ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
https://doi.org/10.17559/TV-20180916120558
Original scientific paper

Evaluating Riak Key Value Cluster for Big Data

Aimen Mukhtar RMIS, Ahmet Ercan TOPCU

Abstract: NoSQL database has become an important alternative to traditional relational databases. Those databases are prepared by the management of large, continuously and variably changing data sets. They are widely used in cloud databases and distributed systems. With NoSQL databases, static schemes and many other restrictions are avoided. In the era of big data, such databases provide scalable high availability solutions. Their key-value feature allows fast retrieval of data and the ability to store a lot of it. There are many kinds of NoSQL databases with various performances. Therefore, comparing those different types of databases in terms of performance and verifying the relationship between performance and database type has become very important. In this paper, we test and evaluate the Riak key-value database for big data clusters using benchmark tools, where huge amounts of data are stored and retrieved in different sizes in a distributed database environment. Execution times of the NoSQL database over different types of workloads and different sizes of data are compared. The results show that the Riak key-value is stable in execution time for both small and large amounts of data, and the throughput performance increases as the number of threads increases.

Keywords: big data; cluster; NoSQL database; Riak; YCSB

1 INTRODUCTION

Relational databases have become non-scalable in the era of the exponential growth of data. Several organizations now store huge amounts of scientific, sales, customer, and other data for future analysis. Historically, most organizations have compiled and stored structured data for access and analysis in relational databases. Because those relational databases are not efficient, now various types of non-relational databases, frequently called 'Not Only Structured Query Language' (NoSQL) databases, are being widely developed [1].

There are many benefits to using NoSQL databases. For example, they provide elastic scaling, high availability, and reliability, which have become especially important as cloud computing has gained wide adoption. In fact, NoSQL databases were introduced to counter the limitations of relational database methodology. NoSQL databases provide superior performance and their data model addresses several short comings of traditional relational databases. Organizations use NoSQL databases for a number of varying usage [2, 3, 4]. NoSQL databases are mostly open-source projects, which means a comparatively low-cost method of developing, sharing and implementing software [5, 6]. Commodity computers are now cheap, and the Internet is more widespread than ever before. Huge amounts of structured, unstructured, and semi-structured data are being collected and stored for a variety of applications. All these data are now referred to as 'big data' [7].

Processing data on distributed systems requires processors with decent speeds. The need for fast processors by relational databases could grow exponentially as the amount of data grows. This leads to the emergence of a number of NoSQL database offerings as well as open-source and commercial implementations of NoSQL databases [8]. Nowadays, there are various NoSQL databases that are widely adopted for handling big data. Although, all these databases are designed to handle large amounts of data, it should be understood that every NoSQL database is designed to mostly address a specific kind of data and do so in a specific way. Except for all being non-relational in nature, NoSQL databases are quite different from each other and they likely do not yield the best results in terms of performance and consistency in random situations. It is therefore important to methodically examine and test these databases under different circumstances, such as different workload conditions. The factors that cause these databases to perform best and worst in a cluster configuration need to be identified [9, 10].

The term big data that is very related with NoSQL database systems. Big data is a term utilized to refer to the increase in the amount of data that are difficult to store, analyze, and process through traditional database technologies. It is estimated that the total data size is growing 40 percent per year and will grow 44 times between 2009 and 2020. However, due to its structure, most of this data is unmanaged. This data comes from many sources; for example, sensors are used to collect weather information, digital videos and pictures, social media sites, buying and selling transactions, cell phone data and market analysis among others [11].

Big data has become a very important engine for growth and innovation that depend on emerging technologies such as the Internet of Things (IoT), cloud computing and analytics. Big data is thus very important to enhance output increase in the world since it is affecting software intensive industries, education, administration and health sectors. [12]

In this paper, we study, analyze, and evaluate the NoSQL databases, mostly Riak KV (Key-Value), for big data clusters. We experimentally analyze the different workload execution times while adding threads. For the experimental work, we use the Yahoo! Cloud Services Benchmark (YCSB) to provide a general workload to evaluate the performance of various NoSQL databases.

The main purpose of this paper is to generate a fictitious workload and a data access pattern on the cluster that matches the workloads of real-world applications and monitors its performance with large data volumes and different types of workloads. Unlike previous studies, to simulate a popular use case for these systems we use commodity hardware for our cluster hardware. We then monitor the performance of the Riak KV (execution times according to number of threads) when data is being read and update operations. We then analyse the effect of the

parallelization on these operations in this hardware constrained environment.

The rest of this paper is organized as follows. Section 2 takes a deeper look at related work. Section 3 presents types of NoSQL databases, which typically fall into one of four categories. Section 4 presents the results of experiments realized by testing the Riak KV NoSQL database with the YCSB. Section 5 provides our experimental results. Section 6 concludes the paper.

2 RELATED WORKS

There have been numerous papers that test and evaluate NoSQL database, such as that by Klein et al. [13], reporting on the methodology and the results of their work on three NoSQL databases (MongoDB, Cassandra, and Riak) for huge distributed healthcare organizations. They studied the availability, partition tolerance, consistency, and other quality attributes that affect the throughput based on selection decision, which changed from 225 to 3200 operations per second between database products, and it was found that obtaining strong consistency diminished throughput by 10%-25% compared to definitive consistency. In the study introduced by Kabakus and Kara [8], various in-memory NoSQL databases were tested in order to determine their performances in terms of memory efficiency during operations and the time it takes to complete processes. The in-memory databases that were tested in this paper were MongoDB, Redis, Memcached, Cassandra, and H2. Based on the results obtained in that work, no database offered the best performance for all data operations. Even if relational database-management system (RDBMS) stores its data in memory, in general the performance is worse than that of NoSQL databases. Abramova et al. [14] tested the performance of Cassandra based on a number of factors, including the number of nodes, workload characteristics, number of threads, and data size, and analyzed whether it provides the desired acceleration and scalability attributes. Scaling nodes and the number of data-sets do not guarantee performance. However, Cassandra handles concurrent request threads well and extends well with concurrent threads. A summary of the results of that paper concluded that when the number of nodes in a cluster has increased from 1 or 3 to 6, even for relatively large data sets, this trend cannot guarantee an improvement in performance.

Li and Manoharan [15] compared the performance of five experiments with NoSQL databases: 1) time to fetch all keys; 2) the time to write the value corresponding to the given key; 3) time to read KV pairs; 4) time to delete KV pairs; and 5) time to instantiate database buckets. These experiments were also tested with different data amounts, ranging from 10 records to 100 000 records. The databases tested were RavenDB, MongoDB, Cassandra, Hypertable, CouchDB, Couchbase, and MS SQL Express. The authors reported that MongoDB and Couchbase were the fastest two overall to write, read and delete operations. They also noticed that Couchbase was lacking its ability to fetch all keys from the database.

In the study presented by Tang and Fan [16] an attempt was made to evaluate and test the performance of five NoSQL clusters (HBase, MongoDB, Cassandra, Couchbase, and Redis) by using a-measurement tool in the YCSB. The experiments mainly involved the following three workloads: 1) Workload A, update heavy, 50/50 of reads/updates; 2) workload C, read only, 100% reads operation; and 3) workload H, update only, 100% updates operation. Based on an analysis of the mechanisms of each database, the experimental results provided guidance for NoSQL users and developers. The conclusion was that Redis is especially suitable for loading and executing workloads. It also offers the best efficiency, the databases of documents, followed by the databases of columns of the family, also have good average performance since they have both scalability and efficiency.

Manoj [17] evaluated Cassandra's performance by comparing a RDBMS and Cassandra read and write performances, which were calculated on the basis of a number of threads. In the experimental setup, they used three nodes of the Cassandra cluster with each machine's OS: CentOS5.7, 100 GB disk space, 1 GB memory, and a Java v.1.6.0. The test case focused on the Cassandra concurrency was required for logging systems and for write performance. Manoj concluded that Cassandra's write performance was faster than that of the RDBMS and the reading performance was slower when tested with 1000 threads. Abramova et al. [18] used the YCSB to compare and evaluate the performance of NoSQL databases. They generated 600 000 records and used them with different workloads by changing the proportions of reading, updating, and insertion operations. The databases used in their experimental evaluation were MongoDB, Redis, OrientDB, HBase, and Cassandra. They reported that, in general, the Redis database provided the best performance. Moreover, they reported that the HBase and Cassandra column-family databases showed better update performance.

In other work, Tudorica and Bucur [19] compared various NoSQL systems using multiple criteria. He [20] theoretically discussed why NoSQL databases would be better in a big data setup. It was proven that a NoSQL solution would outperform relational database solutions. Tsuyuzaki and Onizuka [21] discussed the characteristics of NoSQL databases and their benchmark systems. They reported results from running a benchmark on a MongoDB database. They tested it for elasticity and availability and revealed that data size has a significant impact on database performance when the system is extended or when machines are taken off-line.

There are many factors that differentiate this study from any previous study. In our research, we ran all our experiments on commodity hardware which simulates the reality on how the NoSQL databases are used. We wanted to test the limitation of the software in this hardware constrained environment. We gradually increased the data to the point where it became huge. Along with this we tested the effect of parallel implementation in the software in the distributed environment. The study monitors the performance of the Riak KV (execution times according to number of threads) when data is being read and update operations. The execution times of various workloads of different record volumes are the main comparative indicators; In addition, as the versions of Riak KV are continually updated, the performances are optimized quickly, while the others are increasing relatively slowly, which drives to the result that the performance evaluation

may not be the same as before. Because of this, we evaluated the latest version of Riak KV 2.2.3, and the results were interesting as shown in the paper. Previous studies were conducted in a rather hardware flexible environment. The focus of the bulk of previous studies has been on how these software perform with different benchmarks in a normal manner [13, 6]. The reality is organizations that use this software tend to run it on hardware limited environments and with big data and that is what we tried to evaluate in this research.

3 NoSQL DATABASE OVERVIEW

NoSQL databases can be classified into four categories.

3.1 Key-Value (KV)

In general, NoSQL databases allow the use of various types of relational data tools. These are becoming common in new business plans and big data analysis in which classified data should be stored in a practical and efficient manner [10].

Within this context, key-value store databases are the simplest NoSQL databases. They can help developers in the absence of a predefined schema. Different kinds of objects, data types, and data containers are used to accommodate this [22, 23].

High query speed with a simple structure, where KV is the data model, supports benefits such as high concurrency and mass storage. Data modification and query operations are well supported through primary keys, such as Riak KV [24] and Redis [25].

3.2 Column-oriented

A table in a column-oriented database can be used for the data model; however, table associations are not supported. A column-oriented database has the following features:

- Data are stored separately for each column.
- Every data column is an index of the database.
- It only accesses columns involving query results to reduce system input and output.
- Synchronous process queries mean that each column is handled by one process.
- There are the same data types, a good compression ratio, and similar characteristics.

In general, the benefit of this data model is a more appropriate application on aggregation and data warehouses, such as HBase [26] or Cassandra [27].

3.3 Document

The KV and document database structures are very similar, but the value of the document database is stored in XML or JSON format. In addition, the document database can usually use the secondary index to facilitate the value of the upper application [28, 23], such as MongoDB [29].

3.4 Graph Databases

A graph database comprises nodes that are connected by edges. Data can be stored in edges and nodes. One advantage of a graph database is that it can traverse relationships very quickly. Similar to the other three kinds of NoSQL databases abovementioned, graph databases have some difficulties with horizontal scaling. This is why every node can connect to any other node. Traversing nodes on various physical machines can have a negative effect on performance. Another difference from the above three is that most graphics databases support ACID (atomicity, consistency, isolation, and durability) transactions. Graph databases are often used to deal with complex issues such as social networks or path-finding problems [28]. Neo4j [30], is example of graph database.

4 EXPERIMENT

In this section, we will introduce the results of experiments realized by the testing of the Riak KV NoSQL database with the YCSB.

4.1 Riak KV

Key-value database, or key-value store is a simple database that is used for managing, retrieving, and storing associative arrays, a more commonly known data structure today is a hash table or dictionary. Dictionaries contain a collection of records, or objects. They contain many different fields, each of which contains data. These records are retrieved and stored using a key that uniquely identifies records for quickly finding data in the database. In general, KV databases do not have a query language. They provide a way to update, retrieve and store data using simple put, delete and get commands [31].

Riak is one of a KVdatabases. It is an open-source enterprise version of Riak Enterprise DS. It is a KV database developed by Basho in 2007 and written in Erlang and C. The enterprise version adds multi-data center monitoring, replication, and additional support [28].

Riak is a distributed NoSQL database that is extremely scalable, available, and straightforward to work with. It automatically assigns the data in a cluster to ensure quick performance and fault tolerance. Riak Enterprise contains multi-cluster replication that guarantees low latency and strong business continuity. Riak KV is an appropriated distributed NoSQL KV database that ensures read and write functions even in cases of hardware failure or network partitions by supporting both local and multi-cluster replication. Riak KV is designed to work and deal with a combination of challenges facing big data applications that combine following session or client data, storing data from connected devices, and replicating data around the world. It is designed with KV to provide a powerful, simple data model to store large amounts of unstructured data [28, 24].

Riak KV achieves fast performance and robust business continuity by automating data distribution across the cluster, where there is easily added capacity without a large operational burden with a masterless architecture that guarantees high availability and scales that are nearly linear using commodity hardware [24]. Nodes in Riak form a

cluster. This cluster is isolated into partitions and virtual nodes (Vnodes) to form a ring to obtain all the benefits of Riak. The ring is a 160-bit integer space separated into a similarly sized partitions, as shown in Fig. 1.

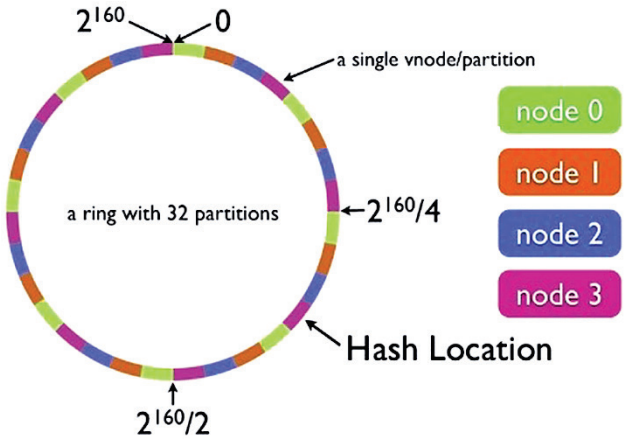

Figure 1 Architecture of the Riak cluster

Each node (also called a physical node) in the ring runs a certain number of virtual nodes (Vnodes). Each Vnode occupies one partition in the ring. It defines the partition size of the ring when configuring Riak or when the cluster is initialized [32].

4.2 Yahoo! Cloud Serving Benchmark (YCSB)

Yahoo engineers developed YCSB as an open source, it supports many NoSQL data stores. The YCSB tool implements a vector-based approach to improve benchmarks to better address the performance of specific applications. This tool is designed for database systems expanded on the cloud. It is understood that these systems usually do not have SQL interfaces. They only support part of the relational operations and the use cases are usually quite different from applications used with traditional relational databases; subsequently, they do not apply to existing tools used to benchmark these systems, as shown in Fig. 2 [35].

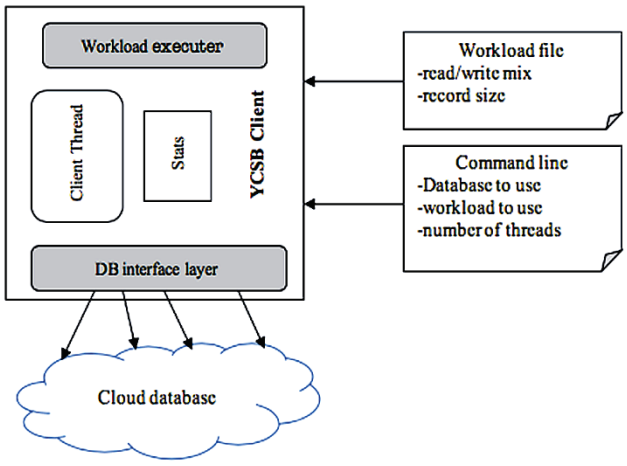

Figure 2 The architecture of the YCSB client

The YCSB core software package is designed to evaluate different aspects of system performance, including a series of workloads, to assess the suitability of the system for different workload characteristics at different points in the performance space [33, 34].

4.3 Dataset for the Experiments

The default data size in the basic core differs from the same basic application type. With this benchmark, there is a record table with 10 fields for each record. Each record is identified by a primary key, such as a string "user412356". Every field is named field_0, field_1, and so on. The value of each field is a random ASCII string, with 1 KB records (10 fields, 100 bytes each, plus key). Tab. 1 shows the dataset used.

Table 1 The dataset used in the experiment

Record count	Record size (for each record)	Total size (of the dataset)
10 K	1 KB	10 MB
100 K	1 KB	100 MB
1000 K	1 KB	1 GB
10000 K	1 KB	10 GB
20000 K	1 KB	200 GB

4.4 Experimental Setup

The experiments were performed in the following environment using 5 nodes of the cluster, as shown in Tab. 2.

Table 2 Specification of the environment used in the conducted experiment

Operating-system	Ubuntu-16.04 (64-bit)
Memory	16 GB
CPU	Intel®-Xeon(R)-CPU E3-1241 v3-@ 3.50 GHz×8
Database	Riak KV 2.2.3
Benchmark tools	YCSB 0.12.0

Fig. 3 illustrates the experimental structure, with details of the primary components.

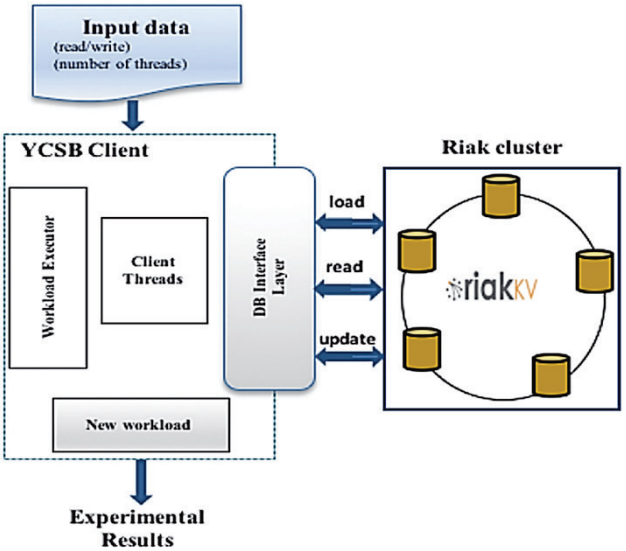

Figure 3 Experimental structure

4.5 Performance Configuration

The YCSB is a test tool to perform reads, updates, and writes based on workload and measure performance. The YCSB is commonly used for the evaluation and performance of NoSQL databases. The benchmark

contains two parts: a data producer and a set of performance tests that evaluate, update, and read operations. Each test scenario is called a workload and is defined by a set of operations such as update and read, some used records, and a total number of transactions. The benchmark tools give a set of predetermined workloads that can be executed as follows:

Workload A: Updates are heavy. It consists of a 50/50 proportion of reads and updates.

Workload B: Mostly reads. It consists of a 95/5 proportion of reads/updates.

Workload C: Reads only. The workload is 100% reads.

To evaluate the loading time, we generated a different number of records (10 K, 100 K, 1000 K, 10,000 K, and 200,000 K) and a varying number of threads (1, 4, 8, and 12). Each experiment comprised 10,000 operations.

5 EXPERIMENTAL RESULTS

5.1 Workload (A): Update heavy workload. This workload has a mix of 50/50 reads and updates.

The results of the execution of the 50% read-50% update workload are shown in Fig. 4. From the results, we observe that Riak KV generally has better throughput performance when we increase the number of threads. However, the execution time using a small number of records (for example, 10 K) was very close to the execution time using a large number of records (200000 K), as shown in Fig. 4. We notice from the figure with 1 thread that when the number of records in the cluster increased from 10 K and 100 K to 1000 K, the execution time was 8.06 minutes, 8.28 minutes, and 8.58 minutes, respectively. However, when the number of records is 10 000 K and 200 000 K, the execution time becomes 8.43 minutes.

The results shown in Fig. 5, comparison of execution times, with 1 thread, the execution time increased by about 7 times compared to 8 and 12 threads. The case is the same for 4 threads, where the execution time decreases by almost 6 times.

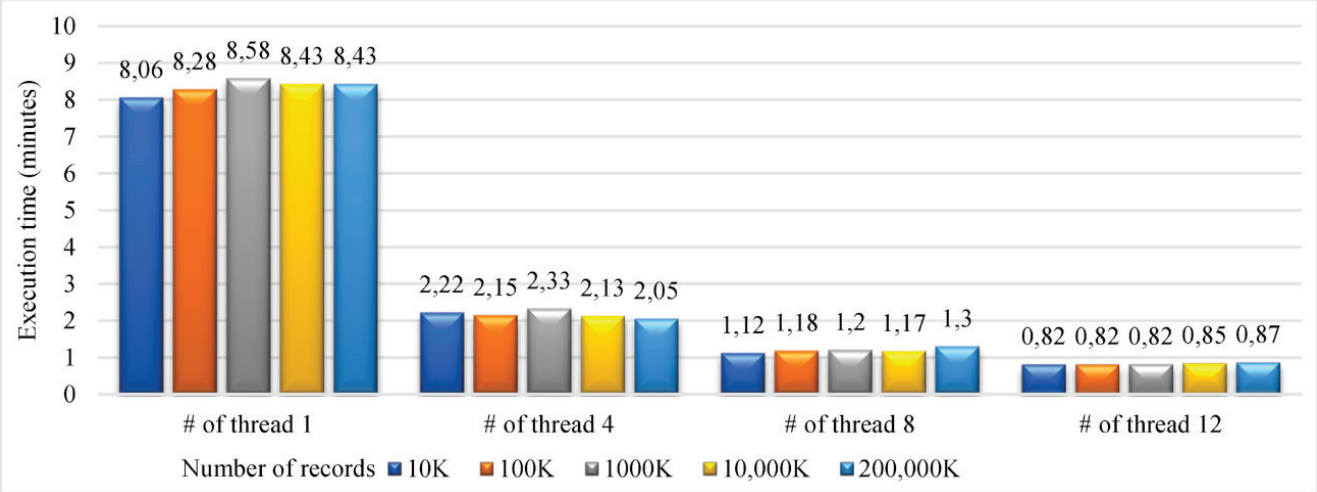

Figure 4 Execution time for workload (A) (50/50 read/update)

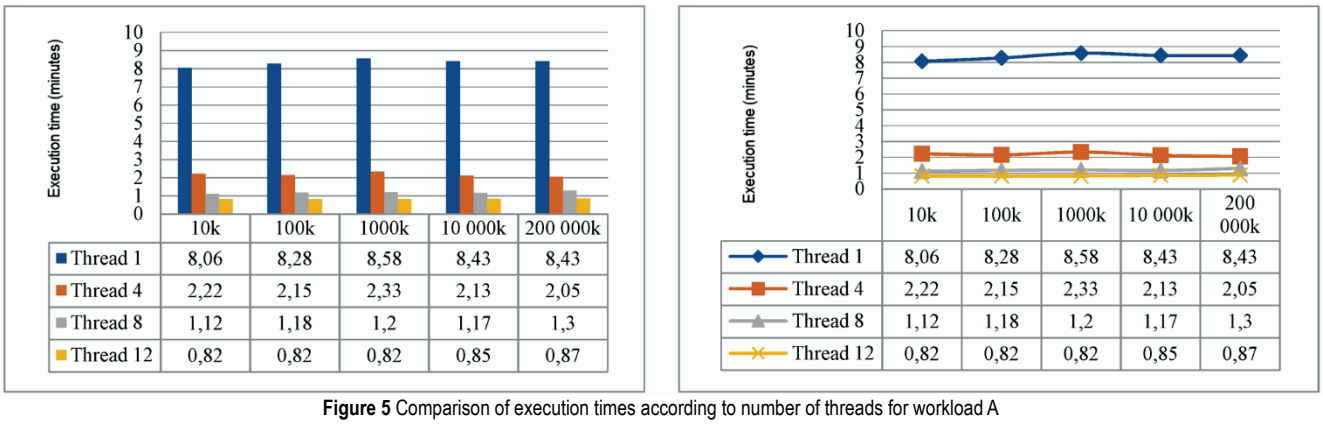

Figure 5 Comparison of execution times according to number of threads for workload A

The result was that 12 threads differed from the other cases (1, 4, and 8). When increasing the number of records to 10 K, 100 K, 1000 K, 10000 K, and 200000 K, the execution time increased by 0.82, 0.82, 0.82, 0.85 and 0.87 minutes, respectively.

5.2 Workload B: 95% reads and 5% updates

The performance behavior exhibited in workload A differed from the experiment conducted for workload B (95% reads, 5% updates). Moreover, the execution time in workload B was higher than the execution time of workload A with all threads. Furthermore, the execution

time decreased when we increased the number of threads; for example, for number of records 10000 K with 1, 4, 8, and 12 threads, the difference in execution time was very large (15.92, 4.00, 2.04 and 1.35 minutes, respectively).

As can be seen in the figure above, the number of records has no significant effect on the performance of Riak KV. For example, in Fig. 6, the execution times using 10 K records and 200000 K are 15.55 minutes and 15.66 minutes, respectively, which is a very small difference.

Similar to the previous results obtained with 8 and 12 threads, the execution time using a smaller number of records (10 K, 100 K) was very close to the execution time using a large number of records (200000 K), as shown in Fig. 7. On the other hand, the results are shown for 1 thread, where the execution time increased about 7 times compared to 4, 8 and 12 threads. We noted alike behavior compared to the execution of workload A.

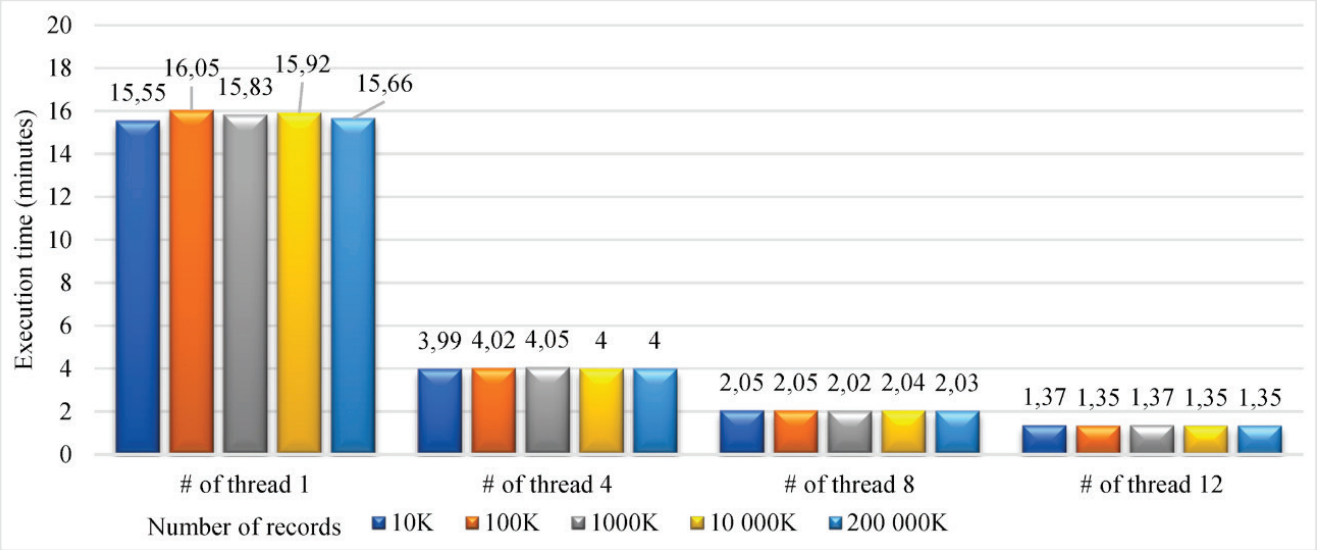

Figure 6 Execution time for workload B (95/5 of reads/updates)

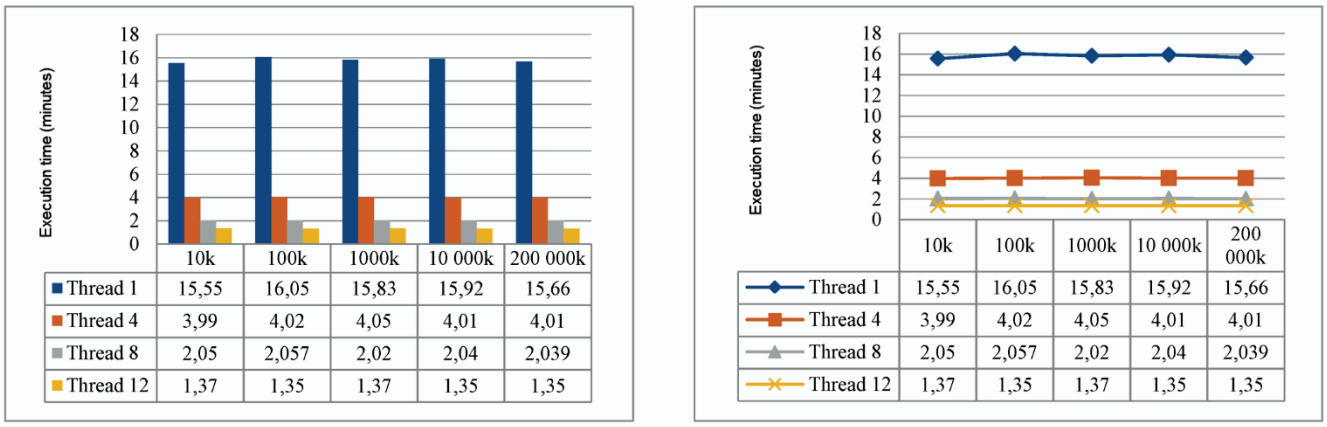

Figure 7 Comparison of execution times according to number of threads for workload B

5.3 Workload C: Read-only, with workload read ratio 100%

The results of the execution of workload C with 100% reads are shown in Fig. 8.

In workload C test, the increase of reading records increases the execution times lightly increased, which confirms again that the number of records has no significant effect. For example, Fig. 8 shows the execution time of using 100 K records as being 2.09 minutes, and when the number of records reaches 200000 K, the execution time is still the same (2.09 minutes).Overall, performance is worse than the execution time of workload A.

This workload is 100% read and the Riak KV is not enhanced for performing reads compared to the update. As the number of reads performed by retrieving a disk increases, performance is degraded.

Fig. 9, shows the execution time for 100 K and 200000K records as being 4.17 minutes and 1.40 minutes, respectively. The end result shows that as threads increase, execution time is decreasing regardless of workload type.

As the number of threads increases, this advantage still exists. The system takes advantage of this opportunity to effectively handle more threads.

In summary, the Riak KV database provided the best throughput performance when we increase the number of threads. However, with a small number of records, the execution time in every workload was higher. We observed that Riak KV generally has better throughput performance when we increase the number of threads, which is perfect

for handling big data applications and provides a powerful tool for storing massive amounts of unstructured data.

The examination of the effect of synchronization is very important for the sake of enhancing benchmarks. The more we know on how processing data simultaneously on these systems effect their performance, the more likely we will be able to make the benchmarks for these systems better tools for measuring them.

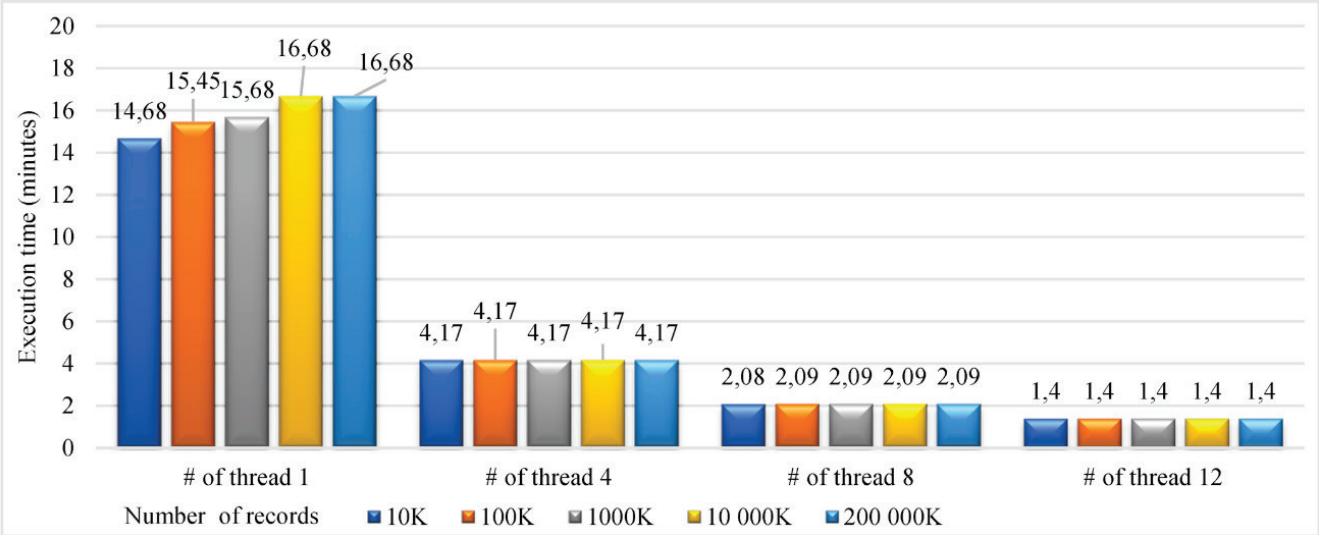

Figure 8 Execution times for workload C (100% reads)

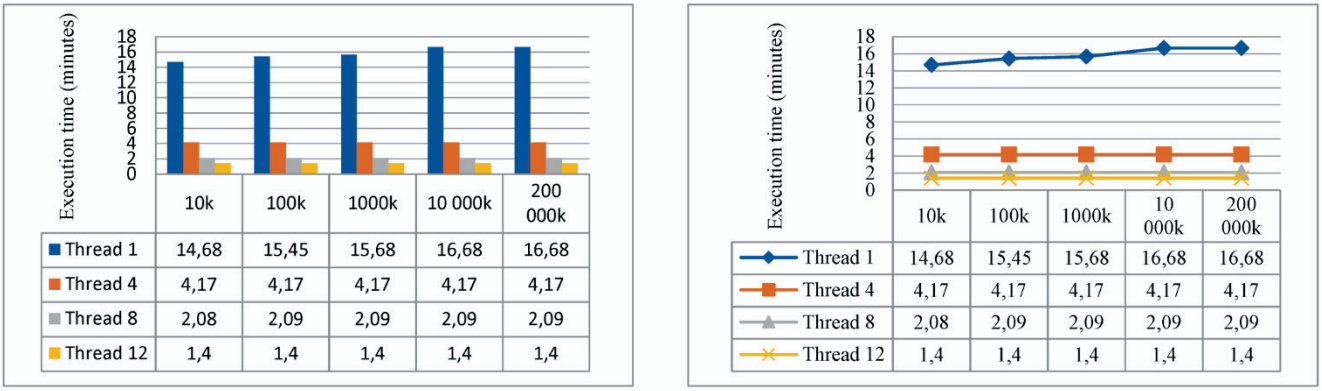

Figure 9 Comparisons of execution times according to number of threads for workload C

6 CONCLUSIONS

NoSQL technologies have become enterprises to provide services and rich applications to more users and developers than ever before. With the collection and processing of massive data, the popularity of NoSQL databases has also increased. Compared with relational databases, these databases have many advantages, especially for unstructured or semi-structured big data. There are various types of NoSQL databases, each with its own set of characteristics and properties, which results in performance differences. Therefore, it is necessary to analyze and compare the execution time differences of NoSQL databases to provide a performance reference [1, 25]. After a rough process of setting up and installing the Riak KV NoSQL database management systems of the cluster environment, to simulate a popular use case for this NoSQL database, we use commodity hardware for our cluster hardware for tested and evaluated Riak KV using the YCSB also tested factors such as data size and number of threads. We compared the execution times of this NoSQL database over different types of workloads and different numbers of records.

In our tests we used 1, 4, 8, and 12 threads, assuming that adding more threads would result in the execution time decreasing and in the best performance. However, we found that the execution times of the small and large records were approximately equal to each other. We also found that with the increase in the number of threads, the throughput performance of Riak KV improves with bigger datasets. We found that Riak is overall more efficient in the mix operations: 50% read and 50% update, with workload A.

Contrary to the intuitive assumption that parallelization always enhances performance, our results show that in small amounts of data, the performance does not necessarily improve when we launch more threads. There has not been a lot of research on the effect of parallelization of the performance of the NoSQL database systems when it is deployed on commodity hardware environment.

7 REFERENCES

[1] Tauro, C. J. M., Aravindh, S., & Shreeharsha, A. B. (2012). Comparative Study of the New Generation, Agile, Scalable, High Performance NoSQL Databases. In *International Journal of Computer Applications*, (0975-888), 48(20), 1-4. https://doi.org/10.5120/7461-0336

[2] Khan, M., Khan, S., & Iqbal, N. (2017). Computational Performance Analysis of Cluster-based Technologies for Big Data Analytics. *IEEE International Conference on Internet*

of Things (iThings). https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2017.239

[3] Assunção, M. D., Calheiros, R. N., Bianchi, S., Netto, M. A. S., & Buyya, Rajkumar. (2015). Big Data computing and clouds: Trends and future directions. *J. Parallel Distrib. Comput. Elsevier Inc.*, 79-80, 3-15. https://doi.org/10.1016/j.jpdc.2014.08.003

[4] big data made simple. (2018). Retrieved from http://bigdata-madesimple.com/top-five-advantages-and-disadvantages-of-nosql/.

[5] Jain, V. & Upadhyay, A. (2017). MongoDB and NoSQL Databases. *International Journal of Computer Applications (0975-8887), 167*(10), 16-20. https://doi.org/10.5120/ijca2017914385

[6] Salehnia, A. (2015). Comparisons of Relational Databases with Big Data: a Teaching Approach. South Dakota State University Brookings, Retrieved from https://www.asee.org/documents/zones/zone3/2015/Comparisons-of-Relational-Databases-with-Big-Data-a-Teaching-Approach.pdf /.

[7] Li, Y. & Manoharan, S. (2013). A performance comparison of SQL and NoSQL databases. *In IEEE conference on communications, computers and signal Processing (PACRIM)*, Victoria, BC, 2013, 15-19. https://doi.org/10.1109/PACRIM.2013.6625441

[8] Kabakus, A. T. & Kara, R. (2017). A performance evaluation of in-memory databases. *Journal of King Saud University Computer and Information Sciences, 29*(4), 520-525. https://doi.org/10.1016/j.jksuci.2016.06.007

[9] Gupta, A., Tyagi, S., Panwar, N., Sachdeva, S., & Saxena, U. (2017). NoSQL Databases: Critical Analysis and Comparison. *International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*, 293-299. https://doi.org/10.1109/IC3TSN.2017.8284494

[10] Asadulla Khan Zaki. (2014). NoSQL databases: new millennium database for big data, big users, cloud computing and its security challenges. *IJRET: International Journal of Research in Engineering and Technology, 03*(03), 403-409. https://doi.org/10.15623/ijret.2014.0315080

[11] Vinay, J. & Kumar, S. (2015). Big Data Analytic Using Cloud Computing. *Second International Conference on Advances in Computing and Communication Engineering.* 1-2 May 2015. Dehradun, India. https://doi.org/10.1109/ICACCE.2015.112

[12] Kousiouris, G., Menychtas, A., Kyriazis, D., & Varvarigou, T. (2016). Comparison of Database and Workload Types Performance in Cloud Environments. *Conference Paper in Lecture Notes in Computer Science*. https://doi.org/10.1007/978-3-319-29919-8_11

[13] Klein, J., Gorton, I., Ernst, N., Donohoe, P., Pham, K., & Matser, C. (2015). Performance Evaluation of NoSQL Databases: *A Case Study*. In *Proceedings of the 1st Workshop on Performance Analysis of Big Data Systems (PABS '15)*. ACM, New York, NY, USA, 5-10. https://doi.or/10.1145/2694730.2694731

[14] Abramova, V., Bernardino, J., & Furtado, P. (2014). Testing Cloud Benchmark Scalability with Cassandra. In *IEEE conference on 10th World Congress on Services*, Anchorage, AK, 2014, 434-441. https://doi.org/10.1109/SERVICES.2014.81

[15] Li, Y. & Manoharan, S. (2013). A performance comparison of SQL and NoSQL databases. *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*. https://doi.org/10.1109/PACRIM.2013.6625441

[16] Tang, E. & Fan, Y. (2016). Performance Comparison between Five NoSQL Databases. *IEEE 7th International Conference on Cloud Computing and Big Data*. https://doi.org/10.1109/CCBD.2016.030

[17] Manoj, V. (2014). Comparative Study of NoSQL Document, Column Store Databases and Evaluation of Cassandra. *International Journal of Database Management Systems (IJDMS), 6*(4). https://doi.org/10.5121/ijdms.2014.6402

[18] Abramova, V., Bernardino, J., & Furtado, P. (2014). Which NOSQL database? A performance overview. *Open Journal Databases, 1*(2), 17-24.

[19] Tudorica, B. G. & Bucur, C. (2011). A comparison between several NoSQL databases with comments and notes. *IEEE Roedunet International Conference (RoEduNet)*. https://doi.org/10.1109/RoEduNet.2011.5993686

[20] He, Changlin. (2015). Survey on NoSQL Database Technology. *Journal of Applied Science and Engineering Innovation, 2*(2), 50-54.

[21] Tsuyuzaki, K. & Onizuka, M. (2012). NoSQL Database Characteristics and Benchmark System. *NTT Technical Review*. Retrieved from https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201212fa3.html

[22] Techopedia. (2018). Retrieved from https://www.techopedia.com/definition/26284/key-value-store.

[23] Han, J., Haihong, E., Le, G., & Du, J. (2011). Survey on NoSQL Database. In *IEEE 6th International Conference on Pervasive Computing and Applications (ICPCA)*, Port Elizabeth, 2011, 363-366. https://doi.org/10.1109/ICPCA.2011.6106531

[24] RIAK KV. (2018). Retrieved from http://basho.com/products/riak-kv/

[25] Redis database. (2018). Retrieved from https://redis.io/.

[26] Hbase database. (2018). Retrieved from http://hbase.apache.org/.

[27] Cassandra database. (2018). Retrieved from http://cassandra.apache.org/.

[28] Man Qi. Digital Forensics and NoSQL Databases. (2014). In *IEEE 11th International Conference on Fuzzy Systems and Knowledge Discovery*. https://doi.org/10.1109/FSKD.2014.6980927

[29] MongodB database. (2018). Retrieved from https://www.mongodb.com/.

[30] Neo4j database. (2018). Retrieved from https://neo4j.com/

[31] See https://www.aerospike.com/learn/what-is-a-key-value-store/

[32] Muhammad, Y. (2011). Evaluation and Implementation of Distributed NoSQL Database for MMO Gaming Environment. Uppsala University, Retrieved from http://uu.divaportal.org/smash/get/diva2:447210/FULLTEXT01.pdf.

[33] Haughian, G. (2014). Benchmarking Replication in NoSQL Data Stores. *Master's thesis*. http://www.doc.ic.ac.uk/teaching/distinguished-projects/2014/g.haughian.pdf

[34] Swaminathan, S. N. (2015). Quantitative Analysis of Scalable NoSQL Databases. *Master's thesis*. https://rc.library.uta.edu/uta-ir/bitstream/handle/10106/25497/SWAMINATHAN-THESIS-2015.pdf?sequence=1

[35] Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., & Sears, R. (2010). 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. https://doi.org/10.1145/1807128.1807152

Contact information:

Aimen Mukhtar RIMS, PhD student
Ankara Yıldırım Beyazıt University
Department of Computer Engineering
Ayvalı Mah., Gazze Cd. No: 7, 06010 Etlik- Keçiören/Keçiören/Ankara, Turkey
aymenarmess2015@gmail.com

Ahmet Ercan TOPCU, PhD, Asst. Prof.
College of Engineering and Technology
American University of the Middle East
Eqaila, Kuwait
ahmet.topcu@aum.edu.kw
and
Ankara Yıldırım Beyazıt University
Department of Computer Engineering
Ayvalı Mah., Gazze Cd. No: 7, 06010 Etlik- Keçiören/Keçiören/Ankara, Turkey
aetopcu@ybu.edu.tr