Comparing NoSQL Databases with YCSB Standard Benchmark

Nadia Ben Seghier[1], Okba Kazar[2]
LINFI Laboratory, Computer science department, University of Biskra, Algeria
inf_nadia@yahoo.fr, kazarokba@gmail.com

Abstract. NoSQL data-stores are commonly used to provide flexibility and availability for Big Data handling. The important companies in the IT sector find these NoSQL systems, new solutions to respond to scalability needs. These databases can be broadly classified as key value stores, column based, document stores and graph database depending on their mechanism of data storage and other features. NoSQL databases assert that their performance is better than legacy Relational Database systems for higher workloads, particularly common in Big Data and Cloud Computing applications. 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 three solutions widely employed: Redis 3.0.504, MongoDB 4.4.0 and Cassandra 3.11.1, and tests the runtime for different proportions of read, update, scan, read-modify-write and insert operations using six workloads by YCSB 0.17.0 tool on Windows OS. The purpose of our comparative study 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: Cloud Computing, Big Data, NoSQL, MongoDB, Cassandra, Redis, YCSB, Windows OS.

1 Introduction

The standard for storing data in computer systems today is with the use of database management systems. The dominating standard among these systems has been relational database management systems, commonly programmed with the SQL language. With the rise of big data, however, different systems have started to take growth and are quite common among today's big companies. These are commonly known as NoSQL databases.

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 [1]. 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 [2]. 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 [3]. 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, this 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. This paper compares the performance of three different NoSQL systems which are Redis, MongoDB and Cassandra, and tests the runtime for different proportions of read, update, scan, read-modify-write and insert operations. The authors use the popular Yahoo Cloud Serving Benchmarking Tool (YCSB), which provides execution of put and get operations. This allows a user to better understand performance of databases and find out which system is better for a particular workload.

The remainder of this paper is structured as follows. First, Section 2 discusses related work. Then, Section 3 describes the test configuration details. Section 4 shows the experimental evaluation and results. After the evaluation of performance of each database, various experimental results of this comparative study will be synthesized in Section 5. Finally, the authors conclude this article with a summary and some perspectives for future works in section 6.

2 Related Work

Benchmarking of Database systems has become a very important research topic for testing and comparing the performance of various database systems. This section provides a brief overview of the current research trends in the domain of database benchmarking related to our goal of comparing the performance of four NoSQL database systems on a single node. Many workbench tool has been proposed to effectively run benchmark tests to find out the performance of database systems across various workloads.

There are four main challenges that are considered in big data applications: (1) scalability of computing algorithms; (2) querying and processing technologies (including data organization and system management); (3) planning techniques and tools; and (4) fault tolerance [4]. Hence, the characteristics of big data performance evaluation are different from traditional performance evaluation. Cattell [5] studies NoSQL systems based on their data model, consistency measures, storage mechanisms, durability guarantees, availability, and query support. Trade-offs between these metrics are emphasized in this study. In [6], the authors compare the performance of Hadoop Distributed File System (HDFS) against MongoDB using simple queries. In addition, analysis on connecting Hadoop and MongoDB to avoid the problem of data locality was studied.

There have been various research efforts to benchmark distributed computing frameworks. Among them, Big- DataBench [7] and Bigbench [8] are benchmark suites that evaluate structured, unstructured and semi-structured data on various workloads (e.g. Sort, Index, Page-Rank and Kmeans). Their analysis focuses on identifying performance trends. With the growth of Internet of Things (IoT), data from sensors is prevalent. Hence, Van et al. compared Cassandra and MongoDB for the storage of sensor data [9].

With the increased usage of NoSQL datastore in the industry, there have been comparative studies on industrial use-cases. Tilmann et al. [10] presented a comprehensive performance evaluation of six data stores (HBase, Cassandra, Voldemort, Redis, VoltDB, MySQL) in the context of application performance monitoring as part of CA Technologies initiative. They evaluated these systems using workloads that include, but not limited to, performance monitoring, online advertisement and power monitoring. Endpoint Corporation did a broad comparison between the then-current top NoSQL databases [11]. This evaluation focuses more on the basic operations, and focuses less on complex queries. There is an inherent trade-off between latency and throughput.

[12] presented a comparative study of five NoSQL databases: Cassandra, MongoDB, CouchDB, Hbase and SimpleDBd. This theoretical survey focuses more on the NoSQL models and their de-

scriptions, and when they are best used. The compared advantages and disadvantages of these data stores are listed to discuss selecting appropriate NoSQL database which processes huge volumes of data; and provides global overview of these non-relational NoSQL databases.

In [13], the authors develop a comparative study about the performance of six solutions NoSQL, employed by the important companies in the IT sector: MongoDB, Cassandra, HBase, Redis, Couchbase, and OrientDB. To compare the performance of these NoSQL systems, they used Yahoo! Cloud Serving Benchmark tool using workload A, workload B, workload C, workload F.

In [14], performance evaluation of web collection data in data stores,such as NoSQL-Cassandra and MongoDB is presented. The authors design a PBRA (Passphrase Based REST API) system to deal with highly unstructured big data emerging from the twitter social networking service, which is new of its kind to strengthen the bigdata security.

In this paper, we develop a theoretical and practical comparative study about the performance of three main NoSQL data-stores: Redis, MongoDB and Cassandra in aspects relating with, query performance, based on reads, inserts, scans and updates by running custom workloads (A, B, C, D, E and F) using the YCSB tool. 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.

3 Test Environment

The table below provides the complete hardware and software components used for our study.

Table 1. Test environment details.

Hardware	Software
Processor : Intel® Core™ i3-3210M CPU @ 2.50 GHz RAM : 8.00 GB HDD: 370 GB	OS Windows10, 64 bits YCSB 0.17.0 Redis 3.0.504 MongoDB 4.4.0 Cassandra 3.11.1

4 Evaluation and Result

4.1 Workloads

YCSB provides the results in a fairly structured text format for each workload, 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 800 000 records generated with YCSB. We also present the running time of Workloads obtained during the operations: read, updates, insert, scan and read-modify-write. All the Workloads execute 100 000 operations and measures the average time of execution. An overall evaluation of running of all workloads (A-B-C-D-E-F) will be performed thereafter. Before concluding by a synthesis and analysis of our experimental results, we will compare Redis, MongoDB and Cassandra on two distinct aspects that are read operations and write operations separately.

4

4.2 Data Loading

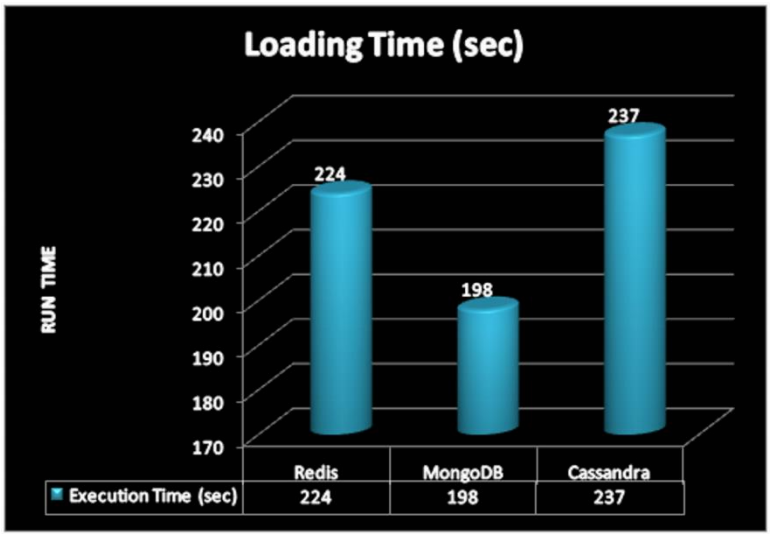

Fig .1. The execution time for the databases when running the loading phase

Figure 1 shows the average execution time for the loading of 800 000 records in the three databases. We note that MongoDB has a better insertion time with only 198 sec, more efficient than Redis with 224 sec and Cassandra with 237 sec. The reason is that MongoDB does not require a large amount of memory to run initial loading operations.

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

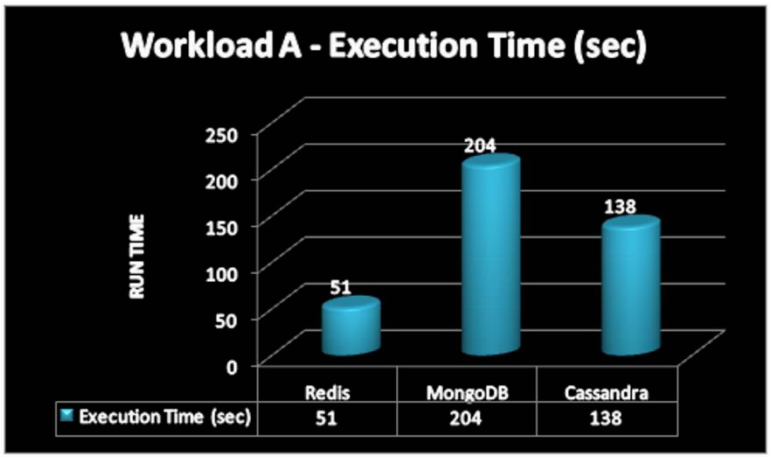

Fig.2. The execution time for the databases when running workload A

This workload allows to verify the performance of the databases when executing, read and update operations. Column databases, like Cassandra, are more optimized for updates, as they load large volumes of records into memory. Document store database, MongoDB, shows the greatest performance time (higher is worst) with 204 sec. The main reason for this difference is the way records are

kept on disk, rather than on memory. MongoDB poor results occur because of the locking mechanisms in place when performing updates, leading to an increase in execution time. Finally, key-value database (Redis), operate mainly in volatile memory. Records are directly mapped from memory; therefore, performance increases substantially.

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

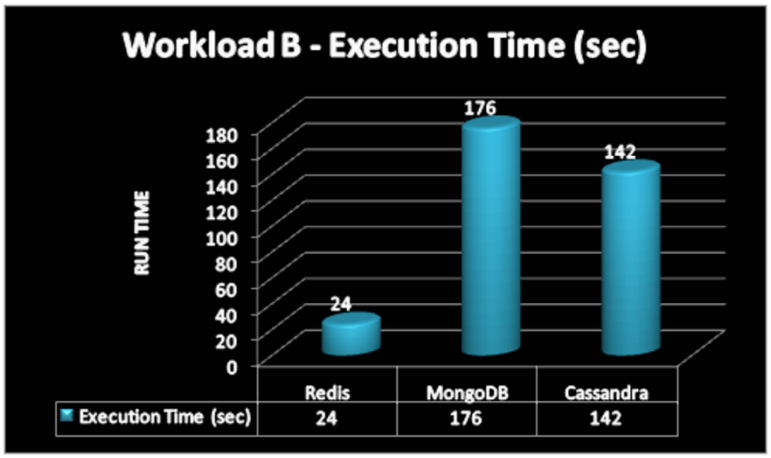

Fig.3. The execution time for the databases when running workload B

Like Figure 2, in the read mostly experimentations which appear in Figure 3, Redis is largely better compared to MongoDB and Cassandra. Redis has the best performances with 24 sec. The speed of Redis for reading operations is justified by the use of volatile memory, to store and retrieve data, allowing this way lower execution times.

4.5 Workload C (100% Read)

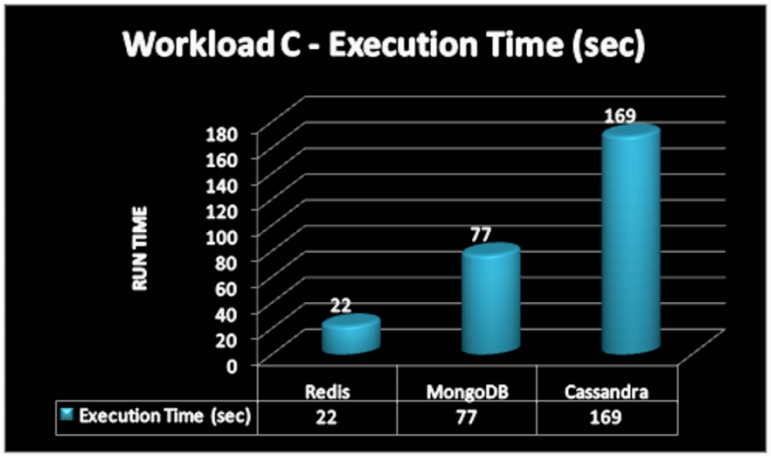

Fig.4. The execution time for the databases when running workload C

6
In this workload (Figure 4), Redis has the best performance, followed by MongoDB based on the time they take to perform reading operations. Redis is obviously optimized for read operations. It is the faster due to its in-memory nature. Cassandra presented more difficulty on reads operations.

4.6 Workload D (5% Insert - 95% Read)

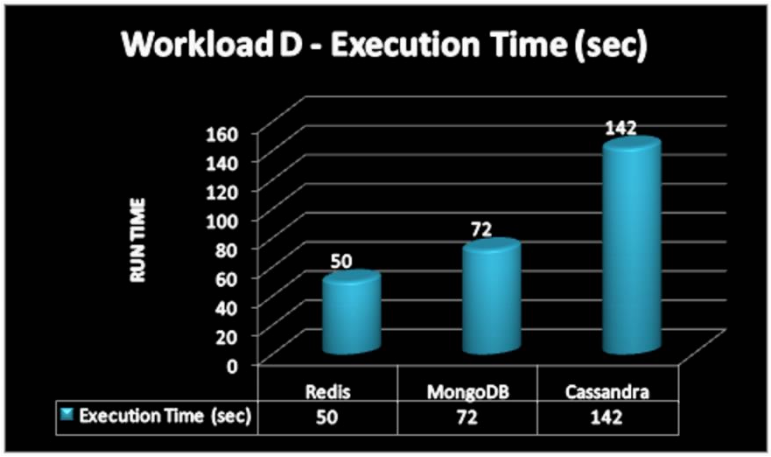

Fig.5. The execution time for the databases when running workload D

In Figure 5, Redis has the best performance with 50 sec, followed by MongoDB with 72 sec. We confirm the weak performance of Cassandra due to its lack of optimization to run this kind of operations.

4.7 Workload E (95% Scan - 5% Insert)

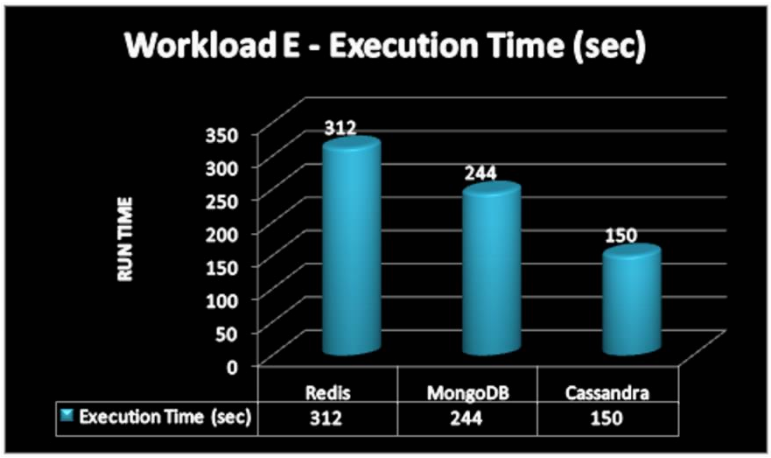

Fig.6. The execution time for the databases when running workload E

The performance of the three NoSQL systems as we use the scan and insert proportions is reported in Figure 6. We can observe that Redis performs the worst proportions of scan and insert. Cassandra is best at handling this type of workloads.

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

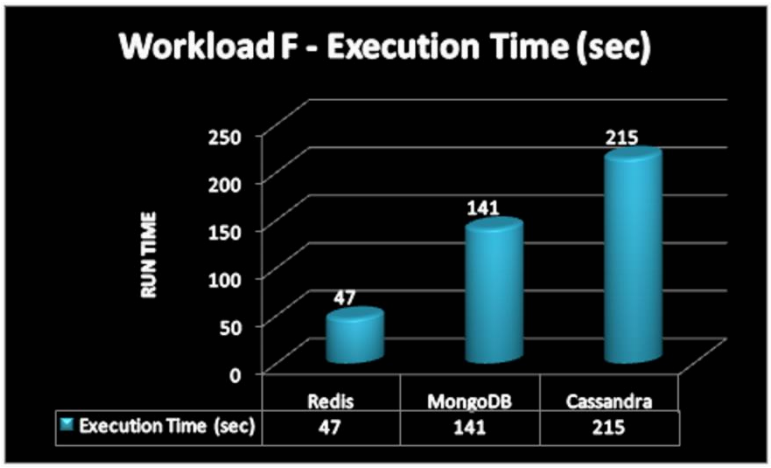

Fig.7. The execution time for the databases when running workload F

In this workload (Figure 7), 50% of records are read, modified, and the updates are written in the database permanently. In this workload, Redis takes the ascendancy over MongoDB and provides a run time 47 sec. We confirm the weak performance of Cassandra due to its lack of optimization to run read operations.

4.9 Overage Running Time Comparison

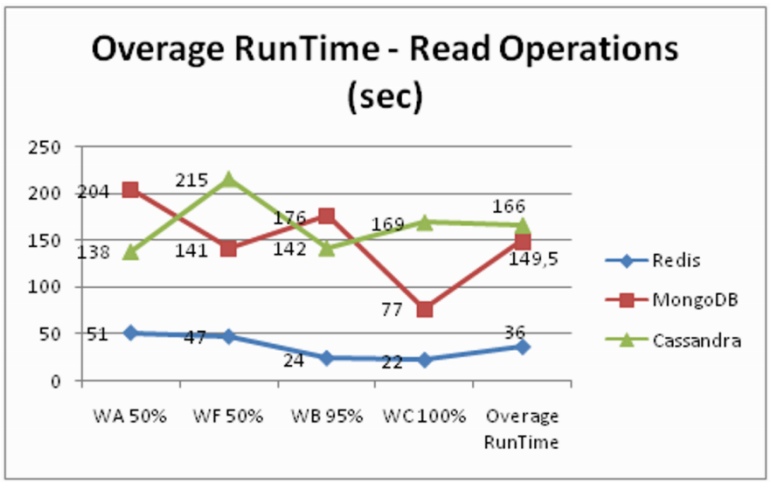

Fig.8. Evaluation Redis, MongoDB and Cassandra for Read operations

8

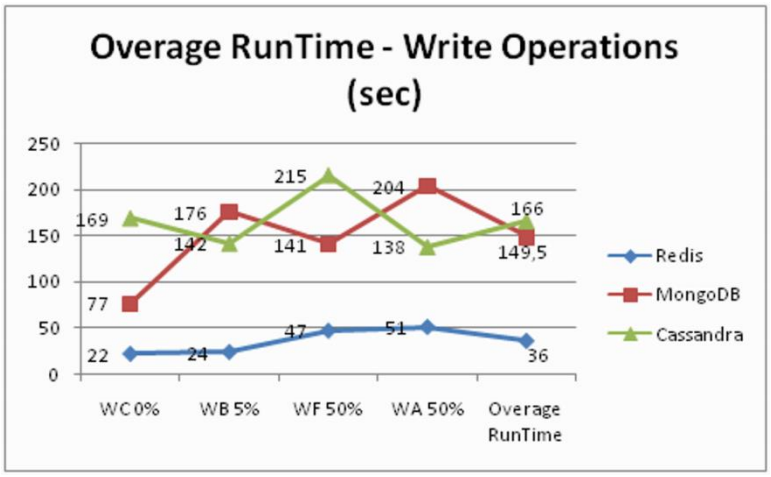

Fig.9. Evaluation Redis, MongoDB and Cassandra for Write operations

After execution of the four workloads, consisting of 100 000 combinations of read and update operations on 800 000 inserted records, we can see that as the proportion of read operations increase, the performance of the three NoSQL systems increases comparatively. And when the proportion of write operations increases, the performance of the three NoSQL systems decreases comparatively. Cassandra has highest average runtime with 166 sec. Redis and MongoDB performs almost in a similar manner. They deliver better performance than Cassandra. So we can deduce that for workloads with high read and update operations, we should choose Redis, it takes lowest average runtime with 36 sec.

5 Synthesis and Analysis of Results

After interpreting the results obtained by the three systems, Redis 3.0.504 as key-value store, MongoDB 4.4.0 as document Store and Cassandra 3.11.1 as Column store, and after execution of six workloads, consisting of 100 000 operations on 800 000 inserted records, 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 Redis, MongoDB and Cassandra, several lessons can be learned:

- In the read and write operations, Redis has the best performance of all. This is due mainly to the use of volatile memory, to store and retrieve data.
- MongoDB was more efficient in the read operations than Cassandra. This is due mainly to the registers mapping of MongoDB loaded into memory, which increases the reading performance;
- Cassandra presented more difficulty on reads. This is due mainly to its lack of optimization to run this kind of operations.
- Cassandra was more efficient in the update operations than MongoDB.

- 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.
- Cassandra was more efficient in the scan operations than Redis and MongoDB.
- MongoDB was more efficient in the scan operations than Redis.

6 Conclusion and Future Work

NoSQL popularity dealing with huge amounts of data, for storage and process, has increased substantially in the last decade, boosting the development of such systems. Different NoSQL databases types, provide different features each with its performance. Performance evaluation, for reference purposes, is relevant for deciding which database is more relevant according to applications necessities.

In this article, we have developed a performance comparative study between three very successful models currently, namely Redis as key-value store, MongoDB as Document Store model and Cassandra Column 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 100 000 varied operations of reading and writing have been executed several times in different days on databases containing 800 000 records initially loaded. After analysis and interpretation of the results, we found that Redis is overall more efficient compared to MongoDB and Cassandra in the read operations: Read Only, Read Mostly, Heavy Update, Read Latest and Read- modify-write, unlike Cassandra which confirmed its better performance in scan operations: Short Ranges. MongoDB also confirmed its better performance in insert operations: Data loading.

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.

As related future work research, we will pass to higher scales by increasing the number of operations performed to achieve workloads exceeding 200 000 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. Sahay, R. (2020). Relational Databases. Dans Microsoft Azure Architect Technologies Study Companion. Apress, Berkeley, CA.
2. Özsu, M. T., & Valduriez, P. (2020). NoSQL, NewSQL, and Polystores. Dans Principles of Distributed Database Systems. Springer, Cham.
3. Cloud Firestore. From https://firebase.google.com/products/firestore?gclid= EAIaIQobChMImP-nkx4yl9AIV9I9oCR3xNAa5EAAYASAAEgIaV_D_BwE&gclsrc=aw.ds
4. Barbierato, E., Gribaudo, M., & Iacono, M. (2014). Performance evaluation of nosql big-data applications using multi-formalism models. Future Generation Computer Systems , 37, 345–353.
5. Cattell, R. (2011). Scalable SQL and NoSQL Data Stores. ACM SIGMOD Record , 39 (4), 12-27.

6. Dede, E., Govindaraju, M., Gunter, D., Canon, S. R., & Ramakrishnan, L. (2013). Performance evaluation of a mongodb and hadoop platform for scientific data analysis. In Proceedings of the 4th ACM workshop on Scientific cloud computing (pp. 13-20). ACM.
7. Wang, L., Zhan, J., Luo, C., Zhu, Y., yang, Q., He, Y., et al. (2014). Bigdatabench: A big data benchmark suite from internet services. In High Performance Computer Architecture (HPCA), IEEE 20th International Symposium on (pp. 488–499). IEEE.
8. Ghazal, A., Rabl, T., Hu, M., Raab, F., Poess, M., Crolotte, A., et al. (2013). Jacobsen. Bigbench: towards an industry standard benchmark for big data analytics. In Proceedings of the 2013 ACM SIGMOD international conference on Management of data (pp. 1197–1208). ACM.
9. Vanderveen, S. J., Vanderwaaij, B., & Meijer, J. R. (2012). Sensor data storage performance: Sql or nosql, physical or virtual. In Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on (pp. 431–438). IEEE.
10. Rabl, T., Gomez-Villamor, S., Sadoghi, M., Muntès-Mulero, V., Jacobsen, A. H., & Mankovskii, S. (2012). Solving big data challenges for enterprise application performance management. Proceedings of the VLDB Endowment, 5(12), pp. 1724–1735.
11. E.Corporation. (2015, April). Benchmarking top nosql databases. https://www.datastax.com/wp-content/themes/datastax-2014- 08/files/NoSQL Benchmarks EndPoint.pdf
12. Aquel, J. M., Al-Sakran, A., & Hunaity, M. (2019). Comparative Study of NoSQL Databases. Biosci. Biotech. Res. Comm. Special Issue , 12 (1).
13. Matallah, H., Belalem, G., & Bouamrane, K. Evaluation of NoSQL Databases: MongoDB, Cassandra, HBase, Redis, Couchbase, OrientDB. International Journal of Software Science and Computational Intelligence , 12 (4), 71-91.
14. Gupta, S. (2018). Performance Evaluation of Unstructured PBRA for Bigdata with Cassandra and MongoDB in Cloud. International Journal of Cloud Applications and Computing , 8 (3), 48-59.