ISSN 2320 - 2602
Volume 7 No.6, June 2018
International Journal of Advances in Computer Science and Technology
Available Online at http://www.warse.org/IJACST/static/pdf/file/ijacst05762018.pdf
https://doi.org/10.30534/ijacst/2018/05762018

Performance Analysis of Big Data Tools

Ibrahim Rai[1], Fahad Al Qurashi[2]
[1]Department of Computer Science KAU, KSA, ir4group@gmail.com
[2]Department of Computer Science KAU, KSA, fahad@kau.edu.sa

ABSTRACT

The limitation in traditional Database Management Systems to process and analysis the growing data from various sources has led to emergence of Big Data. A lot of tools have been developed under Hadoop architecture to utilize the powerful functionality provide by Hadoop. In this paper, we have evaluated the performance of Big Data tools, Pig and Hive. We have shown that Hive is 2.82 faster than Pig with respect to data size ranging from one hundred thousand to one million rows. Moreover, Hive achieved 3.95 faster performance over Pig when more complex query is performed. However, the learning curve of Pig is higher than Hive, but the last required preparing the data warehouse before starting processing them.

Key words: Database, Big Data, Performance of Big Data.

1. INTRODUCTION

The limitation in traditional Database Management Systems to process and analysis the growing data from various sources has led to emergence of Big Data. Big Data is more capable in analyzing and processing large collections of unstructured data [1]. One common implementation of Big Data is Hadoop, there are several tools used under Hadoop architecture for processing data, some of these tools are Pig and Hive.

In this paper, we aim to evaluate the performance of Big Data tools, Pig and Hive with respect to execution time needed to process Data resided in HDFS. We performed our experiment on CentOS 6.7, an open source Linux distribution and Cloudera, an open source Hadoop distribution.

We have prepared the data in our experiment from a real world example by gathering data from an SQL database of a training institute, in this database we have seven tables Employees, Areas, Locations, Groups, Main Categories, Sub Categories and Enrollments. For enrollments table, we have three different size of tables, one hundred thousand rows, five hundred thousand rows and one million rows.

To manipulate data, we use scripts in Pig and queries in Hive, in addition before processing data using Pig we have to load them in relations despite Hive which directly deals with tables. We started by computing the execution time needed to process one relation/table to count number of record with data size ranging from one hundred thousand to one million rows.

Then we calculated the execution time to process two joined relations/tables with the same data size in order to count number of enrolled employees per area.

We also computing the execution time to process three joined relations/tables to count number of enrolled employees per profession type per area.

We have also evaluated the impact of data size in addition to language complexity used to retrieve data from HDFS on the performance of both Pig and Hive.

2. RELATED WORK

Wlodarczyk et al. [2] analyze the performance of Hadoop cluster in query processing for NoSQL, they found that the performance limitation comes from IO disks. Carstoui et al. [3] evaluate the performance of HBase different versions, Hbase-0.20.0 and Hbase-0.20.20, they show that the later version is superior. Huang et al. [4] propose an HBase design with Remote Direct Access Memory capability to evaluate HBase performance, the study finds that the new design achieves 3 times faster than the traditional one. Khaliq et al. [5] demonstrate performance of CPU both Dual Core and Core i5 and power consumption when implementing Hadoop, the result of this study shows that Core i5 has better performance and less power consumption. Alshammari et al. [6] propose a new design of Hadoop architecture, the new design shows a performance improvement with respect to the traditional design by reducing the size of the read and executed data.

These works papers address a single aspect of big data with focus on evaluation.

Vora [7] evaluates a hybrid architecture that stores images in HDFS located in HBase and compare it with MySQL, the study concludes that HBase performs better and is more scalable than traditional database. Xiao [8] proposes a design that combines both structure and unstructured data, the study shows an improvement in data processing.

These research papers focus on the evaluation of multiple tools of big date with respect to the design, our work complements these works as both have different approaches.
Jogi et al. [9] compare the performance of three databases, MySQL, Cassandra and HBase, the authors find that Cassandra is the most scalable database with fast read and write performance while HBase is twice faster than MySQL.

Harleen et al. [10] demonstrate performance of PostgreSQL as structure data and Hadoop as unstructured data, the study shows that the load time using Hadoop has less latency and more capable in adopting new data.
Our work is similar to these works, but the big data tools we have evaluated have not addressed by any of these works.

3. BIG DATA

The limitation in traditional Database Management Systems to process and analysis the growing data from various sources has led to emergence of Big Data, which is more capable in analyzing and processing large collections of unstructured data [1]. Big Data has four characteristics; volume, velocity, variety, veracity and value [11]. Volume represents the large data generated by organization activities and users' interactions, while velocity means fast growth of data with respect to the time, variety refers to diverse formats of collected data, veracity reflects the quality of data, hence only valid data need to be processed after extracting the noise and value means the usefulness of the collected data when it is converted to information [11].

3.1. HADOOP

The concept of Hadoop is based on distributed storage and computation, it consists of HDFS (Hadoop Distributed File System), YARN (Yet Another Resource Negotiator) and MapReduce [12]. Figure 1 shows Hadoop architecture.

The main function of HDFS is to process large files and it has three components: NameNode, Secondary NameNode and DataNode [13]. NameNode is the master node that manages DataNodes, in addition the metadata of each files in the HDFS is stored in the NameNode while the Secondary NameNode is dedicated for additional functions that support the NameNode [13]. Large files in HDFS is divided into several blocks, those blocks are stored in the DataNodes, hence DataNode considered an identical nodes that reside on Hadoop cluster [13].
When working with Hadoop cluster means several nodes are bound together to perform tasks, this cluster has resources such as NameNode and DataNode. YARN is responsible for managing those resources and allocate them to a particular application [14].

The traditional method of dealing with data is storing them in a database to be processed through an SQL queries, however, this approach becomes inefficient when dealing with large and unstructured data, the MapReduce, as in Figure 2 developed to address this problem and provide a mechanism to divide large data into smaller units to be processed by the map function passing the result to the reduce function to produce the output [15].

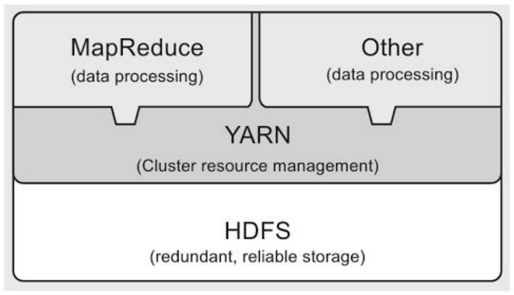

Figure 1 Hadoop Architecture [12]

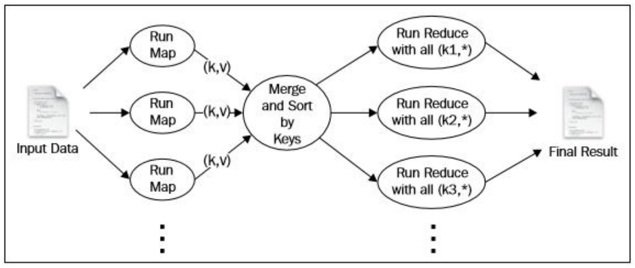

Figure 2 MapReduce Execution [15]

3.2. PIG

Pig is a dataflow language which allow the user to describe how the data would be read, in addition it takes advantage of Hadoop MapReduce functionality by providing an engine to the user to execute parallel operations by writing scripts using Pig Latin language [16]. HDFS stores the blocks of data on all nodes of the Hadoop cluster and Pig reads and processes the data from HDFS and writes the result back to HDFS [17].

3.3. HIVE

Hive is an SQL abstraction tool that provides data access through queries that run MapReduce jobs [18]. Hive relies on several components as shown in Figure 3.

Implementing Hive implies importing data to Hive warehouse and before this process a user should create a database and its tables, all these functionalities are facilitated by HCatalog, however, the metadata that define the schema of Hive tables reside on HCatalog while the data themselves are stored in HDFS [18]. Hive also offers an ODBC connectivity provided by Hiveserver2 to allow users access the data resides in Hive warehouse using business intelligence applications such as Microsoft Excel [18].

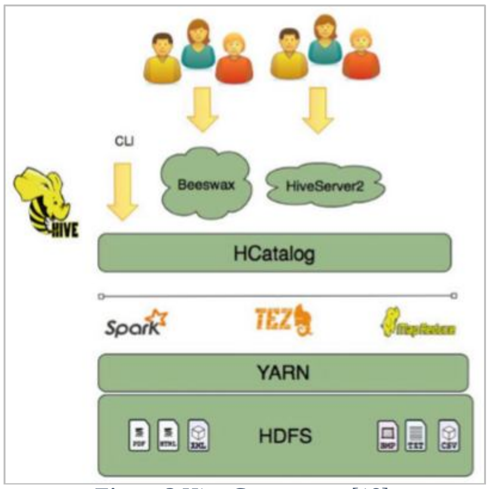

Figure 3 Hive Components [18]

4. EXPERIMENT

4.1. PIG TEST CASES

A. LOADING DATA FOR PIG TESTS

There are two ways to load the data to HDFS, CLI (Command Language Interface) and HUE (Hadoop User Experience) the free open source interface. In this experiment, we used the second method and loaded the only Enrollments table in the three different size as described in the previous section. In fact, it is a file that contains an unstructured data rather than a table because it has no rows and columns; it is just pieces of data in tab separated value.

B. QUERY DATA

We switch to Pig editor, then run the scripts and record the results of the query scripts of one relation, two joined relations and three joined relations. All query scripts were performed on the three different size files, one hundred thousand rows, five hundred thousand rows and one million rows.

i. ONE RELATIONS QUERY SCRIPT

Here we run the first script to count number of records insides Enrollments file.

Script 1 One relation query script

```
Courses = LOAD
'/user/Ibrahim_Experiment/Enrollments_HDF_100K.tsv';
Courses_GROUP = GROUP Courses ALL;
Courses_COUNT = FOREACH Courses_GROUP GENERATE
COUNT(Courses);
Store Courses_COUNT into
'/user/Ibrahim_Experiment/Pig_Output/Result1.out';
```

ii. TOW JOINED RELATIONS QUERY SCRIPT

Here we run the second script to count number of enrolled employees per area.

Script 2 Pig Tow joined relations query script

```
Courses = LOAD
'/user/Ibrahim_Experiment/Enrollments_HDF_100K.tsv'
as (EmpID:chararray, EmpName:chararray,
courseName:chararray,
sDate:datetime, eDate:datetime, location:chararray,
subCat:chararray,
mCat:chararray, costSR:float, Area:chararray,
empCat:chararray);
GroupedByArea = GROUP Courses by Area;
countByArea = FOREACH GroupedByArea GENERATE group,
COUNT(Courses) as countArea;
Store countByArea into
'/user/Ibrahim_Experiment/Pig_Output/Result2.out';
```

iii. THREE JOINED TABLES QUERY SCRIPT

Here we run the third script to count number of enrolled employees per profession type per area.

Script 3 Pig Three joined relations query script

```
Courses = LOAD
'/user/Ibrahim_Experiment/Enrollments_HDF_100K.tsv'
as (EmpID:chararray, EmpName:chararray,
courseName:chararray,
sDate:datetime, eDate:datetime, location:chararray,
subCat:chararray,
mCat:chararray, costSR:float, Area:chararray,
empCat:chararray);
GroupedByArea_empCat = GROUP Courses by
(Area,empCat);
GroupedByArea_empCat_count = foreach
GroupedByArea_empCat generate group.Area as
Area, group.empCat as empCat, COUNT(Courses) as
empCatCount;
GroupedByArea = group GroupedByArea_empCat_count by
Area;
result = foreach GroupedByArea{
startedArea = order GroupedByArea_empCat_count by
empCatCount desc;
Generate FLATTEN(startedArea);
};
Store result into
'/user/Ibrahim_Experiment/Pig_Output/Result3.out';
```

The elapsed time of all three Pig scripts are shown in table 1.

4.2. HIVE TEST CASES

A. LOADING DATA FOR HIVE TESTS

To query the data using Hive we have to first create the database, within Hue interface we have navigated to Metastore Manager, then created the database. After that we have created the tables and loaded the data to each. When performing this process, we have to carefully choose the schema of the data base by designing the tables with respect to keys that should be matched between tables. The Metastore does not impose any constrains between tables as the relations will be created during execution.

B. QUERY DATA

We switch to Hive editor, then run the queries and record the results of the one table query, two joined tables and three joined tables. All queries were performed on the three different size tables, one hundred thousand rows, five hundred thousand rows and one million rows.

i. ONE TABLE QUERY

Here we run the first query to count number of records insides Enrollments file.

Table 1 Pig scripts elapsed time

Rows	Script	Test 1	Test 2	Test 3	Average
100K	Script 1	29	29	30	29
	Script 2	36	35	30	34
	Script 3	57	57	57	57
500K	Script 1	29	29	30	29
	Script 2	35	35	34	35
	Script 3	61	60	60	60
1000K	Script 1	34	35	35	35
	Script 2	45	40	39	41
	Script 3	66	66	65	66

Query 1 Hive One table query

```
Insert overwrite directory
'/user/Ibrahim_Experiment/Hive_Output/Result1.out'
row format delimited fields terminated by '\t'
stored as textfile
select count(enrollment_course_name) as
NumberOfRecords from enrollments_100k;
```

ii. TOW JOINED RELATIONS QUERY

Here we run the second query to count number of enrolled employees per area.

Query 2 Hive Tow joined relations query

```
Insert overwrite directory
'/user/Ibrahim_Experiment/Hive_Output/Result1.out'
row format delimited fields terminated by '\t'
stored as textfile
select a.location_name,
count(enrollment_course_name) as
number_of_participant
from areas a inner join enrollments_100k en on
a.location_id = en.enrollment_area_id group by
a.location_name;
```

iii. THREE JOINED TABLES QUERY

Here we run the third query to count number of enrolled employees per profession type per area.

Query 3 Hive Three joined relations query

```
Insert overwrite directory
'/user/Ibrahim_Experiment/Hive_Output/Result3.out'
row format delimited fields terminated by '\t'
stored as textfile
select a.location_name, g.group_name, count
(enrollment_course_name) as number_of_participant
from
areas a inner join enrollments_100k en on
a.location_id = en.enrollment_area_id inner join
groups g on g.group_id = en.enrollment_group_id
group by a.location_name, g.group_name;
```

The elapsed time of all three Hive queries are shown in table 2.

5. EVALUATION

In this section, we evaluate the performance of Pig and Hive with respect to execution time needed to process one relation/table to count number of record with data size ranging from one hundred thousand to one million rows.

As shown in Figure 4, we found that Pig created two jobs to perform this action and the execution time is directly proportional to the size of data while the size of data has no impact on the execution time when performing the same action using Hive.

Figure 5 shows that there was slight increase in execution time for Hive while Pig continued demanding more time and creating two jobs when processing two joined relations/tables with data size ranging from one hundred thousand to one million rows to count number of enrolled employees per area.

Table 2 Hive Query elapsed time

Rows	Query	Test 1	Test 2	Test 3	Average
100K	Query 1	11	11	11	11
	Query 2	11	11	11	11
	Query 3	12	11	11	11
500K	Query 1	10	11	11	11
	Query 2	12	12	12	12
	Query 3	12	13	13	13
1000K	Query 1	11	11	12	11
	Query 2	12	12	12	12
	Query 3	13	13	13	13

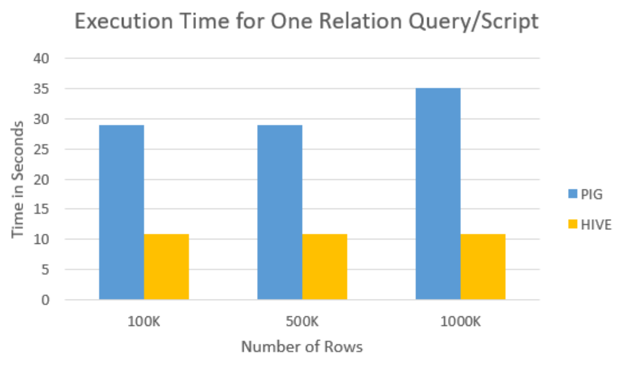

Figure 4 Elapsed Time for Processing 1 Table/Relation

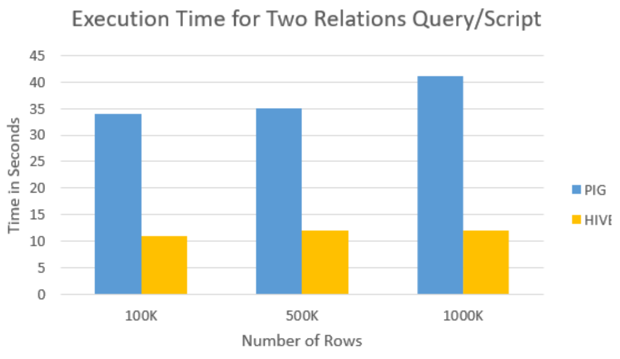

Figure 5 Elapsed Time for Processing 2 Tables/Relations

We have noticed that Pig needed three jobs to count number of enrolled employees per profession type per area which implies joining three relations with data size ranging from one hundred thousand to one million rows and the time it toke is 1.7 greater than the time needed to process two relations, while Hive showed no significant change in processing both two and three joined tables as shown in Figure 6.

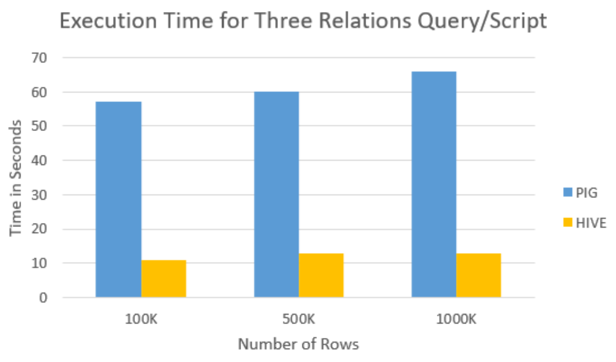

Figure 6 Elapsed Time for Processing 3 Tables/Relations

We have also evaluated the impact of data size on the performance of both Pig and Hive as shown in Figure 7 by analyzing the data from Table 1 and Table2, we have found that Hive is 2.82 faster than Pig.

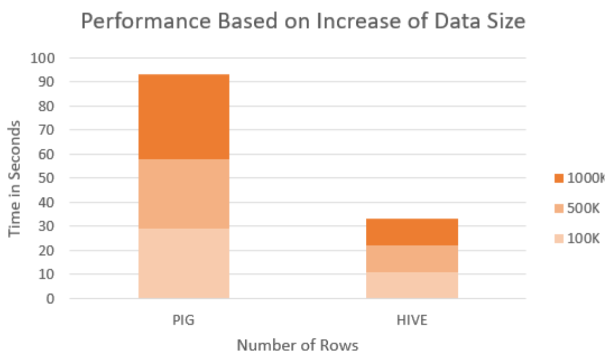

Figure 7 Comparison of Pig and Hive performance with respect to data size,

$$\text{Performance} = \frac{Pig \sum t}{Hive \sum t} = \frac{29 + 29 + 35}{11 + 11 + 11} = \frac{93}{33} = 2.82$$

Where t is the execution time observed by Pig and Hive.

Hive also showed better performance over Pig with respect to language complexity used to retrieve data from HDFS as it scored 3.95 faster performance.

$$\text{Performance} = \frac{Pig \sum t}{Hive \sum t} = \frac{35 + 41 + 66}{11 + 12 + 13} = \frac{142}{36} = 3.95$$

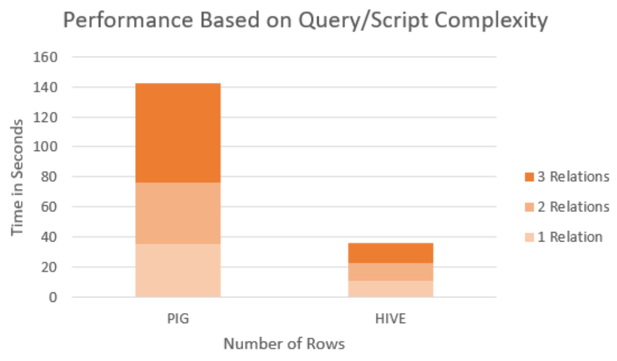

Figure 8 Comparison of Pig and Hive performance with respect to language complexity

6. CONCLUSION

This paper has presented a performance evaluation of Big Data tools, Pig and Hive. We have shown that Hive is 2.82 faster than Pig with respect to data size ranging from one hundred thousand to one million rows. Moreover, we have demonstrated that the more complex language used to retrieve data from HDFS the better performance Hive achieved over Pig which went to 3.95 faster performance. However, Pig required special skills to write Pig scripts but no additional configuration when loading the data to HDFS and it is capable of processing unstructured data while Hive query is similar to SQL and required no additional skills but implied creating a database with schema and importing the data to the Metastore.

REFERENCES

1. T. Erl, W. Khattak and P. Buhler. ***Big Data Fundamentals: Concepts, Drivers & Techniques***, 1st ed. Upper Saddle River, NJ: Prentice Hall, 2016, pp. 71-72.
2. T. W. Wlodarczyk, Y. Han and C. Rong. **Performance Analysis of Hadoop for Query Processing**, *2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications*, Biopolis, 2011, pp. 507-513..
3. D. Carstoui, A. Cermian and A. Olteanu. **Hadoop Hbase-0.20.2 Performance Evaluation**, *4th International Conference on New Trends in Information Science and Service Science*, Gyeongju, 2010, pp. 84-87
4. J. Huang, et al. **High-Performance Design of HBase with RDMA over InfiniBand**, *2012 IEEE 26th International Parallel and Distributed Processing Symposium*, Shanghai, 2012, pp. 774-785.
5. Y. Khaliq, G. Abbas, A. Qureshi and F. Zeeshan. **Calculation of CPU Performance, Power and Cost using Hadoop**, *2016 Sixth International Conference on Innovative Computing Technology (INTECH)*, Dublin, 2016, pp. 122-127.
https://doi.org/10.1109/INTECH.2016.7845093

6. H. Alshammari, H. Bajwa and J. Lee. **Enhancing Performance of Hadoop and Mapreduce for Scientific Data using NoSQL Database**, *2015 Long Island Systems, Applications and Technology, Farmingdale, NY, 2015, pp. 1-5.*
7. M. N. Vora. **Hadoop-HBase for Large-Scal Data**, *Proceedings of 2011 International Conference on Computer Science and Network Technology*, Harbin, 2011, pp. 601-605.

   https://doi.org/10.1109/ICCSNT.2011.6182030
8. X. Dawei. **Exploration on Big Data Oriented Data Analyzing and Processing Technology**, *International Journal of Computer Science Issues (IJCSI)*, vol. 10, no. 1, pp. 13-18, Jan. 2013.
9. V. D. Jogi and A. Sinha. **Performance Evaluation of MySQL**, Cassandra and HBase for Heavy Write Operation", *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, Dhanbad, 2016, pp. 586-590.
10. Harleen and N. Garg. **Analysis of Hadoop Performance And Unstructured Data Using Zeppelin**, *2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS)*, Bangalore, 2016, pp. 1-6.
11. B. Bengfort and J. Kim. ***Data Analytics with Hadoop***, Sebastopol, CA :O'REILLY, 2016, pp. 131-224.
12. V. Jain. ***Big Data and Hadoop***, New Delhi, India: Khanna Book Publishing Co. Ltd., 2017, pp. 69-70.
13. S. Wadkar, M. Siddalingaiah and J. Venner. ***Pro Apache Hadoop***, New Yourk: Apress, 2014, pp. 16-17.
14. G. Turkington. ***Hadoop Beginner's Guide***, Birmingham, UK: Packt Publishing Ltd.,2013, pp. 348.
15. S. Perera and T. Gunarathne. ***Hadoop MapReduce Cookbook***, Birmingham, UK: Packt Publishing Ltd., 2013, pp. 7-8.
16. A. Gates. ***Programming Pig***, Sebastopol, CA: O'REILLY Media Inc., 2011, pp. 4-5.
17. P. Pasupuleti. ***Pig Design Patterns***, Birmingham, UK: Packt Publishing Ltd., 2014, pp. 26-28.
18. S. Shaw, A. Vermeulen, A. Gupta and D. Kjerrumgaard. ***Practical Hive: A Guide to Hadoop's Data Warehouse System***, New Yourk: Apress, 2016, pp. 37-40.