ISSN:0975-9646

Big Data Performance and Comparison with Different DB Systems

Matteo D'Aloia[#1], Ruggero Russo[#2], Gianpaolo Cice[#3], Angela Montingelli[#4], Gaetano Frulli[*5], Egidio Frulli[*5], Francesca Mancini[*5], Maria Rizzi[+6], Annalisa Longo[+7]

[#]*Masvis srl*
Via Dalla Chiesa 27, 70014 Conversano (Bari), Italy

[*]*Fruman Rappresentanze srl,*
Via Demetrio Marin, 35 - 70125 Bari (BA) – Italy

+Dipartimento di Ingegneria Elettrica e dell'Informazione
Politecnico di Bari, Bari, Italy

***Abstract*— In this work we analyze the computational costs of write, read and delete operations for five database (DB) systems. Starting from a complete overview of database systems, two standard DB systems, such as MySQL and Postgres, and three Big Data systems such as Hbase, Cassandra and MongoDB have been analyzed. The computation time estimation has been performed for a single node implementation and for the computation of 50.000 and 1.000.000 records. The operations has been executed on random data. Authors found best performances for Cassandra DB system performing, for 1.000.000 records, an average computational time of: 1,46'' for reading, 5'58'' for writing, and 1'05'' for deleting.**

***Keywords*—Big Data, DB, DBMS, DB performances**

I. INTRODUCTION

In order to choose some representative database systems to analyze, it is important to introduce before some aspects of database technologies.

Apache Cassandra database [1] represents one of the possible choices for the need of scalability and high availability of data for a good performance of the entire system. In particular Cassandra model offers the possibility of an indexation of columnar type approach very useful in order to overcome fault-tolerance and generic interruptions. Apache Cassandra is a scalable, open source, NoSQL database useful for the management of large amounts of data structured, semi-structured and unstructured available on multiple data centers and cloud. Moreover, Cassandra provides continuous availability, a linear scalability and an operational simplicity through the use of many servers (important aspects for the system flexibility and for response times). Here are some aspects of the "column indices" which characterize a Casandra table and Cassandra DB. An index provides a means to access data in Cassandra using different attributes of the partitioning key. The benefit is therefore to accelerate efficiently research data corresponding to a determined condition of interrogation (query) by using the index column values. The Cassandra layout is characterized by the following points: the rows are organized in tables; the first component of a primary key of the table is the partitioning key; other columns can be indexed separately from the primary key (properties of relational database); tables can be created, deleted, and edited at runtime without blocking updating process and queries.

Another DB platform is Hbase [2]. HBase is an open source DB system modeled on BigTable. It was developed as part of the project Hadoop of the Apache Software Foundation and run on HDFS (Hadoop Distributed File System), providing capacity similar to those of BigTable for Hadoop. It is observed that BigTable introduces a data template quite innovative. Here are reported some characteristics [3]:

- The database is constituted only by a large table with a certain number of columns. There are no Join, there is no pattern;
- Each row is a set of columns, inside of which the data are homogeneous for type;
- The columns are stored on disk in separate files and are easy to compress being homogeneous data;
- Each row has a key that identifies itself, and a timestamp (encoding of the day, month, year, hour, minutes and seconds);
- The columns are always sorted in lexicographic order (ASCII order) for key to make searches easier;
- The DB is structured into tables or "tablet" (regions);
- For each tablet is assigned, to a "tablet server": (worker node of the distributed system adapted to stored data and able to extract data by using query).

Moreover, the data access model of BigTable is characterized by three levels:

1. a main root table that contains only metadata with pointers directed to the second level of the tables;
2. a second level of metadata table;
3. the user tablet.

MongoDB [4] instead is a system characterized by a scheme rather flexible: unlike SQL databases, where it is necessary to determine and declare a table of the schema before inserting data, MongoDB does not force the structure of the document. This flexibility facilitates the mapping of documents to an entity or to an object. MongoDB is therefore constituted by databases that contain indexed collections of documents, where each document is composed of fields.

A DB platform which implements an innovative features of Graph-Based Search is Neo4J [5]. Neo4j is a graph DBMS open source, produced by the software house Neo Technology. It is robust, scalable, high performance and is characterized by:

- ACID (set of properties that you would like to apply when modifying a database) transactions;
- High Availability;
- Can store billions of nodes and relations;
- High speed.

A suitable system for the web is the CouchDB [6] able to store data by means JSON script and to interrogate indices via HTTP. CouchDB works well with modern web systems and mobile apps. It offers the most advanced functionality of data replication and parallel research (Map/Reduce). CouchDB is a database document-oriented, where data are not stored in tables, but in real "documents" organized in a heterogeneous manner.

Riak [7] is a NoSQL database distributed open source. Riak is extremely operative for storing large of unstructured data accessed constantly by applications and users. It easily adds (or removes) nodes and server from a cluster and automatically scales the data within the cluster. Riak was designed to ensure the continuity of the reading/writing operations also for failure status of the nodes.

The HyperTable system [8] is part of technologies "scalable computing" of Google: data are taken from a web crawlers that stores information in a table row for each page that examines [9].

Another system is InfoBright [10] which is an open source designed to provide a scalable and optimized data warehouse. Its architecture provides the following benefits:

- ideal for data volumes up to 50TB, data compression (from 10:1 to 40:1);
- fast query response times;
- the performance remains constant by increasing the database size;
- does not require any specific scheme;
- does not require complex partitioning/indexing strategies;
- it is executable on low cost hardware systems;
- it is compatible with all the major Business Intelligence (B.I.) tool of as Pentaho, Jaspersoft, Cognos, Business Objects and others.
- Finally we remember Infinispan DBMS [11] which allows to operate with the following four major caching strategies:
- Local caching: the objects are inserted only in a local memory cache and are not available on the other nodes;
- Caching of replication: the objects inserted in the cache are replicated on all nodes of the cluster, thus all objects are available on all nodes. This mode is recommended for small clusters;
- Caching of distribution: objects are replicated only on a fixed subset of nodes (this subset is configurable);
- Caching of invalidation: there is no replication of objects.

All the performance of mentioned database systems are important to evaluate and to compare. For Big Data System we retain that MongoDB, Hbase, and Cassandra offers different flexible facilities for Big Data Application. For this reason we will compare in the work these DB systems with other no-Big Data systems. In particular we think that the standard MySQL and Postgres could be compared in order highlight the performance gap concerning the computational cost between Big Data and standard DB Systems.

We summarize below the topics that will be discussed in the paper:

- we describe some aspects about application and implementation of Big Data Systems;
- we implement a single node cluster for each of the following DB systems: MySQL, Postgres, Hbase, Cassandra and MongoDB;
- we estimate the computational cost for the same layout of a data table concerning read, write and delete operations.

II. Some Big Data Applications, Implementation Aspects and Performance Results

Big data represents a platform for collecting, organizing and analyzing massive data sets handled by Hadoop Distributed File system (HDFS). Hadoop represents the environment where the framework is applied for different applications such as family jobs scheduling by using genetics algorithms [12].

Apache system actually operates in Hadoop environments deal with big data improving PRAMR approach [13], food applications [14], for the management of the 2G/3G mobile data traffic [15], and for intrusiveness control [16].

Other interesting applications involve big data-based education and e-Learning industry [17], and Frequent Itemset Mining (FIM) technique able to extract knowledge from data [18].

The framework environment represents an import issue for data security involving replication procedures. As example Cassandra System in Hadoop environment is suitable for the replication of different node groups as sketched in Fig. 1 where only a replica group is interfaced with different server applications.

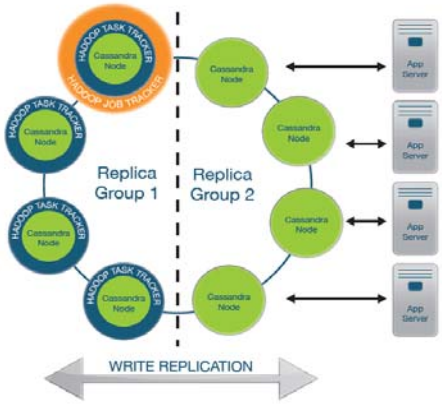

Fig. 1. Example of Cassandra node replication in Hadoop environment (single node replication).

The effective implementation of Big Data requires appropriate action procedures, and resources to enable analysis of the ever-growing data sets [19]. In this context, the implementation aspects are important key issues. Organizations use many various techniques and technologies to aggregate, manipulate, analyze, and visualize Big Data. The implementations are usually optimized for different goals such as data fusion, data integration, data mining, machine learning, predictive modeling, sentiment analysis, spatial analysis, simulation and time series analysis [20]. Generally, although Big Data solutions have a huge potential for both commercial organizations and governments, there is an uncertainty concerning the speed with which they can be utilized in a secure and useful way [21]. In order to fix some parameter and to analyze performance, we have configured a single node machine operating for deleting, reading and writing of records.

In particular, in our approach we have used for the performance comparison the following technologies/tools:

- Operative System: Ubuntu 14.04 LTS3 version, 64 bit
- Minimum RAM/CPU: 4 core , 8 Gbyte di RAM (INTEL XEON E 5);
- Single node in Hadoop Platform/environment;
- Datastax platform DSE: performance control;
- Java tool: Yahoo Cloud Serving Benchmark (YCSB) for computational cost testing.

The performance results will be referred to above listed specifications.

The comparison of the computational cost between the five DB Systems (Cassandra, Hbase, MongoDB, Postgres, MySQL) has been performed by considering a single node for a single node implementation and for the computation of 50.000 and 1.000.000 rows made by five column. During the first step has been compared the computational time for read, write and delete operations concerning the processing of 1000000 of rows and 50000 rows. An example of data records rows are reported in appendix where each row was written by random data (integer data type). In Table 1 we show the comparison of computational time for read, write and delete operations concerning 1.000.000 and 50.000 rows , respectively (the table above shows the maximum values, the table below indicates the average measured values). By analyzing this table is evident how Casandra DB exhibits the best performance if compared with the other Big Data systems (HBase and MongoDB). By analyzing also the computational costs of the "standard DB" systems (MySQL and Postgres), it is clear how the advantages of the Big Data are mainly for reading and writing of records. In particular the best performance is detected for writing process.

	1000000 rows			50000 rows		
	read	write	delete	read	write	delete
mysql	2,38 sec	11 min 29 sec	1,36 sec	0.12 sec	6 min 48 sec	0,21 sec
postgres	2,12 sec	11 min 13 sec	1,27 sec	0,11 sec	6 min 37 sec	0,19 sec
mongoDB	1,56 sec	6 min 57 sec	1,15 sec	0,09 sec	3 min 45 sec	0,16 sec
hbase	1,52 sec	6 min 46 sec	1,11 sec	0,086 sec	3 min 47 sec	0,17 sec
cassandra	1,49 sec	6 min 26 sec	1,08 sec	0,076 sec	3 min 36 sec	0,15 sec

	1000000 rows			50000 rows		
	read	write	delete	read	write	delete
mysql	2,11 sec	11 min 12 sec	1,31 sec	0,089 sec	6 min 36 sec	0,20 sec
postgres	2,07 sec	10 min 46 sec	1,24 sec	0,086 sec	6 min 18 sec	0,18 sec
mongoDB	1,53 sec	6 min 20 sec	1,09 sec	0,08 sec	3 min 28 sec	0,15 sec
hbase	1,49 sec	6 min 11 sec	1,07 sec	0,078 sec	3 min 32 sec	0,16 sec
cassandra	1,46 sec	5 min 58 sec	1,05 sec	0,07 sec	3 min 26 sec	0,14 sec

Table 1. Comparison of computational time for *read*, *write* and *delete* operations (1.000.000 of rows and 50.000 rows).

The same information, for each type of analysis, are reported graphically from figures 2,3,4,5,6,7,8, and 9. The deleting time is similar to each case but could change drastically in the case of multi-node implementations. The best performances are found for Cassandra DB system performing for 1000000 of records an average computational time of:1,46'' for reading, 5'58'' for writing, and 1'05'' for deleting.

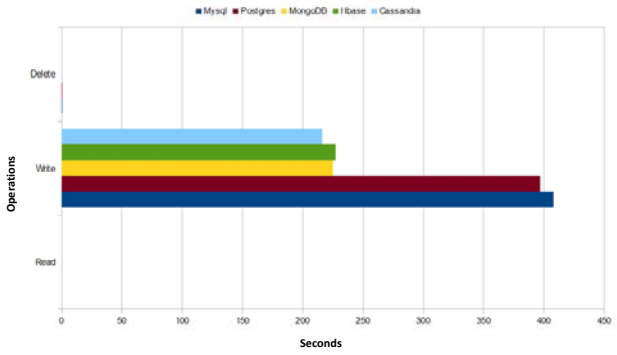

Fig. 2. Comparison of computational time for *write* operation (maximum value for 50.000 rows).

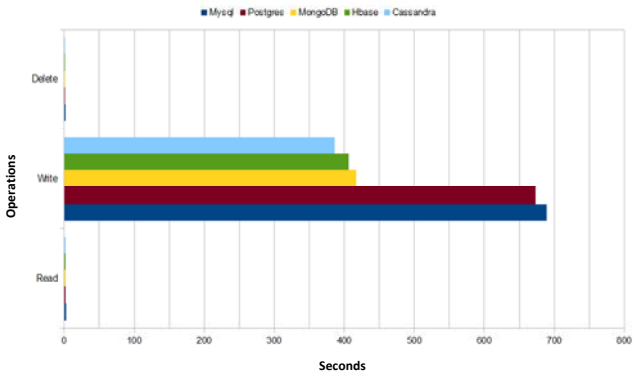

Fig. 3. Comparison of computational time for *write* operation (maximum value for 1.000.000 rows).

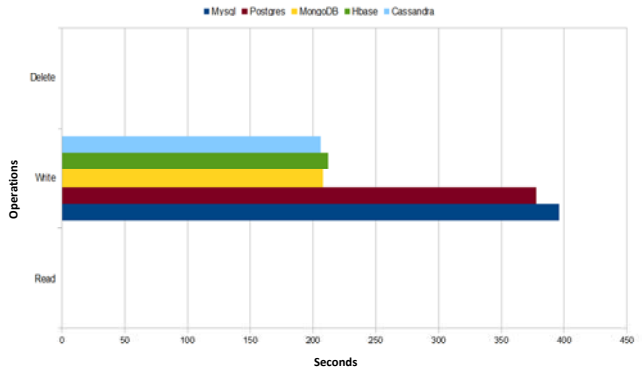

Fig. 4. Comparison of computational time for *write* operation (average value for 50.000 rows).

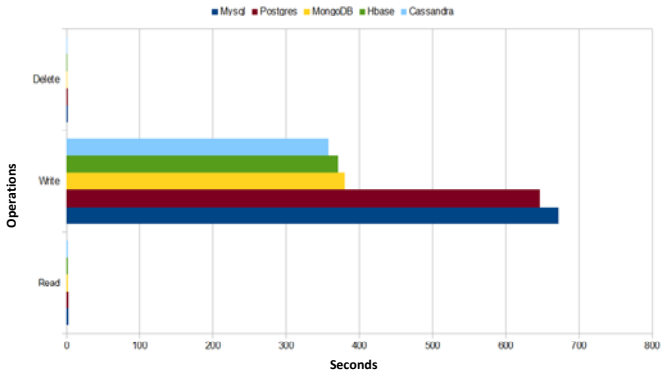

Fig. 5. Comparison of computational time for *write* operation (average value for 1.000.000 rows).

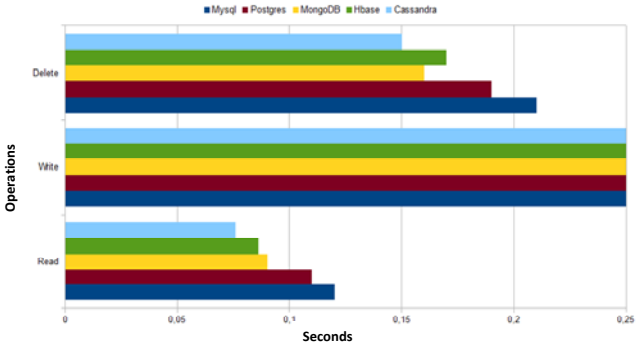

Fig. 6. Comparison of computational time for *read*, *write* and *delete* operations (maximum value for 50.000 rows).

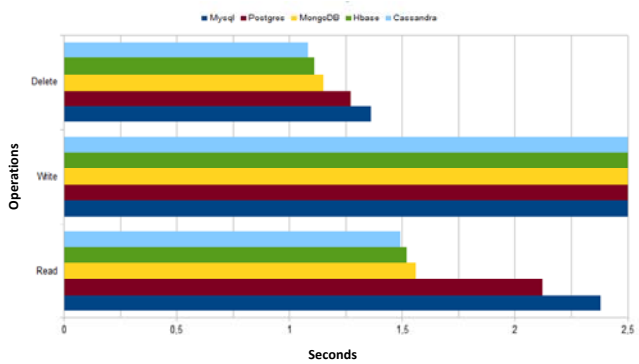

Fig. 7. Comparison of computational time for *read*, *write* and *delete* operations (maximum value for 1.000.000 rows).

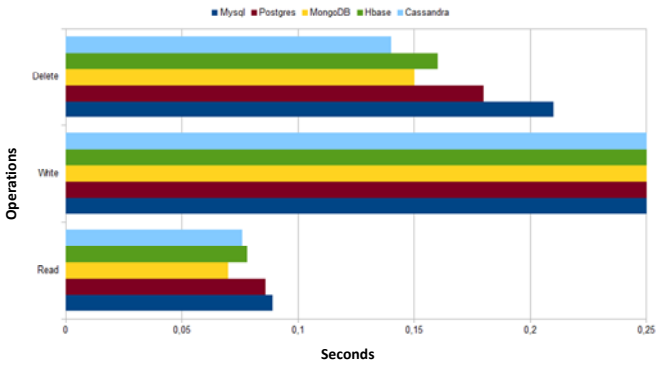

Fig. 8. Comparison of computational time for *read*, *write* and *delete* operations (average value for 50.000 rows).

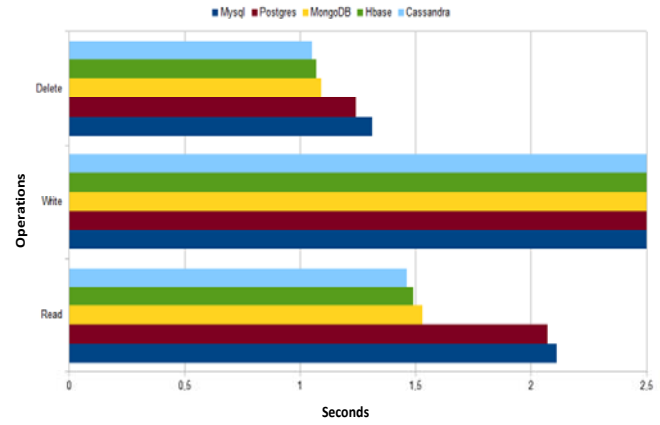

Fig. 9. Comparison of computational time for *read*, *write* and *delete* operations (average value for 1.000.000 rows).

III. Conclusion

In this paper the authors compare computational costs of different database systems. A good performance for single node write, delete and read operation is found for Cassandra Big Data System. This suggest the use of Cassandra DB for analytics and business intelligence operations requiring massive data. This study is useful for the choice of the technology to use for a single node cluster implementation, and could help the reader to find information about the order of magnitude of the computational cost. The computational cost could change drastically by using platforms different from the tools used for the experimentation of this work.

Acknowledgment

The work has been developed within the framework of Research Project titled: "Sistema di business analytics per la valorizzazione di indicatori di opportunità commerciali relative a servizi di rappresentanza basato su metodologie di business intelligence e big data analysis"- (System of business analytics, based on methodologies of business intelligence and big data analysis, for the valorisation of indicators of commercial opportunities relating to representation services).

REFERENCES

[1] http://cassandra.apache.org/
[2] https://hbase.apache.org/
[3] N. Dimiduk e A. Khurana, HBase in Action, ISBN 9781617290527, Ch1 ., November 2012
[4] https://www.mongodb.org/
[5] http://neo4j.com/
[6] http://couchdb.apache.org/
[7] http://docs.basho.com/riak/latest/
[8] http://hypertable.org/
[9] http://www.hypertable.com/collateral/whitepaper-hypertable-architecture.pdf
[10] https://www.infobright.com/
[11] http://infinispan.org/
[12] Umadevi T, Dyvia Zion G, Supryia U. Scheduling the family jobs using Genetic algorithm for Big Data Clouds. (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 7 (4), 2016, 1857-1867.
[13] Somasekhar G, Karthikeyan K. The pre big data matching redundancy avoidance algorithm with mapreduce. Indian Journal of Science and Technology. 2015 Dec; 8(33). DOI: 10.17485/ijst/2015/v8i33/77477.
[14] Lakshmi M, Sowmya K. Sensitivity analysis for safe grain storage using big data. Indian Journal of Science and Technology. 2015 Apr; 8(S7). DOI: 10.17485/ijst/2015/v8iS7/71225.
[15] Liu J Liu F, Ansari N. Monitoring and analyzing big traffic data of a largescale cellular network with Hadoop. IEEE Network. 2014; 28(4):32–9.
[16] Mamlouk L, Segard O. Big data and intrusiveness: Marketing issues. Indian Journal of Science and Technology. 2015 Feb; 8(S4). DOI: 10.17485/ijst/2015/v8iS4/71219.
[17] Noh K-S. Plan for vitalisation of application of big data for e-learning in South Korea. Indian Journal of Science and Technology. 2015 Mar; 8(S5).DOI:10. 17485/ijst/2015 v8iS5/62034.
[18] Moens S, Aksehirli E, Goethals B. Frequent itemset mining for big data. IEEE International Conference on Big Data, 2013, Oct 6–9. p. 111–8.
[19] Wielki J. Implementation of the Big Data concept in organizations – possibilities, impediments and challenges, Proceedings of the 2013 Federated Conference on Computer Science and Information Systems, pp. 985–989.
[20] McKinsey Global Institute. Big data: The next frontier for innovation, competition, and productivity, McKinsey report 2011.
[21] National Intelligence Council, "Global Trends 2030".