ISSN: 2723-9535

Available online at www.HighTechJournal.org

HighTech and Innovation Journal

Vol. 6, No. 3, September, 2025

Evaluating the Performance of NoSQL Databases for Big Data in Cloud Computing Environments

Ahmet E. Topcu [1*], Aimen M. Rmis [2], Yehia I. Alzoubi [3], Ali O. Çibikdiken [4], Dababrata Chowdhury [5], Ersin Elbasi [1], Mohamed Abdel-Maguid [5], Salem Almadhun [6]

[1] *College of Engineering and Technology, American University of the Middle East, Egaila 54200, Kuwait.*

[2] *Department of Computer Science, Faculty of Science, Alasmarya Islamic University, Zliten, Libya.*

[3] *College of Business Administration, American University of the Middle East, Egaila 54200, Kuwait.*

[4] *Department of Computer Engineering, KTO Karatay University, 42020 Karatay/Konya, Turkey.*

[5] *Faculty of Science, Engineering & Social Sciences, Canterbury Christ Church University, Canterbury CT1 1QU, United Kingdom.*

[6] *Department of Computer Science, Elmergib University, Al Khums, Libya.*

Received 14 May 2025; Revised 11 August 2025; Accepted 18 August 2025; Published 01 September 2025

Abstract

This study aims to evaluate the performance of NoSQL databases in distributed cloud computing environments, addressing the lack of comprehensive benchmarking in this domain. Specifically, it investigates MongoDB and Riak KV, two widely used NoSQL systems, across diverse cloud platforms, including Google Cloud, DigitalOcean, and OpenStack. Using the Yahoo Cloud Serving Benchmark, we designed and implemented a benchmarking model to measure key performance indicators, including latency, throughput, and scalability, under varying workloads and data sizes. The analysis revealed that MongoDB integrated with Google Cloud consistently outperformed other configurations, demonstrating superior throughput and lower latency in read and write operations. In contrast, Riak Key Value generally exhibited higher latency, especially in scan-intensive workloads. To support practical decision-making, a decision tree model was developed based on empirical findings to guide optimal selection of cloud computing platforms and databases. The proposed benchmarking framework is modular and extensible, allowing adaptation to other NoSQL technologies, cloud providers, and performance metrics. This research presents a novel, systematic methodology for evaluating NoSQL database performance in cloud environments, providing actionable insights for selecting high-performing, scalable solutions in big data applications. This modular design enables the addition of more database technologies, deployment options, and performance standards in the future, thereby supporting broader research and real-world applications in distributed systems and cloud computing.

Keywords: NoSQL; Cloud Computing; MongoDB; Riak KV; Big Data; Cluster.

1. Introduction

With the growth of Cloud Computing (CC) technology, traditional network models are failing to provide adequate services for handling the large volumes of data continuously generated at high speeds across various domains [1]. This data is increasingly unstructured or semi-structured, making it complex and heterogeneous. Effective processing and analysis remain a high priority, especially in areas such as telecom and transportation optimization, where a standard

* Corresponding author: ahmet.topcu@aum.edu.kw

http://dx.doi.org/10.28991/HIJ-2025-06-03-05

model for mobile users is needed to support business decisions [2]. The rise of IoT, multimedia, and social media has further increased the volume of unstructured data [3], intensifying the need for efficient software and hardware techniques to process it [4]. Similar requirements exist in engineering, business, science, healthcare, and society [5]. The emergence of numerous NoSQL databases and diverse CC applications has drawn global attention, with CC-driven big data now gaining interest from academia, industry, and governments [6, 7].

Despite the growing adoption of NoSQL databases, significant gaps persist in the literature. Prior studies have evaluated NoSQL databases, such as MongoDB, Cassandra, and Couchbase, using benchmarks like the Yahoo Cloud Serving Benchmark (YCSB) [3]. However, few have examined key-value stores like Riak KV in distributed big data contexts. Moreover, existing research often lacks systematic comparisons across multiple cloud platforms (e.g., Google Cloud, DigitalOcean, OpenStack) under diverse workloads, limiting insights into how cloud infrastructure impacts database performance [8-10]. The field of big data remains complex, with solutions heavily dependent on the technology and specific objectives, and there is a scarcity of practical, experimentally driven frameworks to guide database and cloud platform selection [11, 12]. For instance, while studies such as [3] these compared MongoDB's performance, they did not include Riak KV or evaluate multiple cloud platforms. Additionally, Weitzenboeck et al. [10] focused on priority-based modifications without a standardized benchmarking approach. Accordingly, this study aims to propose a framework for a big data cluster based on a cloud platform by answering the following research question:

RQ: What is the optimal technique for analyzing large datasets in cloud computing: Riak KV or MongoDB?

This research significantly enhances our understanding of CC and big data by introducing a specialized framework for evaluating NoSQL databases in cloud data centers. The key advantage of this framework is its comprehensive approach, addressing the lack of robust evaluation methodologies in distributed and parallel processing environments. It enables detailed performance analysis, including throughput and latency under various conditions, providing clear insights into the strengths and weaknesses of databases such as Riak KV and MongoDB [13-15]. Furthermore, the decision tree constructed from multiple experiments offers a practical tool for database selection and optimization. Importantly, the adaptable nature of the model means it can be extended to other databases and cloud platforms, making it a valuable, versatile tool for researchers and practitioners aiming to compare and optimize database technologies in diverse CC environments.

Table 1 summarizes the abbreviations that appeared in this paper. The following sections of this paper are organized as follows: Section 2 presents the research background and related literature. Section 3 discusses research methodology. Section 4 presents the experimental results and subsequent discussion. Section 5 presents the findings, outlines future research directions, and discusses research limitations. Conclusions are presented in Section 6.

Table 1. Table of abbreviations used in the paper

Abbreviation	Definition	Abbreviation	Definition
BSON	Binary JavaScript Object Notation	PaaS	Platform as a Service
CCP	Cloud Computing Provider	RDBMS	Relational Database Management System
CC	Cloud Computing	SQL	Structured Query Language
DB	Data Base	SSD	Solid-State Drive
IaaS	Infrastructure as a Service	SSH	Secure Shell
JSON	JavaScript Object Notation	VM	Virtual Machine
KV	Key Value	XML	Extensible Markup Language
NoSQL	Not only Structured Query Language	YCSB	Yahoo Cloud Serving Benchmark

2. Background and Related Literature

This section presents the background of the research. First, the CC platform is discussed, followed by an overview of big data, and then an overview of NoSQL, including Riak KV and MongoDB. Ultimately, the recent literature related to the focus of this study is reviewed.

2.1. Cloud Computing Platform Overview

CC involves service provisioning, where companies offer computer-based services to customers over a network, typically following a pay-per-use model. Major platforms like Google Cloud, Amazon EC2, and Microsoft Azure provide attractive services for running applications in the cloud [1, 16]. The CC model offers several benefits, including lower operating costs, minimal upfront investment, elasticity, higher scalability, easy access via the web, and reduced maintenance costs and business risks. This study focuses on three CC platforms: DigitalOcean, OpenStack, and Google Cloud [17].

DigitalOcean is popular with developers for its infrastructure as a service (IaaS) platform, offering private Virtual Machines (VMs) called "droplets" that can be configured in size, geographic area, data center, and operating system [18]. OpenStack, an open-source cloud software, manages a wide range of computing, networking, and storage resources and allows users to access on-demand resources [19, 20]. Google Cloud provides integrated cloud services, such as IaaS, Platform as a Service (PaaS), and serverless computing, built on the original Google App Engine framework to support web applications and data analytics. Each platform offers unique features and tools for managing and scaling cloud resources efficiently [21].

2.2. Big Data Overview

Big data is understood and defined as large amounts of data that require a management tool to process at the right time and speed [22]. It refers to various types of data, including unstructured, semi-structured, and structured data, that can be thoroughly analyzed to uncover valuable information. It is impossible to manage this data type using a Relational Database Management System (RDBMS) [23]. The engine of big data has become the source of growth and innovation in the industry, which depends on various other emerging technologies, including IoT, CC, and analytics [24]. In other words, it is a vital resource for optimizing global output due to its impact on software-intensive industries, healthcare, administration, and education. Day by day, new data emerges from photographs and videos taken, social media posts, and other sources. The growing complexity of data has made processing it with available DBMSs more challenging [25]. A solution recently proposed to handle the rapid growth of data resources is to utilize more effective hardware. However, this approach is insufficient since massive amounts of data still exceed hardware developments [11].

The nature of big data is often described using four significant characteristics, termed the 4 V's: volume, velocity, variety, and veracity [26]. Each of these four Vs has a unique impact on data analysis. Volume refers to the quantity of all types of data produced and continuously expanded across a broad spectrum of sources. Most datasets are too big to be stored and analyzed using standard RDBMS technology. As a result, deficiencies and weaknesses became more apparent in traditional databases. With this issue in mind, distributed systems became a critical emerging form of technology. For example, by 2020, 43 trillion gigabytes of data had been created, and about 2.3 trillion gigabytes are generated daily [27]. Variety refers to the diverse categories of data collected through social networks, smartphones, or sensors. These include audio, image, video, text, and data logs in unstructured both structured formats [28]. Velocity describes the speed at which data is generated, saved, analyzed, and visualized. In the era of big data, new information is produced in real-time [29]. Due to the absorption of supplementary datasets, previously archived data, or legacy collection sets, streamed data from multiple sources is introduced. For example, approximately 204 million emails are sent out every minute, 2.46 million items are shared on Facebook, 100 hours of video are uploaded to YouTube, and more than 4 million search queries are performed on Google [30]. Veracity covers the level of certainty that the data represents. Various amounts of data from differing sources move around rapidly. This requires organizations to ensure the accuracy of the information they are exposed to [31].

2.3. NoSQL Databases Overview

NoSQL is a more modern database category developed to address the limitations of RDBMSs in meeting the needs of big data. As a category, NoSQL describes various technologies that consider three aspects of big data: the exponentially increasing amount of data, the rate at which that data needs to be processed, and the most significant changes in data being created by today's applications [31]. Due to the constant growth in database size, efficient data access and information extraction have become recurring issues. There are many types of NoSQL databases, many of which differ in structure and efficiency, necessitating a reevaluation and analysis of their performance. Accordingly, several key issues need to be addressed in NoSQL databases for big data in CC. Research and development are necessary to overcome these challenges and ensure that NoSQL databases can meet the demands of big data in CC environments [32, 33].

Additionally, performance in various environments needs to be accurately measured, and standards should be established for comparing different NoSQL databases. Further research is required to develop improved solutions for addressing these challenges and to ensure that NoSQL databases can meet the demands of big data in the CC [31, 34]. Accordingly, this study assesses the performance of three major database environments (DigitalOcean, OpenStack, and Google Cloud), deploying MongoDB and Riak KV as NoSQL tools.

2.3.1. Riak Key Value

The two NoSQL databases evaluated in this research (i.e., MongoDB and Riak KV) have different feature sets. Each database provides a core set of characteristics that can be used as a basis for performance evaluation. Riak KV is considered an open-source professional alternative to Riak Enterprise DS. The Enterprise Edition includes multi-data center tracking, replication, and additional functionalities [35]. Riak KV achieves rapid efficiency and superior operational continuity through automated data sharing across the cluster. With a masterless design that ensures high reliability and virtually linear scaling utilizing ordinary equipment, capacity can be extended simply without a significant operational strain [36]. The nodes of Riak create a cluster. To get the full benefits of Riak, this cluster is partitioned into virtual nodes and formed into a ring. The ring is a 160-bit integer space divided into partitions of identical sizes. Each node (also known as a physical node) in the ring hosts several virtual nodes (Vnodes) [37].

2.3.2. Mongo Databases

MongoDB is a case of a document-oriented database. The primary storage components in a document database (such as MongoDB) are sets, rather than tables, in the case of RDBMS. These collections in MongoDB contain various JSON and BSON-based sub-documents or documents. Documents with similar structures are grouped into sets. It can be done as needed without any prior definition. MongoDB may include instances of documents, even arrays or lists of documents, within an instance or document [6]. Documents in MongoDB can belong to any basic data type (such as date, array, number, string, or subdocument). It has a unique variety of storage engines for a single deployment. This order facilitates data transfer between storage device technologies [38]. MongoDB has automatic sharing properties in which supplementary replica server nodes are added to the system. It is a high-speed database that not only provides indexing for primary attributes but also provides auxiliary attributes. This feature is available even in subdocuments. Using aggregation frameworks, Hadoop systems, and MapReduce, various collections can be compared [38]. To better understand the variances between Riak KV and MongoDB, this study analyzes several characteristics of the NoSQL databases, including replication, development language, data storage, and usage, among others.

2.4. Literature Review

As a critical property of CC, many studies have analyzed and evaluated big data in CC and distributed systems through NoSQL databases. This section reviews the most recent literature related to the focus of this study (i.e., NoSQL performance in distributed big data). Here, the focus was on the last five years, with a special emphasis on 2024 and 2023. Table 2 summarizes the previous literature findings, highlights the distinctions of this study, and addresses the research gap.

Table 2. Findings of the previous studies

Study	Year	Study domain	Findings
Beckermann (2025) [3]	2025	Evaluation of ACID-compliant NoSQL systems.	• Transactional YCSB enables realistic benchmarking of NoSQL systems that support ACID properties.
Ferreira et al. (2025) [8]	2025	Consistency level in NoSQL (Cassandra, MongoDB, and Redis).	• Increasing the consistency level often results in performance degradation.
Souza et al. (2025) [2]	2025	NoSQL eventual consistency.	• Proposes a stochastic Petri net-based model to estimate energy consumption in NoSQL systems using quorum techniques.
Araúo et al. (2024) [14]	2024	Reliability levels of NoSQL using Cassandra and Riak KV	• The findings indicate the effect of the consistency level on system performance.
Krishan et al. (2024) [5]	2024	Investigating how major NoSQL databases handle data consistency in distributed environments.	• Each database offers unique mechanisms and trade-offs for maintaining consistency. • Importance of flexible and scalable strategies for ensuring data consistency.
Martinez-Mosquera et al. (2024) [23]	2024	Integrating online analytical processing with NoSQL databases in Big Data environments.	• NoSQL databases have superior scalability and high availability advantages.
Andreoli et al. (2023) [13]	2023	Modifying MongoDB to support priority-based user performance using OS-level scheduling tools	• The solution proposed in the study lowers response times for high-priority users in mixed-priority scenarios.
Bansal et al. (2023) [15]	2023	Column, document, and Key-value NoSQL	• Architecture decisions influence NoSQL database performance.
Carvalho et al. (2023) [6]	2023	MongoDB, CouchDB, and Couchbase performance comparison using YCSB benchmark	• MongoDB has the highest performance, except for scan operations. • CouchDB had the highest scale-up if the number of threads varied.
da Silva & Lima (2023) [39]	2023	Docker Swarm's affordability using Cassandra, Citus, and HBase	• Cassandra's customizable reliability outperformed Citus and Hbase.
Gomes et al. (2023) [7]	2023	A generalized stochastic Petri net-based method to evaluate NoSQL-based cloud storage using quorum.	• Experimental results confirm the practical feasibility and effectiveness of the proposed approach.
Khan et al. (2023) [40]	2023	Clustering to segment NoSQL solutions, compare them by data models and CAP theorem, and analyze big graph applications across six domains.	• Offered a decision tree approach and a web tool to assist users in selecting NoSQL databases.
Kim et al. (2023) [41]	2023	MongoDB document storages using the GeoYCSB benchmark	• Couchbase and MongoDB can scale effectively under different workload combinations.
Nurhadi et al. (2021) [42]	2021	MongoDB, Casandra, Redis, and Neo4j performance comparison	• MongoDB is the best stable NoSQL for large amounts of data.
Seghier & Kazar (2021) [43]		MongoDB, Casandra, Redis performance comparison using YCSB tool	• Redis performed the best in the read operation. • MongoDB performed the best in the scan operation. • Casandra was the worst in the update operation.
Celesti et al. (2020) [44]	2020	Casandra, MongoDB, Hbase, and Neo4j performance comparison	• MongoDB provided the highest performance. • MongoDB also had the best NoSQL response time.
Rmis & Topcu (2020) [12]	2020	Evaluated Riak KV performance using distributed databases	• As data size expanded, so did the number of threads, resulting in higher throughput and lower latency. • Only read operations yielded high performance.
Present study	2024	MongoDB and Riak KV performance comparison for big data in cloud environments	• MongDB outperformed Riak KV in several tests, including read, insert, scan, and many modify operations. • Riak KV outperformed MongoDB in latency tests for read and insert operations, while latency results were very close in scan and modify operations. • Google Cloud platform outperformed other platforms, including OpenStack and DigitalOcean.

Beckermann (2025) presented Transactional YCSB, a new extension to YCSB that enables benchmarking of ACID-compliant NoSQL systems using workloads with multi-operation transactions. Using this tool, the paper evaluates the performance of FoundationDB, MongoDB, and OrientDB, providing insights into how these systems handle complex transactional workloads [3]. Ferreira et al. (2025) Studied NoSQL consistency levels, highlighting their impact on data synchronization, availability, and system performance trade-offs. Increasing the consistency level often results in performance degradation. For example, Cassandra experiences significant slowdowns when switching to strong consistency [8]. This trade-off is a critical consideration for database engineers when selecting appropriate consistency levels for their applications [8]. Souza et al. (2025) proposed a stochastic Petri net-based model to estimate energy consumption in NoSQL systems using quorum techniques. This model enables different consistency levels to be chosen, allowing designers to balance performance and availability based on system needs [2].

Araújo et al. (2024) suggested a stochastic Petri net-based approach to measuring the reliability levels of NoSQL database systems using the quorum methodology. The models incorporate varying degrees of consistency and duplicate nodes to predict the system's availability, performance, and the likelihood of obtaining the most recent data. The experimental findings demonstrate that this strategy is feasible in practice [14]. Krishan et al. (2024) reviewed the challenges of maintaining data consistency in popular NoSQL databases, including Redis, CouchDB, MongoDB, Neo4J, and others. By analyzing the unique features, consistency models, and architectural approaches of each database, the study highlights how different NoSQL technologies balance the trade-offs between consistency, availability, and scalability. The findings emphasize the value of flexible and scalable strategies for ensuring data consistency in evolving NoSQL systems, particularly as the digital landscape continues to transform [5]. Martinez-Mosquera et al. (2024) reviewed literature on integrating OLAP with NoSQL databases in Big Data environments. Their findings consistently highlight NoSQL databases' superior scalability and high availability advantages of NoSQL database [23].

Andreoli et al. (2023) proposed a modified MongoDB to support priority-based user performance using OS-level scheduling tools. Modified MongoDB NoSQL to support OS-level priority-based performance changes. Tests show that this strategy prioritizes response times for high-priority consumers in mixed-client scenarios. The suggested technique fails to compare the two approaches on a common basis adequately [13]. Bansal et al. (2023) investigated the performance of column, document, and key-value NoSQL databases in terms of response time, speed, and database size. The findings revealed that architecture decisions influence NoSQL database performance [15]. The suggested approach is reliable and identifies relevant criteria for informed decision-making. Carvalho et al. (2023) compared MongoDB, CouchDB, and Couchbase using the YCSB benchmark. The performance and scale-up were evaluated using YCSB workloads with varying records and thread counts [6]. The findings indicated that MongoDB has the highest throughput, except for scan operations. Furthermore, CouchDB achieved the highest scale-up when the number of threads varied. da Silva and Lima (2023) evaluated the affordability of Docker Swarm for cloud-based availability and scalability. The results find a balance between effectiveness and reproduction [39]. Cassandra's customizable reliability exceeded Citus and HBase in reproduction effectiveness [39]. This research identifies the ideal balance for low-power distribution systems. However, it lacks the efficiency and latency of the optimal technique investigated in the proposal.

Gomes et al. (2023) employed stochastic Petri nets to evaluate the stability of NoSQL systems. They assessed system response, accessibility, and data quality. This study examines key components of NoSQL and identifies many traits [7]. Khan et al. (2023) conduct clustering to segment NoSQL solutions, compare them by data models and the CAP theorem, and analyze big graph applications across six domains. These methods were grouped, contrasted, and applied to large graphs in six themes. They offered a decision tree approach and a web tool to help users select NoSQL databases. It is unique research that highlights essential aspects of NoSQL database deployments [40]. However, this work leaves open the possibility of asking a question concerning the impact of SQL databases on identical settings. The suggested analysis does not encompass this scope. Kim et al. (2023) evaluated Couchbase and MongoDB document storage using the YCSB benchmark. A complex dataset was utilized to assess Apache Accumulo with the GeoMesa system, demonstrating GeoYCSB's flexibility [41]. This is a utility-based examination of two distinct kinds of NoSQL databases. This study compares SQL and NoSQL, presents novel performance evaluation methods, and illustrates a novel approach for analyzing energy consumption and latency in industrial IoT [41] systems.

Nurhadi et al. (2021) investigated the NoSQL characteristics and capabilities using four indicators: efficiency, scalability, correctness, and complexity, to assess the suitability of NoSQL for various data types. They compared four databases: MongoDB, Cassandra, Redis, and Neo4j [42]. The experiment's results showed that MongoDB is the most stable NoSQL when dealing with large amounts of data. Seghier & Kazar (2021) conducted a comparative investigation of the performance of three regularly used systems, MongoDB, Redis, and Cassandra, using the YCSB tool to execute six custom workloads. Redis outperformed all other databases in read operations, and MongoDB outperformed Cassandra [43]. In scan operations, MongoDB outperformed both Redis and Cassandra, with Cassandra being more efficient than Redis. Cassandra made updating operations more challenging [43].

Celesti et al. (2020) tested four NoSQL databases: Casandra, MongoDB, HBase, and Neo4j. The NoSQL document-based strategy implemented using MongoDB proved to be the most performant option for managing large amounts of telemonitoring data. MongoDB also had the best NoSQL response speeds across all queries [44]. Rmis & Topcu (2020) evaluated the performance of the Riak KV database, which stores and retrieves large distributed data. The findings revealed that as the data size expanded, so did the number of threads, resulting in higher throughput and lower latency. Only read operations yielded high performance [12].

This study only examined literature that evaluated NoSQL on various platforms, particularly MongoDB and Riak KV. However, a large body of research compares relational databases, such as MySQL, to non-relational databases, including NoSQL options like MongoDB, Redis, Cassandra, HBase, Neo4j, and CouchDB. While these comparisons are helpful, this study focused only on evaluating NoSQL technologies for distributed big data, specifically MongoDB and Riak KV. Furthermore, the only identified work that examined Riak KV (i.e., [12]); however, in that paper, only Riak KV was assessed in big, distributed data without comparing it to other platforms. This study distinguishes itself by comparing two NoSQL platforms, MongoDB and Riak KV, in a big, distributed data environment.

3. Proposed Method

The performance behavior of the NoSQL database under various conditions is critical, as is the environment in which the database will run, and the best way to evaluate platforms. Using benchmarks (like parameters, expected data, and production configurations) and concurrent user workloads provides both businesses and IT with great insight into platforms [45]. This section provides an overview of the YCSB as a benchmark tool for running experiments. It also describes the methodology setting used to run the experiments. Figure 1 depicts the structure of this study. It is important to mention here that MongoDB and Riak KV were chosen because of resource limitations and the topic of our current study. These databases are important for having NoSQL distributed databases and were chosen because they were well-liked and pertinent to the goals of our study. Our provided framework allows us to add new databases. However, adding each database require to do installation, database managements and operation that need to have more insight operations.

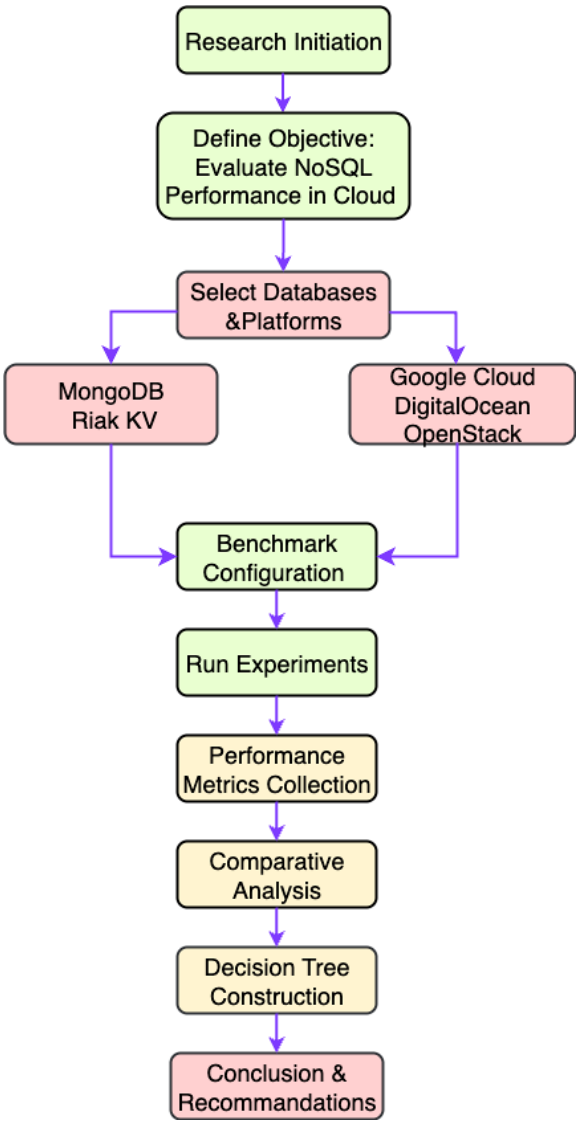


Figure 1. flowchart of the methodolgy

3.1. Yahoo Cloud Serving Benchmark

YCSB is an open-source tool designed by Yahoo to establish benchmarking clients for various NoSQL data stores. Usually, they do not come with a SQL interface. The YCSB tool employs a vector-based method to develop benchmarks that accurately reflect an application's specific performance. YCSB tools support only a subset of relational operations,

and instances of their implementation are typically very different from traditional RDBMS applications and are unsuitable for existing tools.

The YCSB kernel package was created to evaluate various system performance characteristics, depending on a set of workloads, to evaluate a system's appropriateness for different project types at various locales within the performance space [46]. The YCSB reporting framework conveys the total throughput of operations per second. The final tally is gauged by dividing the exact number of performed operations (writes and reads) by the workload time. The execution time refers to the duration between the commencement of the first operation in the workload and the completion of the final process, excluding the initial setup and final cleanup times. This execution time is likewise reported individually as the overall run time. Every workload execution generates a separate output file, which includes metadata and metrics. The YCSB architecture consists of the following components [41, 46]:

- Workload Configuration: The workload configuration specifies the parameters of the workload generator, such as the type and mix of operations, the number of threads, and the test duration.
- Workload: The YCSB workload specifies the types of operations that the client will perform on the database, including read, write, update, and delete. The workload can be customized to simulate different types of applications and data access patterns.
- Core: The YCSB core provides the benchmark's primary functionality, including the ability to generate random data, execute database operations, and measure performance metrics. The YCSB client generates the workload and sends requests to the database. The client can be configured to simulate various workloads, including read-only and write-heavy workloads.
- Database or Data Store: The database can be any NoSQL database that supports the YCSB API, such as Cassandra, MongoDB, or Riak.

Overall, the YCSB architecture is designed to be flexible and extendable, allowing users to customize the benchmark to their specific needs and test a wide range of NoSQL databases. By simulating various workloads and measuring performance metrics such as latency, throughput, and scalability, the YCSB can help users evaluate the performance of different NoSQL databases and choose the one that best meets their needs.

3.2. Experiment Settings

The experiments conducted in this study included three main components of CC: benchmarks and the NoSQL database. NoSQL data stores emerge as alternatives to traditional data processing methods, offering scalability while managing large amounts of data. The primary reason for choosing MongoDB or Riak is their ability to handle and manage massive data in CC easily. MongoDB is one of the first non-relational databases, so many researchers and companies use it in comparison with new databases to know the strengths and weaknesses of recent versions of other non-relational databases, including Riak. *Many MongoDB use cases can also be applied to Riak.* Riak, however, is very competitive in terms of basic features, such as design, and is significantly easier to manage [35].

The experiments were conducted in various CC environments using a benchmark to generate a common set of workloads and evaluate the performance of different databases. Figure 2 illustrates the experimental structure, which outlines the functions of the main components. The workload generator is responsible for fabricating a workload that simulates a set of operations that might be performed on a database or data store. The workload can be customized to simulate different types of applications and usage patterns. A benchmark file was prepared that consisted of various operations to be run on different NoSQL databases for testing purposes. The number of operations was equal to the record count for each database. The number of threads was identified to run the desired workload. This was done for varying proportions of data, from small to large. With each of these datasets, the number of chosen operations (read, update, delete, and others) was then run. All file operations can specify the following properties [47]:

- The database name to use can be specified on the command line.
- Thread count: number of client threads; alternatively, this may be defined on the command line.
- Record count: the number of records in the dataset at the start of the workload.
- Field count: the number of fields in a record.
- Field length: the size of any field.
- Min field length: the minimum size of each field.
- Read proportion: it determines the percentage of data reads.
- Update proportion: it determines the percentage of data updates.
- Insert proportion: it determines the percentage of data inserted.

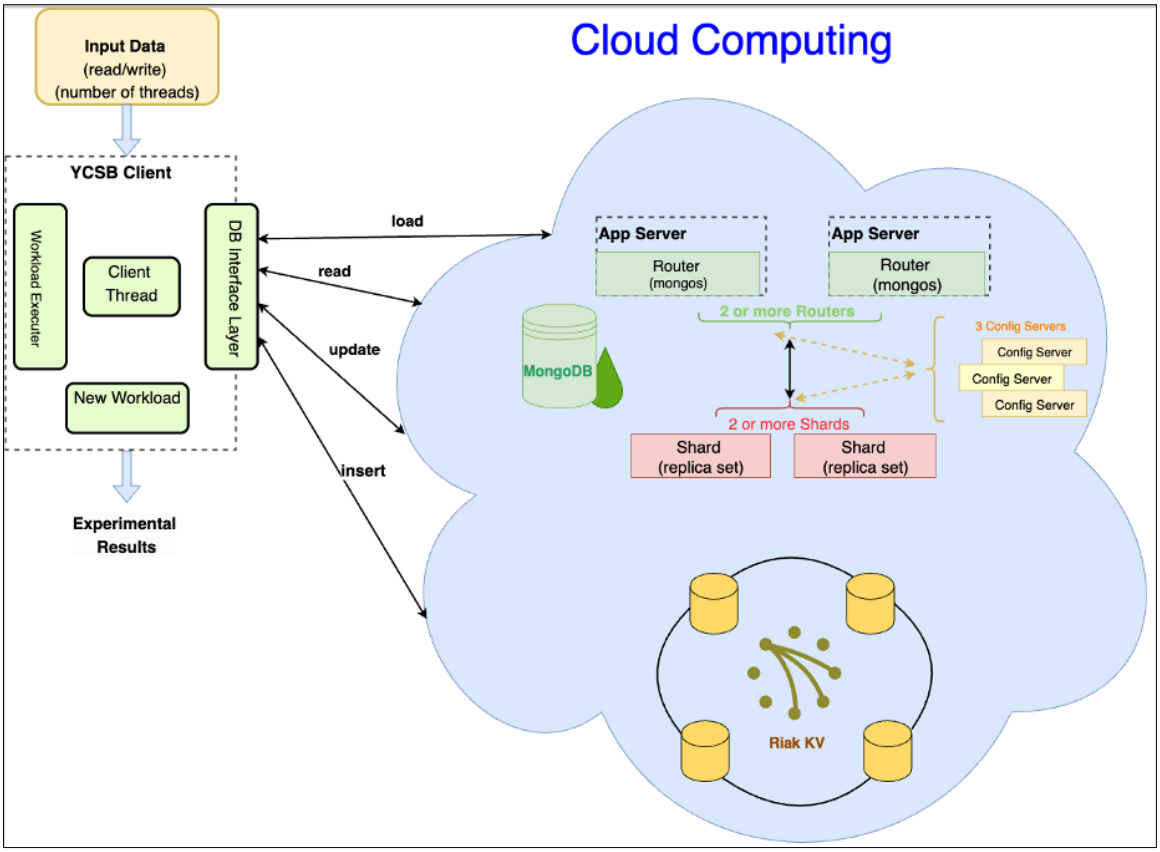


Figure 2. Experimental structure

In summary, the goal of this research is to create workload and data access patterns on the cluster that are consistent with real-world application workloads, and to monitor its performance in a CP. It also examined the performance of NoSQL databases (Riak KV, MongoDB) with large amounts of data and various workload operations (read, update, or a combination of reads and updates). Furthermore, the performance of NoSQL databases (throughput and latency) was tracked while data was read, added, and scanned throughout update operations. Additionally, a decision tree was developed that could be used by developers and database operators to select MongoDB and Riak KV.

To decrease the issue of limited control over communication quality in cloud environments, which could impact benchmarking results, several measures were implemented. First, the experiments were repeated multiple times to evaluate and verify the network communication quality. By conducting multiple runs of benchmarks and using statistical methods to average the results, the impact of transient network issues was reduced. This approach ensures these findings are more reliable and reflective of typical performance in cloud environments. We did use confidence intervals of 95%.

4. Experimental Results

In this study, various scenarios were conducted using different CC platforms according to these specific application areas. The study used VMs from three Cloud Computing Providers (CCPs). For each experimental analysis, the YCSB had a client with two sections: a set of scenarios and a workload generator. These scenarios are referred to as workloads and consist of a combination of read, update, and write operations performed on randomly selected records. In this study, VM was used in three CCPs. For each experimental analysis, the YCSB had a client with two sections: a) a set of scenarios and b) a load generator. These scenarios were referred to as workloads and consisted of a combination of read, update, and write operations. These workloads were also randomly selected. The predefined workloads are shown in Table 3.

Table 3. The predefined workloads

Workload	Read	Insert	Update	Scans	Read-Modify-Write
A	50	0	50	0	0
B	95	0	5	0	0
C	100	0	0	0	0
D	95	5	0	0	0
E	0	5	0	95	0
F	50	0	0	0	50

The default data size in the basic core varies depending on the type of basic application. From this benchmark, 100K, 5,000K, and 20,000K records were created. A record table was utilized, having ten fields per record. Each is identifiable by a primary key, such as the text "user412356". Every field is labeled as field_0, field_1, and so on. Each field's value is a random ASCII string containing a 1 KB record (10 fields, 100 bytes each, plus the key) and a configurable number of 12 threads. Each experiment consisted of 10,000 operations. The dataset is presented in Table 4.

Table 4. The dataset is used in experiments

Record count	Record size	Total size
10K	1 KB	10 MB
5000K	1 KB	5 GB
20000K	1 KB	20 GB

A series of tests were performed on Droplet Type A for all platforms used (i.e., Google Cloud Platform, DigitalOcean, and OpenStack). Droplet Type A: 4 vCPUs, 4 GB RAM, 40 GB SSD. Five cluster VMs were used to perform experiments in the noted environment, as explained in Table 5. The subsequent section presents and evaluates the results of the YCSB-generated load (100K, 5,000K, 20,000K) records. OpenStack environment with 5 VMs running Ubuntu 14.04.5 x64 with 4 GB RAM, size: 4 vCPUs/40 GB. The tested versions of the NoSQL databases are MongoDB version 4.0, Riak KV version 2.2.3, and YCSB 0.15.0. Moreover, the OpenStack version was initially released in February 2017. Ubuntu 14.04 was chosen to ensure compatibility with the 2017 OpenStack release used in our testbed environment (refer to Table 5). This version also supports reproducibility, aligning our setup with conditions from prior studies (e.g., [36]) to enable meaningful comparisons. One known constraint is that Ubuntu 14.04's kernel, version 3.13, lacks modern I/O schedulers, such as BFQ, which may potentially limit absolute performance. However, since our study focuses on relative performance comparisons between MongoDB and Riak KV, this limitation introduces minimal bias to the findings. To further address potential OS-induced variability, experiments were repeated multiple times, and average values were reported (see Section 3.2). This approach enhances the reliability of our results, ensuring that observed performance differences stem from the database systems themselves rather than kernel-level discrepancies.

Table 5. The environment specifications used in this study

Operating-system	Ubuntu- 14.04 (64-bit)
Memory	4 GB
CPU	4 vCPUs / 40 GB
Databases	Riak KV 2.2.3, MongoDB 4.0
Benchmark tools	YCSB 0.15.0
Clouds	OpenStack, DigitalOcean, Google

To compare the cost-effectiveness of different cloud providers, the pricing structure for virtual machine instances offered by Google Cloud Platform and DigitalOcean was investigated. This analysis focused specifically on the hourly cost of various instance types. Table 6 presents a selection of instance types from both providers, along with their corresponding hourly pricing. Based on the instance cost comparison, DigitalOcean typically offers lower hourly rates than Google Cloud Platform for the chosen instance types. However, it is crucial to consider other factors, such as performance, scalability, and additional features, when conducting a comprehensive assessment. These costs may vary due to factors like region, instance configuration, and usage patterns.

Table 6. Cost-effectiveness analysis

Instance type	Google Cloud Platform ($)	DigitalOcean ($)
Instance type 1	0.05	0.03
Instance type 2	0.08	0.06
Instance type 3	0.12	0.09
Instance type 4	0.16	0.12

To compare throughput and the loading rate, different-sized records were loaded (100K, 5,000K, and 20,000K), as illustrated in Figure 3. In the figure, the results focused on data loading between OpenStack, DigitalOcean, and Google Cloud, and it was apparent that the load time on OpenStack was higher than on DigitalOcean. Additionally, no significant difference was evident between MongoDB and Riak KV. MongoDB possessed a shorter insert time, regardless of the

number of records, compared to Riak KV. Loading time also appeared to increase with the increase in the number of records until it reached OpenStack Riak KV 12 and 11 for OpenStack MongoDB when inserting 20,000K records. DigitalOcean Cloud results were better than Google Cloud in almost all record sizes.

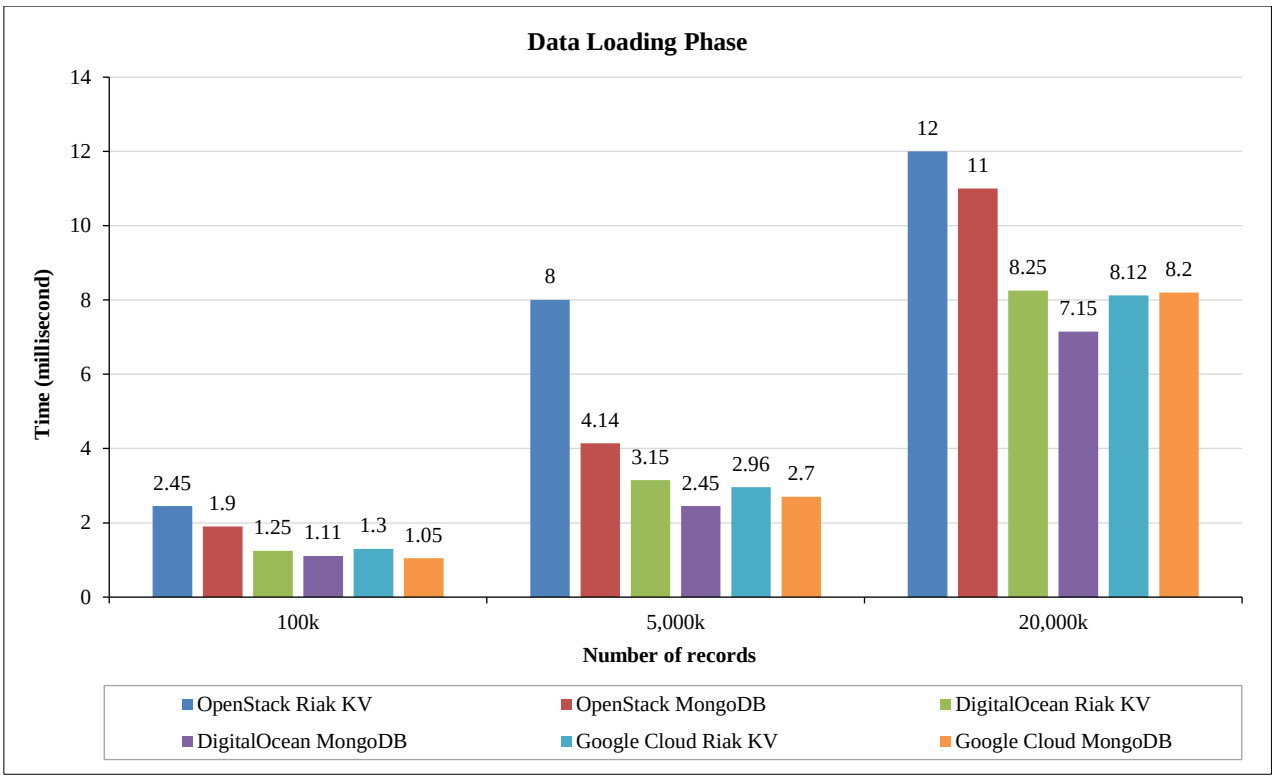
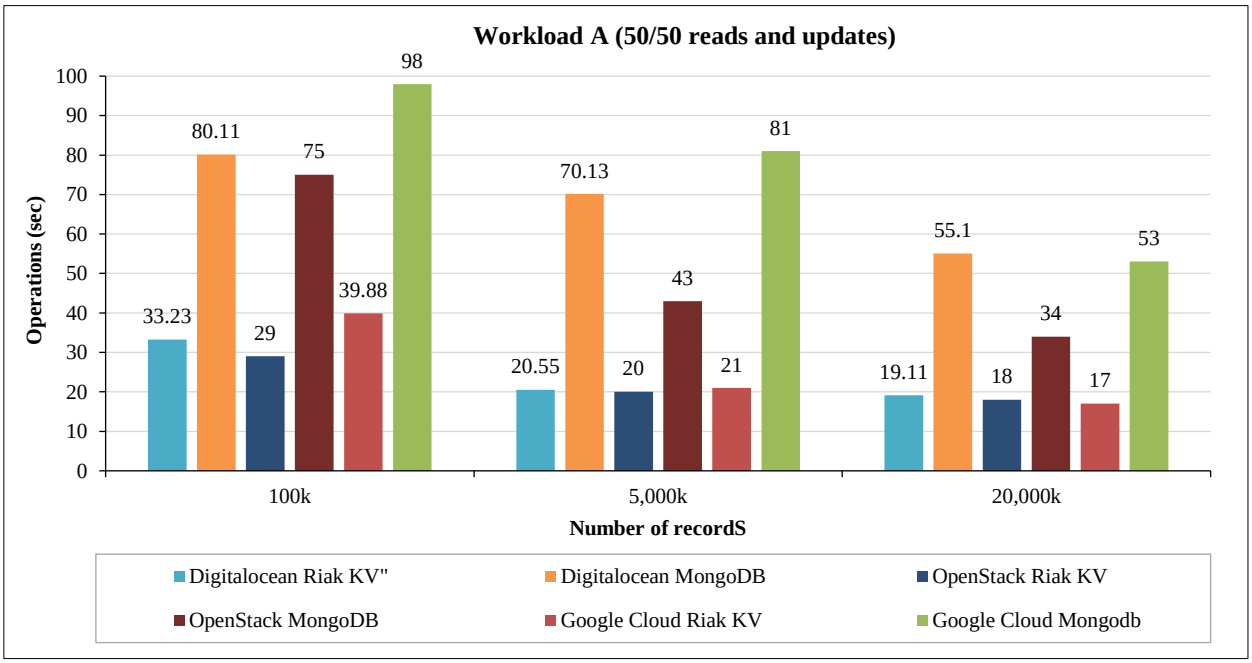


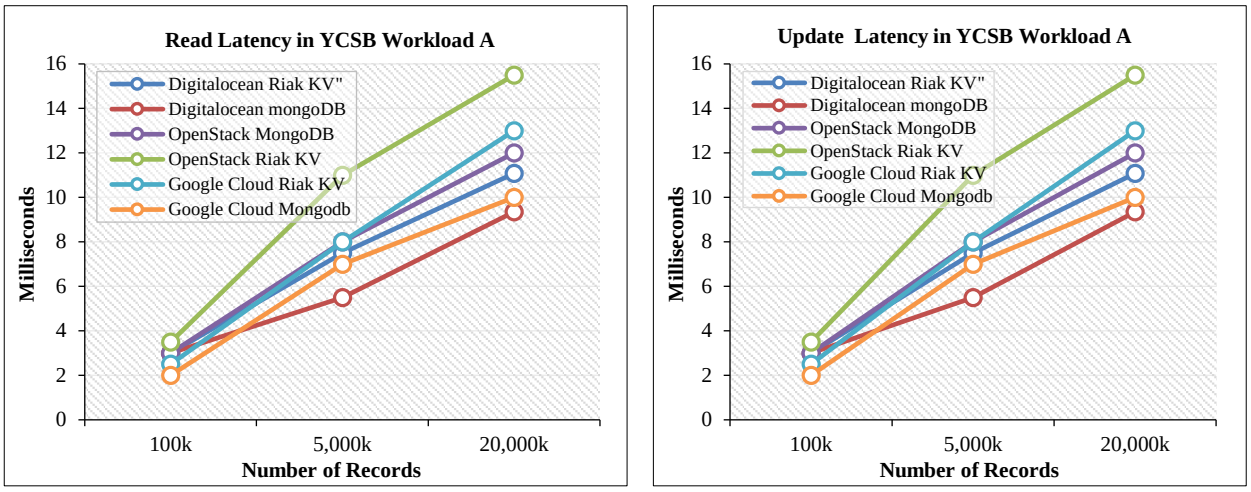
Figure 3. Data loading test

4.1. Workload A

This workload focused on read and update operations. Figure 4-a shows that MongoDB in Google Cloud achieved the best throughput with 100K and 5,000K keys at 98 and 81 operations/second, respectively. When the number of keys was increased to 20,000K for Riak KV, lower throughput was obtained compared to the other database, at 19.11 versus 18 operations/second, respectively. MongoDB in Google Cloud achieved the best throughput and the lowest latency for reads. The update operations showed that Riak KV in OpenStack provided the highest latency for updates, with the highest latency value of 15.5 milliseconds, as shown in Figure 4-b.



(a) Throughput performance

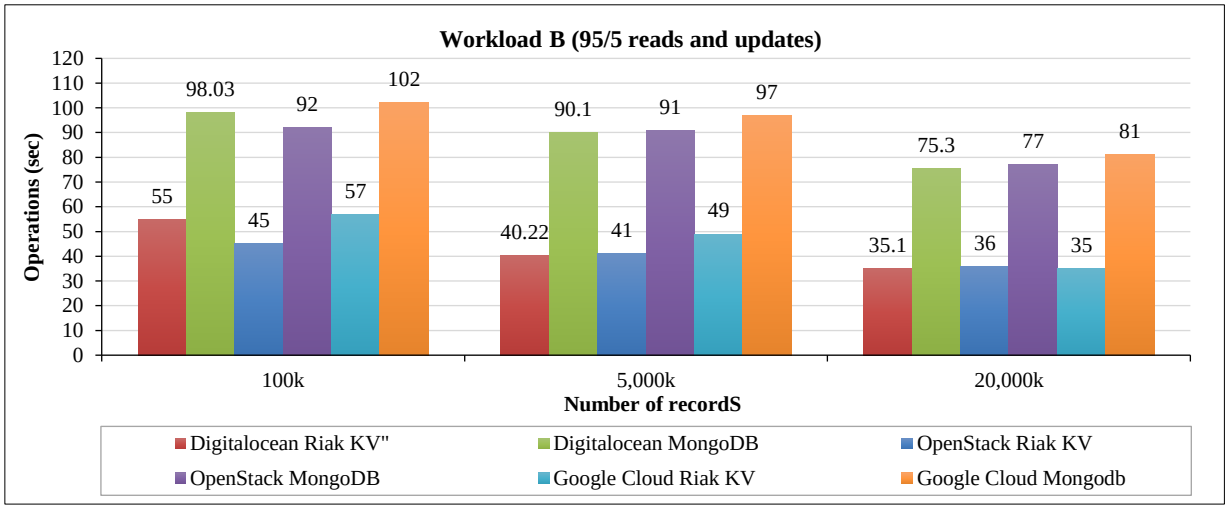


(b) Latency time

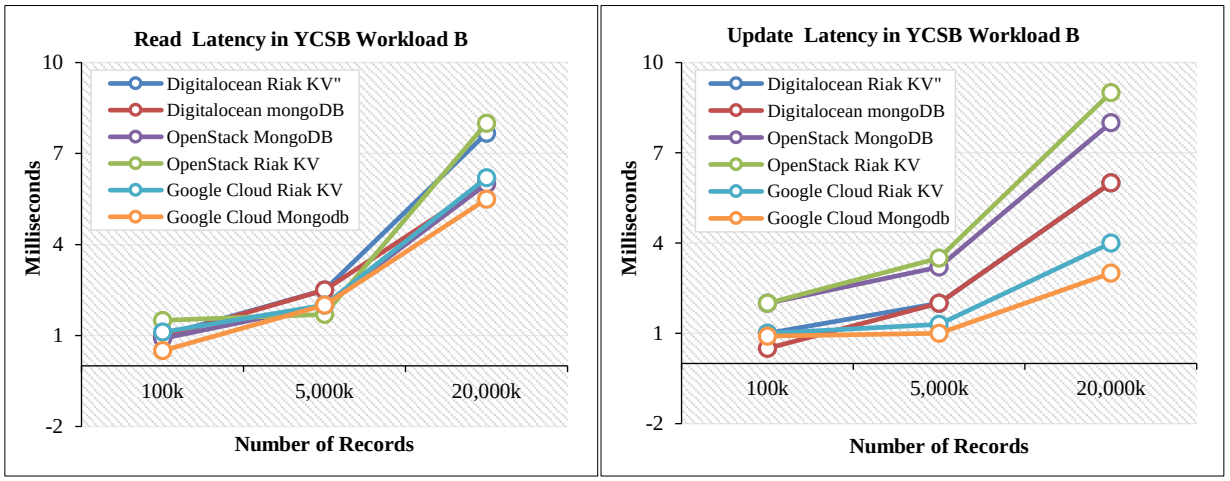
Figure 4. Throughput performance and latency time for the workload (A)

4.2. Workload B

This workload focused on read and update operations. Figure 5-a shows read and low update operations. MongoDB in Google Cloud peaked at 102, 97, and 81 operations per second in 100K, 5,000K, and 20,000K records, respectively. Riak KV achieved the highest latency results in OpenStack. In the read and update operations, it was 8 and 9 milliseconds, respectively. For updated operations, MongoDB in Google Cloud achieved the lowest latency for both read and update operations; see Figure 5-b for details.



(a) Throughput performance

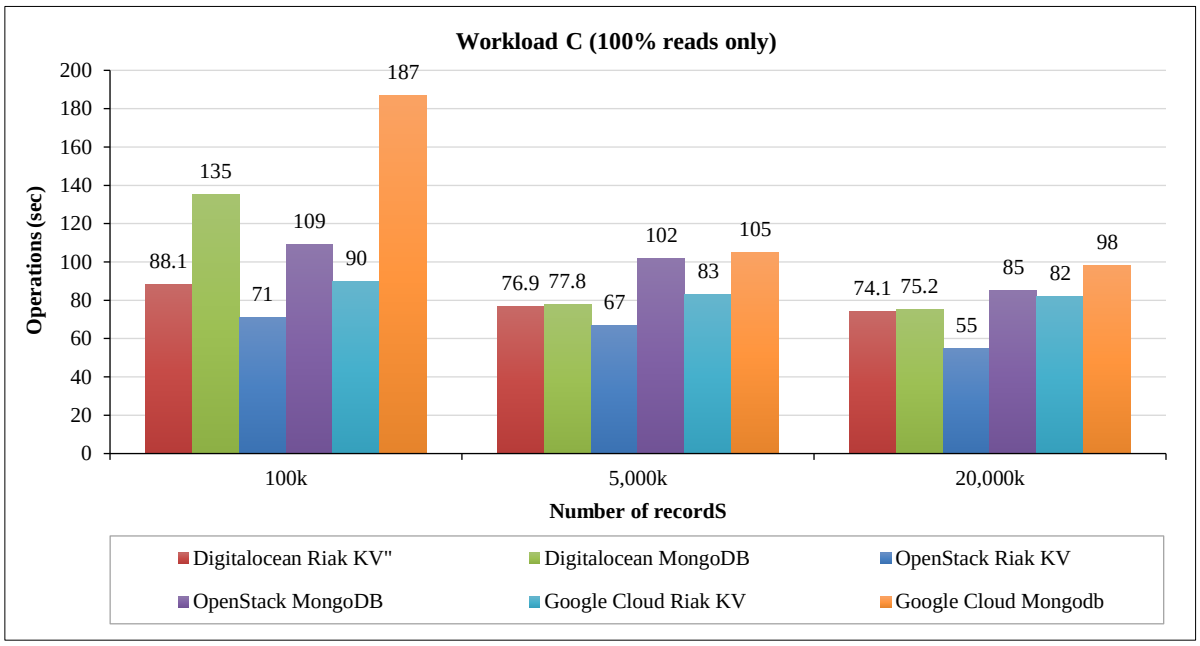


(b) Latency time

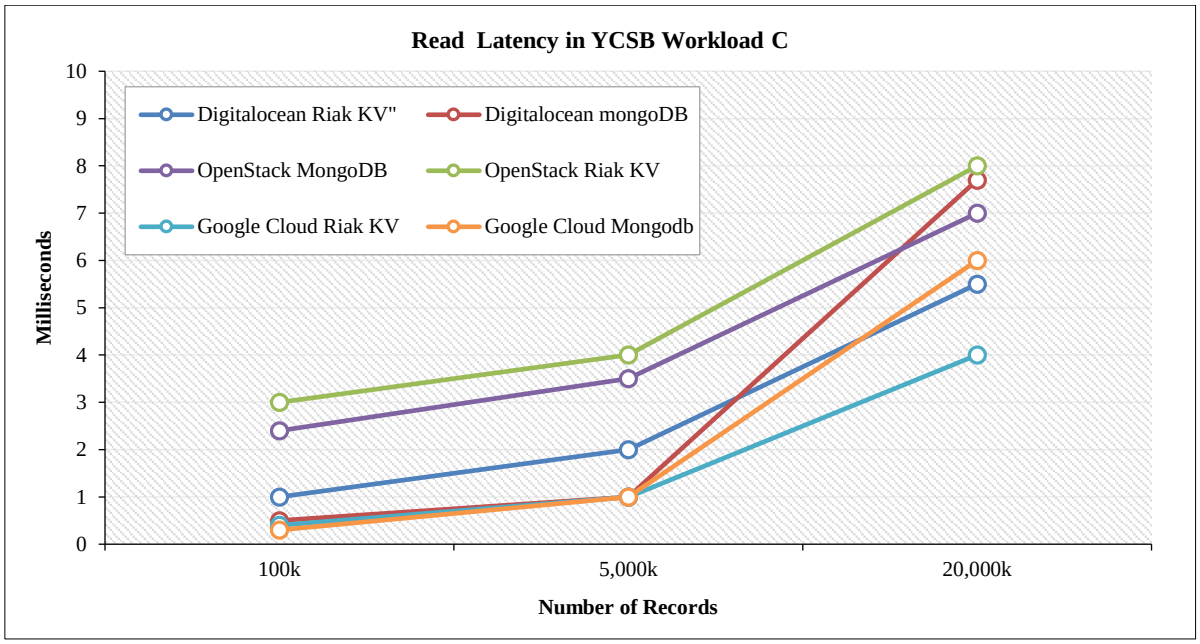
Figure 5. Throughput performance and latency time for the workload (B)

4.3. Workload C

In this workload, all the operations were read-only. The results are illustrated in Figure 6-a. Google Cloud MongoDB achieved the highest performance for all three records. The lowest latency value was achieved for Riak KV in Google Cloud with all records in general, while the highest was for Riak in OpenStack. Figure 6-b shows the results.



(a) Throughput performance

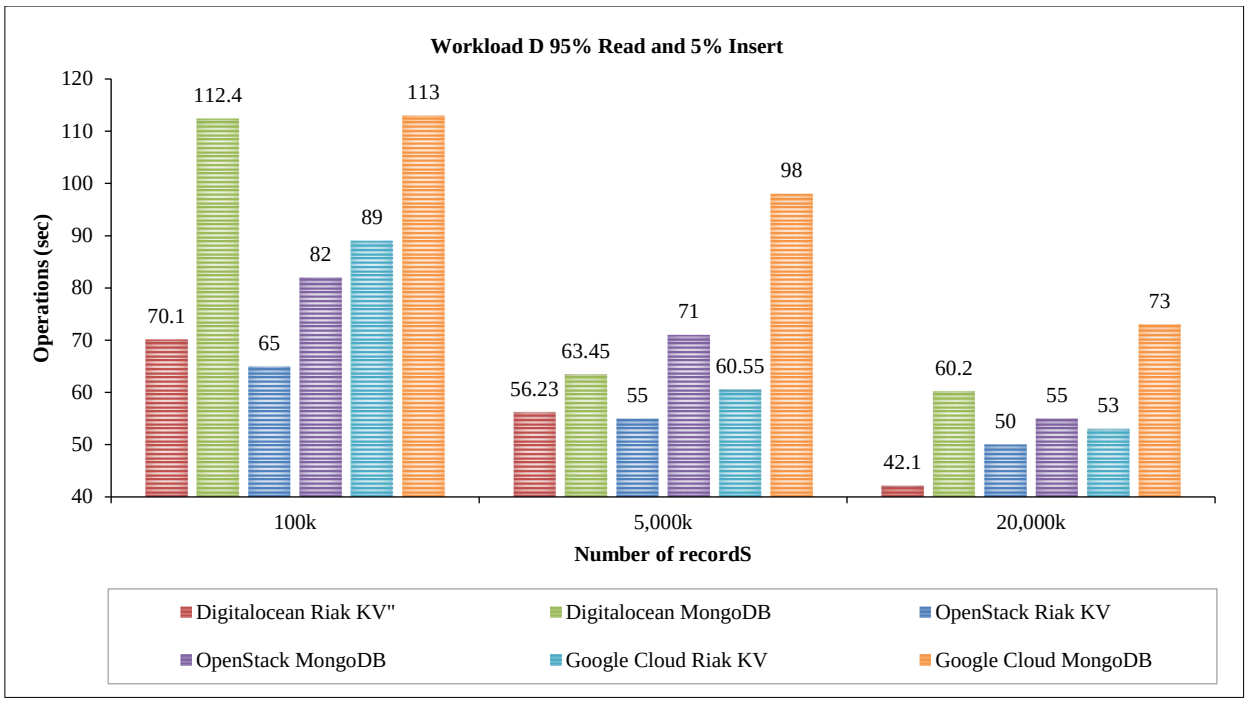


(b) Latency time

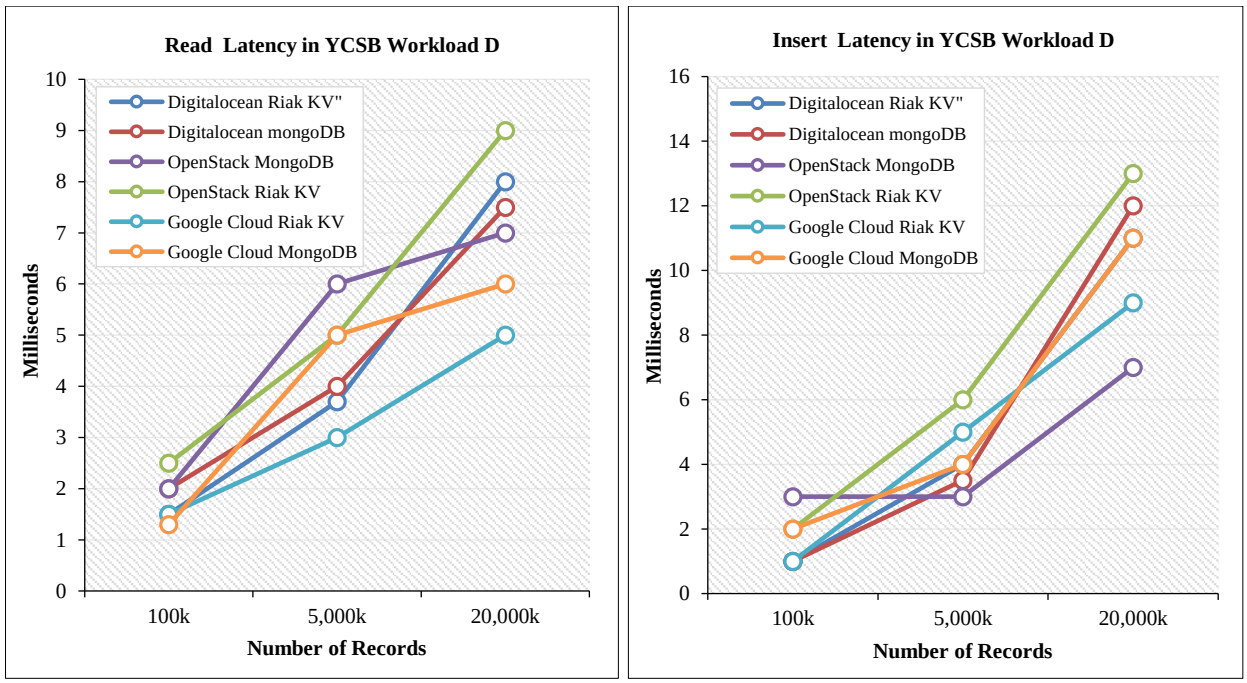
Figure 6. Throughput performance and latency time for the workload (C)

4.4. Workload D

This workload focused on read and insert operations. Workload D was run with 95% reads and 5% inserts. MongoDB in Google Cloud showed superiority for all database volumes. Its performance was high for all dataset sizes at 113, 98, and 73 operations/second. With an increase in data size, both MongoDB and Riak KV in Google Cloud began showing reduced throughput, but the performance of Riak KV was not even close to that of MongoDB, as shown in Figure 7-a. The latencies of both MongoDB and Riak KV in Google Cloud were close, although Riak KV had lower latency values. OpenStack Riak KV achieved the highest latency for both read and insert operations. However, the lowest latency for read operations was achieved by Google Cloud Riak KV, and for insert operations, it was achieved by OpenStack MongoDB. Results are shown in Figure 7-b.



(a) Throughput performance

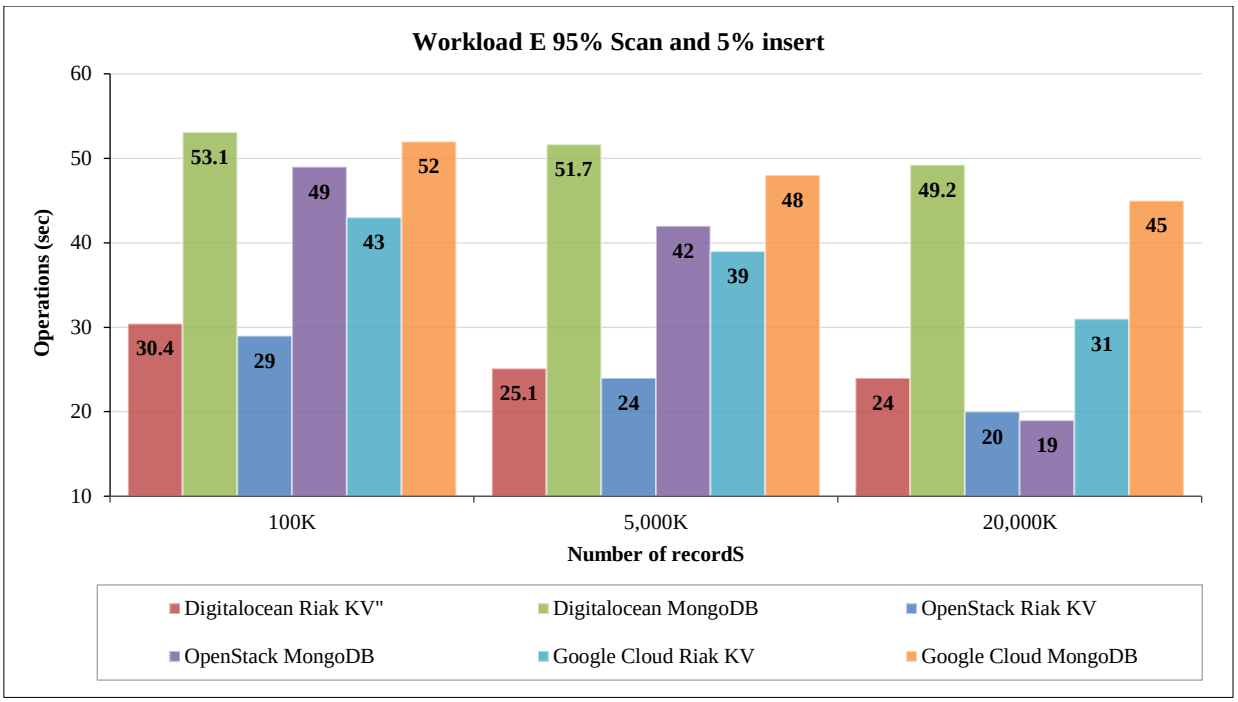


(b) Latency time

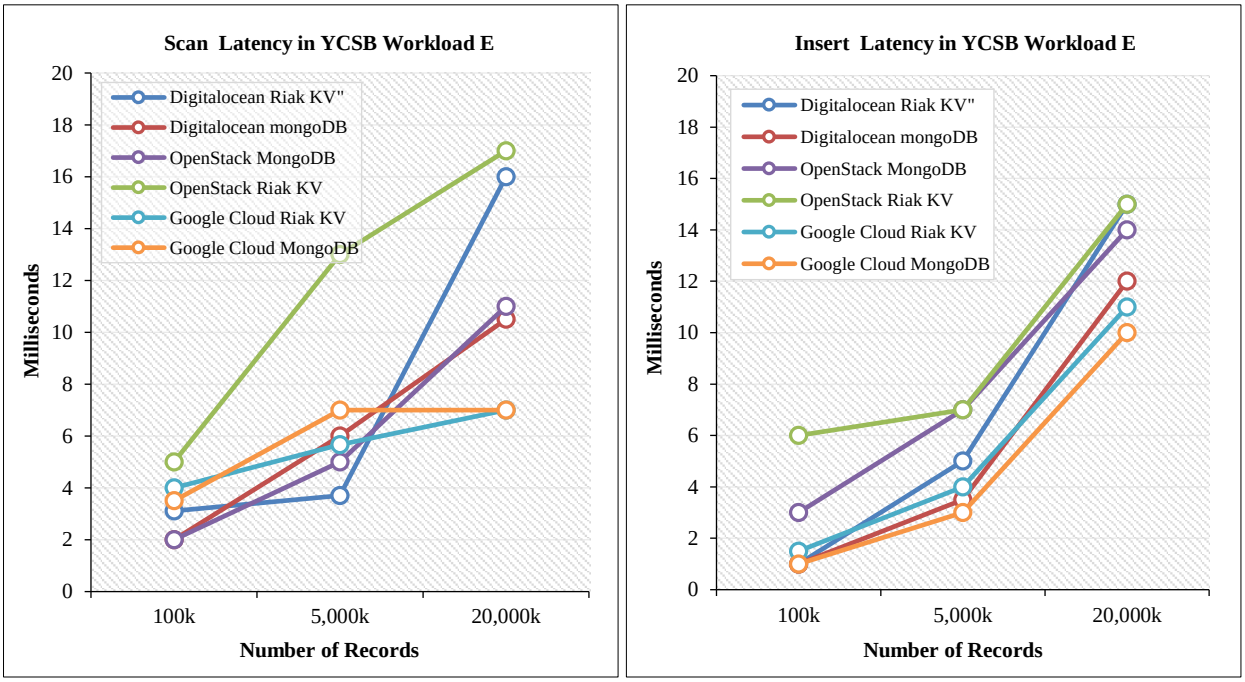
Figure 7. Throughput performance and latency time for the workload (D)

4.5. Workload E

This workload focused on insert and scan operations. DigitalOcean MongoDB performed better than others in this workload. The throughput was generally low, as observed in Figure 8-a, due to the scanning process. The performance failed to match that of previous tests. There were no more than 53.1 operations per second when the number of records was 100K. The latency results show very close values between MongoDB and Riak KV in Google Cloud, with slightly lower values for MongoDB. The highest latency was achieved by OpenStack Riak KV for both scan and insert operations. The lowest latency was achieved by Google Cloud MongoDB for insert operations and by Google Cloud Riak KV for scan operations. Figure 8-b shows the results.



(a) Throughput performance

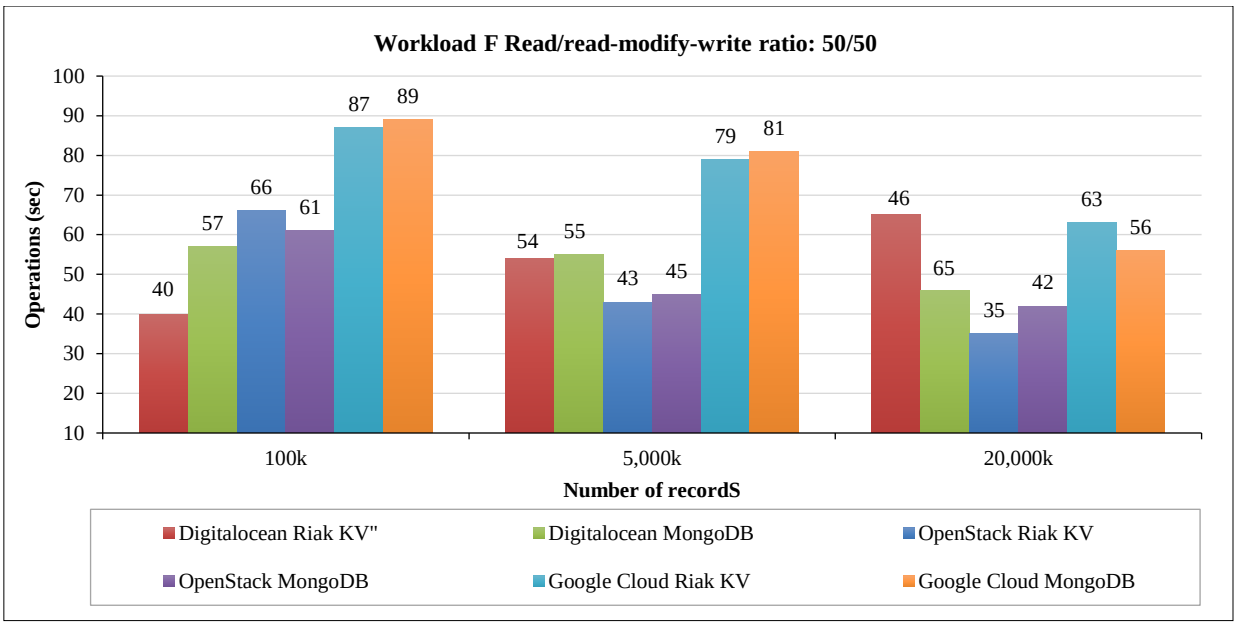


(b) Latency time

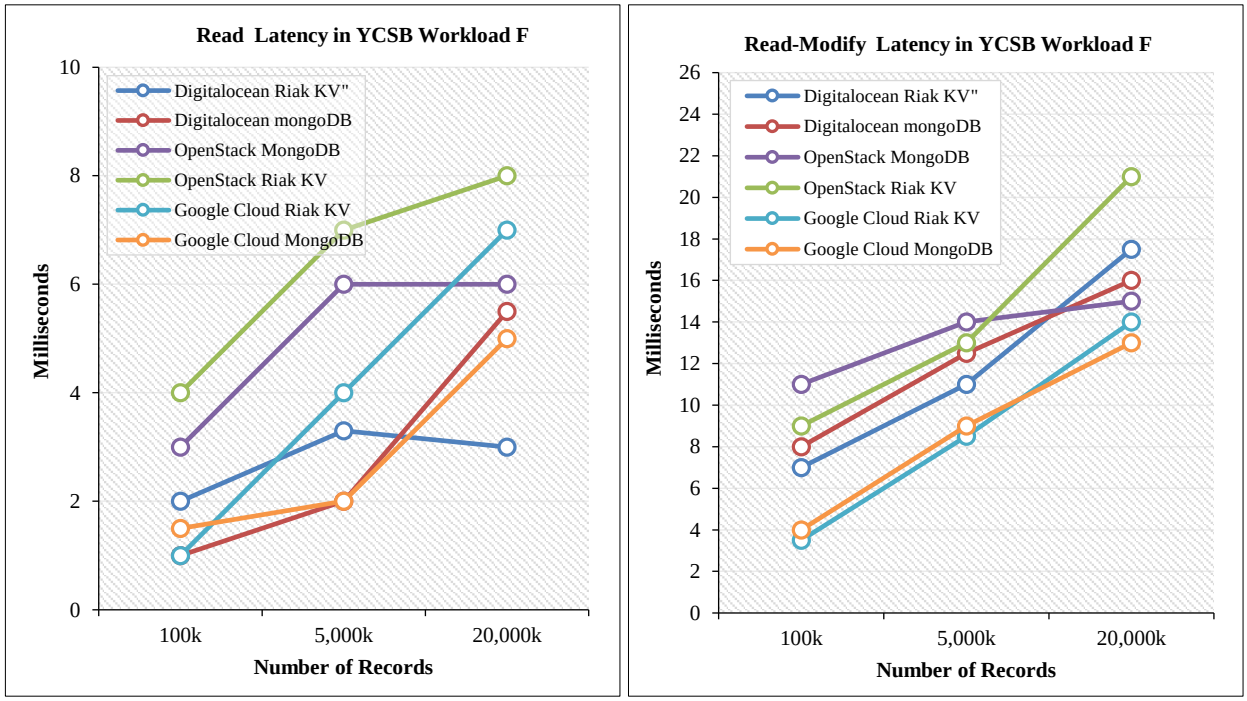
Figure 8. Throughput performance and latency time for the workload (E)

4.6. Workload F

This workload focused on read and read-modify operations. For this workload, the throughput was high for both databases, a likely consequence of the increase in data size. While Google Cloud MongoDB achieved higher values for both 1,000 and 5,000 reads than Google Cloud Riak KV, Riak KV in Google Cloud achieved higher performance when the record number increased to 20,000. The latency results were higher for read-modify than for read operation, as expected, because the process of reading and modifying takes more time than the process of reading only. For read operations, DigitalOcean Rika KV achieved the lowest latency overall, while for read-modify, Google Cloud MongoDB achieved the lowest, but the value was very close to Google Cloud Rika KV. See Figure 9 (a, b).



(a) Throughput performance



(b) Latency time

Figure 9. Throughput performance and latency time for the workload (F)

5. Discussion

This study aims to investigate the key performance factors of big databases in a CC environment. To do that, RQ1 (the key performance factors) and RQ2 (the difference between MongoDB and Riak KV) were answered in Section 4. This section discusses the results by providing a thorough evaluation of the results and suggesting a decision tree that can help identify the key considerations in adopting the NoSQL database and the CC environment that suit a more specific application. The research and practical implications are also discussed, as well as study limitations and future research directions.

5.1. Overall Evaluation

5.1.1. Performance Comparison

This study highlights MongoDB's superior throughput performance across a range of workloads when deployed on Google Cloud, confirming findings from earlier research (e.g., [6, 48, 49). Table 7 presents the total throughput for all

tested databases across all workloads. In Workload A (50% read, 50% update), MongoDB achieved 232 operations per second (ops/sec), outperforming Riak KV's 77.88 ops/sec. This performance advantage is attributed to MongoDB's document-oriented architecture, which leverages BSON storage and in-memory processing to optimize mixed operations. Similarly, in Workload C (100% read), MongoDB delivered 390 operations per second (ops/sec) compared to Riak KV's 255 ops/sec, benefiting from efficient indexing and caching mechanisms that minimize read latency. An exception was noted in Workload E (scan-insert), where Riak KV exhibited lower latency (5.5 ms) than MongoDB (6.1 ms), due to its distributed hash table structure, which performs well in scatter-gather queries.

Table 7. The sum of throughput (operations/second)

Workload		A	B	C	D	E	F
MongoDB	DigitalOcean	205.34	263.1	288.02	236.05	154.01	158
	OpenStack	152	260	296	208	110	148
	Google Cloud	232	280	390	284	145	226
Riak KV	DigitalOcean	72.89	130.32	239.08	168.33	79.57	159
	OpenStack	67	122	193	170	73	144
	Google Cloud	77.88	141	255	202.55	113	229

Overall, MongoDB outperformed Riak KV in Workloads A, B, C, and D on Google Cloud and showed better results than on DigitalOcean and OpenStack in most cases. However, for scan and read-modify workloads (E and F), MongoDB performed more efficiently in the DigitalOcean environment. MongoDB's scale-out architecture, combined with its flexibility, consistency, fault tolerance, and agility, positions it as a highly adaptable solution for modern data-intensive applications. Its ability to manage evolving data schemas, support rapid development, and ensure low downtime further strengthens its appeal for developers working in dynamic cloud-based environments. These results reinforce MongoDB's value in Big Data analytics and operational scenarios requiring high performance and scalability.

In write-heavy workloads (D and F), MongoDB demonstrated consistent performance, maintaining stable throughput around 200 operations per second. This stability is largely due to its use of journaling and replication, which enhance data durability and write efficiency. In contrast, Riak KV exhibited greater variability, with latency spikes fluctuating by up to ±20%. This inconsistency is attributed to its quorum-based write mechanism, which, while designed for high availability, can introduce delays under certain conditions [12]. For read-modify-write operations (Workload F), Riak KV performed nearly on par with MongoDB, achieving 229 ops/sec compared to MongoDB's 226 ops/sec. This close performance gap highlights Riak KV's effectiveness in handling transactional tasks, largely due to its use of atomic counters—a feature that supports consistent updates across distributed nodes. These findings suggest that while MongoDB is generally more stable in high-write scenarios, Riak KV can be a strong contender for specific transactional workflows. The results also emphasize the importance of understanding workload characteristics when selecting a NoSQL solution, as different architectures yield varying performance outcomes depending on the type of operation.

Table 8 compares the latency between MongoDB and Riak KV in the three cloud environments. MongoDB in Google Cloud achieved the lowest latency in workload C (read operation), workload E (scan and insert operations), and workload F (read and read-modify operations) compared to Digital Ocean and OpenStack environments. However, the latency results for workload A (read 50% and update 50% operations), workload B (read 95% and update 5% operations), and workload D (read and insert operations) showed mixed values. In workload A, DigitalOcean demonstrated lower latency compared to Google Cloud and OpenStack. Moreover, in workload B, Google Cloud showed the lowest latency in update operations, while OpenStack showed the lowest latency in read operations. Similarly, in workload D, Google Cloud showed the lowest latency in read operations, while OpenStack showed the lowest latency in insert operations.

Table 8. Mean latency (operations/second) R: Read; U: Update; I: Insert; S: Scan. RM: Read-Modify

	A	B	C	D	E	F
Workload: MongoDB						
DigitalOcean	R = 2.2, U = 5.9	R = 3.1, U = 2.8	R = 3.0	R = 4.5, I = 5.5	S = 6.1, I = 5.5	R = 2.8, RM = 12.1
OpenStack	R = 5.6, U = 7.6	R = 2.4, U = 4.4	R = 4.3	R = 5.0, I = 4.3	S = 6.0, I = 9.3	R = 5, RM = 13.3
Google Cloud	R = 2.4, U = 6.3	R = 3.1, U = 2.6	R = 2.4	R = 4.1, I = 5.6	S = 5.8, I = 4.6	R = 2.8, RM = 8.6
	A	**B**	**C**	**D**	**E**	**F**
Workload: Riak KV						
DigitalOcean	R = 2.5, U = 7.1	R = 3.7, U = 3.0	R = 2.8	R = 4.4, I = 5.3	S = 7.6, I = 7.0	R = 2.7, RM = 11.8
OpenStack	R = 8.5, U = 10.7	R = 3.7, U = 4.8	R = 5	R = 5.5, I = 7.0	S = 11.0, I = 8.0	R = 6.3, RM = 14.3
Google Cloud	R = 2.6, U = 7.8	R = 3.1, U = 2.1	R = 1.8	R = 3.1, I = 5.0	S = 5.5, I = 5.5	R = 4.0, RM = 8.6

Riak KV in Google Cloud exhibited lower latency values compared to Riak KV in DigitalOcean and Riak KV in OpenStack across workloads B, C, D, E, and the read-modify operation in workload F. However, Riak KV in DigitalOcean showed the lowest latency in workload A and for read operations in workload F. On the other hand, when comparing MongoDB and Riak KV, Riak KV achieved lower latency in workloads C and D, while MongoDB achieved lower latency in workloads A, B, E, and F in general.

5.1.2. Cloud Platform Efficiency Comparison

Google Cloud Riak KV showed higher performance when compared to other cloud environments for all workloads. This again emphasizes the capabilities of Google Cloud. Moreover, Google Cloud MongoDB showed higher performance values compared to Google Cloud Riak KV in all workloads, except for workload F (read and read-modify), where Riak KV showed slightly higher performance. Google Cloud emerged as the dominant environment for NoSQL performance in this study, particularly in supporting MongoDB workloads. For instance, in Workload D (read and insert), MongoDB on Google Cloud achieved 284 operations per second (ops/sec), outperforming its performance on DigitalOcean (236 ops/sec). This performance advantage is mainly due to Google Cloud's robust infrastructure, including its proprietary fiber-optic network, which significantly reduces latency [21]. Additionally, the optimized virtual machine configurations, such as instances with four vCPUs and 4 GB of RAM, are well-suited to MongoDB's memory-mapped storage engine, ensuring efficient resource utilization.

In contrast, OpenStack demonstrated the weakest performance, with the highest latency observed for Riak KV in Workload F (14.3 ms). This underperformance is likely linked to the absence of proprietary optimizations found in commercial cloud platforms, which can negatively impact input/output (I/O) throughput [19]. Notably, Riak KV also performed better on Google Cloud than on other platforms across all workloads, reinforcing Google Cloud's technical advantage. However, MongoDB consistently outperformed Riak KV within the Google Cloud environment in all workloads except for Workload F (read and read-modify), where Riak KV held a slight edge [38]. These results underscore Google Cloud's effectiveness in handling high-performance NoSQL deployments. Its advanced networking, infrastructure tuning, and optimized resource provisioning make it a compelling choice for scalable, low-latency, data-intensive applications. The findings demonstrate that both MongoDB and Riak KV benefit from Google Cloud's capabilities, with MongoDB showing especially strong performance across a broad range of analytical workloads.

5.2. Suggested Decision Tree

Based on the results discussed in Sections 4 and 5.1, a decision tree was developed to help developers and researchers determine whether to use MongoDB or Riak KV. Some difficulties and possible solutions for overcoming problems in selecting the correct database and the right CC environment, considering various variables, are addressed here. The decision tree, presented in Figure 10, is divided into six sections based on the operation type, as discussed below.

- The top split in the tree is workload A, which comprises 50/50 reads and updates. The highest performance was achieved with MongoDB in Google Cloud. Riak KV in OpenStack may be avoided. MongoDB in DigitalOcean has the lowest latency for reads and updates.
- The second split, for workload B, yields the same results as for workload A; however, the difference in low latency represents the most suitable option. MongoDB in Google Cloud is recommended for both read and update operations.
- Workload C focuses on read operations only. The highest performance was achieved with MongoDB in Google Cloud. Riak KV in Google Cloud has the lowest latency for read operations. However, Riak KV in OpenStack may be avoided for both low performance and high latency for read-only operations.
- In the fourth split of the decision tree, for workload D, Google Cloud gives MongoDB the highest performance and the lowest latency with read operations. MongoDB in OpenStack achieves the lowest latency for insert operations.
- In workload E, short ranges of records are queried. The highest performance was achieved by DigitalOcean MongoDB. Most of the poor results in OpenStack with Riak KV are due to the limited support for this cloud and the difficulty in managing it. OpenStack with Riak KV achieved the highest latency for both scan and insert operations, while Google Cloud Riak KV achieved the lowest latency for scan operations, and Google Cloud MongoDB achieved the lowest latency for insert operations.
- Workload F involves the client reading a record, modifying it, and then writing back the modifications. Google Cloud also achieved the highest performance and lowest latency for both read and read-modify operations, while OpenStack Riak KV again achieved the lowest performance and the highest latency for both read and read-modify operations.

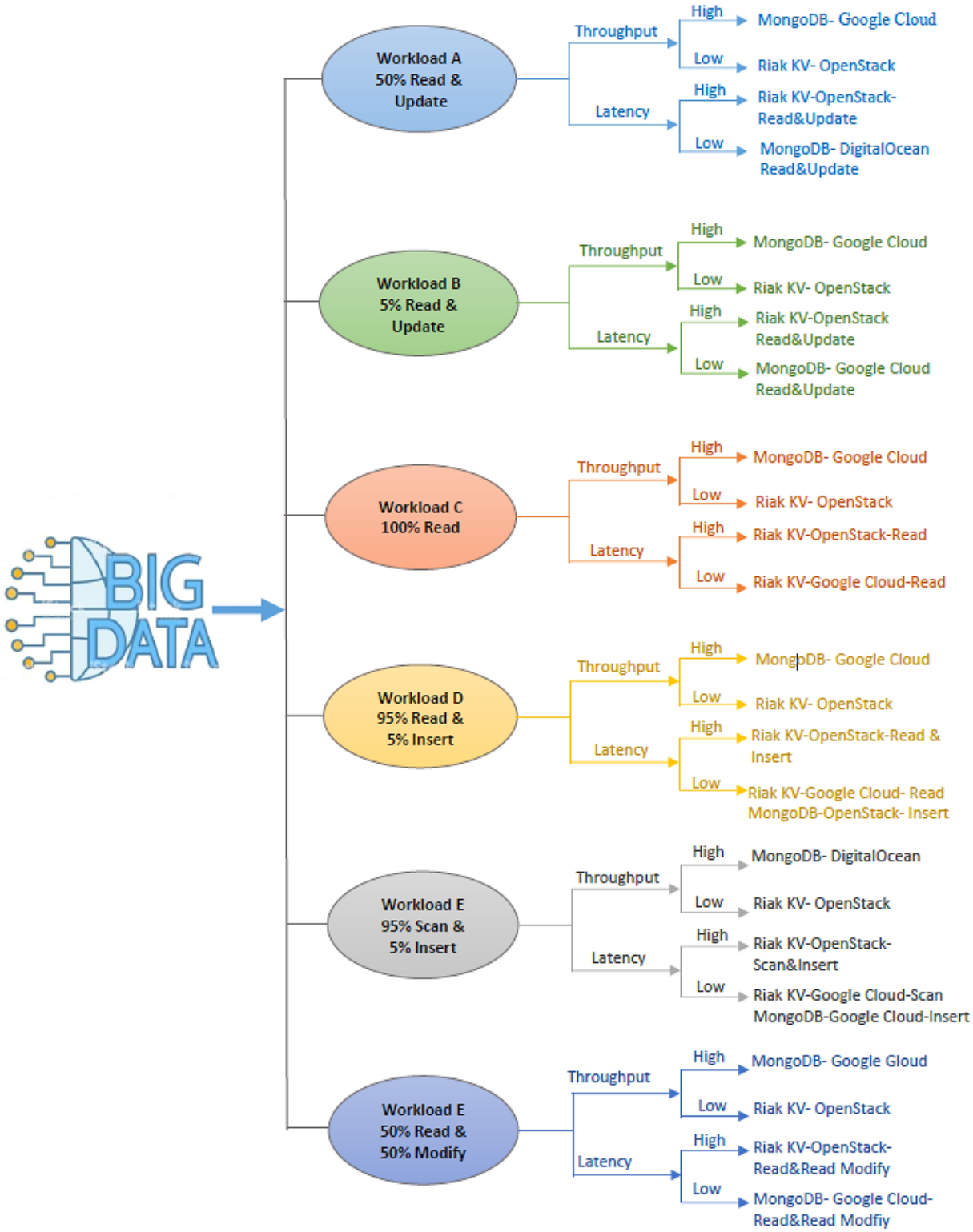


Figure 10. Decision tree for evaluating and testing in CC platforms

5.3. Research Implications

Although numerous studies have assessed the performance of NoSQL databases, including MongoDB and Riak KV, under diverse conditions, there is presently no direct research that examines database performance in a model identical to or closely resembling ours, especially within cloud environments utilizing Google Cloud, DigitalOcean, and OpenStack. This study offers a comparative analysis of MongoDB and Riak KV, emphasizing performance measures such as throughput and latency within a cloud-based context. Although other research (e.g., [42, 43, 48, 49]) has juxtaposed MongoDB with alternative NoSQL systems, such as Cassandra, Redis, and Neo4j, none have explicitly concentrated on OpenStack-based infrastructure or employed the identical benchmarking methodology used in this study.

The findings of this study indicate that MongoDB performed better than Riak KV. These results are in line with previous findings, such as [48, 49] those reported by, who reported that MongoDB can achieve high performance compared to Cassandra, HBase, and Microsoft SQL. Additionally, the findings validated those of [42] those who noted that MongoDB's performance was higher than that of Cassandra, Redis, and Neo4j. This is partly because MongoDB's registration conversion is copied into memory, which enhances the reading rate [43]. However, these results opposed the finding [43] that Redis outperformed all other databases in read operations. This may be because Redis stores and retrieves information using volatile memory. Given the consistent performance superiority of MongoDB over Riak KV, further investigation is warranted to examine the specific architectural and operational factors that contribute to MongoDB's efficiency. Future research could delve into the intricate details of MongoDB's memory management and query optimization strategies to uncover the underlying mechanisms that drive its high performance.

The findings also support the findings of Carvalho et al. [6], who reported that MongoDB has the best runtime, except for workloads constituted by scan operations. The results showed that in scan operations, Riak KV showed less latency than MongoDB. However, this opposed the findings reported by Seghier et al. [43] MongoDB, which performed the best in scan operations. The discrepancy in findings regarding MongoDB's latency in scan operations highlights the need for further investigation into the factors that influence scan operation efficiency across different database systems. Future research could focus on analyzing the specific configurations, indexing strategies, and data access patterns that contribute to outcomes in scan operations. Understanding these nuances can provide valuable insights into optimizing database performance for specific types of workloads.

The findings confirm that Google Cloud performed better than other cloud platforms, particularly in conjunction with MongoDB. Google Cloud has a significant advantage in its relationship with Google; its ability to access Google’s private fiber networking infrastructure facilitates stronger performance compared to standard network infrastructures [50]. These results indicate Google Cloud's superior performance and highlight the potential influence of cloud platform infrastructure on database performance. This suggests a need for further investigation into the specific technical capabilities and optimizations offered by cloud providers, particularly in terms of networking infrastructure and resource allocation mechanisms [51]. Future research could explore the underlying mechanisms and architectural features that contribute to Google Cloud's performance advantages, providing insights into best practices for leveraging cloud platforms to optimize database performance.

5.4. Practical Implications

For organizations seeking optimal database solutions for their applications, the empirical evidence suggesting MongoDB's superior performance compared to Riak KV underscores the practical advantage of considering MongoDB as a preferred choice, particularly in scenarios where read operations play a crucial role. Understanding the performance characteristics of different databases can inform strategic decision-making in database selection and architecture design, potentially leading to improved system efficiency and user experience.

Developers aiming for high-performance analytics should opt for MongoDB on Google Cloud, which offers superior read throughput and stable performance across diverse workloads. However, for IoT-driven data streams where low scan latency is critical, Riak KV proves more suitable due to its efficient handling of scatter-gather operations. Developers should avoid deploying latency-sensitive applications on OpenStack unless the environment is optimized explicitly for NoSQL workloads, as it tends to exhibit higher latency and inconsistent performance.

The divergent findings regarding MongoDB's performance in scan operations highlight the importance of conducting thorough performance evaluations tailored to the specific requirements of the application. While MongoDB may exhibit superior runtime in many scenarios, the observed latency in scan operations highlights the need to evaluate database performance across a range of workload types. This underscores the importance of comprehensive testing and benchmarking when selecting databases to ensure optimal performance across diverse usage scenarios.

For organizations and cloud architects considering cloud platforms for hosting databases, the demonstrated performance advantage of Google Cloud, particularly when coupled with MongoDB, underscores the practical benefits of leveraging Google's Cloud infrastructure. Understanding the technical advantages afforded by Google Cloud's private fiber networking infrastructure can inform strategic decisions regarding cloud platform selection and resource allocation. By leveraging Google Cloud's robust networking capabilities, organizations can potentially achieve enhanced database performance and reliability, thereby improving overall system efficiency and user experience. When designing cloud infrastructure, architects should consider the cost-performance trade-off. While Google Cloud delivers higher throughput, DigitalOcean’s lower hourly rate ($0.06/hour vs. Google’s $0.08/hour for similar VM configurations) makes it a viable option for small-scale or budget-sensitive deployments, especially where peak performance is not the primary concern. Strategic platform selection can help balance operational costs with performance goals.

6. Conclusion

Selecting the appropriate NoSQL database and CC environment remains a daunting task for both designers and business stakeholders. The primary objectives of this study were twofold: to assess the performance of big data in CC environments and to delineate the challenges and unresolved issues associated with big data in CC. This study includes a comparative analysis of performance across MongoDB and Riak KV databases. We scrutinized the throughput and latency analysis for these two databases within diverse CC environments, including Google Cloud, DigitalOcean, and OpenStack. We employed YCSB as the benchmark, owing to its widespread implementation in NoSQL database testing.

The experiments consistently demonstrated MongoDB's superior performance within the Google Cloud environment. However, limitations in MongoDB's performance were observed within both DigitalOcean and OpenStack clouds. Moreover, OpenStack Riak KV showed high latency compared to others. A decision tree was subsequently devised to establish performance criteria for selecting databases and CC platforms. Future research endeavors could expand upon this study by comparing the leading NoSQL databases across different CCPs to identify the optimal database. Such analysis could encompass various benchmarks, including throughput, latency, scalability, and cost, while evaluating performance under diverse workloads and data types. This comparative assessment holds the potential to guide the selection of the most suitable database and CCP for specific use cases.

6.1. Research Limitations and Future Research Directions

While the study provides insightful results, it is essential to acknowledge certain limitations that affect the ability to definitively conclude which cloud platform is superior. The integration of MongoDB with Google Cloud demonstrated significant performance gains, particularly in managing extensive data reading and writing operations. Additionally, Google Cloud's infrastructure demonstrated superior manageability and efficiency in handling distributed environments and large datasets compared to other cloud platforms. However, these observations cannot be fully generalized due to the lack of detailed information regarding the specific instance types used in the comparison. Instance types, which define the virtual hardware configurations in cloud environments, play a crucial role in performance outcomes. Different instance types can have varying amounts of CPU, memory, storage, and network capabilities, which significantly influence the performance of database operations and big data processing. Without knowing the exact instance types used for Google Cloud, as well as for the other platforms, it is challenging to attribute the performance gains solely to the cloud platform itself.

This study focuses solely on MongoDB and Riak KV as the NoSQL databases under evaluation. While these are popular choices, the NoSQL landscape is rapidly evolving, with numerous other databases offering different architectures and performance characteristics, such as Cassandra, Couchbase, and HBase. Broadening the study to encompass a wider variety of NoSQL databases would provide a more comprehensive view of the relative advantages, disadvantages, and applicability of different databases for various big data workloads in cloud environments. This extension would enhance comprehension and facilitate the identification of the most suitable databases for specific use cases. Similarly, the study deploys only the YCSB benchmark. Incorporating additional benchmarks or workload generators that can simulate more realistic and diverse workload scenarios, including mixed workloads, ad-hoc queries, and data models with varying complexities, would provide a more comprehensive evaluation. Additionally, the study focuses on three cloud platforms: DigitalOcean, OpenStack, and Google Cloud. While these are popular choices, the cloud computing landscape is rapidly evolving, with new platforms and services emerging constantly (e.g., Amazon Web Services, Microsoft Azure, IBM Cloud, etc.). Expanding the research to include these additional platforms would provide a more comprehensive understanding of the cloud ecosystem.

The decision tree which was proposed is specifically modeled based on experiments conducted with text-based data which were the focus of the performance evaluation. While it provides useful guidance for similar workloads, it may not be directly applicable to other data types, such as multimedia files, which have distinct storage and access requirements. Furthermore, in this work, communication is used solely for setting up the nodes and, subsequently, running the benchmark on the cloud platform. This means we cannot control or manage the communication quality on the cloud server, as it is controlled by the server’s infrastructure. Future research could explore techniques for mitigating the impact of fluctuating communication on benchmarking results. This may include adaptive workload distribution strategies or performance profiling approaches that consider network latency and bandwidth constraints. The recognition of inherent limitations in controlling communication quality underscores the importance of designing experiments that are robust to fluctuations in network performance. This may involve implementing redundancy measures, optimizing data transfer protocols, or selecting cloud regions with reliable network infrastructure. Additionally, transparent reporting of communication constraints and their potential impact on benchmarking results can facilitate informed decision-making and the interpretation of performance metrics in cloud-based evaluation.

7. Declarations

7.1. Author Contributions

Conceptualization, A.T. and Y.A.; methodology, A.R.; software, A.R.; validation, E.E. and A.C.; formal analysis, E.E. and A.T.; investigation, A.R.; resources, D.C.; data curation, M.A.; writing—original draft preparation, S.A., M.A., E.E., A.R., and A.T.; writing—review and editing, Y.A., D.C., N.M., and A.C.; visualization, E.E.; supervision, A.T. and A.R. All authors have read and agreed to the published version of the manuscript.

7.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

7.3. Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

7.4. Institutional Review Board Statement

Not applicable.

7.5. Informed Consent Statement

Not applicable.

7.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

8. References

[1] Deepa, N., Pham, Q. V., Nguyen, D. C., Bhattacharya, S., Prabadevi, B., Gadekallu, T. R., ... & Pathirana, P. N. (2022). A survey on blockchain for big data: Approaches, opportunities, and future directions. Future Generation Computer Systems, 131, 209-226. doi:10.1016/j.future.2022.01.017.

[2] Souza, F., Tavares, E., & Araújo, C. (2025). A modelling approach for estimating energy consumption of NoSQL-based storage systems. Journal of Supercomputing, 81(6), 797. doi:10.1007/s11227-025-07298-4.

[3] Beckermann, B. M. (2025). Transactional YCSB: Benchmarking ACID-Compliant NoSQL Systems with Multi-Operation Transactions. Datenbanksysteme für Business, Technologie und Web (BTW 2025), 1019-1030. doi:10.18420/BTW2025-67.

[4] Adnan, K., & Akbar, R. (2019). An analytical study of information extraction from unstructured and multidimensional big data. Journal of Big Data, 6(1), 1-38. doi:10.1186/s40537-019-0254-8.

[5] Krishan, K., Gupta, G., & Bhathal, G. S. (2024). Striking the Balance: Comprehensive Insights into Data Consistency in NoSQL Realms. Proceedings of the 18th INDIAcom; 2024 11th International Conference on Computing for Sustainable Global Development, INDIACom 2024, 715–720. doi:10.23919/INDIACom61295.2024.10498626.

[6] Carvalho, I., Sá, F., & Bernardino, J. (2023). Performance Evaluation of NoSQL Document Databases: Couchbase, CouchDB, and MongoDB. Algorithms, 16(2), 78. doi:10.3390/a16020078.

[7] Gomes, C., Meuse, M. N., Nogueira, B., Maciel, P., & Tavares, E. (2023). NoSQL-based storage systems: influence of consistency on performance, availability and energy consumption. Journal of Supercomputing, 79(18), 21424–21448. doi:10.1007/s11227-023-05488-6.

[8] Ferreira, S., Mendonça, J., & Andrade, E. (2025). Experimental Performance Analysis of Data Consistency Levels in NoSQL Databases. Software - Practice and Experience, 55(6), 1059–1070. doi:10.1002/spe.3412.

[9] Pramanik, S., & Bandyopadhyay, S. K. (2023). Analysis of big data. Encyclopedia of data science and machine learning, IGI Global, 97-115. doi:10.4018/978-1-7998-9220-5.ch006.

[10] Weitzenboeck, E. M., Lison, P., Cyndecka, M., & Langford, M. (2022). The GDPR and unstructured data: is anonymization possible? International Data Privacy Law, 12(3), 184–206. doi:10.1093/idpl/ipac008.

[11] Sandhu, A. K. (2022). Big Data with Cloud Computing: Discussions and Challenges. Big Data Mining and Analytics, 5(1), 32–40. doi:10.26599/BDMA.2021.9020016.

[12] Rmis, A. M., & Topcu, A. E. (2020). Evaluating RIAK key value cluster for big data. Tehnicki Vjesnik, 27(1), 157–165. doi:10.17559/TV-20180916120558.

[13] Andreoli, R., Cucinotta, T., & De Oliveira, D. B. (2023). Priority-Driven Differentiated Performance for NoSQL Database-As-A-Service. IEEE Transactions on Cloud Computing, 11(4), 3469–3482. doi:10.1109/TCC.2023.3292031.

[14] Araújo, C., Oliveira, M., Nogueira, B., Maciel, P., & Tavares, E. (2024). Performability evaluation of NoSQL-based storage systems. Journal of Systems and Software, 208, 111885. doi:10.1016/j.jss.2023.111885.

[15] Bansal, N., Sachdeva, S., & Awasthi, L. K. (2024). Are NoSQL Databases Affected by Schema? IETE Journal of Research, 70(5), 4770–4791. doi:10.1080/03772063.2023.2237478.

[16] Aceto, G., Persico, V., & Pescapé, A. (2020). Industry 4.0 and Health: Internet of Things, Big Data, and Cloud Computing for Healthcare 4.0. Journal of Industrial Information Integration, 18, 100129. doi:10.1016/j.jii.2020.100129.

[17] Awaysheh, F. M., Aladwan, M. N., Alazab, M., Alawadi, S., Cabaleiro, J. C., & Pena, T. F. (2022). Security by Design for Big Data Frameworks Over Cloud Computing. IEEE Transactions on Engineering Management, 69(6), 3676–3693. doi:10.1109/TEM.2020.3045661.

[18] Gillis, A.S. (2022). DigitalOcean. Available online: https://www.techtarget.com/searchcloudcomputing/definition/DigitalOcean (accessed on August 2025).

[19] Al-Dhaqm, A., Ikuesan, R. A., Kebande, V. R., Razak, S. A., Grispos, G., Choo, K. K. R., Al-Rimy, B. A. S., & Alsewari, A. A. (2021). Digital Forensics Subdomains: The State of the Art and Future Directions. IEEE Access, 9, 152476–152502. doi:10.1109/ACCESS.2021.3124262.

[20] Singh, B., Martyr, R., Medland, T., Astin, J., Hunter, G., & Nebel, J. C. (2022). Cloud based evaluation of databases for stock market data. Journal of Cloud Computing, 11(1), 53. doi:10.1186/s13677-022-00323-4.

[21] Barkat, A., Dos Santos, A. D., & Ho, T. T. N. (2015). Open stack and cloud stack: Open source solutions for building public and private clouds. Proceedings - 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2014, 429–436. doi:10.1109/SYNASC.2014.64.

[22] Osman, A. M. S. (2019). A novel big data analytics framework for smart cities. Future Generation Computer Systems, 91, 620–633. doi:10.1016/j.future.2018.06.046.

[23] Martinez-Mosquera, D., Navarrete, R., Luján-Mora, S., Recalde, L., & Andrade-Cabrera, A. (2024). Integrating OLAP with NoSQL Databases in Big Data Environments: Systematic Mapping. Big Data and Cognitive Computing, 8(6), 64. doi:10.3390/bdcc8060064.

[24] Kanchan, S., Kaur, P., & Apoorva, P. (2020). Empirical Evaluation of NoSQL and Relational Database Systems. Recent Advances in Computer Science and Communications, 14(8), 2637–2650. doi:10.2174/2666255813999200612113208.

[25] Alzoubi, Y. I., Topcu, A. E., & Erkaya, A. E. (2023). Machine Learning-Based Text Classification Comparison: Turkish Language Context. Applied Sciences (Switzerland), 13(16), 9428. doi:10.3390/app13169428.

[26] Topcu, A. E., Alzoubi, Y. I., & Karacabey, H. A. (2023). Text Analysis of Smart Cities: A Big Data-based Model. International Journal of Intelligent Systems and Applications in Engineering, 11(4), 724–733.

[27] Obschonka, M., & Audretsch, D. B. (2020). Artificial intelligence and big data in entrepreneurship: a new era has begun. Small Business Economics, 55(3), 529–539. doi:10.1007/s11187-019-00202-4.

[28] Luan, H., Geczy, P., Lai, H., Gobert, J., Yang, S. J. H., Ogata, H., Baltes, J., Guerra, R., Li, P., & Tsai, C. C. (2020). Challenges and Future Directions of Big Data and Artificial Intelligence in Education. Frontiers in Psychology, 11, 580820. doi:10.3389/fpsyg.2020.580820.

[29] Kenitar, S. B., Arioua, M., & Yahyaoui, M. (2023). A Novel Approach of Latency and Energy Efficiency Analysis of IIoT with SQL and NoSQL Databases Communication. IEEE Access, 11, 129247–129257. doi:10.1109/ACCESS.2023.3332483.

[30] Hofmann, E. (2017). Big data and supply chain decisions: the impact of volume, variety and velocity properties on the bullwhip effect. International Journal of Production Research, 55(17), 5108–5126. doi:10.1080/00207543.2015.1061222.

[31] Khan, W., Kumar, T., Zhang, C., Raj, K., Roy, A. M., & Luo, B. (2023). SQL and NoSQL Database Software Architecture Performance Analysis and Assessments—A Systematic Literature Review. Big Data and Cognitive Computing, 7(2), 97. doi:10.3390/bdcc7020097.

[32] Mishra, A., Jabar, T. S., Alzoubi, Y. I., & Mishra, K. N. (2023). Enhancing privacy-preserving mechanisms in Cloud storage: A novel conceptual framework. Concurrency and Computation: Practice and Experience, 35(26), 7831. doi:10.1002/cpe.7831.

[33] Park, J., & Lee, D. H. (2022). Parallelly Running and Privacy-Preserving k-Nearest Neighbor Classification in Outsourced Cloud Computing Environments. Electronics (Switzerland), 11(24), 4132. doi:10.3390/electronics11244132.

[34] Zeghib, N. E. I., Alwan, A. A., Abualkishik, A. Z., & Gulzar, Y. (2022). Multi-Route Plan for Reliable Services in Fog-Based Healthcare Monitoring Systems. International Journal of Grid and High Performance Computing, 14(1), 1–20. doi:10.4018/IJGHPC.304908.

[35] Riak. (2025). Riak – a distributed, decentralised data storage system. Available online: https://github.com/basho/riak (accessed on August 2025).

[36] Topcu, A. E., & Rmis, A. M. (2020). Analysis and evaluation of the Riak cluster environment in distributed databases. Computer Standards and Interfaces, 72, 103452. doi:10.1016/j.csi.2020.103452.

[37] Eyada, M. M., Saber, W., El Genidy, M. M., & Amer, F. (2020). Performance Evaluation of IoT Data Management Using MongoDB Versus MySQL Databases in Different Cloud Environments. IEEE Access, 8, 110656–110668. doi:10.1109/ACCESS.2020.3002164.

[38] MongoDB. (2025). What is MongoDB? Available online: https://www.mongodb.com/docs/manual/ (accessed on August 2025).

[39] da Silva, L. F., & Lima, J. V. F. (2023). An evaluation of relational and NoSQL distributed databases on a low-power cluster. Journal of Supercomputing, 79(12), 13402–13420. doi:10.1007/s11227-023-05166-7.

[40] Khan, S., Liu, X., Ali, S. A., & Alam, M. (2023). Bivariate, cluster, and suitability analysis of NoSQL solutions for big graph applications. Advances in Computers, 128, 39–105. doi:10.1016/bs.adcom.2021.09.006.

[41] Kim, S., Hoang, Y., Yu, T. T., & Kanwar, Y. S. (2023). GeoYCSB: A Benchmark Framework for the Performance and Scalability Evaluation of Geospatial NoSQL Databases. Big Data Research, 31, 100368. doi:10.1016/j.bdr.2023.100368.

[42] Nurhadi, Kadir, R. B. A., & Surin, E. S. B. M. (2021). Evaluation of NoSQL Databases Features and Capabilities for Smart City Data Lake Management. Lecture Notes in Electrical Engineering: Vol. 739 LNEE, 383–392. doi:10.1007/978-981-33-6385-4_35.

[43] Seghier, N. Ben, & Kazar, O. (2021). Performance Benchmarking and Comparison of NoSQL Databases: Redis vs MongoDB vs Cassandra Using YCSB Tool. Proceedings - 2021 IEEE International Conference on Recent Advances in Mathematics and Informatics, ICRAMI 2021, 9585956. doi:10.1109/ICRAMI52622.2021.9585956.

[44] Celesti, A., Lay-Ekuakille, A., Wan, J., Fazio, M., Celesti, F., Romano, A., Bramanti, P., & Villari, M. (2020). Information management in IoT cloud-based tele-rehabilitation as a service for smart cities: Comparison of NoSQL approaches. Measurement: Journal of the International Measurement Confederation, 151, 107218. doi:10.1016/j.measurement.2019.107218.

[45] Kausar, M. A., & Nasar, M. (2019). SQL Versus NoSQL Databases to Assess Their Appropriateness for Big Data Application. Recent Advances in Computer Science and Communications, 14(4), 1098–1108. doi:10.2174/2213275912666191028111632.

[46] Copper, B. F. (2020). Core YCSB Properties. GitHub. Available online: https://github.com/brianfrankcooper/YCSB/wiki/Core-Properties (accessed on August 2025).

[47] Cribbs, S. (2025). Schema design in Riak – introduction. Available online: https://riak.com/posts/technical/schema-design-in-riak-introduction/index.html (accessed on August 2025).

[48] Capris, T., Melo, P., Garcia, N. M., Pires, I. M., & Zdravevski, E. (2022). Comparison of SQL and NoSQL databases with different workloads: MongoDB vs MySQL evaluation. 2022 International Conference on Data Analytics for Business and Industry, ICDABI 2022, 214–218. doi:10.1109/ICDABI56818.2022.10041513.

[49] Antas, J., Silva, R. R., & Bernardino, J. (2022). Assessment of SQL and NoSQL Systems to Store and Mine COVID-19 Data. Computers, 11(2), 29. doi:10.3390/computers11020029.

[50] Negi, S., Rauthan, M. M. S., Vaisla, K. S., & Panwar, N. (2021). CMODLB: an efficient load balancing approach in cloud computing environment. Journal of Supercomputing, 77(8), 8787–8839. doi:10.1007/s11227-020-03601-7.

[51] Fernandez, R. (2023). Google cloud platform: What is it, and should you use it? Available online: https://www.techrepublic.com/article/google-cloud-platform-the-smart-persons-guide/ (accessed on August 2025).