基于 Hadoop 的网络测量方法研究

马永 [1,2]，程光 [1,2]

（1.东南大学计算机科学与工程学院，南京，210096；2.网络和信息集成教育部重点实验室，南京，210096）

摘　要：随着网络数据量飞速增长且规模变得日渐庞大，目前常用的利用集中处理平台的网络测量方法已经很难满足要求。本文设计了一种基于 Hadoop 的网络测量系统，提出了基于数据报文的往返时延计算和 Netflow 流聚合的 MapReduce 算法，通过实验验证了算法的正确性和可扩展性，提出一种解决海量测量数据处理的方法。

关键词：Hadoop；HDFS；MapReduce；网络测量

Network Measurement System based on Hadoop

MA Yong[1,2] ,CHENG Guang[1,2]

(1. College of Computer Science and Engineering, Southeast University, Nanjing, 210096;

2. Key Laboratory of Computer Network and Information Integration, Ministry of Education Nanjing, 210096)

Abstract: With the rapid growth of network traffic and the scale becomes increasingly large, measurement methods currently used with centralized processing platform has been difficult to meet the requirements. This paper presents a Hadoop-based network measurement system, proposed round-trip delay calculation with packets and Netflow flow aggregation algorithm based on MapReduce, through experimental verification algorithm correctness and scalability, proposed a solution to solve the mass measurement data processing.

Key words: Hadoop; HDFS; MapReduce; Network Measurement

1. 引言

我们生活在一个“大数据”的时代。数据的爆炸式增长出乎人们的想象，据预计分析，2020 年，全球以电子格式存储的数据量将达到 35ZB，是 2009 年数据量的 40 倍，截至 2013 年 12 月，中国国际出口带宽为 3406824Mbps，年增长率 79.3%。

为了应对大数据时代的挑战，云计算的概念被提了出来，随着云计算技术的推广，利用云平台强大的数据处理能力来处理海量数据，已经成为当前网络测量的重要研究点，另外，云平台的计算资源和存储资源的可扩展性正是目前海量网络测量系统所需要的。Apache 的 Hadoop[1]是一个开源云计算项目，项目目标是建立一个能够对海量数据进行分布式处理的可扩展云计算框架；Hadoop 实现了一个分布式文件系统 HDFS，并利用 Map/Reduce 算法框架，同时提供了 HBase 这种分布式数据库存储技术，Hadoop 云平台可以部署在低廉的硬件上减少硬件的投入成本；Hadoop 充分考虑到了数据的安全性，设计了副本保存机制；项目充分考虑到可扩展性，Hadoop 集群可以进行运算节点的增加以获得处理更大的数据量的能力，加上平台是开源免费的，使得 Hadoop 技术得到了迅猛发展，目前，Hadoop 广泛使用在雅虎，Facebook，IBM，百度，阿里巴巴等公司，成为了最著名的云计算项目之一。

目前的网络测量环境多采用单机处理，无论是实时数据还是离线数据都是将数据实时发送或者存储在一台服务器主机上，然后进行测量分析，但是这种方法会因服务器主机的计算能力和存储能力有限导致其处理的数据量受限；目前面对海量网络测量数据常用方法是采用抽样技术，但是对抽样后的数据进行分析对网络的真实情况反映有所欠缺。本文根据网络测量的实际需要，设计了基于 Hadoop 云平台的可扩展的网络流量测量系统，在云平台的集群系统中通过测量探针对海量测量数据进行可靠的回收，然后利用云平台强大的数据处理能力来处理海量数据。对于数据报文，设计了往返时延 MapReduce 实验，对于 Netflow 流数据，设计了流聚合 MapReduce 实验，通过这两个实验分析基于

作者简介：马永，（1989-），男，硕士研究生，E-mail: yma@njnet.edu.cn；程光，（1973-），男，教授，博导，E-mail: gcheng@njnet.edu.cn.

Hadoop 的网络测量方法解决海量数据处理问题。

论文结构如下，第一部分是相关工作介绍；第二部分是系统的设计以及并行算法相关问题研究；第三部分是系统实验及其效果的评估，分别从准确性，可扩展性等对系统进行评估；第四部分是总结部分，主要是分析系统的可取之处以及改进的方向。

2. 相关工作

过去一段时间，很多测量工具被开发并广泛用于互联网流量监测中。其中，TcpDump 是用于捕获和最常用的工具网络包的分析的工具，使用的比较流行的 LibPcap 库。Wireshark 的是一个流行的流量分析仪，提供用户友好的图形节界面和统计功能。CAIDA 开发了 CoralReef，提供灵活的流量捕获，分析，并重新端口功能。Snort 的是一个开源的入侵检测工具，设计用于支持实时的分析网络入侵信息。Tcptrace，用于分析网络传输层的各种测度。Ntop 是一个用来实时监视网络使用情况的工具，Ntop 同时具备深度报文检测的功能。另一方面，思科的 NetFlow 是一个众所周知的流监控格式，有很多开源分析工具，比如，Flowscan 和 Flow-tools。然而，一般而言，上述的多数互联网流量测量和分析工具在一台服务器上运行。

对于并行测量分析，DIPStorage[2]使用一个 P2P 平台，以并行的方法处理网络数据流；Chen 等人开发了一种使用 Hadoop 的 snort 日志数据分析方法[3]应用于大规模网络安全应用；RIPE[4]宣布了开发了一个用于 Hadoop 云平台的 Pcap 接口库用于对 Pcap 直接并行读处理，JongSuk R. Lee 等人利用 Hadoop 平台来检测电信网络异常业务[5]。

本文设计一个基于 Hadoop 的网络测量系统并研究网络测量的并行算法，提出一种解决海量测量数据处理的方法。

3. 基于 Hadoop 的网络测量系统设计

在本节中，论文描述了基于 Hadoop 网络测量系统的框架图和物理图，并设计往返时延的 MapReduce 算法和流聚合 MapReduce 算法。

3.1 测量系统设计

下图 1 是基于 Hadoop 网络测量的框架图，从下到上可以将整个框架层分成三层，数据收集层，并行算法层，查询层。数据收集层的功能主要是接受数据报文，或者 Netflow 流数据，这些数据主要来自分布在网络中的探测器或者路由器镜像数据；并行算法层是整个框架图的最重要的部分，论文利用 MapReduce 思想编写了基于数据包的往返时延算法和基于 Netflow 的流聚合算法，这两种算法一个是基于数据包的，一个是基于流的，论文研究这两种算法主要为海量报文数据和海量流数据探索寻求在 Hadoop 云平台下的解决方法；MapReduce 是一种处理海量数据的并行编程模型和计算框架，用于对大规模数据集的并行运算；MapReduce 计算框架最早是由 Google 提出来的，Hadoop 的 MapReduce 是其的开源实现；查询层负责对数据结果的形象化展示。

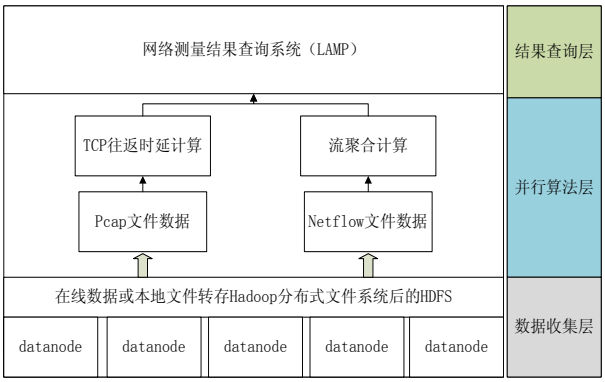

图 1 系统框架图

基于 Hadoop 的测量系统的数据收集物理图如下图 2 所示，数据来自边界路由器，经过数据分流点进行分流后，发送到各个 DataNode 节点，各个 DataNode 节点和 NameNode 节点用交换机连接共同组成一个 Hadoop 集群。

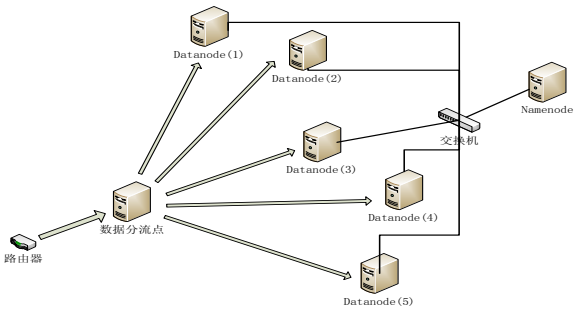

图 2 数据收集物理图

论文存入 HDFS 的数据是处理后的报文数据和

Netflow 流数据，这些数据是行记录数据，既当数据发送到 DataNode 节点后，DataNode 所在节点的本地主机会做转换处理，无论是包数据还是流数据，将需要的测度以文本格式按行输出存入 HDFS，论文具体的处理方法是使用 Libpcap 提供的接口读取网卡上数据包，然后在 DateNode 所在节点进行组流处理，然后利用 Hadoop 提供的接口，将数据写入 HDFS 文件系统中，此时，NameNode 节点会对存入的文件建立文件目录（因为在调用接口写时，Namenode 节点会对即将写入的文件建立文件元数据信息）。

3.2 基于 TCP 协议的往返时延并行算法

对于全报文无抽样的数据包进行 RTT 测量，模型和方法相对简单，我们可以通过分析 TCP 的建立，传输过程，结束过程来计算 RTT。下图 3 展示了 TCP 的建立，传输和结束释放的过程，然后标记了获取数据的位置点，位于中间的那条线。

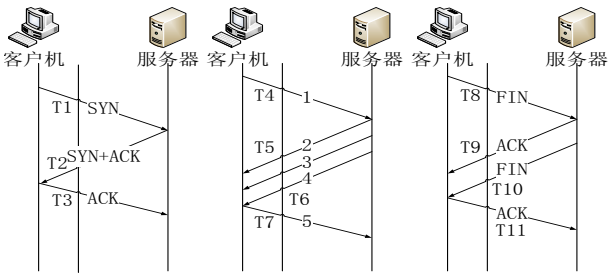

图 3 RTT 计算并行图

在 TCP 连接建立阶段：

$RTT = (T3 - T2) + (T2 - T1)$。

在 TCP 正常传输阶段：

$RTT - (T7 - T6) + (T5 - T4)$。

在 TCP 连接释放阶段：

$RTT = (T11 - T10) + (T9 - T8)$。

在利用 MapReduce 算法对 TCP 的往返时延进行计算时，我们面临着两个挑战：一个是将分散在不同 HDFS 块中的往返数据包进行整合；另一个是如何选择高效合理的方法在 MapReduce 架构中计算 RTT。

如下图 4 所示，这里假设有两个 Block 文件，输入文件 1 和输入文件 2，文件中的每一个字母加符号代表一个报文，A+，A-，B+，B-，C+，C- 代表了 6 个报文，这里用正负号代表方向，既 A+和 A-共同组成了具有往返通信的报文组合。

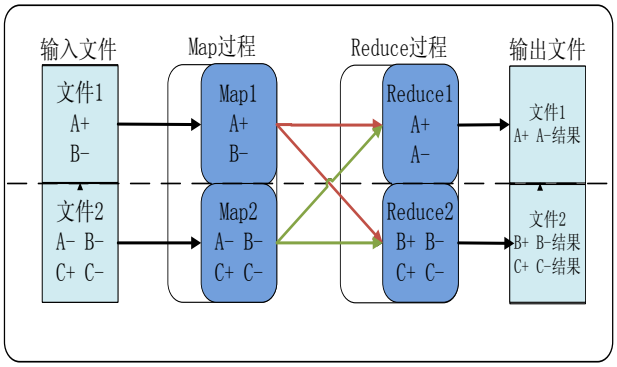

图 4 RTT 计算并行图

在计算 RTT，需要把往返方向的数据包聚合在一起才能够计算出结果，在单机处理时，基本方法是利用四元组或者五元组进行求 Hash 的 Key 值将往返通信的报文组合在一起；考虑到 MapReduce 有自己的<Key, Value>对，为了将往返的报文聚合到一起，论文采用了一种重新定义数据包方向的方法保证往返数据包聚合在一起。

在 Map 阶段，输入的是数据报文的各种信息，处理过程中，论文将 Key 定义为大 IP 地址连接小 IP 地址（既比较源 IP 地址和目的 IP 大小，然后将大 IP 地址作为源 IP 地址，小 IP 地址作为目的 IP 地址）连接大 IP 地址对应端口连接小 IP 地址对应端口组成的四元组作为 Key 值（因为协议这里面只用 TCP 协议，所以五元组当做 Key 变成了四元组 Key），Value 定义成标志位（本身就是大 IP 地址为源 IP 地址，小 IP 地址为目的 IP 地址的报文设置标志为 1，对于需要进行调整的报文，标志为 2）连接报文 Seq 值连接报文 Ack_seq 值连接 Fin 字段连接 Syn 字段连接 Ack 字段连接时间秒连接时间微妙。到此我们定义出来 Map 阶段输出的<Key, Value>对，经过 Map 函数的处理，往返的 A+,A-可以归类到同一个 Reduce 去处理，同理 B+，B-，C+，C- 。

Reduce 阶段，程序处理的是具有相同 Key 值的 Value 的集合，然后根据每一个 Key 会运行一次 Reduce 函数，Reduce 开始阶段，利用 Static 数据不会在程序结束之前释放和 Static{}结构会在类生成对象前运行且只运行一次的特点，论文在 Static{}结构中申请大量空间用于 TCP 计算，避免重复申请浪费计算机资源和减慢程序速度。在计算 RTT 的时候，论文使用两个 Hash 数组（使用链表来解决 Hash 冲突）分别去存储标志 1 和标志 2 的数据，对于每个 Hash 数组，结构形式都如下如 5 所示：

图 5 Hash 数据结构形式图

这里的 Key 都是报文的 Ack_Seq 值（上图黑色斜体数据值），然后记录这个 Ack_Seq 值的最小的 Seq 值和最大的 Seq 值，以及它们的负载长度，时间值等（对于 Syn 报文和 Fin 报文，论文人为的对负载长度加 1，用于抵消 Syn 报文和 Fin 报文用去 1 个报文负载），此举目的主要是只选择连续发送报文的两头报文用于计算，以减少不必要的存储和计算，当对同一个 Key 的所有 Value 计算完后，我们用标志 1 的 Hash 数组 Seq 值加上报文负载长度作为 Key 去求标志 2 的相应位置有无数据，如果有数据记录，那么说明说明从标志 1 的报文有一个标志 2 反方向的报文回复，即可计算得到半个方向的 RTT1，然用同样的方法反方向计算，有可得到另外半个方向的 RTT2，将 RTT1 加上 RTT2 可以得到这个报文中两个 IP 地址通信之间的 RTT 时间。

3.3 流聚合并行算法研究

集成环境下的流聚合统计可以有许多种，我们可以是基于往返流的聚合，可以是基于三元组的聚合（源 IP 地址，目的 IP 地址，源端口或者目的端口），可以是基于一个 IP 地址的聚合。不同的流聚合方式有不同的后续测量操作目的，但是流聚合的基本方法是一样的，都是利用要聚合的值作为 Key 值利用 Hash 函数处理，论文下面介绍 MapReduce 框架下的基于 IP 的流聚合统计算法，输入的是 Netflow 格式数据。

用 Hadoop 云计算平台对流进行聚合运算有天然的优越性，Hadoop 云平台的计算，存储等资源的可扩展性，能很好的满足大数据的需要，该并行化实现的主要工作就是设计过程中的几个函数：Map 函数，Combine 函数和 Reduce 函数函数的设计[1]。

A）Map 函数的设计:

函数的输入为<Key1, Value1>键值对，这些键值对是 Netflow v5 格式按行记录的文本数据，Key1 是 Block 文件中的文件行数，Value1 是每行中的内容；函数的输出为<Key2, Value2>键值对，具体的内容见 Map 函数的伪代码。函数 Map 的伪代码为：

```
Map(<Key1,Value1>，<Key2,Value2>)
{
    不考虑 Key1 值，对于 Value1 分解出数值 V1 V2 V3 V4 V5 V6 V7 V8 V9；//V1~V9 是输入文件测度值

    输出 Key2 = V1；
    输出 Value2 = 1+V2+V3+V4+V5+V6+V7+V8+V9；//加号是连接符
    输出 Key2 = V2；
    输出 Value2 = 2+V1+V3+V4+V5+V6+V7+V8+V9；//加号是连接符
}
```

B)Combine 函数的设计:

输入的是 Map 函数输出的结果，本函数对结果中具有相同的 Key 键值对进行合并，产生一个新的元组输出，这种操作可以减少中间结果，既键值对的数目，从而减少数据传输中的网络流量。

函数的伪代码为:

```
Combine(<Key1,Value1>，<Key2,Value2>)
{
    List（Key1）；
    对同样 Key1 的 Value1 集合进行用特殊符号“|”进行连接运算；

    输出 Key2 = Key1；
    输出 Value2 = Value1|Value1|Value1……；
}
```

C)Reduce 函数的设计：

这个阶段是流聚合的主要阶段，所有的运算操作都在这个阶段完成，主要包括申请静态空间保证整个阶段不会重复申请来减少时间和资源的花费，然后利用 Map，Hash 等数据结构统计基于 Key 的情况，然后计算输出需要的结果。

函数的伪代码为:

```
Reduce(<Key1,Value1>，<Key2,Value2>)
{
    Static 变量指针；//定义类使用的静态变量指针
    Static
    {
        对变量指针进行内存的申请；
    }

    接收一个<Key1,Value1>；
    For（Value1）
```

根据“|”分解 Value1，得结果 valtemp；

For（valtemp）

根据“+”分解 valtemp，得结果 val1 val2....；

根据 val 里面的标志分成源 IP 情况和目的 IP 地址情况分别统计；//这里面的标志是 Map 函数阶段标记的“1”或者“2”

用 HashMap 数据结构统计与此 IP 地址有关的源 IP 地址个数以及宿 IP 有关的个数。

利用变量来统计 Byte 个数，流个数，报文个数等；

利用变量统计协议情况，不同端口个数等；

用数组结构统计排名在前三的端口号及比例；

对上面的统计结果进行输出整理；

输出 Key2 = Key1；

输出 Value2 = 各种统计后的结果值；//论文一个统计出来 50 个测度值

}

对上述算法进行处理后，会形成 50 个测度值[6]，如下表 1 所示：

表 1 流聚合结果测度值

源 IP 地址属性		目的 IP 地址属性	
X_{1s}	开始时间	X_{1d}	开始时间
X_{2s}	结束时间	X_{2d}	结束时间
X_{3s}	数据包字节数	X_{3d}	数据包字节数
X_{4s}	数据包数量	X_{4d}	数据包数量
X_{5s}	流数量	X_{5d}	流数量
X_{6s}	源端口数量	X_{6d}	源端口数量
X_{7s}	目的端口数量	X_{7d}	目的端口数量
X_{8s}	目的端 IP 数量	X_{8d}	源端 IP 数量
X_{9s}	TCP 比例	X_{9d}	TCP 比例
X_{10s}	UDP 比例	X_{10d}	UDP 比例
X_{11s}	Other 协议比例	X_{11d}	Other 协议比例
$X_{12\sim18s}$	前 3 个源端口号以及比例	$X_{12\sim18d}$	前 3 个源端口号以及比例
$X_{19\sim25s}$	前 3 个目的端号以及比例	$X_{19\sim25d}$	前 3 个目的端口号以及比例

注：表中源 IP 地址属性是指聚类的 IP 作为源 IP 地址，目的 IP 地址属性是指聚合的 IP 地址作为目的 IP 地址。

利用上述测度值的不同组合可以反映出很多网络信息，比如，可以利用上述的开始时间，结束时间，字节数三个测度计算此 IP 地址的进出流量速率，可以使用字节数，流数量，不同 IP 数量三个测度的比率可以粗粒度的定位此 IP 地址的用户行为（客户端，服务器端，P2P 节点），另外，可以通过对结果设定阀值来确定活跃主机的 IP 地址，论文将在实验部分简单分析基于上面测度对网络用户行为粗粒度的分类。

4. 实验结果

本实验搭建的 Hadoop 环境是使用在一台服务器上创建的 6 台 Vbox 虚拟机组成的集群，其中虚拟机 CPU Intel Xeon E5-2403 1.80GHz，内存 700M，Disk 30GB，网络通信速率 100Mbps，操作系统采用 Centos6.2（内核是 2.6.32）。Hadoop 集群部署信息如表 2 所示：

表 2 集群部属情况

机器名	IP 地址	进程信息
Master.hadoop	10.129.3.200	NameNode，JobTracker SecondaryNameNode
Slave1.hadoop	10.129.3.201	DataNode，TaskTracker
Slave2.hadoop	10.129.3.202	DataNode，TaskTracker
Slave3.hadoop	10.129.3.203	DataNode，TaskTracker
Slave4.hadoop	10.129.3.204	DataNode，TaskTracker
Slave5.hadoop	10.129.3.205	DataNode，TaskTracker

注：表中机器名主机名，IP 地址是主机名对应于的 IP 地址，进程信息反映了该机器的功能作用。

实验的数据来自部分东南大学 Bras 动态分配地址对江苏省教育网外地址的流量。日期为 2013 年 3 月 25 日 10 点 34 分开始，时长为 60 分钟的数据。在实验中，我们将这 60 分钟的数据根据粒度划分为 5 组测量数据。表 3 为本次实验数据的相关信息。其中第 1 列为组标号，第 2 列为当前组时长，第 3 列为当前子区间内的报文数，第 4 列为当前子区间内的流数，第 5 列为当前子区间内的字节数。

因为轮文采用的使用虚拟机来实现整个集群，因此，对于上面的数据论文是直接事先处理好，形成了行记录的数据报文数据和组好的 Netflow 数据，然后调用系统命令 Hadoop fs -put 将数据集存如 HDFS 文件系统中。

表 3 实验数据

组标号	时长	报文数	流数	字节数
第一组	5min	1581519	305095	2.58G
第二组	10min	3210460	623586	4.52G
第三组	20min	6489216	1275033	9.24G
第四组	40min	13159528	2743533	16.34G
第五组	60min	19876481	3424516	24.29G

注：表中的字节数是指对应时长的全负载报文字节数。

对于往返时延的计算，论文用 Tcptace[12]开源软件论文的并行算法进行了正确性实验，在抽样 100 个 IP 对进行比较后，如下图 6 所示是 Tcptace 计算的往返时延和论文并行算法计算的时延值的差的绝对值，结果误差在 100 μs 内，证明了论文并行往返时延算法的正确性。

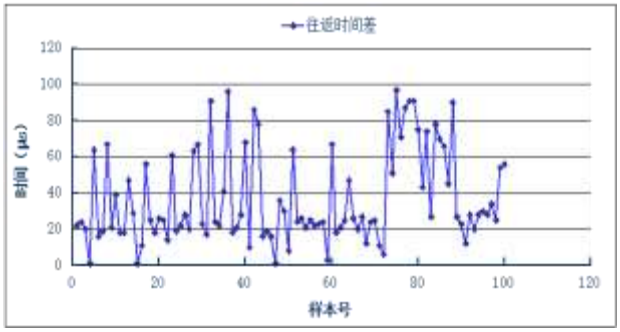

图 6 往返时延差值图

对于流聚合部分，论文也进行了正确性的分析，具体方法是抽样 100 个样本，然后用 Tcpdump 过滤相关 IP 地址，用单机处理方法进行流聚合，论文比较了两种方法的得出的 50 个特征值结果，对于同一个 IP 的聚合结果是一致的。另外，论文对结果进行了简单的统计分析，如下图 7 所示。

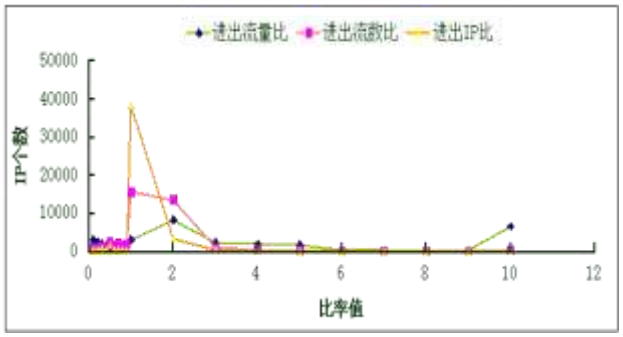

图 7 流聚合统计图

因为流聚合往往是某些具体的网络测量的预处理部分，比如可能通过分析单个 IP 地址的聚合结果去分析 IP 地址用户行为，或者去分析此 IP 地址是否有异常情况等。因此，研究流聚合的并行算法有很重要的实际意义。

上图统计了进出流量比，进出流数比，进出 IP 比三条信息，从图可知，大部分比率都分布在 1 附近，反应了网络中用户行为的情况，大部分是 P2P 主机（P2P 主机的进出流量比率，进出 IP 比率接近 1），分布在两边的，分别是服务器主机和客户端主机，上图可以间接证明流聚合结果的正确性。

根据 Hadoop 集群的介绍，随着节点数的增加，计算能力不断扩展，任务完成的时间逐渐减少。论文使用第一组数据做了扩展性实验，如下 8 图所示：节点数的增加让运行时间越来越少，但是任务完成时间减少的幅度在小任务量时间的作业上反映出来的不是很明显，且节点数的增加并没有导致作业运行时间成相对应倍数的减少。

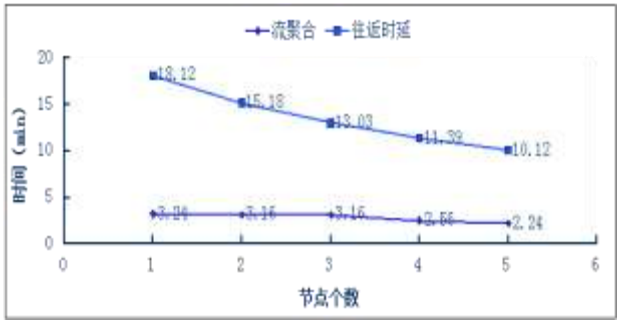

图 8 节点/运行时间对比图

造成这种现象的原因主要是论文的实验平台是利用虚拟机环境，因为所有节点共用一台服务器的资源，当部分节点运行作业时，同样利用整个服务器资源（可以理解成节点性能相对得到了提升），另外一个原因是当作业量很少时，作业真正计算 CPU 时间占比很少，大部分时间是 MapReduce 流程中的必要时间，这也就解释了为什么在流聚合时间在节点数变化时基本没有时间的变化。

5. 结论

论文的主要工作是设计一个可扩展的基于 Hadoop 云计算机的网络测量分析系统，可以用于分析海量网络数据中有用的信息。论文基于 Hadoop 的网络测量系统的数据源既可以是数据包，也可以是 Netflow 记录，论文实验了基于报文的往返时延计算和基于 Netflow 的流聚合统计，验证了正确性

和可扩展性。通过 Hadoop 实验测试平台，我们已经证明，在基于 MapReduce 的流量分析方法，实现了比单机更快的处理效率。论文证明，用 Hadoop 提供的向外扩展能力，是一个很好的解决暴涨的网络数据测量的方法。本文的不足之处是 Hadoop 平台是一个批处理的云计算平台，使得论文对实时的大数据流量无法处理，本文下一部研究如何利用 Hadoop 或者其他实时云计算平台解决实时海量数据处理问题。

参考文献

[1] Hadoop. http://hadoop.apache.org.

[2] C. Morariu, T. Kramis, B. Stiller DIPStorage. Distributed Architecture for Storage of IP Flow Records. 16th Workshop on Local and Metropolitan Area Networks,September 2008.

[3] W. Chen and J. Wang, Building a Cloud Computing Analysis System for Intrusion Detection System. Cloud Slam 2009.

[4] RIPE Hadoop PCAP. https://labs.ripe.net/Members/wnagele/large-scale-pcap-data-analysis-using-apache-hadoop.

[5] JongSuk R. Lee Sang-Kug Ye and Hae-Duck J. Jeong. Detecting Anomaly Teletraffic Using Stochastic Self-similarity Based on Hadoop, Network-Based Information 2013.

[6] 刘璇, 张凤荔, 叶李. 智基于 Netflow 的用户行为挖掘算法设计. 计算机应用研究, 2009.

[7] K.Cho, K.Fukuda, H.TEsaki, A.Kato. Observing slow crustal movement in residential user traffic. ACMCoNEXT2008.

[8] Youngseok Lee, Wonchul Kang and Hyeongu Son, An Internet Traffic Analysis Method with MapReduce, Network Operations and Management, 2010.

[9] 冷芳玲, 鲍玉斌, 高伟, 于戈. 基于 MapReduce 的数据聚集运算算法.北中国科技论文在线, 2011.

[10] Y QIAO, Z LEI, L YUAN, M GUO. Offline traffic analysis system based on Hadoop. The Journal of China Universities of Posts, 2013

[11] Cisco Netflowt. http://www.cisco.coom/web/go/netflow.

[12] tcptrace. http://tcptrace.org.

[13] J. Dean and S. Ghemawat, MapReduce: Simplified Data Processing on Large Cluster. USENIX OSDI, 2004.

[14] 刘鹏. 实战云计算. 北京:电子工业出版社, 2010.