IOSR Journal of Mobile Computing & Application (IOSR-JMCA)
e-ISSN: 2394-0050, P-ISSN: 2394-0042.Volume 3, Issue 5 (Sep. - Oct. 2016), PP 55-60
www.iosrjournals.org

A Tiered Distributed Network Traffic Analyzer in Big Data using Hadoop Framework

Sharmi Sankar[*], Ishtiaque Mahmood[*], Jehad Al Khalaf Baniyounis[*]
[*]*College of Applied Sciences, Ibri, Postal Code 516, Sultanate of Oman.*

Abstract: *Elevating swiftness on networks with growing traffic dimensions oblige and demand a better computation domain in today's era.The growing volume of distributed network traffic which needs to be analyzed levies new-fangled challenges on the Quality of Service (QoS).The captured traffic assists to comprehend the network's state awareness such as spotting the network abnormalities, refining the QoS of the network and propose better future architectural deliberations. The traffic prediction is quite resource intensive in term of storage and processing. With a predetermined evaluation on the record growth of the traffic to be stored and on ensuring the scalability issue nowadays,administrations move towards Hadoop to obtain better precision in their investigation. A nifty tiered network traffic predictor algorithm, evaluates the critical traffic in detail and helps the network administrators to mitigate the QoS issues by predicting the traffic in detail well in advance. This research paper, confers upon the thoughts and verdicts to be made whilescheming a tiered network traffic predictor version.It comprises of a scheme on Hadoop framework, to detect network behavior and accordingly parsing the pcap file and classifying the parameters, based on a network traffic predictor algorithm that uses the MapReduce methodology implemented on a Hadoop platform which can help the administrators to make decisions and plan early, to overcome the network traffic irregularities and refine QoS issues. In this research, we have used real-world traffic to showcase the tiered predictor benefits to the realm.*
Keywords: *QoS, Hadoop, Map Reduce, predictor.*

I. Introduction

Architectures are basically built on tier-style so as to facilitate different functionalities segregated into sections or segments. The layered style collectively reported as a section is to be located physically on the computer. This tier architecture on Hadoop framework characterizes and decomposes the application based on its role play.Jointly stated as "endpoints", the number of devices linked to every single network infrastructure is rising by a massive rate. The latest revolution has fueled a major problem for the network administrating teams to maintain corporate endpoint security and strategy. The progression of networked devices all over our family unit and organizations, have given ample opportunities for attack to the hackers, hence it is necessary to have the traffic behavior analyzedprior. A deployment of this predictor architecture of this kind facilitates the features such as scalability, readiness and proper resource utilization. Each tier encompasses of layers as more than one functional dependent services are encountered. This traffic flow prediction architecture will be used by network operators to manage a network. The service assured can deliverseamless benefits even after apartialset up. If completelyset up in a network, then this can provisionand guaranteeat most all services. Internet Service Providers always ensures and sets-up guaranteed services in its backbone and offers guaranteed service among clients. Hadoop provides a precise track for being well-equipped, easiness of management, and maximum performance. The methodical performance of Hadoop permits counter tack to afford near real-time response to possible threats, which vividly contracts the gap of exposure for its clients on the network.

II. Related Work

Predictions are highly possible by having a thorough analysis on the independent parameters that influence the dependent. Hadoop is one such massive data analytics facility we ought to use for processing, such dynamically growing data and storage amenities[1]. The elasticity and scalable property of the Hadoop has immensely impressed us to deploy this working infrastructure in our research. Hadoop has been showcased as an open-source computing platform, which has two primary use cases, the Map-reduce and the HDFS (Hadoop Distributed File System)[2]. Despite of using the traffic captured or downloaded from research supportive data stores on the web it can also be created with synthesized traffic to imitate Internet traffic among the subnets and the Internet. The internal traffic inside the local subnet was not simulated. A simplified simulated traffic data has been used by some researchers for preciseness[3].HT Condor aids to manage jobs or tasks on a devoted cluster of PCs and make use of uncommitted resources, for instance idle workstations, below a distributed possession. Thus, it can efficiently yoke the use of all existing computing possessions [4].

III. Methodological facts on the proposed scheme

This division describes various prefaces of the mathematical context cast-off in the algorithm and the methodological facts of the structures used to implement it. It reports on Hadoop and MapReduce, HTCondor, network analyzer a terminal built form of Wireshark referred as Tshark and a classifier using Weka that reduces the number of extracted parameters by embedding the input data into a lower dimension space and finally the results are analyzed using SPSS. These procedures and information are needed for the implementation of the NTP in Big Data using Hadoop framework as depicted in Figure 1.

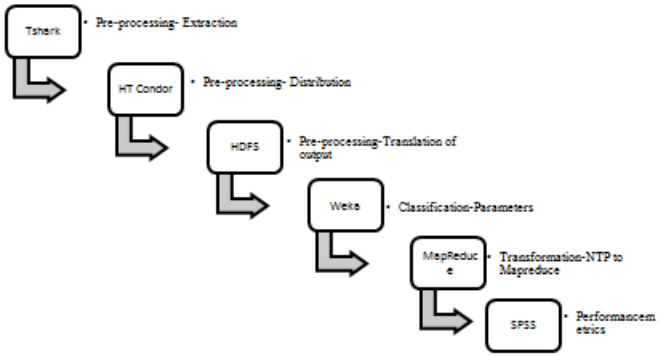

Figure 1. NTP in Big Data using Hadoop framework

IV. Hadoop and MapReduce

Hadoop [5] is a well-established prominent Apache open source platform that offers tools for parallel processing of enormous amount of data using the MapReduce archetype [6]. It can process and handle huge amounts of data in a distributed way on huge clusters in a consistent and fault-tolerant manner [7]. MapReduce is a programming model for processing large datasets which was earlier recommended by Google during 2004 [8].

4.1 HTCondor

HTCondor[9]is an open-source referred as High-Throughput Computing (HTC) software infra-structure for distributed batch tasks. Like other batch systems, HTCondor affords task administration/organizing mechanism that withholds precedence scheme, resource monitoring and management [10]. As a task is rendered to HTCondor, it tracks an available device on the network to execute the task on that device. HTCondor has the ability to identify or spot a machine running an HTCondor job which is no longer available and can drift the task to another idle device which will continue the job exactly from where it had left-off.

4.2 Network Traffic Analyzer & classifier –Tshark, Weka

Tshark is a packet capture tool that does potentsensing and describing features for pcap scrutiny[11]. It captures packet-data as of from an alive network, or inspect packets from an earlier file captured and decodes the form of those packets to the standard output file. TShark's built-in default capture file format is pcap[12]. Weka comprises of tools for data pre-processing, classification, regression, clustering, association and visualization that well-suits for developing new schemes[13].

4.3 Network Traffic Prediction Algorithm

The input dataset X for the NTP is a matrix of size N x n. Each row in the matrix Xrelates to features in vectors, which is referred as data point DP with n mined features. The matrixis prepared subsequently after the preprocessing stage of features extraction, which is achieved from unprocessed pcap files. This algorithm for prediction is to be transformed to the MapReduce jobs in Hadoop as shown below in the job flow diagram for NTP in MapReduce in Figure 2.

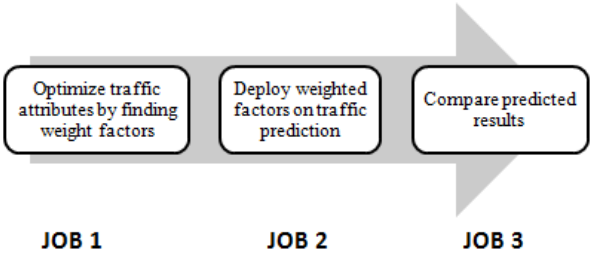

Figure 2. Job flow diagram for NTP in Map Reduce

V. Proposed scheme

The proposed scheme involves a tiered architecture, which encompasses 4 levels of progress to achieve or accomplish the need. The levels are pre-processing, classification, transformation and evaluation. The tiered-architecture flow diagram for the proposed scheme is shown below in Figure 3.

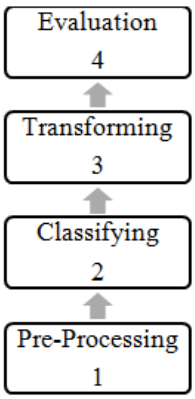

Figure 3. A tiered-architecture flow diagram for the proposed scheme

5.1 Preprocessing stage:

The preprocessing stage is accomplished by the packet analyzer Tshark[14], a version of Wireshark) and by the HTCondor framework. The course of mining features from each pcap file is made by Tshark and the distributed computation by HTCondor. The analysis of each pcap file is allotted as a distinct job by HTCondor[2]. After the parsing of a pcap file is finished, the derived output file is moved to HDFS.The preprocessing stage converts the pcap files into appropriate input format to Hadoop. Each line in the output file from the preprocessing stage signifies a data point in the matrix X. Each line is divided into a key and a value parts separated by a tab. A sample record in matrix X is shown below:
12/12/2015 9:09 80,0,0,64,1655,0,0,64,0,134,0,0,0,695,15,0,0,772,915,32,32.

5.2 Classification

Weka has been used here for classification of the reliable parameters based on E and M estimations made by this EM algorithm[15]. E(xpectation) it acquires the probable values for the missing data using an initial parameter estimate and M(aximization) tries to acquire the maximum likelihood approximation for the parameters by means of the estimated data[16]. The classified parameters are used as the required input-data for NTP in MapReduce. The unclassified parameters are henceforth removed from HDFS.A sample record in matrix X after classification is shown below:
12/12/2015 9:19 80,64,1655,64,134,695,772,915,32.

5.3 Transformation of NTP to MapReduce

The MapReducer's map, relates the input-data in the usage form of key and value pairs, to a list the intermediary key-value pairs by the map technique which is referred as Map-Attribute (Key1; Value1) List < Key2; Value2 >. The mapped intermediate-records need not be of the similar form as that of the input-data records. An input-data pair can get mapped to any number of multiple output-data pairs or may even be mapped to zero or none.

The MapReduce architecture shows the working fashion in the below given Figure 4. The MapReducer's reducer reduces the set of intermediate-values that share a common key to a smaller set of values by a reduce function, which is referred as Reduce-Attribute (Key2; List < Value2 >) List < Value3 >. The reducer has three major stages, they are sort, shuffle and reduce accordingly. The sort stage framework clusters the inputs for each reducer by a key as various mappers can produce output pair for the similar key. The shuffle stage framework will reduce the copy of related sorted output pair from each mapper by means of HTTP through the network. The shuffle and sort phases shall work hand-in-hand at the same time as a result of which as the outputs are produced they are merged together. Finally, the reduce function is called for each key-value pairs and are grouped. The outcome yielded by the reducer is copied to Hadoop Distributed File System (HDFS).The algorithm is transformed to predict the traffic by internally dividing the process into corresponding jobs to enable distributed computation using MapReduce work-structure. The flow and the division of the jobs are shown in the job-flow diagram that has 3 divisions based on their movement in the process activity. Job 1optimizes; job 2 deploys and job 3 compares as stated above in Figure 2.

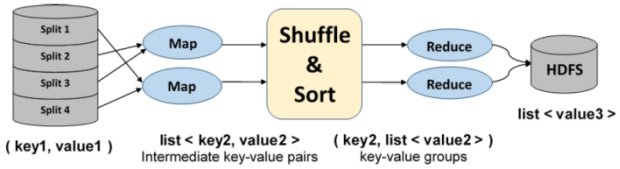

Figure 4. MapReduce work-flow architecture.

Algorithm for Job2 in MapReduce – Deploy the WF's for NTP

Function Map-Deploy
Called once at the beginning of the map task.
Z- Read into memory from a file on HDFS.
Num-Permutations = Number of rows in matrix P
(R) = Read into memory from file on HDFS.
Xn= Read as parameter.
Yn = Read as parameter
# Yn is an array of size n
n Number of tokens in Yn.
For iAttribute = 1 to n do
Yn[iAttribute] =Yn[iAttribute] + iR-WF
iR-WF # Optimize the initial designated DP
End for

Function Map (Key; Value)
# Key = Timestamp of the DP.
# Value = DP. The delimiter is ",".
Xn = Parse value to tokens by the delimiter `,'
# Xn is an array of size n
For iAttribute = 1 to n do
Xn[iAttribute] = Xn[iAttribute]+ iR-WF
iR-WF # Optimize the input DP
End for
# For each permutation in P:
For iPermutation = 1 to Num-Permutations do
iR-WF # Initialize WF(weight-factor)
Permutation Parse line iPermutation in P to tokens by the delimiter `,'
For iAttribute = 1 to n do
# For each feature in the permutation iPermutation
j=0
if Permutation[iAttribute] = 1 then
Xk[j] = Xn[iAttribute]
Yk[j] = Yn[iAttribute]
iR-WF = iR-WF + (Xk[iAttribute] + Yk[iAttribute])2
j = j + 1
End if
End for
If Value = Yn then # The DP is the initial designated DP
Give (iPermutation; (TimeStamp; 0; Xk))
End if
If iR-WF (Value) < 1 then # The DP is added with iR-WF
Give (iPermutation; (TimeStamp; Xn; Xk))
End if
End for

Algorithm for Job3 in MapReduce– Compare actual vs predicted traffic
Function Map(Key; Value)

# Key = Timestamp of Data Point (DP).
# Value = DP. (It is separated by ",").
Xn = an array of size n
n = Total number of tokens in Xn
normalFlag= true
For iRow = 1 to numRows do
if iRow[Xn] = iRow[Xk] then
Give (eventsCells[iRow]; 1)
normalFlag=false# This DP is not matching with actual.
Break for loop # Terminate the for loop when the mismatch is found
End if
End for
If normalFlag = true then # If this DP is matching (Actual traffic and Predicted traffic match)
Give(Normal, 1)

```
End if

Function Reduce (Key; Values)
# Key = pattern type
# Values = A list of f1ags.
sum = 0
For all value belonging to values do
sum  = sum + Value
End for
Give ( key; sum )
```

5.4 Evaluation – Performance metrics

The permutation matrix castoff for the trials are of size 21,648 X 31i.e., 21,648 permutations with 31 attributes for each permutation. The attributes were well defined in section 3.The parameters of the permutation matrix were chosen to be min-attributesas 3 andmax-attributesas 31.The periodchosen for the attribute classification was 0.9 minutes. The parameter classification preferredhere, deploys EM algorithm with a minor time interval, such as 30 milliseconds, 0.5 seconds, etc. The error rate eventually sinks low due to the use of the weight adjustment factors done in Map Reduce. Hence, the performance evaluation here counting on the True Positive (TP) and False Positive (FP) showcases that the error rate is down as TP signs up than FP as shown below in Table 1.

Table 1. Error rate (%) on Netrsec dataset.

Traffic Type	Parameters			Number of TP	Number of FP	Error rate (%)
	x1	x2	x3			
Mail	3	1	2	2.95	0.05	0.0005
TCP	3	2.8	3	2.99	0.001	0.00001
UDP	2	0	2	1.98	0.02	0.0002
FDP	1	1	0	0.94	0.06	0.0006
RTP	4	3	0	3.9	0.1	0.001
ATTACK	6	6	5	5.97	0.03	0.0003
SPAM	4	0	4	3.99	0.01	0.0001

VI. Experimental Results and comparison

The performance assessment was executed on a cluster and a single Hadoop-node. Figure 5 demonstrates that 10 iterations of the NTP algorithm performed on a cluster got executed completely in 9 minutes for 1TB, which show casts an excessive speed in computing of about 3 to 10 times faster than their actual completion time performed by only one node. These results illustrate the increased performance enhancement as the volume of the input traffic grows bigger.

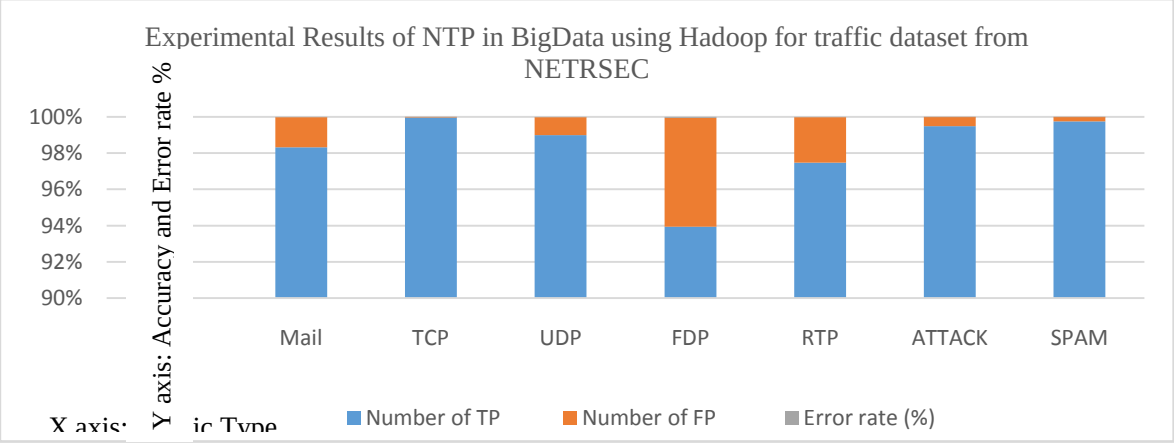

Figure 5. Experimental Results of NTP in BigData using Hadoop for dataset from NETRSEC

VII. Conclusion and Future Work

The experimental tests were performed on the dataset downloaded from Netresec a software vendor who emphasis on the network security field.The application described in this paper, is based on the development and application of a Network traffic prediction algorithm in Big Data using MapReduce methodology deployed on a Hadoop framework.Though the outcomes obtained at this time have proven the efficiency of the NTP algorithm, it may possibly be further technologically advanced in a number of means. There can be extensions done as follows:

1. Outspreading the algorithm to deploy a more erudite feature selection procedure that can also remove redundant and irrelevant characteristics to produce better accuracy by increasing the matching rate and decrease the false positive rate.
2. Spreading the algorithm to apply the preprocessing stage too over Hadoop.

References

[1]. Venkatesh, C. and D. Kumar, Increasing Efficiency of Recommendation System using Big Data Analysis.
[2]. Ackerman, D., et al., Similarity Detection via Random Subsets for Cyber War Protection in Big Data using Hadoop Framework. 2015.
[3]. Gharaibeh, M. and C. Papadopoulos, DARPA-2009 Intrusion Detection Dataset Report. 2014, Technical report, Colorado State University.
[4]. Macdonald, A., The Architecture Of An Autonomic, Resource-Aware, Workstation-Based Distributed Database System. 2012.
[5]. Pries, K.H. and R. Dunnigan, Big Data Analytics: A practical guide for managers. 2015: CRC Press.
[6]. Fahringer, T., et al., ASKALON: a tool set for cluster and Grid computing. Concurrency and Computation: Practice & Experience, 2005. 17(2): p. 143-169.
[7]. Zaharia, M., et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. in Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. 2012. USENIX Association.
[8]. Dean, J. and S. Ghemawat, MapReduce: a flexible data processing tool. Communications of the ACM, 2010. 53(1): p. 72-77.
[9]. Sobie, R., et al. HTC scientific computing in a distributed cloud environment. in Proceedings of the 4th ACM workshop on Scientific cloud computing. 2013. ACM.
[10]. De, K., et al., Extending the ATLAS PanDA Workload Management System for New Big Data Applications. 2013, ATL-COM-SOFT-2013-017.
[11]. Davidoff, S. and J. Ham, Network forensics: tracking hackers through cyberspace. 2012: Prentice hall.
[12]. Lee, Y. and Y. Lee, Toward scalable internet traffic measurement and analysis with hadoop. ACM SIGCOMM Computer Communication Review, 2013. 43(1): p. 5-13.
[13]. Menon, R. and O.G. MENON, Mining of textual databases within the product development process. 2004.
[14]. Davis, J.J. and A.J. Clark, Data preprocessing for anomaly based network intrusion detection: A review. Computers & Security, 2011. 30(6): p. 353-375.
[15]. Hall, M., et al., The WEKA data mining software: an update. ACM SIGKDD explorations newsletter, 2009. 11(1): p. 10-18.
[16]. Park, H., P.H. Bland, and C.R. Meyer, Construction of an abdominal probabilistic atlas and its application in segmentation. Medical Imaging, IEEE Transactions on, 2003. 22(4): p. 483-492.