TITLE

A Dissertation Submitted in partial fulfillment of the degree of

Master of Technology
in
Computer Science

By
STUDENT NAME

School of Computer and Information Sciences
University of Hyderabad, Gachibowli
Hyderabad - 500046, India

Month, Year

CERTIFICATE

This is to certify that the dissertation entitled "**TITLE**" submitted by **STUDENT NAME** bearing Reg. No. XXXXXXX, in partial fulfillment of the requirements for the award of Master of Technology in Computer Science, is a bona fide work carried out by him/her under my/our supervision and guidance.

The dissertation has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

FACULTY NAME
School of Computer and Information Sciences,
University of Hyderabad

Dean,
School of Computer and Information Sciences,
University of Hyderabad

DECLARATION

I, **NAME OF THE STUDENT** hereby declare that this dissertation entitled "**TITLE**" submitted by me under the guidance and supervision of Professor/Dr. FACULTY NAME is a bona fide work. I also declare that it has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Date:

STUDENT NAME
Reg. No.: XXXXXXXX

Signature of the Student

To,

My Parents and Supervisor.

Contents

List of Figures

List of Tables

Chapter 1

Related Work

Flow monitoring protocols like NetFlow[4] and sFlow[8] can provide important information about traffic that passes through a network. However, contemporary networking with its 10Gbps and higher NICs is outpacing the ability to monitor them efficiently. As data centers are getting virtualized with virtual software switches and scaling to thousands of nodes, it is an immediate requirement to have monitoring systems that can scale effectively. There are not many solutions proposed that provide scalable flow monitoring in data center networks. In this chapter, we review some of the literature that deals with scalable flow monitoring in data center networks.

1.1 Edge Monitoring and Collection for Cloud (EMC2)[12]

Edge Monitoring and Collection for Cloud (EMC2) is a scalable network-wide monitoring service for data centers. EMC2 stays inside the host computer to monitor virtual switches. Monitoring at virtual switch is scalable due to the distributed nature of the storage of the collected information.

1.1.1 Architecture

Figure 1.1 shows the architecture of EMC2.

EMC2 is a multi-threaded application that contains the following modules:

1. Flow-Table : Flow-Table is an in-memory 2-level hash table.

2. NetFlowParser : It parses the NetFlow datagrams and updates the Flow-Table.

3. sFlowParser : It parses the sFlow datagrams and updates the Flow-Table.

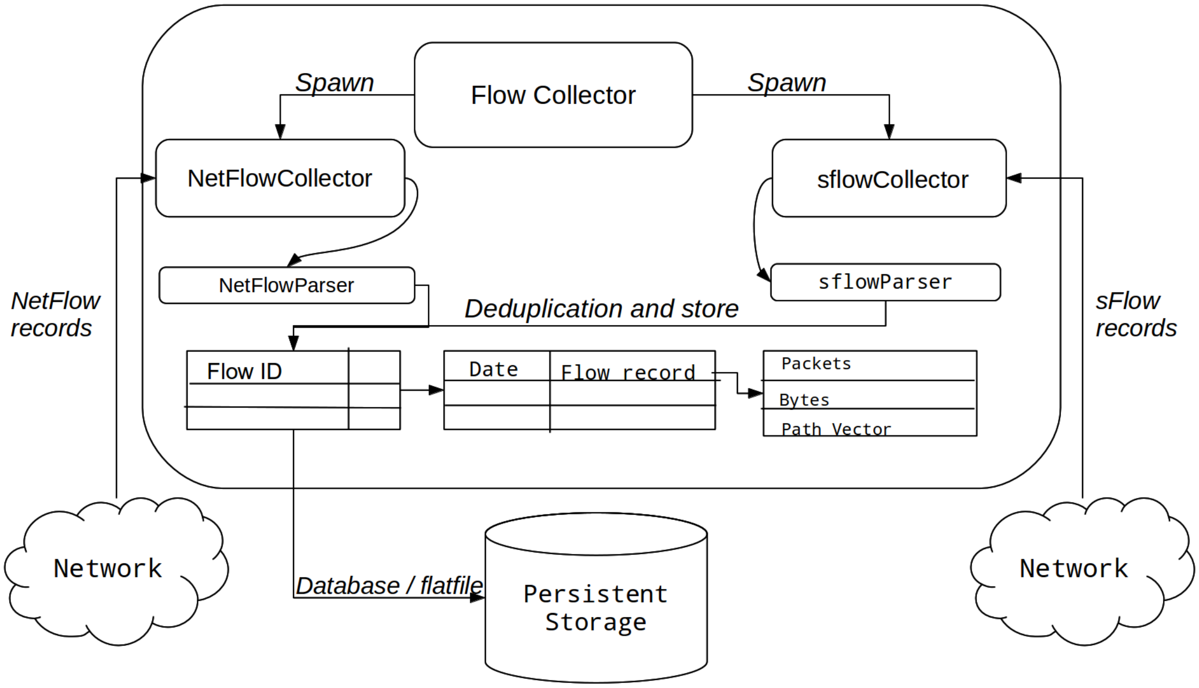

Figure 1.1: Architecture of EMC2.

4. NetFlowCollector : It accepts the NetFlow datagrams and creates the parser thread upon receiving NetFlow datagrams.

5. sFlowCollector : It does the same task like the NetFlowCollector but for sFlow datagrams.

6. FlowCollector : It invokes two thread – NetFlowCollector and sFlowCollector – for accepting flow datagrams.

1.1.1.1 Flow-Table

Flow-Table is a 2-level in-memory hash table. The primary key for the hash table is the Flow ID which is formed based on the layer 3 source and destination addresses. Flow ID maps to another hash table where timestamp is the key and flow record is the value. Each flow record contains number of packets, number of bytes and optional path vector.

1.1.1.2 NetFlowParser/sFlowParser

NetFlowCollector and sFlowCollector create these two parser threads upon receiving a NetFlow/sFlow datagram respectively. These parser threads parse the datagram and update the Flow-Table. They also perform two important tasks:

1. Deduplication.

2. Data rate prediction in presence of sampling.

Deduplication: Deduplication prevents duplicate flow records from being added to the Flow-Table. It uses the following algorithm.

Algorithm 1: Detect Duplicate Flow

```
if $flow - ID$ not exist then
  add flow to the flow table.
  return
else
  if Same exporter then
    update the flow table.
    return
  else
    report duplicate flow.
    update path vector.
    return
  end if
end if
```

Data Rate Prediction in Presence of Sampling: Sampling rate is specified in flow datagrams. Parser thread predicts the data rate by multiplying sampling rate with the length of the packet.

1.1.1.3 NetFlowCollector/sFlowCollector

NetFlowCollector/sFlowCollector are collector thread that wait for new NetFlow or sFlow datagrams and spawn a new NetFlowParser/sFlowParser thread upon receiving a datagram.

1.1.2 Advantages and Limitations

The authors state the following advantages of EMC2:

- Scalable monitoring as EMC2 monitors host *vswitches* in a distributed fashion by storing the information as flat files in those hosts itself instead of sending them to a centralized collector.

However, flat files are not really built for scalability unlike many other distributed databases available today such as Cassandra, Big Table etc.. Therefore, it is not clear how much of performance can be obtained by storing the information in flat files in the host itself. Considering that most data centers work with SANs rather than local disks, this may not be as scalable as claimed.

1.2 Scalable Internet Traffic Measurement and Analysis with Hadoop[11]

Hadoop[1] is a distributed computing platform that uses distributed file system(HDFS) and MapReduce[5] programming model. Hadoop cluster consists of commodity hardware that can scale thousands of nodes to store huge amount of data. It can perform massive data analytics operations on the available data using MapReduce. Storing NetFlow data on Hadoop and analysis using MapReduce offers scalable traffic measurement and analysis.

1.2.1 Architecture

Traffic measurement and analysis system with Hadoop consists of following modules

1. Traffic collector.

2. IP packet and NetFlow reader in Hadoop.

3. Analysis in MapReduce.

4. Interactive query interface with Hive[3].

1.2.1.1 Traffic collector

Traffic collection is done by high-speed packet capture driver and load balancer. Load balancer forwards packets into multiple Hadoop DataNodes evenly. Traffic collector also can read trace files stored on the disk and writes them into HDFS. Trace files contains Netflow packets or IP packets in *libpcap* format.

1.2.1.2 IP packet and NetFlow reader in Hadoop

Storing binary trace file into Hadoop specific sequence file needs more computation power as every packet has to be sequentially read from the trace file and stored into HDFS. Authors has developed new Hadoop APIs to store trace file directly into HDFS. As there is no distinct mark to find out end of a packet records in *libpcap* format, authors proposed a heuristic algorithm using timestamp-based bit pattern.

Timestamp-based heuristic algorithm using MapReduce to identify packet records: MapReduce job invokes multiple map tasks to process each HDFS block in parallel. Each of the map tasks follow these steps to identify packet records in HDFS block:

1. Read two records using *libpcap* 16-byte header.
2. Check timestamp value, that should be within duration of captured time.
3. Difference between wired packet length and captured length should be less then maximum packet length.
4. Check whether timestamp difference between two packet records within the define threshold.

1.2.1.3 Analysis in MapReduce

Authors has implemented MapReduce algorithms for processing IP packet as well as NetFlow packets. Here is the list of analysis tools

1. IP Layer analysis provide following analysis jobs
    (a) Host and port count statistics.
    (b) Periodic flow statistics.
    (c) Periodic sample traffic statistics.

2. TCP layer analysis compute following statistics
   (a) RTT.
   (b) Retransmission.
   (c) Throughput.
3. NetFlow analysis provides
   (a) Human readable flow statistics.
   (b) Aggregated flow statistics.
   (c) Top N flows sorted by key such as packet count or byte count.

1.2.1.4 Interactive query interface with Hive

Hive is a data warehousing system build on the top of Hadoop that allows to generate MapReduce jobs using SQl like query. IP analysis MapReduce jobs process NetFlow packets on HDFS and store flow record and IP statistics into Hive tables. A user can query on Hive tables using interactive web-based user interface.

1.2.2 Advantages and Limitations

Advantages of Hadoop based traffic measurement and analysis are

1. Scalable storage.
2. MapReduce operations on flow data.

Disadvantages of Hadoop based traffic measurement and analysis are

1. Low response time- "the fasted MapReduce job takes 15+ seconds"[**?**].
2. Hadoop NameNode was a single point of failure which solved in later version of Hadoop 2.0.0 with passive NameNodes.
3. Multiple NameNodes require to get high availability[2].

1.3 *nfdump[6]*

nfdump provides set of tools to capture, analysis NetFlow packets. These set of tools are

1. *nfcapd*

2. *nfdump*

3. NfSen

nfcapd reads data from the network and stores them into disk. It also rotate the file to limit size of file. *nfdump* allows *tcpdump* style filter expression for processing NetFlow data stored by *nfcapd* and display them on terminal or write into file. NfSen gives graphical overview of NetFlow data using RRDTool[7];

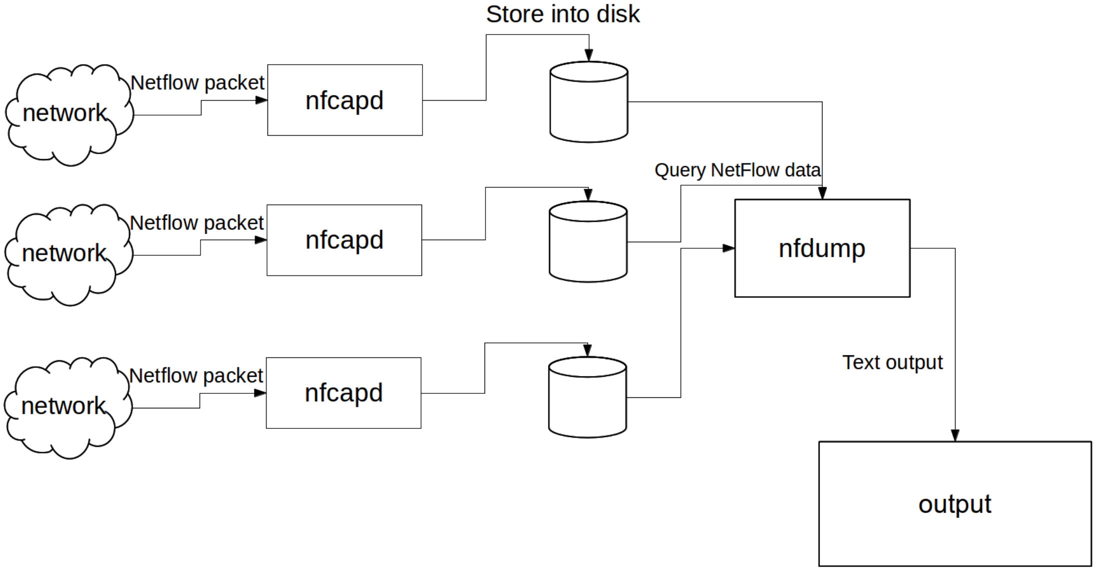

Figure 1.2: Architecture of *nfdump*.

1.3.1 Advantages and Limitations

nfdump is suitable for small scale NetFlow analysis. In case of large scale NetFlow monitoring disk file based storage is not sufficient.

1.4 Summary

In this section I have discussed available solutions for scalable flow monitoring. The issues with current solutions are

1. Lack of scalable storage.[12][6].
2. Lack of high availability data storage [11].
3. Real-time flow analysis is difficult with current systems. [11]

In the next chapter I describe design and implementation of modified *ntop*(a NetFlow collector) that tries to solve few of the issues.

Chapter 2

Cassandra plugin for *ntop*

2.1 Overview

In this chapter I describe about *ntop*, Cassandra and implementation of Cassandra plugin for *ntop*. I also specify design and implementation of Cassandra client for C developed using C-python API. I have done few testing of my solution that I describe in results section.

2.2 *ntop*

ntop is open-source traffic measurement application written in C. *ntop* design follow UNIX philosophy: applications can be divided into small independent pieces that co-operate to achieve a common goal. *ntop* has following module

1. Packet Sniffer - Capture packet using *libpcap* library and also from UNIX sockets.

2. Packet Analyzer - Analysis packet captured by Packet Sniffer.

3. Traffic Rules - *ntop* allows traffic rules for capturing packets to filter out unnecessary packets.

4. Report Engine - Report Engine display analyzed output in an interactive web-based user interface.

5. Plugins - Using plugins anyone can extend *ntop* to support extra features.

2.2.1 Packet Sniffer

Packet Sniffer captures packet using *libpcap* library and store them into internal buffer, that helps to reduce packet drops in a busty traffic environment. *libpcap* is supported by all major Operating Systems, that allows *ntop* to be portable to windows and UNIX variants.

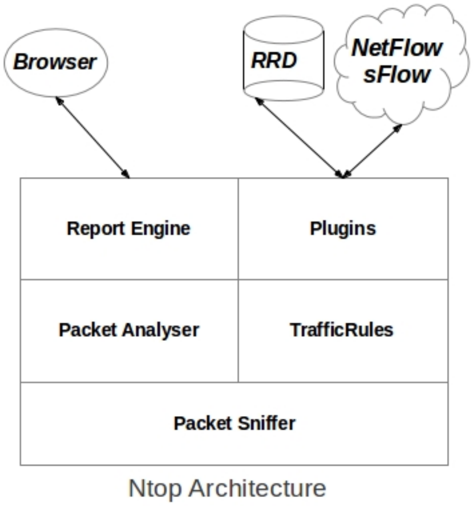

Figure 2.1: Architecture of *ntop*.

2.2.2 Packet Analyzer

Packet Analyzer gets packets from Packet Sniffer and process those packets. Sniffed packets contained information about status of network and that information calculated by Packet Analyzer then stored in RRD for future references.

2.2.3 Traffic Rules

ntop allows user to specify what kind traffic a user is interested. Using *libpcap* filter expression *ntop* achieve this goal. Traffic Rules helps *ntop* to reduce some burden on memory as well CPU, makes *ntop* more faster as it process less amount of packets.

2.2.4 Report Engine

ntop contains a web-server by which user from any geological location can monitor their network. Report Engine provides a beautiful user interface with time series graph drawn using RRDTool. Using Report Engine a user can change behavior of *ntop* by changing its configuration parameters.

2.2.5 Plugins

ntop has flexible design that allows user to add their own plugins. At startup *ntop* searches shared libraries (like .so, .dll files) to load plugins. A plugin can access *ntop*'s global data structure and can use API exported by *ntop*.

2.3 Cassandra

Cassandra is a highly scalable and highly available database initially developed by Facebook using two famous approach GFS[10] from Google and Dynamo[9] from Amazon. Cassandra currently highly used in ebay and Netflix. Cassandra's big data features are

1. Elastic scalability.
2. High availability.
3. Distributed database design with no single point of failure.
4. Blistering linear performance.
5. Multiple datacenter based data distribution.

Why Cassandra: In one of our testing we are able to generate approx 1000 NetFlow packets per second with only two node of 1 Gbps NIC card. Bellow table provides amount of data generated by our testbed.

Packets generated	Time	Size of generated data height
1000	1 second	1.4MB
60 k	1 minute	85 MB
3.6 m	1 hour	5 GB

Bibliography

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

[2] **Apache Hadoop and Hbase**. http://www.slideshare.net/cloudera/sf-nosql2011/58.

[3] **Apache Hive**. http://hive.apache.org/.

[4] **Cisco IOS NetFlow**. http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html.

[5] **Mapreduce**. http://en.wikipedia.org/wiki/MapReduce.

[6] **nfdump Home Page**. http://nfdump.sourceforge.net/.

[7] **RRDTool Home Page**. http://oss.oetiker.ch/rrdtool/.

[8] **sFlow Official Site**. http://www.sflow.org.

[9] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. **Dynamo: amazon's highly available key-value store**. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, SOSP '07, pages 205–220, New York, NY, USA, 2007. ACM.

[10] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. **The Google file system**. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 29–43, New York, NY, USA, 2003. ACM.

[11] Yeonhee Lee and Youngseok Lee. **Toward scalable internet traffic measurement and analysis with Hadoop**. *SIGCOMM Comput. Commun. Rev.*, **43**(1):5–13, January 2012.

[12] Vijay Mann, Anilkumar Vishnoi, and Sarvesh Bidkar. **Living on the edge: Monitoring network flows at the edge in cloud data centers**. In *Fifth International Conference on Communication Systems and Networks, COMSNETS 2013*, 2013.