A Review on Hive and Pig

Kadhar Basha J
Research Scholar, School of Computer Science,
Engineering and Applications,
Bharathidasan University
Trichy, Tamilnadu, India

Dr. M. Balamurugan,
Associate Professor, School of Computer Science,
Engineering and Applications,
Bharathidasan University
Trichy, Tamilnadu, India

***Abstract* -** From the last few decades it has been perceived that technology is growing to be progress at a quicker rate. Growth of technology leads to generation of large amount of data. Handling of huge size of data is an immense issue for the organizations, is becoming difficult using traditional database management techniques. MapReduce has received great attention due to the expeditious growth of unstructured data. Analysts need an economical, flexible, fault tolerance and feasible platform for huge volume of data processing eagerly. This paper grasps the comparative study of Hive and Pig. The processing time of hive and pig is implemented on a data set with simple queries. This paper includes the working of hive and pig and compares the performance of Hive and Pig in order to understand which platform is a better choice for data analysis.

Keywords—Pig; Hive; Big Data; Hadoop; Map Reduce; HDFS

I. INTRODUCTION

In the past, the types of information available were limited. But current era is the emerged source of data. We are surrounded by the data. This data is massive in volume and being produced exponentially from various sources, like Face book, Twitter, Mail systems, Research articles, Forums, YouTube, Online transactions, Healthcare systems etc., The data in the original form has multiple formats too. Also this data is changing over time at rapid speed, which means, it is no longer static in nature. This nature of current data put a lot of challenges before us, in the process of storage and computation. As a result, the traditional data storage and management techniques as well as computing tools and algorithms have become helpless to deal with these data. Despite of so many dares related with these data, we cannot ignore the potentials and possibilities lying in it that can support for analytics and for hidden pattern identifications.

Big data is massive collection of data over a decade that is so complicated and hard to processing and handling using traditional database management tools. Big data is not a single technology but a combination of existing and new technologies that helps organization gain actionable insight.[5]

Comparison of Traditional Data Vs Big Data is as follows:

The traditional data can save only slight amount of data varying from gigabytes to terabytes. However, big data can save and access massive amount of data which contains zettabytes of data or yottabytes of data. Traditional data utilize centralized database structure in which huge and complicated problems are resolve by a single computer system. Big data is depending on the distributed database structure where a huge block of data is decoded by dividing it into several smaller sizes. Traditional data is belonging to the structured data. Big data containing the semi-structured and unstructured data. In traditional data association between the data items can be traverse simply as the few number of information saved is small. However, big data hold massive or voluminous data which improve the level of issue in figuring out the relationship between the data items.[4]

The paper is organized as follows; the section II will present an outline of Hadoop Framework. The section III deals with the theoretical aspects of Hive. The section IV deals with the theoretical aspects of Pig. The section V represents the performance analysis of Hive and Pig using Inpatient Charge Dataset. Finally, conclusion will be given in section VI

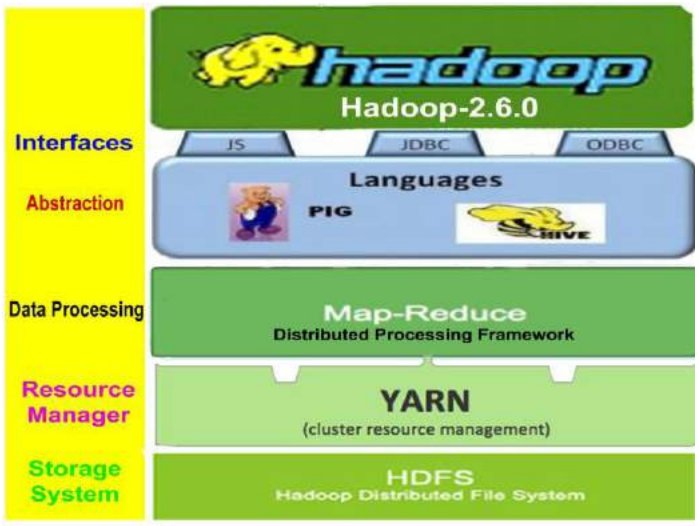


Fig 1: Hadoop Architecture

II. Hadoop Framework

Hadoop, a currently appeared Java-based software framework which helps applications that can execute a huge amount of data in an efficient manner for distributed application developers because it reduces complicated managements manner. Hadoop is a popular application of the Map Reduce model platform, which is an open-source maintained by the Apache Software Foundation.

Hadoop is designed to scale up from single servers to thousands of machines, each providing local computation and storage. Rather than depend on hardware to provide high-availability, the hadoop itself is designed to identify and manage failures at the application layer, so providing a highly-available service on top of a cluster of computers, each of which may be subject to failures. Hadoop enables running applications on systems with thousands of nodes with thousands of terabytes of data. The architecture framework of Hadoop is depicted in Fig 1.

A. Structure of Hadoop

Hadoop is especially designed for two key concepts: HDFS and MapReduce. Both are associated to distributed computation. Hadoop architecture is fundamentally a distributed master slave framework that made up of a single master and many slaves. The framework consists of the Hadoop Distributed File System (HDFS) for storage and MapReduce for computational ability. The functions of Hadoop in the framework are data partitioning and parallel computation of large datasets. Its storage and computational ability scale with the inclusion of hosts to a Hadoop cluster, and can attain volume sizes in the yottabytes on clusters with thousands of hosts.

B. HDFS

HDFS is a distributed, scalable, and portable file-system composed in Java for the Hadoop framework. It is a LINUX-based data storage layer of Hadoop. HDFS keeps huge files over multiple machines. HDFS is obtained from the idea of Google file system. A HDFS cluster primarily made up of a NameNode that handles the file system metadata and DataNodes that keep the actual data. The main characteristic of Hadoop is data partitioning and computation over thousands of hosts, and the execution of application computations in parallel manner. On HDFS, data files are reproduced as sequences of blocks in the cluster. When attaching commodity servers, a Hadoop cluster scales computation capacity, storage capacity, and I/O bandwidth. HDFS can be accessed from applications in many different ways.

Features of HDFS:

- ❖ HDFS is suitable for distributed storage and distributed processing using commodity hardware.
- ❖ It is fault tolerant, scalable, and particularly easy to widen.
- ❖ HDFS is extremely configurable with a default configuration suitable for many installations.
- ❖ Hadoop is composed in Java and is supported on all main platforms.
- ❖ Hadoop allows shell-like commands to communicate with HDFS directly.
- ❖ The NameNode and DataNodes have built in web servers that make it simple to verify the current status of the cluster

C. MapReduce

MapReduce is a programming prototype that is used for executing huge datasets distributed on a large cluster. MapReduce is the kernel of Hadoop. Its programming paradigm allows accomplishing huge data processing over thousands of servers configured with Hadoop clusters. This is obtained from Google MapReduce.[6]

It is software architecture for writing applications simply, which execute bulk amount of data in parallel on large clusters involving of thousands of nodes of commodity hardware in an authentic, fault-tolerant approach. The MapReduce paradigm is categorized into two phases, Map and Reduce that mostly deal with key and value pairs of data. The Map and Reduce task run sequentially in a cluster; the output of the Map phase turns the input for the Reduce phase. These phases are described as below:

Map phase: The datasets are first categorized and then allotted to the task tracker to perform the Map phase. The data functional operation will be executed over the data, producing the mapped key and value pairs as the output of the Map phase.

Reduce phase: The master node then accumulates the answers to all the sub problems and integrates them in some way to structure the output which is the solution to the problem it was initially trying to solve.

The framework of Hadoop made up of a Hadoop Distributed File System (HDFS) layer, which is used for the storage and execution of data and the execution model called MapReduce, both of which are mandatory for storing and executing huge volumes of data. Both HDFS and MapReduce are deployed on master slave architecture. A Hadoop cluster contains a single master and multiple slave nodes. The master node consists of a JobTracker, TaskTracker, NameNode and DataNode. A slave node reacts as both a DataNode and TaskTracker.[1]

III. HIVE

Hive is evolved on top of Hadoop as its data warehouse framework for querying and analysing of data that is kept in HDFS. Hive is an open source product that lets programmers analyze large data sets on Hadoop. Hive makes the job simple for accomplishing operations like analysis of large datasets.

Map Reduce doesn't provide the features of optimization and usability, but Hive framework provides those features. Hive's SQL-inspired language segregates the user from the complication of Map Reduce programming. It reuses known notions from the relational database paradigm, such as tables, rows, columns and schema, to ease learning.

The hive can utilize directory structures to partition data to make better performance on specific queries. Metastore is used for keeping schema information. This metastore typically presents in a relational database.

The user can communicate with hive using various ways; those are Web GUI and Java Database Connectivity (JDBC) interface. Most communications incline to take place over a command line interface (CLI). Hive extends a CLI to write hive queries using Hive Query Language (HQL). In general, HQL syntax is identical to the SQL syntax.

Hive supports four file formats such as TEXTFILE, SEQUENCEFILE, ORC and RCFILE (Record Columnar File). In a single user scenario, hive utilizes derby database for metadata storage and for multi user scenario, hive utilizes MYSQL to store metadata.

Important difference between HQL and SQL is that, hive query processes on the Hadoop infrastructure rather than conventional database. Hadoop is a distributed storage, so when we submit hive query, it will appeal on huge data sets. The data sets are so large that high-end, expensive, conventional databases would fail to accomplish operations.

Hive supports partition and buckets concepts for simple access of data when client executes the query. Hive supports custom specific User Defined Functions for data cleansing and filtering. [7]

A. *Hive Architecture*

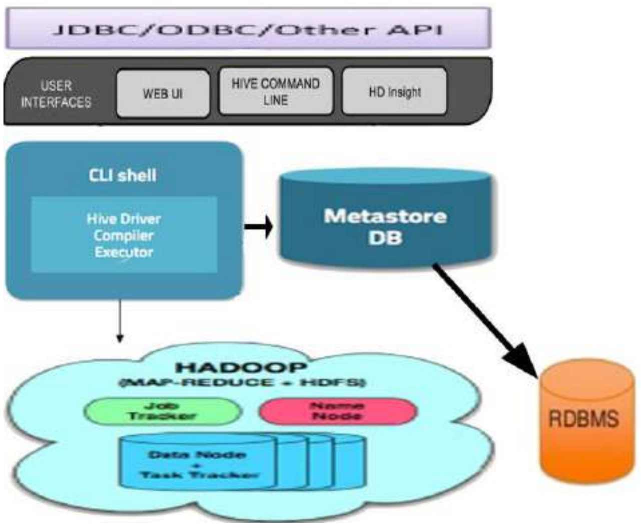


Fig 2: Hive Architecture

There are four major components in Hive. They are User Interfaces, CLI shell, Meta Store, Hadoop that is represented in Fig 2.

B. *Hive Functionality.*

Hive Server is an API that enables the clients to process the queries on hive data warehouse and obtain the required results. Under hive server driver, compiler and execution engine communicate with each other and execute the query.

The client posts the query via a GUI. The driver accepts the queries in the first instance from GUI and it will define session handlers, which will fetch required APIs that is modelled with different interfaces like JDBC or ODBC. The compiler generates the plan for the job to be processed. Compiler in turn is in touch with matter and it obtains metadata from Meta Store.

Execution Engine is the vital component here to process a query by directly interacting with Job Tracker, *Name Node* and *Data nodes.* By running hive query at the backend, it will produce a series of Map Reduce Jobs. In this scenario, the execution engine acts like a bridge between hive and Hadoop to execute the query. For HDFS operations, Execution Engine communicates Name Node.

At the end, it is going to fetch required results from Data Nodes. It will be having duplex communication with Metastore.

C. *Modes of Hive Execution*

Hive employs in two modes. They are Interactive mode and Non Interactive mode. In Interactive mode, it directly moves to hive shell when you enter hive in command shell. The non interactive mode is about processing commands directly in console file mode. The user can create two types of tables – Internal table and External table in both modes. The execution process of Hive with Hadoop environment is represented in Fig 3.

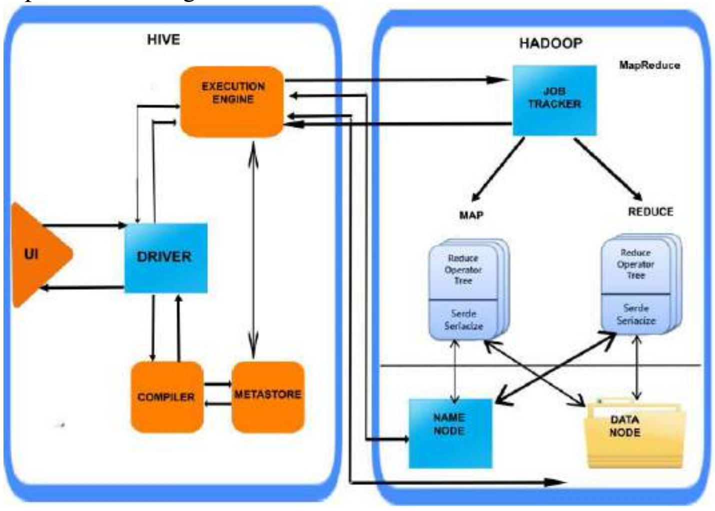


Fig 3: Structure of Execution

D. Hive Data Modelling

Hive works with two types of table structures - internal and external, rely on the design of schema and how the data is getting placed into Hive

I. Internal Table:

- Create tables
- Load the data.

By dropping this type of table, both data and schema will be ousted. The stored location of this table will be at /user/hive/warehouse.

II. External Table:

Data will be obtainable in HDFS; the table is going to get created with HDFS data. By dropping the table, it removed only schema, data will be obtainable in HDFS as before. External tables give a choice to create multiple schemas for the data kept in HDFS instead of dropping the data every time whenever schema upgrades.[3]

IV. PIG

Pig issues the data flowing scenario for executing large sets of data. It is another abstraction on top of Map Reduce (MR). Pig supports parallelization mechanism. For implementation purpose, it provides Pig Latin language.

Pig accepts Pig Latin scripts and turns it into a series of Map Reduce jobs. Pig scripting possess benefits of running the applications on Hadoop from the client side. The nature of programming is simple, compared to low level languages such as Java. It issues simple parallel execution, the user can write and use its own custom defined functions to execute unique processing.

Pig Latin provides various operators like LOAD, FILTER, SORT, JOIN, GROUP, FOREACH and STORE for executing relational data base operations. The users apply data transformation tasks with few lines of codes. Compared to Map Reduce code, the Pig Latin codes are very fewer in lines and provide improved flexibility.

A. Apache Pig Architecture

Pig Architecture contains Pig Latin Interpreter and it will be processed on client Machine. It utilizes Pig Latin scripts and it transforms the script into a series of Map Reduce jobs. Then it will process Map Reduce jobs and stores the output into HDFS. Meanwhile, it executes various operations such as Parse, Compile, and Optimize and plans the execution on data that comes into the system. The architectural framework of Pig is shown in Fig 4.[8]

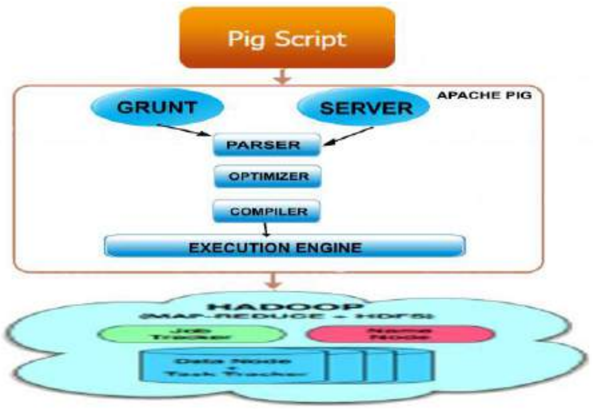


Fig 4: Pig Architecture

B. Pig Functionality

When Apache Pig programmer develops scripts, they are saved in the local file system in the kind of user defined functions. When we post Pig Script, it comes in touch with Pig Latin Compiler which breaks the task and executes a series of Map Reduce jobs; meanwhile Pig Compiler gets data from HDFS. After executing Map Reduce jobs, the output file is saved in HDFS.

C. Modes of Pig Execution

Pig can execute in two modes. The modes based on where the Pig script is going to execute and where the data is going to reside. The data can be saved in a single machine, i.e. local file system or it can be saved in a distributed environment like classic Hadoop Cluster environment. The user can execute Pig scripts in three different modes such as non interactive shell, grunt shell and embedded mode.

In non interactive mode, the user must create a file, load the code in the file and execute the script. In grunt mode, it acts as an interactive shell for running Pig commands. In embedded mode, the user use JDBC to run SQL programs from Java.

1) Pig Local mode

In this mode, the pig executes on stand-alone JVM and accesses the local file system. This mode is well suited for handling with the smaller data sets. In this mode, parallel mapper execution is impossible because the prior versions of the Hadoop are not thread safe. By issuing –x local command, the user can enter in to Pig local mode of execution. In this mode, Pig always focuses for the local file system path when data is loaded. $pig –x local indicates that it's in local mode.

2) Pig MapReduce mode

In this mode, the user should have complete Hadoop cluster setup and Hadoop installations on it. By default, the pig executes on Map Reduce mode. Pig transforms the posted queries into Map reduce jobs and executes them on top of Hadoop cluster. The user can say this mode as a Map reduce mode on a fully distributed cluster. Pig Latin statements like LOAD, STORE are used to fetch data from the HDFS file system and to produce output. These Statements are used to execute data.

3) Storing Results

Duration of the processing and execution of Map Reduce jobs, intermediate data will be produced. Pig saves this data in a temporary location on HDFS storage. In order to save this intermediate data, temporary file location has to be created inside HDFS. By using DUMP, the user can obtain the final results presented to the output screen. In a production environment, the output will be saved using STORE operator.

D. Pig Data Modelling

Pig uses two types of data types, namely scalar and complex. Scalar values represent single values, whereas complex is combinations of other types. With regard to scalar types Pig uses the usual numeric suspects in its data types, namely int, long, float, and double. These types are depicted to their equivalent Java data types, which issues the same constraints that Java places on them. To handle string data, Pig uses the chararray type, which depicts to Java strings and it supports Unicode characters as well. The final scalar data type is the bytearray which can be used to support arrays of bytes. It also depicts to the Java data type DataByteArray. Pig's complex data types consists of maps, tuples, and bags. Maps are the classic map type found in various programming environments, but with a few restriction. Tuples function is similar to rows in an SQL database, and as such can support schemas. Bags are unordered collections of tuples and are distinctive among the Pig data types in that they alone aren't need to fit completely into memory at once.[2]

V. PERFORMANCE ANALYSIS OF HIVE AND PIG

Analysis of data is done using Inpatient Charge Data set. It consists of 1,63,066 records and its size is 14.3 mb. It has huge amount of data include hospital-specific charges for the more than 3,000 U.S. hospitals that collect Medicare Inpatient Prospective Payment System payments for the top 100 most frequently billed discharges, paid under Medicare based on a rate per discharge using the Medicare Severity Diagnosis Related Group for Fiscal Year 2011[9]
The data description is as shown in Table 1.

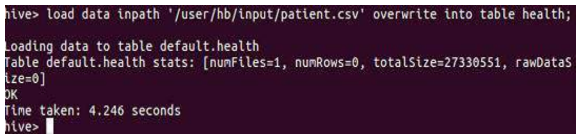
TABLE 1: INPATIENT CHARGE DATA SET

Attribute	**Description**
DRG Definition	DRGs are a classification system that groups similar clinical conditions (diagnoses) and the procedures furnished by the hospital during the stay.
Provider ID	Provider Identifier billing for inpatient hospital services.
Provider Name	Name of the Provider.
Provider Street Address	Street address in which the provider is physically located.
Provider City	City in which the provider is physically located.
Provider State	State in which the provider is physically located.
Provider Zip Code	Zip Code in which the provider is physically located.
Hospital Referral Region	HRR in which the provider is physically located.
Total Discharges	The number of discharges billed by the provider for inpatient hospital services.
Average Covered Charges	The provider's average charge for services covered by Medicare for all discharges in the DRG. These will vary from hospital to hospital because of differences in hospital charge structures.
Average Total Payments	The average of Medicare payments to the provider for the DRG including the DRG amount, teaching, disproportionate share, capital, and outlier payments for all cases.
Average Medicare Payments	The average of all payments to the provider for the DRG.

A. Analysis with Hive

The following steps are followed in the analysis phase:

1. Load the data set into table 'health' as shown in Fig 5:

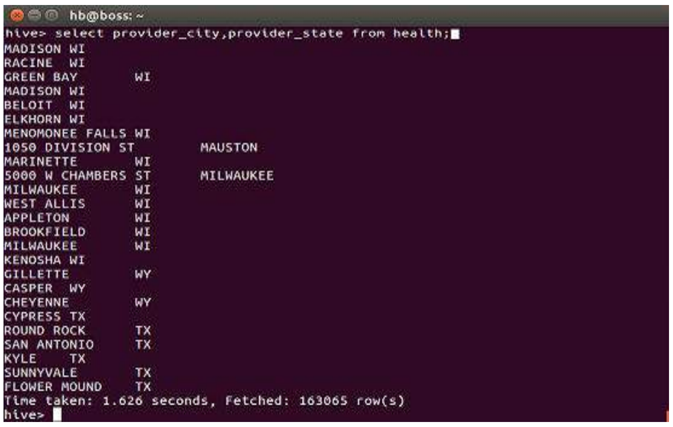


```
hive> load data inpath '/user/hb/input/patient.csv' overwrite into table health;

Loading data to table default.health
Table default.health stats: [numFiles=1, numRows=0, totalSize=27330551, rawDataS
ize=0]
OK
Time taken: 4.246 seconds
hive>
```

Fig 5: Loading Data set in Hive

2. Analyze the data set using select operation as shown in Fig 6:



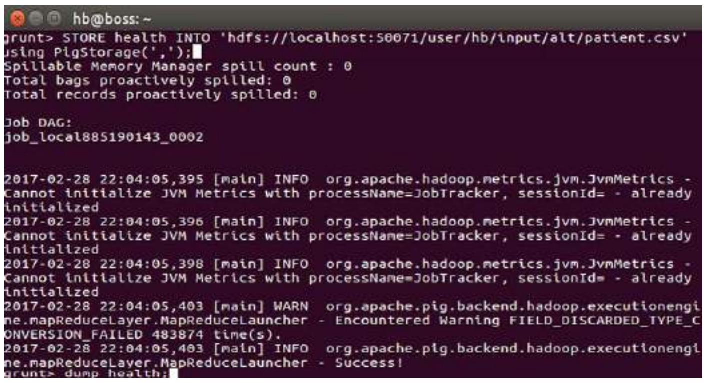
```
hb@boss: ~
hive> select provider_city,provider_state from health;
MADISON WI
RACINE  WI
GREEN BAY       WI
MADISON WI
BELOIT  WI
ELKHORN WI
MENOMONEE FALLS WI
1050 DIVISION ST        MAUSTON
MARINETTE       WI
5000 W CHAMBERS ST      MILWAUKEE
MILWAUKEE       WI
WEST ALLIS      WI
APPLETON        WI
BROOKFIELD      WI
MILWAUKEE       WI
KENOSHA WI
GILLETTE        WY
CASPER  WY
CHEYENNE        WY
CYPRESS TX
ROUND ROCK      TX
SAN ANTONIO     TX
KYLE    TX
SUNNYVALE       TX
FLOWER MOUND    TX
Time taken: 1.626 seconds, Fetched: 163065 row(s)
hive>
```

Fig 6: Select Operation in Hive

B. Analysis with Pig

The following steps are followed in the analysis phase:

1. Load the data set into table 'health as shown in Fig 7:



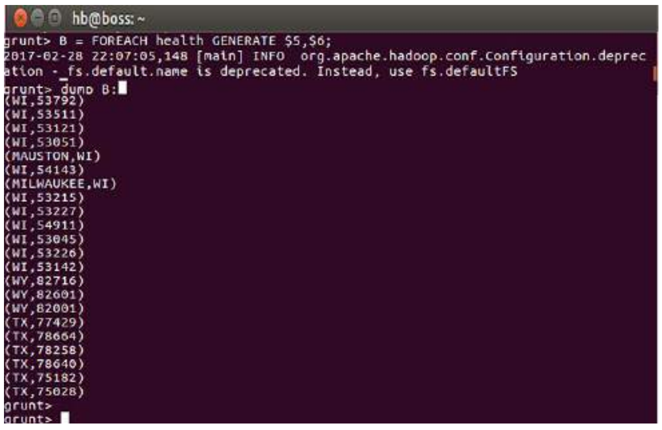
```
hb@boss: ~
grunt> STORE health INTO 'hdfs://localhost:50071/user/hb/input/alt/patient.csv'
using PigStorage(',');
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_local885190143_0002


2017-02-28 22:04:05,395 [main] INFO  org.apache.hadoop.metrics.jvm.JvmMetrics -
Cannot initialize JVM Metrics with processName=JobTracker, sessionId= - already
initialized
2017-02-28 22:04:05,396 [main] INFO  org.apache.hadoop.metrics.jvm.JvmMetrics -
Cannot initialize JVM Metrics with processName=JobTracker, sessionId= - already
initialized
2017-02-28 22:04:05,398 [main] INFO  org.apache.hadoop.metrics.jvm.JvmMetrics -
Cannot initialize JVM Metrics with processName=JobTracker, sessionId= - already
initialized
2017-02-28 22:04:05,403 [main] WARN  org.apache.pig.backend.hadoop.executionengi
ne.mapReduceLayer.MapReduceLauncher - Encountered Warning FIELD_DISCARDED_TYPE_C
ONVERSION_FAILED 483874 time(s).
2017-02-28 22:04:05,403 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.mapReduceLayer.MapReduceLauncher - Success!
grunt> dump health;
```

Fig 7: Loading data set in pig

2. Analyze the data set using select operations as shown in Fig 8:



```
grunt> B = FOREACH health GENERATE $5,$6;
2017-02-28 22:07:05,148 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> dump B:
(WI,53792)
(WI,53511)
(WI,53121)
(WI,53051)
(MAUSTON,WI)
(WI,54143)
(MILWAUKEE,WI)
(WI,53215)
(WI,53227)
(WI,54911)
(WI,53045)
(WI,53226)
(WI,53142)
(WY,82716)
(WY,82601)
(WY,82001)
(TX,77429)
(TX,78664)
(TX,78258)
(TX,78640)
(TX,75182)
(TX,75028)
grunt>
grunt>
```

Fig 8: Select operation in Pig

The numerical experimental results of Pig and Hive are summarized in Table 2 according to Time taken for each operation.

TABLE 2: EXPERIMENTAL RESULTS OF HIVE AND PIG

Data Analysis	Hive	Pig
Creating Table	3.139 seconds	2 seconds
Loading/Storing data	4.246 seconds	1 min 27 seconds
Querying data set	1.626 seconds	14 seconds

From the above table, we observed that Time taken by Hive is lesser than Time taken by Pig in all aspects.

Difference between Hive and Pig:

1. Hive is used for fully structured data but Pig is used for semi-structured data.
2. Hive is mostly used for generating reports but Pig is mostly used for programming.
3. Hive works on the server part of any cluster whereas Pig operates on client part of any cluster.
4. Hive makes use of accurate form of the SQL DLL language by defining the tables prior to and save the database information in any local database but in case of Pig there is no require of metadata and the schemas are defined in the script itself.
5. Hive has a provision of partition in order to execute the subset of data in an alphabetical order but in case of Pig no provision of partition but can be attained by using filters.
6. Pig can be installed comfortably as compared with Hive as Pig is rooted with shell interaction

VI. Conclusion

In this paper, a comparison between Hive and Pig has been carried out, to see how well each platform performs during data analysis. The execution time of pig and hive are different in terms of seconds. From the presented time measurements it is quite evident that Hive outperformed Pig for loading and querying the dataset. However, this study was only performed on hive and pig. Future work is in implementing map reduce paradigm with HBase for big data analytics.

REFERENCES

[1] Vibha Sarjolta, Dr. A.J Singh, "A Study of Hadoop: Structure and Performance Issues," International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 6, June 2015, ISSN: 2277 128X.

[2] http://a4academics.com/tutorials/83-hadoop/837-hadoop-pig

[3] http://a4academics.com/tutorials/83-hadoop/836-hadoop-hive

[4] https://www.projectguru.in/publications/difference-traditional-data-big-data/

[5] Vibhavari Chavan, Rajesh, N. Phursule, " Survey Paper On Big Data", International Journal of Computer Science and Information Technologies, Vol 5(6), 2014, ISSN : 0975-9646.

[6] Johnu George et al., "Hadoop MapReduce for Tactical Clouds, 2014, IEEE 3rd International Conference on Cloud Networking.

[7] Jay Mehta, Jongwook Woo, "Big Data Analysis of Historical Stock Data Using Hive", ARPN Journal of Systems and software, vol 5, No 2, August 2015, ISSN : 2222-9833.

[8] Dr. Urmila R. Pol, "Big Data Analysis: Comparison Of Hadoop MapReduce, Pig and Hive International Journal of Innovative Research in Science, Engineering and Technology, vol 5, Issue 6, June 2016, ISSN : 2319- 8753.

[9] Inpatient Charge data set available at https://www.data.gov/health/