文章编号:1001-9081(2019)S1-0070-05

基于 Sketch 数据结构的海量网络流量实时排名系统

方 澄[1*],殷明瑞[2],张礼哲[3],孙佳慧[1]
(1. 中国民航大学 电子信息与自动化学院,天津 300300; 2. 北京邮电大学 国际学院,北京 100876;
3. 中国民航大学 信息安全测评中心,天津 300300)
(* 通信作者电子邮箱 cfangcauc@163.com)

摘 要:海量互联网流量数据服从幂律分布,因此对流量中排名前 k 个的服务提供商(SP)进行实时监控和了解,有助于运营商实时了解网络状态,便于网络管理。针对这个事实,提出一种采用概要(Sketch)数据结构的互联网流量实时排名系统。该系统实时记录网络数据流信息到 Sketch 数据结构,用 Sketch 图来保留整个数据流所有元素的概要信息。与数据流所有元素信息相比,Sketch 图占用更少的计算和存储资源,因此可以实现实时流量统计。此外,为适应大规模流式数据的需求,系统算法进行并行化,并部署在并行流式工作框架 Spark Streaming 之上,从而实现对海量网络流量的实时排名。该系统应用于运营商真实网络环境下,对真实网络环境下流量进行了实时跟踪和分析。通过大量实验首次给出了某省网络运营商真实流量的实时排名变化情况,发现在一天内流量排名虽然有小幅的变化,但整体排名顺序基本保持不变,腾讯公司是流量最大的服务提供商;同时还发现为了分流网络流量,减小网络流量压力,服务提供商将网页内嵌的图片、视频等对象转移到其他域名下。通过实际应用和测试验证了该实时排名系统的有效性。

关键词:概要数据结构;实时;大规模流式数据;Spark Streaming
中图分类号:TN711.1 **文献标志码**:A

Real-time ranking system for massive network traffic based on sketch data structure

FANG Cheng[1*], YIN Mingrui[2], ZHANG Lizhe[3], SUN Jiahui[1]
(1. *College of Electronic Information and Automation, Civil Aviation University of China, Tianjin* 300300, *China*;
2. *International College, Beijing University of Posts and Telecommunications, Beijing* 100876, *China*;
3. *Information Technology Security Evaluation Center, Civil Aviation University of China, Tianjin* 300300, *China*))

Abstract: Massive Internet traffic data follows power law distribution, therefore real-time monitoring and understanding of Service Providers (SP) with top-k traffic can help network operators to know the network state in real time and facilitate network management. Aiming at this fact, a system based on sketch data structure was proposed to rank Internet real time traffic. In the system, the Internet traffic data was stored in sketch data structure in real time, so that the sketch information of all elements in traffic flow was stored in sketch graph. Compared with the whole information of traffic flow, sketch graph needs smaller computation time and storage space, so as to be used to realize the real-time traffic statistics. Moreover, to meet the need of massive streaming data, the algorithms in the system were paralleled and deployed on the parallel streaming framework called Spark Streaming to rank massive Internet traffic in real time. The system was applied to real network environment of SP to trace and analyze traffic data in real time. The variation trend of the network traffic of a province was obtained for the fist time. The experimental results show that although the traffic rankings change slightly in a day, the overall ranking order remains basically the same, and Tencent is the SP with the largest traffic. It also can be seen that in order to divert network traffic and reduce network traffic pressure, SP transfer embedded images and videos of Web pages to other domain names. The availability of the system was validated by pratical application and testing.

Key words: sketch data structure; real-time; massive streaming data; Spark Streaming

0 引言

随着新兴互联网应用的不断涌现,以及移动互联网用户的不断增长,网络运营商的主要收入来源从语音服务转向了互联网服务。但由于近年来网络流量的爆炸式增加,网络运营商要想对网络上的所有服务提供商进行流量统计,其工作量巨大且很难实现。一些研究者已经开始使用并行架构来分析互联网网络流量,例如,Lee 等[1] 使用 MapReduce 编程模型构建了一个对万亿字节(Trillionbyte,TB)及千万亿字节(Petabyte,PB)级流数据进行分析的软件系统,并证明使用 MapReduce 分布式并行计算计算相比传统流数据分析软件,可以显著提高处理速度;Samak 等[2] 使用 Hadoop 技术和 Pig 分析语言,构建了一个可对美国 Internet2 互联网平台中产生的海量流量数据进行分析的数据处理平台;Francois 等[3] 使

收稿日期:2018-10-19;修回日期:2018-12-05。 基金项目:中央高校基本科研业务费资助项目(3122018C005);天津市智能信号与图像处理重点实验室开放基金资助项目(2017ASP-TJ04);中国民航大学科研启动基金资助项目(2017QD05S)。

作者简介:方澄(1980—),男,天津人,讲师,博士,主要研究方向:数据挖掘、大数据; 殷明瑞(1997—),男,天津人,主要研究方向:电信系统、图像识别、深度学习; 张礼哲(1986—),男,山东莱西人,助理研究员,硕士,主要研究方向:网络安全; 孙佳慧(1994—),女,内蒙古呼伦贝尔人,硕士研究生,主要研究方向:人工智能、情感分析。

用 MapReduce 技术对海量 IP 流量进行分析,进而识别出行为异常的僵尸网络节点。但以上研究都不能实时处理统计数据,无法实时反映网络流量的变化。众多网络流量的研究[4]表明,互联网流量分布呈现出幂律分布的特点,即绝大部分的网络流量仅仅由少数几个大型服务提供商占据,因此对于流量排名前 k 个(Top-k)的服务提供商的实时管控和了解更加有意义;将全流量分析聚焦到 Top-k 流量分析也使得对海量互联网流量的实时分析成为可能。Sketch 数据结构经常用于快速实时数据的统计应用,例如 Cormode 等[5]对 Sketch 数据结构在流式数据处理中的应用进行了总结; Charikar 等[6]使用 Sketch 数据结构利用有限的存储空间对流式数据中的频繁项进行统计; Yang 等[7]设计了 FID-Sketch 数据结构,并在图像处理器(Graphics Processing Unit, GPU)和多核中央处理器(Central Processing Unit, CPU)环境上进行了测试,在计算速度和准确性上都有很好的表现。虽然 Sketch 数据结构的研究在数据流式处理应用上取得了很多成果,但并没有将 Sketch 数据结构算法并行化,没有在真实海量流量上进行实际应用。

本文提出了一种基于 Sketch 概要数据结构的互联网流量实时排名系统,并将该系统并行化,以实现对大规模数据的处理能力。系统被部署到运营商真实网络中,使用真实网络的海量流量进行实验,并对真实网络中流量排名结果以及变化情况进行了分析。

1　Count-Min Sketch 算法

传统的计算 Top-k 的方式需要将全部数据信息存储,而在大数据时代,用传统的方式计算 Top-k 排名已经远远不能满足数据量的要求[8]。海量数据只能一次处理,将全部数据存储的空间代价是巨大的,因此出现了各种基于 Sketch 概要结构的算法。

概要图(Sketch)[9]是一种数据结构,其将数据流投射在一个小的存储空间内作为整个数据流的概要,这个小空间存储的概要数据称为概要图。相比存储数据流所有元素信息,用 Sketch 概要图来保留整个数据流所有元素的概要信息将占用更少的计算和存储资源,因此它更适合于大数据时代的数据流计算模式。

Count Min Sketch[5]就是基于 Sketch 概要图的代表性算法之一。它提出的是一个次线性空间的数据结构,由二维数组构成,如图 1 所示。

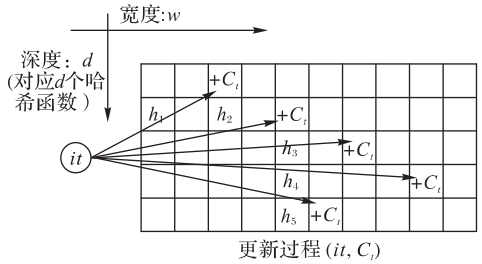

图 1　Count-Min Sketch 结构

二维数组宽为 w,深为 d,数组的每个元素对应一个计数器,即 $count[1,1]$, …, $count[d,w]$。数组的每个 $counter$ 计数值初始化为 0。用户规定两个参数 ε 和 δ 来保证一定的准确度,在此前提下,规定二维数组宽度为 $w = \lceil e/\varepsilon \rceil$,深度为 $d = \lceil \ln(1/\sigma) \rceil$。

算法采用一定的哈希映射规则,将数据流中的元素在每一行都映射到该行对应的 counter 中,由于是采用 hash 映射,不同元素对应的 counter 可能会重叠,因此会产生一定的冲突。该数据结构保留了数据流所有元素的概要信息,形成了整体的 Sketch 数据结构,因此可以对全部元素进行查询也可以对单个或部分特殊元素集进行查询[10]。

具体算法过程如下:

1) 函数映射: 二维数组共有 d 行,在每一行都有一个哈希函数对新来的元素进行哈希映射,将其映射到该行的某一个 $counter$ 中。这里采用 d 个两两相互独立的哈希函数 $hash_1$, $hash_2$, …, $hash_d$,各对应其中的一行进行映射。

哈希函数的构造采用随机数取模法,即设第 i 行的哈希函数为 $hash_i(j)$:

$$hash_i(j) = \{ [k1_i \times j) + k2_i] \% mod \} \% width \quad (1)$$

其中: $k1$、$k2$ 分别为随机产生的整数,mod 为取模的模底,$width$ 为二维数组的宽度。

函数将到来的每个元素 j 通过 $hash(j)$ 计算映射到了一个整数上,即 $[1, 2.\ n]$ 空间映射到 $[1, 2.\ w]$ 空间,其中 n 远远大于 w。

2) 更新过程: 当 t 时刻,新元素 (j_t, c_t) 到达时,表项被更新,对二维数组中的每一行,分别用 $hash_i(j)$ 计算元素 j 在第 i 行的 $index$,将映射的 $counter$ 值加上 c_t。公式为:

$$counter(i, hash_i(j)) = counter(i, hash_i(j)) + c \quad (2)$$

当所有数据流全部通过此表完成更新后,就得到了最终的二维数组。

3) 查询结果: 查询元素 j 所对应值的公式为:

$$Q(j) = \min(counter(i, hash_i(j))); \ i = 1, 2, \cdots, depth \quad (3)$$

哈希函数存在着映射冲突,不同的元素如 j,k,如果经过 hash 函数计算映射到同一个 counter 上,便会产生冲突。即该 counter 中的值为产生冲突的两元素的值之和。若元素的频数 c 为非负的,那么查询结果一定大于或等于实际的频数值。每一行都是如此,故选取查询结果最小的那一行最为近似的结果[11]。

2　Count-Min Sketch 并行流式算法

Spark Streaming 框架是基于 Spark 集群框架的流式计算框架[12],在处理大规模实时数据时具有以下优点:

1) 高容错性: 在任务执行上使用有向无环图(Directed Acyclic Graph, DAG),一旦后续任务失败,系统根据 DAG 中记录的操作回溯到前序任务,利用原始输入数据通过转换操作而重新恢复任务。

2) Spark Streaming 是基于 Spark 集群框架基础之上的,因此 Spark Streamin 可以使用 Spark 中丰富的操作函数来表达复杂的计算过程。

Spark Streaming 的工作原理如图 2 所示,Spark Streaming 首先将数据流切分成离散数据流,将该离散数据流定义为 DStream。每个 DStream 包含了一组连续的弹性分布数据集(Resilient Distributed Dataset, RDD)。RDD 是 Spark 框架中最为核心的概念,任何数据在 Spark 中都被抽象为 RDD 数据结构类型。Spark Streaming 将连续的 RDD 序列输入到 Spark 引擎,RDD 序列分别代表着不同时间段内的数据集。Spark 引擎持续不断地以并行的方式对输入的 RDD 流式数据进行快速处理,从而实现了对大规模数据源的流式处理。

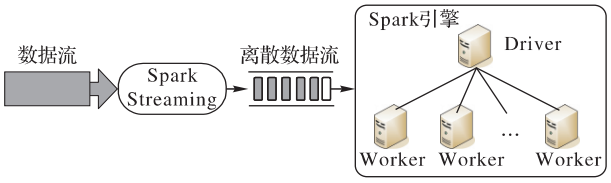

图 2 Count Min Sketch 并行流式算法原理

本文的 Count Min Sketch 并行流式算法基于 Spark Streaming 计算框架。算法的主要设计思想是将输入的数据进行分区处理，保证每一次更新迭代的过程中，相同的元素能被映射到固定的分区，从而保证了数据的完整性，不同的分区分别由 Spark 引擎中不同的 worker 计算节点进行计算处理。每个分区都维持了一个独立的 Count-Min Sketch 数据结构和一个针对该独立分区的 Top-k 排名数据。在一个 RDD 更新完成后，保存所有分区对应的 sketch 结构和 Top-k 数据，将所有分区的 sketch 结构和 Top-k 数据发给 Driver 节点进行存储，获取整体数据的 Top-k 元素。算法结构如图 3 所示。

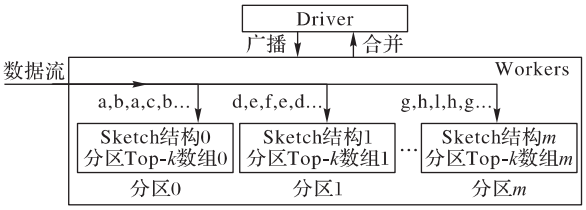

图 3 Count Min Sketch 并行流式算法原理

整个算法过程可分为 4 步:

1) 数据分区。

在分布式并行计算 Top-k 元素时，需要保证流式输入的相同元素固定分配到同一个分区中，这样才能保证各个分区的 Top-k 元素不会因为数据的分散而产生误差。算法中根据输入元素的 hashcode 进行分区，设分区数目为 N，将所有输入数据的 hashcode 对 N 取余数，余数作为分区的标识值 key，key 相同的元素会被分配到同一个分区，由同一个 worker 计算节点处理。

2) 数据处理。

在每个分区中建立 Count Min Sketch 数据结构，同时维持一个有序的 Top-k 元素数组 Array【String，Long)】。当有新元素(item，value) 到来时，向 sketch 二维数组中更新对应 counter 的值。在更新后查询该元素的 currentValue = query (item)，向已有的 Top-k 数组中添加或修改该元素，添加如下:

①若 Top-k 数组长度不超过 k (用户设定的 Top-k 数目)，则添加该新元素二元组(item，currentValue)，同时按照属性值从大到小排序。

②若 Top-k 数组中已有该新元素，且当前数组中 value 为 v，则将 v 更新为当前值 *currentValue*，同时按照属性值将 Top-k 数组从大到小排序进行重新排序。

③若新元素不在 Top-k 数组中，并且该元素的 currentValue 大于 Top-k 数组中属性值最小的二元组，则用该新元素替换 Top-k 数组中的最小值二元组，同时对 Top-k 数组进行重新排序。

这样在每次新元素到来时都会更新当前的 Top-k 元素。

3) 数据合并。

在当前 RDD 更新完成后，下一个 RDD 处理之前，需要在 driver 节点内存中保存各个分区的 sketch 数据结构以及 Top-k 数组。在下一个 RDD 处理时，Driver 节点会将各个分区的 sketch 数组和 Top-k 数组 Broadcast 到各个 worker 节点，接收到广播变量的 worker 节点会根据当前分配的分区标识值 Key 获取对应的 sketch 和 Top-k 数组，并以此为基础上，对新的 RDD 进行排序处理。

4) 获取全局 Top-k 元素。

当需要查询某一时刻的全局的 Top-k 元素时，只需将 driver 节点内存中的各个分区的 Top-k 数组进行合并、重排，从而获取全局的 Top-k 元素。

3 算法评估

3.1 可扩展性评估

Spark Sreaming 是基于 Spark 集群的并行流式计算框架，算法的并行度主要体现在 Spark 并行计算集群上。为此首先测试在处理单个文件时算法在 Spark 集群上的可扩展性。

实验使用的数据为某省运营商一天的流量数据集，共约 300 万条用户请求记录。在实验中，使用加速比 *Speedup* 来衡量并行算法的并行度，从而验证并行算法的可扩展性。加速比定义为 $Speedup = T_0/T_n$，其中 T_0 表示一个计算节点的运行时间，T_n 表示 n 个节点的运行时间。实验时分别使用 1、2、3、4、5 台计算节点，每台计算节点使用 1 个内核进行测试，来计算算法的加速比。

实验结果如图 4 所示，可以看出，随着节点数的增加，加速比也不断地增加，但加速比并不完全符合线性增加，而是慢慢减缓增加比例，这是由于随着节点数量的增加，节点的启动时间以及节点间通信开销也不断加大，因此算法的实际加速比会受到一定的影响。

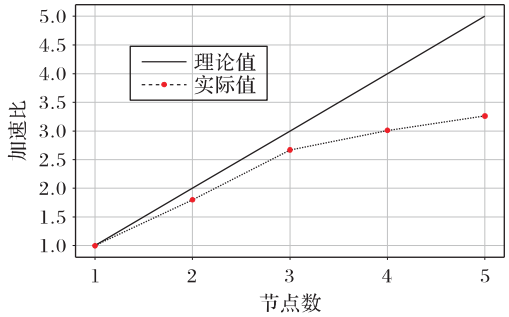

图 4 算法并行加速比

3.2 算法误差评估

Count Min Sketch 算法使用概要图来描述整体数据的分布，概要图使用很小存储空间，降低了存储比例，提高了统计速率，但概要图不可避免地引入了统计误差。本文对 Count Min Sketch 并行算法的误差进行了分析。

实验使用的数据集为经过分析处理的流量话单，话单记录了某省运营商某日的集成流量。话单以 10 min 为间隔，采集的是该 10 min 内各 SP 地址的下行累计流量。经过统计，一天的流量请求总数为 1 046 362 条，需要统计一天中不同时刻的 top500 的 SP 地址对应的流量值。

为了使实验环境更加真实地模拟应用环境，使用 kafka 与 spark streaming 进行对接，产生大规模数据流。实时的 Top-k 系统采取了多线程处理的方式，一个线程进行接收数据，一个线程进行处理数据，在第三个线程内实时地输出 top500 的记录值，用于前台分析。实验分别设置了 12 个输出点，分别为 2、4、6、…、24 将一天按照小时分为了 12 段，实时地统计该日的各时刻累积 top500 记录。

图 5 为一天每个统计时刻的 top500 平均相对误差。由

于统计的是数据流的累积流量，随着时间的增加，每个 counter 的计数值增加，各个 SP 地址哈希映射到 counter 时产生更多的误差，这些误差包含了别的元素的增幅，因此误差累积值在不断地增加，且增加的幅度比自生的实际流量增幅大，相对误差也在增大。但总体看来，相对误差的值整体较小，在 0.2% 以下。

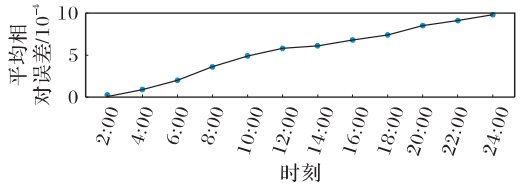

图 5　各时刻 top500 平均相对误差

3.3　实时性评估

本文实现的 Count Min Sketch 并行流式算法需满足实时性的要求。为此实验使用了一个持续时长为 2 h 的数据集，整个数据集的大小为 35.38 GB。将整个数据集以 5 min 为时间单位进行分割，对分割后的数据集发送给 Count Min Sketch 并行流式算法进行测试，测试每个小数据集的算法运行时长。并行计算集群由 12 台计算节点组成，每个计算节点有 8 个内核。算法的执行效率如图 6。

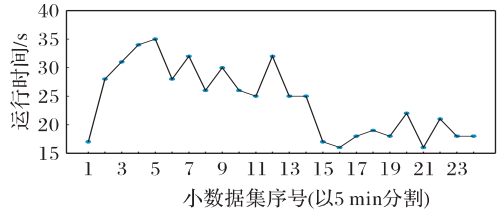

图 6　实时性评估

从图 6 中可以看出，算法计算效率基本稳定，平均处理 5 min流量数据仅仅需要 24 s。

4　分析

4.1　网络环境

通过以上实验，证明了本文中实现的CountMinSketch并行流式算法的可扩展性、准确性和实时性，该算法完全可以满足网络运营商在真实网络环境下的网络流量排名分析的需求。将该算法部署在某省网络运营商真实移动网络环境下的架构，如图 7。

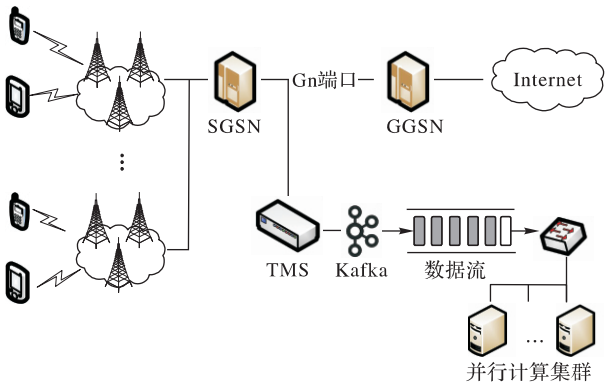

图 7　系统架构

该架构主要包括 3 个模块: 1) 流量镜像系统(Traffic Monitor System，TMS) 。TMS 是作者实验室自主研发的高性能流量镜像监测系统。系统采用深度包监测(Deep Packet Inspection，DPI) 技术，可以实现10 Gb带宽的实时网络流量镜像。TMS 设备被部署到运营商服务 GPRS 支持节点(Serving GPRS Support Node，SGSN) 和网关 GPRS 支持节点(Serving GPRS Supporting Node，GGSN) 间的 GN 接口上。GGSN 作为运营商与互联网的网关，为运营商的用户提供互联网接入服务，因此通过镜像 SGSN 和 GGSN 间的 GN 接口流量可以捕获所有用户的上网数据。2) Kafka。Kafka 负责将 TMS 镜像的数据流式地输入到并行计算平台中。3) 算法并行计算集群。并行计算集群将收到的数据进行分布式的存储，同时运行本文中的 Count Min Sketch 并行流式算法对流量进行实时的排名统计。

4.2　结果分析

分别对 8:00、14:00、20:00、24:00 四个运营商最为关注的时间点进行了流量累加排名观察，各个时间点的 Top-10 排名如图 8。

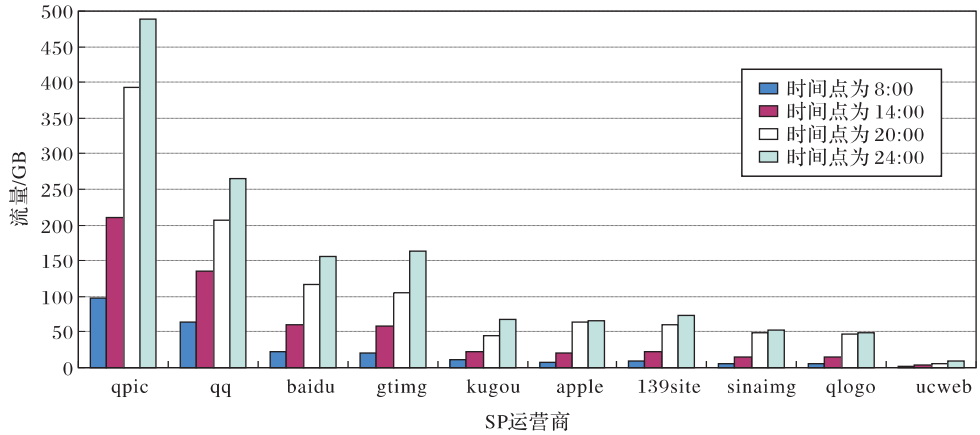

图 8　各时刻 top-10 排名

从图 8 中可以看出，一天 24 小时的流量排名虽然有小幅的变化，但整体排名顺序基本保持不变。腾讯公司因其具有庞大的用户群和多样的服务业务，该公司的三个主要域名“qpic”“qq”“gtimg”始终占据流量排名的前几位。另外从图中还可以看出，随着 SP 运营商服务内容变得越来越复杂，网页内容变得越来越复杂。为了分流网络流量，减小网络流量压力，SP 运营商将网页内嵌的图片、视频、层叠样式表(Cascading Style Sheet，CSS) 等对象转移到其他域名下，来分流主站域名的网络流量，例如“qpic”“gtimg”“qlogo”“sinaimg”等。此外还注意到，“baidu”和“kugou”作为国内主要的两个搜索引擎也在排名中居前，这是因为用户已经习惯于将搜索引擎作为自己访问互联网的入口，用户首先会通过搜索引擎搜索自己想访问的内容，再根据搜索结果进行访问。但同时也注意到，“kugou”作为国内第二大搜索引擎运营商无

论是网络流量的大小还是一天网络流量的增长速度都远远小于第一大搜索引擎运营商“baidu”,百度公司提供的搜索引擎服务占据了国内搜索引擎市场的大部分份额。

5 结语

为了满足网络运营商实时网络流量排名的需求,本文提出了一种基于 Sketch 概要图的并行流式算法,并将算法部署在 Spark Streaming 上,以实现对大规模流式数据的处理,通过实验测试,验证了该系统的可扩展性、准确性和实时性。下一步研究的工作有:1) 对网络流量进行实时跟踪,对 SP 网络流量变化进行建模。2) 实时监控网络流量变化,根据模型发现网络流量中的异常变化。

参考文献:

[1] LEE Y, KANG W, SON H. An Internet traffic analysis method with MapReduce[C]// Proceedings of the 2010 IEEE/IFIPNetwork Operations and Management Symposium Workshops. Piscataway: IEEE, 2010: 357 – 361.

[2] SAMAK T, GUNTER D, HENDRIX V. Scalable analysis of network measurements with Hadoop and Pig[C]// Proceedings of the 2010 IEEE Network Operations and Management Symposium. Piscataway: IEEE, 2012: 1254 – 1259.

[3] FRANCOIS J, WANG S, BRONZI W, et al. BotCloud: Detecting botnets using MapReduce[C]// Proceedings of the 2011 IEEE International Workshop on Information Forensics and Security. Piscataway: IEEE, 2011: 1 – 6

[4] SARRAR N, UHLIG S, FELDMANN A, et al. Leveraging Zipf' s law for traffic offloading[J]. ACM SIGCOMM Computer Communication Review, 2012, 42(1): 16 – 22.

[5] CORMODE G, MUTHUKRISHNAN S. An improved data stream summary: the count-min sketch and its applications[C]// Proceedings of the 2004 Latin American Symposium on Theoretical Informatics, LNCS 2976. Heidelberg: Springer-Verlag Berlin, 2004: 29 – 38

[6] CHARIKAR M, CHEN K, FARACH-COLTON M. Finding frequent items in data streams[C]// Proceedings of the 29th International Colloquium on Automata, Languages, and Programming. Heidelberg: Springer-Verlag Berlin, 2002: 693 – 703.

[7] YANG T, ZHANG H, WANG H, et al. FID-sketch: an accurate sketch to store frequencies in data streams [J]. World Wide Web, 2018, 2018: 1 – 22.

[8] 夏靖波, 任高明. 大流识别方法综述[J]. 控制与决策, 2013, 28(6): 801 – 807.

[9] BABCOCK B, BABU S, DATAR M, et al. Models and issues in data stream systems[C]// Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. New York: ACM, 2002: 1 – 16.

[10] DENG F, RAFIEI D. New estimation algorithms for streaming data: count-min can do more[EB/OL]. (2007)[2018-10-01]. file: ///C: /Documents% 20and% 20Settings/Administrator/My% 20Documents/Downloads/New_estimation_algorithms_for_streaming_data_Count. pdf

[11] BALACHANDRAN A, AGGARWAL V, HALEPOVIC E, et al. Modeling Web quality-of-experience on cellular networks[C]// Proceedings of 20th Annual International Conference on Mobile Computing and Networking. New York: ACM, 2014: 213 – 224.

[12] ZAHARIA M, CHOWDHURY M, FRANKLIN M J, et al. Spark: cluster computing with working sets[C]// Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing. Berkeley: USENIX Association, 2010: 10 – 10.

[13] FANG C, LIU J, LEI Z. Parallelized user clicks recognition from massive HTTP data based on dependency graph model[J]. China Communications, 2014, 11(12): 13 – 25.

[14] BHOLE Y, POPESCU A. Measurement and analysis of http traffic [J]. Journal of Network and Systems Management, 2005, 13(4): 357 – 371.

[15] KAMIYAMA N, NAKANO Y, SHIOMOTO K, et al. Investigating structure of modern Web traffic[C]// Proceedings of the 2015 IEEE 16th International Conference on High Performance Switching and Routing. Piscataway: IEEE, 2015: 1 – 8.

[16] BUTKIEWICZ M, MADHYASTHA H V, SEKAR V. Understanding website complexity: measurements, metrics, and implications [C]// Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference. New York: ACM, 2011: 313 – 328.

[17] de la OSSA B, GIL J A, SAHUQUILLO J, et al. Referrer Graph: A cost-effective algorithm and pruning method for predicting Web accesses[J]. Computer Communications, 2013, 36(8): 881 – 894.