QUERY PERFORMANCE EVALUATION OVER HEALTH DATA

Ozgun Pinarer, Sultan Turhan

QUERY PERFORMANCE EVALUATION OVER HEALTH DATA

Sultan Turhan and Ozgun Pinarer
Galatasaray University Ciragan Cad. No:36 34349 Istanbul, Turkey

ABSTRACT

In recent years, there has been a significant increase in the number and variety of application scenarios studied under the e-health. Each application generates an immense data that is growing constantly. In this context, it becomes an important challenge to store and analyze the data efficiently and economically via conventional database management tools. The traditional relational database systems may sometimes not answer the requirements of the increased type, volume, velocity and dynamic structure of the new datasets. Effective healthcare data management and its transformation into information/knowledge are therefore challenging issues. So, organizations especially hospitals and medical centers that deal with immense data, either have to purchase new systems or re-tool what they already have. The new data models so-called NOSQL, its management tool Hadoop Distributed File Systems is replacing RDBMs especially in real-time healthcare data analytics processes. It becomes a real challenge to perform complex reporting in these applications as the size of the data grows exponentially. Along with that, there is customers demand complex analysis and reporting on those data. Compared to the traditional DBs, Hadoop Framework is designed to process a large volume of data. In this study, we examine the query performance of a traditional DBs and Big Data platforms on healthcare data. In this paper, we try to explore whether it is really necessary to invest on big data environment to run queries on the high volume data or this can also be done with the current relational database management systems and their supporting hardware infrastructure. We present our experience and a comprehensive performance evaluation of data management systems in the context of application performance.

KEYWORDS

Healthcare Data Analysis, e-Health, SQL, HiveQL, Hadoop, Relational Databases

1. INTRODUCTION

Aging world population and increasing hospital care costs have lead to allocation of large resources for e-health research studies. In today's world, healthcare data becomes the most valuable asset for the health organizations. To fulfill the customer requirements, researchers focus on analyzing the immense data: Big Data. Big Data has changed the way we manage, analyze and leverage data in any industry. One of the most promising areas where it can be applied to make a change is healthcare. Healthcare analytics have the potential to reduce costs of treatment, predict outbreaks of epidemics, avoid preventable diseases and improve the quality of life in general. Not only in health sector but, according to the new business world's dynamics, any organization may become one of the leading organizations in their sector, if they transform this infinite volume of data to fit their interests and transform it into added value for their management (Loebbecke, 2015).

As the volume of data is becoming increasingly large, it poses a challenge to executives in the analysis area. All the decision makers want to process and analyze this voluminous data in order to reveal the valuable knowledge for the healthcare data management. Capturing, storing, managing, retrieving and processing large scale data in an acceptable time frame (often real time) is one of the crucial challenges in health domain. Besides, there are several different ways of analyzing data; from querying on a traditional relational database system to machine learning algorithms. Of course each of them requires its own infrastructure investment of different sizes. As the complexity of analysis that the executives want to get increase, needs for specific data storage and processing infrastructure increase also (Wixom, 2014).

The digitization of the healthcare industry is happening fast. A major result of this transformation from paper to electronic records is the proliferation of healthcare data. It is really hard to choose the right database and business analytics systems to satisfy the organization's needs and working scenario. Various database management systems are in use today. Majority of the systems relies on the high-end hardware and/or special-purpose architecture to deliver the desired performance. Health organizations are faced to finance higher capital expenditure for the acquisition of infrastructure and higher licensing cost because of the competition in the market, even they don't really need them.

Proper database system choice has a great impact on user friendliness and scalability of data analytics applications (Chen, 2014). The traditional relational database model (RDBM) is the most common and proven approach to storing and querying data in various forms. However, the major disadvantage is the need to pre-design the exact field structures of the data, which is necessary in the normalization process of the database to ensure data consistency. A relational database storing this type of data will contain many empty fields, which will result in inefficient storage and poor query performance On the other hand, querying high level normalized data on real time basis is a highly costly operation. For this reason, database programmers denormalize the data before running the queries but this may result sometimes with data inconsistency. Besides, tackling the large scale data require a distinct approach that sometimes runs counter to traditional models of storage which provides good scalability and desired level of performance with insignificant or little cost (Zhao, 2017).

Limitations of the traditional relational database system gave birth to a new concept called "NOSQL database". There are a substantial number of projects as an alternative to traditional database system. Google's BigTable (Chang, 2008), Amazon's Dynamo (Giuseppe, 2007) Apache's Cassandra (Cassandra, 2014), Hypertable (Judd, 2008), Apache's CouchDB (Anderson, 2009), LinkedIn's Project Voldermort (Sumbaly, 2012), MongoDB (Kristina, 2010) are just to name a few. The Apache Hadoop-based project - HBase (Team, 2016) is one such approach. These applications are mainly served on big data analytics projects.

In this study, we try to explore whether it is really necessary to invest on a NOSQL database environment to run the queries on the high volume data or this can also be done with the traditional relational database management systems and their supporting hardware infrastructure? In order to answer this question, a bunch of queries are run on two well-known databases, one from NOSQL domain; Hadoop and one from the domain of relational databases SQL Server 2014 and their performances are evaluated. For NOSQL domain, the queries are written with HiveQL and for the relational database domain with SQL. The databases are populated with actual depersonalized prescriptions' record from a public hospital, written between 2015 – 2016.

Rest of the paper is organized as follows: Section 2 describes the methodology and the database platforms and working environment compared in this study. Section 3 describes the dataset, queries used for performance evaluation and presents the experiment results. Finally, Section 4 discusses the results and presents the future works.

2. METHODOLOGY

2.1 Experimental Setup

In order to evaluate the performance of the queries, two environments are set up. To simulate the traditional relational database environment, SQL Server 2014 is used. The hardware infrastructure on which SQL Server 2014 is running, consists of a virtual server equipped with a 2.6 GHz Intel Xenon E5-2690 v4 processor, 16.00 GB of RAM. The queries are written with SQL []. For NOSQL database environment, Apache Hadoop-based project and its database HBase are chosen. HBase is a distributed, fault-tolerant, highly scalable, NOSQL database, built on top of Hadoop Distributed File System (HDFS). The hardware infrastructure supporting the system with Hadoop architecture, includes 9 physical servers and a total of 1 TB of RAM. The queries are written with HiveQL (Lydia, 2015).

As all these NOSQL environment's products mentioned here, have recently entered the market and become prevalent, following subsections give brief technical details about them.

2.2 NOSQL DATABASES

2.2.1 Hadoop

Hadoop is an open-source distributed processing framework for large-scale distributed data storage and high-performance computation on a clustered system network of inexpensive pieces of commodity hardware (Khetrapal, 2006, Dorin, 2010, Taylor, 2010). It is at the center of a growing ecosystem of leading data technologies that are primarily used to support advanced analytics initiatives, including predictive analytics, data mining, and machine learning applications. It can handle various forms of structured and unstructured data, giving users more flexibility in collecting, processing, and analyzing data than relational databases and data warehouses. Hadoop framework developed in java programming language rely on two major components: Hadoop Distributed File System (HDFS) and Map/Reduce.

2.2.2 Hadoop Distributed File System (HDFS)

The Hadoop Distributed File System is the primary data storage system used by Hadoop applications. It uses a NameNode and DataNode architecture to implement a distributed file system that provides high-performance data access through highly scalable Hadoop clusters (Narayan, 2012). In HDFS, data is organized into files and directories. Files are divided into uniform sized blocks and distributed across cluster nodes and thereby removing the file size restriction. Also blocks are significantly larger than block sizes in standard file systems to minimize the cost of seeks and thereby enhancing the application-performance. HDFS adopts a master-slave architecture. NameNode has the master role and it maintains the file namespace including metadata, directory structure, files' list, blocks' list for each file, location for each block, attributes, authorization and authentication information. DataNodes play the slaves' role. They are responsible of creating, deleting or replicating the actual data blocks based on the instructions received from the NameNode and they report back periodically to NameNode (Narayan, 2012).

The Hadoop Distributed File System is specially designed to be highly fault-tolerant. The NameNode is a single point of failure for the HDFS cluster and a DataNode stores data in the Hadoop file management system. The file system replicates, or copies, each piece of data multiple times and distributes the copies to individual nodes, placing at least one copy on a different server rack than the others. As a result, the data on nodes that crash can be found elsewhere within a cluster. This ensures that processing can continue while data is recovered.

2.2.3 Map/Reduce

Map/Reduce a linearly scalable programming model to process large scale data stored in Hadoop File Distribution System (Pol, 2016). As it can be easily understood from the title, the model consists of two phase; The map phase corresponds to the map operation, whereas reduce phase corresponds to the fold operation. Data may be structured or unstructured. Map/Reduce performs two essential functions: it filters and distributes the work between the different nodes of the cluster or the map. The operation logic is based on "divide & conquer" principle and it partitions the large problem into smaller subproblems to the extent that the sub-problems are independent and they can be tackled in parallel by different slaves. To process the data, the programmer writes two functions; a function sometimes called mapper, and it organizes and reduces the results of each node in a coherent response to a request, called reducer. Each of these functions defines a mapping from one set of key-value pairs to another. The same functions can be used for a small as well as for a large-scale database without any modification since these functions are written in a fashion that is independent from data size as well from the cluster-size.

2.2.4 HBase

HBase is an Apache open-source project which presents a new data model similar to Google's big table. It is a column-oriented, distributed fault-tolerant and highly scalable database management systems, running on HDFS. To realize real-time read/write operations on a large scale database, HBase is a powerful tool [19]. The data in HBase is organized in labeled tables with rows and columns. Each row has absolutely one sorting key but the number of columns that it owns may differentiate.

Each cell in the table are versioned by a timestamp auto-assigned at the time of insertion. Its content is uniquely identified with a special set consisting of Table Name, Row-Key, Column Family, Column Name and the Timestamp. Each table uses the Row Key as primary key and they are easily accessible via this primary key. Thanks to this table structure, parallel scan in terms of Map/Reduce operations results into faster query response time and better overall throughput. Similar to HDFS and Map/Reduce, HBase also adopts master/slave architecture (Team, 2016).

2.2.5 Apache Hive – HiveQL

Apache Hive is a data warehouse solution for Hadoop environment (Huai, 2014). Hive provides data analysis operations, data summarization operations and querying while managing large datasets residing in distributed storage. It is one of the easiest to use of the high-level MapReduce (MR) frameworks. The Hive queries is written with a SQL-like language called HiveQL which runs over Hadoop Map/Reduce framework itself but hides complexity from the developer. HiveQL is composed of a subset of SQL features. It has also special extensions which are useful for batch processing systems. Hive supports analysis of large datasets stored in Hadoop's HDFS as well as easily compatible file systems and perfectly fits low level interface requirement of Hadoop.

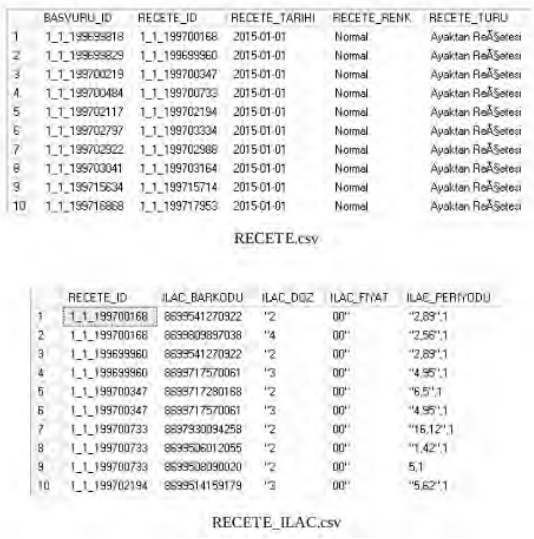
Apache Hive supports a SQL-like query language known as the Hive query language over one or multiple data files located either in a local file system or in HDFS. Hive query language runs over Hadoop map-reduce framework itself, but hides complexity from the developer, Hive query language (HiveQL) supports SQL features of data definition language and data manipulation language. It supports also all types of joint operations as well as aggregate functions and operations and provides always good results on primitive as well as complex data types. HiveQL adds a dialect of SQL and JDBC bindings for HBase.

HiveQL does have some limitations compared with traditional RDBMS SQL. In Hive, HBase automatically partitions tables horizontally into regions. Each region comprises a subset of a table's rows called partitions. Each table can have one or many partitions in Hive which allows insertion of data in single or multiple tables but does not allow deletion of updating of data. For read operations (SELECT) the complex filtering operations such as HAVING are not supported by HiveQL. SQL offers hundreds of built-in functions whereas HiveQL has dozens which obliges the programmer to have more efforts while querying the data. Besides, SQL supports the transactions and their management as well as indexes but HiveSQL does not support neither transactions nor the indexes. Finally, latency can be measured by subseconds in SQL but by minutes in HiveQL (Ahmed, 2017).

3. PERFORMANCE EVALUATION

3.1 Description of the Data

Actually this study is the preliminary work of the development of a web service designed to reveal the interactions between the drugs written in a single prescription on a real time basis. During the development of this web service, in order to measure the performance of the data on different environment, the queries of this study are written and executed. The data subject to this study contains the prescription information written for inpatients and outpatients in a public hospital between the years 2015 – 2016. For the patients who undergo an outpatient treatment, minimum 1, maximum 5 medicine can be described per prescription. However, for inpatients, dozens of medicine can be prescribed per prescription, or even hundreds of them can be prescribed per patient. The datasets consist of text-based structured data. For the purpose of this performance evaluation study, the datasets are stored in both the tables on SQL Server environment and the files on Hadoop environment. Prescriptions contain 35 million records (approximately 3 GB of text). Figure 1 shows the relational table format and a brief example of records containing in them.



	BASVURU_ID	RECETE_ID	RECETE_TARIHI	RECETE_RENK	RECETE_TURU
1	1_1_199699818	1_1_199700168	2015-01-01	Normal	Ayaktan ReÃ§etesi
2	1_1_199699829	1_1_199699960	2015-01-01	Normal	Ayaktan ReÃ§etesi
3	1_1_199700219	1_1_199700347	2015-01-01	Normal	Ayaktan ReÃ§etesi
4	1_1_199700484	1_1_199700733	2015-01-01	Normal	Ayaktan ReÃ§etesi
5	1_1_199702117	1_1_199702194	2015-01-01	Normal	Ayaktan ReÃ§etesi
6	1_1_199702797	1_1_199703334	2015-01-01	Normal	Ayaktan ReÃ§etesi
7	1_1_199702922	1_1_199702988	2015-01-01	Normal	Ayaktan ReÃ§etesi
8	1_1_199703041	1_1_199703164	2015-01-01	Normal	Ayaktan ReÃ§etesi
9	1_1_199715634	1_1_199715714	2015-01-01	Normal	Ayaktan ReÃ§etesi
10	1_1_199716868	1_1_199717953	2015-01-01	Normal	Ayaktan ReÃ§etesi

RECETE.csv

	RECETE_ID	ILAC_BARKODU	ILAC_DOZ	ILAC_FIYAT	ILAC_PERIYODU
1	1_1_199700168	8699541270922	"2	00"	"2,89",1
2	1_1_199700168	8699809897038	"4	00"	"2,56",1
3	1_1_199699960	8699541270922	"2	00"	"2,89",1
4	1_1_199699960	8699717570061	"3	00"	"4,95",1
5	1_1_199700347	8699717280168	"2	00"	"6,5",1
6	1_1_199700347	8699717570061	"3	00"	"4,95",1
7	1_1_199700733	8697930094258	"2	00"	"16,12",1
8	1_1_199700733	8699506012055	"2	00"	"1,42",1
9	1_1_199700733	8699508090020	"2	00"	5,1
10	1_1_199702194	8699514159179	"3	00"	"5,62",1

RECETE_ILAC.csv

Figure 1. Data format on RDBS. (PatentID, PrescriptionID, PrescriptionDate, PrescriptionColor,PrescriptionType) (PrescriptionID, DrugBarcode, DrugDosage, DrugPrice, DrugPeriodicity)

3.2 Hardware Features

Two database systems are developed respectively to implement the database approaches involved in the study. The traditional database system is built using a virtual server equipped with an Intel Xenon E5-2690 v4 2.6 GHz processor, 16.00 GB of RAM. The system with Hadoop architecture includes 9 physical servers and a total of 1 TB of RAM.

3.3 Queries

The query time of the two databases is evaluated by making three different queries with varied complexity. The queries are designed to perform read operations. The first query is responsible to execute a single SELECT operation on several tables with “inner join”. The second and third queries are also using aggregate functions and group by clause which put a serious burden on the query runtime. Figure 2 shows the queries executed in the study.

```
select re.BASVURU_ID, re.RECETE_ID, re.RECETE_RENK,
re.RECETE_TARIHI, re.RECETE_TURU, il.ILAC_BARKODU
il.ILAC_DOZ, il.ILAC_FIYAT, il.ILAC_PERIYODU
from drug.dbo.RECETE as re
inner join drug.dbo.RECETE_ILAC as il
on re.RECETE_ID=il.RECETE_ID
```

```
select il.ILAC_BARKODU, count(*) as num
from drug.dbo.RECETE as re
inner join drug.dbo.RECETE_ILAC as il
on re.RECETE_ID = il.RECETE_ID
group by il.ILAC_BARKODU
order by num desc
```

```
select RECETE_TARIHI, COUNT(*) as num
from  drug.dbo.RECETE
group by RECETE_TARIHI
order by num desc
```

Figure 2. Queries

3.4 Results

The variation of the query time with the size of the database is also studied. For each of the two database approaches, the time required to perform the queries with a variable complexity specified above is measured with databases containing respectively 1.8, 10, 20, 30 and 37 million records. At each parameter, the query is performed 10 times to calculate the average query time and the standard deviation. The query synchronization performance is given in Figure 3 respectively.

Query I - (seconds)				
	1,8 millions	10 millions	20 millions	30 millions
SQL	27	96	186	260
Hadoop	8	13	25	43

Query II - (seconds)					
	1,8 millions	10 millions	20 millions	30 millions	37 millions
SQL	13	21	31	37	43
Hadoop	12	15	19	22	23

Query III - (seconds)					
	1,8 millions	10 millions	20 millions	30 millions	37 millions
SQL	2	3	4	5	6
Hadoop	12	14	15	16	15

Figure 3. Query Synchronization Performance

The query time on the first query (shown in Figure 4) in the SQL database of more than 30 million records is quite disappointing. The reason is that the result of the query contains about 30 million records. The database with the Hadoop architecture has a very good performance on the same query and the same number of records. With Hadoop, parallel processing capability, query results can be returned faster than traditional databases.

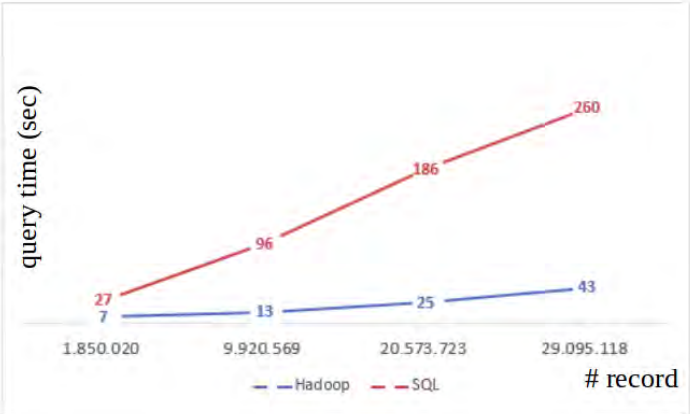


Figure 4. Query time according to row number for 1st Query

Although the amount of data in the second query was the same, the fact that the results contain about 15,000 records reduced the performance gap between the databases (see the Figure 5).

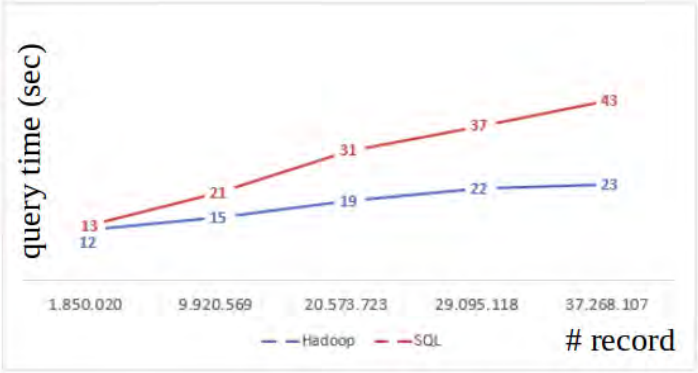


Figure 5. Query time according to row number for 2nd Query

In both queries, the Hadoop architecture has responded better to the increase in the amount of data. With the reduced amount of data, the Hadoop architecture performance on the third query is worse than the SQL database. The reason is that the result of the query contains about 800 records. The Map and Reduce operations of the Hadoop architecture have become useless with the reduction of the amount of data generated in the result of the query. So the performance of the Hadoop architecture in the third query is worse than the SQL database as shown in Figure 6.

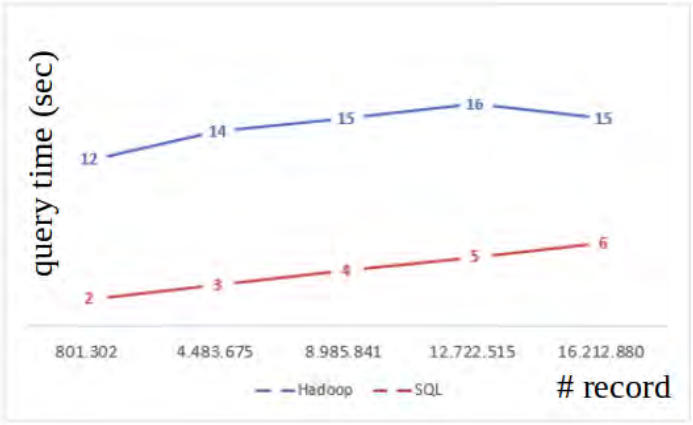


Figure 6. Query time according to row number for 3rd Query

4. CONCLUSION AND DISCUSSION

An examination of current approaches to prescription data storage indicates that the Hadoop approach is a viable alternative to relational database design because it provides better query performance.

As seen in the results of the first and second queries, Hive is designed specifically for large dataset analysis and works well for a range of complex queries. Hive is the most accessible way to quickly query and inspect datasets already stored in Hadoop. But in the third query, we see that the performance of Hadoop decreases when the amount of data decreases. Hive works ideally in large datasets, but MySQL works much better with smaller datasets and can be optimized in a variety of ways.

This study attempts to explore the vast opportunities of Hadoop and Sql query technologies in the management of prescription data. The prototype system developed is initially tested with a maximum of 30 million records only. Further evaluation using larger datasets, or even multiple databases and a data warehouse, will provide a more complete results on the performance of the Sql and Hadoop databases.

ACKNOWLEDGEMENT

This work is supported by Galatasaray University Research Foundation under the Grant No. 19.401.002 and Consortium of Galatasaray University.

REFERENCES

Ahmed, Nadeem, et al. 2017, "Data processing in Hive vs. SQL server: A comparative analysis in the query performance." 2017 IEEE 3rd International Conference on Engineering Technologies and Social Sciences (ICETSS).

Anderson, J. et al., 2009, “CouchDB: The Definitive Guide” , 1st edition, O'Reilly Media.

Cassandra, A., 2014. Apache cassandra. Website. Available online at http://planetcassandra. org/what-is-apache-cassandra, p.13.

Chang, F. et al., 2008. Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), 26(2), p.4.

Chen, C.P. and Zhang, C.Y., 2014. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Information sciences, 275, pp.314-347.

Dorin Carstoiu et al, 2010, "Hbase - non SQL Database, Performances Evaluation", International Journal of Advancements in Computing Technology, 2(5).

Giuseppe De Candia et al., 2007 “Dynamo: Amazon's Highly Available Key-Value Store”, in the Proceedings of the 21st ACM Symposium on Operating Systems Principles, Stevenson.

Huai, Y., et al., 2014, June. Major technical advancements in apache hive. In Proceedings of the 2014 ACM SIGMOD international conference on Management of data (pp. 1235-1246). ACM.

Judd, D., 2008. Scale out with HyperTable. Linux magazine, August 7th, 1.

Khetrapal, A. and Ganesh, V., 2006. HBase and Hypertable for large scale distributed storage systems. Dept. of Computer Science, Purdue University, 10(1376616.1376726).

Kristina Chodorow, Michael Dirolf, 2010, "MongoDB: The Definitive Guide", 1st edition, O'Reilly Media.

Loebbecke, C. and Picot, A., 2015. Reflections on societal and business model transformation arising from digitization and big data analytics: A research agenda. The Journal of Strategic Information Systems, 24(3), pp.149-157.

Lydia, E.L. and Swarup, M.B., 2015. Big data analysis using hadoop components like flume, mapreduce, pig and hive. International Journal of Science, Engineering and Computer Technology, 5(11), p.390.

Narayan, S. et al, 2012, Hadoop acceleration in an openflow-based cluster. 2012 SC Companion In High Performance Computing, Networking, Storage and Analysis (SCC),(pp. 535-538). IEEE.

Pol, U.R., 2016. Big data analysis: comparison of hadoop mapreduce, pig and hive. Int. J. Innov. Res. Sci. Eng. Technol, 5(6).

Sumbaly, R. etal., 2012, February. Serving large-scale batch computed data with project voldemort. In Proceedings of the 10th USENIX conference on File and Storage Technologies (pp. 18-18). USENIX Association.

Taylor, R.C., 2010, December. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. In BMC bioinformatics (Vol. 11, No. 12, p. S1). BioMed Central.

Team, A.H., 2016. Apache hbase reference guide. Apache, version, 2(0).

Wixom, B. et al., 2014. The current state of business intelligence in academia: The arrival of big data. CAIS, 34(1), pp.1-13.

Zhao, W. et al., 2017, November. Querying big data from a database perspective. In 2017 4th International Conference on Systems and Informatics (ICSAI) (pp. 1433-1437). IEEE.