IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper **Vol.-7, Issue-8, Aug 2019** **E-ISSN: 2347-2693**

Comparative Analysis of Big Data Technologies

C. Jasmine[1*], A. Abinaya[2]

[1,2]Dept.of Computer Science, sengamala thayaar educational trust women's college, Sundarakkottai, Mannargudi, Tamilnadu, India

[*]*Corresponding Author: jasmine.mscit@gmail.com, Tel.: +91 9976508314*

DOI: https://doi.org/10.26438/ijcse/v7i8.4957 | Available online at: www.ijcseonline.org

Accepted: 07/Aug/2019, Published: 31/Aug/2019

Abstract— Recent technological advances and reduction in storage prices has led to accumulation of huge amount of data known as Big Data. This data, belonging to different applications and timelines, is difficult for organisations to process. In order to solve this difficulty, Doug Cutting and Mike Cafarella came up with a framework called Hadoop. Becoming open source in 2012, Hadoop went on to include Pig, Hive and many more products. Following this, Spark was developed by MatieZaharia in 2009 which was open sourced in 2010. Meanwhile, many organisations came up with their own platforms to deal with Big Data. Hence, sprouting from Google's MapReduce paper, these tools have grown into a wide array of technologies. This project focusses on comparing three main big data technologies which are used widely these days namely Pig, Hive and R. Similar problem statements are executed on all three platforms and performance is judged based upon the query execution time.

***Keywords*—** Hadoop, HDFS, Big Data, Pig, Hive, R.

I. INTRODUCTION

Big Data

Big Data defines a large amount of data, usually greater than I TB of data. Also, Big Data refers to a collection of data that grows at an exponential rate. Hence, within a small span of time, this data becomes so big and complex that traditional data management tools cannot store or process this data any longer. Some examples of Big Data are –

1. Twitter produces somewhere around 90 million tweets every day.
2. Walmart handles close to 1 million customer transactions every single day. As a result, Walmart imports close to 2.5 petabytes of data into their databases every single day.
3. eBay uses two data warehouses, containing 7.4 petabytes and 40 petabytes of data and a Hadoop cluster to store its entire data. This data contains search, customer recommendations and merchandising.
4. NYSE (New York Stock Exchange) generates close to 1 terabyte of new trade data every day.
5. Facebook generates more than 500 terabytes of data every day and puts that data into its databases. This data contains photos, video uploads, messages that are exchanged, comments and many other things.
6. A jet engine is capable of generating more than 10 terabytes of data in a 30 minute of a flight. Since a thousand flights take off every day, the total data generated in a single day goes into petabytes.

Big Data encompasses three types of data. These are –

1. Structured Data – Data that has a well-defined structure falls under Structured Data. This data has records divided into rows and columns. As a result, it is easy to read and manage this data. Usually, data in databases falls under Structured Data. Well defined software are present to read structured data. Furthermore, it is easy to set constrains for this type of data.
   However, Structured Data accounts for only 20% of the total data available. Usually, this data has two types of sources, machines and humans. Hence, specialized programs can be written to read this data and write into databases. On the other hand, users can manually enter data into databases.
2. Semi-structured data – The main difference between Structured and Semi-structured is that Structured data cannot be stored in traditional databases. However, Semi-Structured still retains some organizational properties which makes it easier to process than Unstructured data.
3. Some common examples of Semi-structured data is data stores in CSV files and NoSQL documents.

Unstructured data – In structured data, data has a particular pre-defined format in which is gets stores. However, in Unstructured data, no such format is present. Hence, unstructured data can be used to store any kind of data to it. The downside of using unstructured data is that the data does not have any constraints to it. Hence storing this data and managing and manipulating this data prove to be a difficult task. Unstructured data provides a way to store data like images and videos which cannot be processed using traditional databases.

5 V's of Big Data

Big Data is always described as having at least three distinct dimensions: volume, velocity, and variety. With the passage of time, two more Vs were added to the list, namely variability and value. These are defined as the five V's of Big Data. These are

1. **Volume:** Big data first and foremost has to be "big,"and size in this case is measured as volume. From clinical data associated with lab tests and physician visits, to the administrative data surrounding transactions, this well of information is already expanding. When that data is coupled with greater use of precision medicine and stock exchange, there will be a big data explosion in any industry, be it medicine or business.

2. **Velocity:** Velocity in the context of big data refers to two related concepts: the rapidly increasing speed at which new data is being created by technological advances, and the corresponding need for that data to be digested and analyzed in near real-time. For example, as more and more medical devices are designed to monitor patients and collect data, there is great demand to be able to analyze that data and then to transmit it back to clinicians and others.

3. **Variety:** With increasing volume and velocity comesincreasing variety. This third "V" describes just what you'd think: the huge diversity of data types that organizations see every day. In case of medical applications, each medical device might collect a different kind of data, which in turn might be interpreted differently by different physicians—or made available to a specialist but not a primary care provider. Standardizing and distributing all of that information so that everyone involved is on the same page. With increasing adoption of population health and big data analytics, we are seeing greater variety of data by combining traditional clinical and administrative data with unstructured notes, socioeconomic data, and even social media data.

4. **Variability:** The way care is provided to any givenpatient depends on all kinds of factors—and the way the care is delivered and more importantly the way the data is captured may vary from time to time or place to place. For example, what a clinician reads in the medical literature, where they trained, or the professional opinion of a colleague down the hall, or how a patient expresses herself during her initial exam all may play a role in what happens next. Such variability means data can only be meaningfully interpreted when care setting and delivery process is taken into context. For example, a diagnosis of "CP" may mean chest pain when entered by a cardiologist or primary care physician but may mean "cerebralpalsy" when entered by a neurologist or pediatrician. Because true interoperability is still somewhat elusive in health care data, variability remains a constant challenge.

5. **Value:** Last but not least, big data must have value. That is, if you're going to invest in the infrastructure required to collect and interpret data on a system-wide scale, it's important to ensure that the insights that are generated are based on accurate data and lead to measurable improvements at the end of the day.

Three technologies i.e. Pig, Hive and R are described below.

1. PIG

Apache Pig [21] is a higher level of abstraction over MapReduce. Founded in 2006 at Yahoo, Pig was mover into the Apache Software Foundation in 2007. Pig has a few distinctive features that makes it an ideal choice for big data processing.

- Uses a simple language called Pig Latin, for writing code to process data.
- Since it is a higher level of abstraction over MapReduce, Pig code gets converted into MapReduce code internally.
- Users can use UDF (User Defined Functions) that can be written in other programming languages like Java, etc.
- Compared to MapReduce, Pig requires a smaller amount of code to accomplish the same task.
- User can choose between two execution modes, Local mode and Map Reduce mode.
- Pig uses multiple-query approach, thus reducing the size of the code.
- There are many built in operations and nested data types present in Pig.
- The tasks in Pig are automatically optimized by the Pig Engine.

Perhaps the biggest advantage that Pig offers is its ease to handle different types of data. Pig can handle all three types of data, structured, semi structured and unstructured with the same ease and using the same tools.

Pig consists of two parts, Latin Pig and Pig Engine. Latin Pig

[20] is the high-level language in which all the Pig scripts are written. Pig Engine is the execution engine that takes in 3 Pig script and outputs MapReduce code. This code then runs on the Hadoop platform on the data stored in HDFS and produces the information.

The various parts of Pig Engine are –

- Grunt shell
- Pig Server
- Parser
- Optimizer
- Compiler
  Execution Engine

However, since the code needs to be converted into MapReduce code, the time taken by the system to produce the results is high in Pig. Still, owing to its simplicity and ease to learn, Pig has been used by a large number of users as their primary Big Data analysis tool.

2. HIVE

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive.

The features of Hive are -

- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP.
- It provides SQL type language for querying called HiveQL [19] or HQL.
- It is familiar, fast, scalable, and extensible.

The working of Hive is as follows:

1. Execute Query: The Hive interface such as Command Line or Web UI sends query to Driver (any database driver such as JDBC, ODBC, etc.) to execute.
2. Get Plan: The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query.
3. Get Metadata: The compiler sends metadata request to Metastore (any database).
4. Send Metadata: Metastore sends metadata as a response to the compiler.
5. Send Plan: The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete.
6. Execute Plan: The driver sends the execute plan to the execution engine.
7. Execute Job: Internally, the process of execution job is a MapReduce job. The execution engine sends the job to JobTracker, which is in Name node and it assigns this.
8. Metadata Ops: Meanwhile in execution, the execution engine can execute metadata operations with Metastore.
9. Fetch and send Result: The execution engine receives the results from Data nodes. The execution engine sends those resultant values to the driver the driver sends the results to Hive Interfaces.

3. R

R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering and many more) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

One of R’s strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

R is a programming language and software environment for statistical analysis, graphics representation and reporting. The following are the important features of R −

- R is a well-developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.
- R has an effective data handling and storage facility,
- R provides a suite of operators for calculations on arrays, lists, vectors and matrices.
- R provides a large, coherent and integrated collection of tools for data analysis.
- R provides graphical facilities for data analysis and display either directly at the computer or printing at the papers.

II. RELATED WORK

Big Data is present in almost every domain today. Lots of research has been done in this vast field. In [1] authors have proposed that sensing technologies, cloud computing, internet of things and big data analytics systems as emerging technologies which has made it possible to achieve impressive progress in the field of computational field and storage which play a major role in efficiency and effectiveness of the healthcare systems. Cloud computing [2] is nowhere left behind in big data as well. System consisting of monitoring agents, cloud infrastructure, and operation centre by using HadoopMapReduce and Spark to enhance the processing by splitting and processing data streams concurrently. But with every positive thing comes the challenges and issues related as well. There is a model [3] proposed to tackle challenges like data complexity, computational complexity, and system complexity and also presented suggestions for carrying out Big Data projects. The emergence of big data, considered both the opportunities and the ethical challenges for the market research [7] as proposed in this research.

There are various applications that are developed in this domain as well. Big data is suitable for every trend these days. Therefore, applications like [4] an electricity generation forecasting system which can predict the required power generation close to 99% of the actual usage by using Big Data Analytics was proposed. Also, a Location-Aware Analytics.

System [5] using effective spatio-textual indexes and incremental algorithms that has been deployed and widely accepted. Another application is where authors proposed a model [17] for monitoring and analyzing Internet Traffic which is in the form of Big Data. They provided analysis and forecasts, including traffic management and network upgrade to enhance the quality that can be used to promote investments as well.

Data privacy is another big issue these days. For this author have proposed [6] six data management research challenges for Big Data and Cloud such as Data Privacy, Approximate Results, Data exploration to enable Deep Analytics, Enterprise data enrichment with web and social media. Another application is where authors have reported [9] a bibliometric study of critical BI&A publications, researchers, and research topics based on more than a decade of related academic and industry publications.

One of the bigger research was proposed in one the publication where authors described the HACE theorem [12] that characterizes the features of the Big Data Revolution and also proposes a big data processing model, from the data mining perspective.

Harnessing the full potential of any technology is necessary as well. In a paper [13] authors proposed a comprehensive overview of the applications of data processing platform designed to harness the potential of big data in the field of road transport policies in Europe. Also, to implement big data algorithms efficiently, authors proposed [14] FB+-tree that builds fast indexing structure using multi-level Key ranges, which is explained on the basis of B+ trees. Point searches and range searches are helped by early termination of searches for non-existent data.

For any technology to perform up to its full potential it should be as robust it can be. In one of the paper, authors proposed a model [15] based on robust data analytics, high performance computing, efficient data network management and cloud computing techniques that are critical towards optimized performance of Smart Grid's so as to reduce the cost for customer.

Big data is surely very important and big terms these days. Organizations have understood the importance of big data for managing their data efficiently and to obtain day to day analysis of the data generated. In one the paper this functionality was proposed and a deep understanding of this whole concept was given to the users.

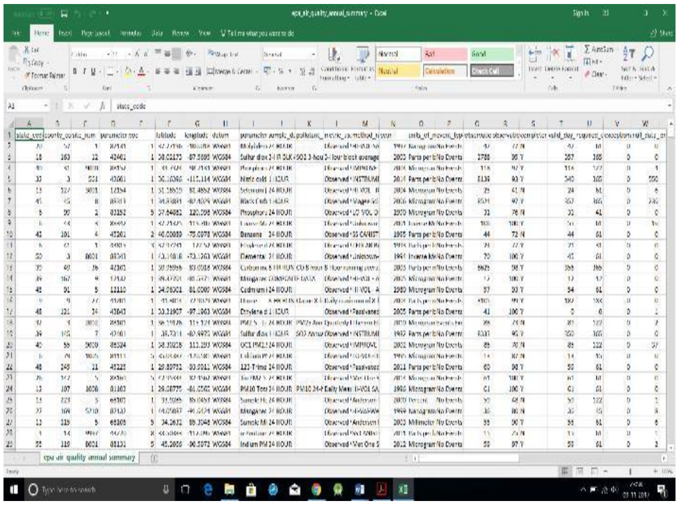
In another research [18] authors proposed an overview of big data, significance of big data, how hadoop works and systems, which is based on analysis of published implementation architectures of big data use cases. Different flavours of Hadoop.

There are many researches that have been proposed in this domain and also so many are being going on to make the people aware of any new and unseen facts of this technology.

III. METHODOLOGY

PROPOSED MODEL

We have taken an Air Quality dataset from Kaggle.com containing approximately 7-8 million records. Here is a snippet of the dataset-



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	state_cod	county_co	site_num	parameter_coc		latitude	longitude	datum	parameter	sample_du	pollutant_	metric_use	method_n	year	units_of_m	event_typ	observatio	observatio	completen	valid_day_	required_c	exception	null_data_
2	[illegible]	[illegible]	1	[illegible]	1	[illegible]	[illegible]	WGS84	[illegible]	24 HOUR			Observed	[illegible]	1997	Nanogram	No Events	[illegible]	77	N	[illegible]	[illegible]	0	0
3	18	163	12	42401	1	38.02173	-87.5695	WGS84	Sulfur diox	3-HR BLK A	SO2 3-hou	3-Hour block average		2003	Parts per b	No Events	2788	95	Y	357	365	0	0
4	[illegible]	[illegible]	[illegible]	[illegible]	1	[illegible]	[illegible]	WGS84	Phosphoru	24 HOUR			Observed	IMPROVE	2003	Microgram	No Events	118	97	Y	118	122	0	1
5	32	3	561	42601	1	36.16366	-115.114	WGS84	Nitric oxid	1 HOUR			Observed	INSTRUM	2014	Parts per b	No Events	8139	93	Y	340	365	0	550
6	13	127	3001	12154	1	31.16933	-81.4882	WGS84	Selenium	24 HOUR			Observed	HI-VOL	2004	Microgram	No Events	25	41	N	24	61	0	6
7	45	45	8	88313	1	34.83881	-82.4029	WGS84	Black Carb	1 HOUR			Observed	Magee Sci	2006	Microgram	No Events	8521	97	Y	357	365	0	236
8	6	99	2	83152	5	37.64882	120.998	WGS84	Phosphoru	24 HOUR			Observed	LO VOL	1990	Microgram	No Events	31	76	N	31	41	0	0
9	[illegible]	[illegible]	[illegible]	[illegible]	1	[illegible]	[illegible]	WGS84	[illegible]	24 HOUR			Observed	Unknown	2007	Inverse Mi	No Events	[illegible]	100	Y	[illegible]	[illegible]	0	[illegible]
10	42	101	4	45201	2	40.00889	-75.0978	WGS84	Benzene	24 HOUR			Observed	SS CANIST	1995	Parts per b	No Events	44	72	N	44	61	0	0
11	[illegible]	[illegible]	1	[illegible]	[illegible]	[illegible]	[illegible]	WGS84	[illegible]	24 HOUR			Observed	[illegible]	[illegible]	Parts per b	No Events	[illegible]	77	Y	[illegible]	[illegible]	0	0
12	50	3	8001	88341	1	43.14016	-73.1363	WGS84	Elemental	24 HOUR			Observed	Unknown	1991	Inverse Mi	No Events	70	100	Y	45	61	0	0
13	39	49	36	42101	1	[illegible]	-83.0018	WGS84	Carbon mo	8-HR RUN	CO 8-hour	8-Hour running avera		2003	Parts per m	No Events	8625	98	Y	355	365	0	0
14	[illegible]	167	9	12132	1	[illegible]	[illegible]	WGS84	Manganese	COMPOSITE DATA			Observed	HI-VOL	2005	Microgram	No Events	17	100	Y	17	17	0	0
15	42	91	5	12110	1	34.06301	-81.0000	WGS84	Cadmium	24 HOUR			Observed	HI-VOL	1980	Microgram	No Events	57	93	Y	54	61	0	0
16	[illegible]	[illegible]	[illegible]	[illegible]	1	[illegible]	[illegible]	WGS84	Ozone	8-HR RUN	Ozone 8-H	Daily maximum of 8		[illegible]	Parts per m	No Events	[illegible]	99	Y	[illegible]	[illegible]	0	0
17	48	121	34	43043	1	33.21907	-97.1963	WGS84	Ethylene d	1 HOUR			Observed	Passivated	2005	Parts per b	No Events	41	100	Y	0	0	0	1
18	32	3	[illegible]	88101	1	[illegible]	[illegible]	WGS84	PM2.5 - Lo	24 HOUR	PM25 Ann	Quarterly Means		2010	Microgram	Events Inc	88	73	N	[illegible]	122	0	2
19	39	145	7	42401	1	38.7311	-82.9975	WGS84	Sulfur diox	1 HOUR	SO2 Annu	Observed	INSTRUM	1992	Parts per b	No Events	[illegible]	95	Y	350	365	0	0
20	40	55	9000	88324	1	38.30216	113.293	WGS84	OC1 PM2.5	24 HOUR			Observed	IMPROVE	2002	Microgram	No Events	85	70	N	85	122	0	37
21	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	WGS84	Calcium PM	24 HOUR			Observed	[illegible]	1995	Microgram	No Events	14	87	N	14	[illegible]	0	0
22	48	245	11	45225	1	29.89752	-93.9911	WGS84	123-Trime	24 HOUR			Observed	Passivated	2011	Parts per b	No Events	60	98	Y	59	61	0	0
23	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	WGS84	Tin PM2.5	24 HOUR			Observed	Met One	2014	Microgram	No Events	61	100	Y	61	[illegible]	0	0
24	12	107	1008	81102	1	29.08775	-81.0562	WGS84	PM10 Tota	24 HOUR	PM10 24-h	Daily Mean	HI-VOL SA	1986	Microgram	No Events	61	100	Y	61	61	0	0
25	[illegible]	[illegible]	[illegible]	[illegible]	1	[illegible]	[illegible]	WGS84	Sample Fl	24 HOUR			Observed	Andersen	2000	Percent	No Events	59	48	N	57	122	0	1
26	[illegible]	109	5210	[illegible]	1	[illegible]	[illegible]	WGS84	Manganese	24 HOUR			Observed	HI-VOL-HW	1999	Nanogram	No Events	30	80	N	30	[illegible]	0	8
27	12	115	5	68105	5	34.2632	86.3048	WGS84	Sample Mi	24 HOUR			Observed	Andersen	2003	Millimeter	No Events	55	90	Y	55	61	0	6
28	[illegible]	14	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	WGS84	[illegible]	24 HOUR			Observed	[illegible]	2013	Parts per b	No Events	15	71	N	15	[illegible]	0	[illegible]
29	55	119	8001	88131	5	45.2066	-90.5972	WGS84	Indium PM	24 HOUR			Observed	Met One S	2012	Microgram	No Events	59	97	Y	59	61	0	2

Figure 1- Database Snippet 1

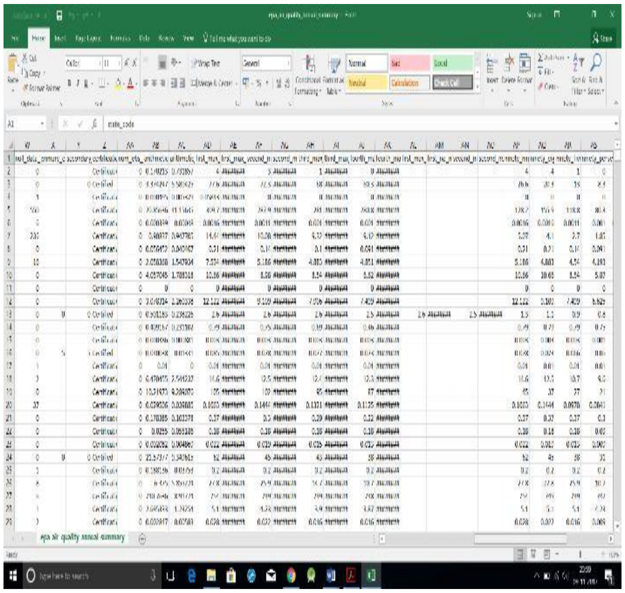


Figure 2- Database Snippet 2

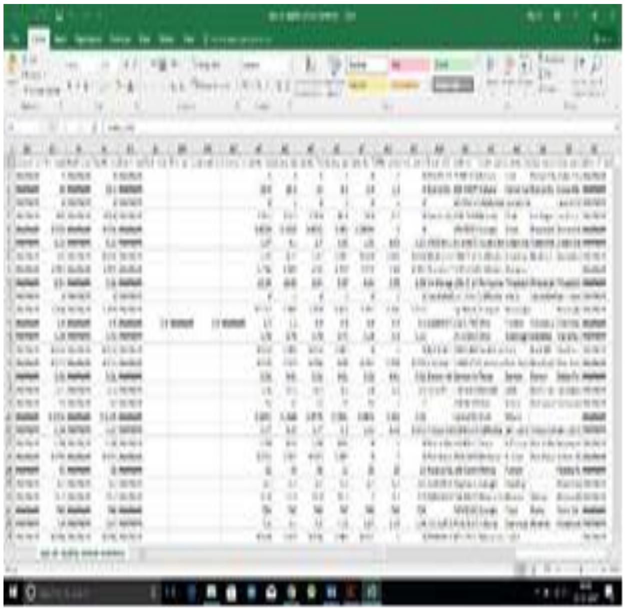


Figure 3- Database Snippet 3

Based upon this dataset we have formulated two problem statements which will be executed on all three platforms.

Problem Statement 1- Sort all the parameter affecting the air quality according to mean values recorded over the year.

Problem Statement 2- Sort all the states which are affected due to parameters present in air according to standard deviation values calculated over the years.

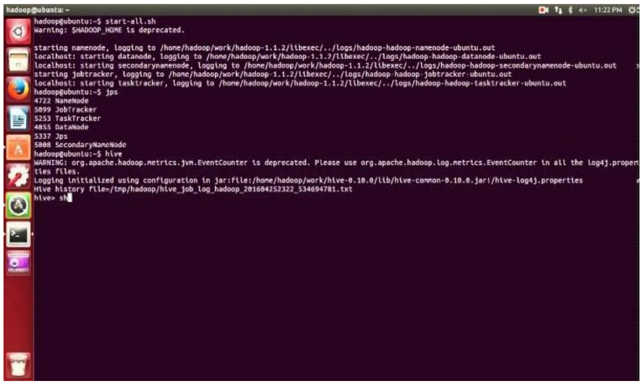
For running these queries, we need to setup all components and then we have to execute them. All major steps are listed as below-

1. Setting up Hadoop Multi node Cluster

First and the foremost step in our project was to setup Hadoop multi node cluster. For this we used VirtualBox to host our ubuntu 16.04 virtual machines. A master and a slave node setup was built up. Key steps involved in setting up the cluster are-

- □ Install VirtualBox and host Ubuntu 16.04 virtual machine
- □ Install Java on Ubuntu and setting up path variable in bash.rc file
- □ Create a user account
- □ In /etc/hosts file, slave IP address followed by name is mentioned
- □ Setting up SSH using RSA algorithm for secure communication without any prompt for password
- □ Install Hadoop in master node
- □ Configuring various Hadoop files like- **core-site.xml, hdfs-site.xml, mapred-site.xml** and **hadoop-env.sh**
- □ Install Hadoop on slave node
- □ Configuring Master and slave node
- □ Format name node on Master node and start the Hadoop services.

Using start-all.sh Hadoop services start and we can verify it using jps command as shown below-



```
hadoop@ubuntu: ~
hadoop@ubuntu:~$ start-all.sh
Warning: $HADOOP_HOME is deprecated.

starting namenode, logging to /home/hadoop/work/hadoop-1.1.2/libexec/../logs/hadoop-hadoop-namenode-ubuntu.out
localhost: starting datanode, logging to /home/hadoop/work/hadoop-1.1.2/libexec/../logs/hadoop-hadoop-datanode-ubuntu.out
localhost: starting secondarynamenode, logging to /home/hadoop/work/hadoop-1.1.2/libexec/../logs/hadoop-hadoop-secondarynamenode-ubuntu.out
starting jobtracker, logging to /home/hadoop/work/hadoop-1.1.2/libexec/../logs/hadoop-hadoop-jobtracker-ubuntu.out
localhost: starting tasktracker, logging to /home/hadoop/work/hadoop-1.1.2/libexec/../logs/hadoop-hadoop-tasktracker-ubuntu.out
hadoop@ubuntu:~$ jps
4722 NameNode
5099 JobTracker
5253 TaskTracker
4855 DataNode
5337 Jps
5008 SecondaryNameNode
hadoop@ubuntu:~$ hive
WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated. Please use org.apache.hadoop.log.metrics.EventCounter in all the log4j.proper
ties files.
Logging initialized using configuration in jar:file:/home/hadoop/work/hive-0.10.0/lib/hive-common-0.10.0.jar!/hive-log4j.properties
Hive history file=/tmp/hadoop/hive_job_log_hadoop_201604252322_534694781.txt
hive> sh
```

Figure 4- Hadoop Installed

Then we have to load data into Hadoop file system. Command used to put data from local directory into Hadoop FS is-hadoopfs -put /path/in/linux /hdfs/path

2. Setting up and Using Pig

Now that we have multi node Hadoop cluster running, we can install pig and use Pig Latin to query our dataset. Steps involved in setup-

- Download and unzip Pig tar file from the official Apache Pig Website.
- Edit the bash.rc file and setup the environment variables
- Compile the bash.rc file
- Check for pig –version

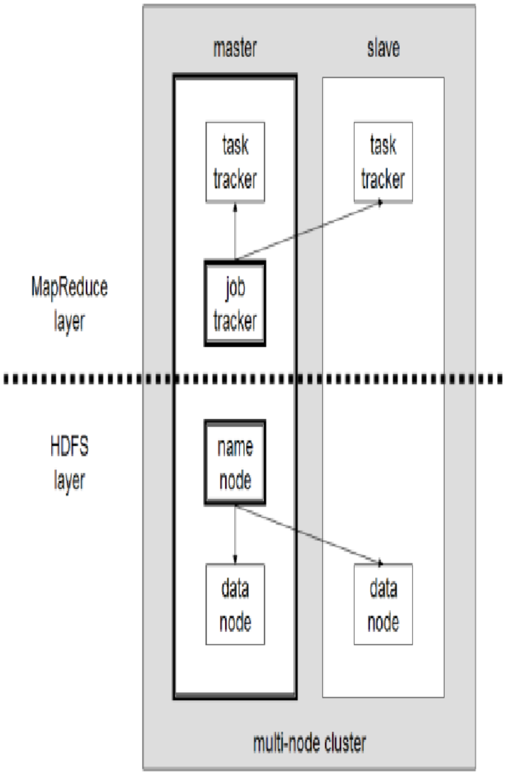


Figure 5- Hadoop Cluster

Now that the pig is successfully setup, we enter into Grunt shell where we can run pig commands to meet our requirements.

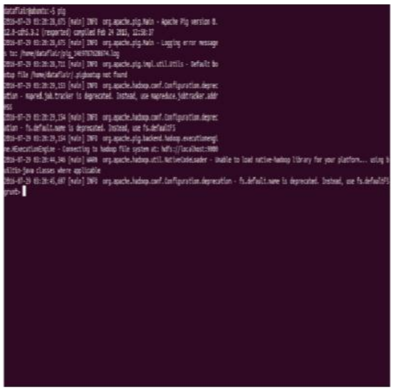
Following are the steps that we have taken to execute queries-

For generating the results of first query we have to execute following commands in grunt shell of Pig.

- abcde = LOAD '/pigInput' using PigStorage(',') AS (pname:chararray,mean:chararray);
- dump abcde;
- mean_order= order abcde by mean desc;
- dump mean_order;
- STORE mean_order into '/pigorderresult';

For generating the results of second query we have to execute following commands in grunt shell of Pig.

- abc = LOAD '/pigInput2' using PigStorage(',') AS (sdev:chararray,state:chararray);
- dump abc;
- sdev_order= order abc by sdevdesc;
- dump sdev_order;
- STORE mean_order into '/pigsdevorderresult';

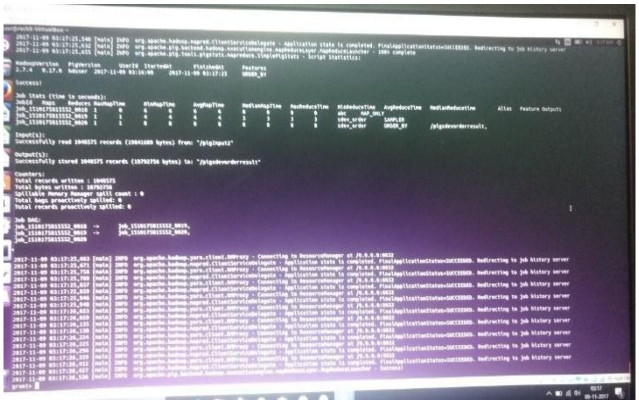


```
dataflair@ubuntu:~$ pig
2016-07-29 03:28:28,675 [main] INFO  org.apache.pig.Main - Apache Pig version 0.
12.0-cdh5.3.2 (rexported) compiled Feb 24 2015, 12:58:37
2016-07-29 03:28:28,675 [main] INFO  org.apache.pig.Main - Logging error message
s to: /home/dataflair/pig_1469787628674.log
2016-07-29 03:28:28,711 [main] INFO  org.apache.pig.impl.util.Utils - Default bo
otup file /home/dataflair/.pigbootup not found
2016-07-29 03:28:29,153 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addr
ess
2016-07-29 03:28:29,154 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2016-07-29 03:28:29,154 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - Connecting to hadoop file system at: hdfs://localhost:9000
2016-07-29 03:28:44,346 [main] WARN  org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your platform... using b
uiltin-java classes where applicable
2016-07-29 03:28:45,487 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt>
```

Figure 6- Pig Installed

Output generated from both the queries was stored on HDFS in files pigorderresult and pigsdevorderresult.

Below is the real-time photo after the query was executed successfully.



```
2017-11-09 03:17:25,540 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2017-11-09 03:17:25,632 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 100% complete
2017-11-09 03:17:25,655 [main] INFO  org.apache.pig.tools.pigstats.mapreduce.SimplePigStats - Script Statistics:

HadoopVersion   PigVersion   UserId   StartedAt             FinishedAt            Features
2.7.4           0.17.0       hduser   2017-11-09 03:16:00   2017-11-09 03:17:25   ORDER_BY

Success!

Job Stats (time in seconds):
JobId   Maps   Reduces   MaxMapTime   MinMapTime   AvgMapTime   MedianMapTime   MaxReduceTime   MinReduceTime   AvgReduceTime   MedianReducetime   Alias   Feature Outputs
job_1510175815552_0018  1  0  6  6  6  6  0  0  0  0  abc  MAP_ONLY
job_1510175815552_0019  1  1  4  4  4  4  3  3  3  3  sdev_order  SAMPLER
job_1510175815552_0020  1  1  8  8  8  8  8  8  8  8  sdev_order  ORDER_BY  /pigsdevorderresult,

Input(s):
Successfully read 1048575 records (15041689 bytes) from: "/pigInput2"

Output(s):
Successfully stored 1048575 records (18792756 bytes) in: "/pigsdevorderresult"

Counters:
Total records written : 1048575
Total bytes written : 18792756
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1510175815552_0018  ->  job_1510175815552_0019,
job_1510175815552_0019  ->  job_1510175815552_0020,
job_1510175815552_0020


2017-11-09 03:17:25,603 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2017-11-09 03:17:25,671 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2017-11-09 03:17:25,756 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2017-11-09 03:17:25,789 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2017-11-09 03:17:25,837 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2017-11-09 03:17:25,859 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2017-11-09 03:17:25,946 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2017-11-09 03:17:25,940 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2017-11-09 03:17:26,053 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2017-11-09 03:17:26,059 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2017-11-09 03:17:26,135 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2017-11-09 03:17:26,138 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2017-11-09 03:17:26,224 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2017-11-09 03:17:26,225 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2017-11-09 03:17:26,296 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2017-11-09 03:17:26,339 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2017-11-09 03:17:26,412 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /0.0.0.0:8032
2017-11-09 03:17:26,427 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2017-11-09 03:17:26,536 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
grunt>
```

Figure 7- Query Executed on Pig

3. Setting up and using Hive

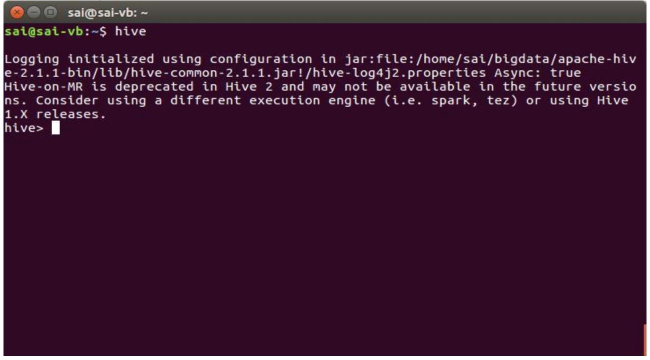
Now since we had multi node Hadoop cluster running and also executed Pig queries, our next aim was to install Hive and use HiveQL to query our dataset. Steps involved in setup-

- Download and unzip Hive tar file from the official Apache Hive Website
- Edit the bash.rc file and setup the environment variables
- Compile the bash.rc file
- Check for hive -version
- Create Hive directories within HDFS. The directory 'warehouse' is the location to store the table or data related to hive

- Configure Hive files.
    - Initialize Derby database

    - Launch Hive

6- Pig **Figure 8-** Hive Installed



```
sai@sai-vb:~$ hive

Logging initialized using configuration in jar:file:/home/sai/bigdata/apache-hiv
e-2.1.1-bin/lib/hive-common-2.1.1.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versio
ns. Consider using a different execution engine (i.e. spark, tez) or using Hive
1.X releases.
hive>
```

Figure 8- Hive Installed

Now that the hive is successfully setup, we enter into hive shell where we can run commands to meet our requirements.

Following are the steps that we have taken to execute queries-

For generating the results of first query we have to execute following commands in hive shell of Hive.

- CREATE TABLE ABCDE
  (
  PNAME STRING,
  MEAN FLOAT
)
  ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  :
- LOAD DATA LOCAL INPATH '/home/hduser/Downloads/Book1.csv' OVERWRITE INTO TABLE ABCDE

- INSERT OVERWRITE DIRECTORY '/tmp/hive' SELECT * FROM ABCDE SORT BY MEAN DESC;

For generating the results of second query we have to execute following commands in hive shell of Hive.

- CREATE TABLE ABCDEF

  (

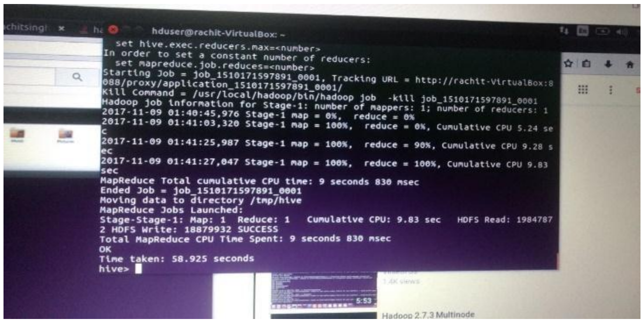
SDEV FLOAT,
STATE
STRING
)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
;

- INSERT OVERWRITE DIRECTORY '/USER' SELECT * FROM ABCDEF SORT BY SDEV DESC;

Output generated from both the queries was stored on HDFS in directories /tmp/hive and /USER.

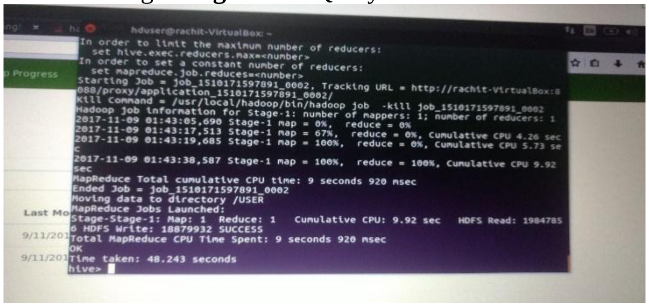
Below are some of the real-time photo after the query was executed successfully.



```
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1510171597891_0001, Tracking URL = http://rachit-VirtualBox:8
088/proxy/application_1510171597891_0001/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1510171597891_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-11-09 01:40:45,976 Stage-1 map = 0%,  reduce = 0%
2017-11-09 01:41:03,320 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.24 se
c
2017-11-09 01:41:25,987 Stage-1 map = 100%,  reduce = 90%, Cumulative CPU 9.28 s
ec
2017-11-09 01:41:27,047 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 9.83
sec
MapReduce Total cumulative CPU time: 9 seconds 830 msec
Ended Job = job_1510171597891_0001
Moving data to directory /tmp/hive
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 9.83 sec   HDFS Read: 1984787
2 HDFS Write: 18879932 SUCCESS
Total MapReduce CPU Time Spent: 9 seconds 830 msec
OK
Time taken: 58.925 seconds
hive>
```

Figure 9- Query 1 executed

6- Pig In **Figure 10-** Query 2 executedstalled



```
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1510171597891_0002, Tracking URL = http://rachit-VirtualBox:8
088/proxy/application_1510171597891_0002/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1510171597891_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-11-09 01:43:05,690 Stage-1 map = 0%,  reduce = 0%
2017-11-09 01:43:17,513 Stage-1 map = 67%,  reduce = 0%, Cumulative CPU 4.26 sec
2017-11-09 01:43:19,685 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.73 se
c
2017-11-09 01:43:38,587 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 9.92
sec
MapReduce Total cumulative CPU time: 9 seconds 920 msec
Ended Job = job_1510171597891_0002
Moving data to directory /USER
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 9.92 sec   HDFS Read: 1984785
6 HDFS Write: 18879932 SUCCESS
Total MapReduce CPU Time Spent: 9 seconds 920 msec
OK
Time taken: 48.243 seconds
hive>
```

Figure 10- Query 2 executed

4. Setting up and using R

Rstudio can be downloaded from its official website www.rstudio.com

Following are the steps that we have taken to execute queries-

□ reading in csv
r2 <- order(b2$SD,decreasing= TRUE)

□ arranging entire dataFrameacc to columns "mean_val" and "SD"
b1<-read.csv(file.choose())

b2<-read.csv(file.choose())

newb1<-b1[r1,]

newb2<-b2[r2,]

- setting column name

colnames(b2) <- c("SD","city")

```
colnames(b1) <- c("para_name","mean_val")
```

- Arranging columns “mean_val” and “SD” IN decending order

```
r1 <- order(b1$mean_val,decreasing= TRUE)
```

□ reading out the csv files

```
write.csv(newb1, file = "MyData.csv")
write.csv(newb2, file =
"MyData22.csv")
```

Therefore, we have executed both the queries on same dataset in all three technologies. In the next section we will be analysing the results generated.

The following results were obtained after executing queries-

1. **Pig Results**

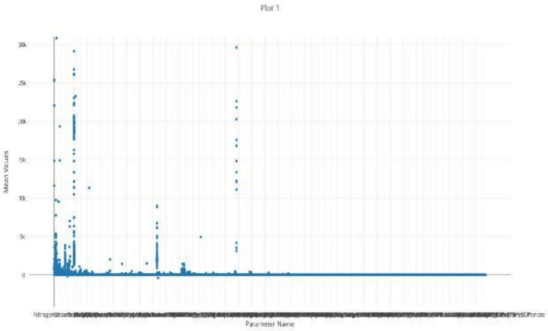


Figure 11- First Problem Statement Output

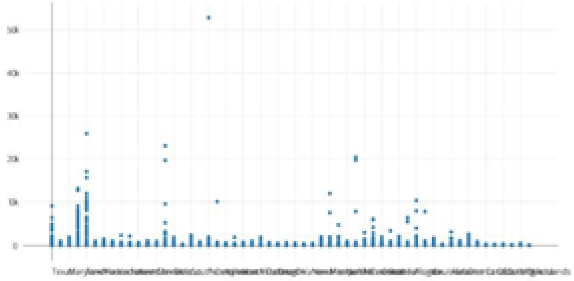


Figure 12- Second Problem Statement Output

2. **Hive Results**

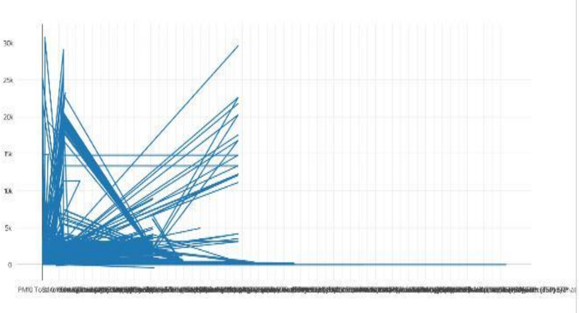


Figure 13- First Problem Statement Output

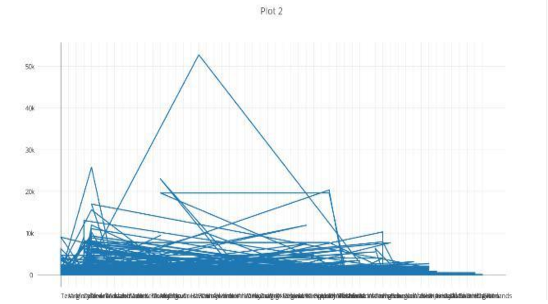


Figure 14- Second Problem Statement Output

3. R Results

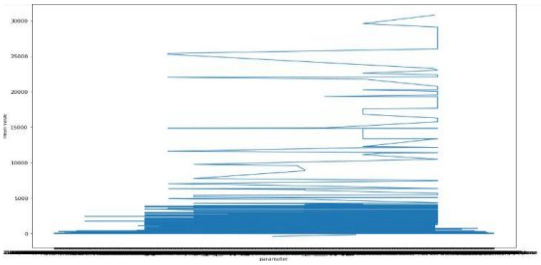


Figure 15- First Problem Statement Output

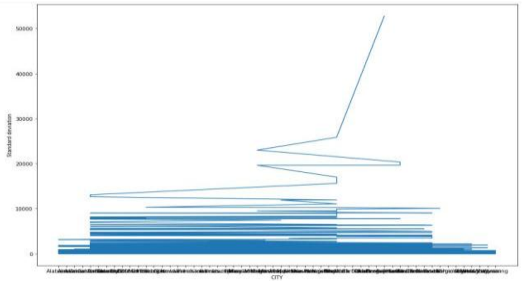


Figure 16- Second Problem Statement Output

Results are obtained and to make useful analysis from them we had to use python for plotting the graphs using python library. So, both the queries are executed successfully and results of both the queries are realized using different type of graphs. It can be easily understood that due to presence of big amount of data, X-axis parameters are overlapping.

4. Now below are the graphs of execution times of both the queries in all three technologies-

□ Query 1 execution time comparison-

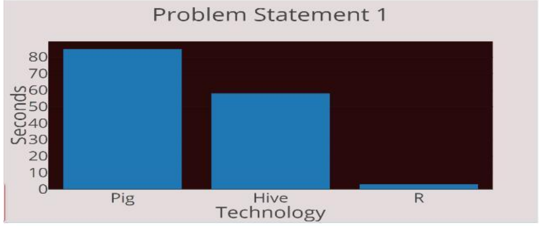


Figure 17- Comparison of First Problem Statement ExecutionTimes

- Query 2 execution time comparison-

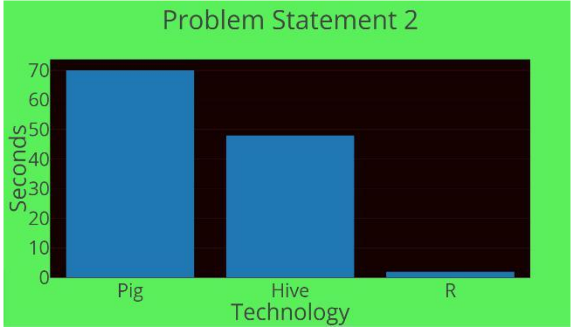


Figure 18- Comparison of Second Problem Statement Execution Times

Therefore, it can be easily observed that execution time of both the queries in R is minimum as compared to other two technologies.

IV. RESULTS AND DISCUSSION

In this we proposed the analysis of comparing three major technologies of handling big data i.e. Pig, Hive and R. Journey started from firstly setting up all the technologies in our virtual machine setup. Secondly, we had to formulate the queries and execute them accordingly in all the three technologies listed above. After successful results obtained, it can be clearly observed that query execution time in R is minimum as compared to Pig and Hive. Also, among Pig and Hive execution time is minimum in Hive. But this does not mean that in every case Pig will be slower than hive or hive can take place of Pig. It can be stated that depending upon the user requirement both the technologies can be used as both are based on map reduce concept. R is surely overtaking the big data business of Big Data by providing fancy algorithms of analysing these large datasets and also efficiently.

This research can take many paths from here like more technologies can be used for comparison of efficiency of big data technologies. Thereby, giving more accurate and more enhanced results. Also, from here we can go on to analyse streaming datasets using Apache Kafka, Spark etc. as the world is moving towards streaming data and real-time analysis is the need of the hour.

REFERENCES

[1] SherifSakr, AmalElgammal, Towards a Comprehensive Data Analytics Framework for Smart Healthcare Services, Intl Journal of Big Data Research (Elsevier), Pg:44-58, Vol. 4, (2016)

[2] Zhijiiang Chen, GuobinXu, VivekMahalingam, LinqiangGe, James Nguyen, Wei Yu, Chao Lu, A Cloud Computing Based Network Monitoring and Threat Detection System for Critical Infrastructures, Pg:44-58, Vol. 4, (2016)

[3] Xiaolong Jin, Benjamin W. Wah, Xueqi Cheng, Yuanzhuo Wang, Signifance and Challenges of Big Data Research, Pg:59-64, Vol. 2, (2015)

[4] Mohammad NaimurRahman, Amir Esmailpour, Junhui Zhao, Machine Learning Generation Forecasting System, Pg:9-15, Vol. 5, (2015)

[5] Yaxiao Liu, Henan Wang, Guoliang Li, JunyangGao, Huiqi Hu, Wen-Syan Li, ELAN: An Efficient Location-Aware Analytics System, , Pg:16-21, Vol. 5, (2016)

[6] SurajitChaudhuri, What Next? A Half-Dozen Data Management Research Goals for Big Data and the Cloud

[7] Daniel Nunan, Maria Di Domenico, Market research & the ethics of big data

[8] Jeffrey Dean, Sanjay Ghemawat, MapReduce Simplified Data Processing on Large Clusters.

[9] Hsinchun Chen, Roger H. L. Chiang, Veda C. Storey, BUSINESS INTELLIGENCE AND ANALYTICS: FROM BIG DATA TO BIG IMPACT

[10] Kyong-Ha Lee Yoon-Joon Lee, Hyunsik Choi Yon Dohn Chung, Bongki Moon, Parallel Data Processing with MapReduce: A Survey, SIGMOD Record, December 2011 (Vol. 40, No. 4)

[11] AbdelrahmanElsayed, Osama Ismail, and Mohamed E. El-Sharkawi, MapReduce: State-of-the-Art and Research Directions.

[12] Xindong Wu, Xingquan Zhu, Gong-Qing Wu and Wei Ding, Data Mining with Big Data (IEEE), Vol. 26, NO. 1, JANUARY 2014

[13] Michele De Gennaro, Elena Paffumi, Giorgio Martini, Big Data for Supporting Low-Carbon Road Transport Policies in Europe: Applications, Challenges and Opportunities, Intl Journal of Big Data Research (Elsevier), 2 June 2016

[14] Cui Yu, Josef Boyd, FB+- tree for Big Data Management, Intl Big Data Research (Elsevier), Pg: 25-36, Vol. 4, June 2016

[15] Diamantoulakis, P.D., Kapinas, V.M. Karagiannidis, G.K., Big Data Analytics for Dynamic Energy Management in Smart Grids, Intl Big Data Research (Elsevier), Pg: 94-101, Vol. 2, September 01 2015

[16] Paakkonen, P., Pakkala, D., Reference Architecture and Classification of Technologies, Products and Services for Big Data Systems, Intl Big Data Research (Elsevier), Pg: 166-186, Vol. 2, December 01 2015

[17] Bong-Hwa-Hong and Hae-Jong Joo, A Study on The Monitoring Model for Traffic Analysis and Application of Big Data, Intl Research on Big Data (Elsevier), Pg.- 30-35, Vol. 43, 2013

[18] Mikin K. Dagli and Brijesh B. Mehta, Big Data and Hadoop: Review, Intl research on Big Data (Elsevier), Pg.-192-196, Vol.2, February 2014

[19] Thusoo, Ashish, JoydeepSenSarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu, and Raghotham Murthy. "Hive-a petabyte scale data warehouse using hadoop." In Data Engineering (ICDE), 2010 IEEE 26th International Conference on, pp. 996-1005. IEEE, 2010

[20] Olston, Christopher, Benjamin Reed, UtkarshSrivastava, Ravi Kumar, and Andrew Tomkins. "Pig latin: a not-so-foreign language for data processing." In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pp. 1099-1110. ACM, 2008

[21] Gates, Alan F., Olga Natkovich, Shubham Chopra, PradeepKamath, Shravan M. Narayanamurthy, Christopher Olston, Benjamin Reed, SanthoshSrinivasan, and UtkarshSrivastava. "Building a high-level dataflow system on top of Map-Reduce: the Pig experience." Proceedings of the VLDB Endowment 2, no. 2 (2009): 1414-1425