An Internet Traffic Analysis Method with MapReduce

Youngseok Lee
Chungnam National University
Daejeon, 305-764, Republic of Korea
lee@cnu.ac.kr

Wonchul Kang
Chungnam National University
Daejeon, 305-764, Republic of Korea
teshi85@cnu.ac.kr

Hyeongu Son
Chungnam National University
Daejeon, 305-764, Republic of Korea
hgson@cnu.ac.kr

***Abstract*—Internet traffic measurement and analysis have been usually performed on a high performance server that collects and examines packet or flow traces. However, when we monitor a large volume of traffic data for detailed statistics, a long-period or a large-scale network, it is not easy to handle Tera or Peta-byte traffic data with a single server. Common ways to reduce a large volume of continuously monitored traffic data are packet sampling or flow aggregation that results in coarse traffic statistics. As distributed parallel processing schemes have been recently developed due to the cloud computing platform and the cluster filesystem, they could be usefully applied to analyzing big traffic data. Thus, in this paper, we propose an Internet flow analysis method based on the MapReduce software framework of the cloud computing platform for a large-scale network. From the experiments with an open-source MapReduce system, Hadoop, we have verified that the MapReduce-based flow analysis method improves the flow statistics computation time by 72%, when compared with the popular flow data processing tool, flow-tools, on a single host. In addition, we showed that MapReduce-based programs complete the flow analysis job against a single node failure.**

Keywords: MapReduce, Hadoop, cloud computing, NetFlow, traffic monitoring

I. Introduction

In Internet traffic measurement and analysis, flow-based traffic monitoring methods are widely deployed throughout Internet Service Providers (ISPs), because the volume of processed data is reduced and many convenient flow statistics tools are available. Cisco NetFlow [1] is the popular flow monitoring format with which we could easily monitor flows passing through routers or switches without observing every packet. Though routers or switches do not support NetFlow, we could use NetFlow-compatible flow generators such as nProbe [2] to monitor packet streams in flow units. Based on Cisco NetFlow v9, IETF standardizes the flow-based traffic monitoring method at the IP Flow Information eXport (IPFIX) [3] working group. As networks grow and evolve, we have to manage and monitor more and more switches and routers for security, traffic engineering, Quality of Service (QoS), and accounting reasons.

Generally, Internet traffic measurement and analysis are executed on a high performance central server. Popular tools such as tcpdump[4] or Coralreef [5] are usually run on a single host to capture and process packets at a specific monitoring point. Flow analysis tools such as flow-tools [6] or flowscan [7] are widely used to generate traffic statistics with NetFlow data. Typically, ISPs employ a high-performance server with a large storage system to collect and analyze flow data from many routers. However, when we anatomize traffic in a large-scale network, we are often confronted with hard challenges of handling a huge amount of traffic data for processing and management. For example, when ISPs monitor traffic in a nation-wide network consisting of hundreds or thousands of routers capable of exporting Cisco NetFlow data, it is not easy to compute traffic statistics from many large flow files in short time. In order to lessen the volume of continuously streaming flow data, we normally use packet sampling or flow aggregation techniques. Otherwise, after processing packet traces, we leave only the statistics information results.

When we analyze flow data for a large-scale network, we need to handle and manage a few Tera or Peta-byte packet or flow files simultaneously. When the outbreak of global Internet worms or DDoS attacks happens, we also have to process fast a large volume of flow data at once. Yet, with a single-server approach, we are not able to cope efficiently and quickly with a large measurement data for scalable storage and analysis. From recent developments of cluster filesystems and cloud computing platforms, we could benefit two features: distributed parallel computing and fault tolerance. Google, Yahoo, Amazon, and Facebook are rigorously developing or making use of cluster filesystems and cloud computing platforms. Google has first developed the MapReduce [8] programming model for page ranking or web log analysis. MapReduce is a software framework that supports distributed computing with two functions of map and reduce on large data sets on clusters. Google operates thousands of machines for MapReduce to process large web data sets. After Google announced the MapReduce model, Yahoo has released an open-source system for the cloud computing platform, called Hadoop [9], which could process easily very large files with streaming access patterns. Amazon provides the Hadoop-based cloud computing service such as Elastic Compute Cloud (EC2) or Simple Storage Service (S3). Facebook also uses Hadoop to analyze the web log data for its social network service. On the other hand, cloud computing on cluster filesystems provides easy fault-tolerant services for managing a huge amount of large files. Moreover, we could build the Hadoop-based large flow analysis system inexpensively with the commodity hardware. Hence, it is useful to apply distributed parallel computing environments of cluster filesystems and cloud computing systems to the large-scale Internet traffic measurement and analysis application.

In this paper, we propose an Internet flow analysis method on the cloud computing platform. Specifically, we present a MapReduce-based flow analysis scheme that could easily process Tera or Peta-byte flow files collected from many routers or monitoring servers. From experiments on our testbed with four Hadoop data nodes, we achieved that flow statistics computation time for large flow files could dramatically decrease when compared with a popular flow analysis tool run on a single host. In addition, we showed that the MapReduce-based flow analysis program finishes successfully against a single-machine failure.

The remainder of this paper is organized as follows. In Section 2, we describe the related work. Our MapReduce-based flow analysis method is explained in Section 3, and its experimental results are presented in Section 4. Finally Section 5 concludes this paper.

II. Related Work

Flow analysis tools such as flow-tools, flowscan or CoralReef are popular and widely used for generating flow statistics such as port or protocol breakdown, because the port-based traffic classification method is reasonable for well-known Internet applications as shown in [10]. These flow monitoring tools are usually run on a single server with a large storage system such as RAID or Network Attached Storage (NAS). To reduce a large volume of observed flow data, these tools often aggregate flows or save only flow statistics after analyzing flow files. Yet, with these tools, it is not easy to process quickly a few Tera or Peta-byte flow data in a parallel or distributed way.

For analyzing traffic by parallel processing, several solutions have been proposed. Among them, DIPStorage [11] uses a P2P platform, called storage tanks, to process flow data in a parallel way. However, each storage tank associated with a specific flow processing rule might be increasing the computation overhead.

Recently many MapReduce programs have been developed by Google, Yahoo, Amazon, etc. to analyze big data of web search documents, text files or web log. Chen et al. developed a snort [12] log data analysis method with Hadoop [13] for a large-scale network security application. Hive [14] and HBase [15] are often used with Hadoop for easy management of big data. To the best of our knowledge, however, our work is the first study to propose an Internet-scale flow analysis method with MapReduce.

III. MapReduce-based Flow Analysis

A. *Overview*

Figure 1 shows the architecture of our flow measurement and analysis system. The cloud platform provides the cluster filesystem and the cloud computing functions. Flow data from routers are delivered to the cluster through unicasting or anycasting. Cluster nodes are operated by a master node to

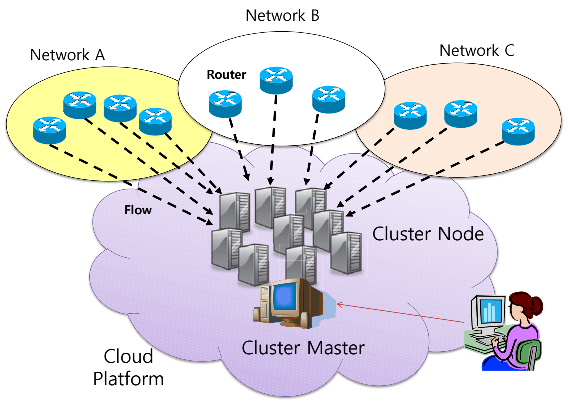

Fig. 1. Architecture of the proposed flow measurement and analysis system

save and process flow data as well as to manage the cluster configuration. When flow data are archived on the cluster filesystem, the MapReduce flow analysis program is run on the cloud platform.

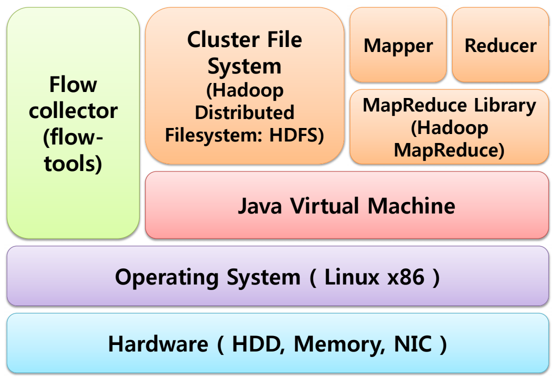
Flow collector (flow-tools)
Cluster File System (Hadoop Distributed Filesystem: HDFS)
Mapper
Reducer
MapReduce Library (Hadoop MapReduce)
Java Virtual Machine
Operating System (Linux x86)
Hardware (HDD, Memory, NIC)

Fig. 2. Functional components of a cluster node

Each cluster node is equipped with a flow collector, a distributed cluster filesystem, and a MapReduce library as shown in Fig. 2. A flow collector receives flow packets, stores them to files, and moves flow files at local disk to the cluster filesystem. NetFlow packets from routers or monitoring servers are usually sent to cluster nodes in unicast. Since NetFlow packets are delivered in UDP, it does not guarantee the reliability. In IPFIX, we could use SCTP instead of UDP for reliability. Anycasting could be used like DNS to provide load balancing with the cluster nodes when receiving NetFlow packets. Then, flow data are saved into files associated with each flow-exporting router periodically (e.g., five minutes), which will be uploaded to the cluster filesystem. For the flow collector, we use flow-tools for the NetFlow collecting and processing tool. Mapper and Reducer will analyze flow data with Hadoop MapReduce library. To meet the customized purpose of flow analysis, we could implement appropriate Mapper and Reducer programs. The distributed filesystem provides easy management of very large files as well as fault-

TABLE I
INPUT FILES

Duration	Flow count (million)	Flow file count	Total binary file size (GB)	Total text file size (GB)
1 day	3.2	228	0.2	1.2
1 week	19.0	1596	0.3	2.3
1 month	109.1	7068	2.0	13.1

tolerant service. For our cloud computing platform, we employ Hadoop that provides open MapReduce software framework and cluster file system on the Java virtual machine (VM). HDFS is suitable for handling very large files with the streaming data access pattern that is a write-once and read-many-times pattern. In HDFS, a name node operates management of the filesystem metadata and provides management and control services, while a data node supplies block storage and retrieval services. A name node at the master will perform recovery and automatic backup of name nodes. The block size (64 MB by default) and the number of replicated blocks in HDFS could be reconfigured according to the fault-tolerance policy.

B. *Flow Analysis Method with MapReduce*

In the MapReduce programming model, the computation takes a set of input key/value pairs, and produces a set of output key/value pairs. Map and Reduce are two basic functions in the MapReduce computation. Users write Map that takes an input pair and produces intermediate key/value pairs. The Hadoop MapReduce library will group the intermediate values according to the same key. Reduce that is also written by users will merge the intermediate values for smaller values.

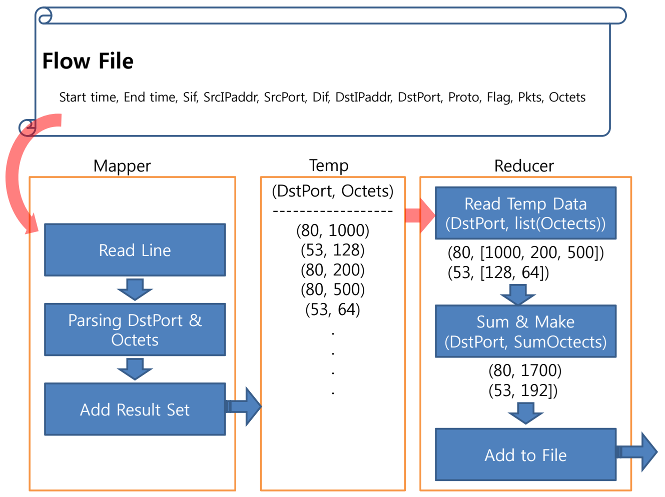

Fig. 3. A MapReduce flow analysis program for destination port breakdown

To implement various flow analysis programs with MapReduce, we have to determine appropriate input key/value pairs for each analysis program. For example, when we analyze traffic by port breakdown, which sums up the octet count for the port number, the key/value pair will be (port, octet). In this work, we have written a simple port-breakdown program for the performance evaluation. With MapReduce, accordingly, we could realize typical flow analysis functions provided by well-known Internet traffic statistics programs. Figure 3 shows the procedure of the MapReduce-based program that performs port-breakdown analysis of flow data.

1) **Input flow files:** At first, after storing flow data from flow probes on the local disk, we move the raw NetFlow v5 files to the cluster filesystem, HDFS. As the current Hadoop mapper supports only text files for the input format, we convert NetFlow files to text-format ones. As the size of text-format flow files is much larger than that of binary-format ones, we need to support binary flow files to the inputs for Mapper. Otherwise, the gzip-compressed text flow files could be used for the input format.
2) **Mapper:** Our flow mapper reads each flow record split by new lines. A flow record has attributes of timestamp, IP addresses, port, protocol, flag, octet count, packet count, and interface numbers. Though we use only the NetFlow v5 flow format in this work, we could extend the supported flow format to NetFlow v9 or IPFIX. After reading a flow record, we filter out necessary flow attributes for a flow analysis job. As shown in Fig. 3, when the flow analysis job sums up octet counts per destination port number, we set key/value pairs as (dst port, octets). The flow map task will write its temporary results on the local disk.
3) **Reducer:** The flow reducer will be called with the inputs as the intermediate values generated by flow mappers. As in the port-breakdown example, a value list of octets belonging to the same destination port number will be summed up. After merging octet values associated with the destination port, the flow reducer writes the octet value for each port number.

IV. Performance Evaluation

A. *Experimental environment*

For the performance evaluation of flow analysis with MapReduce, we built up a small Hadoop testbed consisting of a master node and four data nodes. Each node has quad-core 2.83 GHz CPU, 4 GB memory, and 1.5 TB hard disk. HDFS is used for the cluster filesystem. All Hadoop nodes are connected with 1 Gigabit Ethernet cards.

With flow-tools we collected NetFlow v5 packets sent by a flow generation tool, nProbe, that exports flow data for a Gigabit Ethernet link in our campus network. Then, we saved exported flows on a file every five minutes. As we captured flow data on a small network of a /24 prefix subnet, a five-minute flow file is not enough to assess the performance of MapReduce. Thus, to evaluate the flow statistics computation time for large data sets, we used input flow files collected for one day, one week, and one month in Table I. One-day flow files include about 3.2 million flow records, one-week flow files 19.0 million, and one-month flow files 108.1 million. The binary flow files are used inputs to flow-tools, whereas the text flow files to our MapReduce program.

B. *Flow statistics computation time*

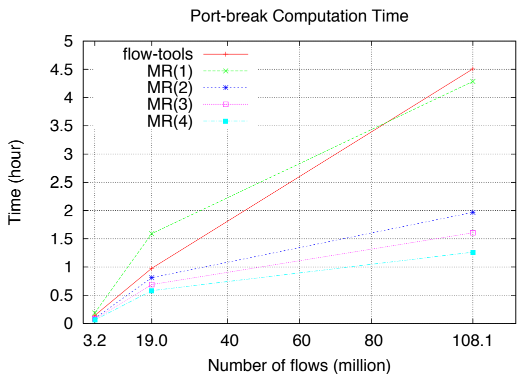

Fig. 4. Destination port breakdown completion time:flow-tools vs. MapReduce

We compare the popular flow statistics program, flow-tools, on a single server with our flow MapReduce program in Fig. 4. The purpose of tested programs is to compute the octet count for each destination port number. We ran "`flow-cat /flowdirectory/ | flow-stat -f 5 > result`" commands of flow-tools to concatenate binary flow files stored in a directory and to calculate the flow statistics for the destination port. Our MapReduce program reads text flow files and produces the octet count for each destination port. To observe the impacts of the number of data nodes on the performance of the MapReduce program, we carried out the experiments with 1, 2, 3, and 4 data nodes.

As shown in Fig. 4, the port-breakdown computation time of flow-tools increases linearly as the number of input flows grows, whereas that of MapReduce does not quickly build up. With a single Hadoop data node, MR(1), the MapReduce program for the input files of 3.2 million and 19.0 million flow records does not outperform flow-tools. However, when we tested the MapReduce program with two or more Hadoop data nodes (MR(2), MR(3), and MR(4)), we obtained that MapReduce reduces dramatically the flow computation time for all input files. Given the input files of 108.1 million flow records, it took 4.5 hours for flow-tools to complete the job, whereas only 1.25 hours for MapReduce with four Hadoop data nodes, MR(4). The flow statistics computation time of MR(4) has decreased by 72% for 108.1 million flow records. From the experiments, we could verify that the MapReduce flow analysis method computes efficiently flow statistics for a big data set, which will be scalable in a large-scale network.

C. *Recovery of a single node failure*

As processes or machines often fail because of software or hardware malfunctions, we need to provide a fault-tolerant flow analysis method against failures to the cloud computing platform. Among several failure scenarios, we have experimented two fault recovery cases for 3.2 million flows as shown in Fig. 5 and Fig. 6 where a single Hadoop data node fails when either a Map or Reduce task is in progress. First, Fig. 5 depicts how the Map task is recovered after a node among four Hadoop data nodes running Map tasks is forced to reboot. A Hadoop data node fails at 4 seconds when the completion percentage of Map tasks is only 9%. Thus, the Map task is re-executed at 266 seconds when it finds that the intermediately-mapped results are not complete even though it has already reached 100%. Second, we can observe that the Reduce task is recovered at 320 seconds after a Hadoop node running Reduce tasks is shutdown at 29 seconds as shown in Fig. 6. When a failure occurs, it takes longer time for MapReduce to complete the task as given in Table II. Under a large data set of 108.2 million flows, however, MapReduce with four data nodes spent only 1.5 times more seconds to complete the job by recovering Map/Reduce failures. Through experiments, we have verified that the flow computation job could successfully finish against a single node failure through the Hadoop fault-tolerant service.

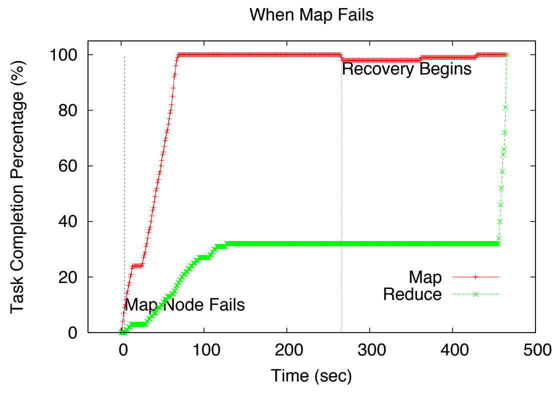

Fig. 5. MapReduce failure recovery: when Map fails

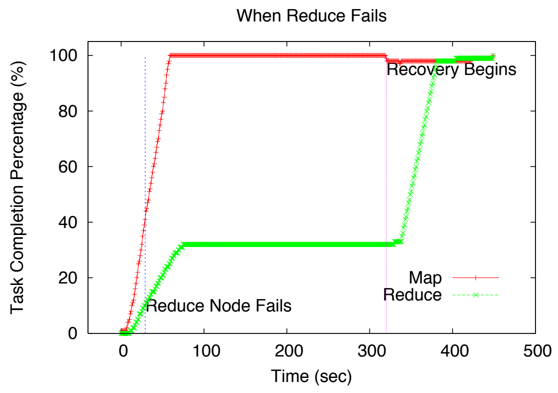

Fig. 6. MapReduce failure recovery: when Reduce fails

TABLE II
TASK COMPLETION TIME WITH OR WITHOUT A SINGLE NODE FAILURE

Flow counts	Time of MR(4) (seconds)	Time of MR(4) with failure (seconds)
3.2 M	220.2	380.2
19.0 M	2096.7	2588.5
108.2 M	4549.1	6754.4

V. CONCLUSION

In this paper, we presented a MapReduce-based flow analysis method for a large-scale networks that could analyze efficiently and quickly big flow data against failures. On the Hadoop system, we have evaluated the performance of the MapReduce-based flow analysis method by developing a port-breakdown program. From the experiments with four Hadoop data nodes, we achieved that flow computation time could be dramatically improved by 72% compared with the typical flow analysis tools. In addition, we showed that the fault-tolerant service against a single machine failure could be easily provided by MapReduce-based flow analysis. Though our MapReduce-based flow analysis scheme outperforms the legacy single-host tools, we need to improve a few drawbacks of the current MapReduce-based approach such as batch processing jobs or text input file formats, and to develop convenient flow analysis tools based on MapReduce.

ACKNOWLEDGMENT

This work was partly supported by the MKE(Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency) (NIPA-2010-(C1090-0902-0016)), and partly by the IT R&D program of MKE/KEIT [KI001878, CASFI(Collect, Analyze, and Share for Future Internet): High-Precision Measurement and Analysis Research].

REFERENCES

[1] Cisco NetFlow, http://www.cisco.com/web/go/netflow.
[2] L. Deri, *nProbe: an Open Source NetFlow Probe for Gigabit Networks*, TERENA Networking Conference, May 2003.
[3] J. Quittek, T. Zseby, B. Claise, and S. Zander, *Requirements for IP Flow Information Export (IPFIX)*, IETF RFC 3917, October 2004.
[4] tcpdump, http://www.tcpdump.org.
[5] CAIDA CoralReef Software Suite, http://www.caida.org/tools/measurement/co alreef.
[6] M. Fullmer and S. Romig, *The OSU Flow-tools Package and Cisco NetFlow Logs*, USENIX LISA, 2000.
[7] D. Plonka, *FlowScan: a Network Traffic Flow Reporting and Visualizing Tool*, USENIX Conference on System Administration, 2000.
[8] J. Dean and S. Ghemawat, *MapReduce: Simplified Data Processing on Large Cluster*, OSDI, 2004.
[9] Hadoop, http://hadoop.apache.org/.
[10] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee *Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices*, ACM CoNEXT, 2008.
[11] C. Morariu, T. Kramis, B. Stiller *DIPStorage: Distributed Architecture for Storage of IP Flow Records.*, 16thWorkshop on Local and Metropolitan Area Networks, September 2008.
[12] M. Roesch, *Snort - Lightweight Intrusion Detection for Networks*, USENIX LISA, 1999.
[13] W. Chen and J. Wang, *Building a Cloud Computing Analysis System for Intrusion Detection System*, CloudSlam 2009.
[14] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, Raghotham Murthy *Hive: a warehousing solution over a map-reduce framework.*, Proceedings of the VLDB Endowment Volume 2 , Issue 2 (August 2009) Pages: 1626-1629
[15] HBase, http://hadoop.apache.org/hbase/.