2016 7th International Conference on Cloud Computing and Big Data

Performance Comparison between Five NoSQL Databases

EnqingTang
Tsinghua National Laboratory for Information Science and Technology, Department of Automation
Tsinghua University
Beijing 100084, China
teq14@mails.tsinghua.edu.cn

Yushun Fan*
Tsinghua National Laboratory for Information Science and Technology, Department of Automation
Tsinghua University
Beijing 100084, China
fanyus@tsinghua.edu.cn

***Abstract*—Recently NoSQL databases and their related technologies are developing rapidly and are widely applied in many scenarios with their BASE (Basic Availability, Soft state, Eventual consistency) features. At present, there are more than 225 kinds of NoSQL databases. However, the overwhelming amount and constantly updated versions of databases make it challenging for people to compare their performance and choose an appropriate one. This paper is trying to evaluate the performance of five NoSQL clusters (Redis, MongoDB, Couchbase, Cassandra, HBase) by using a measurement tool – YCSB (Yahoo! Cloud Serving Benchmark), explain the experimental results by analyzing each database's data model and mechanism, and provide advice to NoSQL developers and users.**

Keywords—NoSQL database; performance comparison; YCSB; NoSQL cluster;

I. INTRODUCTION

Relational databases represented by MySQL and Oracle database have played an important role in data storage, data query and data management in the last 40 years of development [1]. However, with the advent of the Internet, mobile Internet and big data era, certain limitations have emerged in traditional relational databases: query performance becomes poor with huge amounts of data; It is difficult to get horizontal scalability due to relational data; Furthermore, limited scalability will lead to dramatic increase of management cost [2]. The NoSQL databases appear just to make up for the drawbacks mentioned above. Compared with relational databases, unstructured or non-strict data structure of NoSQL makes the database more flexible with a natural level of scalability; It is easy to expand nodes without increase of management costs and complexity; Database performance can be greatly improved by adding numbers of machines to an existing cluster [3].

At the beginning of its appearance, the biggest shortcoming of NoSQL is weak consistency, but with the popularity and development of NoSQL technology, many databases have added new features and performance optimization with the updated version of the iteration. Many NoSQL databases, such as MongoDB, DynamoDB and SimpleDB 3, already have both strong consistency and eventual consistency, and users are able to change the configuration depending on the application [4].

Although NoSQL technology has been rapidly developed, the comparison between their performance is not clear. Nowadays there are more than 225 NoSQL databases according to the real-time statistics in [5]; Moreover, different NoSQL databases have different implementation mechanisms, storage characteristics, configurations and optimization methods, which brings more challenges in NoSQL selection. In this paper we try to test NoSQL performance to give suggestion to anyone who need to select NoSQL in different application scenarios.

We selected five NoSQL databases (Redis, MongoDB, Couchbase, Cassandra, HBase) based on their popularity and database type, and created a 4-node cluster for each database in experimental evaluation. YCSB (Yahoo! Cloud Serving Benchmark), an open-source NoSQL benchmarking system and program suite for evaluating relative performance of NoSQL databases, was used in our experiment [5, 6]. We can design different operating ratio (insert, read and update operations) and different data loads depending on the application scenes to compare the performance of NoSQL databases. Execution time and throughput of every test are the main performance respects we focus on in this paper.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 introduces NoSQL databases. The experimental setup is in Section 4 followed by section 5 with benchmarking result and analysis. Finally, we conclude and explain our future work.

II. RELATED WORK

NoSQL prototype can be traced back to Google's BigTable model which is developed by Google and is used by a number of Google applications, such as Google Maps, Google Book Search and Gmail [7,8]. Hbase and Cassandra are based on Google BigTable model while Voldemort is an open source implementation of Dynamo which is designed independently by Amazon [23, 25]. With the development of technology over the last a few years, there is much research on performance comparison: [9] described and compared various NoSQL databases from the aspects of data structure, consistency, and other extensible framework in theoretical level; [2] compared execution time of 10 kinds of NoSQL databases in stand-alone mode and analyzed the results in detail; [10] analyzed Cassandra, HBase and MySQL from theoretical and

*Corresponding Author

978-1-5090-3555-7/16 $31.00 © 2016 IEEE
DOI 10.1109/CCBD.2016.69

experimental aspects; [3] evaluated three NoSQL databases (MongoDB, HBase, Cassandra) with different configurations (CPU nuclear, the number of clusters, replication) on Amazon EC2 platform.

Unlike these efforts, we focus on the performance of different NoSQL databases in distributed mode. The execution time of different workloads and throughput of various record sizes are the main comparative indicators; In addition, as the versions of NoSQL are constantly updated, the performance of some databases are optimized quickly, while the others are growing relatively slowly, which leads to the result that the performance evaluation may be not the same as before. Because of this, we try to compare the latest stable version of each NoSQL, and the experimental results can be more instructive and useful for NoSQL selection.

III. NOSQL DATABASES

NoSQL is on the basis of the BASE principles (Basically Available, Soft State, and Eventually Consistent), which sacrifices strong consistency in exchange for high availability [11]. There are several classifications of NoSQL while a more wide-accepted way is to divide NoSQL into the following four categories [2, 24]:

- Key-value Store: data is stored in the form of key-value pairs, which is called hash table where value can be got quickly by using key.
- Document Store: As the name suggests, it is designed for storing document or semi-structured information, and data is stored and managed in document style like XML [12] or JSON [13], which can be used in a wider range of applications [22].
- Column Family Store: data is stored as rows and columns while the similar columns are stored together in which is called column family. It is easier in expansion and distribution and more suitable for storing large data.
- Graph Database: it is used to store data which is similar to the graph structure data, such as data in social networking or recommendation systems.

The key-value databases usually store data in memory, so they have a high speed in data process which can be applied to high-load cache access; Document databases are more suitable for a large number of applications where developers need to store document-type data especially in the Internet web applications; Column family store can be used to store large amounts of structured and unstructured data, which is more suitable in scenes where write operation is more frequent than the read operation, such as logging system; Graph databases are more appropriate for connected data and we can use mature graph algorithms directly to analyze and calculate data in the graph databases.

We did not choose Graph database in our experiments because the assessment indicators and use cases (CRUD) of graph databases are different from the other three types of databases. We need other benchmark tools to evaluate graph databases, e.g. XGDBench [2, 14].

The five databases selected for performance evaluation in this article are:

Redis [15]: Key-value store, version 3.2.1.

MongoDB [16]: Document Store, version 3.2.7

Couchbase [17]: Document Store, version 4.0.0

Cassandra [18]: Column family Store, version 2.2.7

HBase [19]: Column family Store, version 0.98.20

IV. EXPERIMRNTAL SETUP

We used the YCSB (Yahoo! Cloud Serving Benchmark) developed by Yahoo as our benchmarking framework. The YCSB is a workload generator and provides tools for apples-to-apples comparison of different data stores, and the evaluation is mainly from aspects such as performance, elasticity, availability and replication [20, 21]. It operates in two phases: data initialization (loading) and test execution phase (transaction). Users can define each test workload by setting the read and write ratio and changing the number of records and the operation count.

In initialization phase, data is loaded onto the database while each record consists of 10 columns and each column is 100 bytes, nearly 1kb total per record. These data are randomly generated and are uniquely identified by a key which is a combination of string “user” and some other digits [2]. In the execution phase, YCSB provides a package of standard workloads which define the proportion of different operations and the operation amount. Our experiments mainly involve the following three workloads:

TABLE I. YCSB WORKLOAD

Workload	Operations
Workload A	Update Heavy. 50/50 of read/update
Workload C	Read Only. 100% read operation
Workload H	Update Only. 100% update operation

In order to compare the performance of the five NoSQL databases, we carried out two experiments :

- Experiment 1: We fixed the record size and operation count while changing the workload type in every test. To be specific, we loaded 100,000 records to the database and the number of operations performed was set to 10,000 in every test, and we executed workload A, C and H one by one for five NoSQL databases to get the execution time of each NoSQL. This experiment is in order to compare execution speed of the database under certain conditions.
- Experiment 2: We analyzed throughput in executing workload A while altering the record size and operands of every test, and then explained every result. This experiment is aiming at comparing storage expansion tolerance of every database.

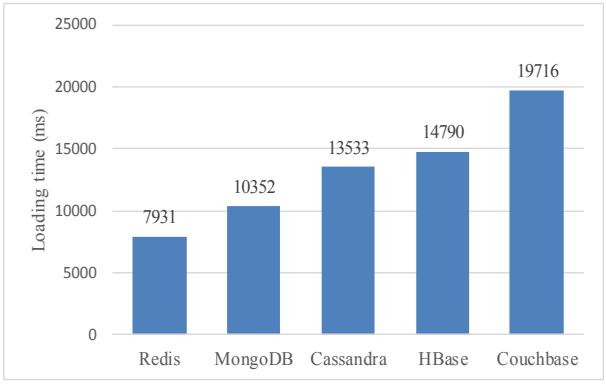

Fig. 1. Loading time for 100,000 records with five databases.

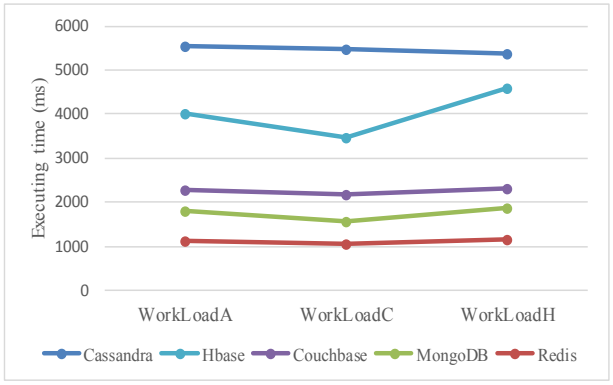

Fig. 2. Executing time for 100,000 records with five databases.

All tests were executed in 5 virtual machines and each machine is Ubuntu server with 12GB RAM, 256GB HDD and quad-core CPUs available. We set up a 4-node cluster for each NoSQL database in four servers and installed YCSB in the fifth server in order not to affect performance experiments. The experimental results will be presented in the next section

V. Results Analysis

A. Experiment 1

In accordance with the YCSB standard process, firstly, we need to load 100,000 records into database and the database will automatically store data according to the storage configuration; Next, we will execute different workloads to test the performance. As it is difficult for us to focus on various aspects of NoSQL performance, in this experiment we only care the total time when loading data and executing workloads to compare the speed and efficiency of execution workloads of different databases. The following are the results and analysis of relevant experiments.

1) Data loading

We present the loading time of five different NoSQL databases when inserting 100,000 records in Figure 1. We sort the databases from fast to slow according to the loading speed. Redis got the best performance of inserting operation among all databases with a loading time of nearly 8 seconds which is 1.31 times faster than MongoDB. The wonderful performance of Redis can be attributed to its semi-persistent mode, which is that all the data is stored in memory and then asynchronously saved to disk on a regular basis for perpetual storage. In our experiment, Redis was set to call fsync function to refresh AOF files every second, which results in the perpetual storage of data. Redis can also be set to other modes according to your applications, such as In-Memory mode, Fully Persistent mode. Obviously, different modes have different features.

The two column-family databases, Cassandra and HBase, were 1.71 times and 1.86 times slower than Redis. There are some write optimizations in these two databases. The update operations will return success if the updates are written to a log file, then cached in memory, which makes the insertion process faster. The worst performance was presented by Couchbase, which was 2.49 times slower than Redis. Couchbase is a combination of two open source NoSQL databases, Couchdb (a document store) and Membase (a key-value database). Before loading the data, you need to build buckets for Couchbase where documents with intrinsic unique id are stored, and configure the resources like RAM and disk quota that your bucket will allocate on per node. The buckets will be divided into vBuckets which will be mapped to a certain machine. It is important to optimize hardware resources and configuration in real application if developers want to get a good performance of Couchbase.

2) Workloads execution

Figure 2 shows execution time of 5 tested databases when executing three different workloads. Workload A is 50% reads and 50% updates operation and workload C is 100% reads, in addition, workload H is 100% updates. Overall, Redis showed the best performance with the average execution time of 1.1 seconds, which was 1.56 times better than MongoDB, 2.02 times faster than Couchbase. Column-family databases, HBase and Cassandra were the slowest, 3.61 times and 4.90 times slower than Redis respectively. Document databases, MongoDB and Couchbase presented good performance in the workloads execution. MongoDB supports asynchronous read and write operations which greatly speeds up the execution process. Couchbase implements object managed cache which is a distributed hash table, and provides memory storage architecture which intelligently keeps accessed documents and indexes in RAM. We can quickly locate data and efficiently execute CRUD operations by document key in-memory, which reduces the read and write latency.

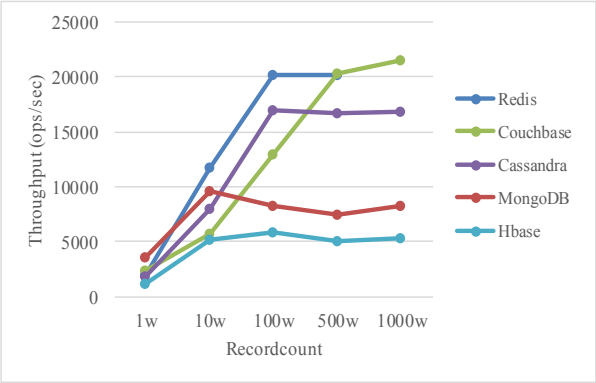

Fig. 3. Throughput of loading data with increasing recordcount.

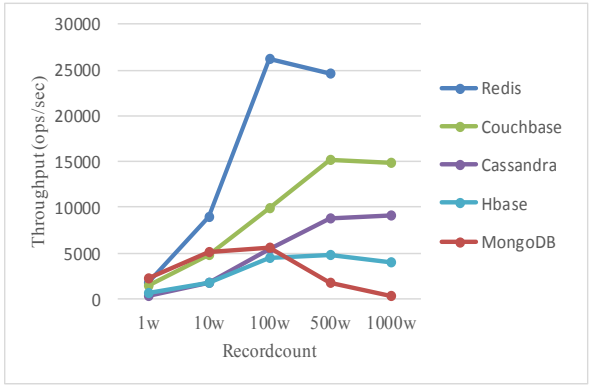

Fig. 4. Throughput of executing workload A with increasing recordcount.

In workload C test, although HBase and Cassandra still performed the slowest speed, HBase was 1.58 times faster than Cassandra, which is because the new version of HBase has realized a certain degree of optimization in read operation. Part of memory on the RegionServer (an important role in HBase cluster) is used to support BlockCache policy which uses LRU strategy to cache the latest read results and eliminate the oldest data. This mechanism is enabled by default and any read operations will be loaded to the LRU cache, which greatly improved the efficiency of the read operation in HBase. There is an inevitable question for HBase users that how much memory should be available for caching in order to get better performance. We used the default value in our experiments, which can be optimized in real applications.

B. *Experiment 2*

To compare performance with the increase in the number of record and operation, we respectively set the record size to 1w, 10w, 100w, 500w, 1000w, while the operation count was 10% of the record amount in this experiment. We only executed workload A (50% reads and 50% updates) and focused on the throughput performance in loading and execution phase. The following are the results and analysis of relevant experiments.

1) Data loading

The throughput curves of Five databases with loaded data increasing are considered in Figure 3. It is evident that the five curves have the same trend where the throughput increases firstly and goes to a relatively stable condition as the record count increases. At the rising phase, Redis was increasing at the fastest pace to over 2w ops / sec throughput finally, which was mainly because the data was stored in memory rather than disk. The memory storage mechanism was also the limitation of Redis at the same time. When the record size reached 1000w, Redis failed to handle the load requests since it had only 12GB of memory available, so that the curve of Redis had only 4 points while the others had five. Although Redis developer groups are working on the optimization to try to store data in a more memory efficient way, Redis is still the last choice when we need to store big data for cost reasons.

In addition, we can also observe that the throughput of Cassandra and Couchbase grew more rapidly compared to HBase and MongoDB. Cassandra and Couchbase have a similar architecture characteristic that all the nodes in the cluster make up a peer-to-peer distributed system, which means there is no independent management node or master in the cluster. Each node exchanges information across the cluster so that each node can know where to find the data and what the status of the other nodes are in the cluster, which means any node can be the role of routers when the requests come. Each machine not only provides data query functions but also data management, which greatly improve its efficiency.

2) Workload A execution

In Figure 4, we present the throughput performance of five databases when executing workload A as the record size increases. It turned out that Redis still showed the best throughput performance when the record was under 1000w. In addition, it was 3.42 times better than HBase, 1.55 times better than Couchbase. Moreover, it is easy to find that Couchbase, Cassandra and HBase presented a similar trend as the loading stage (insertion process) with the description above.

It is obvious to find that throughput and efficiency of MongoDB decreased when the record exceeded a certain count limit (100w). Through the analysis, we found that MongoDB is using master-slave mode and sharding is used for distributing data across multiple servers. When creating a mongo cluster where shard collection was used, we need to set and optimize sharding key according to the actual data. The sharding key is not only important to MongoDB's horizontal scalability ,but also decides data distribution in clusters, which is the key to support deployments with very large data sets and high throughput operations. In our experiment, we did not do any optimizations for a particular database and did not optimize shard key for MongoDB, which may be the reason of performance reduction of MongoDB, and we intend to do further exploration in our future work.

VI. Conclusion and Future Work

Since NoSQL technologies are developing quickly, performance comparison of NoSQL is necessary and useful for

NoSQL developers and users. In this paper, we have got some valuable and instructive conclusions by comparing five No-SQL databases in a 4-node cluster.

The conclusions are that Redis is particularly well suited for loading and executing workloads. Although it shows the best efficiency, Redis also has its limitation when in the face of extremely large data; Document databases, followed by column-family databases, have a good average performance since they own both efficiency and Scalability. At the same time, we also found that database framework in master-master mode, where each node is equivalent, has more advantages over those master-slave architectures. Specifically, the master-master framework is less likely to encounter single point block when a large number of concurrent operations are coming, and it is easy for horizontal scalability when stored data is growing rapidly.

It is important to say that the comparison results of execution time and throughput under four-node cluster condition are not a comprehensive evaluation of NoSQL databases. In addition, in some specific applications, we often need to optimize NoSQL configuration firstly according to the actual needs, and then compare performance after optimization, which is more commonsensible and more significant in NoSQL selection. Therefore, how to select NoSQL in a specific application will be one of the highlights of our future research work.

More specifically, in the future work, we will continue to study the following issues : how to balance efficiency and cost in NoSQL selection; comparison NoSQL in other aspects, such as operating delay, horizontal scaling efficiency, etc. We also plan to make research on how to do optimization firstly and then compare and select NoSQL in a specific application.

ACKNOWLEDGMENT

This research has been supported by the National Natural Science Foundation of China (No. 61174169).

REFERENCES

[1] Abramova, Veronika, Jorge Bernardino, and Pedro Furtado. "Which nosql database? a performance overview." Open Journal of Databases (OJDB) 1.2 (2014): 17-24.

[2] Abramova, Veronika, Jorge Bernardino, and Pedro Furtado. "Experimental evaluation of NoSQL databases." International Journal of Database Management Systems 6.3 (2014): 1.

[3] Gandini, Andrea, et al. "Performance evaluation of NoSQL databases."European Workshop on Performance Engineering. Springer International Publishing, 2014.

[4] Elbushra, Mawahib Musa, and Jan Lindström. "Eventual consistent databases: State of the art." Open Journal of Databases (OJDB) 1.1 (2014): 26-41.

[5] "NoSQL." Internet: http://nosql-database.org, [15.08.16]

[6] "YCSB." Internet: https://en.wikipedia.org/wiki/YCSB, [15.08.16]

[7] Chang, Fay, et al. "Bigtable: A distributed storage system for structured data." ACM Transactions on Computer Systems (TOCS) 26.2 (2008): 4.

[8] "BigTable." Internet: https://en.wikipedia.org/wiki/Bigtable, [15.08.16]

[9] Hecht, Robin, and Stefan Jablonski. "Nosql evaluation." International conference on cloud and service computing. IEEE, 2011.

[10] Tudorica, Bogdan George, and Cristian Bucur. "A comparison between several NoSQL databases with comments and notes." 2011 RoEduNet International Conference 10th Edition: Networking in Education and Research. IEEE, 2011.

[11] Cook, John D. "ACID versus BASE for database transactions, 2009." URL http://www. johndcook. com/blog/2009/07/06/brewer-cap-theorem-base.

[12] Crockford, Douglas. JavaScript: The Good Parts: The Good Parts. " O'Reilly Media, Inc.", 2008.

[13] Carro, Massimo. "NoSQL Databases." arXiv preprint arXiv:1401.2101 (2014).

[14] Armstrong, Timothy G., et al. "LinkBench: a database benchmark based on the Facebook social graph." Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. ACM, 2013.

[15] "Redis." Internet: http://redis.io, [15.08.16]

[16] "MongoDB." Internet: http://www.mongodb.org, [15.08.16]

[17] "Couchbase." Internet: http://www.couchbase.com, [15.08.16]

[18] "Cassandra." Internet: http://cassandra.apache.org, [15.08.16]

[19] "HBase." Internet: http://hbase.apache.org, [15.08.16]

[20] Cooper, Brian F., et al. "Benchmarking cloud serving systems with YCSB."Proceedings of the 1st ACM symposium on Cloud computing. ACM, 2010.

[21] Barata, Melyssa, Jorge Bernardino, and Pedro Furtado. "YCSB and TPC-H: Big Data and Decision Support Benchmarks." 2014 IEEE International Congress on Big Data. IEEE, 2014.

[22] "Document Store." Internet: https://en.wikipedia.org/wiki/Document-oriented_database, [15.08.16]

[23] DeCandia, Giuseppe, et al. "Dynamo: amazon's highly available key-value store." ACM SIGOPS Operating Systems Review 41.6 (2007): 205-220.

[24] Orend, Kai. "Analysis and classification of NoSQL databases and evaluation of their ability to replace an object-relational Persistence Layer." Architecture(2010): 1

[25] Kellerman, Jim. "Hbase: Structured storage of sparse data for hadoop." (2009).