众核与 Spark 结合的高速流量监测系统①

周小宇, 雒江涛, 罗 林, 唐 刚

(重庆邮电大学 电子信息与网络工程研究院, 重庆 400065)

摘 要: 互联网应用的广泛普及导致了互联网流量的高速增长, 这给网络运营商运营带来了巨大挑战, 传统的流量监测系统的性能和可扩展性已经无法满足运营商的需求. 本文尝试将众核技术与 Spark 相结合, 基于校园网流量, 提出了众核与 Spark 结合的高速流量监测系统. 其中, 众核处理器负责高速的流量采集、处理以及流量日志生成; Spark 平台负责存储流量日志数据, 并对其进行高速并行分析; Web Server 负责数据的可视化. 本文以校园网 DNS 流量为监测对象充分验证了该方案的可行性与扩展性.

关键词: 众核; Spark; DNS; 流量监测系统

High - Speed Traffic Monitoring System Based on Many-Core and Spark

ZHOU Xiao-Yu, LUO Jiang-Tao, LUO Lin, TANG Gang

(Electronic Information and Networking Research Institute, Chongqing University of Posts and Telecommunications, Chongqing 400065, China)

Abstract: Popularity of Internet applications has led to the rapid growth of Internet traffic, which brings great challenges for Network operators, and the performance and scalability of traditional flow monitor system cannot meet the needs of operators. Based on the campus network traffic monitoring, this paper attempts to combine many-core technology with Spark, and presents a high-speed traffic monitor system. Many-core processor part is responsible for high-speed traffic collection, processing and log generation; Spark platform handles the distributed storage of large amounts of log data and its high-speed parallel processing. Web Server is used for data visualization. The DNS traffic of the campus network is taken as the object of monitoring, which verifies the feasibility and extensibility of the system.

Key words: many-core; Spark; DNS; flow monitoring system

随着网络流量的爆炸式增长, 如何提升流量监测系统的整体性能是当前面临的严峻挑战, 近年来多核众核与大数据平台的引入成为解决该问题的突破点. 其中众核具有集成度高, 处理能力强等特点, 文献[1]利用众核平台大幅提升了网络入侵检测系统的性能; 文献[2]在 tilera 众核平台上实现了高吞吐量低功耗的网络应用识别; 文献[3]设计了一个基于 tilera 众核平台的高强度的流媒体流量发生系统. 大数据平台则具有强大的并行处理能力, 文献[4]提出一种基于 Hadoop 的网络流量分析系统, 它实现了基于 IP 层和 HTTP 层的流量分析; 文献[5]则实现了 Hadoop 平台下的用户网购偏好分析; 文献[6]提出了将 Spark 平台应用到 IDS 系统中, 完成高效的特征提取. 基于以上的研究方法和思路, 为了解决流量监测系统的性能问题, 本文提出了将众核与 Spark 相结合的方法, 它充分利用了众核的高速数据包处理能力, 以及 Spark 平台在大数据日志分析方面的优势, 并且具有较强的可扩展性.

该系统可以为运营商提供数据参考, 减少跨运营商流量结算费用, 以及为流量监测与分析提供新的视觉和参考.

① 基金项目:重庆市基础科学与前沿技术研究重点项目(cstc2015jcyjB0415);重庆市科技创新领军人才计划支持项目(CSTCKJCXLJRC20)

收稿时间:2016-11-22;收到修改稿时间:2017-01-04 [doi:10.15888/j.cnki.csa.005884]

1 系统组成

为了能实现高速的互联网流量处理, 以及准实时的日志分析, 可视化展示, 我们将整个系统设计分成三部分. 第一部分: 众核日志系统, 该部分的功能是高速捕获、分析网络流量, 并生成 DNS 会话日志; 第二部分: Spark 日志处理平台, 负责对 DNS 会话日志进行高效处理分析, 并将分析的结果传给 Web Server 的 MySQL 数据库中. 第三部分: Web Server, 负责对 Spark 平台传来的数据进行可视化.

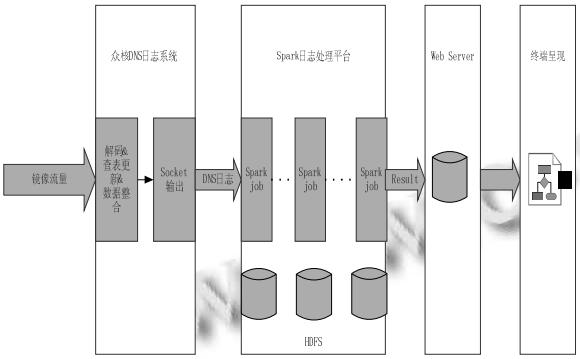

图 1 系统流程图

系统的流程图如图 1: 镜像的网络流量进入众核服务器, 经过解码、查表更新、数据整合之后通过 socket 将日志输出到 Spark 平台主节点, 主节点将日志数据存储到 HDFS(Hadoop Distributed File System)上, 并行的 Spark job 对 HDFS 上的日志进行分析处理, 并将处理结果传送给 Web Server, Web Server 对传来的数据进行可视化处理并最终在 Web 客户端得到呈现.

2 主要功能模块实现

2.1 基于众核的 DNS 日志系统

众核处理器芯片上集成了几十到上百个的处理器核心, 相比于多核处理器, 大大提高了处理能力. 和硬件芯片相比, 在众核处理器上编程灵活, 用户开发和维护的成本显著降低[7]. 本文采用的 TILERA-GX36 众核处理器芯片上集成了 36 个同构的处理器核心. 另外为了提高数据包处理能力, 该芯片上还集成了一个协处理器 MPIPE(multicore programmable intelligent packet engine), 它主要完成高速的数据包捕获、分类、负载均衡以及缓冲管理的功能.

MPIPE 的优点在于: 普通的 x86 服务器使用 libpcap 进行数据包的捕获, 此时 CPU 的多数时间都用来把网卡收到的数据包经过内核队列发送到用户空间. 即从网卡缓冲区拷贝到系统内核缓冲区, 再从内核缓冲区拷贝到用户空间这两个步骤, 耗费了大量的 CPU 时间[8]. 而 MPIPE 可以直接将数据包从网卡处送到用户态, 避免了内存的两次拷贝从而大大提高了数据包的捕获性能.

众核处理器的优点在于并行计算, 多核处理器上主要有数据并行(run-to-complete, RTC)和任务并行(software-pipeline, SPL)两种并行结构[9], 众核的并行结构与此类似. 以下将对这两种并行结构在本文中的应用与实现做详细介绍.

1) 数据并行

数据并行的结构中多个并行的线程被独立的绑定在一个核心上, 完成日志系统的所有工作, 核心之间没有竞争. 图 2 给出了这种结构的流程图

解码部分是根据 DNS 的日志分析需求从 DNS 数据包中提取相应的内容, DNS 日志的格式如下: “172.18.62.15|connc.gj.qq.com|20160719080000|120.198.201.206|183.232.119.217|183.232.103.218”其中‘|’作为字段分隔符, 第一个字段表示原 IP 地址, 也就是本次 DNS 请求发出方的 IP 地址, 第二个字段是请求域名, 第三个字段是本次 DNS 请求发出的时间(精确到秒), 剩下的部分是本次 DNS 请求对应的回复中含有的 IP 地址(如果有多个 IP 地址, 他们之间也通过‘|’分隔符隔开, 如无 IP 地址则不含该字段.). 解码过程就是从 DNS 请求包以及回复包中提取出来五元组、DNS 会话的 TransactionID, 并针对前者从中提取原 IP 地址、请求域名、请求时间, 针对后者从中提取回复 IP 地址. 存放解码信息的结构体是 Packetinfo, 因为解码部分得到的结果后面会拷贝到 hash 表的节点上, 所以每个处理线程只需一个 Packetinfo 结构体.

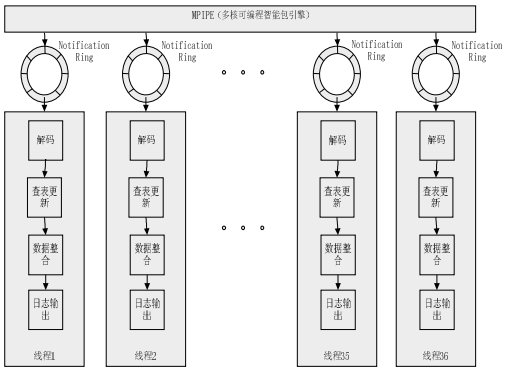

图 2 数据并行结构处理流程图

查表更新部分的主要工作是根据五元组作为关键字创建并维护一个 hash 表用来存储 DNS 会话的有效信息. 这里为了减少线程之间的竞争, 给每个处理线程都分配了一张 hash 表. 当接收到 DNS 请求包时, 查看表中是否有同样的请求, 有则表示这是一条重复的请求包, 该数据包处理过程结束, 没有则在表中添加新的节点并将该请求的有效信息复制到节点上; 当接收到 DNS 回复包时, 查看表中是否已经有了相应的请求, 无则该数据包处理过程结束, 有则将该数据包中的有效信息添加到相应节点, 至此本次会话结束, 节点会进入到数据整合部分.

为了提高性能, hash 表中存放 DNS 会话有效信息的结构体 Recordinfo 我们使用内存池的方式进行优化. 此处内存池是用链表的形式实现的队列, 我们还设计了维护该队列节点个数的函数. 为了减少处理线程之间的竞争, 我们分别为每个处理线程都分配了这样的内存池. 当需要在 hash 表中增加新节点的时, 从 Recordinfo 内存池中取出一个空结构体以供使用. 等日志中的信息被拷贝到发送的缓存区的时, 系统会将该节点删掉, 并对其数据进行清理之后重新放回到 Recordinfo 内存池中.

数据整合的作用就是将 hash 表中节点存放的有效 DNS 会话信息以字符串的形式整合起来方便输出.

由于本地存储空间有限, 所以日志输出的任务是将数据整合部分的结果通过 socket 输出到 Spark 平台.

2) 任务并行

由于解码, 查表更新, 数据整合几个部分联系紧密, 所以在任务并行的实现结构中我们仅将日志输出部分的任务拆分出来作为独立的线程运行, 任务并行结构的流程图如图 3. 从图中可以看到我们在数据整合之后增加了一个 ringbuffer 结构体, 通过共享内存的方式完成线程之间的通信, 为了避免多个写进程之间的竞争, 我们为每个处理线程都分配了 ringbuffer. 同时为了提高输出的效率, 我们又增加了一个缓冲结构, 它用来缓存多个 DNS 会话数据整合后的结果, 当该缓冲区中的数据达到一定的字节后, 处理线程会将该缓冲结构通过 ringbuffer 传递给日志输出线程. 对于缓冲结构的管理我们同样采用内存池的方式.

任务并行结构中日志输出线程的任务是循环的从各输出队列中取内容发送, 然后归还缓冲结构到相应的内存池中.

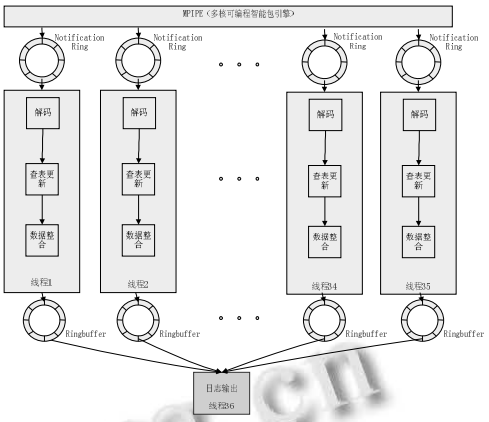

图 3　任务并行结构处理流程图

2.2 Spark 大数据平台

Spark 是 UC Berkeley AMP lab 所开发的类 Hadoop MapReduce 的开源集群计算框架, 使用 Scala 语言实现的, 建立在 HDFS 的上层. Spark 通过构建弹性分布数据集(RDD, Resilient Distributed DataSet)提供内存集群计算, 将 job 处理过程中的中间输出和结果保存在内存中, 从而减少读写 HDFS 次数,大幅提升处理效率, 使它非常适合机器学习与数据挖掘等需要迭代的 MapReaduce 的算法[10].

本文中 Spark 平台负责分布式存储大量日志数据, 并对其进行高速并行处理. 它主要完成日志采集、HDFS 存储、日志分析、结果输出等任务. 平台的主节点接收众核日志系统发送来的日志数据之后, 通过脚本将这些数据存储到 HDFS 上. 每隔一段时间对采集到的日志数据进行处理分析, 并将分析结果传送给 Web Server.

Spark 日志分析模块负责分析 DNS 会话日志, 分析内容主要包括:

(1) 发出 DNS 请求的原 IP 地址的热度.

(2) DNS 请求域名的热度.

(3) DNS 请求成功, 失败次数、请求域名 Ipv6 地址的成功的次数统计.

(4) 域名对应的 IP 地址个数的分布情况.

由于 Spark 对以上内容的分析流程大致相同, 在此以平台对域名热度进行的分析为例做详细的阐述如图 4: Spark 的 job 是由 RDD 进行计算的. 首先 Spark 从 HDFS 中将数据读入内存并转换为 RDD, 然后以“|”作为分隔符, 通过调用 Map 算子对每一条日志数据进行切割. 再调用 filter 算子过滤掉无效数据(主要是

DNS 会话日志中没有得到回复的数据), 并将每条数据集转换为以 name 为 key 以整数 1 为 value 的键值对. 以上的处理过程在 Spark 平台中并行有多个副本. 之后 Spark 使用 reducebykey 算子通过数据之间的 shuffle 统计各个域名访问的总次数. 最后将各个域名访问次数进行排序输出. 内容(1)的分析流程与以上流程类似.

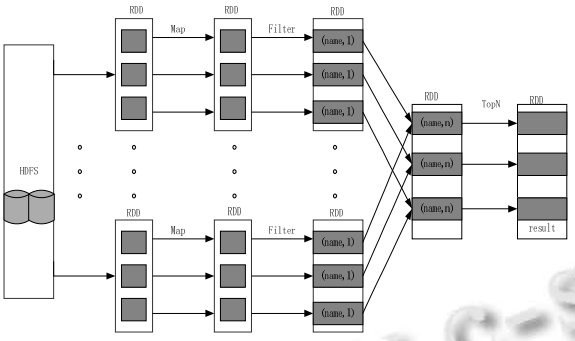

图 4 Spark 分析域名热度流程图

Spark 平台对内容(3)的分析是首先调用算子 Map 对数据切割, 然后根据成功、失败、Ipv6 的 DNS 会话日志的不同特点调用 filter 算子过滤掉无效数据, 最后对剩余数据统计次数即可.

内容(4)的分析相对于域名热度的分析, 只是将 filter 过滤后的每条数据集转换为以 IP 地址回复个数为 key 以 1 为 value 的键值对, 然后使用 reducebykey 算子通过数据之间的 shuffle 统计 IP 地址个数为正整数 n 的对应的域名的个数.

2.3 可视化呈现

数据可视化采用 Web 的形式呈现, 前端数据可视化工具采用的是 Echarts, HTML 框架为 Bootstrap 以及 Ajax. Web 后台采用的是 Java Web 框架 SpringMVC 和 Dbutils. Web Server 的 Mysql 数据库存放 Spark 平台传送来的处理结果. 可视化的整体流程如下: 前端界面采用 Ajax 异步加载的方式向 Web Server 请求相应数据, Web Server 收到请求, 从数据库中提取数据并转换 Json 格式返回给前端界面. 前端界面接收到数据之后, 对数据进行分析并加载.

3 实验分析

实验使用的众核平台为一个独立的 ATCA 刀片服务器, 服务器内部含有两个对称的众核系统, 每一个众核系统含有一颗 36 核处理器, 64 位指令集, 主频 1.2GHz, 内存大小为 16G. 服务器前面板有四个万兆光口, 连接服务器内部的 BCM 交换芯片, 并通过它连接对称的众核系统. Spark 平台由 5 台 Red Hat Enterprise Linux x64 虚拟机搭建而成, 每台虚拟机配置 500G 硬盘和 4G 内存.

在众核日志系统并行结构的选择上, 我们分别实现了两种并行结构. 经过测试我们给出两种并行结构的性能对比如图 5.

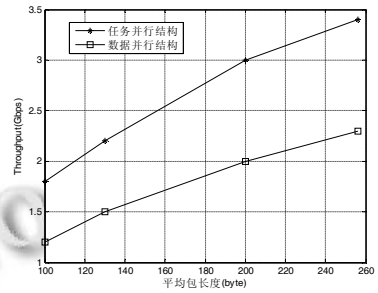

图 5 数据并行与任务并行性能

使用不同包长度的数据进行测试的结果显示任务并行具有一定的优势, 这主要是因为任务并行结构和数据并行结构相比, 程序的访问局部性和系统的缓存命中率都得到提高. 英特尔提出的 Supra-linear 模型利用任务并行方法对 Snort 并行加速[11]. 在配置 8 个处理器核心的服务器上, 和单进程 Snort 相比, 系统性能提高了 75%.

实验过程使用的流量是 8: 00: :00 到当天 23:59:59 之间从校园网 8 个端口截取的. 流量截取的时间选择主要是因为 8 点之前和凌晨之后的流量极小. 总的流量的大小为 5.6Gbps, 由于单颗众核 CPU 处理单纯 DNS 数据的性能为 2.2Gbps 而校园网流量中 DNS 流量较少, 所以系统完全可以处理实验中的流量. 实验过程中众核系统负责 DNS 日志生成的工作, Spark 平台每隔 20 分钟会处理一次日志, 并将结果传给 Web server, 通过终端浏览器进行可视化呈现.

实验结果的可视化展示如下:

图 6 表示一天时间不同时段发出 DNS 请求的源 IP 地址的热度.

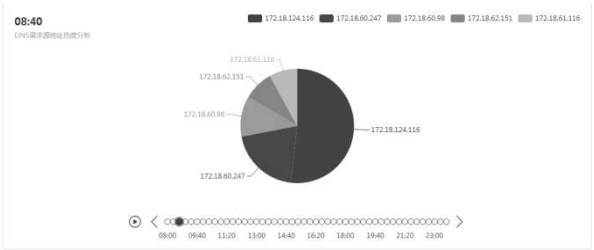

图 6 DNS 请求原 IP 地址热度分析

图 6 中使用饼图给出来从 8:40 开始的 20 分钟时间里发出 DNS 请求的 top5 源 IP 地址, 此图有助于分析当前发出 DNS 请求的活跃用户, 以及访问量.

图 7 表示一天时间不同时段 DNS 请求成功、失败、Ipv6 请求次数.

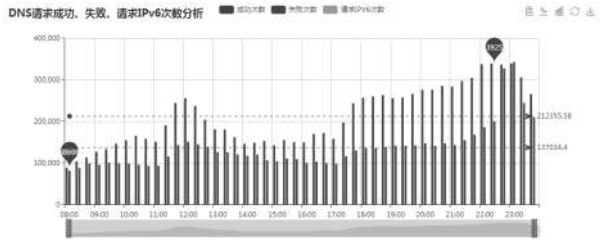

图 7　DNS 请求成功失败、请求 Ipv6 次数分析

图 7 中每隔 20 分钟给出三者的数据柱状图, 由于请求域名 Ipv6 地址的次数较少, 所以不易分辨, 但图中给出了具体的各种请求的次数. 采集时间内 DNS 请求成功的次数为 10193068、失败的次数为 6577651、Ipv6 的请求成功次数是 45647. 由此得到 DNS 请求的失败率为 39%, 可见校园网 DNS 请求失败率并不高. 请求成功的数据中 Ipv6 占比约 0.45%, 根据全球最大的 CDN 服务商 Akamai 给出的互联网现状 Ipv6 采用情况可视化结果显示中国的 Ipv6 的平均采用情况仅为 0.5%[12], 而 Ipv6 采用最多的国家是比利时占比为 46.8%, 可见当前中国 Ipv6 的商用规模很小相比发达国家落后很多. 另根据 2016 年中国互联网络发展状况统计报告显示我国 Ipv6 地址半年仅增长 0.9%, 可见中国的 Ipv6 发展非常缓慢, 该问题亟待解决.

另外, 从一天的 DNS 请求次数可以明显看出学生在宿舍的上网情况, 上午和下午的请求次数相近, 12 点左右数据量有所增加表示有部分学生中午回宿舍上网, 18 点以后 DNS 请求次数逐步上升直到 22 点到 23 点 20 分之间达到顶峰, 由此也可以分析学生在宿舍的上网习惯.

对 DNS 请求域名的分析可以用于了解学生的上网行为习惯. 图 8 表示一天中不同时段 DNS 请求域名的 Top10.

图 8 中每隔 20 分钟用饼图给出本时段内发出的 DNS 请求的域名热度情况, 这里域名的请求次数只计算给出相应域名 IP 地址的请求. 从对一天的数据分析可以得到的结果是 DNS 请求域名热度 top1 为 www.sina.com 而排在第二位的是 dns.msftncsi.com, 对该域名的请求之所以多的原因是 win7 系统默认的一个名为 Network Connectivity Status Indicator 的机制, 每当用户连接到网络时, 系统会自动的发出一条 DNS 请求, 请求的域名即 dns.msftncsi.com, 当 DNS 回复的 IP 地址与正确的 IP 地址: 131.107.255.255 不匹配时, 系统就会认为网络连接工作不正常.

图 9 是 20 点到 20 点 20 分的域名请求热度分析的结果.

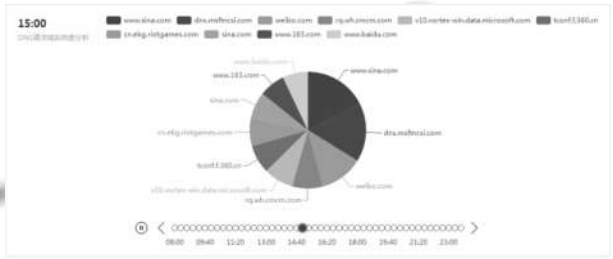

图 8　DNS 请求域名热度分析

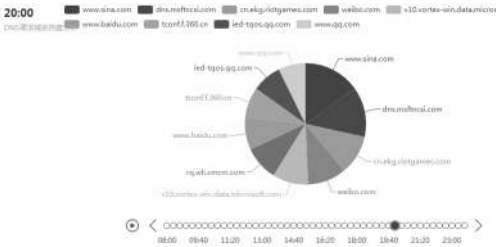

图 9　20 点域名请求热度分析

可以看到该时段请求的热度排名第三的域名是 cn.ekg.riotgame.com, 该时段之后的各个时段 top3 中都有该域名, 经查该域名属于美国的一家名叫拳头的游戏公司, 而该游戏公司至今为止只开发了一款游戏—英雄联盟, 该游戏在当下校园中的热度可见一斑.

4　结语

本文提出了一种将众核与 Spark 相结合提升基于校园网的 DNS 流量监测系统的性能的方法. 利用众核平台的高速数据包捕获和强大的计算能力可以提高日志系统吞吐量. 利用 Spark 平台的大数据实时并行处理能力可以提升系统的实时性和可扩展性. 系统未来将增加监测的流量类型, 进一步扩展系统的功能和性能.

参考文献

1 Jiang H, Zhang G, Xie G, et al. Scalable high-performance parallel design for network intrusion detection systems on many-core processors. IEEE symposium on Architectures for

Networking and Communications Systems. 2013. 137–146.
2 吴舜,苏丹,吴佳,等.基于 Tilera 平台的网络细粒度应用行为识别.电信科学,2013,29(11):94–98.
3 曾帅,高宗彬,赵国锋.基于 Tilera 众核平台的流媒体流量发生系统的设计.电子技术应用,2016,42(4):56–59.
4 Lee Y, Lee Y. Toward scalable internet traffic measurement and analysis with Hadoop. Computer Communication Review, 2013, 43(1): 5–13.
5 Yang JC, Luo JG, Shen J, et al. Online shopping preference analysis of campus network users based on MapReduce. International Conference on Cloud Computing. 2014. 138–143.
6 Karimi A M, Niyaz Q, Sun W, et al. Distributed network traffic feature extraction for a real-time IDS. 2016 IEEE International Conference on Electro Information Technology. 2016. 522–526.
7 姜海洋,谢高岗.众核处理器上的高性能网络入侵检测系统.高技术通讯,2014,24(9):935–941.
8 王佰玲,方滨兴,云晓春.零拷贝报文捕获平台的研究与实现.计算机学报,2005,28(1):46–52.
9 Verdu J, Nemirovsky M, Valero M, et al. MultiLayer processing-an execution model for parallel stateful packet processing. IEEE Symposium on Architectures for Networking and Communications Systems. 2008. 79–88.
10 刘季函.基于 Spark 的网络日志分析系统的设计与实现[硕士学位论文].南京:南京大学,2014.
11 Intel Corporation. Removing system bottlenecks in multi-threaded applications. Intel White Paper, 2008
12 互联网现状 Ipv6 采用情况可视化 Akamai. https://www.akamai.com/cn/zh/our-thinking/state-of-the-internet-report/state-of-the-internet-ipv6-adoption-visualization.jsp, 2016.