Processing Performance on Apache Pig, Apache Hive and MySQL Cluster

Ammar Fuad, Alva Erwin, Heru Purnomo Ipung

Information Technology, Swiss German University
Edutown BSD City, Tangerang 15339, Indonesia

[1]ammar.fuad[at]student.sgu.ac.id

[2]alva_erwin[at]yahoo.com

[3]heru.ipung[at]sgu.ac.id

Abstract—**MySQL Cluster is a famous clustered database that is used to store and manipulate data. The problem with MySQL Cluster is that as the data grows larger, the time required to process the data increases and additional resources may be needed. With Hadoop and Hive and Pig, processing time can be faster than MySQL Cluster. In this paper, three data testers with the same data model will run simple queries and to find out at how many rows Hive or Pig is faster than MySQL Cluster. The data model taken from GroupLens Research Project [12] showed a result that Hive is the most appropriate for this data model in a low-cost hardware environment.**

Keywords— **Hadoop; Hive; Pig; MySQL; MySQL Cluster; Processing big data;**

I. INTRODUCTION

Hadoop is a popular open-source implementation of MapReduce that is used by academics, goverments, and industrial organizations. Hadoop can be used for storing large data and for processing data such as data mining, report generation, file analysis, web indexing, and bioinformatic research [2].

MySQL Cluster is a MySQL server with one or more data storages and management servers to configure the cluster and data replication. MySQL Cluster provides 99.999% availability to the data. MySQL Cluster is designed for distributed node architecture with no single point of failure. It consists of multiple nodes that are distributed across machines to make sure the system can work, even in case a node having a problem such as network failure [11].

Apache Hive and Apache Pig are open source programs for analyzing large data sets in a high-level language. Apache Pig is a simple query algebra that lets the user declare data transformation to files or groups of files. Hive is data warehouse software that facilitates queries and manages a large data set in distributed storage. Hive and Pig run on top of Hadoop [5]-[9].

When it comes to querying large data sets on MySQL Cluster, it can take seconds (assuming that the query is pretty complex). As the data grows larger, the time required to process the data increases too. This is where Hadoop fits in with Hive and Pig.

This paper presents the processing time of Hive, Pig, and MySQL Cluster on a simple data model with simple queries while the data is growing. Section 3 discusses a proposed method. Section 4 shows the results and explanations. And the last section, section 5 provides a conclusion and possible future work.

II. RELATED WORKS

Hive and Pig are a high-level language for processing data. Both are used for working with petabyte scale data [5][9]. Working at low-scale data can also be done with Hive or Pig. But processing low-scale data can consume more time with Hive or Pig rather than using other data processing software such as MySQL. As the data grows larger, MySQL requires more time to process the data until it reaches a point where Hive or Pig is faster than MySQL.

But when exactly do users need to change from MySQL to Hive or Pig for a faster processing time? This research indicates to users when they can switch to Hive or Pig as their rows of data become bigger. This test is done in a low-cost hardware environment.

III. PROPOSED METHOD

There are three aspects that will determine the result: 1) the data set file size (how many rows); 2) query statements; 3) query average time. There are three data sets with the same data model. The first data set is called ml100k (movie lens 100,000 rows) containing a total of 102,580 rows. The second data set is called ml1m containing a total of 1,075,611 rows. The last data set is called ml10m containing a total of 10,069,372 rows.

A. Hadoop Environment

For the Hadoop environment, there is one Hadoop master, three Hadoop slaves, one Sqoop, one Hive, and one Pig as shown in Fig 1. Sqoop only pulls the data from the MySQL Server and imports directly to Hadoop Distributed File System (HDFS). This is only for ease of use of importing data to HDFS. Data replication is set to 2 in the HDFS configuration.

B. MySQL Cluster Environment

MySQL Cluster has one management node, four data

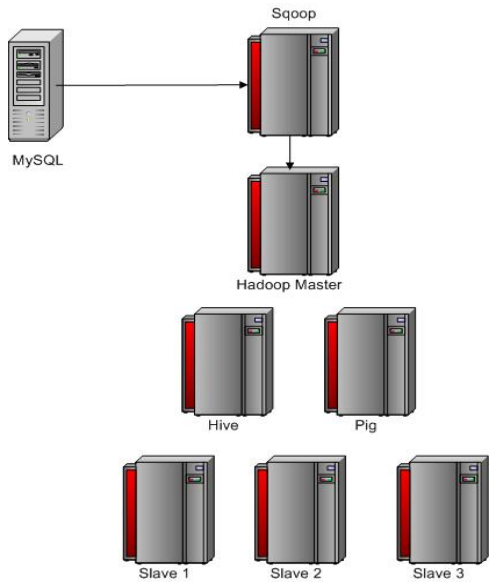

Fig. 1. Hadoop environment

nodes, and one MySQL server as the application node as shown in Fig 2. Each node is deployed on one machine. The number of replica is set to 2, which means it creates two node groups. The replica number is set to 2 because this is the minimum requirement to make MySQL Cluster to prevent a single point of failure.

C. Data model

The data tester uses a data set from the GroupLens Research Project at the University Of Minnesota [12]. The original data has been modified by the authors to make it easier to analyse and it has the same data model. Fig. 3 shows the data model. Note: Age description table contains a group of age. For example, ages between 18 and 25 are in id 1.

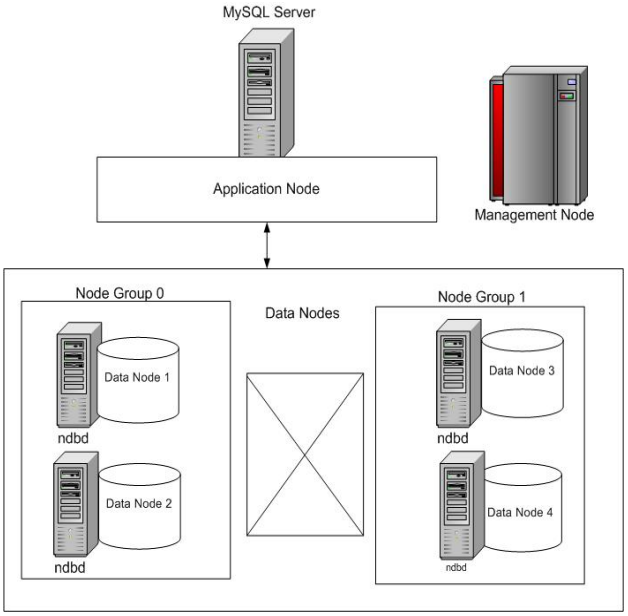

Fig. 3. MySQL Cluster environment

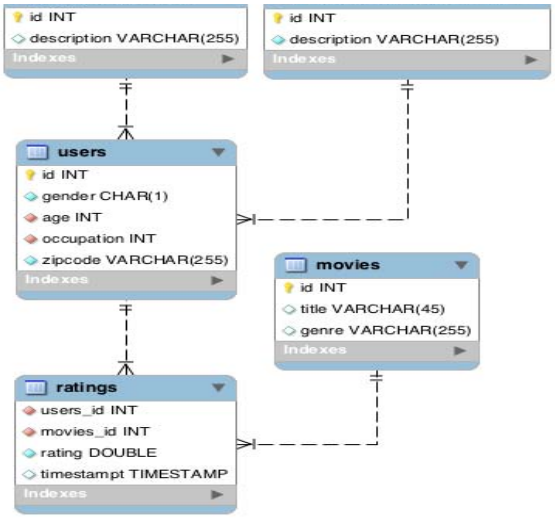

Fig. 2. Data model

D. Query Statements

There are seven queries executed to MySQL and Hive and Pig. Each query is run five times to get the average time of execution. The list of queries statements that are executed is shown in Table I.

IV. Experimental Results

The authors developed the system and performed the experiment in a virtual machine. Each system is run on top of CentOS version 6.5 with 1 processor core and 1GB RAM.

A. MySQL Cluster Result

The following result is a MySQL Cluster query on the ml100k, ml1m, and ml10m data set as shown in Table II and the mean result is shown in Fig 4.

TABLE I. Query Statements

Query	Description	Statements
Query 1	Report of each movie that has been rated and the score	Join query
Query 2	Movies that have been rated by people aged over 25	Join and sorting query
Query 3	Report of each movie that has been rated by the users and the average score on each movie	Join and aggregation function query
Query 4	Report of each movie that has been rated by the users and the average rated score on each movie and sorted by movie name	Join, aggregation function, and sorting query
Query 5	Sort movies that have been rated based on rating	Sorting query
Query 6	Report of each user how many movies have been rated and sorted the users by many of rating	Aggregation function and sorting query
Query 7	Counting the minimum, average, and maximum rating, and how many rating have been submitted on all movies	Aggregation function query

TABLE II. ml100k, ml1m, and ml10m query time taken on MySQL Cluster

ml100k	Attempt 1	Attempt 2	Attempt 3	Attempt 4	Attempt 5	Mean
Query1	2.26	2.09	2.08	2.15	2.05	2.126
Query2	3.43	3.55	4.09	3.56	3.61	3.648
Query3	2.16	2.11	1.94	1.97	1.93	2.022
Query4	2.03	2.06	2.02	2.09	2.04	2.048
Query5	0.76	0.67	0.73	0.7	0.69	0.71
Query6	2.29	2.03	1.61	2.2	1.79	1.984
Query7	0.66	0.58	0.56	0.58	0.57	0.59

ml1m	Attempt 1	Attempt 2	Attempt 3	Attempt 4	Attempt 5	Mean
Query1	11.62	9.94	9.81	10.76	10.67	10.56
Query2	15.43	15.98	16.1	15.45	15.6	15.712
Query3	10.35	9.33	9.89	9.07	9.27	9.582
Query4	10.44	9.88	9.06	9.46	10.02	9.772
Query5	7.73	6.95	6.98	7.1	7.19	7.19
Query6	20.41	18.08	19.24	18.57	18.05	18.87
Query7	5.83	5.78	5.77	5.5	5.54	5.684

ml10m	Attempt 1	Attempt 2	Attempt 3	Attempt 4	Attempt 5	Mean
Query1	77.75	75.59	87.7	88.63	83.1	82.554
Query2	92	90	89	90	103	92.8
Query3	71.64	71.59	75.21	70.93	71.57	72.188
Query4	76.71	79.28	76.9	77.44	76.77	77.42
Query5	85.8	92.12	86.86	84.33	84.94	86.81
Query6	182.94	182.73	182.88	182.81	182.73	182.818
Query7	57.56	55.64	56.56	54.33	56.54	56.126

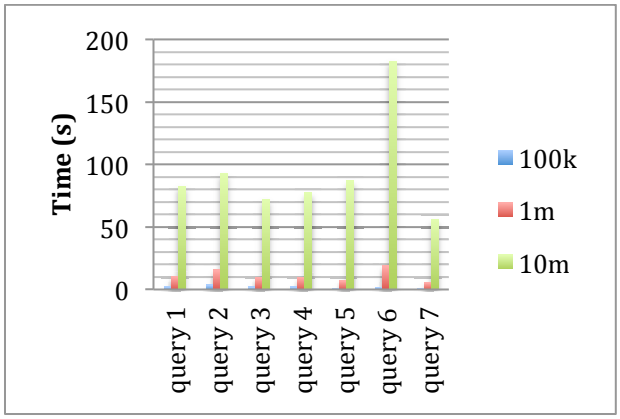

Fig. 4. Mean query MySQL Cluster from ml100k, ml1m, and ml10m

MySQL Cluster query time increases quite high as the data grows larger. One of the fundamental differences between a basic MySQL server and the MySQL Cluster is the storage engine architecture. MySQL Cluster is distributed and shared data that provides scalable performance for applications that often use primary key access and have high concurrency. However, it causes a cost on network access when accessing data between the MySQL server and tables distributed across data nodes. To execute the query, all data nodes must participate in the retrieval of data. This can cause a delay in gaining all the data from the data nodes.

B. Hive Result

The following result is a Hive query on the ml100k, ml1m, and ml10m data set as shown in Table III and the mean result is shown in Fig 5.

Hive data is stored in HDFS, which is also distributed to other nodes. Hive data are stored in a plain text file with CSV, as imported by Sqoop. Sqoop by default splits a table that has a primary key into four part files. Hive notices which id (primary key) is stored in which file. This gives an advantage such as indexing for Hive with a faster reading file because it does not have to read the entire file. Hive will not run a MapReduce job if a query does not have aggregation, join, or sorting function. It will invoke a MapReduce job if one of the functions is called. By submitting an aggregation, join, or sorting function, hive will immediately start a MapReduce job, which can take one until six seconds to start the MapReduce.

TABLE III. ml100k, ml1m, and ml10m query time taken on Hive

ml100k	Attempt 1	Attempt 2	Attempt 3	Attempt 4	Attempt 5	Mean
Query1	16	17	17	18	16	16
Query2	13	14	13	14	16	13
Query3	15	15	16	15	14	15
Query4	27	30	29	28	30	27
Query5	25	27	25	24	24	25
Query6	48	49	48	49	48	48
Query7	25	26	26	27	25	25

ml1m	Attempt 1	Attempt 2	Attempt 3	Attempt 4	Attempt 5	Mean
Query1	23	25	26	25	27	23
Query2	22	24	22	24	24	22
Query3	20	19	23	22	24	20
Query4	42	44	42	44	43	42
Query5	36	37	36	33	36	36
Query6	51	52	55	55	53	51
Query7	26	27	29	30	27	26

ml10m	Attempt 1	Attempt 2	Attempt 3	Attempt 4	Attempt 5	Mean
Query1	85	84	81	80	81	82.2
Query2	59	62	63	62	60	61.2
Query3	73	77	71	77	73	74.2
Query4	74	75	73	77	74	74.6
Query5	118	124	120	119	120	120.2
Query6	70	74	74	73	73	72.8
Query7	35	36	36	35	37	35.8

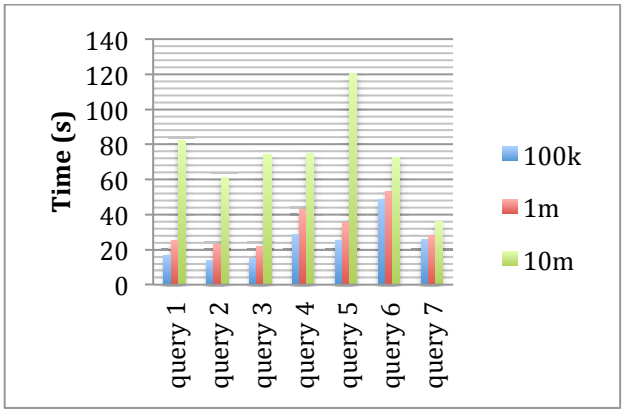

Fig. 5. Mean query Hive from ml100k, ml1m, and ml10m

C. *Pig Result*

The following result is a Pig query on the ml100k, ml1m, and ml10m data set as shown in Table IV and the mean result is shown in Fig 6.

Pig did not work well with this data set. Pig executes a step-by-step approach as defined by the programmer. But this approach does not work well in query that have minimum joins and filters. Pig executes each step one by one, which can consume more time in this data set. When data requires complex queries and many joining data in large set, Pig can handle it efficiently by executing each step and continually executing the next step.

In this data set, Hive performance overcomes MySQL Cluster processing time. Fig 7 shows the increasing time to process, as the data grows larger. MySQL Cluster at some point is faster than Hive. But on average, MySQL Cluster is slower than Hive when the data reaches 8.7 million rows with this data set. At some point, MySQL Cluster query is faster, such as sorting query, than Hive. But when the data grows larger again, MySQL will need more time to process it. As the data grows larger, Hive is suitable for this data model. But as the data model changes and more complex queries are needed with large amounts of data, Hive performance may slow down. Pig on the other hand is not quite suitable with this data set. Pig can take over when the queries are more complex and with bigger data rows and columns.

TABLE IV. Mean query Pig from ml100k, ml1m, and ml10m

ml100k	Attempt 1	Attempt 2	Attempt 3	Attempt 4	Attempt 5	Mean
Query1	25	26	24	25	25	25
Query2	25	27	27	24	27	26
Query3	27	30	29	31	29	29.2
Query4	30	30	29	29	30	29.6
Query5	33	35	33	34	36	34.2
Query6	95	97	99	94	95	96
Query7	30	29	29	30	30	29.6

ml1m	Attempt 1	Attempt 2	Attempt 3	Attempt 4	Attempt 5	Mean
Query1	32	34	33	33	32	32.8
Query2	33	35	36	33	34	34.2
Query3	39	38	39	41	38	39
Query4	41	40	41	42	42	41.2
Query5	56	57	53	55	54	55
Query6	102	105	101	102	102	102.4
Query7	43	43	44	43	44	43.4

ml10m	Attempt 1	Attempt 2	Attempt 3	Attempt 4	Attempt 5	Mean
Query1	100	101	110	105	103	103.8
Query2	77	80	85	77	83	80.4
Query3	100	105	104	104	110	104.6
Query4	119	125	123	124	123	122.8
Query5	94	97	97	96	94	95.6
Query6	315	320	319	322	321	319.4
Query7	120	123	123	121	122	121.8

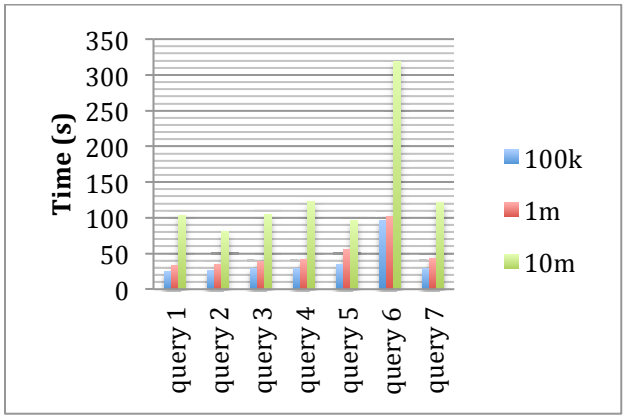

Fig. 6. Mean query Pig from ml100k, ml1m, and ml10m

V. Conclusions and Future Works

Hive is capable of handling large data and is faster than MySQL Cluster on a low-cost hardware machine. Pig is not suitable for this data set. Pig is more suitable for more complex queries and larger data sets. At some point MySQL Cluster queries are faster than Hive and Pig. But as the data grows larger, Hive can be more robust than MySQL Cluster.

One of the MySQL Cluster obstacles is, as the data grows larger, it requires more RAM to process the data. Indexed columns are always stored in memory. The larger the data size, the more memory or machine it will need.

Hive can overcome MySQL Cluster and Pig performance on low-cost hardware with basic query. However, this is tested on a low-cost hardware. Performance may change when better hardware is used for certain software, especially MySQL Cluster. With bigger RAM, MySQL Cluster performance may change to be better and can store more data. This can be the next future work, by comparing each software performance in a better hardware environment.

MySQL Cluster is not the only DBMS that can be clustered. Oracle database also has clustered version called Oracle Real Application Clusters (Oracle RAC). Oracle is designed for large data set with great performance. Comparing Oracle performance with Hive and Pig on large data set could be the next future work.

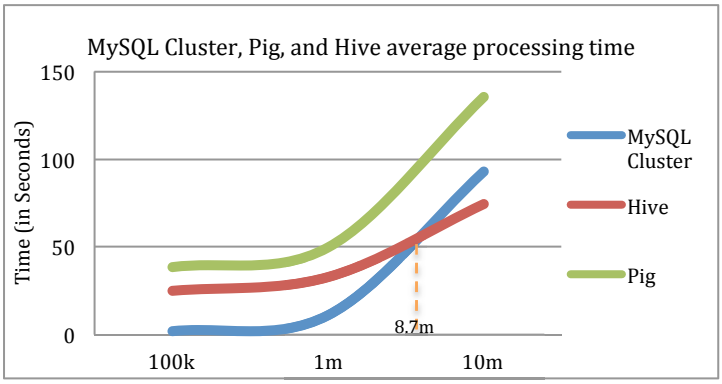

Fig. 7. Processing time query in general between MySQL Cluster, Hive, and Pig

ACKNOWLEDGMENT

The author would like to thank Swiss German University (SGU) for providing the opportunity to conduct and publish this research.

REFERENCES

[1] Kyong-Ha Lee, Yoon-Joon Lee, Hyunsik Choi, Yon Dohn Chung, Bongki Moon. *Parallel Data Processing with MapReduce: A Surver*. KAIST, Department of Computer Science. Korea University, Deparment of Computer Science and Engineering. University of Arizona, Deparment of Computer Science, 2011.

[2] Herodotos Herodotou, Shivnath Babu. *Profiling, What-if Analysis, and Cost-based Optimization of MapReduce Programs*. Duke university, 2011.

[3] Dhruba Borthakur. The Hadoop Distributed File System: Architecture and Design. The apache Software Foundation, 2007.

[4] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler. *The Hadoop Distributed File System.* Sunnyvale, California USA, 2010

[5] Alan F. Gates, Olga Natkovich, Shubham Chopra, Pradeep Kamath, Shravan M. Narayanamurthy, Christopher Olston, Benjamin Reed, Santhosh Srinivasan, and Utkarsh Srivastava. *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, 2009.

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

[7] Alan F. Gates, Jianyong Dai, and Thejas Nair. *Apache Pig's Optimizer*. IEEE Data Engineering Bulletin 36, no. 1, 2013.

[8] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. *Hive: a warehousing solution over a map-reduce framework*. Proceedings of the VLDB Endowment 2, no. 2 (2009): 1626-1629.

[9] Ashish Thusoo, Joydeep Sen Sarma, 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.

[10] Ronstrom, Mikael, and Lars Thalmann. *MySQL cluster architecture overview*. MySQL Technical White Paper, 2004.

[11] Oracle. *MySQL Cluster Evaluation Guide*. A MySQL Technical White Paper, 2013.

[12] http://grouplens.org/datasets/movielens/

This page is left blank on purpose