Journal of Computer Applications
计算机应用,2017,37(5):1263 - 1269
ISSN 1001-9081
CODEN JYIIDU
2017-05-10
http://www.joca.cn

文章编号:1001-9081(2017)05-1263-07
DOI:10.11772/j.issn.1001-9081.2017.05.1263

基于 Spark Streaming 的实时数据分析系统及其应用

韩德志[1*],陈旭光[1],雷雨馨[2],戴永涛[1],张　肖[1]
(1.上海海事大学 信息工程学院,上海 201306;　2.郑州大学 信息工程学院,郑州 450001)
(*通信作者电子邮箱 dezhihan88@sina.com)

摘　要:为了实现对实时网络数据流的快速分析,设计一种分布式实时数据流分析系统(DRDAS),能有效解决并发访问数据流的收集、存储和实时分析问题,为大数据环境的网络安全检测提供了一种有效的数据分析平台;根据 Spark Streaming 运行的原理设计一种动态采样的 *K*-Means 并行算法,与 DRDAS 结合能实时有效地检测大数据环境下的各种分布式拒绝服务(DDoS)攻击。实验结果显示:DRDAS 具有好的可扩展性、容错性和实时处理能力,与动态采样的 *K*-Means 并行算法结合能实时地检测各种 DDoS 攻击,缩短了攻击的检测时间。

关键词:Spark Streaming 框架;分布式流处理;网络数据分析;分布式拒绝服务攻击
中图分类号:TP316.2　　**文献标志码**:A

Real-time data analysis system based on Spark Streaming and its application

HAN Dezhi[1*], CHEN Xuguang[1], LEI Yuxin[2], DAI Yongtao[1], ZHANG Xiao[1]
(1. *College of Information Engineering, Shanghai Maritime University, Shanghai* 201306, *China*;
2. *School of Information Engineering, Zhengzhou University, Zhengzhou Henan* 450001, *China*)

Abstract: In order to realize the rapid analysis of massive real-time data, a Distributed Real-time Data Analysis System (DRDAS) was designed, which resolved the collection, storage and real-time analysis for mass concurrent data. And according to the operation principle of Spark Streaming, a dynamic sampling *K*-means parallel algorithm was proposed, which could quickly and efficiently detect all kinds of DDoS (Distributed Denial of Service) attacks. The experimental results show that the DRDAS has good scalability, fault tolerance and real-time processing ability, and along with new *K*-means parallel algorithm, the DRDAS can real-time detect various DDoS attacks, and shorten the detecting time of attacks.

Key words: Spark Streaming framework; distributed stream processing; network data analysis; Distributed Denial of Service (DDoS) attack

0　引言

随着互联网的高速发展,普通用户带宽普遍升级,尤其是在一些大城市,家庭用户的带宽都已经达到 20 MB/s,甚至更高。此外,随着 3G 网络的普及,以及 4G 网络的逐步推广,移动互联网也进入了一个蓬勃发展的时期。个人网络带宽的快速增长和不断增加的网络用户,对企业网络安全带来了巨大的挑战。因为,这些高带宽的网络用户一旦被黑客控制并参与到分布式拒绝服务(Distributed Denial of Service, DDoS)攻击中时,其影响将无法估量。

全球著名内容分发网络(Content Delivery Network, CDN)服务商 Incapsula,在 2014 年发布的 DDoS 攻击趋势报告[1]中指出,2014 年 DDoS 攻击行为增加了 240%,且流量已经超过了 100 GB。此外,该公司的最近一份分析报告[2]指出,目前或有数十万甚至上百万台的小型或家庭办公环境(Small Office Home Office, SOHO)专用路由器已成为僵尸路由器,被黑客用来执行大规模 DDoS 攻击。该公司 2014 年的一项关于 DDoS 造成损失的调查发现:49% 的 DDoS 攻击会持续 6 ~ 24 h,每小时的平均经济损失为 40 000 美元[3]。

此外,近两年来,网络安全事件更是频繁发生:2014 年 12 月 20 日—21 日,一家将服务器部署在阿里云上的游戏公司遭遇了 DDoS 攻击,453.8 GB/s 的峰值流量使之成为全球最大的 DDoS 攻击受害者;而知名的代码托管网站 GitHub,于 2015 年 3 月 26 日开始遭遇其网站最大规模的 DDoS 攻击,导致部分地区服务中断,且攻击持续了 80 多小时;2015 年 5 月 11 日,网易遭受了最新型的 DDoS 攻击——链路洪泛攻击(Link Flooding Attack, LFA)[3],以至服务中断 9 h,损失超过 1 500万元。

对超大流量的 DDoS 攻击,研究如何高效、及时地检测并报警有重要的理论意义和重大的经济价值。本文将新型的分布式流式计算框架——Spark Streaming[4],应用于实时的大流量网络数据实时分析,为提高大数据环境的 DDoS 检测速度提供保证。

1　相关工作

1.1　大数据流特点

在大数据时代,流式数据已经在很多环境中加以应用,并

收稿日期:2016-07-15;**修回日期**:2016-11-26。　**基金项目**:国家自然科学基金资助项目(61373028,61672338)。
作者简介:韩德志(1966—),男,河南信阳人,教授,博士,CCF 会员,主要研究方向:云计算、云存储及其安全、大数据应用;　陈旭光(1993—),男,河南信阳人,硕士研究生,主要研究方向:云计算、大数据实时分析;　雷雨馨(1996—),女,河南郑州人,主要研究方向:数据挖掘、网络安全;　戴永涛(1991—),男,湖南邵阳人,硕士研究生,主要研究方向:云计算、分布式计算、数据挖掘、网络安全;　张肖(1994—),女,安徽蚌埠人,硕士研究生,主要研究方向:云计算、大数据实时分析。

取得很好的效果。本文在文献[5]的基础上,总结出了大数据流具有以下 6 个特点:

1)并发性。体现在两个方面:服务并发和数据并发。在当前的大部分系统中,流处理都是在一些高并发的应用中,所提供的都是高并发服务,而并发服务必然会并发产生大量数据,这就要求系统具有很好的并发收集、处理和分析的能力。

2)实时性。数据实时产生,也需要实时计算,保证数据的时效性,让数据更快、更好地体现其价值,因为很多场景,比如实时推荐系统、实时监控系统,数据是具有实效性的,若不能在短时间内加以处理和分析,就会失去数据的价值。

3)易失性。原始数据流一般被实时处理和分析后,就会被丢弃,只有少量日志被持久保存,或者只保存处理后的结果,想要再次分析原始日志可能性不大。

4)突发性。因为数据是实时产生的,只受当前服务的影响,跟前一时刻的数据量大小无关,也就是说前后两个时间的数据量可能会相差很大。

5)无序性。在大数据环境下,不同数据源之间的产生和传输并不能保证按照其产生的相对顺序,因此在分析时需要更加完善的处理逻辑,而不是只根据其到达的先后顺序。

6)无限性。只要服务存在,数据就一直会持续、动态地增加,所以数据的大小理论上是无限的,不能用具体的数字来量化。当然,实际使用中不可能用磁盘无限地存储,应定期地对历史日志进行清理或压缩备份。

1.2 大数据流相关研究

由于单台计算机的存储、计算能力的限制,已经不能满足大数据时代海量数据的处理和分析的要求。同样地,在网络安全检测中,单台服务器已经不能及时处理和分析短时间内产生的大量网络数据,这给安全检测系统带来了巨大的挑战。目前,已经有一些学者开始研究用分布式计算框架处理海量的网络数据。

文献[6]使用 Hadoop 对海量数据流的日志进行分析,该方法能提供良好的容错性,且能大幅缩短计算的时间,但是需要先将所有的日志收集起来,再导入到 Hadoop 集群中,这样浪费了大量的时间,不能对大数据流进行实时分析。

文献[7]中将 Hadoop 应用到海量网络数据流攻击的日志分析中,虽然能处理和分析大量的网络监控数据的日志信息,但是由于其采用离线分析方式,该方法只适用于研究网络攻击的一些行为和特征,不能真正用于网络检测系统中。

文献[8]在文献[7]的方法上作了一些改进,将原来的分析文本日志改成了直接分析抓取到的二进制数据,并实现了 P3(Parallel Packet Processor)系统,在一定程度上减轻了数据抓取服务器的负担,但还是采用离线分析处理,缺乏实用性。

文献[9]用 Hadoop 进行 DDoS 攻击检测,实现 HTTP GET flooding 攻击检测算法,由于使用的是 Hadoop 系统,仍然不能解决响应时间慢的问题。

文献[6-9]虽将分布式计算方法用于网络检测和数据流分析中,但是都不能很好地解决时效性的问题。针对以上情况,本文提出了一种基于 Kafka[10] 和 Spark Streaming[4] 的实时流处理和分析方法,对海量的网络数据进行实时分析,适合于大流量的实时网络分析和异常检测。

海量的数据是大数据出现的前提,而数据收集则是大数据的基石。日志数据收集在流数据收集中占有重要比重,许多公司的业务平台每天都会分散地产生大量日志数据,收集并汇总这些业务日志数据,供离线和在线的分析系统使用。日志收集系统所需考虑的基本特征包括:高可靠性、高可用性和高可扩展性。“分散收集,集中处理”是当前日志处理系统的一个主流思想。日志收集也是流式日志处理系统的前提和基础,日志只有被实时收集、汇总后,才能进行后续的相关处理操作。下面针对当前流行的开源日志数据流收集系统,进行介绍和对比。

2 数据流实时数据分析系统总体设计

2.1 数据流实时数据分析系统整体架构

实时网络数据分析系统(Distributed Real-time Data Analysis System, DRDAS)的整体架构如图 1 所示,其组成部分包括数据收集系统、数据分析系统和数据存储系统。在图 1 中,Kafka Broker 是数据收集系统,负责大数据流的实时搜集和传递;Spark Streaming 是数据分析系统,负责对 Kafka Broker 传递的数据流进行实时分析,并将分析结果传递应用系统进行各种处理,同时将正常的数据传递存储系统;Database 表示结构化的数据存储系统,Hbase 表示非结构化的数据存储系统。

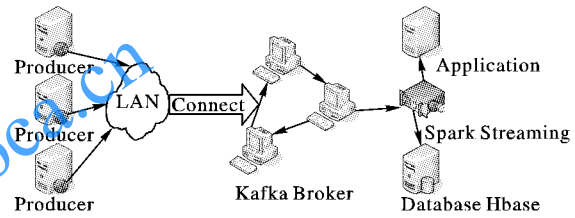

图 1 系统整体架构

Fig. 1 Overall architecture of DRDAS

2.2 数据流实时数据收集系统

大数据流具有如本文 1.1 节所描述的 6 大特性,在大数据流并发产生的情况下,一方面要保证数据的实时性,另一方面尽量减少数数据丢失,这些都给数据的收集带来了巨大挑战。

在一些广告、搜索、推荐、安全系统中,为了满足海量日志或事件的实时传输,及时分析、处理,并实时反馈的需求,很多人都在为解决这一难题而研究。LinkedIn 公司开发了一套专用的分布式消息订阅和发布系统——Kafka[10]。早在 2012 年 LinkedIn 公司就已经将 Kafka 应用在实际生产系统,并每天收集和处理上百亿的消息,峰值时能处理每秒高达 172 000 条消息[8]。LinkedIn 将 Kafka 于 2011 年开源,并成为 Apache 的开源项目之一。

由于 Kafka 是一套分布式系统,其吞吐量可以随着集群的扩展而线性增加。

图 2 为 Kafka 的整体架构。图中给出了 Kafka 的三大构成部分:生产者(Producer),即日志的来源;代理(Broker),消息的中间管理者;消费者(Consumer),消息的使用者。Producer 负责将消息收集并推送(Push)到 Broker,而 Broker 则负责接收 Producer 发送来的消息,并将消息本地持久化,Consumer 则是消息的真正使用者,从 Broker 拉取(Pull)消息并进行处理。

整个 Kafka 架构的核心就是 Broker,Broker 的吞吐量直接关系到整个系统的可用性。而为了加强其处理能力,设计者

从两方面作了考虑:1)巧妙的消息管理策略;2)线性化的扩展方式。

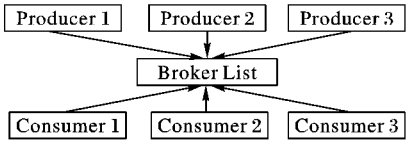

图 2　Kafka 架构

Fig. 2　Kafka architecture

图 3 展示了 Kafka 的消息管理策略,通过 Topic 名称将消息分开管理,一个 Topic 由多个 Partition 组成,而每个 Partition 又是由多个 Segment 文件构成,每个 Segment 存储指定数量的消息,存满后会存放到一个新的 Segment 文件中。消息在同一个 Partition 中是有序的,而且能保证消息读取时是有序(按照 Partition 中的存储顺序)。

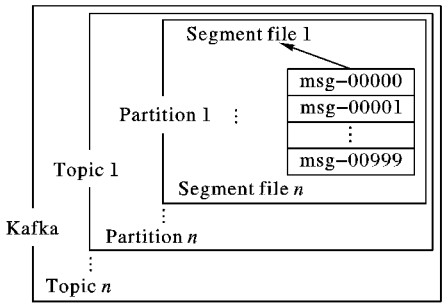

图 3　Kafka 消息管理策略

Fig. 3　Kafka message management strategy

图 4 为 Kafka 对同一个 Topic 的消息读写管理:消息流入时,根据该消息所在 Topic 的分区数和复制数,将不同的消息分别存储到不同的 Partition 上,同时,也将同一条消息复制多份,保存在不同的主机上,达到数据冗余备份和提高读取速度的目的;Consumer 读取消息时,对于属于同一个 Group 的 Consumer,在同一个 Partition 上,不能并发读取,但是在不同的 Partition 上可以并发操作,也就是说同一个 Topic 的 Partition 数量越多,可允许的并发数也就越多。

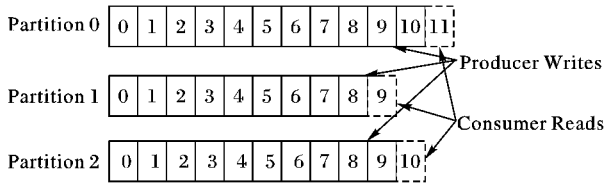

图 4　Kafka 对 Partitions 的读写操作

Fig. 4　Kafka' read and write operations in the Partitions

2.3　数据流实时数据分析系统

Spark[11] 于 2009 年诞生于加州伯克利大学的 APMLab 实验室,2010 年正式开源,并于 2013 年成为 Apache 基金项目,在 2014 年成为 Apache 基金的顶级项目。为了解决 Hadoop[12] 计算时将中间结果存入磁盘导致计算速度缓慢的问题,Spark 应运而生。

Spark 的生态系统如图 5 所示,该生态系统包括了批处理、流处理、机器学习、图计算、数据分析等,相比 Hadoop 生态圈,显得更加全面,是一个更适合大数据应用场景的分布式计算框架。

弹性分布式数据集(Resilient Distribute Datasets,RDD)[13] 是 Spark 的核心,也是 Spark 实现故障恢复、数据依赖的关键。RDD 模型以简单的逻辑——Lineage 很好地解决了数据之间的依赖,且保证很好的容错性,并能将中间结果存入内存,尽量减少数据的磁盘读写,使得计算速度得到了很大的提升,尤其是在迭代式计算中,计算速度提升了一个数量级。

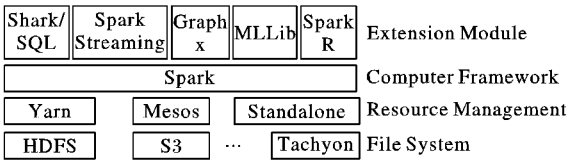

图 5　Spark 生态圈

Fig. 5　Spark ecosystem

不同于 Hadoop 中的 MapReduce,Spark 将相应的 MapReduce 很好地封装在 RDD 中。在 RDD 上可执行两类操作:转换(Transformation)和动作(Action)。图 6 展示了数据从输入到得到最终结果的操作过程。数据在 RDD 中并不是以原始形态存在,而是以数据所在的具体位置的形式包含在 RDD 中,并在 RDD 中经过不同的转换得到新的 RDD,直到执行动作时才执行真正的计算,得到最终想要的结果。

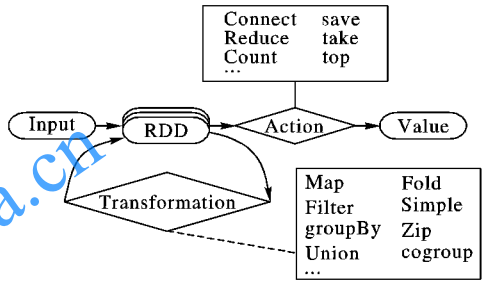

图 6　RDD 上的相关操作

Fig. 6　Related operations in RDD

Spark Streaming[4] 是 Spark 生态系统中用于实时计算的一个框架,其核心也是基于 RDD 的,它可以无缝地和 Spark 衔接,将历史数据和实时数据完美地融合。Spark Streaming 与 Yahoo 推出的 S4[14] 以及 Twitter 推出的 Storm[15] 并列为三大主要数据流处理系统。大数据流处理系统的典型实例见文献[16],Spark Streaming、S4 和 Storm 等分布式流处理系统的对比见文献[17]。

图 7 展示了 Spark Streaming 流处理的实质:将一小段时间内的数据合并,再作微型的批处理,而不是分别对每一条数据进行实时处理。这也是它与其他流处理系统的最大区别,因此它并不是真正意义上的实时处理,而是延时较低的微批处理,正好适合本文中的应用场景。此外,Spark Streaming 还能通过滑动窗口将最近一段时间的数据进行合并,或直接与 Spark 结合将历史数据和实时数据一起分析,为实时数据和历史分析提供了一个统一的处理平台。

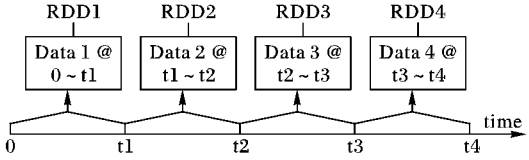

图 7　Spark Streaming 的处理过程

Fig. 7　Processing procedures of Spark Streaming

2.4　日志数据流收集系统对比

日志数据流收集系统具备 3 个基本组件,分别是 Agent(接收原始数据,并将数据发给 Collector)、Collector(接收多个

Agent 发送过来的数据,汇总后将数据发往 Store)和 Store(中央存储系统,将汇总后的数据进行持久化存储)。表 1 综合对比了 Scribe、Flume[18]、Chukwa、LogStash 四种日志数据流收集系统。

表 1 4 种日志收集系统对比

Tab. 1 Comparison of four log collection systems

收集系统	实现语言	发送机制	容错性	负载均衡	可扩展性	社区活跃度
Scribe	C/C++	Push/Push	一般	无	好	低
Flume	Java	Push/Push	好	有	好	高
Chukwa	Java	Push/Push	好	无	好	低
LogStash	JRuby	Push/Push	一般	无	好	高

3 数据流实时数据分析系统性能分析

3.1 数据流分析系统整体数据流图

图 8 为数据流分析系统整体数据流图,整个系统包括:数据收集、数据分析、数据存储、模型(或算法)训练、入侵检测。系统的数据流向为:1) 数据的来源为不同的服务器,通过各种抓包软件,如:TcpDump、NetFlow、Sniff 等,对特定的网卡或端口进行数据包抓取,并通过 Flume 将不同服务器上的网络数据汇总,将数据抓、分析和检测分离,减轻应用服务器的负担。Flume 最初是由 Cloudera 的工程师设计用于合并日志数据的系统,后将其开源出来,并逐渐发展成为一款开源、高可靠、高扩展、易管理、支持客户扩展的分布式数据采集系统,主要是用于日志数据的收集和聚合。2)数据汇聚之后,将所抓取的网络数据作为 Kafka Producer 的消息源,并传送到 Kafka Broker,让 Broker 对所有网络数据进行有序的管理。3)Spark Streaming 则实时从 Kafka Broker 中拉取数据,再将数据分散到不同的 Spark Executor 进行分析和统计。4) Spark 将抓取的网络数据处理后,一方面可以将结果传给其他的应用,作进一步分析;另一方面可以将结果持久化,存储在数据库中,供后续分析使用。5)对于得到的实时数据,可以使之与之前得到的历史数据进行合并,进而进行模型(或算法)训练或者直接通过模型进行 DDoS 检测,并得到检测结果。

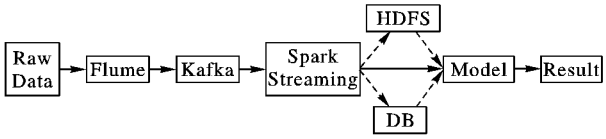

图 8 数据流分析系统整体数据流图

Fig. 8 Data flow of data analysis system

3.2 实时性分析

要做到实时数据分析,就要求系统具有实时的数据收集、处理功能。本系统中,使用了 Flume 和 Kafka 来保证数据能实时地被收集并传到 Spark Streaming 进行实时分析。

Flume 能监控某一指定的目录下的文件或整合自定义的数据收集器,能将新增的数据实时地传输到数据中心服务器,并作为 Kafka 消息队列的消息来源。

Kafka 则能将接收到的实时数据转存为有序的消息队列,以有序的方式分别存储在不同服务器的 Partition 上,读取消息时,通过并发方式从不同 Partition 读取数据并使用 ZeroCopy 技术保证消息快速有序的读取,后面会有相关的实验进行测试和说明。

Spark Streaming 能将接收的数据汇总成一个个小的数据集,并复制多份存储,然后进行微批量的实时处理,其这一特性很好地避免了并发数据处理中频繁的任务分配和调度问题,能达到次秒级延时的实时处理。

3.3 可扩展性分析

为了适应数据量的不断增长,需要系统具有很好的可扩展性。本系统的扩展性体现在两个方面:

1)数据收集可扩展。随着数据总量增长,已有资源不一定能满足数据收集的要求,这就需要数据收集系统具有很好的扩展性。而本系统中的 Flume 和 Kafka 都能通过扩展集群,让处理能力得到近乎线性的增长,很好地满足可扩展性要求。

2)数据分析可扩展。Spark Streaming 是一个分布式的实时处理系统,其数据处理能力也是随着集群数量的扩展而递增,实验部分会有相关的数据证明。目前,已经有多个超过上千台 Spark 服务器组成的集群应用在生产环境当中,证明了其在数据分析方面的可扩展性。

3.4 容错性分析

为了保证系统可靠和稳定的运行,容错性也是绝大多数系统需要考虑的一个关键性因素。在本系统中的容错性体现在:

1)数据收集的可靠性。Flume 能将数据缓存在本地磁盘,并记录已传输数据的偏移量,若由于网络或其他原因导致的数据传输失败,会在故障恢复时,继续之前的数据传输;而 Kafka 则是通过复制因子来控制数据保存的份数,通过将数据复制到不同的机器上,解决了数据丢失或系统单点故障问题。

2)数据处理的可靠性。Spark Streaming 有一种很好的容错机制—— 提前日志写(Write Ahead Logs, WAL)机制,即:先将数据写入日志,再计算。通过将实时收集的数据写入到日志中,并复制多份,能解决计算过程中单点故障造成的数据丢失,便于故障恢复和转移。

4 数据分析系统相关的实验分析

实验环境说明:本实验使用 vSphere 虚拟机搭建 Linux 集群,每个 Linux 虚拟机的配置为 2×2 核 CPU,内存 4 GB,内网网速 1 000 Mb/s,Kafka 版本 0.8.21,Flume 版本 1.6.0,Spark 版本 1.4.1。

为了测试 Kafka 接收和读取消息速度,本实验使用了 Kafka 自带的压测工具,并通过其脚本 kafka-consumer-perf-test.sh 和 kafka-producer-perf-test.sh 进行测试,配置项都使用默认值。图 9 和图 10 是在不同数量 Kafka Server 集群下,Kafka 消息接收和读取速度的测试。其中图 9 是 Kafka 接收和读取消息每秒的字节数测试结果,图 10 是 Kafka 每秒钟接收和读取的消息数量测试结果。虽然,测试结果的好坏由很多因素决定:磁盘读写速度、CPU、内存、网络、配置参数等,但是从图 9 和图 10 可以明显看出,Kafka 消息处理的能力随着集群数量的增加呈近似线性增长关系,证实了 Kafka 具有线性扩展能力。

除了数据接收和读取能力测试外,本文还使用 spark-perf[19] 和 visualvm[20] 等对 Spark Streaming 数据处理能力进行压力测试和监控。图 11 是对应的测试数据结果,从图中可以看出 Spark Streaming 处理数据的能力也跟集群数量成正比关

系。当然,数据处理速度的具体数值会受到处理复杂度的影响,但是总体趋势还是类似的,即通过扩展 Spark 集群数量,可以提高集群数据的处理速度和数据量。

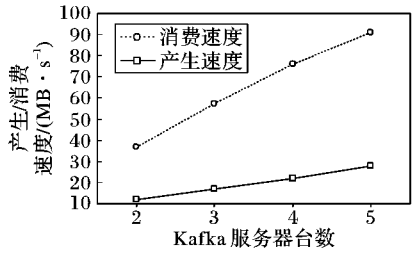

图 9　Kafka 服务器数量与消息吞吐量的关系

Fig. 9　Relationship between Kafka server number and message throughput

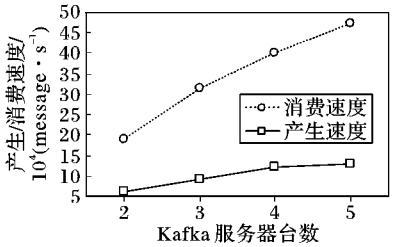

图 10　Kafka 服务器数量与消息吞吐率的关系

Fig. 10　Relationship between Kafka server number and message throughput rate

图 9 ~ 11 对由 Kafka、Flume 和 Spark Streaming 组成的数据分析系统,在数据吞吐量和数据处理速度方面进行测试,测试结果可知:整个数据分析系统的数据吞吐量随 Kafka 集群数成线性增长,数据实时分析处理能力随 Spark Streaming 集群成线性增长。

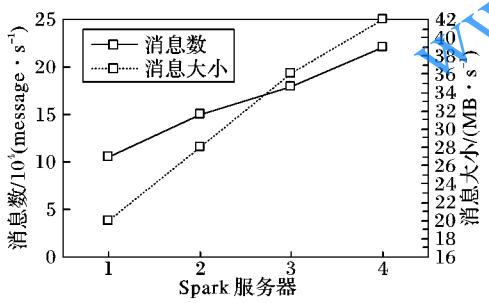

图 11　Spark Streaming 处理速度测试

Fig. 11　Processing speed test for Spark Steaming

表 2 对 3 种不同的数据分析场景和特点进行了对比:单机环境下不能处理大量的数据,没有容错性,不能进行扩展,其数据处理能力是有限制的;Hadoop 虽然可以线性扩展,能处理大量的数据,但其设计初衷就是为了离线处理海量数据。因此,Hadoop 不适合于实时处理场合;Spark Streaming 在各方面有明显的优势。

表 2　单机、Hadoop、Spark Streaming 对比

Tab. 2　Comparison of single-machine, Hadoop, Spark Streaming

应用环境	数据量	容错性	实时性	可扩展性
单机	小	无	好	无
Hadoop	大	好	差	好
Spark Streaming	大	好	较好	好

5　实时数据分析系统的应用

K-Means[21] 是一种典型的局域原型的目标函数聚类算法,属于无监督学习范畴,已成功应用于网络环境的 DDoS 检测系统中。为了将经典的 *K*-Means 算法应用于大数据环境下的实时数据分析系统中,本文根据 Spark 运行的原理设计动态采样的改进 *K*-Means 并行化算法[22],并将该算法成功应用于基于 Spark Streaming 实时数据分析系统中。

5.1　动态采样 *K*-Means 算法

在普通的聚类算法[22] 中,随机均匀选择 k 个点作为聚类中心,k 的取值以及初始中心的选择将直接影响最终聚类的效果。如果 k 选取不当,*K*-Means 算法在计算过程中易收敛于局部最优解,从而无法得到正确结果。在 DDoS 检测系统中,*K*-Means 聚类需要处理大量混杂攻击流的数据,这对初始中心的选择造成了极大的困难。为了解决这一问题,采用一种动态采样的 *K*-means 聚类改进算法,以满足 DDoS 检测系统的需求,算法流程如图 12。

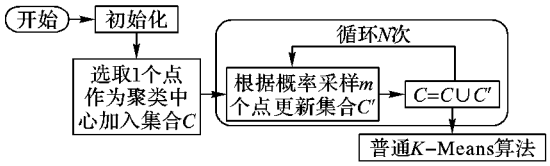

图 12　动态采样改进的 *K*-Means 算法流程

Fig. 12　Flow of improved *K*-Means algorithm by dynamic sampling

改进 *K*-Means 算法的主要思想是预先只选择一个点作为聚类中心,构建一个规模函数 $V(X)$,该函数表示数据点到其所属的聚类中心的距离平方和,通过不断迭代最小化规模函数值使聚类结果收敛。改进 *K*-Means 算法的主要原理:首先从数据集合 X 中选择一个点作为始聚类中心并加入动态采样集合 C,根据规模函数 $V(X)$ 计算得到初始规模 N,然后进行 N 次的循环;在每次循环内再选取 m 个点,每次循环计算采样概率 $P(X)$,采样概率的意义是:聚类中心是相对分散的点,离本聚类中心越远,其成为另外一个聚类中心的可能性越高,即是尽可能地选择远离当前聚类中心的点作为采样数据。每经过一次迭代,重新计算规模函数 $V(X)$ 的值,更新下一次采样的概率。然后将本次采样的簇中心点集合 C' 与原采样集合 C 求并集作为新的采样集合 C。当 N 次循环结束后,产生一个新的采样集合 C,该集合中一共有 $N*m$ 个数据。此时得到的数据集 C 的规模远小于初始数据集 X,且数据是经过筛选的相对集中。最后再对 C 运行普通的 *K*-Means 算法,因为 C 是经过预处理得到的集合,因此整个聚类算法将会异常迅速。本算法在时间复杂度上也有改进,采用了 N 次迭代替代定义收敛阈值的方法,降低了运算过程中执行迭代的次数,对于通过机器学习方法检测大数据环境下的 DDoS 攻击具有重要意义。

具体的算法定义如下:

定义 1　规模函数 $V(X)$ 定义如式(1) 所示:

$$V(X)=\sqrt{\sum_{i=1}^{n}D^2(X,C)}=\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{d}(x_j-x_c)} \tag{1}$$

其中 $D^2(X,C)$ 表示 X 中的点到所处聚类中心的距离平方和。

定义 2　动态采样概率函数 $P(X)$ 定义如式(2) 所示:

$$P(X)=\frac{D^2_{\min}(X,C)}{V(X)} \tag{2}$$

定义 3　初始限定规模函数值 V,初始采样个数 $m(m<k)$。

具体算法实现如下:

输入　数据集，K；

输出　K 个聚类中心。

步骤 1　从集合 X 中随机取 1 个点加入集合 C；

步骤 2　根据式(1)计算 C 的初始限定规模函数值，记为 V；

步骤 3　循环 $\lg V = N$ 次开始，按式(2)计算动态采样概率记为 P，从集合 X 中按概率 P 取出 m 个点加入集合 C'，求 $C \cup C'$，记为 C，循环结束；

步骤 4　采用普通 K-Means 算法求出集合 C 的聚类中心。

5.2　基于 Spark 动态采样改进 K-Means 算法并行化

普通的单机 DDoS 检测算法无法直接在 Spark 平台上运行，根据 Spark 运行的原理设计动态采样的改进 K-Means 算法的并行化。具体的流程如下：

1)算法开始，Master 节点上程序从数据输入源获取初始数据集。这里的输入源是预先定义的接口，可以通过多种途径获取数据，如 InputStream、Hadoop 分布式文件系统（Hadoop Distributed File System，HDFS）、本地文件等，这一设计方便对本算法的测试。获取数据后，系统将其转换为 RDD1，并调用 cache 方法将 RDD1 加载至内存，该 RDD 作为待处理的数据。

2)开始进行数据的分割，为并行化作准备。系统以块为单位(64 MB)将 RDD1 划分成多个分块。接下来 Master 节点调用 map 方法，把众多的数据块分配到多个 Worker 节点上。Worker 节点接收数据块，并执行 Master 的 map 指令，对数据块进行处理。经过这一步骤，原数据集中的 String 文本被转换为 DenseVector 向量对象，这些对象是程序能够直接使用的数据，并且分布在各个 Worker 节点上等待计算。map 方法结束后，RDD1 生成了新的 RDD2。

3)随机选取初始的聚类中心。程序调用 takeSample 方法，从 RDD2 中选取一个作为聚类中心向量，并创建 RDD3 对象。

4)开始进入循环过程，程序根据 5.1 节中具体算法实现中的步骤 3 进行迭代计算。在每次循环体中根据定义 1 和定义 2 重新计算当前采样概率函数 P，接着调用 takeSample 方法根据概率 P 选取新的 RDD 向量作为中心点。经过一次循环，一共采样了 $1 + m$ 个向量，生成 RDD4。紧接着系统调用 union 方法，把 RDD3 和 RDD4 合并成 RDD5。

5)经过 $\lg V$ 次后，循环结束，此时 RDD5 中的向量个数 $\leqslant 1 + m * \lg V$。这一数据量远远小于初始的数据量。

6)系统将 RDD5 作为结果输出。

上述整个采样阶段 RDD 转换流程如图 13 所示。图中圆角矩形框代表 RDD；RDD 内直角矩形框表示该 RDD 中的数据分片，这些分片散布在不同的 Worker 节点上；箭头指向的方向表示 RDD 转换的过程。

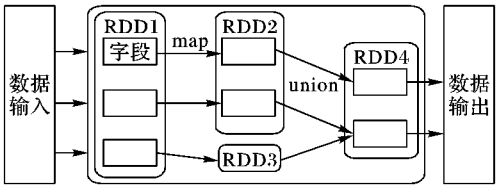

图 13　采样阶段 RDD 转换流程

Fig. 13　RDD conversion process in sampling phase

5.3　基于实时数据分析系统的 DDoS 攻击检测

为了测试动态采样的改进 K-Means 并行化算法在基于 Spark 的数据分析系统上的检测速度和准确率，设计了如下实验：采用 KDD99[23] 数据集的训练全集(500 万条数据)作为实验样本，分别从中抽取 5 组作为实验数据。这五组的数据量分别为：1 万、50 万、100 万、200 万、500 万。

实验设计了 3 个实验：第 1 个实验，采用单机算法串行处理数据样本；第 2 个实验采用在 Spark 集群上实现的普通 K-Means 算法，并行处理数据样本；第 3 个实验采用在 Spark 的数据分析系统实现的动态采样的改进 K-Means 算法。

分别对 3 个实验的时间消耗、每轮迭代平均时间、正确率进行统计分析，最终的结果如图 14～16 所示。

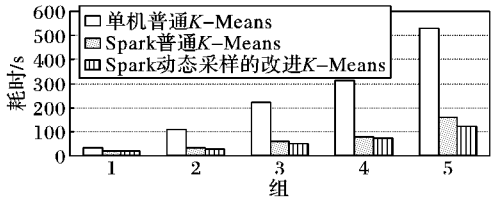

图 14　三种算法在 5 组实验的耗时对比

Fig. 14　Comparison of time-consumption of three algorithms in 5 experiments

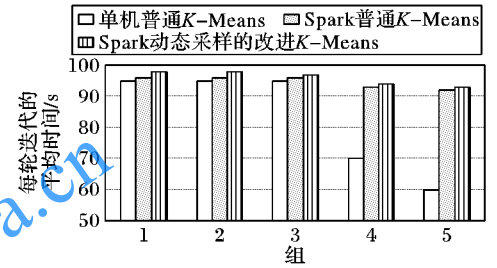

图 15　三种算法在 5 组实验的每轮迭代平均时间对比

Fig. 15　Comparison of average time for each iteration of three algorithms in 5 experiments

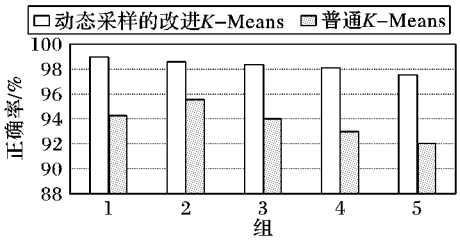

图 16　两种算法的正确率对比

Fig. 16　Comparison of accuracy rates of two algorithms

通过图 14 和图 15 对比发现，在数据量较小的情况下，三种算法的耗时差距不大，单机运行的算法正确率也相对较高，Spark 的数据分析系统并行计算的优势不明显；当数据量大于 100 万时，单机运行的时间急剧增长，准确率迅速下降，此时 Spark 的数据分析系统并行计算的优势相当显著。对比在 Spark 的数据分析系统实现的 K-Means 和动态采样的改进的 K-Means 算法，改进的算法在正确率上保持相对稳定，而算法的效率则比普通方法更高。从图 16 可以进一步看清动态采样改进的 K-Means 算法和普通 K-Means 算法准确率的差别，本实验能较好地体现基于 Spark 的实时数据分析系统和动态采样的改进 K-Means 算法相结合的优势。

6　结语

本文在已有研究基础上，进一步地将网络数据分析与分布式计算相结合，克服已有数据分析方案的时延大、缺乏时效性、不能满足大数据实时处理和分析的缺陷。通过将网络数

据抓取、收集并导入 Kafka,产生实时的消息队列,再使用 Spark Streaming 进行网络数据实时分析:一方面可以实时地对网络数据进行分析,为大数据环境提供快速高效的异常检测;另一方面,满足了大数据环境下数据处理速度和数据处理量的要求。

通过验证分布式实时计算框架 Spark Streaming 在大规模网络环境中对海量网络数据实时分析的可行性;另一方面,确定了大规模实时网络检测的整体架构,为进一步研究大规模网络数据实时分析和检测理论和实现技术奠定基础;最后,为了验证实时数据分析系统在大数据环境下的 DDoS 攻击检测能力,本文根据 Spark 运行的原理设计了动态采样改进 *K*-Means 并行化算法,并将该算法应用于基于 Spark Streaming 实时数据分析系统中,实验结果显示基于 Spark Streaming 的实时分析系统用于大数据环境下的 DDoS 检测是可行的。下一步,将结合已有的隐半马尔可夫模型异常检测算法,进一步设计能实时有效检测大数据环境的各种 DDoS 攻击行为的算法。

参考文献(References)

[1] Incapsula. Report: 2014 DDoS Trends-Botnet activity is up by 240%[EB/OL]. [2014-06-20]. https://www.incapsula.com/blog/ddos-threat-landscape-report-2014.html.

[2] Incapsula. Lax security opens the door for mass-scale abuse of SOHO routers[EB/OL]. [2015-09-20]. https://www.incapsula.com/blog/ddos-botnet-soho-router.html.

[3] Incapsula. DDoS impact survey reveals the actual cost of DDoS attacks[EB/OL]. [2015-08-20]. https://www.incapsula.com/blog/ddos-impact-cost-of-ddos-attack.html.

[4] ZAHARIA M, DAS T, LI H, et al. Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters [C]// Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Ccomputing. Berkeley, CA: USENIX Association, 2012: 10.

[5] 孙大为,张广艳,郑纬民. 大数据流式计算:关键技术及系统实例[J]. 软件学报,2014,25(4):839-862. (SUN D W, ZHANG G Y, ZHENG W M. Big data stream computing: technologies and instances[J]. Journal of Software, 2014,25(4):839-862.)

[6] LEE Y, KANG W, SON H. An Internet traffic analysis method with MapReduce[C]// Proceedings of the 2010 IEEE/IFIP Network Operations & Management Symposium Workshops. Piscataway, NJ: IEEE, 2010:357-361.

[7] KHATTAK R, BANO S, HUSSAIN S, et al. DOFUR: DDoS forensics using MapReduce[C]// Proceedings of the 2011 Frontiers of Information Technology. Washington, DC: IEEE Computer Society, 2011:117-120.

[8] GOODHOPE K, KOSHY J, KREPS J. Building LinkedIn's real-time activity data pipeline[C]// Proceedings of the 2012 IEEE Computer Society Technical Committee on Data Engineering. Washington, DC: IEEE Computer Society, 2012: 33-45.

[9] KREPS J, CORP L, NARKHEDE N, et al. Kafka: a distributed messaging system for log processing[C]// Proceedings of the 2011 ACM SIGMOD Workshop on Networking Meets Databases. New York: ACM, 2011:231-240.

[10] Apache Software Foundation. Apache Kafka[EB/OL]. [2015-09-16]. http://kafka.apache.org/.

[11] Apache Software Foundation. Apache Spark: Lightning-fast cluster computing[EB/OL]. [2015-09-16]. http://spark.apache.org/.

[12] Apache Software Foundation. Welcome to Apache Hadoop[EB/OL]. [2015-09-16]. http://hadoop.apache.org/.

[13] ZAHARIA M, CHOWDHURY M, DAS T, et al. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing[C]// Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. Piscataway, NJ: IEEE, 2011: 141-146.

[14] Apache Software Foundation. S4: distributed stream compute platform[EB/OL]. [2015-09-16]. http://incubator.apache.org/s4/.

[15] Apache Software Foundation. Storm: distributed and fault-tolerance real-time computation[EB/OL]. [2015-09-16]. http://storm.apache.org/.

[16] Apache Software Foundation. Apache Kafka[EB/OL]. [2015-09-16]. http://kafka.apache.org/.

[17] 崔星灿,禹晓辉,刘洋,等. 分布式流处理技术综述[J]. 计算机研究与发展, 2015, 52(2): 318-332. (CUI X C, YU X H, LIU Y, et al. Distributed stream processing: a survey[J]. Journal of Computer Research and Development, 2015, 52(2): 318-332.)

[18] Apache Software Foundation. Apache Flume[EB/OL]. [2016-01-20]. http://flume.apache.org/.

[19] Databricks. Spark-perf [EB/OL]. [2015-09-16]. https://github.com/databricks/spark-perf.

[20] 虞立军,王建光,倪力. 使用 VisualVM 进行性能分析及调优[EB/OL]. [2013-02-18]. http://www.ibm.com/developerworks/cn/java/j-lo-visualvm/. (YU L J, WANG J G, NI L. performance analysis and tuning by using VisualVM[EB/OL]. [2013-02-18]. http://www.ibm.com/developerworks/cn/java/j-lo-visualvm/.)

[21] 周丽娟,王慧,王文伯,等. 面向海量数据的并行 *K*-Means 算法[J]. 华中科技大学学报(自然科学版), 2012,40(S1):150-152. (ZHOU L J, WANG H, WANG W B, et al. Parallel *K*-Means algorithm for mass data[J]. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2012, 40(S1):150-152.)

[22] 刘罕. 基于 Spark 框架的 DDoS 攻击检测系统研究[D]. 上海:上海海事大学, 2016:5. (LIU H. Research of the DDoS Attack Detection System base on the Spark Framework[M]. Shanghai: Shanghai Mritime University, 2016:5.)

[23] HETTICH S, BAY S D. KDD cup 1999 data [EB/OL]. [1999-10-20]. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

This work is partially supported by the National Natural Science Foundation of China (61672338, 61373028).

HAN Dezhi, born in 1966, Ph. D., professor. His research interests include cloud computing, cloud storage and security, big data application.

CHEN Xuguang, born in 1993, M. S. candidate. His research interests include cloud computing, big data real-time analysis.

LEI Yuxin, born in 1996. Her research interests include data mining, network security.

DAI Yongtao, born in 1991, M. S. candidate. His research interests include cloud computing, distributed computing, data mining, network security.

ZHANG Xiao, born in 1994, M. S. candidate. Her research interests include cloud computing, big data real-time analysis.