Experimental comparative study of NoSQL databases: HBASE versus MongoDB by YCSB

Houcine Matallah
Abou Bakr Belkaid University of Tlemcen

EXPERIMENTAL COMPARATIVE STUDY OF NoSQL DATABASES: HBASE VERSUS MONGODB BY YCSB

Houcine MATALLAH[1], Ghalem BELALEM[2], Karim BOUAMRANE[3]

[1] Computer Science Department, University of Abou Bekr Belkaid, Tlemcen, Algeria

[2], [3] Computer Science Department, University of Oran1, Ahmed BENBELLA, Oran, Algeria.

[1] matallahh@yahoo.fr [2] ghalem1dz@gmail.com [3] kbouamrane@gmail.com

Abstract- *Big Data is a set of technologies based on NoSQL databases allowing scalability of volumes, numbers and types of data. The important companies in the IT sector find these NoSQL systems, new solutions to respond to scalability needs. Multiple open-source and proprietary models of NoSQL are available on the market. Because of the large number and diversity of existing solutions, it is difficult to select an appropriate solution for a specific problem. In this paper, we develop a comparative study about the performance of two solutions widely employed HBase and MongoDB, using the YCSB tool. The latter being a very powerful test tool, used in multiple evaluation works of NoSQL databases. The purpose is to provide assistance and support to actors interested of Big Data and Cloud Computing for eventual decisions for the choice of solutions to be adopted.*

Keywords : BigData, Cloud Computing, NoSQL, HBase, MongoDB, YCSB.

1. INTRODUCTION

The explosion of data and the change of scale in volume, numbers and types, in the recent years, has imposed to researchers new challenges and motivated to develop new technologies to contain and process these huge volumes of data. Many concepts “inseparable” currently dominate IT market. We often hear of “Cloud Computing” hosting a “Big Data” as “NoSQL” and processed by a simple program “MapReduce” in “Clusters” distributed all over the world.

Currently, relational databases are facing many new challenges, especially in large scale and high-concurrency applications. In a Cloud architecture composed of many thousands of nodes, the traditional relational systems become unusable, hence the need to create new systems natively adapted to a distributed environment like in the cloud where they can dynamically manage and distribute data in multiple nodes (Scale-out). The main constraint of relational database systems appears with the exponential expansion of data that is currently measured in petabytes at a high speed: with these huge volumes of data, the relational ACID constraints can become a blocking factor that affects performance, although they ensure data consistency. A typical example is that of Internet searches where the consumer of information is more interested to having an immediate response without requiring if this information is instantly updated. This new computing era with its new requirements has forced the different researchers in the scope to search the best solutions to adapt their systems to these imposed changes that led to the new Big Data concepts which have contributed to the emergence of the NewSQL and especially the NoSQL movement. Several solutions NoSQL have been implemented by the big companies in the sector such as Google, Yahoo, Facebook, Twitter, Amazon, etc., to host in their servers these large data volumes. These models based on different architectures are designed, developed and deployed in different sectors. The lack of standardization is an important aspect of the NoSQL movement and the panoply of existing solutions in the computing market, impose on decisions makers many problems in choosing the appropriate model in relation to their operating environment.

In this context, our paper tries to provide some answers to choose the NoSQL system appropriate to the type of data used and the type of processing executed on these data.

Our study focuses on two of these data management solutions characterized by the implementation in their kernels of the same algorithm “MapReduce” which are the MongoDB and HBase models. To evaluate and compare the NoSQL solutions available, a number of benchmarks have been designed, the most commonly used is the YCSB. Several comparative studies in the same context, using YCSB, have recently been developed. The study to which our results will be compared was conducted in a single machine is presented in [1]. Another comparative study has been carried between several NoSQL databases in a distributed environment (1, 2, 4, 8, 16 and 32 nodes) in [2]. Also, another comparison NoSQL system was developed on 12 servers in [3]. Evaluating performance degradation in NoSQL databases generated by virtualization is proposed in [4]. There are even other works that compared the performance of SQL systems with NoSQL [5, 6].

This article will be organized as follows: In the background section, we will begin by presenting the new requirements imposed by the scalability and then expose the limitations of traditional systems. In section 3, we will emphasize on the most presently used software solutions for data management in large scale environments, namely NoSQL systems, since they will be the topic of our comparative study. The two NoSQL systems HBase, MongoDB and YCSB benchmarking tool that will be employed in our study will be presented in Section 4. After the evaluation of performance of each database, various experimental results of this comparative study will be synthesized and compared to the results obtained by Veronika Abramova and al. [1] in section 5. We conclude our article with a summary and some perspectives for our future works.

2. BACKGROUND

The latest studies show that the growth of the volume of data stored by computer is about 30% per year (Giga, Tera, Peta, Exa, Zetta, ...) [7, 8], generated by the multiple sources of information. Thus we exceeded the Databases scale to the Big Data scale. Managing scalable, distributed data that has been the vision of the research community in the domain of databases for more than three decades, has rapidly accelerated in recent years to overcome the harsh constraints of servers, networks and applications operating in distributed environments imposed by new IT trends.

The order of magnitude of the exabyte is already causing several problems for data management. Any traditional data management system can support such number with acceptable response times. So this change of scale volumetric imposed new reflections, new needs and new challenges that have led to search for a new performance level.

The change of scale in volume, the geographic distribution of data is already current problems, which adds another difficulty related to the diversity of the types of these masses of data created and accumulated, without organization or structure. The large and increasing proportion of unstructured data from all digital data is one factor that led to the new concept Big Data; the latest studies show that the percentage of unstructured data from all data is 90% [9]. Thus extending management and processing to semi-structured or unstructured data, is considered as fundamental problematic for both Big Data and for the clouds. These new requirements were the founding objective of the “Storage-as-a-Service” offering storage as a service [10, 11, 12]. It is considered as a new model of storage and data access to replace Relational Data Base Management Systems which were considered almost perfect until these recent years. “Storage-as-a-Service” is a business model in which a provider rents space in their storage infrastructure to a customer. The principal constraint of this service is data security.

The various data management models designed in recent years, are characterized and compared using different factors, the most important are:

1) Scalability: supports capacity increases of any kind, volumetry, number of customers, etc., with more or less constant response times. It may be vertical (Scale-up) which is to keep the same number of servers, by increasing their hardware performances (Make a single CPU as fast as possible, increase clock speed, add RAM, make disk I/O go faster) like in a centralized architecture, as it may be horizontal (Scale-out) by adding additional servers in the network (Make Many CPUs work together, learn how to divide your problems into independent threads) [13]. To support huge volumes of data and very high request rates, cloud computing are architected to scale-out, so that large scale is ensured using large numbers of commodity servers, each hosting copies of the database.

2) Performance: is characterized by the latency which is the time required to locate the first bit or character in a storage location and the throughput which is the amount of bytes transferred.
3) Availability: continuous operating capacity without failure.

Relational Data Base Management Systems (RDBMS) are well suited to the transactional needs through the ACID properties (Atomicity, Consistency, Isolation and Durability). Similarly, the extensions of this model as the cubes allow taking over the needs analysis in large consolidated databases (data warehouse). The ACID properties constrain and limit the deployment of relational systems in the cloud since the main advantage of the Cloud is the scaling of the services used [14].
To remedy this, some relational database providers have expanded the capabilities of their systems by integrating new functionalities to support the management of their databases in a multi-nodes environment. We cite the examples of Oracle RAC (Real Application Clusters) [15], MySQL Cluster [16] and IBM DB2 [14]. If we take the simplest case of a two-node cluster for example, Oracle RAC (included by default in the Standard version) can ensure efficiently the load balancing and failover in case of failure. Oracle RAC management becomes more complex by adding more nodes to the cluster [15]. Another constraint related to the type of data is the fact that relational systems are designed to manage structured data, however, the flexibility of NoSQL data models makes it possible to manage, store and process unstructured data (video, images, sound, tweet, etc.) and semi-structured (XML and JSON).

3. NoSQL

NoSQL databases or "Not Only SQL" are types of database that have begun to emerge since 2009, to meet the new needs of the Cloud and Big Data. The name may seem like an opposed to SQL databases, their primary function is not to replace them but to propose alternatives to relational databases to respond the current trends and new architectures, especially cloud computing [17].

3.1 Basic Concepts

NoSQL refers to a class of DBMS which is not based on the classical architecture of relational databases. The logic unit is not the table, and data are not generally manipulated with SQL. Initially they were used to manage the giant databases for websites with very large audience such as Google, Amazon, eBay or Facebook. Unlike the RDBMS and the data warehouses, these new data storage models have been created to host and operate large amounts of data, they are distributed on many servers and designed to take over thousands or millions users who make updates and readings. They are based on parallel file systems to improve performance and overcome resource constraints by multiplying the hardware devices to enable parallelization of storage and access to information.

However, it is not easy to explicitly define what a NoSQL database because until now, no standard has been established yet. Thus, we can define the NoSQL as follows [13]:

“NoSQL is a set of concepts which allows a fast and efficient data processing with emphasis on performance, reliability and flexibility”. It can be summarized to a simple array associative unidimensional of millions - or even billions - of entries. This design permits scalability to petabytes of data and thousands of concurrent queries [11, 12].

3.2 Characteristics or features

A NoSQL database is mainly characterized by [13, 18]:

1) Supporting mass storage - More than rows in tables: A table can contain millions of rows;
2) No joins: The data is usually embedded in the same "container". This form of storage can be seen as already executed joins (reading data quickly);
3) Schema-Less: The structure and type of data can change at any time without necessarily impacting the application (reading and writing data quickly);
4) Works on multiple processors: Decompose the problem into multiples threads with many CPUs working together;
5) Relies on “shared-nothing architecture” between nodes: Every node in the cluster has its own CPU, RAM and disk;
6) Supports the horizontal scalability (Scale-out) and easy to expand: NoSQL systems are designed to increasing workload by distributing data automatically across multiple nodes and form a cluster to maintain a linear performance at high scalability;
7) Cost model: Lower costs of managing, operating and storing large volumes of data.

We must also note that NoSQL is not:

1) An opposite to the SQL language;
2) Always open source, there are commercial products NoSQL;
3) Always related to Big Data and Cloud Computing, it can be used in other contexts: Many relational systems (not Big Data) migrated or looking to migrate to NoSQL.

In contrast, NoSQL have some weaknesses, such as nonexistence of normalized query languages like SQL, the lack of standardized interfaces, consistency and security of data and other additional features, not mature enough for most NoSQL solutions created in recent years.

3.3 From SQL to NoSQL

To resolve the performance issues related to Big Data, corporate developers have conducted compromises on RDBMS ACID properties. These compromises on relational concept enabled these new systems to break free from their constraints for scalability. NoSQL renounces the classical RDBMS functionalities in favor of simplicity. The performance remains good with scalability by simply multiplying the number of servers, reasonable solution with lower costs. The giant systems are the first affected: comfortable budget, huge amount of data, less importance to the relational structure relative to the speed of access, even to multiply servers.

The relational model is based on mathematical sets. Each set is represented by a table, and its attributes are represented by columns. One of the fundamental principles is the notion of relationship between tables using cardinalities, primary keys and foreign keys; this implies first careful conceptual study of the database schema. Once the model adopted and implemented, it is difficult to change its structure, which causes problems at its reengineering.

The NoSQL movement is more pragmatic, based on data storage needs and a stronger bond with the different customer's languages. Most NoSQL database engines do not use predefined schemas in advance, hence the term "Schema-less", so you can have in the same table records with different fields in type or number, which does not require the database engine to perform checks and impose schema constraints; the data being organized in the client code. However, this principle is not completely verified in practice, since the maintenance of a homogeneous data structure is important for indexation and search criteria.

The notion of "Schema-less" in NoSQL also implies that the management of Null values do not exist, which avoids the additional management of Null values in relational databases.

The other important difference is that in relational databases, a transaction must respect the ACID properties for it to be validated. The respect of these properties in a distributed environment is expensive because the risk of decreased performance is proportional to the number of servers (nodes). NoSQL engines ignore these properties, their priorities being to increase performance by adding nodes. The updates are propagated eventually but the guarantees of consistency of readings are limited. Some authors propose the acronym "BASE: Basically Available, Soft state, Eventually consistency" unlike "ACID" [19]. On the other hand, the idea to combine NoSQL databases and relational databases was also mentioned in a hybrid solution [20]. Other systems have emerged that try to support the processing of SQL queries to data stored in HDFS of Hadoop (SQL-on-Hadoop Systems) [21, 22]. These solutions cannot be considered as databases since they are mainly designed to improve the Hive query language. Another category of data stores aims to provide the relational model the benefits of horizontal scalability and fault tolerance provided by NoSQL solutions, which is the NewSQL. These solutions as well as NoSQL present themselves as alternatives capable of managing a huge volume of data. NewSQL emerged from three types of architectures: relational, NoSQL and data grid. All these NewSQL data stores support the relational model and use SQL as their query language, even though they are based on different assumptions and architectures than RDBMSs, as Google Spanner, VoltDB, Clustrix and NuoDB [23, 24, 25].

3.4 Types of NoSQL databases

There are four main categories of NoSQL databases which differ by the manner of data representation [19, 26].

3.4.1 Key-Value Store

The NoSQL database key-value is considered the most elementary of the other models. Its principle is very simple; each stored value is associated with a unique key. It is only through this key it will be possible to run queries on the value. The structure of the stored object is unrestricted and under the application developer's responsibility.

This key-value type can be comparable to a dictionary where each word (key) has one or more definitions (value). The dictionary is ordered (indexed) so it is not necessary to browse the whole dictionary to find a word. Similarly, the NoSQL key-value is indexed by keys that point directly to their corresponding values [24]. A considerable advantage of this type is that it is easily extensible and therefore facilitates the scale-out. Indeed in the case where the volume of data increases, the load can be easily balanced between nodes by simply redefining the key intervals between each node. The most known solutions having adopted the key-value pair system are [19, 24, 27]: Redis, Memcached, Amazon DynamoDB, Riak, Voldemort Ehcache, Hazelcast (used LinkedIn site), OrientDB, Berkeley DB, Oracle NoSQL, etc.

3.4.2 Wide Column Store

NoSQL databases column-oriented are well adapted to storing and managing large data volumes. They were designed by the web giants for treatment of large volumes of data that grow quickly.

The principle of the column family is in storage by column not by row. Unlike a classical database line-oriented where the data is stored so as to favor the lines containing all the columns in the same row together, databases column-oriented, for their part, will store the data all rows of the same column together. These databases can evolve over time, either in number of lines or number of columns. Contrary to a relational database where the columns are static and present for each row, the columns in this category are dynamic and therefore present only if necessary. Moreover it has no storage of Null values [19, 24, 27]. In some Wide Column Stores such as HBase or Cassandra, there are some additional concepts such as logical grouping of lines (equivalent to a table in a relational model), and the concept of Super columns (new column containing other columns).

NoSQL databases column-oriented include considerable advantages: Their storage capacity is increased through the space saved on null values. They are the closest NoSQL models to the relational model because of their structure. They also provide Interesting speed data access to achieve the rows of a table since unnecessary information from other columns will not be browsed.

However, the use of such databases is only effective in a big data context for large data volumes having the same type. The main NoSQL solutions of column family are [19, 24, 27]: Cassandra (Apache), HBase (Apache), Bigtable (Google), Accumulo (Apache), Hypertable, etc.

3.4.3 Document store

NoSQL databases document-oriented are an evolution of databases key-value. Here the keys are no longer associated with values as a binary block, but with a document whose format is not imposed [18, 26]. It may be of different types such as JSON, BSON or XML. Although the documents are structured, this database is called "Schema-less", therefore, it is not necessary to define the fields of a document. The advantage of the document store is to retrieve a set of hierarchically structured information from a key. A similar operation in a relational system equates to multiple joins tables.
The main NoSQL databases document-oriented are [19, 24, 27]: MongoDB (10gen), CouchDB, CoucHBase, Amazon DynamoDB, MarkLogic, RavenDB, Cloudant, OrientDB, GemFire, RethinkDB, Datameer, Microsoft Azure DocumentDB, ArangoDB, PouchDB, etc.

3.4.4 Graph store

NoSQL databases graph-oriented are used to solve complex problems of connectivity links between data [18, 26]. Social networks like Facebook and Twitter, where millions of users are connected in different ways, are a good example: friends, fans, family, etc. In this context, the challenge is not the number of elements to manage, but the number of relationships that may exist between all these elements. Indeed, there are potentially N^2 relationships for storing N elements. They also allow the elaboration of links between the various interests that an internet user might have, in order to propose products that may to interest him. Thus, the advertisements that appear on Facebook are very often related searches done on Google and the proposals for purchases of online shopping sites such as eBay and Amazon are related to purchases already made [24]. Each entity of this database is named node, connected to each other by edges (the lines) where both the nodes and edges have labels. The W3C has defined a standard called RDF (Resource Description Format) to identify nodes and edges of the graph [28]. The main solutions having adopted the NoSQL graph-oriented are [19, 24, 27]: Neo4j, Orient DB, Titan, ArangoDB, Giraph, InfiniteGraph, Sqrrl, Sparksee, InfoGrid, HyperGraphDB, FlockDB, VelocityGraph, GlobalsDB, GraphDB, etc.

4. COMPARATIVE STUDY

A comparative study is considered very useful for providing, corporate decision makers, indicators and elements of information on the different existing solutions, their performance and adequacies to different fields of use, for any choice and making decisions on the model to be adopted for their projects. These new systems are distinguished based on several criteria: performance, the model used, consistency, methods of storage, durability guarantees, availability, the queries support and other dimensions. These systems generally sacrifice some of these dimensions in favor of others.

Very recent studies show that the NoSQL databases document-oriented, as a model, remains the most used, as shown in Figure 2 [29]. The cause is probably related to its simple format (type JSON). On the other hand, the columns- oriented are in equality with key values. Graph-oriented are in the last position because they are mainly reserved for very specific applications.

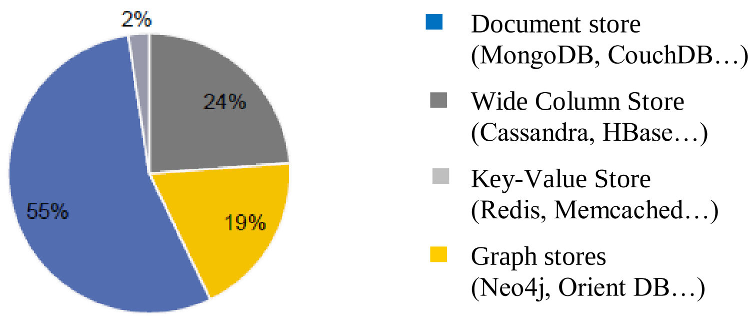


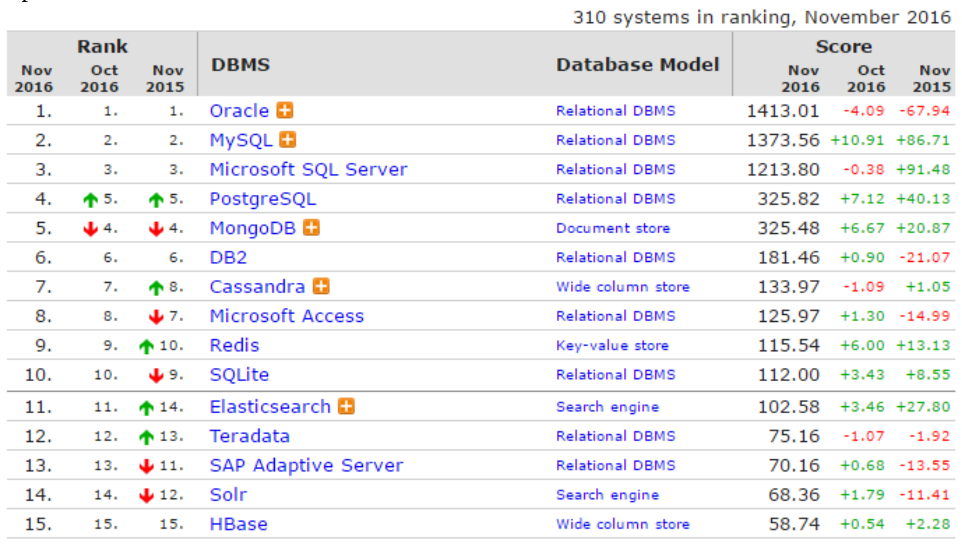
Figure 2. Types of NoSQL mainly used [29]

According to Solid IT [30], the four most popular NoSQL solutions are: MongoDB, Cassandra, Redis, HBase. The DB-Engines Ranking is a list of database management systems ranked by their current popularity. They measure the popularity of a system by using the following parameters:

1) Number of mentions of the system on websites.
2) General interest in the system.
3) Frequency of technical discussions about the system.
4) Number of job offers, in which the system is mentioned.
5) Number of profiles in professional networks, in which the system is mentioned.
6) Relevance in social networks.

The ranking of the Top 15 for November 2016 is shown in Table 2:

Table 2: Top 15 of DBMS [30]



310 systems in ranking, November 2016

Rank			DBMS	Database Model	Score		
Nov 2016	Oct 2016	Nov 2015			Nov 2016	Oct 2016	Nov 2015
1.	1.	1.	Oracle ⊞	Relational DBMS	1413.01	-4.09	-67.94
2.	2.	2.	MySQL ⊞	Relational DBMS	1373.56	+10.91	+86.71
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1213.80	-0.38	+91.48
4.	↑ 5.	↑ 5.	PostgreSQL	Relational DBMS	325.82	+7.12	+40.13
5.	↓ 4.	↓ 4.	MongoDB ⊞	Document store	325.48	+6.67	+20.87
6.	6.	6.	DB2	Relational DBMS	181.46	+0.90	-21.07
7.	7.	↑ 8.	Cassandra ⊞	Wide column store	133.97	-1.09	+1.05
8.	8.	↓ 7.	Microsoft Access	Relational DBMS	125.97	+1.30	-14.99
9.	9.	↑ 10.	Redis	Key-value store	115.54	+6.00	+13.13
10.	10.	↓ 9.	SQLite	Relational DBMS	112.00	+3.43	+8.55
11.	11.	↑ 14.	Elasticsearch ⊞	Search engine	102.58	+3.46	+27.80
12.	12.	↑ 13.	Teradata	Relational DBMS	75.16	-1.07	-1.92
13.	13.	↓ 11.	SAP Adaptive Server	Relational DBMS	70.16	+0.68	-13.55
14.	14.	↓ 12.	Solr	Search engine	68.36	+1.79	-11.41
15.	15.	15.	HBase	Wide column store	58.74	+0.54	+2.28

The four NoSQL systems that are in the Top 15 are of different types and are used in various projects of development and operating. So our choice was on MongoDB as a document-oriented model. For its simplicity of use and development at the client application, as well as it good performance, MongoDB is the most used currently database NoSQL (ranked first NoSQL and fifth all DBMS). The second system is HBase as column-oriented model which is part of the reference implementation Hadoop and very commonly used solution (ranked fourth NoSQL and fifteenth all DBMS).

4.1 HBase

HBase is the Hadoop database, a distributed, scalable, big data store. It is an open source database management system column-oriented store. It is developed as part of Apache Hadoop project. HBase is used by companies like Facebook to store all the messages of the social network [19].

HBase is inspired by the publications on Google BigTable [31]. It uses cache memories for loading the base directly into RAM, which greatly improves performance. It is built to run on top of the Distributed File System of Hadoop (HDFS). Many current researches are interested about this model and in particular to its distributed file management system by searching better performance. An improvement of HDFS has been proposed in our article “New architecture of storage and access in Big Data and Cloud Computing” [32].

4.2 MongoDB

MongoDB is an open-source database document-oriented data that provides high performance, high availability and automatic scalability [33]. Developed since 2007 by 10gen, MongoDB is very adapted to Web applications. Operating as a distributed architecture, it replicates data on multiple servers by the master-slave principle, allowing a larger fault tolerance. The distribution and duplication document is made so that the most requested documents are on the same server and that it is duplicated an enough times [33]. A set of documents is called collection, which can be seen as an SQL table. In order to increase performance while working with documents, MongoDB uses indexing similar to relational databases [34].There are four different modes for MongoDB: Single, Replication Master/Slave, Replica Sets and Sharding. The Sharding is used to share data between multiple Shards; Each Shard must store a part of the data [33].

4.3 Benchmark used and workloads

Yahoo proposes a very powerful tool called YCSB (Yahoo! Cloud Serving Benchmark). It is a new open-source benchmarking methodology, where users can develop their own packages either by defining a new set of workload parameters, or if necessary by writing Java code. It was announced in a paper from Yahoo! in which they presented benchmark results for four widely-used systems:

Apache HBase, Apache Cassandra, Yahoo!'s PNUTS and a sharded MySQL implementation in terms of their performance and elasticity characteristics [35]. In [19], the author makes a call for scalability benchmarks, suggesting YCSB as a good basis for comparison. He affirms that probably, the best benchmark NoSQL is currently YCSB. The benchmark measures raw performance, showing latency characteristics as the server load increases and it measures scaling, showing how the benchmarked system scales as additional servers are added, and how quickly the system adapts to additional servers [19]. It has become the industry standard benchmarking tool for NoSQL systems. This tool is multiplatform and supports the majority of NoSQL systems and easily adapted to these solutions, it often used to compare relative performance of NoSQL database management systems in the cloud and several research studies have used it [1, 2, 3, 4, 5, 6, 19, 34]. YCSB consists of two components: a generator of data and a set of performance tests to evaluate the operations of the inserting, updating, and deleting the records. In each test using some workloads, we can configure the number of records to be loaded, the number of operations to be executed, and the proportion of reading and writing (Example: Workload A: 50% reads, 50% updates). These workloads can be modified and customized depending on the type of test results expected.

5. EXPERIMENTAL RESULTS

Our study was conducted in in a physical machine with the following configuration: Ubuntu 14.10, PC 64-bit, Core i5 and 6 GB RAM. The experiments performed as testing scenarios (Benchmarks) are exposed and compared with the results obtained by [1] in a virtual machine with Ubuntu 32-bit, 2 GB of RAM installed in a PC with OS Windows 7 32-bit and 4 GB RAM. To minimize the effect of changing the CPU and I/O, we ran each test three times on three different days to retain only the average execution time achieved.

After downloading and installing Hadoop 2.6.0, HBase 0.94.8 and MongoDB 2.6.11, we proceeded to the installation and configuration of YCSB 0.8.0, but first, we must be installed Java, Maven and Git in our system. Each test started with an empty database. We started by initializing the YCSB tool with 6 Workloads. Once the data was loaded, workloads (described below) are performed on the two databases. Between each workload, checks the health of the database are performed.

The general objectives of the tests were to:

1) Select workloads typical of today's modern applications.
2) Use data volumes representative of "big data" datasets.
3) Vary the proportions of read / write workloads to test the performance of the two solutions.
4) Retain the same names of workloads with the same rates used in the works of Veronika and Al. [1] for a better similarity of results and a clearer comparison.

The list of workloads consists of:

1) Workload A (Update Heavy): Constituted of a ratio of 50% Read and 50% Update.
2) Workload B (Read Mostly): Constituted of a ratio of 95% Read and 5% Update.
3) Workload C (Read Only): 100% Read.
4) Workload D (Read Latest): 5% Insert, 95% Read (Workload D: Insert 5%, 95% Read (inserts records, with readings of the newly inserted data).
5) Workload E (Short Ranges): 95% Scan, 5% Insert. Small intervals of records are consulted in this test, between 1 and 100, instead of individual records. And inserting the new records.
6) Workload F (Read-modify-write): Constituted of a ratio of 50% Read, 50% Read-Modify-Write

However, in order to better focus on updating operations, we have adopted two additional workloads proposed by [1] and replace the execution of workloads D and E containing insertion operations. Our main interest is the evaluation of the operations of reading, writing and loading of data. The new workloads proposed are:

7) Workload G (Update mostly): Constituted of a ratio of 5% Read and 95% Update.
8) Workload H (Update Only): 100% Update.

YCSB provides the results in a fairly structured text format for each workload (*./bin/ycsb run mongodb -s -P workloads/workloada > mongorunwka.txt* for MongoDB or *./bin/ycsb run hbase094 -P workloads/workloada -p columnfamily=family -s > hbaserunwka.txt* for HBase), which summarizes the overall throughput in operations/second, the average latency and the total test running time. The times of execution chosen as indicators in our study were collected and compared. In the following, we will expose and synthesize the results of loading 600 000 records generated with YCSB. We also present the running time of Workloads obtained during the operations of reading, writing, and updates. All the Workloads execute 1000 operations and measures the average time of execution. Each time, the results are compared with [1]. An overall evaluation of running of all workloads (A-B-C-F-G-H) will be performed thereafter. Before concluding by a synthesis and analysis of our experimental results, we will compare HBase and MongoDB on two distinct aspects that are read operations and write operations separately.

5.1 Data Loading

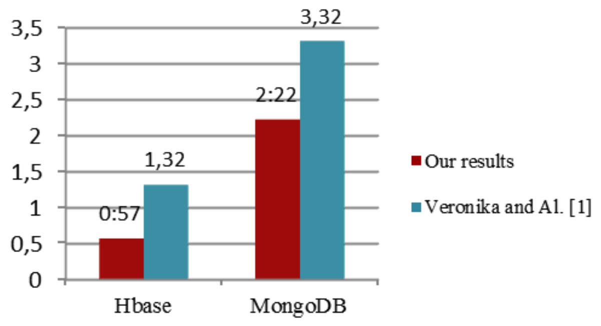


Figure 3. Loading time (Min : Sec)

Figure 3 shows the average execution time for the loadings of 600 000 records in the two databases. We note that HBase has a better insertion time with only 57 sec, more efficient than MongoDB with 2 min 22 sec. The results are very similar to those obtained by [1]. HBase provides a gain of 74% compared with MongoDB in loading operations, against 60% gained in [1]. The reason is that HBase does not require a large amount of memory to run initial loading operations.

5.2 Workload A (50% Read - 50% Update)

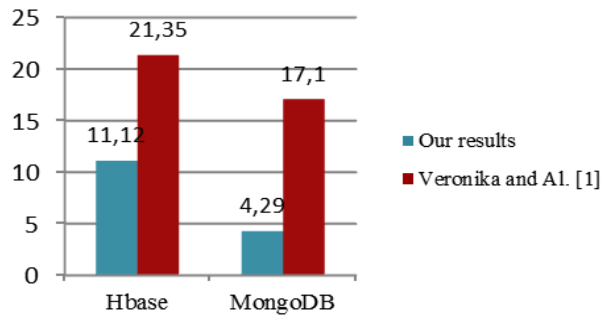


Figure 4. Running time for workload A (Sec)

In the heavy update experimentations illustrated in Figure 4, MongoDB is more efficient to HBase and provides a gain of 61%, against just 20% gained by MongoDB in [1], but it is necessary to see the results of the other workloads to confirm a possible performance.

5.3 Workload B (95% Read - 5% Update)

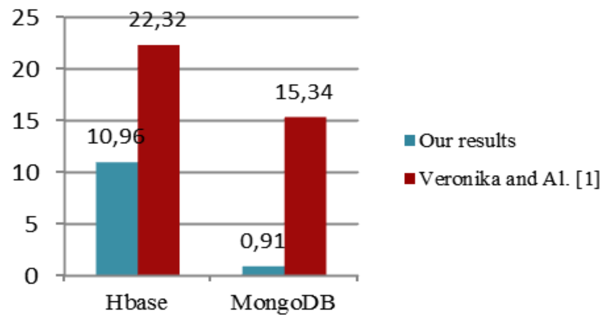


Figure 5. Running time for workload B (Sec)

In the read mostly experimentations which appear in Figure 5, MongoDB is largely better compared to HBase and provides a gain of 91%, against only 31% gained by MongoDB over HBase in [1]. The speed of MongoDB for reading operations is justified by a lack of optimization of HBase to run this kind of operation.

5.4 Workload C (100% Read)

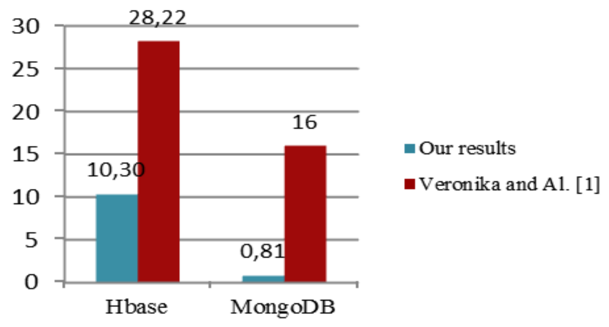


Figure 6. Running time for workload C (Sec)

From Figure 6, we can conclude that for read operations only, MongoDB is even more efficient to HBase and provides a gain of 92% for 43% acquired by MongoDB in [1].

HBase presented more difficulty on reads operations. It is optimized for the writing operations, because it uses sequential write mechanisms where the parts of the same record that are being read are stored in different files, which considerably degrades its running time.

5.5 Workload F (50% Read - 50% Read-Modify-Write)

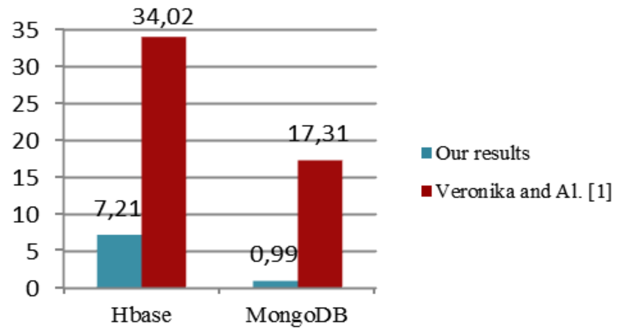


Figure 7. Running time for workload F (Sec)

In this workload (Figure 7), 50% of records are read, modified, and the updates are written in the database permanently. Again in this workload, MongoDB takes the ascendancy over HBase and provides a gain of 86%, against 49% gained by MongoDB over HBase in [1]. We confirm the weak performance of HBase due to its difficulty reading compared to the update.

5.6 Workload G (5% Read - 95% Update)

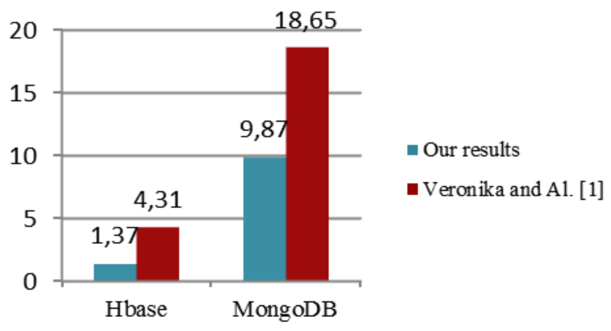


Figure 8. Running time for workload G (Sec)

In the Update mostly experimentations shown in Figure 5, HBase takes the ascendancy again over MongoDB and provides a gain of 86%, against 76% gained by HBase in [1]. HBase has shown better performance by using a log file where all transactions are stored into append mode. By reducing disc I / O operations, due to the records which are stored in cache memories and only the log file that needs to be written on disk, the execution of the updates becomes faster. In general, column-based databases use a sequential write mechanism and these models are well-optimized for running updates.

5.7 Workload H (100% Update)

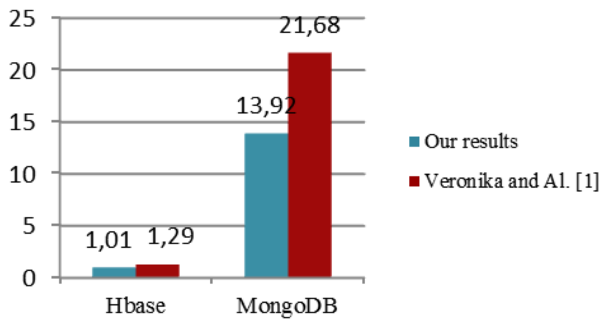


Figure 9. Running time for workload H (Sec)

For update operations only (Figure 9), HBase reconfirms its performance compared with MongoDB and gains 92%. Similarly to [1], HBase provides a gain of 94% over MongoDB. HBase showed a better result compared to MongoDB, due to the mechanisms used by it in the update operations. On the other hand, MongoDB uses a locking mechanism for updating records, which slows down the execution of update operations.

5.8 Overall running time of all Workloads

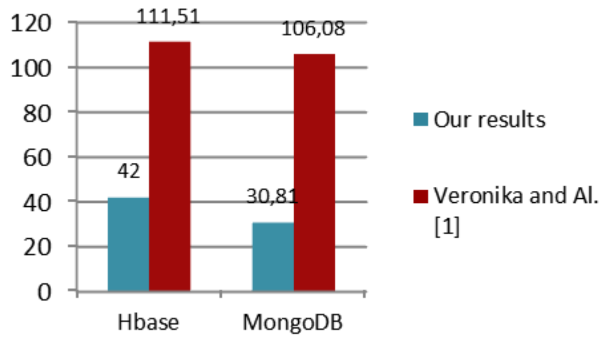


Figure 10. Overall Running Time (A + B + C + F + G + H) (Sec)

Overall in Figure10, experiments performed on all six workloads with different reading and writing rates, proved that MongoDB was slightly performant compared to HBase, with a gain of 26%, against 5% gained by MongoDB over HBase in [1].

5.9 Evaluation globale HBase et MongoDB

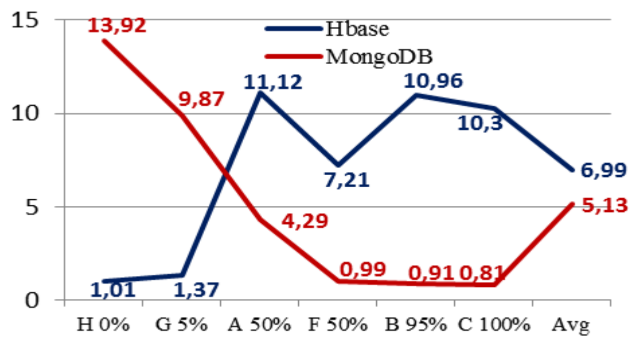


Figure 11. Evaluation HBase and MongoDB for Read operations (Sec)

Figure 11 shows in general that the increased in read operations rate in workloads, is in favor of MongoDB over HBase.

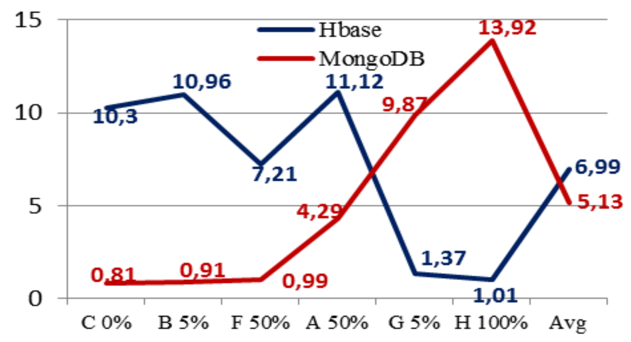


Figure 12. Evaluation HBase and MongoDB for Write operations (Sec)

Unlike figure 11, figure 12 shows in general that the increased write operations rate in workloads, is in favor of HBase compared to MongoDB.

After execution of the six workloads, consisting of 1000 combinations of read and update operations on 600 000 inserted records, we remark a slight advantage on the average performance of MongoDB (5.13 sec) compared to HBase (6.99 sec), representing 22% 26% of gain provided by MongoDB, which confirms the results obtained previously.

6. SYNTHESIS AND ANALYSIS OF RESULTS

After interpreting the results obtained by the two systems, HBase as column-oriented and MongoDB as document-oriented, we have concluded that the various optimizations and mechanisms employed by the designers of NoSQL solutions to enhance performance like good operating of cache memories have a direct influence on the execution time of the different workloads. On the other hand, an optimal hardware environment equipped with fast processors, large RAM memories and data storage on modern disk drives to remedy the slowness of magnetic discs, provide obviously an additional performance to different databases.

Through the experimental evaluations on performances of MongoDB and Hbase, several lessons can be learned:

1) HBase has proved its performance generally in write operations either in the inserting of new records in the database or in the updating of already existing records.
2) The gap in the execution time while loading the 600,000 records is justified by the fact that HBase has a specific memory management which optimizes its use to execute the initial loading.
3) HBase uses a log and caches to save and keep the traceability of all changes to the database, data replication is done automatically, which speeds up the update process and this is also confirmed by [1].
4) For consistency purposes, before running a read operation, HBase compares all copies and returns the most recent copy, which increases the execution time of readings.
5) MongoDB was more efficient in the read operations. This is due mainly to the registers mapping of MongoDB loaded into memory, which increases the reading performance; this is also confirmed by [1].
6) The update process in MongoDB is proportionally slow down as compared to the numbers of updates achieved. This database uses the locking mechanisms that increase the execution time of update.

7. CONCLUSION AND FUTURE WORK

Faced with the dominance of the Internet in most sectors, industry, media, communication and even family, Big Data technologies are now booming. In the coming years, we believe that these technologies will be increasingly used to solve new problems for data management. Logically, NoSQL systems respond favorably to changes of scale and are considered currently as the ideal solution to manage these big volumes of data characterizing the Big Data. The absence of standardization of the NoSQL models and the non-emergence of a particular solution among the existing solutions, were the main motivations that led to the realization of this work.

In this article, we have developed a performance comparative study between two very successful models currently, namely HBase of Hadoop as Wide Column Store model and MongoDB as Document Store model, to provide a set of criteria and indicators to interested users for any decision on the solutions adopted for their companies.

A set of configurable workloads consisting 1000 varied operations of reading and writing have been executed several times in different days on databases containing 600 000 records initially loaded. After analysis and interpretation of the results, we found that MongoDB is overall more efficient compared to HBase in the read operations: Read Only, Read Mostly, Heavy Update and Read-modify-write, unlike HBase which confirmed its better performance in write operations: Data loading, Update Only, Update mostly.

Comparing the results with those obtained by [1], we can note that our results were mostly better.

It can be concluded also that the choice of using a DBMS depends on a number of parameters related to the environment where data are exploited. Indeed the type of data and the type of treatment performed on these data are important indicators to define the solution adopted. The estimated frequencies of reading, inserting, updating and data size are the essential factors in determining the selection of an alternative among others. Currently the trend toward favoring a specific NoSQL solution is very questionable because of the large number of existing systems open-source and proprietary.

Many researches are currently focusing on the topic that is very promising and is considered an open area for researchers. Other comparisons between different families of NoSQL solutions, key-value, document-oriented, column-oriented and graph-oriented, in a cloud environment, are always required and recommended to provide the necessary assistance and help to interested in the domain of Big Data and Cloud Computing for possible decisions on appropriate solutions.

In perspective, in our future studies of evaluations, we will pass to higher scales by increasing the number of operations performed to achieve workloads exceeding 20 000 operations and the number of records inserted to have a big test database hosting millions of records for further testing.

We recall that our comparative study was structured on a single node architecture, testing and performance evaluations in a distributed architecture by multiplying gradually nodes in a parallel and distributed environment, will be the subject matter of further work in the future.

We also plan to extend the comparative study to other popular NoSQL and NewSQL systems as well as SQL relational systems.

REFERENCES

1. Abramova, V, Bernardino, J and Furtado, P (2014). Experimental evaluation of NoSQL databases. *International Journal of Database Management Systems (IJDMS)*, Vol.6, No.3, pp. 1-16.
2. Datastax (2015). Benchmarking Top NoSQL Databases: Apache Cassandra, Couchbase, HBase, and MongoDB. End Point Corporation.
3. Park, HJ (2016). A Study about Performance Evaluation of Various NoSQL Databases. The *Journal of Korea Institute of Information, Electronics, and Communication Technology*, Vol. 9, Issue 3, pp. 298-305.
4. Martins, G, Bezerra, P, Gomes R, Albuquerque F, Costa A (2015). Evaluating performance degradation in NoSQL databases generated by virtualization. In Proceedings IEEE of 8th Latin

American Network Operations and Management Symposium (LANOMS), 1-3 October 2015, João Pessoa, Brazil, pp. 84-91.

5. Abramova, V, Bernardino, J and Furtado, P (2015). SQL or NoSQL? Performance and scalability evaluation. *Int. J. Business Process Integration and Management (IJBPIM)*, Vol. 7, No. 4, pp. 314-321.
6. Lungu, I and Tudorica, BG (2013). The development of a benchmark tool for NoSQL databases. *Database Systems Journal*, Vol. IV, No. 2, p 13-20.
7. Lyman, P, Varian, HR, Swearingen, K, Charles, P, et al. (2003). How much information? UC Berkeley. http://www2.sims.berkeley.edu/research/projects/how-much-info-2003/print.htm
8. Bohn, RE and Short, JE (2009). How Much Information? Report on American Consumers. UC San Diego. http://group47.com/HMI_2009_ConsumerReport_Dec9_2009.pdf
9. Gantz, J and Reinsel, D (2012). Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East. https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf
10. Zhang, Q, Cheng and L, Boutaba, R (2010). Cloud computing state-of-the-art and research challenges. *Journal of Internet Services and Applications*. Vol. 1, Issue 1, pp. 7–18.
11. Agrawal, D, Das, S and El Abbadi, A (2010). Big Data and Cloud Computing: New Wine or just New Bottles? In Proceedings of the VLDB Endowment .Vol. 3, Issue 1-2, pp. 1647-1648.
12. Agrawal, D, Das, S and El Abbadi, A (2011). Big Data and Cloud Computing: Current State and Future Opportunities. In Proceedings of the 14th International Conference on Extending Database Technology (EDBT/ICDT'11), Uppsala, Sweden - March 21 - 24, 2011, pp. 530-533.
13. Mc Creary, D and Kelly, A (2014). Making Sense of NoSQL : A guide for managers and the rest of us. Edition: Manning Publications.
14. Abadi, DJ (2009). Data Management in the Cloud: Limitations and Opportunities. *IEEE Data Engineering Bulletin*, Vol. 32, No. 1, pp. 3–12.
15. Oracle Real Application Clusters (RAC), (2013). An Oracle White Paper.
16. MySQL Cluster CGE, (2013). https://www.mysql.fr/ products/ cluster. Accessed 25 Nov 2016.
17. Rith, J, Lehmayr, PS and Meyer-Wegener, K (2014). Speaking in Tongues: SQL Access to NoSQL Systems. In Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC '14). Gyeongju, Korea, March 24 - 28, 2014, pp. 855-857.
18. Han, J, Haihong, E, Le, G and Dual, J (2011). Survey on NoSQL database. In 6th IEEE International Conference on Pervasive Computing and Applications (ICPCA’2011), 26-28 Oct. 2011, Port Elizabeth, South Africa, pp. 363-366.
19. Cattell, R (2010). Scalable SQL and NoSQL Data Stores. *SIGMOD Record*, Vol. 39, Issue 4, pp. 12-27, Indiana, USA.

20. Nance, C, Losser, T, Iype, R and Harmon, G (2013). NoSQL vs RDBMS - Why There is Room for Both. In Proceedings of the Southern Association for Information Systems Conference, Savannah, GA, USA, March 8th –9th, 2013, pp. 111-116.

21. Chen, Y, Qin, X, Bian, H, Chen, J et al. (2014). A Study of SQL-on-Hadoop Systems. Big Data Benchmarks, Performance Optimization, and Emerging Hardware - 4th and 5th Workshops, BPOE 2014, Salt Lake City, USA, LNCS, Vol. 8807 (Springer), pp. 154-166.

22. Abadi, D, Babu, S, Ozcan, F and Pandis, I (2015). Tutorial: SQL-on-Hadoop Systems. In Proceedings of the VLDB Endowment, Vol. 8, No. 12, pp. 2050–2051, Indiana, USA.

23. Aslett, M (2011). How will the database incumbents respond to NoSQL and NewSQL? https://451research.com/report-short?entityId=66963. (Accessed November 25, 2016).

24. Grolinger, K, Higashino, WA, Tiwari, A and Capretz, MAM (2013). Data management in cloud environments: NoSQL and NewSQL data stores. *Journal of Cloud Computing: Advances, Systems and Applications*, Vol. 2, Issue 1, pp. 1-24.

25. Kumar, R, Gupta, N, Maharwal, H, Charu, S et al. (2014). Critical Analysis of Database Management Using NewSQL. *International Journal of Computer Science and Mobile Computing (IJCSMC)*, Vol. 3, Issue. 5, pp.434- 438.

26. Indrawan-Santiago, M (2012). Database Research: Are We at a Crossroad? Reflection on NoSQL. In Proceedings of the 15th International Conference on Network-Based Information Systems (NBiS'2912), September 26 - 28, 2012, Melbourne, Australia, pp. 45-51.

27. Hecht, R, Jablonski, S (2011). NoSQL evaluation: A use case oriented survey. In Proceedings of the International Conference on Cloud and Service Computing, IEEE. 22 Nov - 24 Nov 2012, Huashan Road, Shanghai, China pp. 336-341.

28. Ma, Z and Yan, L (2016). A Review of RDF Storage in NoSQL Databases. Managing Big Data in Cloud Computing Environments, IGI Global, pp. 210-229.

29. Léonard, M (2014). L'avenir du NoSQL. http://www.leonardmeyer.com/wp-content/uploads/2014/06/avenirDuNoSQL.pdf

30. DB-Engines Ranking http://db-engines.com/en/ranking. Accessed November 5, 2016.

31. Jiang, Y (2012). HBase Administration Cookbook. Packt Publishing Open source community experience distilled.

32. Matallah, H and Belalem, G (2014). New architecture of storage and access in Big Data and Cloud Computing. In Proceedings of the 2nd International Conference on Distributed Systems and Decision (ICDSD'14). Oran, Algeria, pp. 59-66.

33. MongoDB (2016). Release Notes for MongoDB 2.6. http://docs.mongodb.org/manual/release-notes. Accessed November 7, 2016.

34. Abramova, V and Bernardino, J (2013). NoSQL databases: MongoDB vs Cassandra. In Proceedings of the International C* Conference on Computer Science and Software Engineering (C3S2E '13). July 10 - 12, 2013, Porto, Portugal, Pages 14-22.

35. Cooper, BF, Silberstein, A, Tam, E, Ramakrishnan, R and Sears. R (2010). Benchmarking cloud serving systems with YCSB. In Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC'10). ACM, Indianapolis, Indiana, USA, June 10 - 11, 2010, pp.143-154.