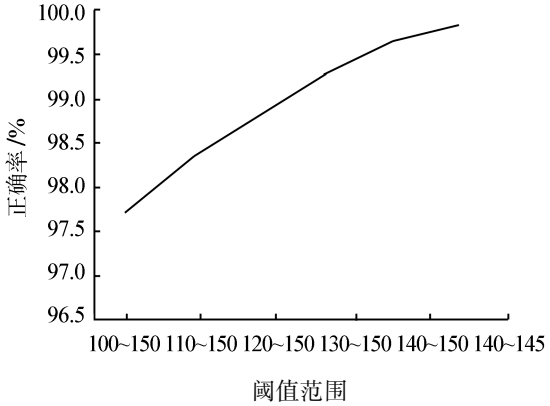

图 4 改变阈值对正确率影响

Fig.4 Influence of threshold changes on accuracy

从图 4 中可以看到,如果把 TCP 数据流个数选为需要检测的特征向量,那么它的阈值可以选择为 140~145。另外,如果选时间间隔值作为特征向量,同样可以测出它的阈值为 595~605;那么,可以一一通过这样的方法获得每个属性的阈值以及它取到阈值时的最高准确率。

实验三通过改变训练、测试数据集大小,比较数据集大小分别对普通串行贝叶斯分类检测僵尸网络和 MapReduce 并行化的贝叶斯分类检测僵尸网络效率的影响。具体实验结果如图 5 所示。实验 *A* 为普通串行贝叶斯检测僵尸网络、实验 *B* 为 MapReduce 并行化贝叶斯检测僵尸网络。从图 5 中看出,与普通串行贝叶斯检测僵尸网络相比,MapReduce 并行化贝叶斯检测僵尸网络效率较高,并且随着数据量增大,效率优势明显增强。

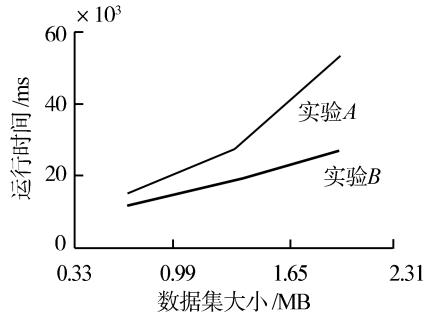

图 5 串行与 MapReduce 化贝叶斯算法检测僵尸网络效率比较

Fig.5 Efficiency Comparison of Serial and MapReduce Bayesian algorithms to detect botnets

4 结束语

本文提出了一种利用云环境下的 Hadoop 机制的 MapReduce 框架设计与实现贝叶斯分类的僵尸网络检测方法。与已有的僵尸网络检测方法不同的是:它以主机对作为分析对象,提取主机对通信的流量特征,然后将这些特征作为贝叶斯分类算法的输入,基于 MapReduce 训练生成贝叶斯分类器,用训练好的贝叶斯分类器进行僵尸网络的检测。这种检测方法有较高的检测率并且提高了检测效率。另外,本文在训练形成贝叶斯分类器阶段存在如何选择各特征值的阈值范围的问题,阈值范围的选取影响僵尸网络的检测率,下一步工作将对此另行研究。

参考文献:

[1] JIANG Hongli, SHAO Xiuli. Detecting P2P botnets by discovering flow dependency in C&C traffic[J]. Peer-to-Peer Networking and Applications, 2012, 5(2): 1-12.

[2] 王威,方滨兴,崔翔.基于终端行为特征的 IRC 僵尸网络检测[J].计算机学报, 2009, 32(10): 1980-1988.
WANG Wei, FANG Binxing, CUI Xiang. IRC botnet detection based on host behavior[J]. Chinese Journal of Computers, 2009, 32(10): 1980-1988.

[3] 蒋鸿玲,邵秀丽.基于神经网络的僵尸网络检测方法[J].

智能系统学报, 2013, 8(2): 113-118.
JIANG Honglin, SHAO Xiuli. Botnet detection algorithm based on neural network[J]. CAAI Transactions on Intelligent Systems, 2013, 8(2): 113-118.
[4]DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large cluster[J]. Communications of the ACM, 2005, 51(1): 107-113.
[5]陶永才,薛正元,石磊.基于 MapReduce 的贝叶斯垃圾邮件过滤机制[J].计算机应用, 2011, 31(9): 2412-2416.
TAO Yongcai, XUE Zhengyuan, SHI Lei. MapReduce-based Bayesian anti-spam filtering mechanism[J]. Journal of Computer Applications, 2011, 31(9): 2412-2416.
[6]杜跃进,崔翔.僵尸网络及其启发[J].中国数据通信, 2005, 7(5): 9-13.
DU Yuejin, CUI Xiang. Botnets and its enlightment[J]. China Data Communication, 2005, 7(5): 9-13.
[7]VALIANT L G. A bridging model for parallel computation [J]. Communications of the ACM, 1990, 33(8): 103-111.
[8]李晓桢,程佳,胡军.基于聚类分析的僵尸网络识别系统[J].计算机系统应用, 2009(8): 130-135.
LI Xiaozhen, CHENG Jia, HU Jun. Botnet recognition system based on the clustering technology[J]. Computer System and Application, 2009(8): 130-135.
[9]STONEBRAKER M, ABADI D J, DEWITT D J, et al. MapReduce and parallel DBMSs: friends or foes? [J]. Communication of the ACM, 2010, 53(1): 64-71.
[10]张鹏,唐世渭.朴素贝叶斯分类中的隐私保护方法研究[J].计算机学报, 2007, 30(8): 1267-1276.
ZHANG Peng, TANG Shiwei. Privacy preserving naive Bayesian classification[J]. Chinese Journal of Computers, 2007, 30(8): 1267-1276.

作者简介:

邵秀丽,女,1963 年生,教授,博士生导师,主要研究方向为云计算与软件工程等。参与或主持国家自然科学基金项目,国家“863”计划项目,天津市青年基金、自然科学基金、重点研究项目、CIMS 重点工程项目等多项科研项目。获得省部级科技进步奖、国家档案局二等奖等 8 项,发表学术论文 80 余篇。

刘一伟,女,1992 年生,本科生,主要研究方向为应用数学,发表学术论文 4 篇。

耿梅洁,女,1988 年生,硕士研究生,主要研究方向为云计算。

第 31 届机器学习国际会议
31st International Conference on Machine Learning

The 31st International Conference on Machine Learning (ICML 2014) will be held in Beijing, China, from June 21 to 26, 2014. The conference will, tentatively, consist of one day of tutorials, followed by three days of main conference sessions, followed by two days of workshops. We invite submissions of papers on all topics related to machine learning for the conference proceedings, and proposals for tutorials and workshops.

After reviewing author and reviewer feedback from the previous conference, ICML 2014 will adopt a two-cycle submission/review format, of which the first submission/review cycle will facilitate both regular one-time review/rebuttal of submissions, as well as invitation-only resubmission into the second cycle, whereas the second cycle will only allow regular first-time submission plus resubmission of papers invited from the first cycle. We are also exploring the possibility of a JMLR track at ICML that allows direct submission of papers intended for JMLR to be reviewed under the same time frame of ICML, more detail will be available soon once agreement with JMLR has been reached. Accepted papers will be announced and posted online shortly after acceptance and will be considered published and available for citation at that time.

Webstie: http://icml.cc/2014/