A Study of Performance NoSQL Databases

Mr. Ambrish Kishan[1], Ms. Rupali Wagh [2]
[1]PG Scholar, [2]Associate professor,
Department of Computer Science, Christ University, Bangalore, India

Manuscript History
Number: IJIRAE/RS/Vol.04/Issue04/APAE10092
Received: 25, March 2017
Final Correction: 10, April 2017
Final Accepted: 15, April 2017
Published: April 2017

Abstract--DBs belonging to NoSQL family store large volume of data in key-value format. The retrieval and storage NoSQL data stores are widely used to store and retrieve possibly large amounts of data, typically in a key-value format. There are numerous NoSQL types with different performances, and thus it is important to evaluate them in terms of performance and verify in ways the performance is related to the database kind. In this paper, we assess five most popular NoSQL DBs: Cassandra, HBase, Redis, OrientDB and MongoDB. We shall measure these DBs in terms of query performance, based on reads and updates, taking into consideration the typical workloads, as represented by the Yahoo! Cloud Serving Benchmark. This shall facilitate users to choose the most appropriate DB according to the specific mechanisms and their application requirements.

Key Words — Cassandra, HBase, MongoDB, OrientDB and Redis

I. INTRODUCTION

In today's era of Information Technology DBs forms a critical part of the organizations and are used all over the world. Traditional DBs uses standard SQL language for data storage, extraction and manipulation and were an optimal enterprise choice. But after constant expansion of stored data RDBMs shows several restrictions in terms of scalability, storage and subsequently their management. Hence the evolution of NoSQL DBs which can complement or solely take the burden of new needs of database management in today's IT world. NoSQL DBs are more elastic and horizontally scalable [2]. They can exploit presence of new clusters and nodes clearly, without anymore. NoSQL DBs are projected to automatically manage and distribute data, recover from faults and repair the whole system automatically [3]. Rightnow, there are over 150 NoSQL DBs with varied features and optimizations [1], and a number of NoSQL DBs provide all new features and advantages while keeping data consistent or even eventually consistent, depending on the system needs [4]. For e.g., MongoDB1, DynamoDB2 and SimpleDB3 wires strong and eventual consistency and CouchDB4 provides the feature of the eventual consistency. Also to increase execution speed of querying, volatile memory is being used. Since I/O data access is slower, associating database or its parts into volatile memory increases performance and reduces the overall execution time of querying. In order to decide an appropriate database, it is imperative to comprehend its main features. Each NoSQL DB provides different mechanisms to store and retrieve data, which directly translates to performance. Each of them has different optimizations, ensuing in different data loading time and execution times for reads or updates. The performed evaluation allows us to compare different types of NoSQL DBs, and test the execution times of read and update operations. We tested five popular NoSQL DBs: Cassandra, HBase, MongoDB, OrientDB and Redis and evaluate their execution speeds for different types of requests. During evaluation we used a benchmark with a typical range of workloads, Yahoo! Cloud Serving Benchmark [5], which provides execution of get and put operations, allowing bettering evaluating the performance of a specific database that whether it is quicker for reads or inserts. The analysis and comparison of the results allowed us to verify how the different features and optimizations influence the performance of these DBs.

II. NOSQL DBS

NoSQL DBs are based on BASE (Basically Available, Soft State, and Eventually Consistent) principle that is characterized by high availability of data, while sacrificing its consistency. On the contrary, relational DBs are represented by ACID (Atomic, Consistent, Isolated, and Durable) principle for consistency in state.

Both principles come from the CAP theorem - Consistency, Availability, and Partition Tolerance [6]. This theorem implies that only 2 of the 3 are guaranteed. So when the consistency of data is more critical, traditional RDBMs should be used. NoSQL DBs can be divided into 4 categories according to different optimizations [7]: Key-value store like in hash table. Document Store. i.e. DBs storing data in documents such as, XML [8] or JSON. Graph Database. i.e. DB that store data as inter-linked leaves for e.g., social networking, road maps or transport routes. Key-value Store DBs is more appropriate for the management of stocks and products, and data analysis in real time, due to the fact that these DBs have good retrieving speed – retrieving values given specific keys - when the greatest amount of data can be mapped into memory. Document Store databases are a good selection when working with large amounts of documents that can be stored into structured files, such as text documents, emails or XML and CMS and CRM systems. Column Family DBs should be used when the number of write operations exceeds reads, and this occurs, for e.g., during system logging. Finally, graph DBs are more suited for working with connected data, for e.g., to analyze social connections among a set of individuals, road maps and transport systems. End synopsis state that, NoSQL DBs are built to easily scale across a large number of servers (by sharding/horizontal partitioning of data items), and to be fault tolerant (through replication, write-ahead logging, and data repair mechanisms). Furthermore, NoSQL supports achieving high write throughput (by employing memory caches and append-only storage semantics), low read latencies (through caching and smart storage data models), and flexibility (with schema-less design and denormalization). In addition, the different systems offer different approaches to issues such as consistency, replication strategies, data types, and data models. The NoSQL DBs evaluated in this paper are from the following categories: Cassandra and HBase: Column Family• DBs. MongoDB and OrientDB: Document Store• DBs. Redis: Key-value Store database

III. EXPERIMENTAL SETUP

For experimental analysis we have employed YCSB - Yahoo! Cloud Serving Benchmark [5], for evaluation and comparison of performance of NoSQL DBs.

THIS BENCHMARK CONSISTS OF 2 COMPONENTS:

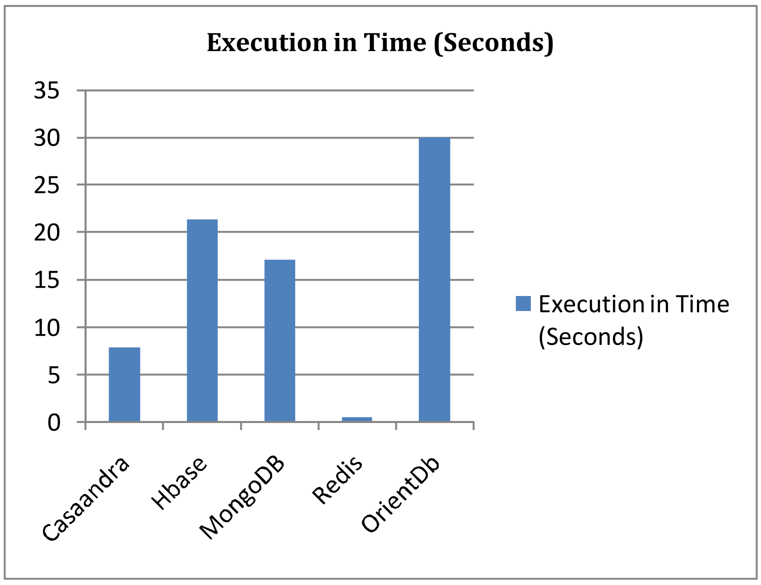
1) A DATA GENERATOR 2) A SET OF PERFORMANCE TESTS CONSISTING, IN A SIMPLISTIC WAY, OF READ AND INSERT OPERATIONS.

Each of the test scenarios is called workload and is defined by a set of features, including a percentage of read and update operations, total number of operations, and number of records used. The benchmark package provides a set of default workloads that may be executed and are defined by read, update, scan and insert percentages. Default workloads are: A (50% read and 50% update), B (95% read and 5% update), C (100% read), D (95% read and 5% insert), E (95% scan and 5% insert) and F (50% read and 50% read-modify-write). Our focus is on comparing execution speed of get and put operations, which are most used operations. Therefore, we only executed workloads A, C and an additional workload H, defined by us, which is 100% update. Table 1 shows the executed workloads and the respective operations. Table 1: Executed Workloads Workload % Read % Update A 50 50 C 100 0 H 0 100 In order to evaluate the DBs, we randomly generated 600,000 records, each with 10 fields of 100 bytes over the key registry identification, resulting in roughly 1kb total per record. The execution of workloads was made using 1000 operations, and this means that there were 1000 requests to the database under test, while varying the number of stored records and operations. There are other benchmarks available, such as, TPC-H or SSB, which could be used to evaluate database performance. All the tests were executed on a VM Ubuntu Server 32bit with 2GB RAM available, hosted on a computer with Windows 7 and a total of 4GB RAM. In this study, during the experimental evaluation, we tested the following NoSQL DBs, which are most used ones:

a) *Cassandra: Column Family database, version• 1.2.1 (http://cassandra.apache.org/).*
b) *HBase: Column Family database, version• 0.94.10 (http://hbase.apache.org/).*
c) *MongoDB: Document Store database, version• 2.4.6 (http://www.mongodb.org/). 1.*
d) *OrientDB: Document Store database, version 1.5 (http://www.orientdb.org/). Redis: Key-value Store database, version 2.6.14• (http://redis.io/). 4*

IV. EXPERIMENTAL EVALUATION

[In the following subsections we present and analyze the execution times based on only reads and only updates, and both operations at the same time. We executed YCSB workloads A, C and H. A Figure 1 show the results, in seconds, obtained while executing workload A that consists of 50% reads and 50% updates, over 600.000 records. Figure 1: Execution time of workload A (50% reads and 50% updates over 600.000 records) When analyzing the results of execution of workload A, a good performance is achieved by the Key-value Store database, Redis. This database highly uses volatile memory for data storage and retrieval, which allows lower execution time of requests. Among the tested DBs of Column Family type, Cassandra exhibited a performance of 7.89 seconds, 2.70 times faster than HBase. The worst performance was presented by the Document Store database OrientDB (30.09 seconds), with an execution time 1.75 times higher compared to another Document Store database MongoDB. The worst execution time of OrientDB is due to the fact that records have to be read from disk, which is much slower in comparison to the volatile memory. Evaluation over Workload C Figure 2 shows the results obtained while executing workload C that consists of execution of 1000 read operations over 600.000 records.



Evaluation over Workload A

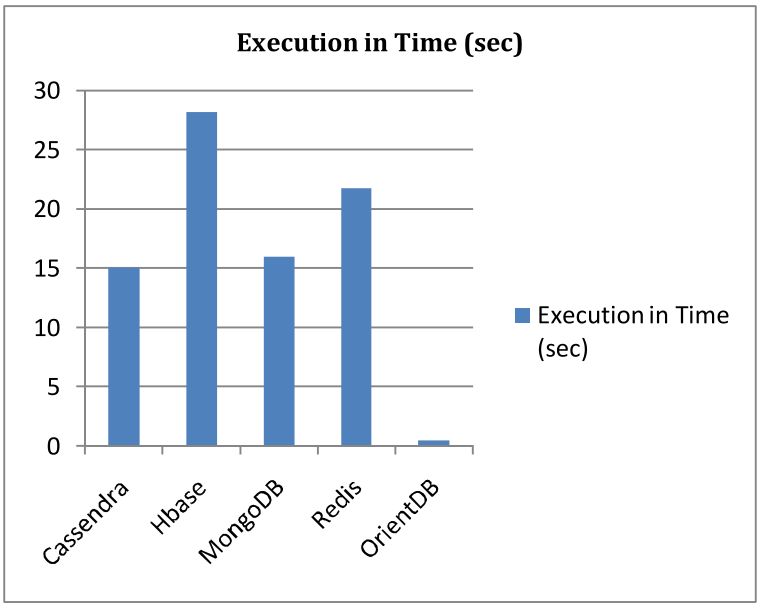
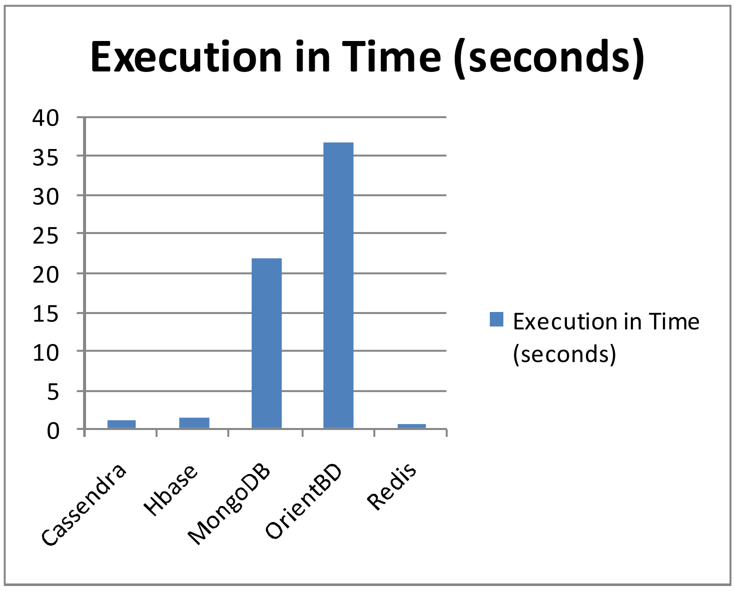


Figure 2: Execution time of workload C (100% reads over 600.000 records)

The results of the execution of workload C indicate that the Document Store DBs, HBase and OrientDB, showed the slow execution time during read operations. HBase presented the worst result, and it is 1.86 times lower compared to the Column Family database Cassandra. Given a large number of records, HBase showed more difficulty during execution of reads. In HBase, parts of the same record may be stored in different disk files, and this results in an increased execution time. HBase is optimized for the execution of updates, but for reads, as we will see later on, HBbase shows a good performance over the workload H with 100% updates. The second worst outcome was shown by OrientDB, which stores data records in disk and does not load data into memory. Redis had good execution time: it kept records in memory and thus showed minimal execution time for read operations. Database Veronika Abramova, Jorge Bernardino, Pedro Furtado: Which NoSQL Database? A Performance Evaluation 21 Redis is projected to fast record retrieval using key due to mapping data in memory. 4.3 Evaluation over Workload H Figure 3 shows the results of the execution of the workload H, which is 1000 updates over 600.000 records. Figure 3: Execution time of workload H (100% update over 600.000 records) With the execution of the workload H with 1000 updates, we observed better optimization of some DBs for the execution of writes (in a simplistic way one update is one write). Two Column family DBs, Cassandra and HBase, are optimized for performing updates: they load records as much as possible into memory, and thus the number of operations performed on the disk is reduced and the performance is increased. Among the evaluated Document Store DBs, OrientDB had the highest execution time with a total of 36.75 seconds, thus having a performance 1.69 times lower compared to the performance shown by MongoDB.



The reason for this difference in execution time is the distinction of the storage type used by these DBs: The OrientDB keeps records on disk rather than loading data into memory. The poor results of MongoDB are due to the use of locking mechanisms to perform update operations, and this increases execution time. Key-value Store Databases are in-memory DBs: they use volatile memory to map records, and thus database performance is increased significantly. 4.4 Overall Evaluation Over previous subsections we presented results obtained over different workloads and data loading.

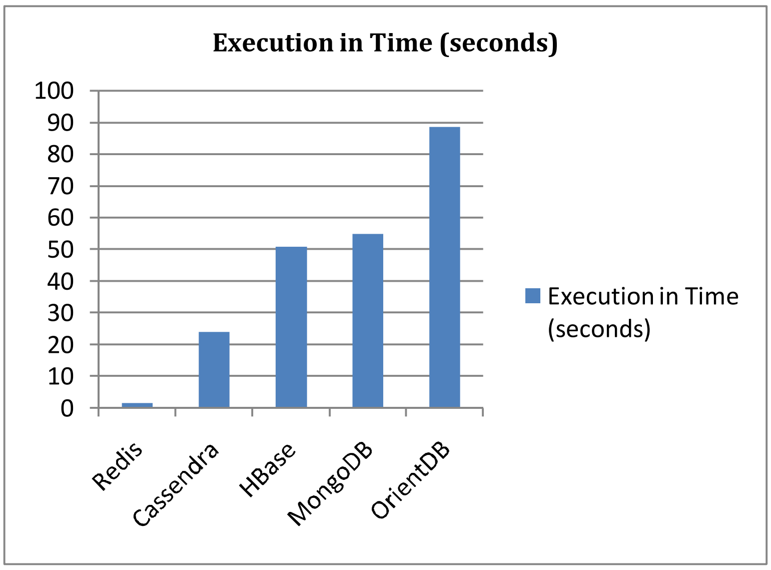


Figure 4: Overall execution time of workloads A+C+H

In order to show more clearly the overall performance of these evaluated DBs regardless of the type of performed operations, Figure 4 is generated. This figure shows the total execution time, values in seconds, for each of the tested DBs. These values were obtained by summing the execution times of all workloads (A + C + H), and sorted in ascending order, from lowest execution time to highest. The overall results show that the in-memory database, Redis, had the best performance. Redis is a Key-value Store database and is highly optimized for performing get and put operations due to mapping data into RAM. It is well-known that in-memory DBs are more efficient in query processing, but quantitative accuracy still lacks. One of our contributions in this study is presenting the quantitative results of the execution speed of the in-memory NoSQL database. Cassandra and HBase, as Column Family DBs, showed good update performance, since they are optimized for update operations. Nevertheless, from the overall evaluation results, those DBs were more than 15 times slower than the Key-value Store database, Redis. Finally, Document Store DBs had the worst execution times, and OrientDB is the database with the lowest overall performance. OrientDB was 1.61 times slower than MongoDB and had 58.32 times lower performance in comparison with Redis.

IV. CONCLUSIONS

NOSQL DBs bring a various advantages, compared to RDBMs, for huge volumes of data, that are non structured or semi structured. There are different types of NoSQL DBs and each possess own set of features and characteristics, and these lead to the performance difference. The performance is an important factor for deciding the database to be used for enterprises and applications. Hence, it is essential to compare and analyze the execution time of different NoSQL DBs, and provide a performance reference. Here, we assessed 5 most popular NOSQLl DBs from three types: Cassandra and HBase from Column Family DBs, MongoDB and OrientDB from Document Store DBs, and Redis from Key-value Store database. We use Yahoo! Cloud Serving Benchmark [5], and compare the execution times of these NoSQL DBs over different types of workloads. Apart from the experimental evaluation, we also analyze the performance differences from the optimization mechanisms and data store approaches used by these DBs. The DBs, which load data into volatile memory, like Redis, showed tremendously fast response times regardless of workloads, due to the fast speeds of volatile memory compared to the extraction of the files stored on the hard drive. However, such DBs depend on the amount of volatile memory, which is a much more expensive storage type compared to the disk. The database with worst performance was the Document Store database, OrientDB, over different workloads. By analyzing obtained results we discover that this database requires more system capacities compared with the capacities provided by the environment used in the evaluation. Consequently, their performances were limited by the memory management, the operating system and the use of virtual machine environment. HBase and Cassandra are DBs that use a log for storing all performed modifications; meanwhile the records are stored in memory for subsequent disk flush. The use of these mechanisms and following sequential writing to disk reduces the amount of disk operations that are characterized by low speed compared to the speed of the volatile memory. So, these DBs are especially optimized for performing updates, while reads are more time consuming when compared with in-memory DBs. MongoDB is the database that showed largest increase in the execution time directly related to the increase of the number of updates performed. This database uses locking mechanisms, which increase execution time. On the other hand, the reads are not exclusive, so the mapping of records in memory increases performance. OrientDB performance also degraded with the increasing number of update operations. As an overall analysis, in terms of optimization, NoSQL DBs can be divided into two categories, the DBs optimized for reads and the DBs optimized for updates. Thus, MongoDB, Redis, and OrientDB are DBs optimized to perform read operations, while Colum Family DBs, Cassandra and HBase, and have a better performance during execution of updates. As future work, we will compare and analyze the performance of NoSQL DBs further: we will increase the number of operations performed and run NoSQL DBs over multiple servers.

V. FUTURE SCOPE

None of graph database was analyzed due to limitation of benchmark in this work; hence its evaluation only can fulfill the true objective of this paper.

REFERENCES

[1]. NoSQL - http://nosql-database.org/.
[2]. Stonebraker, M.: SQL DBs vs. NoSQL DBs. Communications of the ACM, 53(4): 10-11, 2010.
[3]. Gajendran, S.: A Survey on NoSQL DBs, http://ping.sg/story/A-Survey-on-NoSQLDBs---Department-of-Computer-Science, 2012.
[4]. ELBUSHRA, M. M. AND LINDSTRÖM, J.: EVENTUAL CONSISTENT DBS: STATE OF THE ART. OPEN JOURNAL OF DBS (OJDB), RONPUB, 1(1): 26- 41, 2014. ONLINE: HTTP://WWW.RONPUB.COM/ PUBLICATIONS/OJDB-V1I1N03_ELBUSHRA.PDF.
[5]. Cooper, B., Silberstein, A., Tam, E., Ramakrishnan, R., and Sears, R.: Benchmarking cloud serving systems with YCSB. In Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC '10). ACM, New York, NY, USA, 143-154, 2010.
[6]. Browne, J.: Brewer's CAP Theorem, http://www.julianbrowne.com/article/viewer/brew ers-cap-theorem, 2009.
[7]. Indrawan-Santiago, M.: Database Research: Are We at a Crossroad? Reflection on NoSQL. Network-Based Information Systems (NBiS), 15th International Conference on Network-Based Information Systems, pp.45-51, 2012.
[8]. Crockford, D.: JavaScript: The Good Parts. Sebastopol, CA: O'Reilly Media, 2008. [16] Armstrong, T., Ponnekanti, V., Dhruba, B., and Callaghan, M.: LinkBench: a database benchmark based on the Facebook social graph. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13). ACM, New York, NY, USA, 1185-1196, 2013.