A Study over NoSQL Performance

Chapter · April 2019

DOI: 10.1007/978-3-030-16181-1_57

3 authors, including:

Maryam Abbasi
University of Coimbra
41 PUBLICATIONS **57** CITATIONS
SEE PROFILE

Filipe Alexandre Sá
Municipio de Penacova
32 PUBLICATIONS **339** CITATIONS
SEE PROFILE

A Study over NoSQL Performance

Pedro Martins[1], Maryam Abbasi[2], and Filipe Sá[1,3](✉)

[1] Department of Computer Sciences, Polytechnic Institute of Viseu, Viseu, Portugal
{pedromom,filipe.sa}@estgv.ipv.pt
[2] Department of Computer Sciences, University of Coimbra, Coimbra, Portugal
maryam@dei.uc.pt
[3] Câmara Municipal de Penacova, Penacova, Portugal

Abstract. Large amounts of data (BigData) are nowadays stored using NoSQL databases and typically stored and accessed using a key-value format. However, depending on the NoSQL database type, different performance is offered. Thus, in this paper, NoSQL database performance is evaluated and compared in aspects relating with, query performance, based on reads and updates.

In this paper, the five most popular NoSQL databases are tested and evaluated: MongoDB; Cassandra; HBase; OrientDB; Voldemort; Memcached and Redis. To assess the mentioned databases, are used, workloads represented by Yahoo! Cloud Serving Benchmark.

Results allow users to choose the NoSQL database that better fits application needs.

Keywords: NoSQL · Performance · YCSB · Database · Benchmark

1 Introduction

In the last few years, databases became present in all systems, small or big. Up to a certain point on data dimension and complexity, relational databases, alongside with the standard SQL language, were the main logical choice for enterprises. The essential operations were to allow storage, extraction, and manipulation of data. Although, as data volume grows, relational databases have several limitations which bring up limitations of storage and management, for instance: scalability; limited storage size; query efficiency related with data volume. To solve the problems mentioned above, new databases models emerged, known as NoSQL databases [8].

NoSQL databases appeared as a solution to complement relational databases and not to replace them. This complementing includes, performance increase by adding a set of new key characteristics, such as schema flexibility; horizontal scalability [15]; transparent clusters of nodes, that do not require manual data distribution or high maintenance; recover from faults; and automatic repair in case of faults [7]. This characteristics and others easy the task of database administration when faced with large data-sets.

Á. Rocha et al. (Eds.): WorldCIST'19 2019, AISC 930, pp. 603–611, 2019.
https://doi.org/10.1007/978-3-030-16181-1_57

Depending on what each system needs, different NoSQL engines provide different Consistency, Availability, and Partition Tolerance (CAP) feature combination [6]. For instance, SimpleDB, DynamoDB and MongoDB, have support for eventual consistency. With the objective of speeding up query execution NoSQL engines started to make use of in-memory storage, mapping parts of the database into volatile memory.

Despite non-relational databases evolution, for specific purposes/business, it is crucial to understand its main features and characteristics, as in relational systems. NoSQL databases have mechanisms to insert and retrieve data, oriented to performance optimization, resulting in different data load times, and execution times for both updates and read operations.

In this paper are tested seven NoSQL database engines which were considered to be the most popular during the related work review: Redis; Memcached; Voldemort; OrientDB; MongoDB; HBase; and Cassandra. All seven non-relational engines are tested regarding their execution speed upon different benchmark workloads, which provide execution parameters to get and put operations [4], that allow to understand and compare the performance of the tested databases (i.e., is the non-relational database faster for reads or writes/inserts).

In this paper, different features that impact different non-relation databases performance are studied and compared, allowing to understand which NoSQL database performs better facing specific workload scenarios.

This paper is organized as follows. Section 2, makes resumed review of the related work. Section 3, describes the applied experimental setup. Section 4, shows the experimental evaluation and results. Section 5, concludes the work and presents future research guidelines.

2 Related Work

NoSQL refers to an open database model which does not use SQL interface [12]. The origin can be related to Google's BigTable, used to store projects developed by Google, such as Google Earth [2]. Amazon, faced with the evolution, developed by Dynamo [5]. Over the last decades, these model of databases have been perfected, and their performance evaluated. A wide number of papers describing characteristics, proposing new features, business applications, and so on, regarding of NoSQL database engines [8,9,11].

Studies like [4], research performance characteristics, and mechanisms, to evaluate the advantages of NoSQL scalability over relational. The proposed work differs from these contributions by focusing on the comparison and analysis of performance execution time quantitatively.

The principle that NoSQL databases operate is on BASE (i.e., Basically Available, Soft State, and Eventually Consistent), which offers high availability, but low consistency [1,3,13]. On the other hand, ACID (i.e., Atomic, Consistent, Isolated, and Durable) is used to represent relational databases, in which transactions and committed data are always consistent. For both cases, ACID and BASE, come from CAP theorem (i.e., Consistency, Availability, and Partition

Tolerance) [14]. Following CAP theorem, only two out of three guarantees can be provided. If data consistency is crucial, relational databases should be used. If working with distributed data, consistency is more important and harder to achieve. NoSQL can be separated into four main categories, each oriented to different optimization's [10], such as:

- Key-value, where all data is stored as a pair of a key and a value, as happens with "hash tables". Data access is performed by searching for the corresponding key.
- Document, as in XML or JSON, data is stored in documents [16], where the format can variate.
- Column family, data is stored as a set of rows and columns. Columns are grouped according to relations, and often retrieved together.
- Graph, all data is represented as a graph with interlinked elements. For instance, social networks or maps.

In this research paper the testes and evaluated databases have the following type(s):

- Redis, Memcached, and Voldemort are key-value based.
- MongoDB and OrientDB, are document store based. Where OrientDB is an hybrid, also widely used for graphs.
- Cassandra and HBase, are column family based.

3 Experimental Setup

In order to evaluate the performance of the tested NoSQL databases, YCSB benchmark was used [4]. Two components set YCSB benchmark. First, a data generator, second, performance testing, i.e., read and insert operations. The tested workloads are defined by certain parameters that comprehend: percentage of reads; percentage or updates; percentage of scans; percentage of inserts; the number of operations; and the number of records. By default, there are predefined workloads, such as:

- Workload A, 50% reads and 50% updates;
- Workload B, 95% reads and 5% updates;
- Workload C, 100% reads;
- Workload D, 95% reads and 5% inserts;
- Workload E, 95% scan and 5% insert;
- Workload F, 50% reads and 50% read-modify-write;
- Workload H, 100% write;

In this paper, the main focus is to compare the execution speed of getting and put operations. For that purpose, workload A, C, and workload H, consisting of 100% write.

In order to perform the workload testes over the databases, 6.000.000 randomly generated records were used, each with ten fields of 100 bytes for the key registry. In total, around 1 kb per each record. While the number of stored

records was varying, 5.000 requests per second, to the testing database were being performed.

Other benchmarks such as TPC-H or SSB, could be used to test performance. However, this benchmarks, are more suitable for decision support. YCSB was chosen, due to its simplicity, and to test the two principal operations that are performed over NoSQL databases, get and put.

Tests were performed in a single node, with Ubuntu 18.10, i5 processor, 3.4 GHz, 16 GB memory, and 1 TB HDD.

4 Performance Evaluation

The next sections present the execution times, when using YCSB workloads A, C, H. Section 4.1 shows the performance evaluation using workload A. Section 4.2, uses workload C. Section 4.3, uses workload H. Finally in Sect. 4.4, the global performance evaluation of all tested databases are compared.

4.1 Workload A

Workload A, consists of 50% reads and 50% updates, on top of 6.000.000 records. Figure 1, shows the obtained results for each tested database.

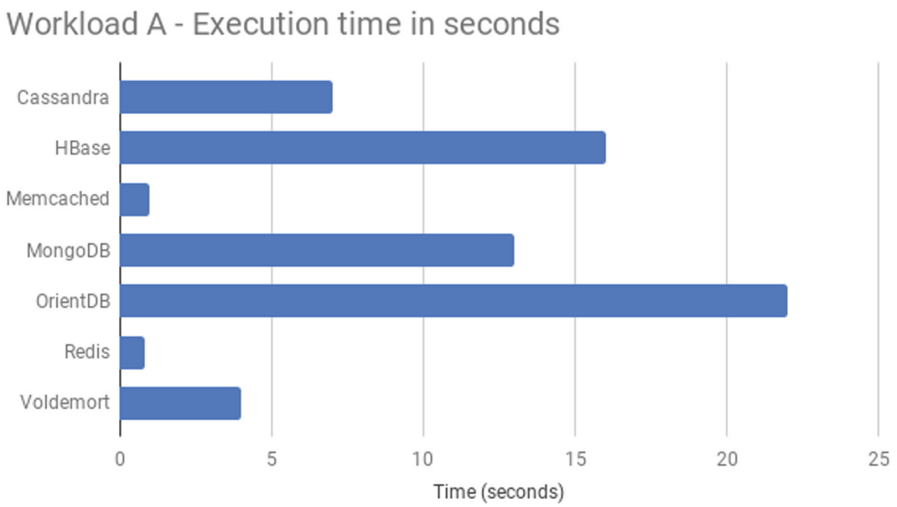

Fig. 1. Workload A

From Fig. 1, Redis and Memcached database, key-value store, achieves good performance. Redis and Memcached, both use volatile memory, to store and retrieve data, allowing this way lower execution times. Voldemort shows slower performance due to the combination of in-memory caching with the storage system.

Column-family databases, such as Cassandra, present better performance tan HBase. Overall, the worst performance was from the Document Store database, OrientDB, this happens because, in OrientDB, records must be read from disk.

4.2 Workload C

Workload C, consists of 5.000 random reads, on top of 6.000.000 records. Figure 2, shows the obtained results for each tested database.

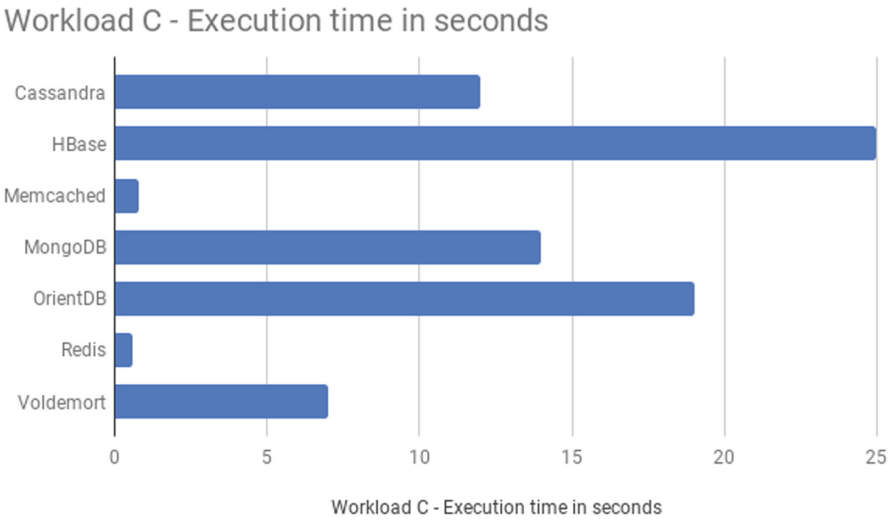

Fig. 2. Workload C

Concerning HBase and OrientDB, these have the lowest execution times when performing read operations. The reason is that different parts of the records are stored in different disk files, resulting in higher overheads. Nevertheless, HBase is optimized for updates as it will be shown later with workload H. OrientDB, stores data in the disk, and does not load it into memory; therefore it presents the second worst result. Since Redis and Memcached are designed to operate in volatile memory (but very fast), reading data is just a matter of key-value mapping in memory. Despite Voldemort being slower than Redis, and Memcached, it shows better performance than other tested databases.

4.3 Workload H

Workload H performance is shown in Fig. 3. This workload is based on 5.000 updates over 6.000.000 records.

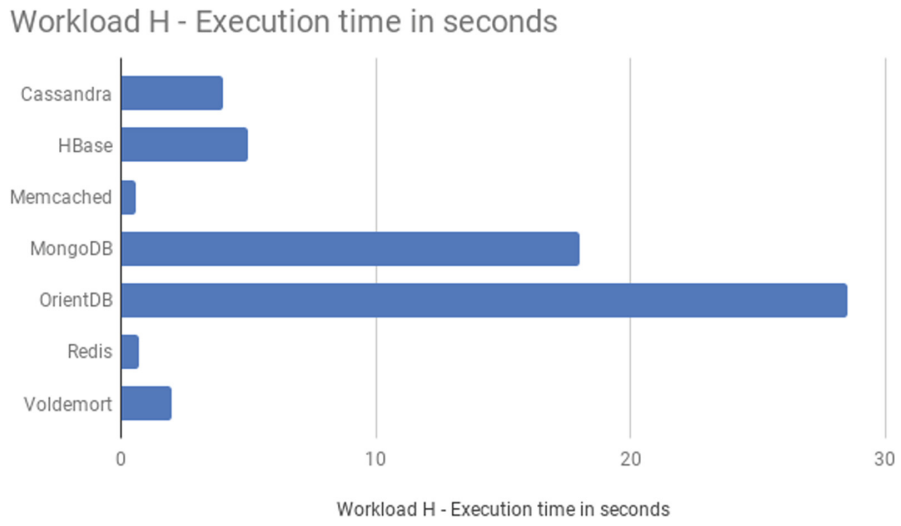

Fig. 3. Workload H

Workload H uses 5.000 write operations. This workload allows to verify the performance of the database when executing, read and update operations. Column databases, like Cassandra and HBase, are more optimized for updates, as they load large volumes of records into memory, the number of operations performed on disk is reduced. This performance is increased.

Document store database, OrientDB, shows the highest performance time (higher is worst), performing slower than MongoDB. 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, operate mainly in volatile memory. Records are directly mapped from memory; therefore, performance increases substantially.

4.4 Global Evaluation

This section, Fig. 4, shows how the tested databases handle a mix of all workloads at once.

In general, results show that the tested in-memory NoSQL database, Redis, Memcache, and Voldemort have the best performance of all. However, in order to assure in-memory performance, it sacrifices data consistency in case of failures.

Column-oriented databases, such as HBase and Cassandra, show good performance during update operations. Although, in general, they show themselves to be quite slow when compared with key-value approaches.

Document store databases, OrientDB, MongoDB, in general, comparing with all the rest, present the slowest performance.

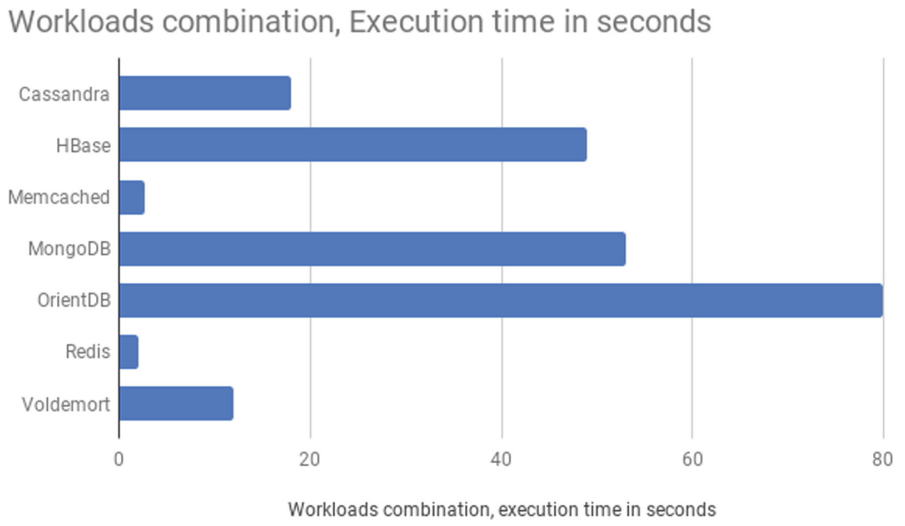

Fig. 4. Workload combination

5 Conclusions 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.

This paper provides a comparative study of different NoSQL databases performance. First, the most popular NoSQL databases, like, Cassandra and HBase, which are column-family based. Second, MongoDB and OrientDB, which are document-store based. Finally, Redis and Memcached, are a key-value in-memory database.

In order to test the performance of the database, YCSB benchmark was used with several workloads. Results show that the in-memory database, Redis and Memcached, are extremely fast, loading, updating and reading data. Nevertheless, a big limitation must be accounted, and the used memory is volatile, more expensive and relatively limited. Voldemort combines key-valued approach with in-memory and file-system for data persistence, however, it comes with a performance cost.

With the worst performance are the document store databases, OrientDB. The use of virtual machine environment, regarding computing resources, is more demanding, therefore implies larger overheads.

HBase and Cassandra, are optimized for sequential reading and writing, reducing the accesses to disk, as well as, number of operations by temporally storing records in memory.

MongoDB shows relatively good performance when dealing with large volumes of records. Performance issues occur from locking mechanisms to keep data consistent.

In conclusion, two groups of databases can be created. First, optimized for reading operations (MongoDB, Redis, Memcached, OrientDB, Voldemort). Second, optimized for updates/writes (Cassandra, HBase).

As related future work research, besides testing other non-relational databases, there will be applied parallelism mechanisms, to test performance in terms of response time, and scalability. More research directions related with this study, comprehend the testing of NoSQL over the cloud.

Acknowledgements. "This article is a result of the CityAction project CENTRO-01-0247-FEDER-017711, supported by Centro Portugal Regional Operational Program (CENTRO 2020), under the Portugal 2020 Partnership Agreement, through the European Regional Development Fund (ERDF), and also financed by national funds through FCT Fundação para a Ciência e Tecnologia, I.P., under the project UID/Multi/04016/2016. Furthermore, we would like to thank the Instituto Politécnico de Viseu for their support."

References

1. Carro, M.: NoSQL databases (2014). arXiv preprint arXiv:1401.2101
2. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: a distributed storage system for structured data. ACM Trans. Comput. Syst. (TOCS) **26**(2), 4 (2008)
3. Cook, J.D.: Acid versus base for database transactions. Cook, J.D., Blog (2009)
4. Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R.: Benchmarking cloud serving systems with YCSB. In Proceedings of the 1st ACM Symposium on Cloud Computing, pp. 143–154. ACM (2010)
5. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W.: Dynamo: amazon's highly available key-value store. In: ACM SIGOPS operating systems review, vol. 41, pp. 205–220. ACM (2007)
6. Elbushra, M.M., Lindström, J.: Eventual consistent databases: state of the art. Open J. Databases (OJDB) **1**(1), 26–41 (2014)
7. Gajendran, S.K.: A survey on NoSQL databases. University of Illinois (2012)
8. Han, J., Haihong, E., Le, G., Du, J.: Survey on NoSQL database. In: 2011 6th International Conference on Pervasive Computing and Applications (ICPCA), pp. 363–366. IEEE (2011)
9. Hecht, R., Jablonski, S.: NoSQL evaluation: a use case oriented survey. In: 2011 International Conference on Cloud and Service Computing (CSC), pp. 336–341. IEEE (2011)
10. Indrawan-Santiago, M.: Database research: are we at a crossroad? reflection on NoSQL. In: 2012 15th International Conference on Network-Based Information Systems (NBiS), pp. 45–51. IEEE (2012)
11. Leavitt, N.: Will NoSQL databases live up to their promise? Computer **43**(2), 12–14 (2010)

12. Moniruzzaman, A., Hossain, S.A.: NoSQL database: new era of databases for big data analytics-classification, characteristics and comparison (2013). arXiv preprint arXiv:1307.0191
13. Pritchett, D.: Base: an acid alternative. Queue **6**(3), 48–55 (2008)
14. Simon, S.: Brewer's Cap Theorem. CS341 Distributed Information Systems, University of Basel (HS2012) (2000)
15. Stonebraker, M.: SQL databases v. NoSQL databases. Commun. ACM **53**(4), 10–11 (2010)
16. Zhang, H., Tompa, F.W.: Querying xml documents by dynamic shredding. In: Proceedings of the 2004 ACM Symposium on Document Engineering, pp. 21–30. ACM (2004)