ISSN:0975-9646

Empowering Service Claims Fraudulent Using Big Data Analytics

[1] U. Kartheek Chandra Patnaik,[2]Ch. Navyatha, [3]B.V.Maliney, [4]B.Mounica, [5]J. Neelima

[1] Associate Professor, [2,3,4,5] Students

[1,2,3,4,5] Department of Computer Science and Engineering.

[1,2,3,4,5] Lendi Institute of Engineering and Technology.

[1,2,3,4,5] Vizianagaram, Andhra Pradesh, India

Abstract-As more and more software moves to Data Analytics as a Service (DAaaS), the web application has become more ubiquitous and log file analysis is becoming a necessary task for analyzing the client's behaviour. Log files are getting generated very fast i.e., at the rate of 1-10 Mb/s per server. A single data centre can generate tens of terabytes of log data in a day which is very huge. In order to analyze such large datasets we need parallel processing system and reliable data storage mechanism. Virtual database system is an effective solution for integrating the data, but it becomes inefficient for large datasets. As log files are continuous stream data from distributed servers, an efficient way to handle such data is needed to store and send data into Storage servers. Our system uses Apache Flume to gather streams of log data from various servers and store them into HDFS, a Hadoop Distributed File System. MapReduce a parallel processing strategy breaks up input data and sends fractions of the original data to several machines in Hadoop cluster. This mechanism helps to process huge amounts of log data in parallel, using all the machines in the Hadoop cluster and computes result efficiently. This approach reduces the computation and response time as well as the load on to the end system. This paper proposes a server log analysis using Apache Flume for continuous streaming, Apache Hadoop through MapReduce and Pig for analyzing such logs, thereby providing accurate results for clients.

Keywords -DAaaS, Server Logs, Apache Flume, HDFS, MapReduce, Apache Pig

INTRODUCTION

Big Data and Cloud, two of the trends that are defining the emerging Enterprise Computing, show a lot of potential for a new era of combined applications. The provision of Big Data analytical capabilities using cloud delivery models could ease adoption for many companies, and in addition to important cost savings, it could simplify useful insights that could provide them with different kinds of competitive advantage.

Along these lines, Data Analytics as a Service (DAaaS) represents the approach to an extensible platform that can provide cloud-based analytical capabilities over a variety of industries and use cases. From a functional perspective, the platform covers the end-to-end capabilities of an analytical solution, from data acquisition to end-user visualization, reporting and interaction. Beyond this traditional functionality, it extends the usual approach with innovative concepts, like Analytical Apps and a related Analytical Appstore. In addition, the platform supports the needs of the different users who interact with it, including those of the emerging 'Data Scientist' role.

Analyzing is foremost scenario in the trends of DAaaS to make end user satisfaction. One such data analytics is Log File Analysis. Server logs are computer-generated log files that capture network and server operations data. They are useful for managing network operations, especially for security and regulatory compliance. Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log streaming data into the Hadoop Distributed File System (HDFS). It has a simple and flexible architecture based on streaming data flows and is robust and fault tolerant with tuneable reliability mechanisms for failover and recovery.

Apache Hadoop is an open source software framework for storing and large scale processing of data-sets on clusters of commodity hardware. Hadoop is an Apache top-level project being built and used by a global community of contributors and users. It is licensed under the Apache License 2.0. All the modules in Hadoop are designed with a fundamental assumption that hardware failures (of individual machines or racks of machines) are common and thus should be automatically handled in software by the framework. Apache Hadoop's MapReduce and HDFS components originally derived respectively from Google's MapReduce and Google File System (GFS) papers.

METHODS

A data is a collection of facts from the grids of web servers usually of unorganized form in the digital universe. Around 90% of data present in today's world are generated in last two years [1]. A large amount of the data available in the internet is generated either by individuals, groups or by the organization over a particular period of time. The volume of data becomes larger day by day as the usage of World Wide Web makes an interdisciplinary part of human activities. Rise of these data leads to a new technology such as big data that acts as a tool to process, manipulate and manage very large dataset along with the storage required.

Big Data is a high volume, high velocity and high variety information assets that demand cost-effective [2], innovative forums of information processing for enhanced insight and decision making. Big data, a buzz word in the business intelligence can handle petabytes or terabytes of data in a reasonable amount of time. Big data is distinct from large existing database which uses Hadoop framework for data intensive distributed applications.

Big Data analytics applies advanced analytical techniques of large datasets to discover hidden patterns and other useful information. It is performed using software tools mainly for predictive analysis and data mining. The growing number of technologies is used to aggregate, manipulate, manage and analyze big data. Some of the most prominent technologies are [3] NoSQL databases that include Cassandra, MongoDb, redis, Hbase and Hadoop framework includes Hadoop, HDFS, Hive, Pig.

Numerous research works are carried out in web log mining, hadoop and some of them are reviewed below. Murat et al., [4] proposed the smart miner framework that extracts the user behavior from web logs. The framework used the smart session construction to trace the frequent user access paths. Sayalee Narkhede et al., [5] introduced the Hadoop-MR log file analysis tool that provides a statistical report on total hits of a web page, user activity, traffic sources. This work was performed in two machines with three instances of hadoop by distributing the log files evenly to all nodes.

Milind Bhandare et al., [6] put forth a generic log analyzer framework for different kinds of log files such as a database or file system. The work was implemented as a distributed query processing to minimize the response time for the users which can be extendable for some format of logs. Parallelization of Genetic Algorithm (PGA) was suggested by Kanchan Sharadchandra Rahate et al., [7]. PGA uses OlexGA package for classifying the document. The train model data is stored in HDFS and the test model categories the text document.

A framework for unstructured data analysis was proposed by Das et al., [8] using big data of public tweets from twitter. The tweets are stored in Hbase using Hadoop cluster through Rest Calls and text mining algorithms are processed for data analysis. The semi structured log files are large datasets which are challenging to store, search, share, visualize and analyze. Almost 26% of web log types of data require big data technology to perform an analysis [9]. In order to improve the usage of a website and to track the user behavior, in online advertising and E-commerce the web log mining is performed using Hadoop.

The related works so far stated above performs the work with good scalability but fails to experiment the time efficiency between the different modes of hadoop and necessity of the scalability. The proposed work analyses the working of both hadoop modes and the time efficiency in each mode specifically for semi structure log data, along which a statistical report is made.

Apache Hadoop is an open source framework for distributed storage. It process large amounts of datasets on a commodity hardware. Hadoop enables is to take instant decisions in businesses from massive amounts of structured and unstructured data.

Hadoop Distributed File System (HDFS) is the core technology for the efficient large storage layer, and is also designed to run across low-cost commodity hardware. It stores huge amounts of datasets in it and provides the datasets for processing when required.

Apache Flume is a reliable, distributed and available service for efficiently collecting, aggregating, and moving large amounts of streaming data into the Hadoop Distributed File System (HDFS). It has a simple and flexible architecture for streaming data flows. It is also robust and fault tolerant with reliability mechanisms for failover and recovery.

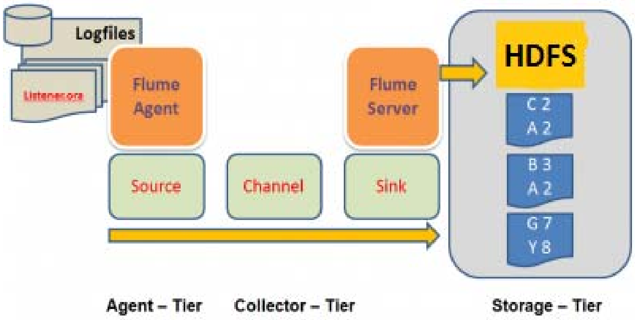


Fig1. Flume Architecture

MapReduce is a framework used for writing applications that process large amounts of structured and unstructured data in parallel across a cluster of hundreds of machines, in a fault-tolerant and reliable manner. *MapReduce* splits a large dataset into independent chunks of data and organizes that data in key, value pairs for parallel processing. This parallel processing of data in clusters improves the speed and reliability of the cluster and returns the solutions more quickly and with more reliability.

The *Map* function divides the input data into chunks by the Input Format and creates a map task for each chunk in the input. The JobTracker distributes tasks to the worker(slave) nodes. The output of each map task is taken and partitioned into a group of key-value pairs for each reduce.

The *Reduce* function then collects the various results from worker nodes and combines them for an efficient answer for the problem, which the master node was trying to solve. Each reduce pulls the appropriate partition from the machines where the maps have executed, and then writes its result back into HDFS. Thus, the reduce is able to collect the data from all of the maps for the keys and is responsible for combining the results in order to solve the problem.

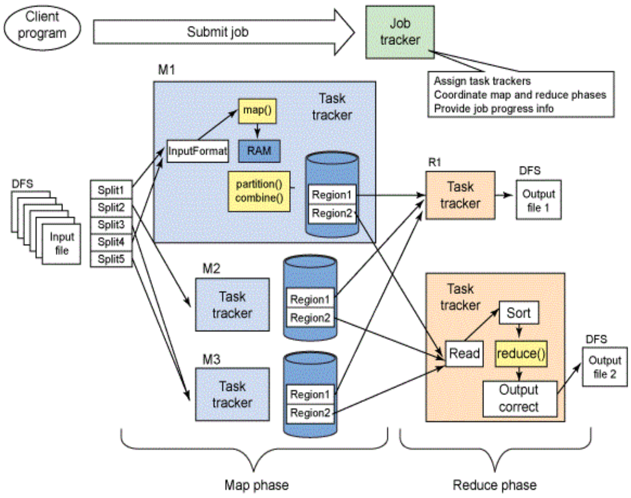


Fig2. MapReduce Architecture

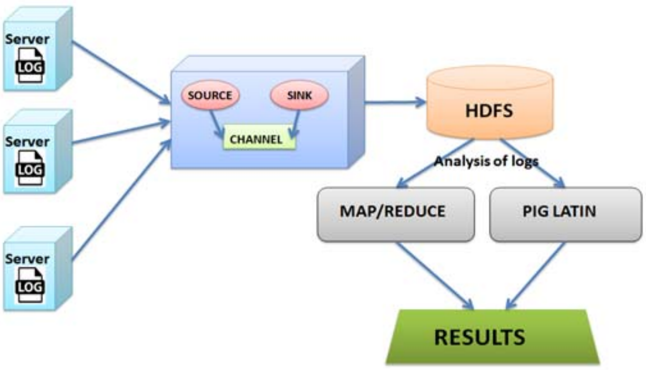


Fig 3.System Architecture

Apache Pig allows us to write complex MapReduce transformations using a very simple and easy scripting language. Pig Latin (the language) defines a set of transformations on a data set such as aggregate, join and sort. Pig translates the Pig Latin script into MapReduce so that it can be executed within Hadoop®. Pig Latin is sometimes extended using UDFs (User Defined Functions), which the user can write in Java or a scripting language and then call directly from the Pig Latin.

Pig runs on Hadoop and makes use of MapReduce and the Hadoop Distributed File System (HDFS). The language for the platform is called Pig Latin, which abstracts from the Java MapReduce idiom into a form similar to SQL. Pig Latin is a flow language whereas SQL is a declarative language. SQL is great for asking a question of your data, while Pig Latin allows you to write a data flow that describes how your data will be transformed. Since Pig Latin scripts can be graphs (instead of requiring a single output) it is possible to build complex data flows involving multiple inputs, transforms, and outputs. Users can extend Pig Latin by writing their own functions, using Java, Python, Ruby, or other scripting languages.

The user can run Pig in two modes:

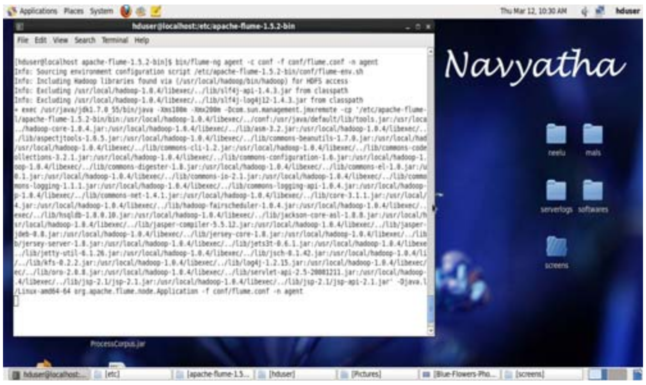
- Local Mode. With access to a single machine, all files are installed and run using a local host and file system.
- MapReduce Mode. This is the default mode, which requires access to a Hadoop cluster.

The user can run Pig in either mode using the “pig” command or the “java” command.

RESULTS & DISCUSSIONS

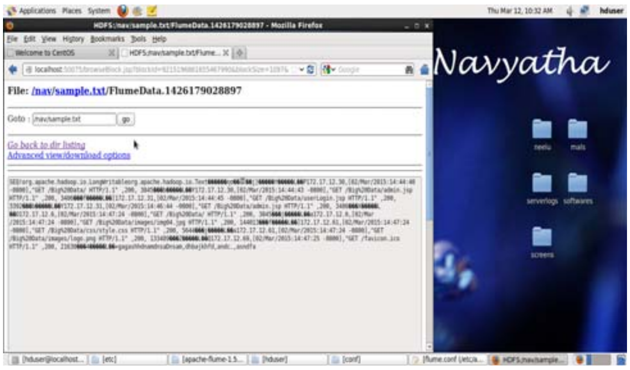
The data from multiple servers leads to Big Data which is meant as semi-structured, and the need to analyze such data is daunting task. We analyzed the log data using both Apache Pig and MapReduce, there by comparing the performance of analyzing and time take to process the map and reduce functions, showed in Fig.10.

Apache Flume, a framework which is used by the Web server to collect different server log data by different agents, there by collected by collector nodes and finally stored in HDFS Sink, as represented in Fig. 4 and Fig.5



```
[hduser@localhost apache-flume-1.5.2-bin]$ bin/flume-ng agent -c conf -f conf/flume.conf -n agent
Info: Sourcing environment configuration script /etc/apache-flume-1.5.2-bin/conf/flume-env.sh
Info: Including Hadoop libraries found via (/usr/local/hadoop/bin/hadoop) for HDFS access
Info: Excluding /usr/local/hadoop-1.0.4/libexec/../lib/slf4j-api-1.4.3.jar from classpath
Info: Excluding /usr/local/hadoop-1.0.4/libexec/../lib/slf4j-log4j12-1.4.3.jar from classpath
+ exec /usr/java/jdk1.7.0_55/bin/java -Xms100m -Xmx200m -Dcom.sun.management.jmxremote -cp '/etc/apache-flume-
1/apache-flume-1.5.2-bin/bin:/usr/local/hadoop-1.0.4/libexec/../conf:/usr/java/default/lib/tools.jar:/usr/loca
../hadoop-core-1.0.4.jar:/usr/local/hadoop-1.0.4/libexec/../lib/asm-3.2.jar:/usr/local/hadoop-1.0.4/libexec/..
./lib/aspectjtools-1.6.5.jar:/usr/local/hadoop-1.0.4/libexec/../lib/commons-beanutils-1.7.0.jar:/usr/local/had
/usr/local/hadoop-1.0.4/libexec/../lib/commons-cli-1.2.jar:/usr/local/hadoop-1.0.4/libexec/../lib/commons-code
ollections-3.2.1.jar:/usr/local/hadoop-1.0.4/libexec/../lib/commons-configuration-1.6.jar:/usr/local/hadoop-1.
oop-1.0.4/libexec/../lib/commons-digester-1.8.jar:/usr/local/hadoop-1.0.4/libexec/../lib/commons-el-1.0.jar:/u
0.1.jar:/usr/local/hadoop-1.0.4/libexec/../lib/commons-io-2.1.jar:/usr/local/hadoop-1.0.4/libexec/../lib/commo
mons-logging-1.1.1.jar:/usr/local/hadoop-1.0.4/libexec/../lib/commons-logging-api-1.0.4.jar:/usr/local/hadoop-
p-1.0.4/libexec/../lib/commons-net-1.4.1.jar:/usr/local/hadoop-1.0.4/libexec/../lib/core-3.1.1.jar:/usr/local/
4.jar:/usr/local/hadoop-1.0.4/libexec/../lib/hadoop-fairscheduler-1.0.4.jar:/usr/local/hadoop-1.0.4/libexec/..
exec/../lib/hsqldb-1.8.0.10.jar:/usr/local/hadoop-1.0.4/libexec/../lib/jackson-core-asl-1.8.8.jar:/usr/local/h
sr/local/hadoop-1.0.4/libexec/../lib/jasper-compiler-5.5.12.jar:/usr/local/hadoop-1.0.4/libexec/../lib/jasper-
jdeb-0.8.jar:/usr/local/hadoop-1.0.4/libexec/../lib/jersey-core-1.8.jar:/usr/local/hadoop-1.0.4/libexec/../lib
b/jersey-server-1.8.jar:/usr/local/hadoop-1.0.4/libexec/../lib/jets3t-0.6.1.jar:/usr/local/hadoop-1.0.4/libexe
../lib/jetty-util-6.1.26.jar:/usr/local/hadoop-1.0.4/libexec/../lib/jsch-0.1.42.jar:/usr/local/hadoop-1.0.4/li
/../lib/kfs-0.2.2.jar:/usr/local/hadoop-1.0.4/libexec/../lib/log4j-1.2.15.jar:/usr/local/hadoop-1.0.4/libexec/
ec/../lib/oro-2.0.8.jar:/usr/local/hadoop-1.0.4/libexec/../lib/servlet-api-2.5-20081211.jar:/usr/local/hadoop-
.4/libexec/../lib/jsp-2.1/jsp-2.1.jar:/usr/local/hadoop-1.0.4/libexec/../lib/jsp-2.1/jsp-api-2.1.jar' -Djava.l
/Linux-amd64-64 org.apache.flume.node.Application -f conf/flume.conf -n agent
```

Fig 4.Apache Flume Agent



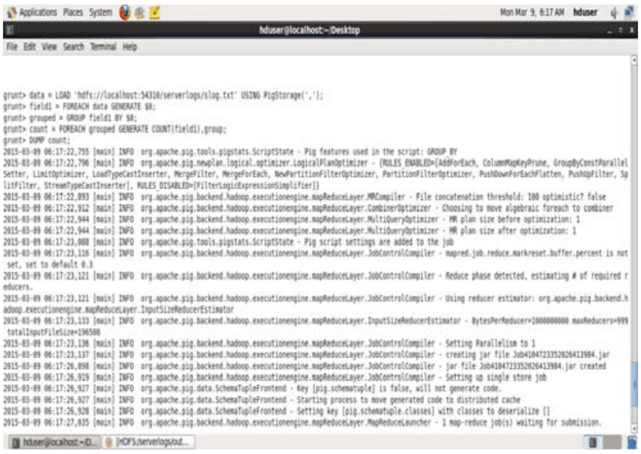
File: /nav/sample.txt/FlumeData.1426179028897

Goto : /nav/sample.txt

Go back to dir listing
Advanced view/download options

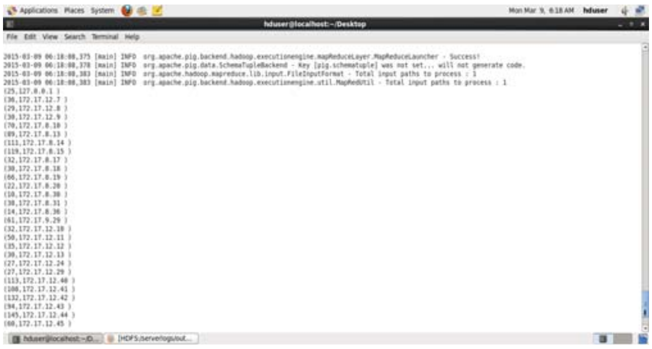
```
SEQ!org.apache.hadoop.io.LongWritableorg.apache.hadoop.io.Text[illegible]172.17.12.30,[02/Mar/2015:14:44:40
-0000],"GET /Big%20Data/ HTTP/1.1" ,200, 3045[illegible]172.17.12.30,[02/Mar/2015:14:44:43 -0000],"GET /Big%20Data/admin.jsp
HTTP/1.1" ,200, 3406[illegible]172.17.12.31,[02/Mar/2015:14:44:45 -0000],"GET /Big%20Data/userLogin.jsp HTTP/1.1" ,200,
3392[illegible]172.17.12.31,[02/Mar/2015:14:46:44 -0000],"GET /Big%20Data/admin.jsp HTTP/1.1" ,200, 3406[illegible]
[illegible]172.17.12.6,[02/Mar/2015:14:47:24 -0000],"GET /Big%20Data/ HTTP/1.1" ,200, 3045[illegible]172.17.12.6,[02/Mar
/2015:14:47:24 -0000],"GET /Big%20Data/images/img04.jpg HTTP/1.1" ,200, 14401[illegible]172.17.12.61,[02/Mar/2015:14:47:24
-0000],"GET /Big%20Data/css/style.css HTTP/1.1" ,200, 5644[illegible]172.17.12.61,[02/Mar/2015:14:47:24 -0000],"GET
/Big%20Data/images/logo.png HTTP/1.1" ,200, 13340[illegible]172.17.12.60,[02/Mar/2015:14:47:25 -0000],"GET /favicon.ico
HTTP/1.1" ,200, 21630[illegible]
```

Fig 5.Data stored in HDFS Sink


Applications Places System
Mon Mar 9, 6:17 AM
hduser
hduser@localhost:~/Desktop
File Edit View Search Terminal Help
hduser@localhost:~/D...
[HDFS:/serverlogs/out...

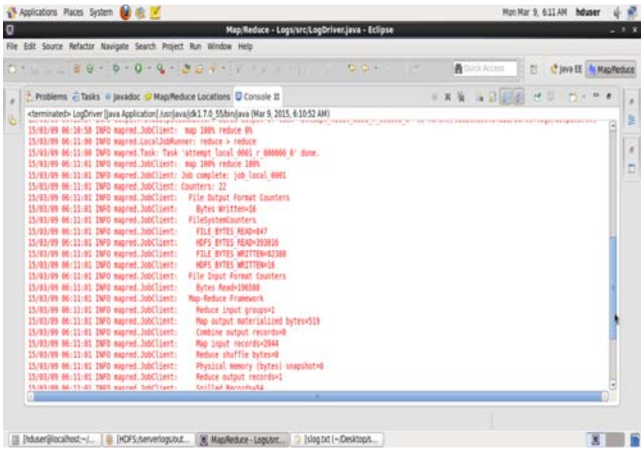
```
grunt> data = LOAD 'hdfs://localhost:54310/serverlogs/slog.txt' USING PigStorage(',');
grunt> field1 = FOREACH data GENERATE $0;
grunt> grouped = GROUP field1 BY $0;
grunt> count = FOREACH grouped GENERATE COUNT(field1),group;
grunt> DUMP count;
2015-03-09 06:17:22,755 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: GROUP_BY
2015-03-09 06:17:22,796 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, GroupByConstParallel
Setter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NewPartitionFilterOptimizer, PartitionFilterOptimizer, PushDownForEachFlatten, PushUpFilter, Sp
litFilter, StreamTypeCastInserter], RULES_DISABLED=[FilterLogicExpressionSimplifier]}
2015-03-09 06:17:22,893 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2015-03-09 06:17:22,912 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.CombinerOptimizer - Choosing to move algebraic foreach to combiner
2015-03-09 06:17:22,944 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2015-03-09 06:17:22,944 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
2015-03-09 06:17:23,088 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig script settings are added to the job
2015-03-09 06:17:23,116 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.percent is not
 set, set to default 0.3
2015-03-09 06:17:23,121 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Reduce phase detected, estimating # of required r
educers.
2015-03-09 06:17:23,121 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Using reducer estimator: org.apache.pig.backend.h
adoop.executionengine.mapReduceLayer.InputSizeReducerEstimator
2015-03-09 06:17:23,133 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.InputSizeReducerEstimator - BytesPerReducer=1000000000 maxReducers=999
 totalInputFileSize=196508
2015-03-09 06:17:23,136 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting Parallelism to 1
2015-03-09 06:17:23,137 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - creating jar file Job4104723352026413984.jar
2015-03-09 06:17:26,898 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - jar file Job4104723352026413984.jar created
2015-03-09 06:17:26,919 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting up single store job
2015-03-09 06:17:26,927 [main] INFO  org.apache.pig.data.SchemaTupleFrontend - Key [pig.schematuple] is false, will not generate code.
2015-03-09 06:17:26,927 [main] INFO  org.apache.pig.data.SchemaTupleFrontend - Starting process to move generated code to distributed cache
2015-03-09 06:17:26,928 [main] INFO  org.apache.pig.data.SchemaTupleFrontend - Setting key [pig.schematuple.classes] with classes to deserialize []
2015-03-09 06:17:27,035 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 1 map-reduce job(s) waiting for submission.
```

Fig 6.Pig Analysis in Cent OS


Applications Places System
Mon Mar 9, 6:18 AM
hduser
hduser@localhost:~/Desktop
File Edit View Search Terminal Help
hduser@localhost:~/D...
[HDFS:/serverlogs/out...

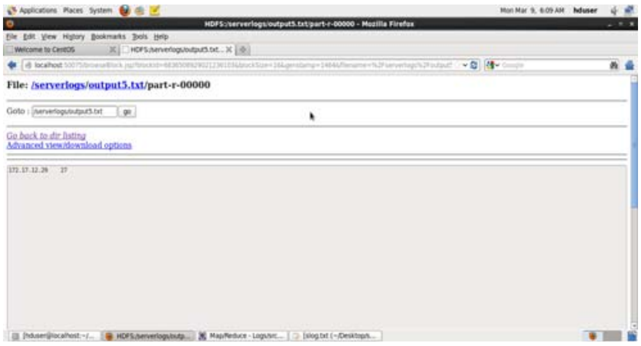
```
2015-03-09 06:18:08,375 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2015-03-09 06:18:08,378 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2015-03-09 06:18:08,383 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2015-03-09 06:18:08,383 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(25,127.0.0.1 )
(36,172.17.12.7 )
(29,172.17.12.8 )
(30,172.17.12.9 )
(70,172.17.8.10 )
(89,172.17.8.13 )
(111,172.17.8.14 )
(119,172.17.8.15 )
(32,172.17.8.17 )
(30,172.17.8.18 )
(66,172.17.8.19 )
(22,172.17.8.20 )
(10,172.17.8.30 )
(38,172.17.8.31 )
(14,172.17.8.36 )
(61,172.17.9.29 )
(32,172.17.12.10 )
(50,172.17.12.11 )
(35,172.17.12.12 )
(30,172.17.12.13 )
(27,172.17.12.24 )
(27,172.17.12.29 )
(113,172.17.12.40 )
(108,172.17.12.41 )
(132,172.17.12.42 )
(94,172.17.12.43 )
(145,172.17.12.44 )
(60,172.17.12.45 )
```

Fig 7.IPAddress Retrieval in Hadoop Using Pig


Map/Reduce - Logs/src/LogDriver.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Tasks Javadoc Map/Reduce Locations Console

```
<terminated> LogDriver [Java Application] /usr/java/jdk1.7.0_55/bin/java (Mar 9, 2015, 6:10:52 AM)
15/03/09 06:10:58 INFO mapred.JobClient:  map 100% reduce 0%
15/03/09 06:11:00 INFO mapred.LocalJobRunner: reduce > reduce
15/03/09 06:11:00 INFO mapred.Task: Task 'attempt_local_0001_r_000000_0' done.
15/03/09 06:11:01 INFO mapred.JobClient:  map 100% reduce 100%
15/03/09 06:11:01 INFO mapred.JobClient: Job complete: job_local_0001
15/03/09 06:11:01 INFO mapred.JobClient: Counters: 22
15/03/09 06:11:01 INFO mapred.JobClient:   File Output Format Counters
15/03/09 06:11:01 INFO mapred.JobClient:     Bytes Written=16
15/03/09 06:11:01 INFO mapred.JobClient:   FileSystemCounters
15/03/09 06:11:01 INFO mapred.JobClient:     FILE_BYTES_READ=847
15/03/09 06:11:01 INFO mapred.JobClient:     HDFS_BYTES_READ=393016
15/03/09 06:11:01 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=82388
15/03/09 06:11:01 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=16
15/03/09 06:11:01 INFO mapred.JobClient:   File Input Format Counters
15/03/09 06:11:01 INFO mapred.JobClient:     Bytes Read=196508
15/03/09 06:11:01 INFO mapred.JobClient:   Map-Reduce Framework
15/03/09 06:11:01 INFO mapred.JobClient:     Reduce input groups=1
15/03/09 06:11:01 INFO mapred.JobClient:     Map output materialized bytes=519
15/03/09 06:11:01 INFO mapred.JobClient:     Combine output records=0
15/03/09 06:11:01 INFO mapred.JobClient:     Map input records=2044
15/03/09 06:11:01 INFO mapred.JobClient:     Reduce shuffle bytes=0
15/03/09 06:11:01 INFO mapred.JobClient:     Physical memory (bytes) snapshot=0
15/03/09 06:11:01 INFO mapred.JobClient:     Reduce output records=1
15/03/09 06:11:01 INFO mapred.JobClient:     Spilled Records=54
```

Fig 8.MapReduce functions for Logs


HDFS:/serverlogs/output5.txt/part-r-00000 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Welcome to CentOS
HDFS:/serverlogs/output5.txt...

File: /serverlogs/output5.txt/part-r-00000

Goto : /serverlogs/output5.txt go

Go back to dir listing
Advanced view/download options

```
172.17.12.28    27
```

Fig 9.IPAddress Count Retrieval in Hadoop Using Mapreduce

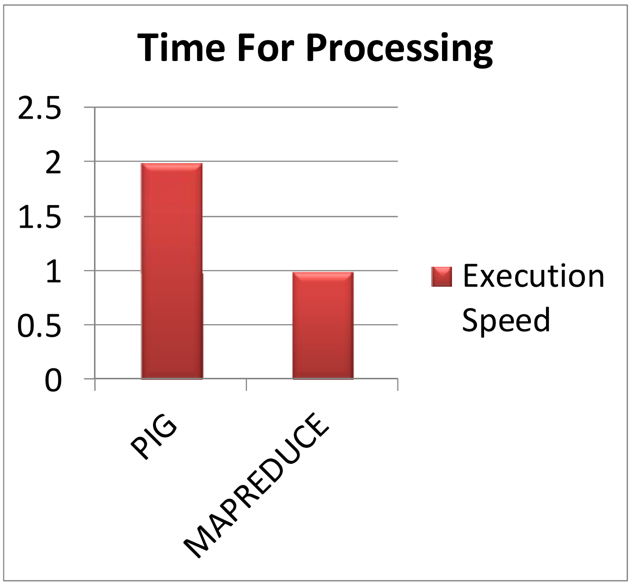


Fig.10 Comparative analysis between Pig and MapReduce

CONCLUSIONS

As the amount of data in real-world increases, the smoother way to analyze such data plays a vital role. Hence our paper enables the easier way to analyze such huge amount by Using Apache Pig and Map Reduce and their by finding the better way to reduce communication cost and time complexity.

ACKNOWLEDGEMENTS

We feel gratitude and privileged to b the students of Lendi Institute of Engineering and Technology and to be a part of doing the paper. We thank our Management, Principal, Head of the Department for their kind support in enabling the resources to us. We thank our respective family members for their support in helping in areas of our interest.

REFERENCES

[1] Ruchi Verma, Sathyan R Mani, “Use of Big Data Tehnologies in Capital Markets,” 2012 Infosys Limited, Bangalore, India.

[2] James Manyika, Brad Brown et.al, “Big Data: The next frontier for Innovation, Competition, and Productivity,” McKinsey Global Institute, June 2011.

[3] Murat Ali , Ismail Hakki Toroslu, “Smart Miner: A New Framework for mining Large Scale Web Usage Data,” WWW 2009, April 20-24. 2009 Madrid, Spain. ACM 978-1-60558-487-4/09/04.

[4] Sayalee Narkhede and Tripti Baraskar, “HMR Log Analyzer: Analyze Web Application Logs Over Hadoop MapReduce,” International Journal of UbiComp (IJU) vol.4, No.3, July 2013.

[5] Milind Bhandare, Vikas Nagare et al., “Generic Log Analyzer Using Hadoop Mapreduce Framework,” International Journal of Emerging Technology and Advanced Engineering (IJETAE), vol.3, issue 9, September 2013.

[6] Kanchan Sharadchandra Rahate et al., “A Novel Technique for Parallelization of Genetic Algorithm using Hadoop,” International Journal of Engineering Trends and Technology (IJETT), vol.4, issue 8, August 2013.

[7] T. K. Das et al., “BIG Data Analytics: A Framework for Unstructured Data Analysis,” International Journal of Engineering and Technology (IJET) vol 5, No 1, Feb-Mar 2013.

[8] Joseph McKendrick, “Big Data, Big Challeneges, Big Opportunities: 2012 IOUG Big Data strategies survey,” September 2012.

[9] Jeffrey Dean and Sanjay Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” Google, Inc

[10] Tom White, “Hadoop: The definitive Guide,” Third Edition, ISBN: 978- 1-449-31152-0-1327616795.

[11] Ian Mitchell, Mark Locke and Aundy Fuller, “The White Book of Big Data”