IJISCS (International Journal of Information System and Computer Science)

Full Paper
eISSN 2598-246X
pISSN 2598-0793

STUDY AND IMPROVEMENT OF PERFORMANCE OF NoSQL DATABASES: MongoDB, HBase and OrientDB.

*Corresponding author
Email :
noel.bila@unikin.ac.cd

BILA KHONDE Noel[1], BOLUMA MANGATA Bopatriciat[2], MBUYI MUKENDI Eugène[3], BUKANGA CHRISTIAN Parfum[4]
[1,2,3,4]Faculty of Science and Technology, University of Kinshasa
[1,2,3,4]Kinshasa, Democratic Republic of the Congo.

Article history:
Received September 17, 2022
Revised November 27, 2022
Accepted December 10, 2022

Keywords:
NoSQL,
MongoDB,
HBase,
OrientDB,
YCSB.

Abstract
This dissertation adds to the various research works in the field of NoSQL "Not only SQL" databases. These new models propose a new way of organizing and storing data designed mainly to remedy the constraints imposed by the ACID properties on relational models. Our objective was to develop a comparative performance study, between three NoSQL solutions widely used in the market, namely: MongoDB, HBase and OrientDB, to propose to decision makers, elements of information for possible choices of the best appropriate solution for their companies. The Benchmark used to decide between these solutions is the Yahoo Cloud Serving Benchmark

1.0 INTRODUCTION

Relational databases were developed as a technology for storing structured and organized data in table form. Over the years they have become the essential element of organizations and the reference model for data management in information systems, however with the continuous increase of stored and analyzed data, relational databases are beginning to present a variety of limitations. It is in this context that NoSQL databases were developed to provide a set of new data management features while overcoming some of the limitations of relational databases.

2.0 METHOD AND MATERIALS

The analysis of the results obtained from the different experiments, in order to evaluate the performance of the different database models in relation to the nature of the operations performed on these databases. On the other hand, this study was carried out in a single-user environment where all the tests were carried out in an HP laptop with a processor: Brand Intel (R) Celeron (R) CPU N2830@ 2.16 Ghz, (2CPUs) with 4 GB of RAM running on an Ubuntu 12.1 operating system.

2.1. Presentation of the Comparison Tool

For the experimental analysis, we used YCSB (Yahoo Cloud Serving Benchmark) which is a widely used open source Framework for evaluating and comparing different types of active data systems (including NoSQL databases like HBase, Apache, Cassandra, Redis, MongoDB, OrientDB, CouchBase, Voldemort, Tarantool, Elasticsearch). The benchmark consists of two components: a data generator and a set of performance tests to evaluate the read and update operations. Each of the test scenarios is called a workload.

A workload is defined by a set of features such as the number of records to be loaded, the number of operations to be performed and the proportion of read, write and update operations. The benchmark package provides a set of default, but configurable, workloads that can be run, as follows:

- ✓ Workload A: 50% Read, 50% Update.
- ✓ Workload B: 95% Read, 5% Update.
- ✓ Workload C: 100% Read.
- ✓ Workload D: 5% Insert, 95% Read (inserts records, with readings of recently inserted data).
- ✓ Workload D: 5% Insert, 95% Read (inserts records, with reads of recently inserted data).
- ✓ Workload E: 95% Scan, 5% Insert.
- ✓ Workload F: 50% Read, 50% Read-Modify-Write.

In order to better understand database optimization and operation update speed, we created two additional workloads with the following characteristics:

- ✓ Workload G: 5% Read, 95% Update
- ✓ Workload H: 100% Update.

To evaluate the loading time, we generated 600,000 records, each with 10 fields of 100 bytes randomly generated on the registry identification key, that is about the total of 1kb per record. Each record is identified by a key consisting of the string "user" followed by several digits, for example "user 3799004308120", which is the record key. Each field of the record is identified as field0, field1, -- Field i respectively. The execution of the workloads consisted of running 1000 operations, which means that there were 1000 requests to the database under test each time. In our study, graph-oriented databases were not evaluated. Because, as stated by Amstrong. T, Ponnekanti. V, Dhruba. B and Callaghan. M, they should not be evaluated according to the scenarios used in the analysis of other types of NoSQL databases (column-oriented, document-oriented, and key-value) because the use of links between records requires a different approach, so there are specific benchmarks developed to evaluate the performance of graph databases such as, XGDbench.

2.2. PRESENTATION OF THE VERSIONS OF NOSQL Solutions

✓ The comparative study developed allowed us to distinguish between the following NoSQL databases:

- ✓ MongoDB: version 2.6.11
- ✓ HBase: version 0.94.8
- ✓ OrientDB: version 2.1.3

2.3 IMPLEMENTATION OF YCSB

To set up YCSB, we first need to install Java, Maven and Git in our system.

a. Java

✓ The site used for downloading Oracle Java JDK and JRE binary archives: http://www.oracle.com/technetwork/java/ja vase/downloads/index.html.

After accessing the directory /home/"username"/Downloads we copied the Oracle Java binary archives.

b. Maven

The site used for the download of apache Maven: https://maven.apache.org/download.cgi

c. Git

We have installed and configured Git with the following commands: sudo apt-get install git git config global user.name"YOUR NAME".git config-global user.email "YOUR EMAIL ADDRESS

2.4. PRESENTATION AND ANALYSIS OF EXPERIMENTAL RESULTS

As a preamble, we emphasize that the results retained for each workload are the averages of several tests performed on at least three different days.

2.4.1. LOADING DATA (LOADPROCESS)

a. MongoDB

The command is: ./bin.ycsb load mongodb-Pworkloads/worloada-p Mongodb.url=mongodb://localhost: 27017/ycsb ?w=0 -s> mongoload.txt

After launching the latter, the loading result obtained from the terminal is as follows:

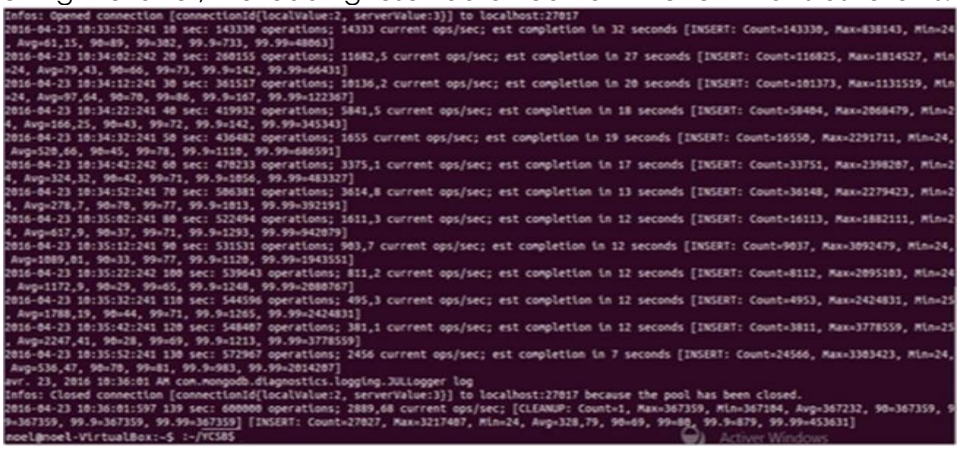

Figure 1. Load Result Obtained from Terminal

On the other hand the loading log generated by YCSB for MongoDB is obtained in a text file "mongoload.txt":

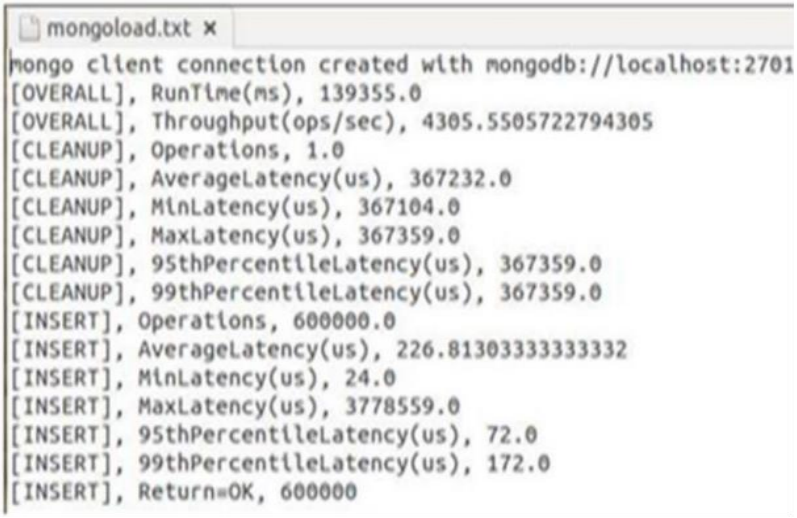

Figure 2. Loading Log Generated by YCXB

b. HBase

./bin.ycsb load hbase094 -P workloada -p columnfamily=family -s

Orient DB

./bin/ycsb load orientdb -P workloads/workloada -p orientdb.url=plocal:/tmp/ycsb -p orientdb.user=admin -p orientdb.password=admin

3.0 RESULTS

The table below shows the average load of each database:

Table 1. Average Loading Time

DATABASES	MongoDB	HBase	OrientDB
Time Min)	2,7	4	3,5

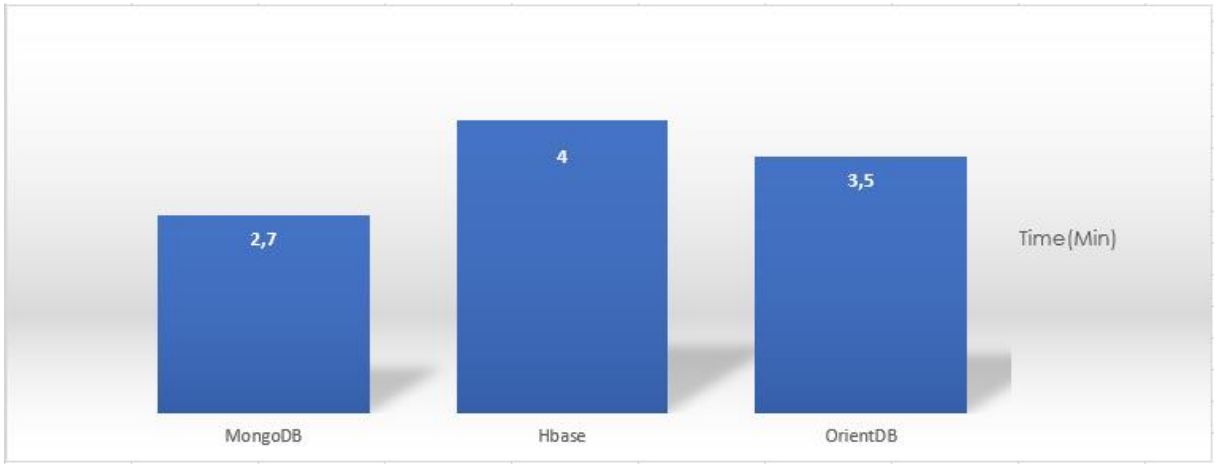

Figure 3 Average Loading Time Histogram

The figure shows the expired times for the 600,000 record load operation for each of the tested databases. The document oriented databases performed better on average. We found that during the loading of 600000 records, the best time is obtained by MongoDB with a loading time of 2 minutes and 7 seconds. For the loading of 600000 records, the column-oriented database (HBase) was less performing compared to the first one. Generally speaking, we can say that the best result obtained among the three tested databases for the loading phase is that of MongoDB, which can be justified by the fact that the latter offers high performance due to its innovations that allow it to be the first to exploit an integrated RAM caching layer.

3.1 WORKLOADS EXECUTION

All workloads consist of a set of 1000 different operations performed on the 600000 records already loaded in the databases.

3.2 WORKLOAD A (50% Read/ 50% Update)

a. MongoDB

./bin/ycsb run mongodb -s -P workloads/workloada> mongorunwka.txt

mongorunwka1.txt ×

```
mongo client connection created with mongodb://localhost:2701
[OVERALL], RunTime(ms), 11926.0
[OVERALL], Throughput(ops/sec), 83.85041086701325
[CLEANUP], Operations, 1.0
[CLEANUP], AverageLatency(us), 2471.0
[CLEANUP], MinLatency(us), 2470.0
[CLEANUP], MaxLatency(us), 2471.0
[CLEANUP], 95thPercentileLatency(us), 2471.0
[CLEANUP], 99thPercentileLatency(us), 2471.0
[READ], Operations, 490.0
[READ], AverageLatency(us), 10949.430612244898
[READ], MinLatency(us), 787.0
[READ], MaxLatency(us), 123711.0
[READ], 95thPercentileLatency(us), 23071.0
[READ], 99thPercentileLatency(us), 40127.0
[READ], Return=OK, 490
[UPDATE], Operations, 510.0
[UPDATE], AverageLatency(us), 11206.825490196079
[UPDATE], MinLatency(us), 746.0
[UPDATE], MaxLatency(us), 63615.0
[UPDATE], 95thPercentileLatency(us), 22735.0
[UPDATE], 99thPercentileLatency(us), 38271.0
[UPDATE], Return=OK, 510
```

Figure 4 Workload Execution

b. HBase

./bin/ycsb run hbase094 -P workloads/workloada -p colunmfamily=family -s >

c. OrientDB

./bin/ycsb run orientdb -P workloads/workloada -p orientdb.url=plocal:/tmp/ycsb -p orientdb.user=admin -p orientdb.password=admin

The following table shows the average execution time of workload A for each of the databases:

Table 2. Workload A Execution Time

DATABASES	MongoDB	HBase	OrientDB
Time (Sec)	10	34	29

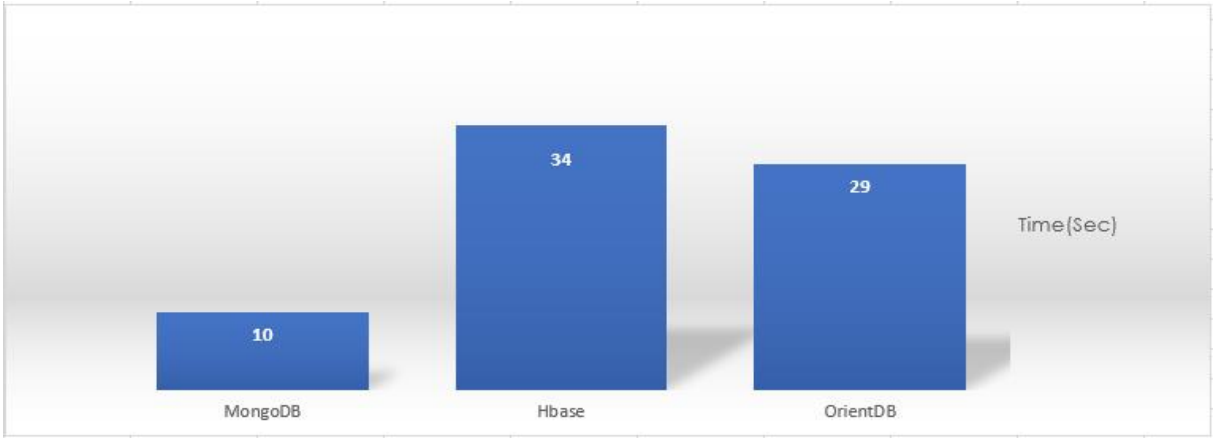

Figure 5. Workload A time histogram

The Figure shows the results obtained during the execution of Workload A composed of 50% Read and 50% Update operations of 1000 operations, performed on 600000 records. After reading the results obtained, we notice that the good performances are presented first by the document oriented category where MongoDB was the fastest. In second place we find the key value category with 29 seconds. The Hbase database is the least performing relative to the previous ones. Indeed, we have to see the results of the M and O loads to favor a NoSQL solution over the others.

3.3 Workload B (95% Read, 5% Update)

The table and figure below show the average execution time of workload B for each of the databases:

Table 3. Workload B Execution Time

DATABASES	MongoDB	HBase	OrientDB
Time (Sec)	10	39	14

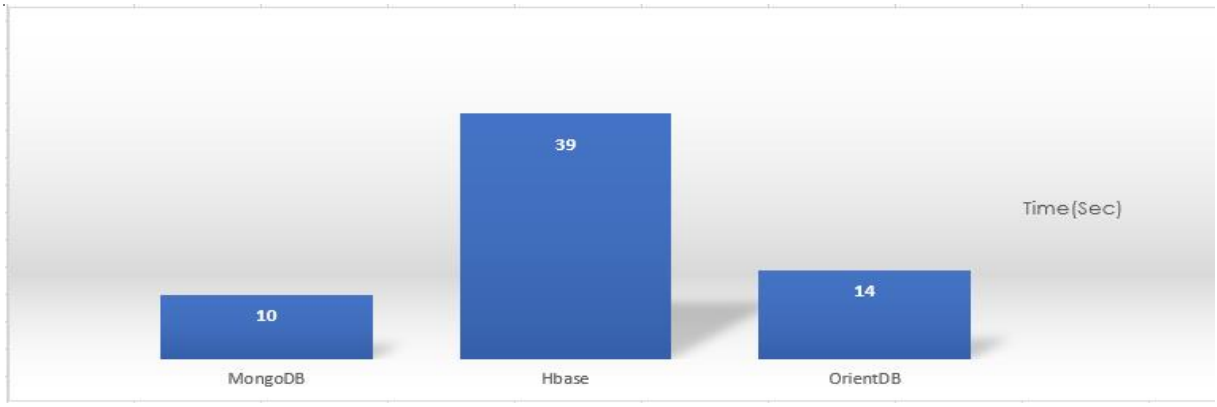

Figure 6 Workload B time histogram

The results in the figure prove the ascendancy once again of document-oriented models for loads composed mainly of read operations. On the other hand, the key-value ones proved their performance over the column-oriented models. Let us also recall that MongoDB won another time over the others with a duration of 10 seconds.

3. 4. Workload C (100% Read)

The table and figure show the average execution time of workload C, consisting only of read operations, for each of the databases:

Table 4 Workload C Execution Time

DATABASES	MongoDB	HBase	OrientDB
Time (Sec)	9	46	11

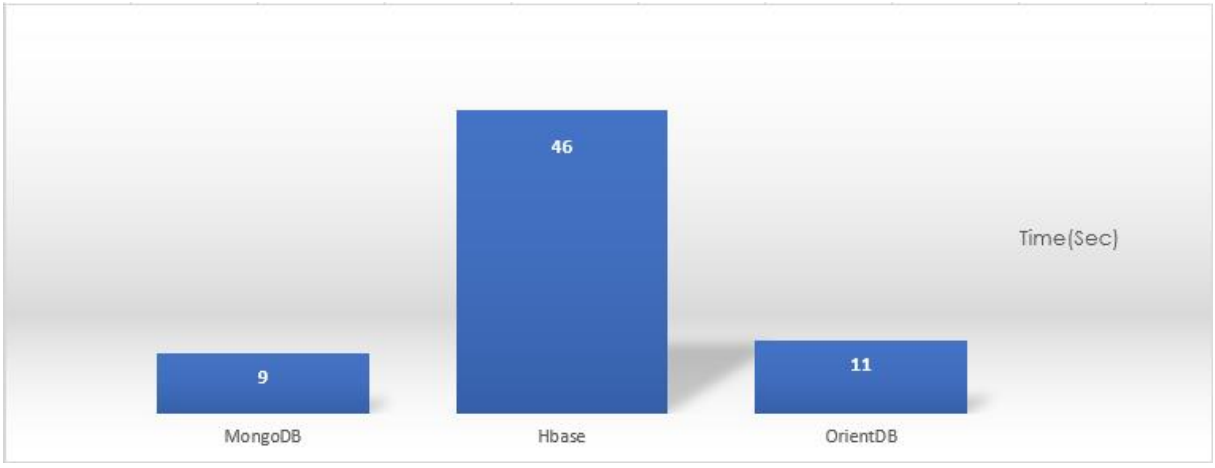

Figure 7 Workload C Time Histogram

For purely reading operations, the results obtained confirm the previous ranking of the bases when executing the B load.

3.5 Workload D (5% Insert, Insert, 95%)

The following table and figure display the average execution time of workload D for each of the databases.

Table 5. Workload D Execution Time

DATABASES	MongoDB	HBase	OrientDB
Time (Sec)	5	43	8

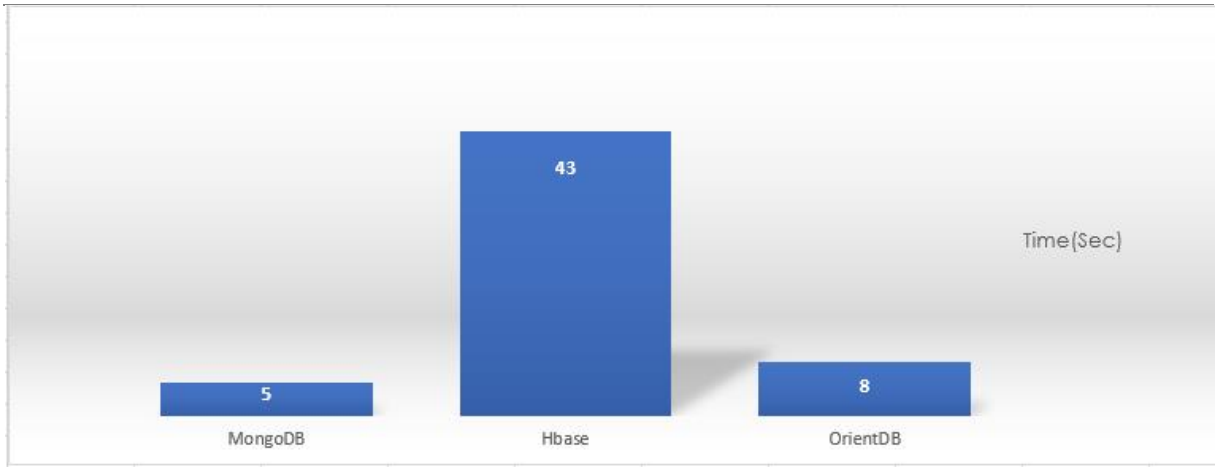

Figure 8 Workload D time histogram

Workload D is composed mainly of read operations (95%) and 5% of insertion operations of new records that are inserted and then read again. By running this workload, we once again confirm the performance of document-oriented systems, especially MongoDB which was very fast by running the task in 5 seconds, and the underperformance of column-oriented and especially HBase with 43 seconds.

3. 6 Workload E (95% Scan, 5% Insert)

The table and figure below show the average execution time of workload E for each database:

Table 6 Workload E Execution Time

DATABASES	MongoDB	HBase	OrientDB
Time (Min)	2,75	1,15	13,5

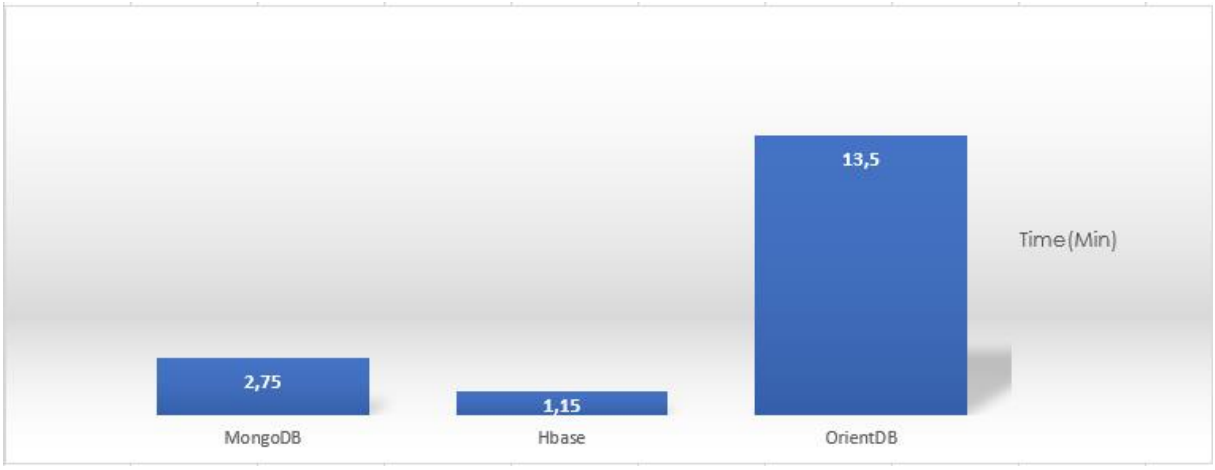

Figure 9. Workload E time histogram

This workload is mainly made up of 95% fast scan operations and 5% insertion of new records. In this test, HBase performed the best by having the best execution time (1 minute and 15 seconds) compared to the other databases, this is explained by the fact that it uses views to query the data. From an overall point of view, the column-oriented database HBase presented the best performance, unlike the key-value database which had the lowest performance and in particular was the slowest in running the workload in 13 minutes and more.

3. 7 Workload F (50% Read, 50% Read- Modify-Write)

The table below shows the average execution time of the F workload, half read and half write, for each of the databases:

Table 7 Workload F Execution Time

DATABASES	MongoDB	HBase	OrientDB
Time (Sec)	11	36	23

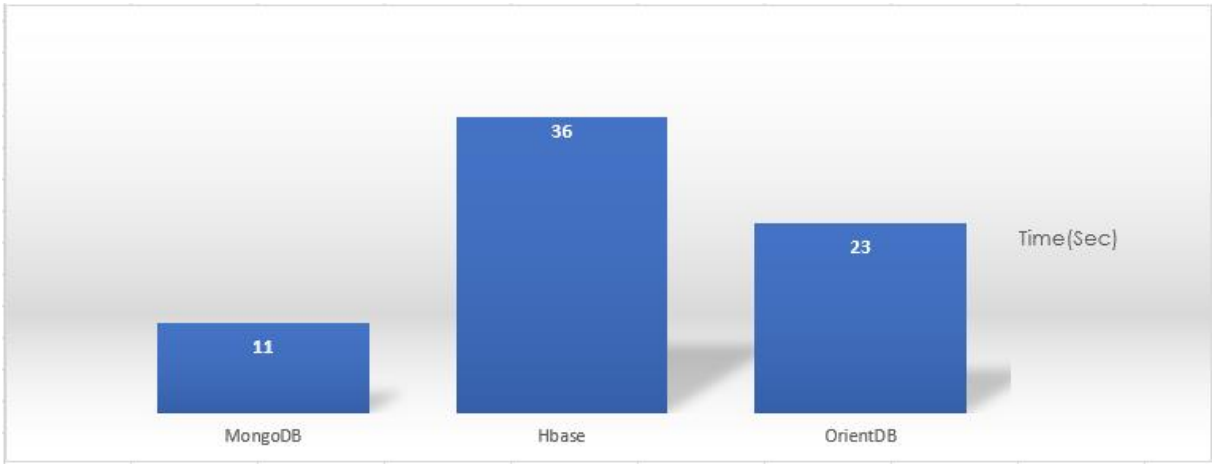

Figure 10. Workload F time histogram

The Figure shows the results obtained after running Workload F composed of 50% read, for the other 50%: records are read first, updated after and then saved. We have seen once again the counter performance of column-oriented system namely HBase because of its read constraint compared to the update. On the other hand the best performance is that of MongoDB which is proving its efficiency for read operations.

3.8 Workload G (5% Read, 95% Update)

The following table and figure show the average execution time of the G workload for each database:

Table 8 Workload G Execution Time

DATABASES	MongoDB	HBase	OrientDB
Time (Sec)	12	7	21

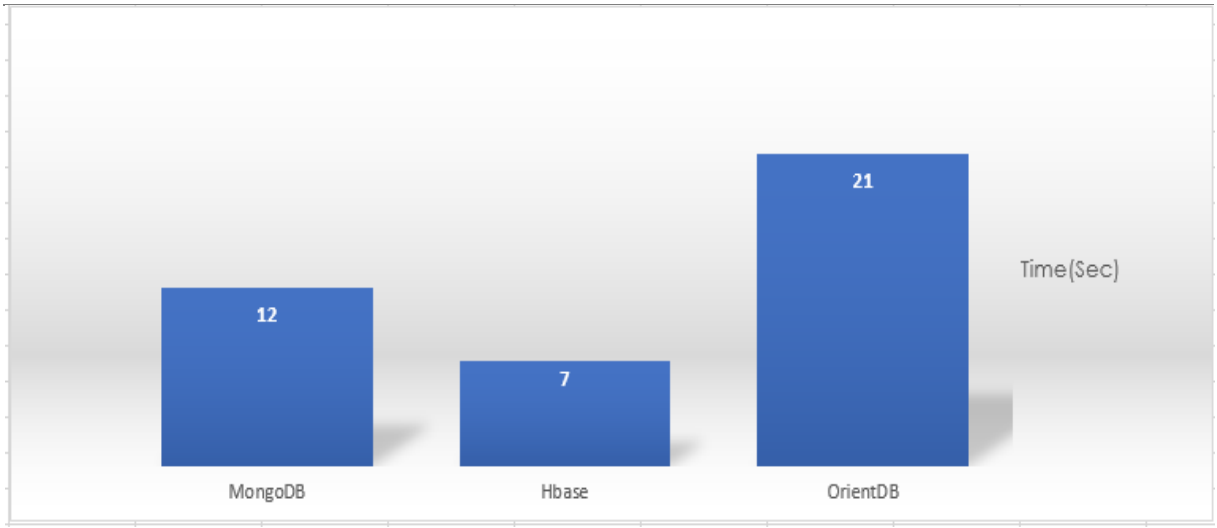

Figure 11 Workload G time histogram

The results reveal that for a load composed mainly of updates, the column-oriented database regained the ascendancy over the other architectures, while those of the key-value were largely above.

3.9 Workload H (100% Update)

The table and figure below show the average execution time of the H workload for each of the databases:

Table 9 Workload H Execution Time

DATABASES	MongoDB	HBase	OrientDB
Time (Sec)	10	6	21

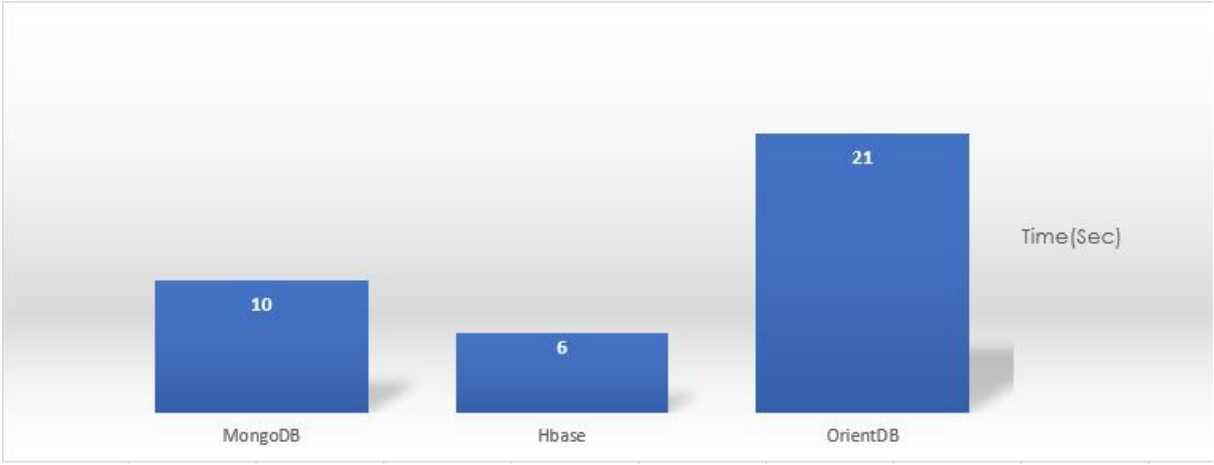

Figure 12 Workload H time histogram

For purely update operations, the HBase column-oriented database confirms its performance achieved during G-load execution compared to all other NoSQL systems.

3.10. Performance summary of all workloads

The table and figure summarize the results obtained by NoSQL databases for all workloads (A + B+ C+ D+ E+ F+ G + H).

Table 10 Global execution time

DATABASES	MongoDB	HBase	OrientDB
Time (Min)	4	5	15

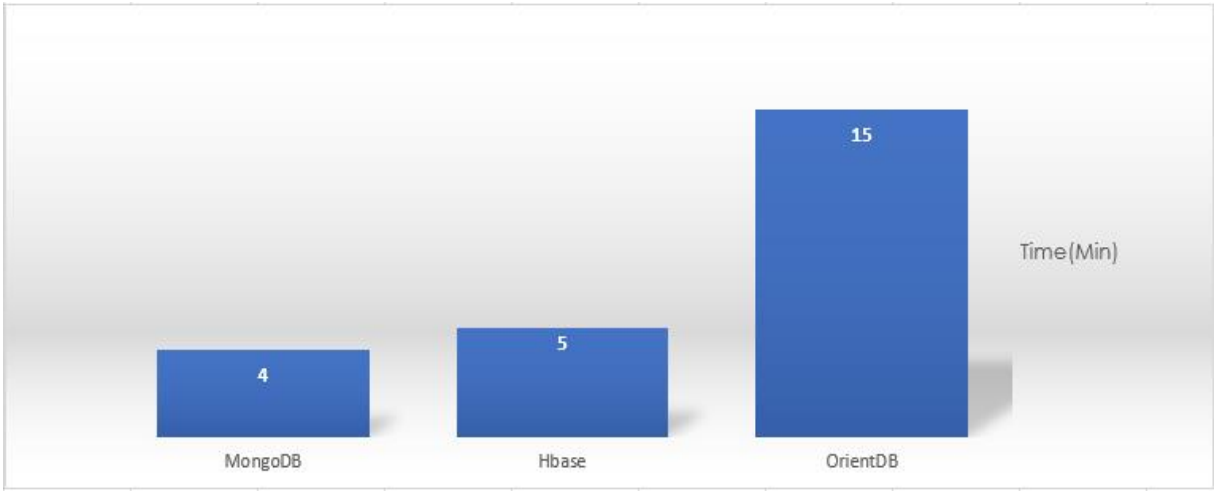

Figure 13 Global Execution Time Histogram

From a global point of view, document-oriented and column-oriented models are largely more efficient than key-value models. The best execution time is presented by MongoDB with 4 minutes followed by Hbase which is in second position with 5 minutes and OrientDB in last position.

4.0 Global evaluation of MongoDB, HBase, OrientDB

The experimental results of the different tests carried out allowed us to evaluate and compare the three types of NoSQL databases: document-oriented, column-oriented and value-key, based on the execution time of the different workloads. After reading the results, we can conclude that the choice criteria depend on the application needs and the nature of the operations performed on the data. For performance and optimization purposes, we can specialize NoSQL databases according to the appropriate model and the context of use of the latter. Among the NoSQL solutions studied, it is stated that there are those optimized for reads, those for updates and others for scan operations:

- ✓ For purely read operations, it is necessary to turn to document-oriented architectures such as MongoDB.
- ✓ For heavy update operations, it is very interesting to adopt column-oriented architectures.
- ✓ For scanning operations, MongoDB and HBase have proven their performance.
- ✓ For key-value architectures, a lot of effort remains to be made by designers to improve their performance.

5.0 CONCLUSION

a. General Conclusion

Our end of studies project consisted in a comparative study of the performances between the different families of NoSQL solutions: document oriented, column oriented and value key oriented namely MongoDB, Hbase, and OrientDB. The objective of this study is to evaluate the performance of these databases by first inserting 600,000 records, then launching a set of tests in the form of workloads composed of 1000 operations each of different natures: read, scan or update. The tool used to arbitrate the three systems is Yahoo! Cloud Serving Benchmark, which is highly recommended for this kind of study in the NoSQL database domain. After reading and analyzing the experimental results, we can say that there are databases that perform very well for particular workloads, unlike others that were better in other workloads. In conclusion, we can retain that the choice of using a DBMS depends on a set of parameters related to the environment in which the data are exploited. Indeed, the type of data and the type of processing carried out on this data are important indicators for defining the solution to adopt. The estimated frequency of reading, writing and updating as well as the size of the data are the essential factors determining the choice of an alternative among others. Currently, the trend towards a specific NoSQL solution is far from being indisputable because of the large number of existing systems. Several open source and paid solutions are presented to the different actors concerned.

b. Perspectives

Finally, we can consider that the objective outlined beforehand has been largely achieved, nevertheless this work could be completed and extended on several aspects, so we can highlight a set of perspectives and research tracks to explore, let's quote: Extend our study to other NoSQL solutions such as: Elasticsearch, Memcached, Amazon Dynamo, CouchDB, Cassandra, Redis, Accumulo and others. Multiply the number of records to reach or exceed one million. Diversify workloads by creating new ones.

REFERENCES

[1] Kaur, S., & Kaur, K. 2016. Visualizing class diagram using orientDB NOSQL data–store. International Journal of Computer Applications, 145(10), 11-16.

[2] Diogo, M., Cabral, B., & Bernardino, J. 2019. Consistency models of NoSQL databases. Future Internet, 11(2), 43.

[3] Solovko, I. S. 2020. System for development, ceaseless integration and configuration of server software.

[4] Yildiz, G., & Wallström, F. 2019. Evaluation of Couchbase As a Tool to Solve a Scalability Problem with Shared Geographical Objects.

[5] Laurent AUDIBERT, 2018. Database and SQL Language, Cours-BD, Institut Universitaire de Technologie de Villetaneuse, Département Informatique.

[6] Marie-France, 2017. LASALLE, Cours du Relationnel a l'objet: Limites du Relationnel.
[7] Mark WHITEHORN, 2016. When to consider using a NoSQL database (rather than a relational database), University of Dundee. Availableat: http://www.lemagit.fr/conseil/Quand- envisager-NoSQL.
[8] Matthieu ROGER, 2017. Bases NoSQL, synthesized studies and intersoftware projects, octera [AT] octera [DOT] info.
[9] Meyer Léonard, 2014. L'avenir du NoSQL Quel avenir pour le NoSQL. Availableat:http://leonardmever.com/wpcontenu/uploads/2014/06/avenirDuNoSQL.pdf.
[10] Rudi BRUCHEZ, 2017. NoSQL databases and BIGDATA understand and implement, 2nd edition: EYROLLE EDITIONS, www.editions-eyrolles.com.
[11] Adriano Girolamo PIAZZA, 2019. NoSQL State of the Art and Benchmark; Bachelor's work carried out with a view to obtaining the Bachelor's degree HES; Geneva, Haute École de Gestion de Genève (HEG-GE).
[12] Kouedi Emmanuel, (May 2018). Relational database migration approach Towards a column-oriented NoSQL database, Dissertation presented for the diploma of MASTER II RESEARCH, Option: S.I & G.L; University of YAOUNDE I.
[13] Lionel Heinrich, 2012. NoSQL Architecture and answer to the CAP Theorem, Bachelor's work carried out in view of obtaining the Bachelor HES in Business Informatics, Geneva, Haute École de Gestion de Genève (HEG-GE).