RESEARCH ARTICLE OPEN ACCESS

Analyzing Performance of Map Reduce, Pig Latin and Hive on Windows Platform

*Mr.Manishkumar R Solanki[1], Yashvi Shah[2], Siddhi Shukla[3], Shrusti Talati[4]
[1]*Sr. Lecturer, Information Technology Department, Shri Bhagubhai Mafatlal Polytechnic, Mumbai*
[2]*Diploma in Computer Engineering, Shri Bhagubhai Mafatlal Polytechnic, Mumbai*
[3]*Diploma in Computer Engineering, Shri Bhagubhai Mafatlal Polytechnic, Mumbai*
[4]*Diploma in Computer Engineering, Shri Bhagubhai Mafatlal Polytechnic, Mumbai*
Corresponding author: Mr.Manishkumar R Solanki

ABSTRACT
The Hadoop framework allows distributed processing of large data sets across clusters of commodity computers efficiently. MapReduce, the core programming language of the Hadoop Ecosystem processes the data stored in Hadoop Distributed File System (HDFS). It is difficult for non programmers to work with MapReduce. Hadoop supports HiveQL (SQL like statements) which implicitly and immediately translates the queries into one or more MapReduce jobs. To help procedural language developers, Hadoop supports Pig Latin language. This paper runs a text data processing application with MapReduce, Hive and Pig on single node windows platform and compares performance in graphical form.
Keywords: *Big data, Distributed Processing, MapReduce, Hive, Pig*

Date of Submission: 29-08-2017 Date of acceptance: 13-09-2017

I. INTRODUCTION

In the age of big data, the volume of data which we produce in day-to-day life has grown beyond the capacity of storage and processing of a single node. Big data brings the challenge of storing and processing these voluminous data to get competitive advantage in the global digital market place. Hadoop fills this gap by providing storage and computational capabilities for huge data effectively. It's a distributed system made up of a distributed file system and it offers a way to parallelize and execute programs on a cluster of machines[1] Hadoop, a Java based open source framework, uses scale out approach for running distributed applications to exploit the power of commodity hardware rather than high end nodes. This paper runs a text processing application on MapReduce, Hive and Pig on single node windows platform and compares performance in form of bar charts. The remaining paper is organized as follows: Section II describes the languages. Section III discusses about development and execution of text processing application with MapReduce, Hive and Pig. Section V concludes the paper.

II. HADOOP LANGUAGES: MAPREDUCE, PIG LATIN AND HIVE

The Hadoop framework allows distributed processing of large structured, unstructured or semi-structured datasets across clusters of commodity computers with great performance.

Map Reduce: The MapReduce language establishes a base for Hadoop Eco System. It processes Hadoop Distributed File System (HDFS) on large clusters which are made of thousands of commodity hardware in a reliable and fault-tolerant manner. The operations of MapReduce are performed in Map and Reduce functions. The Map function works on a set of input values and transforms them into a set of key/value pairs. The reducer receives all the data for an individual "key" from all the mappers and applies Reduce function to achieve the final result.

- **Pig:** The Pig toolkit consists of a compiler that generates MapReduce programs, bundles their dependencies, and executes them on Hadoop. Pig jobs are written in a data flow language called Pig Latin and can be executed in both interactive and batch fashions.[2] Pig does not require the schema for data like SQL, so it is well suited to process unstructured data.

- ***Hive:*** Hive is a data warehousing package built on top of Hadoop. Hive's SQL-inspired language, better known as HiveQL or HQL separates the user from the complexity of MapReduce programming.[3] This approach makes it very fast and adoptable for people that are already familiar with the syntax of SQL. The HQL queries are implicitly translated into one or more MapReduce jobs to process the HDFS.

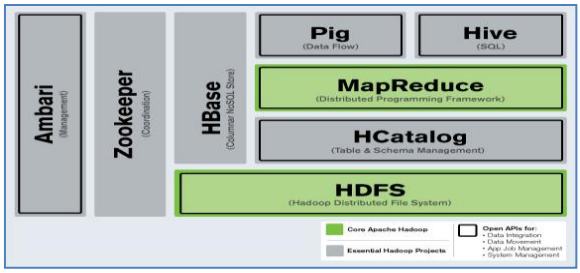

Fig.1 MapReduce, Pig & Hive on Hadoop Framework [4]

III. EXECUTION OF WORD COUNT APPLICATION WITH MAPREDUCE, HIVE AND PIG

In this section we are executing word count text processing application using MapReduce, Pig and Hive with three different sizes of data (text files) as follows:

Case1: file1.txt having 154Kb (Small Size)
Case2: file1.txt having 154Kb & file2.txt having 406Kb thus total 560Kb (Medium Size)
Case3: file3.txt having 2099Kb (Large Size)

- *WORDCOUNT USING MapReduce*: The execution process of word count text processing application is described in our paper. [5]
- *WORDCOUNT USING PIG LATIN* The following steps describes the process of executing small size text file (file1.txt 154Kb) using pig language:

A. Create a directory (say ' input154') in HDFS to keep all the text files to be used for counting words.

c:\hadoop-2.7.1\bin>hadoop fs -mkdir /input154

B. Laod file to HDFS c:\hadoop-2.7.1\bin> hadoop fs -put C:\stories\story154kb.txt /storyinput1/file1.txt

C. To run pig script type following:

- Sh
- Pig

Fig 2 shows that we have entered into pig scripts (grunt mode)

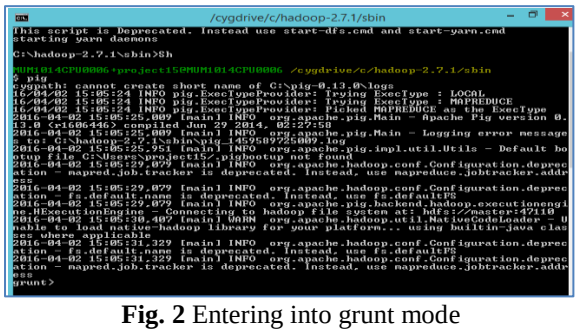

Fig. 2 Entering into grunt mode

D. Load input from the file or folder named input1, and call the single field in the record 'line'.

Lines = LOAD '/input154 ' AS (line:chararray);

E. TOKENIZE splits the line into a field for each word. Flatten will take the collection of records returned by TOKENIZE and produce a separate record for each one, calling the single field in the record word.

words = FOREACH Lines GENERATE FLATTEN(TOKENIZE(line)) as word;

F. Now group them together by each word:

grouped = GROUP words BY word;

G. Count them.

wordcount = FOREACH grouped GENERATE group, COUNT(words);

Print out the results.

DUMP wordcount;

Fig.3 shows the conversion of pig latin script into MapReduce job and Fig. 4 gives the final outcome of count of individual words.

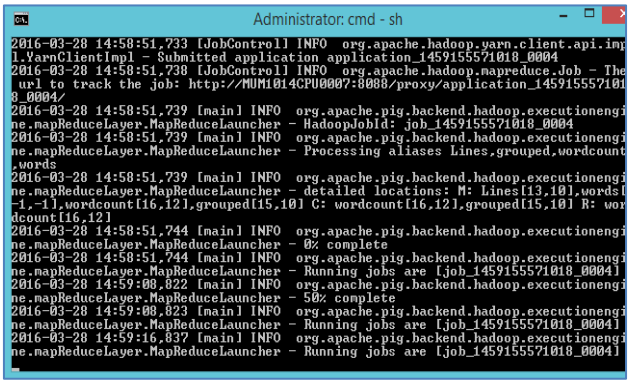

Fig. 3 Conversion from Pig Latin to MapReduce job

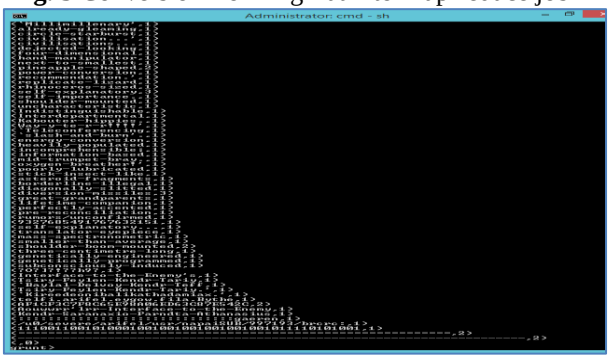

Fig. 4 Final result of word count with Pig

Similar operation was performed for case2 and case3.

- *WORDCOUNT USING HIVE:* The following steps describes the process of executing small size text file (file1.txt 154Kb) using pig language

A. Create a directory (say ' hiveinput1') in HDFS to keep all the text files to be used for counting words.

c:\hadoop-2.7.1\bin>hadoop fs -mkdir /hiveinput154

B. Laod first file to HDFS

c:\hadoop-2.7.1\bin> hadoop fs -put c:\stories\story154kb.txt /storyinput1/file1.txt

C. Enter into hive
c:\hadoop-2.7.1>hive

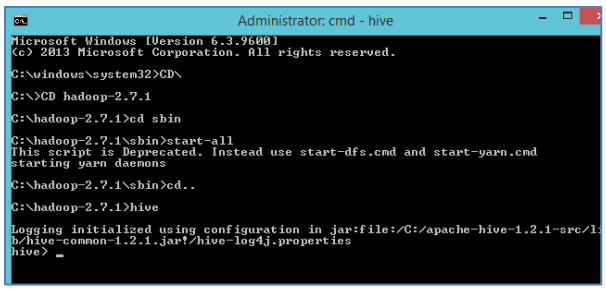

Fig. 5 Entering into hive mode

D. Create table named as hive1
hive>CREATE TABLE hive1 (line STRING);
E. Load full file into table
hive>LOAD DATA INPATH '/input154' OVERWRITE INTO TABLE hive1;
F. Now we have to convert it into words by applying space as delimiter. Use split function of hive. Convert every line of data into multiple rows using explode which generates one intermediate table. Apply group by to count word occurrences.

hive> CREATE TABLE word_count_hive1 AS
SELECT word, count(1) AS count FROM (SELECT explode(split(line, ' ')) AS word FROM hive1) w GROUP BY word
ORDER BY word;

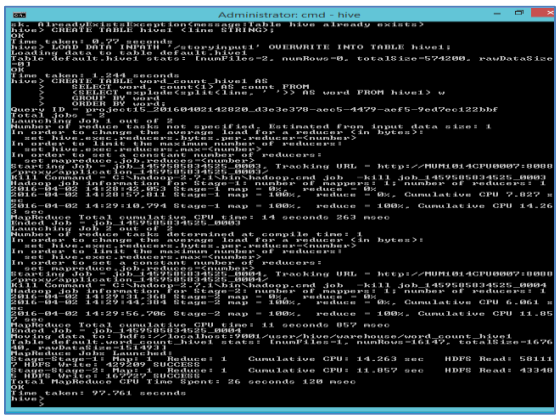

Fig. 6 Conversion of Hive query into MapReduce job

G. Gives all the columns in specified table mentioned in the query.
hive> desc word_count_hive1;
H. Following command will list all the tables in HDFS
hive> show tables;
I. To view output of hive: (word_count_hive1 is name of table)
hive> select* from word_count_hive1;

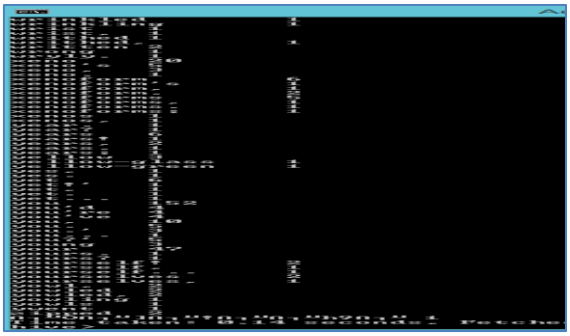

Fig. 7 Final result of word count with Hive

IV. GRAPHICAL REPRESENTATION OF COMPARISION

In this section, we are comparing the performance of MapReduce, Pig Latin and Hive based on their processing time. We could see this time by typing http://localhost:8088 on web browser as shown in Fig. 8. Port 8088, gives information about the cluster i.e. used for viewing currently running jobs, completed ones etc. On this web page there are two columns start time & finish time so we took difference of that and plotted bar charts as shown in Fig. 9, 10 and 11.

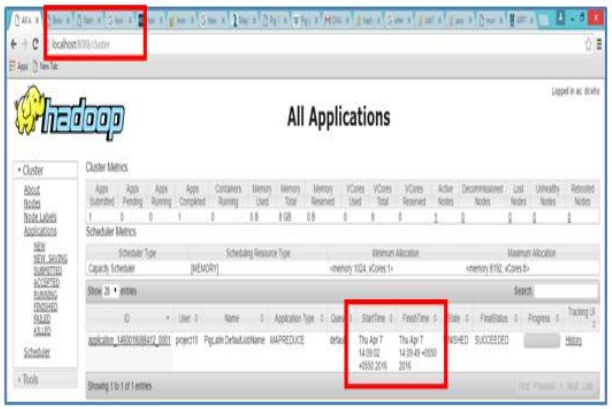

Fig. 8 Cluster information: StartTime & FinishTime

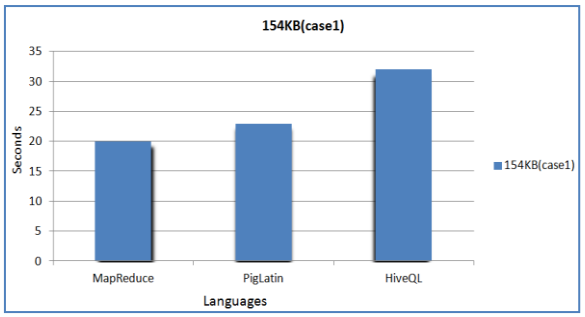

Fig. 9 Performance comparison for 154KB size

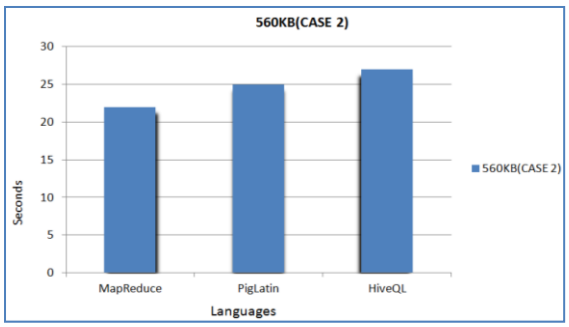

Fig. 10 Performance comparison for 560KB size

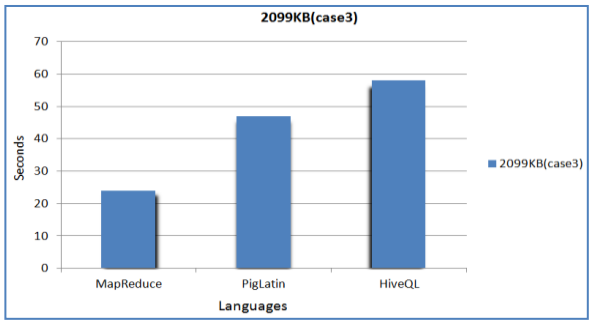

Fig. 11 Performance comparison for 2099KB size

V. CONCLUSION

The MapReduce language is the core programming language of Hadoop. So if we write applications with MapReduce and execute them, we get better performance. Pig Latin language provides flexibility to data flow programmers to work with Hadoop. Hive has gained popularity due to support of SQL like queries. Pig Latin and HiveQL statements take much time to execute because they are ultimately translated into MapReduce jobs. The flexibility of Pig Latin and Hive is achieved at the cost of performance.

REFERENCES

[1]. Alex Holmes, Hadoop In Practice by Manning Publications, ISBN 9781617290237

[2]. Garry Turkington, Gabriele Modena, Learning Hadoop 2, Design and implement data processing, lifecycle management, and analytic workflows with the cutting-edge toolbox of Hadoop 2, Packt Publishing Ltd.

[3]. Chuck Lam, Hadoop In Action, MEAP Edition Manning Early Access Program Derrick Harris, Hadoop: Bigger than SpringSource, JBoss and MySQL combined?

[4]. https://gigaom.com/2012/02/27/hadoop-bigger-than-spring-jboss-and-mysql-combined/

[5]. Manishkumar R Solanki, Yashvi Shah, Siddhi Shukla, Shrusti Talati, Configuring Hadoop On Windows Platform And Runninga Mapreduce Application To Process Text Data On A Single Node, IJRET: International Journal of Research in Engineering and Technology eISSN: 2319-1163|pI : 2321-7308

International Journal of Engineering Research and Applications (IJERA) is **UGC approved** Journal with Sl. No. 4525, Journal no. 47088. Indexed in Cross Ref, Index Copernicus (ICV 80.82), NASA, Ads, Researcher Id Thomson Reuters, DOAJ.

Mr.Manishkumar R Solanki. "Analyzing Performance of Map Reduce, Pig Latin and Hive on Windows Platform ." International Journal of Engineering Research and Applications (IJERA) , vol. 7, no. 9, 2017, pp. 32–35.