A Comparative Survey Based on Processing Network Traffic Data Using Hadoop Pig and Typical Mapreduce

Anjali P P and Binu A

Department of Information Technology, Rajagiri School of Engineering and Technology, Kochi. M G University, Kerala

ABSTRACT

Big data analysis has now become an integral part of many computational and statistical departments. Analysis of peta-byte scale of data is having an enhanced importance in the present day scenario. Big data manipulation is now considered as a key area of research in the field of data analytics and novel techniques are being evolved day by day. Thousands of transaction requests are being processed in every minute by different websites related to e-commerce, shopping carts and online banking. Here comes the need of network traffic and weblog analysis for which Hadoop comes as a suggested solution. It can efficiently process the Netflow data collected from routers, switches or even from website access logs at fixed intervals.

KEYWORDS

Big Data, MapReduce, Hadoop, Pig, netflow, network traffic.

1. INTRODUCTION

Netflow is the network protocol designed by Cisco Systems for assembling network traffic information and has become an industry standard for monitoring netfwork traffic. Netflow is widely supported by various platforms and Cisco's standard NetFlow version 5 defines network flow as a unidirectional sequence of packets which has got seven parameters:

- Ingress interface (SNMP ifIndex)
- Source IP address
- Destination IP address
- IP protocol
- Source port for UDP or TCP, 0 for other protocols
- Destination port for UDP or TCP, type and code for ICMP, or 0 for other protocols
- IP Type of Service

Netflow is normally enabled on a per-interface basis for limiting the amount of exported flow records and the load on the router components. By default, it will capture all packets received by an ingress interface and IP filters can be used to decide if a packet can be captured by NetFlow. Once accurately configured, it can even allow the observation and recording of IP packets on the egress interface. Custom settings can also be done on the Netflow implementation according to the requirements in a variety of practical scenarios. NetFlow data from dedicated probes can be efficiently utilized for the observation of critical links, whereas the data from routers can provide

DOI : 10.5121/ijcses.2014.5101

a network-wide view of the traffic that which will be of great use for capacity planning, accounting, performance monitoring, and security.

Netflow data collected from dedicated probes can be effectively utilized for observing critical links. Data from routers can provide a network-wide view of the traffic and can be made useful in the fields like performance monitoring, capacity planning, accounting and security. This paper comprises of a survey on network traffic analysis using Apache Hadoop MapReduce framework and a comparative study of typical MapReduce against Hadoop Pig which is a platform for analyzing large data sets. It also provides an ease of programming, thereby optimizing opportunities and enhancing extensibility.

2. NETFLOW ANALYSIS USING TYPICAL HADOOP MAPREDUCE FRAMEWORK

The Netflow processing requires sufficient network flow data collected at a particular time or at regular intervals. By analyzing the incoming and outgoing network traffic corresponding to various hosts, required precautions can be taken regarding server security, firewall setup, packet filtering etc. Hence, the significance of network flow analysis cannot be kept aside. Netflow data can be collected from switches or routers using router-console commands and are then filtered on the basis of parameters such as source and destination IP addresses, corresponding Port numbers etc. Similarly when considering websites, every website will generate access logs and such files can be collected from the web-server for further analysis. Such logs collected will contain the details of accessed URLs, the time of access, total number of hits etc which are inevitable for a website manager for the proactive detection and encounter of problems like DDoS attacks and IP address spoofing.

In a distributed environment, Hadoop can be considered as an efficient mechanism for big data manipulation. Hadoop was developed by Apache foundation and is well-known for its scalability as well as efficiency. It works on a computational paradigm called MapReduce which is characterized by a Map and a Reduce phase. Map phase involves division of the computation operation into many fragments and its distribution among the cluster nodes. Individual results evolved at the end of the Map phase will be eventually combined and reduced to a single output in the Reduce phase, thereby producing the result in a very short span of time.

The log files generated every minute by a web-server or router will accumulate to form a large-sized file which needs a great deal of time to get processed. In such an instance, the advantage of Hadoop MapReduce paradigm can be made use of. This big data file can be fed as the input for a MapReduce algorithm which provides an efficiently processed output file in return. Every map function transforms the input data into a certain number of key/value pairs and later sorts them according to the key values whereas a reduce function combines the results with similar key values to produce a single output.

Hadoop stores the data in its distributed file system popularly known as HDFS and will make the requested data easily available to the users. HDFS is highly scalable and advantageous in its portability. It enhances reliability by the mechanism of data replication across multiple nodes in the cluster, thereby eliminating the need for RAID storage on the cluster nodes. High availability can also be incorporated in Hadoop clusters by means of a backup provision for the master node[name node] which is generally known as secondary namenode. The common limitations of using HDFS are the inability to be mounted directly by an existing operating system and the inconvenience of moving data in and out of the HDFS before and after the execution of a job.

Once Hadoop is deployed on a cluster, instances of these processes will be started – Namenode, Datanode, Jobtracker, Tasktracker. Hadoop can process large amount of data such as website logs, network traffic logs obtained from switches etc. The namenode, otherwise known as the master node is the main component of the HDFS which stores the directory tree of the file system and keeps track of the data during execution. The datanode stores the data in HDFS as suggested by its name. The jobtracker delegates the MapReduce tasks to different cluster nodes and the tasktracker spawns and executes the job on the member nodes.

One of the key features of Hadoop MapReduce is its coding complexity and can be accomplished only by programmers with highly developed programming skills. The MapReduce programming model is highly restrictive and Hadoop cluster management also becomes difficult while considering aspects like log collection, debugging and distribution of software distribution. To be precise, the effort to be invested in MapReduce programming restricts its acceptability to a considerable extent. Here arises the need for a platform which can provide ease of programming along with an enhanced extensibility and Apache Pig reveals itself as a novel, but realistic approach in data analytics. The major features and applications of Apache Pig deployed over Hadoop framework is described in Section 3.

3. USING APACHE PIG TO PROCESS NETWORK FLOW DATA

Apache Pig is a platform that supports substantial parallelization which enables the processing of huge data sets. The structure of Pig can be described in two layers – An infrastructure layer with an in-built compiler that can spawn MapReduce programs and a language layer which consists of a text-processing language called “Pig Latin”. Pig makes data analysis and programming easier and understandable even to novices in the Hadoop framework. It uses Pig Latin language which can be considered as a parallel data flow language. The Pig setup has also got provisions for further optimizations and user defined functionalities. Pig Latin queries are compiled into MapReduce jobs and are executed in distributed Hadoop cluster environments. Pig Latin looks similar to SQL [Structured Query Language] but is designed for Hadoop data processing environment and analogous to SQL in RDBMS environment. Another member of Hadoop ecosystem known as Hive is a query language like SQL and it needs data to be loaded into tables before processing. Unlike Hive, Pig Latin can work on schema-less or inconsistent environments and can operate on the available input data as soon as it is loaded into the HDFS. Pig closely resembles scripting languages like Perl and Python in the aspects such as flexibility in syntax and dynamic variables definitions. Hence, Pig Latin can be considered efficient enough to be the native parallel processing language for distributed systems such as Hadoop.

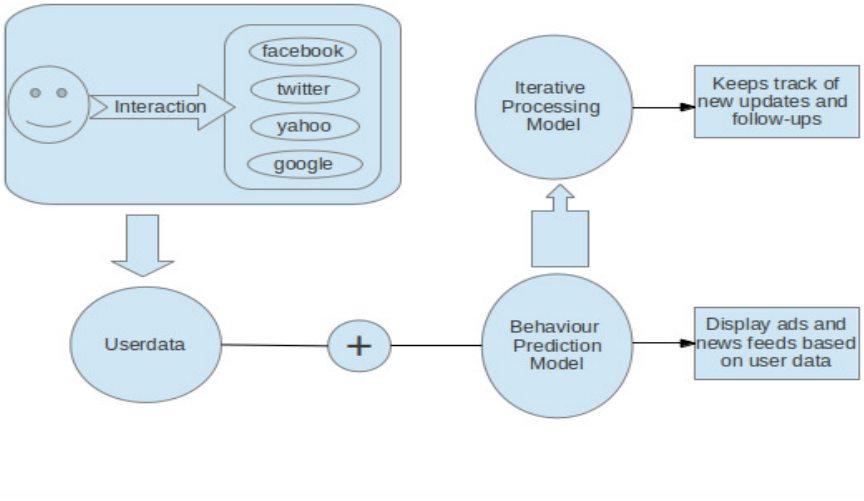

Figure 1. Data processing model in PIG

It is clear that Pig is gaining popularity in different zones of computational and statistical departments. Along with that, Apache Pig is widely used in processing weblogs and wiping off corrupted data from records. It can build behaviour prediction models based on the user interactions with a website and this speciality can be applied in displaying ads and news stories that keeps the user entertained while visiting a website. It also generates an iterative processing model and can keep track of every new updates by joining the behavioural model with the user data. This feature is widely used in social-networking sites like Facebook and micro-blogging sites like Twitter. Pig cannot be considered as effective as MapReduce when it comes to processing small data or scanning multiple records in random order. The data processing models framed by Hadoop Pig is depicted in Figure 1.

In this paper, we are exploring the scope of Pig Latin for processing the netflow data obtained from Cisco routers. The proposed method for using Pig for netflow analysis is described in Section 4.

4. IMPLEMENTATION ASPECTS

Here we present the NetFlow analysis using Hadoop, which is widely used in managing large volume of data, thereby employing parallel processing and comes up with required output in no time. A MapReduce algorithm needs to be deployed to obtain the required result and coding such MapReduce programs for analyzing such a huge amount data is a time consuming task. Here the Apache-Pig will help us to save a good deal of time and also making it possible to be deployed in a real-time scenario.

We set our system with Java platform and the version requirement is above 1.6. The system implemented is completely open source and the Java Development Kit can be downloaded and installed freely. The key factor to be taken care of is to set the proper environment variables for Java so that the base framework can work properly.

Hadoop is a system that makes use of Java framework for its functioning and we're deploying Hadoop on top of Java. Hadoop downloadable packages are also open source and needs to be

extracted and configured properly. The following files need to be tweaked for setting the required clustering as well as replication parameters for the whole system:

- hadoop-env.sh: Set the java directory path.
- core-site.xml: Regarding hadoop name node, port and temporary directories.
- hdfs-site.xml: Regarding parameters like replication values.
- mapred-site.xml: Regarding the host and port of mapreduce job tracker.
- masters: Details of a secondary name node, if any. (Here, it is not needed).
- slaves: Details of the slave nodes, if any.

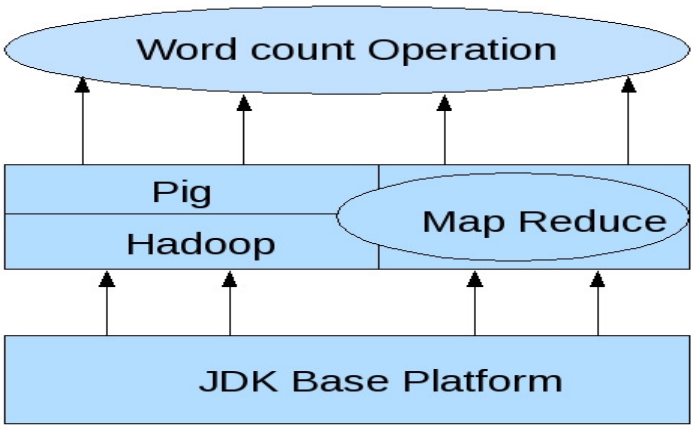

Figure 2. System setup

The hadoop processes should be started after the deployment and we can test the processes by using a command called 'jps' which lists all the Hadoop processes in the system. It should list a Namenode, Datanode, Jobtracker, Tasktracker to ensure that the processes are all working fine and the Hadoop deployment is fine. Testing can be done by creating and deleting directories as well as moving files in and out of HDFS.

Now we need the Pig system for out test setup and we need to deploy that by downloading the opensource Pig tarballs and properly setting its environment variables. The structure of our system setup is given in figure 2.

We need to analyze traffic at a particular time where Pig plays the lead role. For data analysis, we buckets are created with pairs of (SrcIF, SrcIPaddress), (DstIF,DstIPaddress), (Protocal,FlowperSec). After that, they are joined and categorized as SrcIF and FlowperSec; SrcIPaddress and FlowperSec. Netflow data is collected from a busy Cisco switch using tools such as nfdump. The output contains three basic sections- IP Packet section, Source Destination packets section and Protocol Flow section.

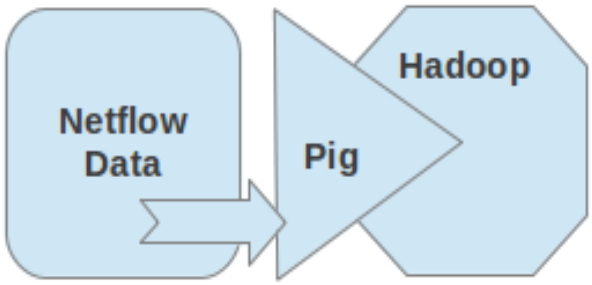

Figure 3. Netflow Analysis using Hadoop Pig

These three sections are created as three different schemas as Pig needs the netflow datasets in arranged format which is then converted to SQL-like schema. Hence. A sample Pig Latin code is given as follows:

Protocol = LOAD 'NetFlow-Data1' AS (protocol:chararray, flow:int, …)
Source = LOAD 'NetFlow-Data2' AS (SrcIF:chararray, SrcIPAddress:chararray, …)
Destination = LOAD 'NetFlow-Data3' AS (DstIF:chararray, DstIPaddress, …)

As shown in Figure 3, the network flow data is deduced into schemas by Pig and later these schemas are combined for analysis to deliver the traffic flow per node as the resultant output.

5. ADVANTAGES OF USING HADOOP PIG OVER MAPREDUCE

The member of Apache Hadoop ecosystem popularly known as Pig can be considered as a procedural version of SQL. The most attractive feature and advantage of Pig is its simplicity itself. It doesn't demand highly skilled Java professionals for its implementation. The flexibility in syntax as well as dynamic variable allocation makes it much more user-friendly. Added extensibility for the framework is provided by the provisions for user-defined functions. The ability of Pig to process un-structured data greatly accounts to its parallel data processing capability. A remarkable difference comes when the computational complexity is considered along with these aspects. Typical map-reduce programs will have large number of java code lines, but a similar task can be accomplished using Pig Latin scripts in very fewer lines of code. Huge network flow data input files can be processed by writing simple programs in Pig Latin which closely resembles scripting languages like Perl and Python. In other words, the computational complexity of Pig Latin scripts are much lesser than similar MapReduce programs written in Java and this particular property makes Pig suitable for the implementation of this project. The procedures and experiments followed for arriving at such a conclusion is described in Section 6.

6. EXPERIMENTS AND RESULTS

Apache Pig was selected as the programming language for Netflow analysis after conducting a short test on sample input files with different sizes.

6.1. Experimental Setup

Apache Hadoop was installed and configured on a stand-alone Ubuntu node initially and Hadoop Pig was deployed on top of this system. SSH Key-based authentication was setup on the machine for convenience as Hadoop prompts for passwords while performing operations. Hadoop instances such as namenode, datanode, jobtracker and tasktracker are started before running Pig and it shows a "grunt" console upon starting. The grunt console is the command prompt specific to Pig and is used for data and file system manipulation. Pig can work both in local mode and MapReduce mode which can be selected as per the user requirement.

A sample Hadoop MapReduce word-count code programmed with Java was used for the comparison. The typical MapReduce program used to process a sample input file of enough size had one hundred thirty lines of code and its time of execution was noted. The same operation coded using Pig Latin contained only five lines of code and corresponding running time for the PigLatin code was also noted for framing the results.

6.2. Experiment Results

At a particular instant, the size of input file is maintained as a constant for MapReduce and Pig whereas the lines of code are more for the former and less for the latter. For the sake of a good comparison, the input file size was taken in the range of 13 kb to 208 kb and corresponding execution time is recorded as in Figure 4.

Turn	Operation	Input File Size(kb)	Time Taken by MapReduce (Lines of Code : 130)	Time Taken by Pig (Lines of Code : 5)
1	Word-Count	13 kb	34 sec	31 sec
2	Word-Count	26 kb	33 sec	31 sec
3	Word-Count	52 kb	35 sec	31 sec
4	Word-Count	104 kb	35 sec	31 sec
5	Word-Count	208 kb	36 sec	31 sec

Figure 4. Hadoop Pig vs. MapReduce

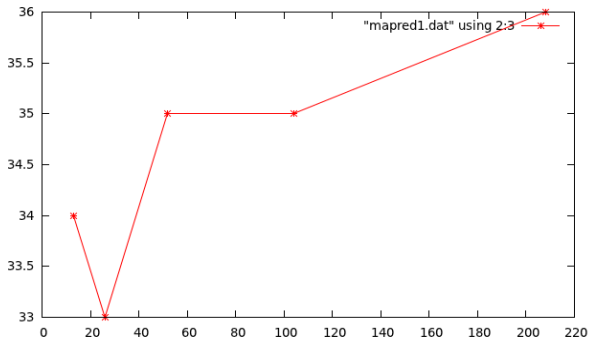

Figure 5. Input file-size vs. execution time for MapReduce

From this we can derive an expression as follows: As the input file size increases in the multiples of x, the execution time for typical MapReduce also increases proportionally whereas Hadoop Pig maintains a constant time at least for x upto 5 times. Graphs are plotted between the input file size and execution time for both cases and are given as Figure 5 and Figure 6.

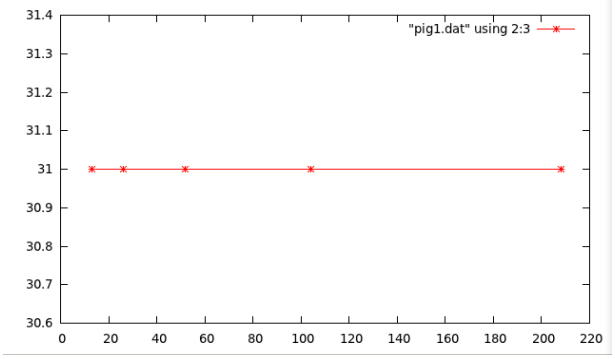

Figure 6. Input file-size vs. execution time for Pig

7. CONCLUSION

In this paper, a detailed survey regarding the wide spread acceptance of Apache Pig platform is conducted. The major features of the network traffic flows are extracted for data analysis using Hadoop Pig for the implementation. Pig was tested and proved to be advantageous in this aspect with a very low computational complexity. The main advantage is that the Network flow capture and flow analysis is done with the help of easily available open-source tools. Upon the implementation of a Pig-based flow analyzer, network traffic flow analysis can be done with increased convenience and easiness even without the help of highly developed programming expertise. It can greatly reduce the time consumed for researching on the complex control loops and programming constructs for implementing a typical MapReduce paradigm using Java, but still enjoys the advantages of MapReduce task division technique by means of the layered architecture and in-built compilers of Apache Pig.

ACKNOWLEDGEMENT

We are greatly indebted to the college management and the faculty members for providing necessary hardware and other facilities along with timely guidance and suggestions for implementing this work.

REFERENCES

[1] An Internet Traffic Analysis Method with MapReduce by Youngseok Lee, Wonchul Kang and Hyeongu Son from Chungnam National University, Daejeon, 305-764, Republic of Korea
[2] http://orzota.com/blog/pig-for-beginners/
[3] http://pig.apache.org/
[4] http://www.hadoop.apache.org
[5] http://jugnu-life.blogspot.com/2012/03/installing-pig-apache-hadoop-pig.html
[6] http://www.hadoop.apache.org
[7] http://www.ibm.com/developerworks/library/l-hadoop-1/
[8] http://www.ibm.com/developerworks/linux/library/l-apachepigdataquery/
[9] http://ankitasblogger.blogspot.in/2011/01/hadoop-cluster-setup.html

[10] http://icanhadoop.blogspot.in/2012/09/configuring-hadoop-is-very-if-you-just.html
[11] http://developer.yahoo.com/hadoop/tutorial/module7.html
[12] https://cyberciti.biz
[13] http://www.quuxlabs.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/
[14] http://www.cisco.com/en/US/docs/ios/ios_xe/netflow/configuration/guide/cfg_nflow_data_expt_xe.html#wp1056663
[15] http://www.cisco.com/en/US/docs/switches/lan/catalyst6500/ios/12.2SX/configuration/guide/netflow.html
[16] http://ofps.oreilly.com/titles/9781449302641/intro.html

Authors

Anjali P P is an MTech student of Information Technology Department in Rajagiri School of Engineering and Technology under Mahathma Gandhi University, who is specializing in Network Engineering. She has received BTech degree in Computer Science and Engineering from Cochin University of Science and Technology. Her industrial experience comes around 2 years in Systems Engineering, Networking and Server Administration. Presently, she is working as a Systems Engineer. Her research interest includes Networking and Big data analysis using Hadoop.

Binu A is working as an Assistant Professor in Department of Information Technology since May 2004. He has one year of industrial experience in Web & Systems Administration. He has graduated (M.Tech. Software Engg.) from Dept. of Computer Science, CUSAT , and took his B.Tech. in Computer Science and Engg. from School of Engineering, CUSAT. At present he is doing PhD in Dept. of Computer Science, CUSAT. His areas of interest include Theoretical Computer Science, Compiler & Architecture.