Annales Mathematicae et Informaticae
56 (2022) pp. 1–9
DOI: https://doi.org/10.33039/ami.2022.12.006
URL: https://ami.uni-eszterhazy.hu

# Benchmarking Redis and HBase NoSQL Databases using Yahoo Cloud Service Benchmarking tool

**Mustafa Alzaidi, Aniko Vagner**

University of Debrecen-Faculty of Informatics
mustafa.alzaidi@inf.unideb.hu
vagner.aniko@inf.unideb.hu

**Abstract.** The Not Structured Query Language (NoSQL) databases have become more relevant to applications developers as the need for scalable and flexible data storage for online applications has increased. Each NoSQL database system provides features that fit particular types of applications. Thus, the developer must carefully select according to the application's needs. Redis is a key-value NoSQL database that provides fast data access. On the other hand, the Apache HBase database is a column-oriented database that offers scalability and fast data access, is a promising alternative to Redis in some types of applications. In this research paper, the goal is to use the Yahoo Cloud Serving Benchmark (YCSB) to compare the performance of two databases (Redis and HBase). The YCSB platform has been developed to determine the throughput of both databases against different workloads. This paper evaluates these NoSQL databases with six workloads and varying threads.

*Keywords:* Redis, HBase, YCSB, Benchmarking, NoSQL Databese

## 1. Introduction

A growing number of NoSQL databases are being developed and used. The promise of quicker and more efficient throughput compared to older Relational Database Management Systems (RDBMS) is one of its most compelling features[14]. There are several advantages to using NoSQL databases for cloud computing, including the ability to rapidly scale vertically and horizontally as needed and the easiness of application development[8]. However, big data and online application developers

Submitted: November 27, 2022
Accepted: December 3, 2022
Published online: December 28, 2022

should be aware that NoSQL databases are not usually equal when it comes to performance [6]. Because NoSQL systems are not yet mature and evolving at various paces, database managers must pick carefully between NoSQL and relational databases based on their demands regarding consistency, security and scalability, performance, prices, and other factors[15]. Choosing a NoSQL system might be a challenge for web application developers because of the large variety of open-source and freely accessible NoSQL systems. In other words, a peer-to-peer comparison of NoSQL systems according to the application activity scenarios to identify the most significant match for different situations would be an appropriate next step. A benchmark in this context refers to a performance assessment of NoSQL solutions that have been suggested or have been deployed. Then, compare the performance of different NoSQL databases; it is necessary to utilize experimental interactions that simulate comparable behavior or activities, as could be the case with applications behavior. Selecting a NoSQL system in this manner can be more appropriate for certain types of user interaction and provide better performance and efficiency than a competitor's systems. key-value, wide column, graph, and document databases are all examples of NoSQL databases[12, 15]. Key-value stores are collections of registers identifiable by a unique key [3]. Usually, this type of NoSQL system is used as a layer that provides cash for the data with time-consuming access[4]. Some researchers[2] use the key-value store when the application needs to retrieve the stored object based on one field value. Javascript and Binary Object-Notations (JSON and BSON) is a kind of document-oriented data[13]. Document-based databases provide more flexibility in terms of schema compared to RDBMS. They store the data in objects format in a similar manner to how programming language logically treats objects. The schema-less model enables the developer to store different types of objects in the same storage entity. This flexibility gave the ability to rapid application development [7]. Document store databases can work well on distributed systems that provide cheaper horizontal scaling as the application needs. Databases like MongoDB, CouchDB, and others fall within this category. The success of Google with BigTable seems to have sparked the development of column stores [5]. The column store databases stores the tables records fields separately, such that subsequent values of that property are saved sequentially [1]. Wide-Column database systems are built on a hybrid method that makes use of both the descriptive qualities of relational databases and the structure of different key-value stores [15]. Accumulo, Cassandra, as well as HBase are fall in this category. Graph databases may be used to store objects data, as well as all connections between them [15]. In this way, Graph databases make use of nodes and edges, the two notions from Graph theory. For example, a foreign or primary keys link between two nodes is an edge in the data domain. Neo4J and OrientDB are two good examples[11]. In this paper, we did use Yahoo Cloud Service Benchmarking (YCSB) tool benchmark Redis and HBase databases. We did the test with six different workload scenarios for each workload, and we recorded ten results by adding a new thread each time with ten threads till the last text.

## 2. Redis NoSQL database

Redis is an open-source in-memory key-value store database that is very customizable and claims to be extremely quick in terms of performance. VMware initially maintained it; later, Pivotal Software has taken over as the company that is sponsoring its development. Typically, the databases in Redis is specified by a numerical value. The number of databases is set at 16 by default, although this may be changed as a custom configuration. It is more customizable than a generic key-value structure in terms of data organization. For example, a value in Redis may be saved as a string, a list of strings with insertions at the beginning and end of the list. Furthermore, searching for objects towards the two ends of a huge list is incredibly quick, but querying for an item in the center of a large list is much more time-consuming. The collection of strings stored in Redis does not allow duplication, which implies that adding the same key (string) more than once will result in just one copy of the collection. The operations of adding and removing only need a constant amount of time (O(1)). Redis provides other structures like Hash, Set, and Sorted Set. Hash is referred to by a unique key and can store a set of unique fields, where each field can have one value. Hash provides high-speed data access in comparison to other structures. For instance, in comparison to List, even a colossal Hash can retrieve any key-field value with O(1). Redis also provide special commands that support synchronized data access. For example, BRPOP takes keys of List structures (one or more) as parameters and an integer number to specify the timeout in seconds. The command checks the specified lists in the same order given to the command and removes and returns the last element on that list. If all the lists are empty, the command blocks the current connection and waits for the amount of time specified by the timeout parameter for any other user connection that may be inserted to one of the lists before it release the connection and return a value to the client

## 3. HBase NoSQL database

A distributed, fault-tolerant, and with high scalability column-store NoSQL database implemented on top of the Apache Hadoop Distributed File System (HDFS), HBase is an Apache open-source database that provides real-time store and retrieving ability to massive data is. The data in HBase is arranged logically into named indexed tables. HBase tables are stored as multidimensional sparsely maps with rows and columns, where rows include a sorting key and an arbitrary number of columns. Versioning is used in table cells. When cells are added to HBase, HBase assigns a timestamp to them that is used to identify the version of that particular cell. For the same row key, many versions of a specific column might exist for that column. Column family and column name are assigned to each cell so that software can always tell what types of data item a particular set of cells contains. The content of a cell is an unbroken array of bytes that is uniquely recognized by the following combinations: Table + Row-Key + Column-Family: Column + Timestamp[9, 16].

A byte array, which also acts as the database's primary key, is used to sort the rows of the table

# 4. Experiments tool setup

## 4.1. Yahoo Cloud Service Benchmarking tool

We will use the Yahoo Cloud Service Benchmarking (YCSB) as the database performance evaluation tool. YCSB was created in 2010 by the research department at Yahoo. The task was to develop a tool that provides the ability to test and compare performance over the service provided by the cloud. Later, this tool becomes widely used by application developers to test database systems. In addition, this test can help during the decisions making to select the system to be used in the project. Figure 1 shows the tool architecture[6]. YCSB is developed using the

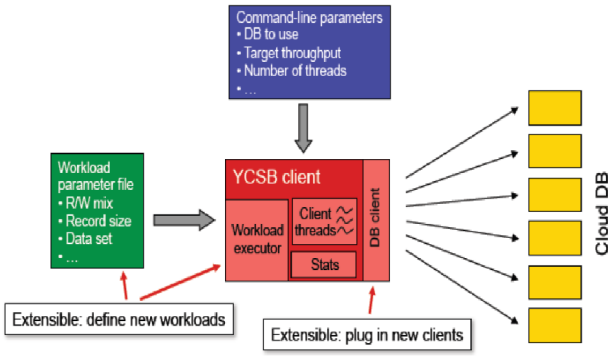


**Figure 1.** Architecture of YCSB.

Java programming language as an open-source project [10]. The code can be compiled with Maven and worked as a command line base. The tool support variety of NoSQL databases. The test is done by specifying the workload to be used. A workload can determine the number of operations and the types of these operations (Read, Write, and Update). There is a set of predefined workloads provided with tools default source code; we will use these workloads in this work, denoting them as (Load A, Load B, Load C, Load D, Load E, Load F). The test is done in two steps: the Load command and the Run command. The database connection information can be provided as a parameter to the tool with the Run and the Load command.

## 4.2. Hardware and software specifications

Table 1 below shows the system specification we used for this work.

We conduct the test using six workloads. We recorded the result by changing the number of threads used in the test. For each test, we build a chart that

**Table 1.** Hardware and software specifications.

| System | |
|---|---|
| Operating System | Window 10 64 bit |
| Memory (RAM) | 8GB |
| CPU | Intel Core i5-1135G7 4 x 2.4 - 4.2 GHz |
| Software | |
| Yahoo Cloud Service Benchmarking<br>Redis | Version ycsb-0.17.0<br>Version 6.2.6 |
| HBase | Version 2.4.9 |
| Maven | Version apache-maven-3.8.4z |

shows the recorded performance (throughput measured by operation per second) for both databases while changing the number of used threads. The number of threads can be determined in practice according to the application. The result for each workload is shown below:

## 4.3. Load A

In this workload, the tool divides the total operation into 50% read, and 50% write operation. Thus this workload can be considered heavy in terms of updates. The result is shown in Figure 2 below. We notified that the HBase started to give better performance when we increased the thread from six to seven threads with this load. However, we got a similar performance gap with more than seven threads. Thus, this load shows better performance for HBase in comparison to Redis.

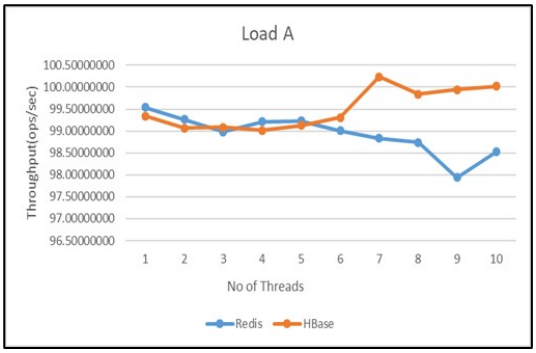


**Figure 2.** Load A.

## 4.4. Load B

The read operation takes 95% of the total operations in this workload. Thus we can denote this workload as reading heavy test. The max recorded throughput for

Redis and HBase is 99.54 100.33 milliseconds, respectively. The result is shown in Figure 3. Again, the HBase performs better than Redis with eight or more threads. Redis has no notifiable change during all the threads experiment.

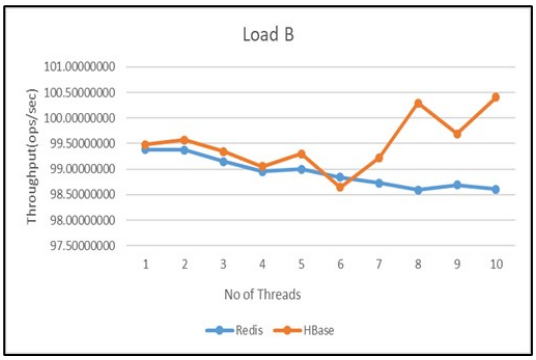


**Figure 3.** Load B.

## 4.5. Load C

This workload consists only of read operations and can be used to test the database when the application is critical to data retrieval, and there is no rapid insertion or update operation that can affect the software. The max recorded throughput for Redis and HBase is 99.36 and 100.39 milliseconds, respectively. The result is shown in Figure 4.

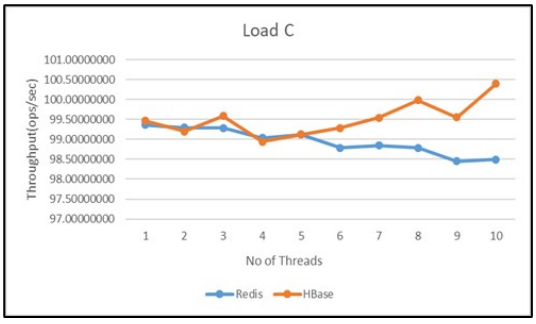


**Figure 4.** Load C.

## 4.6. Load D

This load contains only 5% insert operation with 95% read operations. The read operations are done on the data that was inserted recently. The max recorded throughput for Redis and HBase is 99.48 100.61 milliseconds, respectively. Figure 5

shows the Load D result. HBase shows better performance with increasing the number of threads.

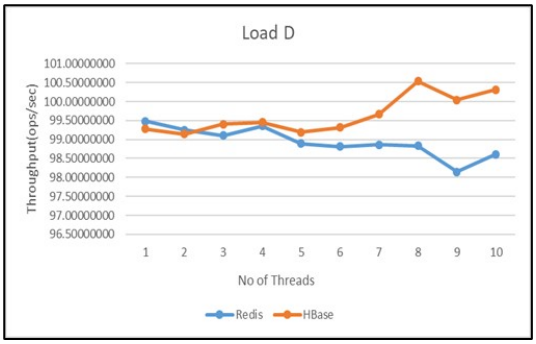


**Figure 5.** Load D.

## 4.7. Load E

95% of the time is spent scanning, and just 5% is spent inserting. It is scan for a short number of records rather than a single one. Figure 6 shows the result comparison for both databases. The max recorded throughput for Redis and HBase is 99.48 100.87 milliseconds. Both databases show similar performance till we use seven threads. However, the performance gap after seven or more threads was smaller compared to the gap we got with the other tests. Again the HBase was slightly better than Redis for this test.

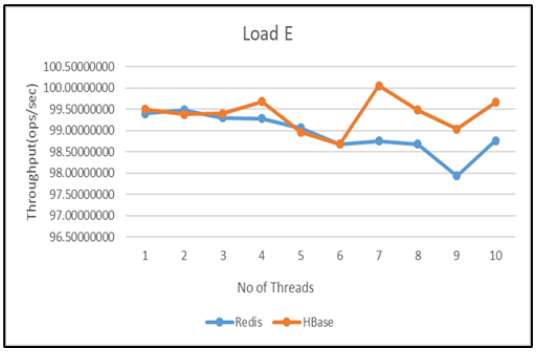


**Figure 6.** Load E.

## 4.8. Load F

This load simulates the situation when the application retrieves the data from the database, updates it, and then stores it back in the database. Figure 7 shows the

result for load F.

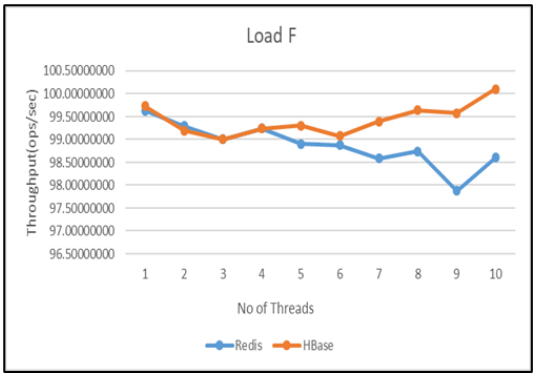


**Figure 7.** Load F.

# 5. Conclusion

Applications programmers may choose between SQL and NoSQL databases. Although their antiquity, SQL databases are still popular among programmers and web designers alike. The NoSQL database systems have become a good alternative to relational databases in some applications during the last decade. As they provide better scalability and schema-less structure, what can make software project development faster and easier. This advantage and popularity lead to the introduction of many NoSQL database systems. However, each may provide some features and miss some others that are provided by another system. Thus, the selection between the available NoSQL databases becomes more complex and needs a comparison between the candidate systems. We use the Yahoo Cloud Service Benchmarking tool to compare two popular NoSQL databases. We used the default workload provided by the tool, and we re-conducted the test using a different number of threads every time (1 to 10 threads). The results show that both databases have almost similar performance when fewer threads are used (less than 7). However, when we increase the number of used threads, the HBase shows higher throughput in compare to Redis.

# References

[1] D. Abadi: *Column Stores for Wide and Sparse Data.* In: Feb. 2007, pp. 292–297.

[2] A. V. M. Alzaidi: *Trip Planning Algorithm For Gtfs Data With Nosql Structure To Improve The Performance*, Journal of Theoretical and Applied Information Technology Vol.99. No (10 31st May 2021 May 2021), pp. 2290–2300.

[3] E. Anderson, X. Li, M. Shah, J. Tucek, J. Wylie: *What consistency does your key-value store actually provide?*, HP Laboratories Technical Report (Feb. 2010).

[4] B. Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, M. Paleczny: *Workload analysis of a large-scale key-value store*, Sigmetrics Performance Evaluation Review - SIGMETRICS 40 (Feb. 2012), doi: https://doi.org/10.1145/2318857.2254766.

[5] R. Cattell: *Scalable SQL and NoSQL data stores*, SIGMOD Record 39 (Feb. 2010), pp. 12–27, doi: https://doi.org/10.1145/1978915.1978919.

[6] C. Chakraborttii: *Performance Evaluation of NoSQL Systems Using Yahoo Cloud Serving Benchmarking Tool*, in: Feb. 2015.

[7] C. Chasseur, Y. Li, J. M. Patel: *Enabling JSON Document Stores in Relational Systems.* In.

[8] B. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, R. Sears: *Benchmarking cloud serving systems with YCSB*, in: Feb. 2010, pp. 143–154, doi: https://doi.org/10.1145/1807128.1807152.

[9] L. George: *HBase: The Definitive Guide: Random Access to Your Planet-Size Data*, 1st, O'Reilly Media, Inc.: Sebastopol, CA, USA, 2011.

[10] https://github.com/brianfrankcooper/YCSB.: *YCSB*, in.

[11] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, H. Karambelkar: *Bidirectional Expansion For Keyword Search on Graph Databases.* In: vol. 2, Feb. 2005, pp. 505–516.

[12] H. Khazaei, M. Fokaefs, S. Zareian, N. Beigi, B. Ramprasad, M. Shtern, P. Gaikwad, M. Litoiu: *How do I choose the right NoSQL solution? A comprehensive theoretical and experimental survey*, Journal of Big Data and Information Analytics (BDIA) 2 (Oct. 2015), doi: https://doi.org/10.3934/bdia.2016004.

[13] K. Ma, A. Abraham: *Toward lightweight transparent data middleware in support of document stores*, in: 2013, pp. 253–257, doi: https://doi.org/10.1109/WICT.2013.7113144.

[14] T. Madushanka, L. Mendis, D. Liyanage, C. Kumarasinghe: *Performance Comparison of NoSQL Databases in Pseudo Distributed Mode: Cassandra, MongoDB & Redis* (Feb. 2015).

[15] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, S. Belfkih: *Comparison and Classification of NoSQL Databases for Big Data*, in: Feb. 2015.

[16] M. N. Vora: *Hadoop-HBase for large-scale data*, Proceedings of 2011 International Conference on Computer Science and Network Technology 1 (2011), pp. 601–605.