

Comparative Analysis of MapReduce, Hive and Pig

¹Sunny Kumar, ²Eesha Goel

Computer Science and Engineering, Giani Zail Singh Campus CET, Bhatinda, India

Computer Science and Engineering, Giani Zail Singh Campus CET, Bhatinda, India

Email: ¹sunnykumar1018@gmail.com ²eesha1992@rediffmail.com

Abstract: The size of data being handled by many applications is becoming alarmingly large and unmanageable by conventional database management techniques. This paper leverages the comparative study of Hadoop's programming paradigm (Map reduce) and Hadoop's ecosystems Hive and Pig. The processing time of map reduce, hive and pig is implemented on a data set with simple queries. It is observed that Pig processes the data in shorter time as compared with Map reduce and Hive. It is not necessary that only Pig is useful other techniques are also useful under different constraints

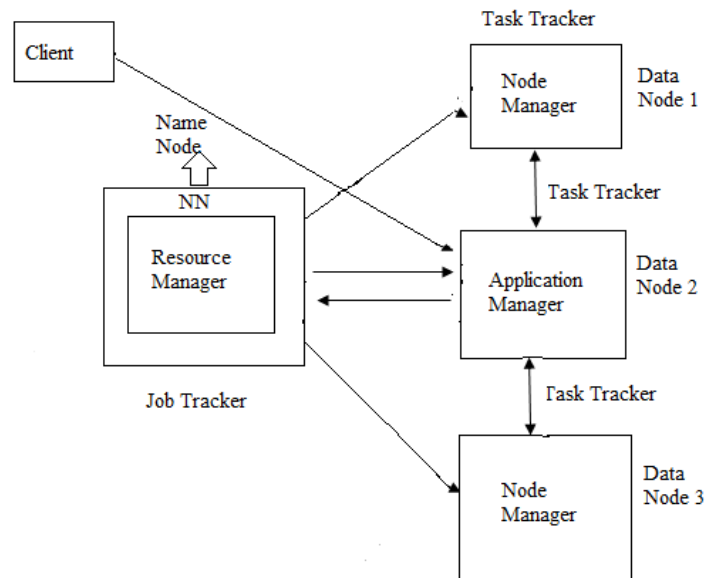
Keywords: Map Reduce, Hadoop, Hive, Hive Server2, Thrift Server, Pig, Pig-Latin.

1. Introduction

From the last few decades it has been observed that technology is becoming advance at a faster rate. Growth of technology leads to generation of large amount of data. Processing of big amount of data is a huge problem for the organizations. So data analysis is very crucial for business intelligence. Raw data, unstructured data and semi-structured data is produced by them which is not stored in data warehouse. Traditional data models cannot store and process large amount of data. So we need such systems that are more flexible, scalable, fault-tolerance, compatible and cheap to process large amount of data. The Apache Foundation Software provides such a framework for us that is Hadoop Platform. It is specially mend for large data sets. Hadoop uses map reduce concept to process data in a cluster. Map Reduce has been designed for development of large scale, distributed and fault tolerant data processing applications. These applications are utilizing by governments and industrial organizations. In map reduce jobs are written that consist of a map and reduce functions and the framework of Hadoop. Hadoop handles all details of paralysing the work, schedules various parts of job on different-different nodes in cluster, monitors and recovers from various failures. This paper includes the working of hive and pig and shows how pig queries are more powerful and taking less time to perform map and reduce task.

A. Features of MapReduce

1. **Simplicity of Development for Various Applications:** the idea behind map reduce framework is rarely simple: no socket programming, no threading or any special synchronization logic, no special techniques to deal with large amount of data. The architecture of Hadoop takes care of all things. The main job of developers is to use functional programming concept to build data processing applications that operate on data at a time. Map reduce operate on records (it may be a single line, single block of data, single file) and produce intermediate key value pairs.[13] The reduce function operate on these intermediate key value and processing all values that have same key together and give the final output.
2. **Scalable:** Since various tasks run parallel on different machines in a cluster they do not communicate with each other explicitly and do not share their state. Additional machines can be easily added to cluster and applications takes advantage of additional hardware.
3. **Automatic Parallelization and Distribution of Work:** Developers only focus on the map and reduce functions that process various records. The distribution of work among various nodes and splitting of a job into multiple tasks allthese responsibilities are taken care by Hadoop framework.
4. **Fault Tolerance:** It is the frame work that takes care of any failure that may occur during processing of data across many nodes in a cluster. [5]



2. Functioning of Hadoop

Job Allocation in Hadoop Framework

In Fig 1 we are describing how the job is allocated in Hadoop. When the client sends the request it will pass to the Resource Manager which manages all the requests. Then that request will be send to the Application Manager. Resource Manager consists of Name Node which is the master of Hadoop Architecture. [12] Name Node is responsible for managing the file namespace. Job Tracker manages all the client requests and passes them to the appropriate Task Tracker that are Data Nodes. Data Nodes are consist of blocks of file and they also report to Name Node what kind of blocks they have stored. There are many Task Trackers in Hadoop cluster that can communicate with each other and help for parallel processing. [2]

Fig 1. Job Allocation in Hadoop Framework

B. Daemons in Hadoop Map reduce: The Job Tracker and Task Tracker:

C. Job Tracker

The Job Tracker is a master process, it accepts job from client and schedule it on different worker node in a cluster and provides administrative functions such as status of each and every worker node and task progress report to cluster. There is only one job tracker per map reduce cluster and usually runs on a reliable hardware since it is a single point of failure if this master process fails that will result in failure of all running jobs. Just similar to data node and name node in HDFS task trackers inform to job tracker about their current health and status by way of regular heartbeats.[11]

D. Task Tracker

The second daemon is the task tracker it accepts jobs from job tracker, execute those jobs locally and send report back to job tracker periodically. There is one task tracker on each worker node. Both data node and task trackers run on the same machines, which makes both a compute and storage respectively.

E. Map Task Execution

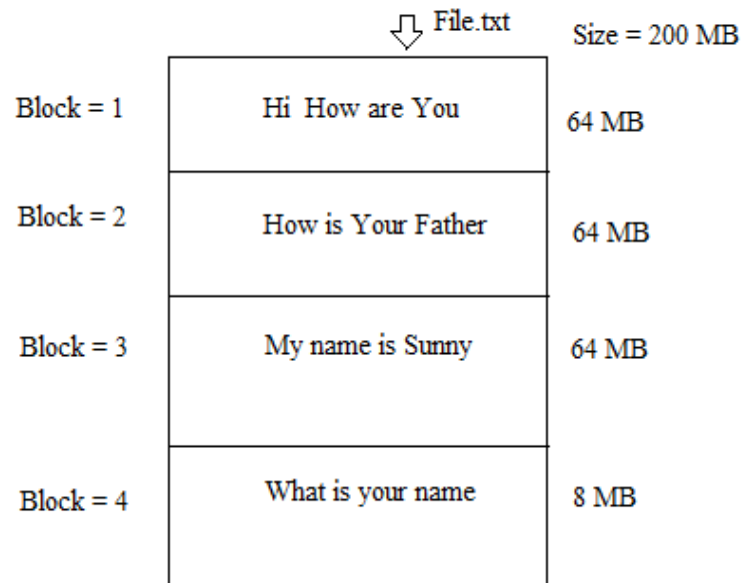


Fig 2. Splitting of files into blocks

Map reduce framework works with HDFS that is Hadoop Distributed File System. Internally, file is being split up into number of blocks. Each block size is 64 MB. These blocks are stored in a set of Data Nodes that manages the storage attached to the nodes that are running.

As shown in Fig 2 text file named as File.txt of size 200 MB is taken. This file will be split up into 4 blocks. First 3 blocks were of size 64 MB but 4th block will be of size 8 MB ($64 \times 3 = 192$ and $200 - 192 = 8$). The content distribution of file to the blocks are managed by Hadoop Distributed File System.[2]

Map Reduce is a framework which is used for processing data parallelly. In Map Reduce jobs are controlled by Job Tracker. Each node will consist of Task Tracker which is responsible for processing the map tasks. These tasks are assigned to it by Job Track Input Split itself as shown in fig 3. These tasks are processed according to the size that is larger size will be processed first so that job running time could be optimized. Input splits are used with the help of Input Format class which selects file as an input and after that file is being splitted into tasks. On Task Tracker the map task passes the split to Record Reader Interface on Input Format to Map task uses this in order to generate record key-value pairs, then passes to the map function. Number of Mappers will be equal to number of splits. The output generated by Mapper will be in the form of <key, value> pair. [10]

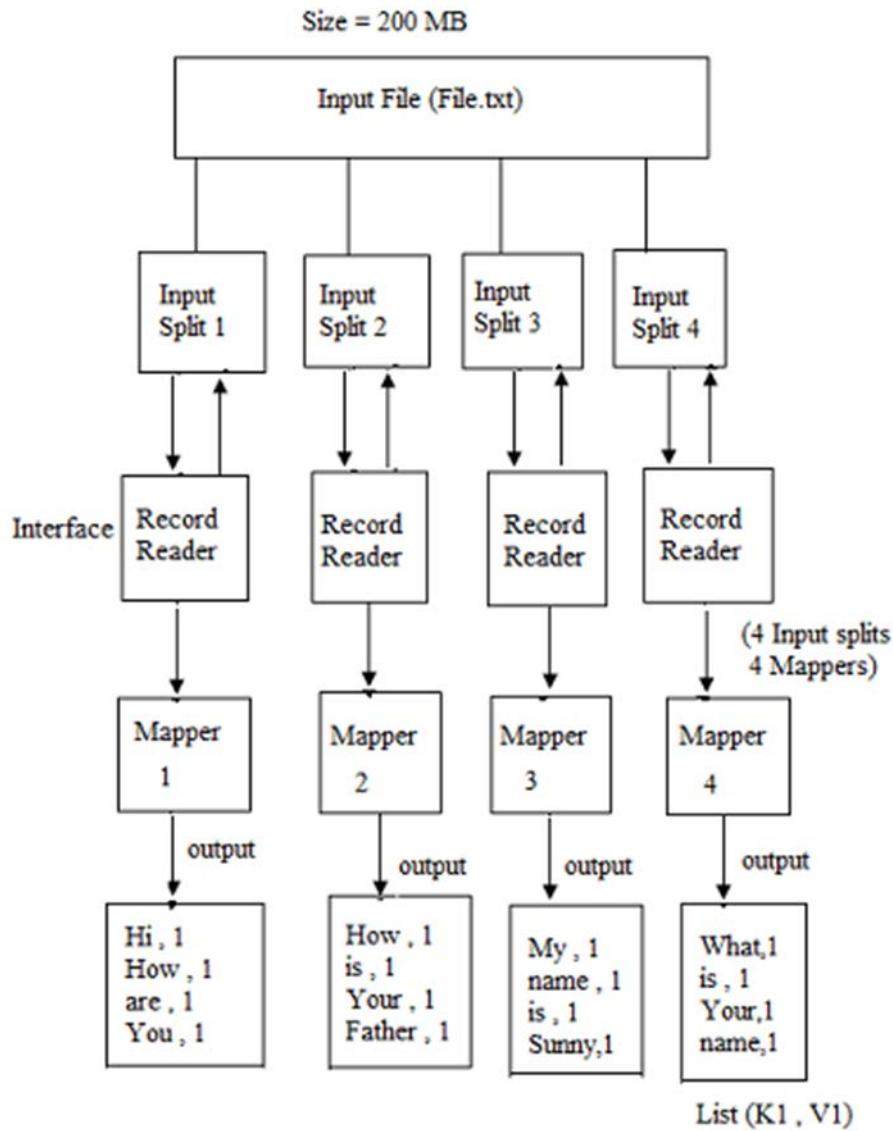


Fig 3. Map Functioning

F. Final Output Task Execution

In case of Reducer function we have to process the Mapper output. Input of the Reducer function is the output of the Mapper function. The output of Mapper function is in the form of <key, value>. For example as shown in fig 4 “Hi How are You” is being represented as [{Hi, 1}, {How, 1}, {are, 1} and {You, 1}]. Then Shuffling, Sorting, Grouping and Filtering is being performed that is all the Mapper output is being grouped together and produce the output in the form of <key, value>. Then the Reducer function is being performed which combines the value and produce the optimized final output. [2]

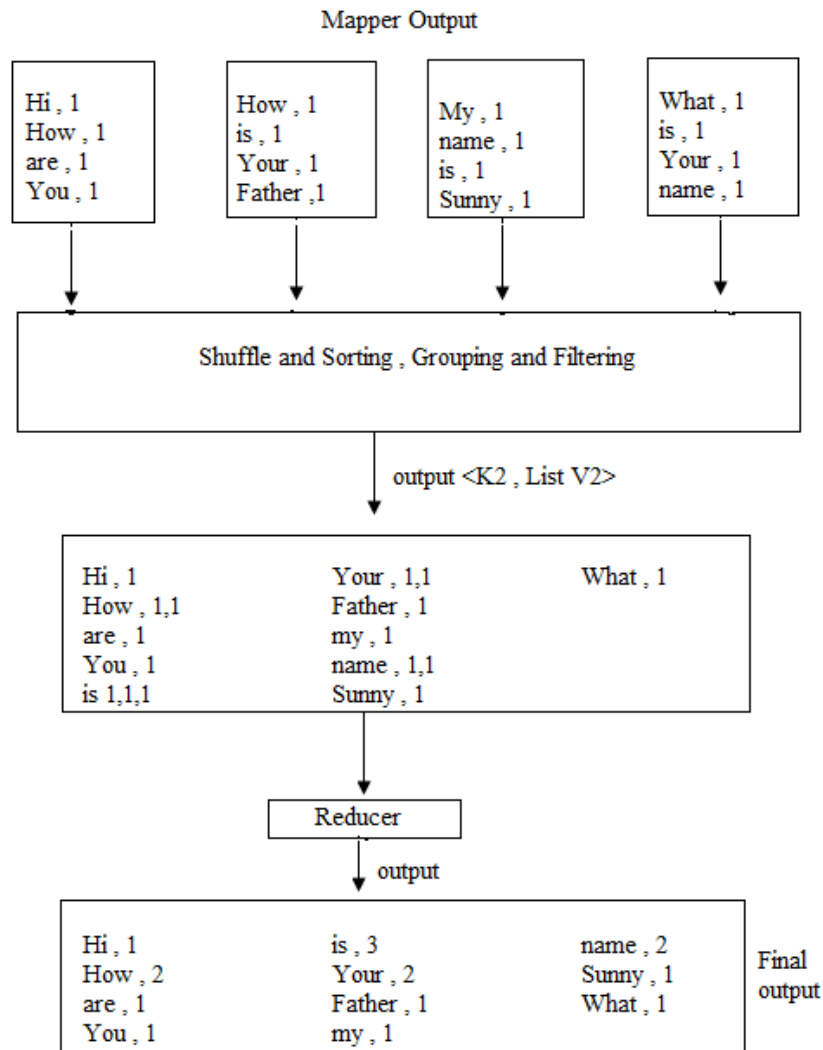


Fig 4. Reduce Function

I. HIVE

Hive is one of the data warehouse infrastructure tool which is used to process structured data in Hadoop. It has been resides on the top of Hadoop to summarize Big Data, making query and analysing them easily. [1]

Firstly, Hive was developed by Facebook, then Apache Software Foundation developed it further as an open source under the Apache Hive.

Hive is not a Relational Database tool nor it has been designed for Online Transaction Processing (OLTP) nor is it a language for real time queries. Rather it has been designed for OLAP.

It stores Schema in a database and process the data into HDFS. [8]

Fig 5 represents that when user comes through CLI that is Hive Terminal it directly connected to the Hive Drivers. When user comes through JDBC/ODBC that is JDBC program at that time with the help of API that is Thrift driver it is connected to the Hive Driver. But when the user comes through Web Graphical User Interface that is Ambari Server it

will directly connected with the Hive Driver. The Hive Driver receives the queries or tasks from the user and perform map reduce task internally with the help of Hadoop Architecture.[9] The Hadoop Architecture uses Name Node, Data Node, Job Tracker and Task Tracker for receiving and dividing the work what Hive have send to the Hadoop which is the job of Map Reduce Architecture. [5]

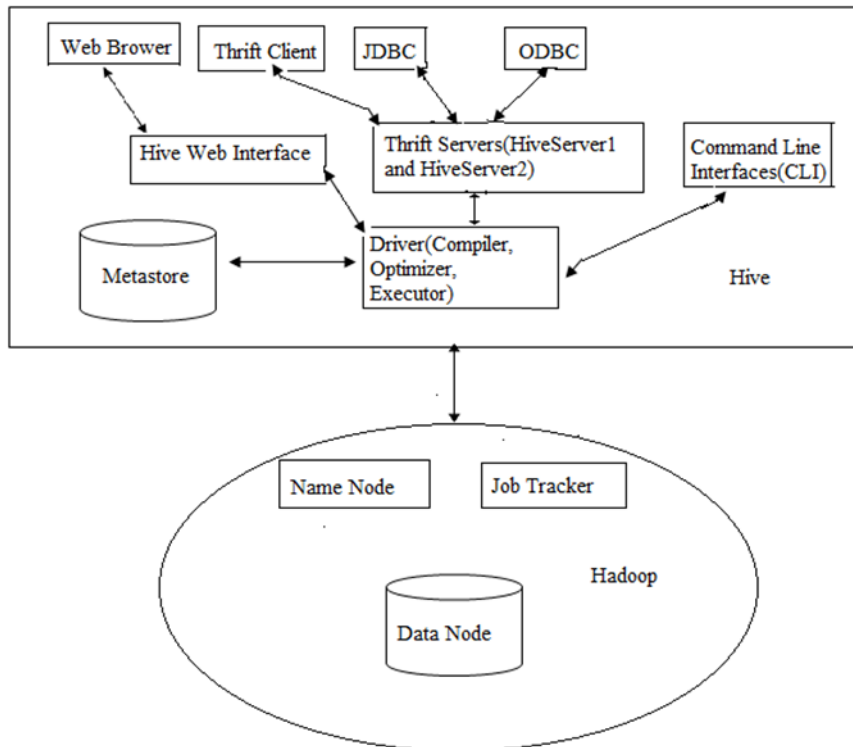


Fig 5. Hive Architecture

Fig 6. Query Execution in Hive



As shown in fig 6 steps of working of hive are as follows: [3]

Step 1: The UI will call the execute interface to the Driver.

Step 2: The Driver creates a session handle for the query and sends the query to the compiler in order to generate an execution plan.

Step 3 and 4: Compiler sends a request for getMetaData because of need of metadata and receives the sendMetaData request from Metastore.

Step 5: Metadata is useful for typechecking the expressions in the query tree as well as to prune partitions that is based on query predicates. The plan that is being generated by the compiler is a DAG of stages. Each stage either a map-reduce job, a metadata operations or a HDFS operations. For map-reduce jobs, the plan consist of map operator trees that are operated on mappers and a reduce operator tree that are operated on reducers.

Step 6: The execution engine will submit all stages to their appropriate components (Step 6, 6.1, 6.2, 6.3). In each task the deserializer associated with table is used to read the rows from the HDFS files. These are passed through the operator tree. After the generation of output, it is being written to a temporary HDFS file through serializer. These temporary files are used to provide the subsequent map-reduce stages of the plan

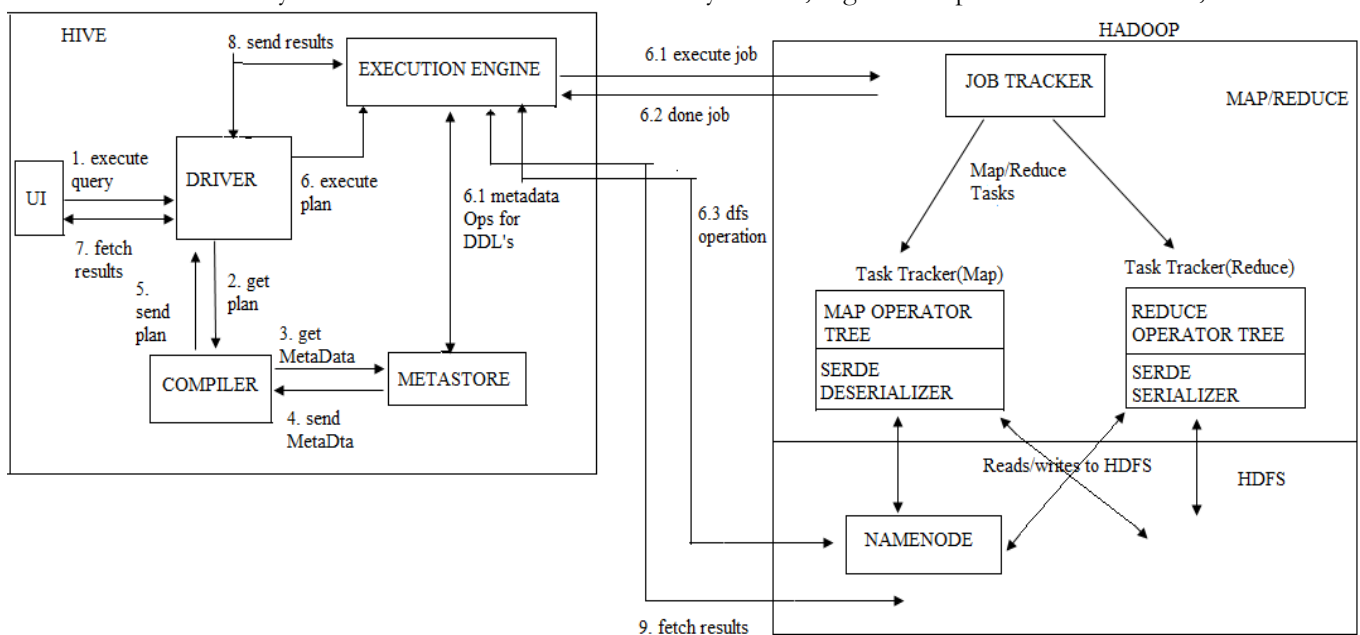
Step 7, 8 and 9: In case of queries, the contents of temporary file are read by the execution engine directly from HDFS when fetch call is made by the Driver.

G. Pig

Pig provides an engine or say platform for executing data flows in parallel on top of Hadoop. It includes a language, Pig Latin (Just like scripting language), for expressing these data flows. [6]

H. Pig on Hadoop

Pig runs on Hadoop. It makes use of both the Hadoop Distributed File System, HDFS, and Hadoop’s processing system, MapReduce. HDFS is a distributed filesystem that stores files across all of the nodes in a Hadoop cluster. It handles breaking the files into large blocks and distributing them across different machines, including making multiple copies of each block so that if any one machine fails no data is lost. By default, Pig reads input files from HDFS, uses HDFS to



store intermediate data between MapReduce jobs, and writes its output to HDFS. It can also read input from and write output to sources other than HDFS. MapReduce is a simple but powerful parallel data-processing paradigm. Every job in MapReduce consists of three main phases: map, shuffle, and reduce.[15]

For example:



Our input will be:

Mary had a little lamb
its fleece was white as snow
and everywhere that Mary went
the lamb was sure to go.

The following pig script finds the number of times a word repeated in a file:

Word Count Example Using Pig Script:

```
-- Load input from the file named Mary, and call the single  
-- field in the record 'line'.
```

```
grunt> lines = load '/home/hduser/Desktop/mary.txt' as (line:chararray);  
  
-- 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.  
grunt> words = foreach lines generate flatten (TOKENIZE(line)) as word;  
-- Now group them together by each word.  
grunt> grouped = group words by word;  
-- Count them.  
grunt> wordcount = foreach grouped generate group,COUNT(words);  
-- Print out the results.  
grunt> dump wordcount;
```

The above pig script, first splits each line into words using the TOKENIZE operator. The tokenize function creates a bag of words. Using the FLATTEN function, the bag is converted into a tuple. In the third statement, the words are grouped together so that the count can be computed which is done in fourth statement.

You can see just with 5 lines of pig program, we have solved the word count problem very easily.

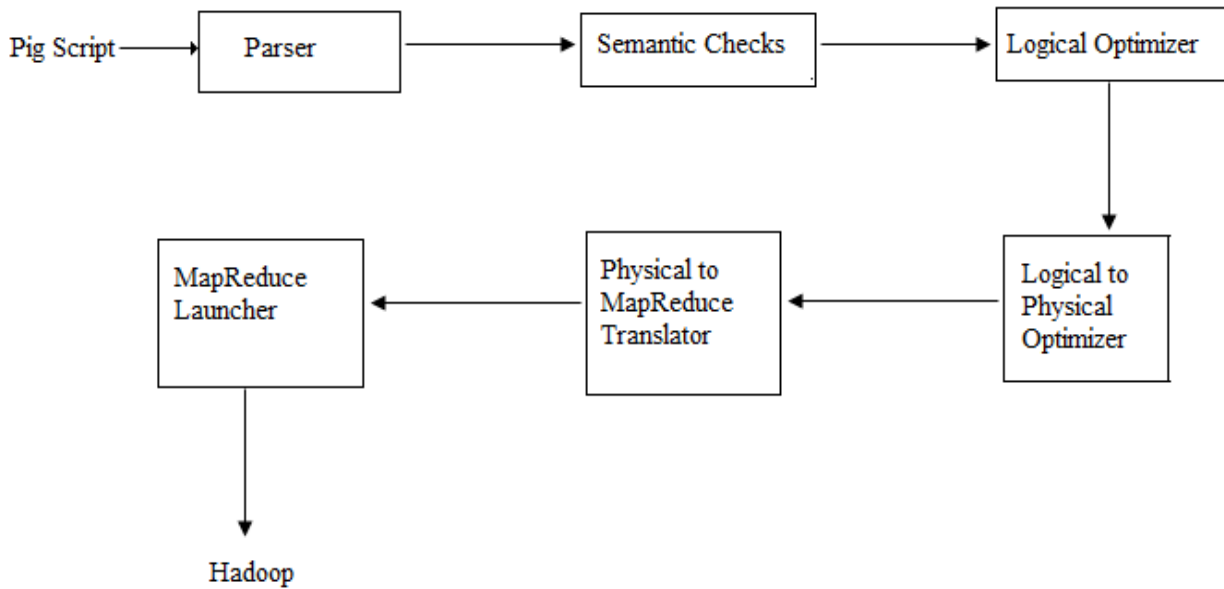
There is no need to be concerned with map, shuffle, and reduce phases when using Pig. It will manage decomposing the operators in your script into the appropriate MapReduce phases.

Pig Latin, a Parallel Dataflow Language[7]

Pig Latin is a dataflow language. This means it allows users to describe how data from one or more inputs should be read, processed, and then stored to one or more outputs in parallel. These data flows can be simple linear flows like the word count example given previously. They can also be complex workflows that include points where multiple inputs are joined, and where data is split into multiple streams to be processed by different operators. To be mathematically precise, a Pig Latin script describes a directed acyclic graph (DAG), where the edges are data flows and the nodes are operators that process the data.

This means that Pig Latin looks different from many of the programming languages you have seen. There are no if statements or for loops in Pig Latin. This is because traditional procedural and object-oriented programming languages describe control flow, and data flow is a side effect of the program. Pig Latin instead focuses on data flow.

Fig 7. Working of Pig



As shown in fig 7 the steps of working of pig are: [3]

1. Parsing
2. Semantic checking
3. Logical optimizer
4. Logical to Physical optimizer
5. Physical to MapReduce translator
6. MapReduce launcher

3. Implementation

Implementation is performed using a map reduce program on raw data. Map reduce program contain 200 lines of code to perform mapping and reducing of data. This code is written in java language. Hadoop component map reduce is a powerful concept that performs mapping and reducing on data. As compare to traditional system if it is performed using map and reduce concept without Hadoop framework it becomes very difficult and thousands of lines of code is written for that. So same concept is implemented using Hadoop ecosystem that is hive. It runs on top of Hadoop framework and perform mapping. First a hive script is written that consist of 7 lines of code. Then “hive -f <name of the hive script file>” is written on the command prompt and ad-hoc-queries are run and map reduce is performed automatically as there is no need to write full code for mapping. Then “hive -e ‘select * from word_count1’” command is used to display output.

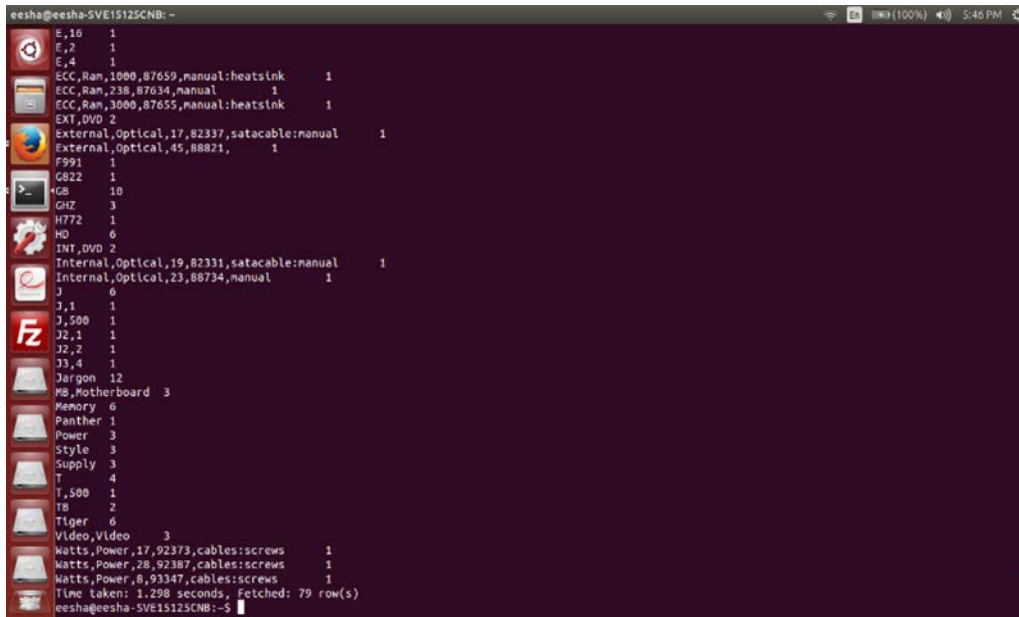
```
Terminal
+ Tiger 1
+ Video 1
+ Video 1
+ Video 1
+ Video 1
+ Video 1
+ Video 1
+ Video 1
+ Video 1
+ Video 1
+ Video 1
+ Watts 1
+ Watts 1
+ Watts 1
+ atacable:manual 1
+ cables:screws 1
+ cables:screws 1
+ cables:screws 1
+ cables:screws 1
+ dvd>manual:game 1
+ dvd>manual:hdmicable 1
+ fans>manual:screws 1
+ fans>manual:screws 1
+ fans>manual:screws:watercooler 1
+ manual 1
+ manual 1
+ manual:game 1
```

Fig 8. Wordcount output in MapReduce

The output of wordcount program in MapReduce is shown in fig 8.
The time taken by wordcount program using MapReduce is 2.13 sec.

A 200 lines Java code written for MapReduce can be reduced to 7 lines of hive code which is given as:
CREATE TABLE <name of the table> (column name DATATYPE);
LOAD DATA INPATH '<path where file is stored>' OVERWRITE INTO TABLE <name of the table>;
CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
(SELECT explode(split(<column name>, '\\s')) AS word FROM <name of the table>) w
GROUP BY word
ORDER BY word;





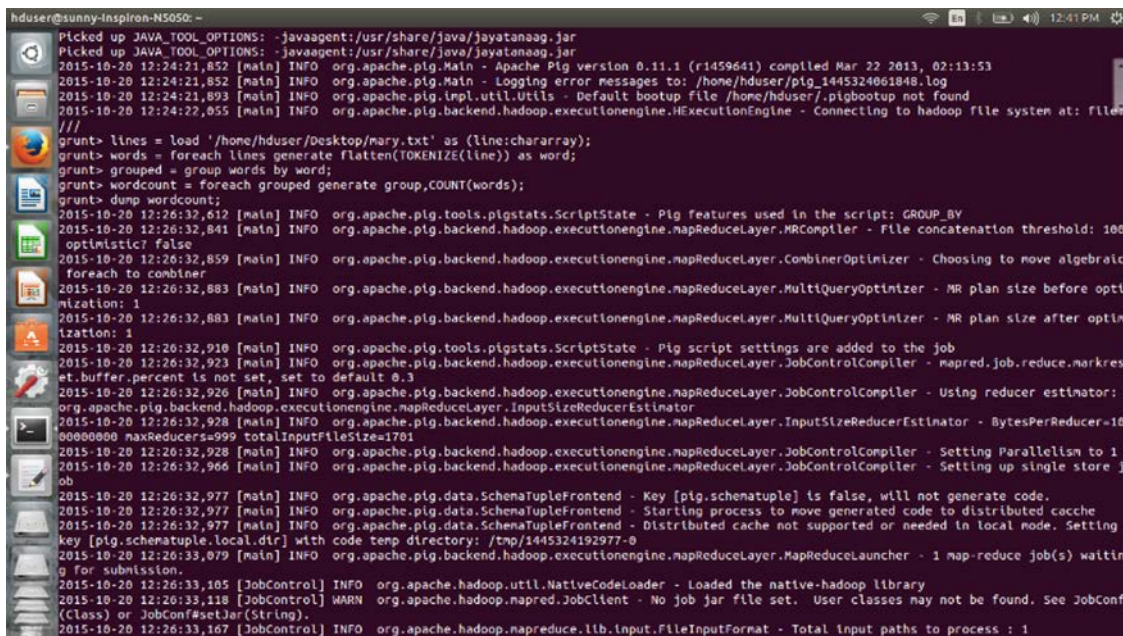
```
eesha@eesha-SVE15125CNB: ~
E,10 1
E,2 1
E,4 1
ECC,Ran,1000,87659,manual:heatsink 1
ECC,Ran,238,87634,manual 1
ECC,Ran,3000,87655,manual:heatsink 1
EXT,DVD 2
External,Optical,17,82337,satacable:manual 1
External,Optical,45,88821, 1
F991 1
G022 1
G0 10
GHZ 3
H772 1
HD 6
INT,DVD 2
Internal,Optical,19,82331,satacable:manual 1
Internal,Optical,23,88734,manual 1
J 6
J,1 1
J,500 1
J2,1 1
J2,2 1
J3,4 1
Jargon 12
MB,Motherboard 3
Memory 6
Panther 1
Power 3
Style 3
Supply 3
T 4
T,500 1
TB 2
Tiger 6
Video,Vldeo 3
Watts,Power,17,92373,cables:screws 1
Watts,Power,28,92387,cables:screws 1
Watts,Power,6,93347,cables:screws 1
Time taken: 1.298 seconds, Fetched: 79 row(s)
eesha@eesha-SVE15125CNB: ~
```

Fig 9. Output of Wordcount in Hive

The output of wordcount program in Hive is shown in fig 9.

The time taken to execute wordcount program in hive is 1.298 sec

But when we execute with pig the 7 lines of code is reduced by 5 lines of code which is described in above section



```
hduser@sunny-inspiron-N5050: ~
Picked up JAVA_TOOL_OPTIONS: -javaagent:/usr/share/java/jayatanaag.jar
Picked up JAVA_TOOL_OPTIONS: -javaagent:/usr/share/java/jayatanaag.jar
2015-10-20 12:24:21,852 [main] INFO org.apache.pig.Main - Apache Pig version 0.11.1 (r1459641) compiled Mar 22 2013, 02:13:53
2015-10-20 12:24:21,852 [main] INFO org.apache.pig.Main - Logging error messages to: /home/hduser/pig_1445324061848.log
2015-10-20 12:24:21,893 [main] INFO org.apache.pig.impl.util.Utils - Default bootstrap file /home/hduser/.pigbootstrap not found
2015-10-20 12:24:22,855 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: file:///
grunt> lines = load '/home/hduser/Desktop/mary.txt' as (line:chararray);
grunt> words = foreach lines generate Flatten(TOKENIZE(line)) as word;
grunt> grouped = group words by word;
grunt> wordcount = foreach grouped generate group,COUNT(words);
grunt> dump wordcount;
2015-10-20 12:26:32,612 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig Features used in the script: GROUP_BY
2015-10-20 12:26:32,841 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100
optimistic? false
2015-10-20 12:26:32,859 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.CombinerOptimizer - Choosing to move algebraic
foreach to combiner
2015-10-20 12:26:32,883 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before opti
mization: 1
2015-10-20 12:26:32,883 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after opti
mization: 1
2015-10-20 12:26:32,910 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig script settings are added to the job
2015-10-20 12:26:32,923 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting up single store j
ob
et.buffer.percent is not set, set to default 0.3
2015-10-20 12:26:32,926 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Using reducer estimator:
org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.InputSizeReducerEstimator
2015-10-20 12:26:32,928 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.InputSizeReducerEstimator - BytesPerReducer=10
00000000 maxReducers=999 totalInputFileSize=1701
2015-10-20 12:26:32,928 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting Parallelism to 1
2015-10-20 12:26:32,966 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting up single store j
ob
2015-10-20 12:26:32,977 [main] INFO org.apache.pig.data.SchemaTupleFrontend - Key [pig.schematuple] is false, will not generate code.
2015-10-20 12:26:32,977 [main] INFO org.apache.pig.data.SchemaTupleFrontend - Starting process to move generated code to distributed cacche
key [pig.schematuple.local.dir] with
2015-10-20 12:26:33,079 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 1 map-reduce job(s) waitin
g for submission.
2015-10-20 12:26:33,105 [JobControl] INFO org.apache.hadoop.util.NativeCodeLoader - Loaded the native-hadoop library
2015-10-20 12:26:33,118 [JobControl] WARN org.apache.hadoop.mapred.JobClient - No job jar file set. User classes may not be found. See JobConf
(Class) or JobConf#setJar(String).
2015-10-20 12:26:33,167 [JobControl] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
```

Fig 10. Output of Wordcount in Pig

The output of wordcount program in Pig is shown in fig 10.

The time taken to execute wordcount program in pig is 0.09 seconds.



4. Comparison

An alternative to Hadoop MapReduce Hive which is an open source was built so that Hadoop developers could do the same thing in Java in a less complex way by writing only fewer lines of code that will be very easy to understand. A Hadoop developer who knows about Hive does not require to learn concepts of Java. [3]

Difference between Hadoop MapReduce and Hive or Pig:

There are various situations when it is useful to implement MapReduce rather than Hive or Pig that are described as under:

1. When Hadoop developers need definite Driver program control then they should use Hadoop MapReduce instead of Hive or Pig.
2. Whenever the job requires the use of a custom partitioner then Hadoop developers should utilize Hadoop MapReduce.
3. If pre-defined library of Java Mappers or Reducers are already present then we should use MapReduce.
4. When there is a need of good amount of testability with the large data sets then it is better to use MapReduce rather than Hive or Pig.
5. If there is a demand of legacy code requirements in the application then MapReduce is a better option.
6. If the requirement of job is optimization at a particular stage of processing by making best use of tricks like in mapper combining then Hadoop MapReduce is a good option.
7. If the job requires the better and efficient use of distributed cache, cross products, grouping and joining then our approach should be MapReduce.[14]

Difference between Hive and Pig:

Depending on the type of data one has to decide which he should use hive or Pig. This can be decided by analysing differences that are described below:

1. Hive is used for completely structured data but Pig is used for semi-structured data.
2. Hive is mainly used for creating reports but Pig is mainly used for programming.
3. Hive operates on the server side of any cluster whereas Pig operates on client side of any cluster.
4. Hive can start an optional thrift based server that can send queries from any nook and corner directly to the Hive Server which will execute them but this feature is not present in Pig.
5. Hive makes use of exact variation of the SQL DLL language by defining the tables before and store the database details in any local database but in case of Pig there is no need of metadata and the schemas or data types are defined in the script itself
6. Hive has a facility of partition in order to process the subset of data in an alphabetical order but in case of Pig no provision of partition but can be achieved by using filters.
7. Pig supports Avro but Hive does not.
8. Pig can be installed easily as compared with Hive as Pig is based on shell interaction.

5. Conclusions

There are various ways for Hadoop to run the job. The three programming approaches that are MapReduce, Hive and Pig are used. But after performing practically we conclude that Pig takes less time as compare to both MapReduce and Hive. But all approaches have pros and cons so we have to choose according to our need and data.

Acknowledgment

This paper is implemented under the guidance of Dr Ashok Kumar Goel who is Director of Colleges of Maharaja Ranjit Singh State Technical University

References

1. http://www.bigdatauniversity.com/web/media/player.php?file=BD030V212EN/Videos/HiveLesson3_HiveDML_Video1_2_1_2.mp4&caption=files.db2university.com/BD030V212EN/Videos/EN/HiveLesson3_Video1_captions.srt, 25 September,2015
2. <https://www.youtube.com/watch?v=DLutRT6K2rM>,10 July,2015
3. <http://www.dezyre.com/Hadoop-Administration/28>, 20 September, 2015
4. <http://www.hadooppoint.com/hadoop-mapreduce-multiple-inputs-and-outputs-program/>,10 October, 2015
5. Eric Sammer, Hadoop Operations. O'ReillyMedia, First Edition, September 2012
6. Alan Gates, Programming Pig. O'Reilly Media, First Edition, October 2011
7. Ammar Fuad, Alva Erwin, Heru Purnomo Ipung,” Processing Performance on Apache Pig, Apache Hiveand MySQL Cluster”, ICTS 2014, Surabaya, Indonesia.
8. Anshu Choudhary and C.S. Satsangi “Query Execution Performance Analysis of Big Data Using Hiveand Pig of Hadoop”,vol. Su-9 no.3,pp.91-93,Sep 2015.
9. Anja Gruenheid,Edward Omiecinski and Leo Mark,” Query Optimization Using Column Statistics in Hive”, in july 2011
10. <http://www.informatica.com>
11. Dirk deRoos, Chris Eaton, George Lapis, PauZikopoulos and Tom Deutsch, Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data.McGraw Hill Osborne Media;1 edition(October19,2011)
12. Arun Murthy, Vinod Vavilapalli, Douglas Eadline , Joseph Niemiec and Jeff Markham,Apache Hadoop YARN: Moving beyond MapReduce and Batch Processing with Apache Hadoop 2.Addison-Wesley Professional;1st Edition(March 2014)
13. http://www.bigdatauniversity.com/web/media/player.php?file=BD001V212EN/Videos/Unit_1_What_is_Hadoop_Part1.mp4&caption=files.db2university.com/BD001V212EN/Videos/EN/Unit_1_What_is_Hadoop_Part1.srt,8 jan,2015
14. <https://www.youtube.com/watch?v=DLutRT6K2rM>,15July, 2015
15. <https://www.whitehouse.gov/webform/clone-learn-more-about-big-data-review>,2 December, 2014

