ISSN:2231-0711

E. Laxmi Lydia et al | IJCSET(www.ijcset.net) | November 2015 | Vol 5, Issue 11,390-394

Big Data Analysis using Hadoop components like Flume, MapReduce, Pig and Hive

E. Laxmi Lydia[1],Dr. M.Ben Swarup[2]

[1]*Associate Professor, Department of Computer Science and Engineering,*
Vignan's Institute Of Information Technology, Visakhapatnam, ndhra Pradesh, India.
[2]*Professor, Computer Science and Engineering,*
Vignan's Institute Of Information Technology, Visakhapatnam, Andhra Pradesh, India.

***Abstract*-In enormous information world, Hadoop Distributed File System (HDFS) is extremely well known. It gives a system to putting away information in a conveyed situation furthermore has set of instruments to recover and prepare. These information set utilizing guide decrease idea. In this paper, an intensive examination has been conveyed to talk about that how enormous information investigation can be performed on information put away on Hadoop dispersed document framework utilizing Pig and Hive. Apache Pig and Hive are two undertakings which are layered on top of Hadoop, and give more elevated amount dialect to utilize Hadoop's MapReduce library. In this paper, as a matter of first importance, the fundamental ideas of MapReduce, Pig and Hive are presented and their performance comparison.**

***Keywords*: Hadoop Distributed File System, Pig, Hive, mapreduce, Framework**

I. INTRODUCTION

Big data is mainly collection of data sets so large and complex that it is very difficult to handle them using on-hand database management tools. The main challenges with Big databases include creation, curation, storage, sharing, search, analysis and visualization. So to manage these databases we need, " highly parallel software's". First of all, data is acquired from different sources such as social media, traditional enterprise data or sensor data etc. Flume can be used to acquire data from social media such as twitter. Then, this data can be organized using distributed file systems such as Google File System or Hadoop File System. These file systems are very efficient when number of reads are very high as compared to writes. At last, data is analyzed using mapreducer so that queries can be run on this data easily and efficiently. Figure 1 showing the hadoop ecosystem.

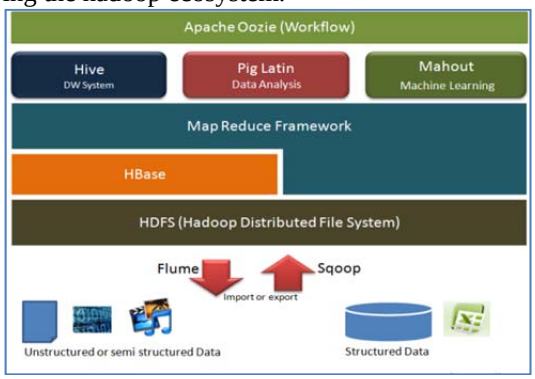

Figure1 Hadoop EcoSystem

II. ACQUIRE DATA

First of all, data has to be acquired from different sources. Main sources of data are:

- ✓ Traditional Organization data – it includes customer info from CRM systems, transactional ERP data or web store transactions and general ledger data.
- ✓ Machine generated or sensor data – it includes Call Detail Records, smart meters, weblogs, sensors, equipment logs and trading systems data.
- ✓ Social data – it includes customer feedback stream and micro blogging sites such as Twitter and social media platforms such as Facebook.

A. Flume

Data from social media is generally acquired using flume. Flume is an open source software program which is developed by cloudera to act as a service for aggregating and moving very large amount of data around a Hadoop cluster as data is produced or shortly thereafter. Primary use case of flume is to gather log files from all machines in cluster to persist them in a centralized store such as HDFS. In it, we have to create data flows by building up chains of logical nodes and connecting them to source and sinks. For example, if you want to move data from an apache access log into HDFS then you have to create a source by tail access.log and use a logical node to route this to an HDFS sink. Most of flume deployments have three tier design. The agent tier have flume agents collocated with sources of data which is to be moved. Collector tier consist of multiple collectors each of which collect data coming in from multiple agents and forward it on to storage tier which consist of file system like HDFS or GFS.

A **Flume agent** is a **JVM** process which has 3 components **-Flume Source**, **Flume Channel** and **Flume Sink-** through which events propagate after initiated at an external source. Figure 2 demonstrating the working of flume.

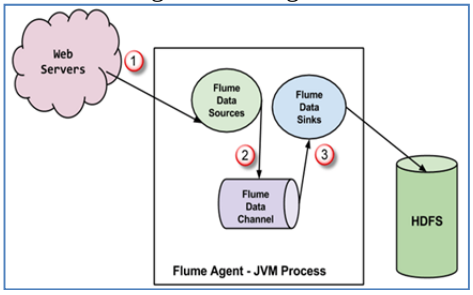

Figure 2 Working of Flume

1. In above diagram, the events generated by external source (WebServer) are consumed by Flume Data Source. The external source sends events to Flume source in a format that is recognized by the target source.
2. Flume Source receives an event and stores it into one or more channels. The channel acts as a store which keeps the event until it is consumed by the flume sink. This channel may use local file system in order to store these events.
3. Flume sink removes the event from channel and stores it into an external repository like e.g., HDFS. There could be multiple flume agents, in which case flume sink forwards the event to the flume source of next flume agent in the flow.

III. ORGANIZE DATA

After acquiring data, it has to be organize using a distributed file system. First of all, we have to break this data into fixed size chunks so that they can store and access easily. Mainly we use GFS and HDFS file systems.

A. Google File System

Google Inc. developed a distributed file system for their own use which was designed for efficient and reliable acess to data using large cluster of commodity hardware. It uses the approach of "BigFiles", which are developed by Larry Page and Sergey Brin. Here files are divided in fixedsize chunks of 64 MB. It has two types of nodes- one master node and many chunkserver nodes.

Files in fixed size chunks are stored on chunkservers which are assigned a 64 bit label by master at creation time. There are atleast 3 replication for every chunk but it can be more. Master node doesn't have data chunks, it keeps the metadata about chunks such as their label, their copy locations and their reading or writing processes. It also have the responsibility to replicate a chunk when it's copies become less than three. Figure 5 showing the architecture of GFS is following.

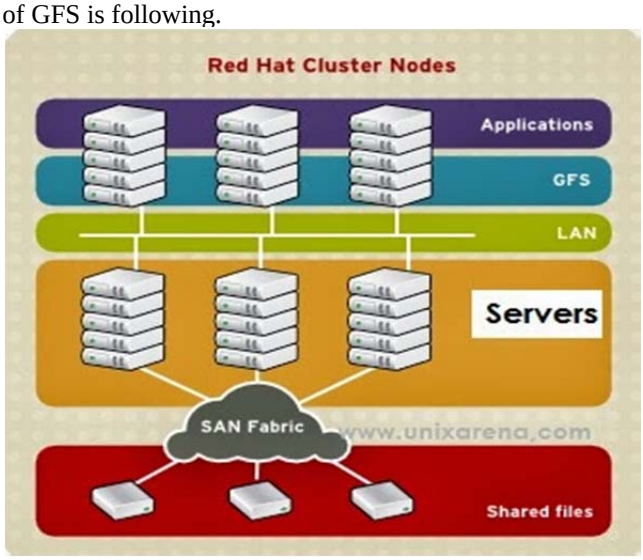

Figure 3 GFS

B. *Hadoop Distributed File System*

Hadoop distributed file system is a distributed, scalable and portable file system which is written in java. All machines which support java can run it. In it, every cluster has a single namenode and many datandes. A datanode has many blocks of same size except last block which have different size. It do communication using TCP/IP layer but clients uses RPC to communicate with each other. Every file in HDFS has a size of 64 MB or multiple of 64 MB. Reliability is due to replication of data. Atleast 3 copies of every datanode are present. Datanodes can communicate with each other to rebalance data or copy data or to keep high replication of data. HDFS has high availability by allowing namenode to be manually failed over to backup in case of failure. Now a days, automatic failover is also developing. It also uses a secondry namenode which continuously takes the snapshots of primary namenode so that it can be active when failure of primary node occurs. Data awareness between tasktracker and jobtracker is an advantage. Jobracker can schedule mapreduce job to tasktracker efficiently due to this data awareness. Figure 4 showing the HDFS.

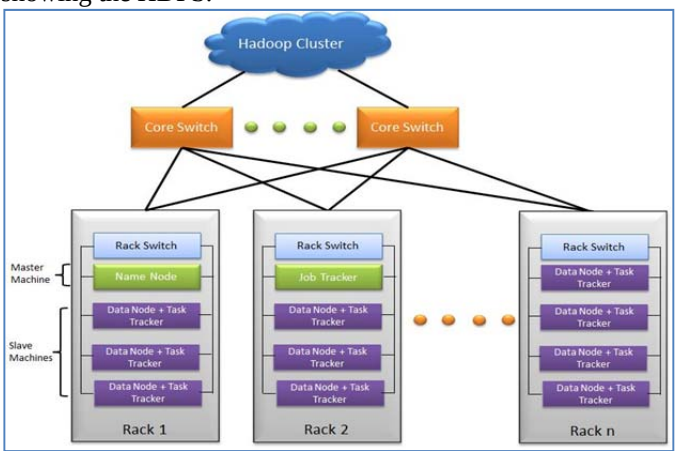

Figure 4 HDFS

IV. ANALYZE DATA

After organizing data, it has to be analyze to get fast and efficient results when a query is made. Mapreducer's are mainly used to analyze data. Mapreducer, Pig and Hive are very efficient for this purpose.

A. Setup for Analysis

An analysis is performed on a big database of 8 lakh records using Pig, Hive and MapReduce. For this purpose, we install Hadoop, Pig, Hive on cloudera. And analysis time is calculated for each [9]. Figure 5 showing the temperature datasets present in the HDFS. From the temperature datasets we are generating the maximum temperature from the year 1900 to 2014.

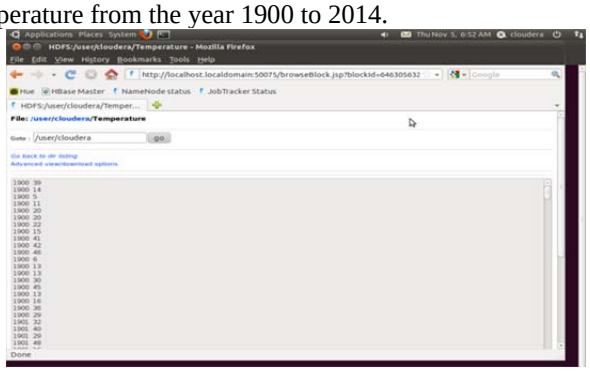

Figure 5 Temperature dataset in the HDFS

B. MapReduce

Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

A MapReduce *job* usually splits the input data-set into independent chunks which are processed by the *map tasks* in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the *reduce tasks*. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

Typically the compute nodes and the storage nodes are the same, that is, the MapReduce framework and the Hadoop Distributed File System are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster.

The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master.

Minimally, applications specify the input/output locations and supply *map* and *reduce* functions via implementations of appropriate interfaces and/or abstract-classes. These, and other job parameters, comprise the *job configuration*. The Hadoop *job client* then submits the job (jar/executable etc.) and configuration to the JobTracker which then assumes the responsibility of distributing the software/configuration to the slaves, scheduling tasks and monitoring them, providing status and diagnostic information to the job-client. Figure 6 demonstrating the execution of MapReduce and Figure 8 showing the status report of jobtracker and 100% completion of map and reduce jobs.

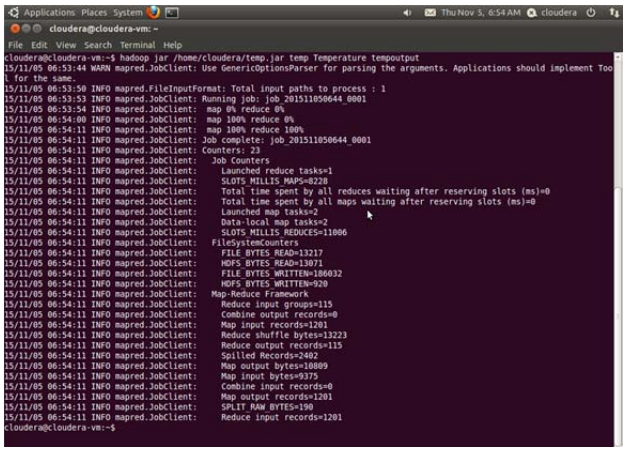

Figure 6 Execution of MapReduce

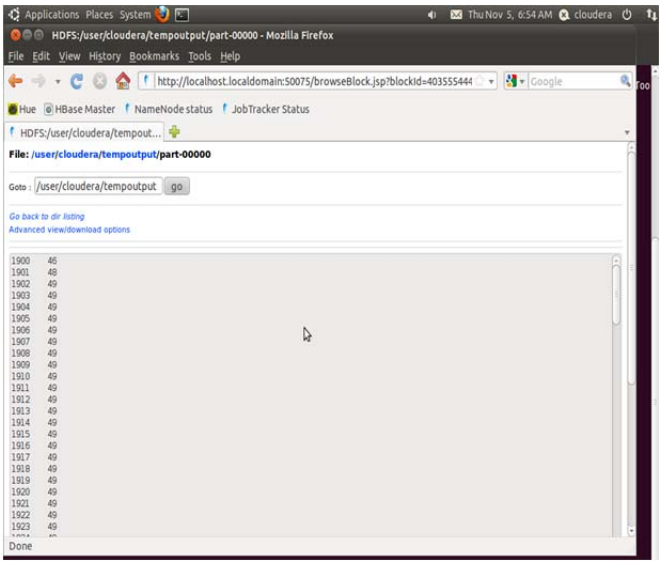

Figure 7 output screen, showing the maximum temperature from the year 1900 to 2014

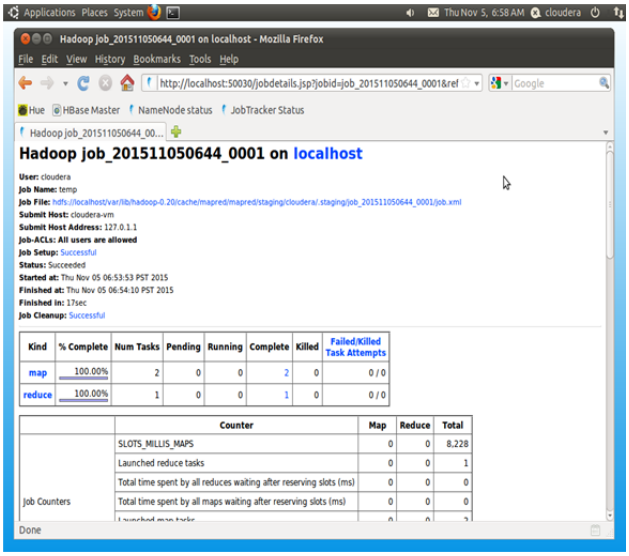

Figure 8 showing the status of jobtracker and 100% completion of map and reduce jobs.

C. Pig

Pig is a Apache Software Foundation project which was initially developed at Yahoo Research in 2006. Pig have a language and an execution environment. Piglatin which is a dataflow language is used by Pig. Piglatin is a type of language in which you program by connecting things together. Pig can handle complex data structure, even those who have levels of nesting. It has two types of execution environment local and distributed environment. Local environment is used for testing when distributed environment cannot be deployed. PigLatin program is collection of statements. A statement can be a operation or command. Here is a program in PigLatin to analyze a database. Figure 9 showing the pig's grunt shell and Figure 10 showing the Loading of temperature datasets to temp relation to perform the temperature analysis with less time than by writing map reduce code.

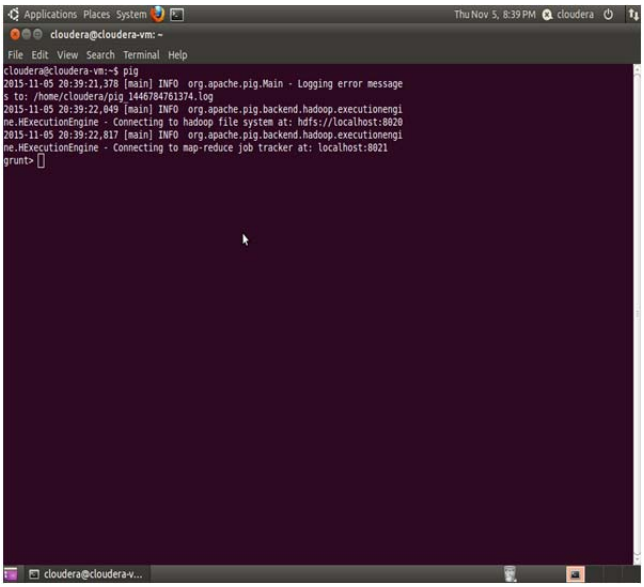

Figure 9 pig's grunt shell

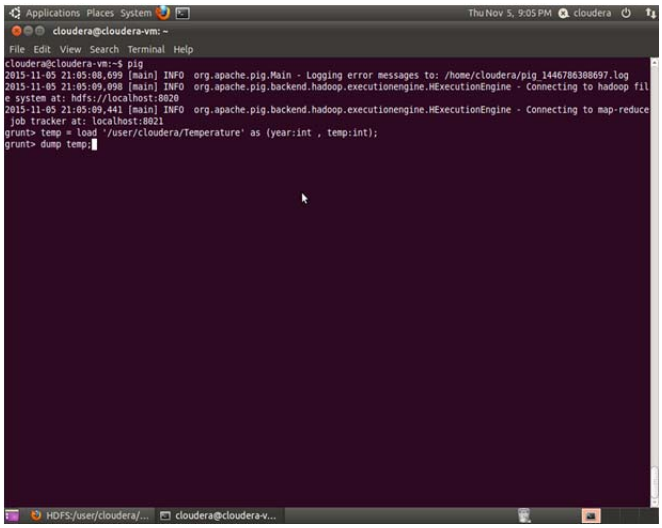

Figure 10 Loading of temperature datasets to temp relation

D. Hive

Hive is a technology which is developed by Facebook and which turns Hadoop into a datawarehouse complete with an extension of sql for querying. HiveQL which is a declarative language is used by Hive. In piglatin, we have to describe the dataflow but in Hive results must be describe. Hive itself find out a dataflow to get those results. Hive must have a schema but it can be more than one.
Hive must be configured before use. It can be configured in three different ways:

- ✓ By editing a file hive-site.xml,
- ✓ By hiveconf option in Hive command shell
- ✓ By using set command.

Here we have a database with more than eight lakh records which is analyzed by using Hive to get a temperature.

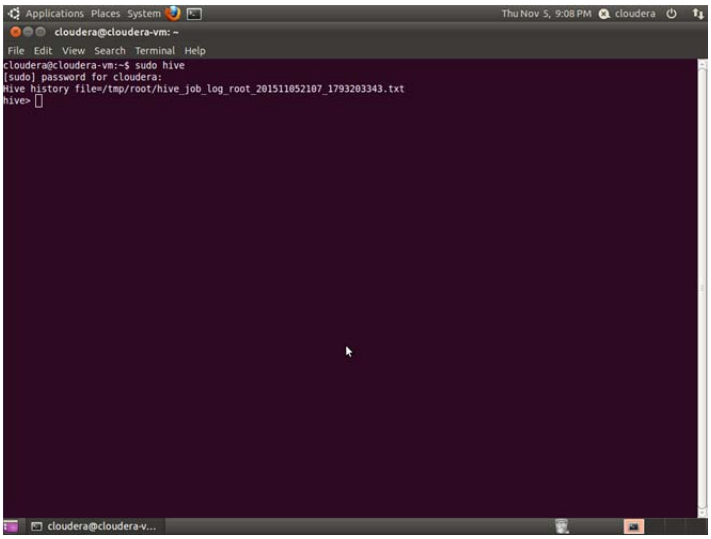

Figure 11 Hive Screen

- ✓ Creating Database
  Create database temp;
  Use temp;
- ✓ Create table for storing temp records
  Create table temps(year INT, year INT)
- ✓ Load the data into the table
  LOAD DATA LOCAL INPATH '/home/cloudera/hive/data/Temperature.csv'
  OVERWRITE INTO TABLE temps;
- ✓ Describing metadata or schema of the table
  Describe temps
- ✓ Selecting data
  *Select * from temps;*

To perform the analysis on temperature datasets , we also used hive along with pig and mapreduce coding, using hive we performed the analysis by using above HIVE Query Language commands. Therefore the research paper, demonstrates that, to perform the quick analysis HIVE and PIG can be used than mapreduce, where we need to write lengthy coding, which is absent in the hive and pig.

V. CONCLUSION

It is not possible to handle Big data using traditional database management systems like relational databases. So we use some highly parallel software to handle big databases. Some components are also used to handle them. Firstly we have to acquire data from different sources which can be easily done by using components like Flume. Flume can directly fetch tweets from websites like twitter and store them in Bid databases. Then we can organize them using distributed file system like GFS and HDFS. At last they can be analyzed for faster access and query response. After analysis access and query response takes lesser time and effort on these big databases. Pig, Hive and MapReduce like components can do analysis in very short time. Hive can analyze a database of more than 8 lakh records in just 34 seconds. So all these components make it possible to handle and to use Big database in an easy and efficient manner.

REFERENCES

[1] Alexandros Biliris, "An Efficient Database Storage Structure for Large Dynamic Objects", IIEEE Data Engineering Conference, Phoenix, Arizona, pp. 301-308 , February 1992.

[2] An Oracle White Paper, "Hadoop and NoSQL Technologiesand the Oracle Database", February 2011.

[3] Cattell, " Scalable sql and nosql data stores", ACM SIGMOD Record, 39(4), pp. 12–27,2010.

[4] Russom, " Big Data Analytics", TDWI Research, 2011.

[5] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, "Google file system", 2003.

[6] Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach,Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber , Fay Chang, Jeffrey Dean , "Bigtable: A Distributed Storage System for Structured Data", OSDI 2006.

[7] Zhifeng YANG, Qichen TU, Kai FAN, Lei ZHU, Rishan CHEN, BoPENG, "Performance Gain with Variable Chunk Size in GFS-like File Systems", Journal of Computational Information Systems4:3 pp-1077-1084, 2008.

[8] Sam Madden , "From Databases to Big Data", IEEE Computer Society , 2012.

[9] Sanjeev Dhawan & Sanjay Rathee, "Big Data Analytics using Hadoop Components like Pig and Hive," American International Journal of Research in Science, Technology, Engineering & Mathematics, pp:1-5,2013.