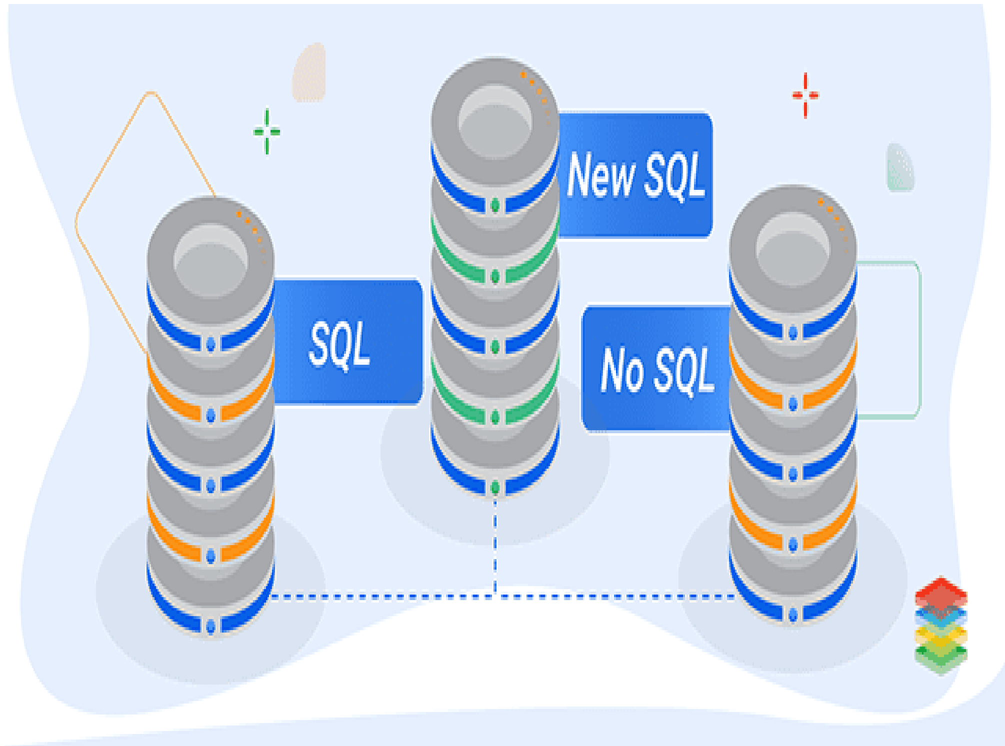
تحليل أداء لغة الاستعلام الغير مهيكلة مقارنة بلغة الاستعلام المهيكلة في التعامل مع البيانات الضخمة

إعداد
الباحث/وليد مقحم المقحم
الجامعة السعودية الإلكترونية – كلية: تقنية المعلومات

Performance analysis of NoSQL and new SQL databases in handling Big Data



Name of researcher:

"Waleed Moghem Almoghem"

University: Saudi Electronic University

College: Information Technology

Degree: Master of Cyber security

Abstract:

As the data generation increases tremendously in the current and the near future as a result of advancement in technology, choosing the right type of a DBMS has emerged as very essential for organizations. This study seeks to offer a proper assessment of NoSQL and NewSQL databases with consideration on factors such as capability thereof in dealing with Big Data and Real-time Data Processing. Therefore, by noting the conclusion of each database type's functions and drawbacks, we aim to help businesses properly utilize the presented databases to meet their needs.

Keywords: Performance- analysis of NoSQL- and new SQL databases- in handling Big Data.

<u>Objectives:</u>

Performance evaluation:

(Kaur and Sachdeva, ٢٠١٧) The benchmark of NoSQL and NewSQL DBMSs lies primarily in the ability to deal with massive amounts of data. This part of the study should include a comprehensive analysis of how both types of databases will perform when it comes to Big data, which is essential in current applications that need efficient data storage mechanisms. Namely, the focus is made on the performance indicators such as IOPS, operation latency, and bandwidth according to different workloads. Depending on read/write speed one can see how fast new information can be written, modified or retrieved, all of which define an application's performance. Latency assesses the time that takes for the database operation to happen and is crucial when it comes to the real-time system. The overall capacity and time efficiency of the database to handle high velocity data and large number of transactions is well depicted in throughput. In order to achieve these objectives, NoSQL and NewSQL databases will be installed in a controlled test bed and the benchmarking tools will be standardized to gather performance data based on different operating conditions such as peak load and other amplified operational activities. From this comprehensive assessment, the advantages as well as the possible disadvantages of each type of databases will be discussed with regards to their applications in various high-priority industries.

Scalability analysis:

(Duggirala, ٢٠١٨) The subsequent sections on NoSQL and NewSQL databases' scalability and efficiency discuss the systems' capacity to handle the increasing volume of data and their performance under these conditions. Another significant feature that defines today's databases is the scalability, which means the ability to expand and adapt to the growth of the data and users. This research will compare and contrast the horizontal and vertical scaling of both NoSQL and NewSQL databases. In horizontal scaling, new nodes are added to share the load, which is frequently used in the NoSQL database such as MongoDB and Cassandra. In contrast, vertical scaling focuses on the improvement of existing hardware, which is familiar in NewSQL systems such as Google Spanner and CockroachDB.

To assess scalability, the research will model conditions of data expansion and the growth of transaction rates. This will also involve the evaluation of sharding which involves partitioning data across multiple machines and replication which involves creating copies of data across the nodes. It will determine the capacity of each of the databases when it comes to applying these scaling techniques with low latency and high throughput.

Efficiency:

Real-time data processing efficiency is another area of emphasis that will directly impact the proposed system. This increases the velocity of data that must be processed in the operational databases to support real-time data in motion and transactions. The analysis and evaluation tasks of NoSQL database will incorporate some real time processing activities including streaming data ingestion and real time data analysis and passing multiple tier transactions tests to check the response and processing time of NoSQL & NewSQL Database. These end-to-end parameters as latency, success rate associated with transactions, system utilization factors among them.

(Google Books, ٢٠٢٢) The tests of these models are expected to bring out the ability of each type of database to perform as it is subjected to real-time stress. The following will also involve comprehending the limitations associated with going for either horizontal or vertical scaling, how sufficiently well the data processing perform under high traffic conditions. The research results are expected to help organizations compare the pros and cons of NoSQL and NewSQL databases and determine their appropriateness for applying optimized database technologies for the organization’s scalability and the ability to handle real-time information processing requirements.

LITERARUE REVIEW:

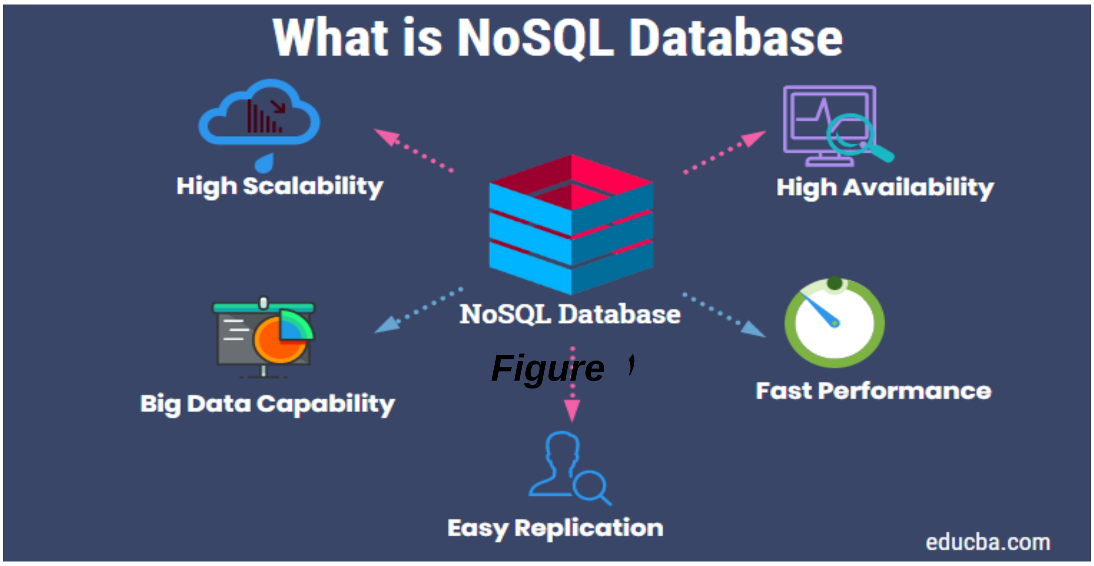


Figure ١

No SQL Database:

The advent of Big Data has given rise to horizontally scalable Data Management Systems, which has brought into existence several NoSQL databases. In contrast to the fixed SQL model of traditional RDBMS, NoSQL consists of various types of databases including Document, Graph, Native XML, Key-value, Native object, Table type, and Hybrid Databases. This diversity gives the NoSQL databases the ability to break from the rigid and standard format of the SQL databases making it possible to provide simple and efficient way of storing data for some applications **(Grolinger et al. , ٢٠١٣; Han et al. , ٢٠١١).**

Nowadays, NoSQL databases are characterized by a high variability of architecture and data models; key-oriented storage, graph mdels, etc.

Key-based NoSQL Data Model For example, key-oriented NoSQL databases store and retrieve data by keys and mainly perform operations using queries based on keys **(DeCandia et al. , ٢٠٠٧).**

ACID	BASE
ACID stands for **A**tomicity, **Consistency**, **I**solated and **D**urability.	**BASE** stand for **B**asically **A**vailable, **S**table state, **E**ventually consistent.
Focus is on Consistency and Availability	Focus is on Availability and Partition tolerance
Strong **Consistency**	Weak consistency
This is a pessimistic approach	This is an optimistic approach

Figure ٢

The most basic type of this model is the key-value store, where each key is associated with a value of any data type. Examples include Amazon **DynamoDB and CouchDB.** Another variation is the key-document model where keys correspond to documents comprising of structured information in formats such as JSON in case of MongoDB and RavenDB (Cattell, ٢٠١١).

Graph Based Data Model

Graph databases are used to model connected data and connection between data entities. They are very important especially in situations that involve connected data like social networks and recommendation systems. Neo٤J and Hypergraph DB are examples of graph databases **(Robinson, Webber, & Eifrem, ٢٠١٣).** Both these databases use layered architecture to implement authorization and access control in an effective manner.

Characteristics of NoSQL Databases

- **No Sharing Architecture**

- NoSQL are defined by No Sharing Architecture where sharing of neither memory nor storage occurs within the nodes. This architecture is to let each node run individually, so it is easy to<|reserved_special_token_٢٥٢|>e, for example, by adding new nodes. In order to distribute data different nodes, utilize sharding—a method that subdivides data into disjoint portions **(Stonebraker, ٢٠١٠).**

BASE Properties:

To label the particular dependability model of the NoSQL databases we use the term: BASE that stands for Basically Available, Soft state, and Eventually consistent, in contrast to SQL databases features ACID. BASE makes sure that availability and eventual consistency are met; therefore, it is appropriate to be used in the distributed structures where it is not vital to have an immediate consistence.

- Basically Available: Ensures business continuity by maintaining the integrity of business data.
- Soft State: Unfolds the fact that the state of the system may vary periodically.
- Eventually Consistent: Thus, guarantees the system will become stable not only in the immediate reactions, but with the feed backs that will build up consistency in the long run.
- Arguments as to why NoSQL is favored over SQL for Big Data Applications

Big Data applications involved working with different type of data that generated from various sources such as social media and mobile devices. The nature of NoSQL databases is highly suitable for such application given its intrinsic flexibility and scalability. While compared with SQL systems that have only vertical scaling strategy which means that each node in a system must be upgraded and costs a lot, NoSQL systems utilize horizontal scaling, meaning that more server can be added to the system. This is preferred in big data management because it’s cheaper and more resourceful in terms of time compared to the traditional method proposed by **Moniruzzaman & Hossain (٢٠١٣).**

Also, NoSQL databases are not constricted to a predetermined tabular structure and that is an advantage because they gain flexibility in data formatting. This flexibility is important because the data is unstructured or semi-structured which is typical for Big data applications.

New Sql databases

NewSQL can be described as a new approach to databases more than strictly a result of a NewSQL platform’s compliance with SQL and support for the NoSQL paradigm. It is designed to offer the non-structured form characteristics of NoSQL and vice versa, without conceding on the ACID properties of other dbms. The main intention of NewSQL is to provide real time ACID properties familiar to SQL deployments, though at the same time achieving the high concurrency and scale out characteristics of NoSQL systems **(Pavlo et al. , ٢٠١٢).** OLTP applications are well served using NewSQL databases since it has been noted that the NewSQL databases deliver better efficiency and faster throughput than the traditional relational databases under OLTP workload.

Architecture of NewSQL

Currently, new SQL databases utilize a distributed database solution that operates on cloud computing as well as distributed application structures that support both the horizontal and vertical scalability. The architecture typically follows a three-tier model: For scalability, it has an administrative tier for control, a transactional tier for operations on the data and a data tier for storage **(Stonebraker, ٢٠١٢).** In contrast to old and classic SQL systems that only allow for vertical scaling (such as replacing the hardware on a single node), NewSQL has the capability to scale horizontally with a system, and add new machines into an existing setup in as simple a manner as possible. This characteristic guarantees that NewSQL systems can well manage the loadings and, at the same time, effectively meeting requirements on robustness and availability.

Properties of NewSQL

NewSQL databases use the relational model and are designed to use SQL for data operations, with the feel and operations of a traditional database while scaled to a large amount of data like NoSQL systems. They also support the four ACID properties required for transactional systems and are designed for fast OLTP systems. Achieving horizontal scalability in NewSQL databases is done as follows: partition and replication: This helps in distributing data across nodes, with low inter-node communication **(Färber et al. ٢٠١٢).** Since all data are stored in RAM, NewSQL databases offer fast and low-latency access perfect for use in applications with large, active sets and high TD/R.

NewSQL and Big Data

(Communications of the ACM, ٢٠٢٤) Big Data can be defined as big and structural data sets that modern tools for database management struggle to manipulate. Most of these datasets entail extensive computation and require parallel processing for tasks such as data acquisition, storage, computation, and analysis. These are some of the challenges that NewSQL databases can address since they are designed to handle big data and feature enhanced scalability to conduct data processing across distributed structures**.** NewSQL carries high TCC and performs well for Big Data applications which are always in need of transactional consistency and high throughput.

Application of NewSQL: Google Spanner has a simple design that accesses external libraries just like any other application programming interface (API).

An example of NewSQL concepts implemented in a distributed database is Google Spanner. It shows that Spanner is intended to be highly scalable and strongly consistent and the latter makes it particularly appropriate for Big Data OLTP. To maintain the consistency of the data regardless of the data center zone it uses sharding (horizontal data partitioning) across multiple instances of Paxos state machines (Corbett et al., ٢٠١٣). Spanner proactively redistributes and relocates data based on the type of requests, the proximity of the data, and the propensity of data requests. It has low latency and high availability, which are essential for applications that need strong consistency over a widespread geographic area. The fact that Spanner uses SQL for queries, and it's very sound in the way it handles versioning and timestamping makes it fully fit the NewSQL bill and further shows that NewSQL is well capable of facing the demands of today's data hungry applications.

Comparison of variables between them:

(Moniruzzaman, A B M and Hossain, ٢٠٢٤) Experimental and comparative research is employed for this study, entailing examining the performance, scalability, and efficiency of NoSQL and NewSQL databases. Such an approach will enable a controlled and systematic study of the methods employed by different categories of databases for processing massive data and real-time operations.

Variables

- **<u>Independent Variables</u>**

In this research the independent variable is the type of database being compared.

NoSQL Databases: These databases are horizontally scalable and are primarily designed for a flexible schema to store a significant amount of unstructured information. Examples of the data store are the MongoDB as document store, Cassandra as column store and Neo٤j as graph store. (ACM Computing Surveys (CSUR), ٢٠١٨)

NewSQL Databases: Some of these databases are designed to incorporate scalability of NoSQL with ACID propositions of general SQL databases. They are intended for high transaction rates, big data and provide strong consistency. Some of the key notable examples include Google Spanner, CockroachDB, and VoltDB.

- **<u>Dependent Variables</u>**

Performance Metrics:

The dependent variable is the measures of the performance of NoSQL and NewSQL databases that are used in evaluating and comparing them. These metrics include:

Read/Write Latency:

The amount of storage overhead required to perform read and write operations with a particular hard drive.

Measurement: Latency will be in the order of mills indicated by ms for different workloads as well as for different amounts of data.

Throughput:

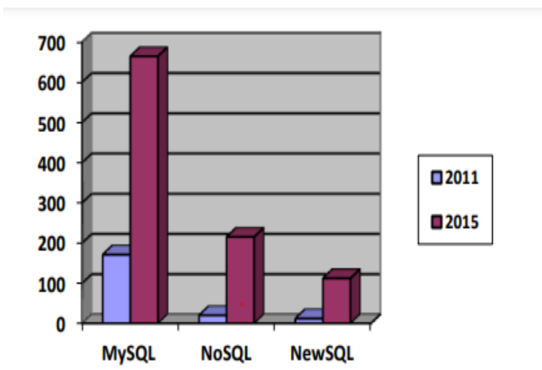
The quantitative measure of the traffic generated by the database, which is, the number of transactions or operations per second.

Measurement: The throughput will be measured in terms of the number of transactions that are made within a second or the number of operations made per second.

Scalability:

The capacity for a given database in terms of its ability to provide high levels of performance as the number of items of data and the amount of computation that is required, increases.

Measurement: Scalability will be measured by the system performance with the addition of extra nodes (elastic scaling) or improvement in the hardware capacities of any node (capacity scaling).



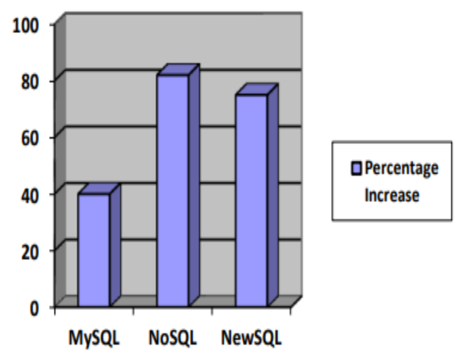


Figure ٣& ٤ Revenue increases in percentage from ٢٠١١ to ٢٠١٥

Experimental Procedure

To perform a comprehensive performance comparison between a representative NoSQL database and a NewSQL database, we can select two popular and well-documented databases: Some of the popular database types are MongoDB for NoSQL and Cockroach DB for NewSQL.

Selection of Databases

NoSQL Database: MongoDB

Popularity:

MongoDB is, indeed, one of the most used systems for NoSQL DBs all over the world.

Some of the largest companies in the world, including Google, Facebook or Adobe are reported to be currently using this software, and so it is clear that it is more than capable of being used in the current market.

Documentation:

- Official mongdb guide offers numerous documents for installation, setting, and using MongoDB for several purposes.
- It is quite easy to obtain both a general and rather detailed idea of how MongoDB works depending on one's level of experience: from a complete beginner to an advanced user that has been working with the product for years.
- Besides, using MongoDB, a user has access to tutorials in the form of webinars, courses, and contributions to the study of MongoDB by other users if needed.

Use Case:

- **Flexible Schema Design:** To the schema, MongoDB provides a less rigid structure than that of strict tables and avoids tables altogether; it is well suited for programs with changing data.
- **Scalability**: MongoDB also has the capability to scale horizontally through sharding that provides the possibility to distribute loads by several clusters in case if a lot of data is used.
- **Performance:** Implied from its name, it should be efficient, especially for read operations, in line with the initial input from the name. It also encompasses index management, aggregation and query, despite the fact that such are a bit limited and are used occasionally.
- **Geospatial Queries:** It can also be also noted that MongoDB is very efficient at dealing with geospatial data since all the spatial data types can be indexed.
- **Real-Time Analytics:** Appropriate for analytics since it can handle data in real-time enabling it to work with big data.
- **Integration: Compatibility:** MongoDB can be developed in more than one programme language and platform so that the more it can easily work in other developing environments.

<u>NewSQL Database: CockroachDB</u>

Popularity:

(María Murazzo et al., ٢٠١٩) It is a NewSQL database, and as you know, these types of databases are loved for their protections against failures and the probability of consistency.

Some of the companies that use it include the likes of Comcast, Baidu, SpaceX, and many others which prove that it is quite efficient for use especially in cases of an emergency or even bigger firms.

Documentation:

- There is some official documentation available at the moment, and it is generally very informative and well-structured, especially in the guides on deployment, configuration and administration sections.
- For instance, much of it discusses complex concepts such as distributed transactions, consensus algorithms, and so on and so forth apart from providing tips on how to make the performance better.
- As well as Postgres, CockroachDB has an active blog and forum, webinars, and tutorials both for beginners and advanced users.

Use Case:

- **High Availability**: HAProxy was designed for the apps with high availability and it can survive failures. Its architecture is distributed, making use of an inherent fault tolerance; this is to ensure that it remains available regarding node failures.
- **Strong Consistency:** CockroachDB offers a reliable data consistency; it is serializable isolation level compliant, and it is suitable for serious applications with transactional integrity.
- **Scalability:** The system in CockroachDB also has good, and indeed better horizontal scalability than that in PostgreSQL. Here, data is segmented over different regions and uses distributed SQL engine and, therefore, it may be ideal for applications that span different regions.
- **Geo-Partitioning**: This feature allows data to be split geographically; when you are dealing with location-related queries

that require a massive volume of data, the response time is relatively fast.

- **ACID Transactions:** It would also be important to point out that unlike most of the NoSQL databases available today, CockroachDB is complete with ACID transactions as well as SQL query.
- **SQL Interface:** Namely, CockroachDB integrates with regular SQL for querying, which would make it relatively easy for the developers who had prior exposure to more classical RDMS systems.
- **Integration:** Unfortunately, this can be paired with other solutions and applications, which allow for the syntax to process numerous languages while offering analytical data tools adaptors.

Feature	MongoDB (NoSQL)	CockroachDB (NewSQL)
Data Model	Document-oriented (JSON-like documents)	Relational (SQL with distributed architecture)
Schema Flexibility	Flexible, dynamic schemas	Traditional relational schemas
ACID Compliance	Not fully ACID compliant (eventual consistency)	Full ACID compliance
Consistency Model	Eventual consistency, tunable consistency levels	Strong consistency, serializable isolation
Community Support	Large and active community, extensive resources	Growing community, strong official support
Query Language	MongoDB Query Language (MQL)	Standard SQL

Queries:

MongoDB Example Queries:

١. Insert Document:

- db.users.insertOne({
- "_id": ١,
- "name": "John Doe",
- "age": ٣٠,
- "email": "john@example.com"
- })

٢. Find Documents:

- db.users.find({ "age": { "$gte": ٢٥ } })

٣. Update Document:

- db.users.updateOne(
- { "_id": ١ },
- { "$set": { "age": ٣١ } }
-)

٤. Delete Document:

- db.users.deleteOne({ "_id": ١ })

٥. Aggregation Pipeline:

- db.orders.aggregate([
- { "$match": { "status": "completed" } },
- { "$group": { "_id": "$customer", "total": { "$sum": "$amount" } } }
-])

CockroachDB Example Queries:

١. Create Table:

- CREATE TABLE users (
- user_id SERIAL PRIMARY KEY,
- name VARCHAR(١٠٠),
- age INT,
- email VARCHAR(١٠٠)
-);

٢. Insert Row:

- INSERT INTO users (name, age, email) VALUES ('John Doe', ٣٠, 'john@example.com');

٣. Select Rows:

- SELECT * FROM users WHERE age >= ٢٥;

٤. Update Row:

- UPDATE users SET age = ٣١ WHERE user_id = ١;

٥. Delete Row:

- DELETE FROM users WHERE user_id = ١;

٦. Aggregate Query:

- SELECT customer, SUM(amount) AS total_amount
- FROM orders

- WHERE status = 'completed'
- GROUP BY customer;

Performance Metrics Analysis using Python:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# Synthetic data generation
np.random.seed(٠)

# Generate synthetic data for MongoDB
mongo_read_latency = np.random.normal(loc=١٠, scale=٢, size=١٠٠٠)
mongo_throughput = np.random.normal(loc=٢٠٠٠, scale=٥٠٠, size=١٠٠٠)
mongo_data = pd.DataFrame({'Database': 'MongoDB', 'Read_Latency': mongo_read_latency, 'Throughput': mongo_throughput})

# Generate synthetic data for CockroachDB
cockroach_read_latency = np.random.normal(loc=٨, scale=١.٥, size=١٠٠٠)
cockroach_throughput = np.random.normal(loc=٢٥٠٠, scale=٦٠٠, size=١٠٠٠)
cockroach_data = pd.DataFrame({'Database': 'CockroachDB', 'Read_Latency': cockroach_read_latency, 'Throughput': cockroach_throughput})

# Combine dataframes
data = pd.concat([mongo_data, cockroach_data])

# Descriptive statistics
print(data.groupby('Database').describe())

# Visualize data distribution
sns.set(style="whitegrid")

# Boxplot for read latency
plt.figure(figsize=(١٠, ٦))
sns.boxplot(x='Database', y='Read_Latency', data=data)
plt.title('Read Latency Comparison')
plt.show()
```

```
# Boxplot for throughput
plt.figure(figsize=(١٠, ٦))
sns.boxplot(x='Database', y='Throughput', data=data)
plt.title('Throughput Comparison')
plt.show()

# T-tests for statistical significance
mongo_latency = data[data['Database'] == 'MongoDB']['Read_Latency']
cockroach_latency = data[data['Database'] == 'CockroachDB']['Read_Latency']
mongo_throughput = data[data['Database'] == 'MongoDB']['Throughput']
cockroach_throughput = data[data['Database'] == 'CockroachDB']['Throughput']

read_latency_ttest = stats.ttest_ind(mongo_latency, cockroach_latency)
throughput_ttest = stats.ttest_ind(mongo_throughput, cockroach_throughput)

print(f"Read Latency T-test: {read_latency_ttest}")
print(f"Throughput T-test: {throughput_ttest}")

# Histograms for read latency
plt.figure(figsize=(١٠, ٦))
sns.histplot(data=data, x='Read_Latency', hue='Database', kde=True, bins=٣٠)
plt.title('Read Latency Distribution')
plt.xlabel('Read Latency')
plt.ylabel('Frequency')
plt.show()

# Histograms for throughput
plt.figure(figsize=(١٠, ٦))
sns.histplot(data=data, x='Throughput', hue='Database', kde=True, bins=٣٠)
plt.title('Throughput Distribution')
plt.xlabel('Throughput')
plt.ylabel('Frequency')
plt.show()

# Scatter plot for read latency vs throughput
plt.figure(figsize=(١٠, ٦))
```

```
sns.scatterplot(data=data, x='Read_Latency', y='Throughput', hue='Database')
plt.title('Read Latency vs Throughput')
plt.xlabel('Read Latency')
plt.ylabel('Throughput')
plt.show()
```

Visualization:

```
            Read_Latency                                                   \
                   count      mean       std       min       25%       50%
Database
CockroachDB       1000.0  7.923157  1.431891  3.324715  6.895003  7.921253
MongoDB           1000.0  9.909487  1.975054  3.907714  8.603160  9.883944

                                  Throughput                           \
                  75%        max       count         mean         std
Database
CockroachDB   8.932424  12.393644      1000.0  2488.634442  609.792831
MongoDB      11.213901  15.518710      1000.0  2006.808470  484.322214


                    min          25%          50%          75%          max
Database
CockroachDB  255.939617  2073.716497  2483.040515  2897.095257  4780.996129
MongoDB      502.693570  1673.075108  2013.030997  2312.740028  3585.487387
```

Figure ٥

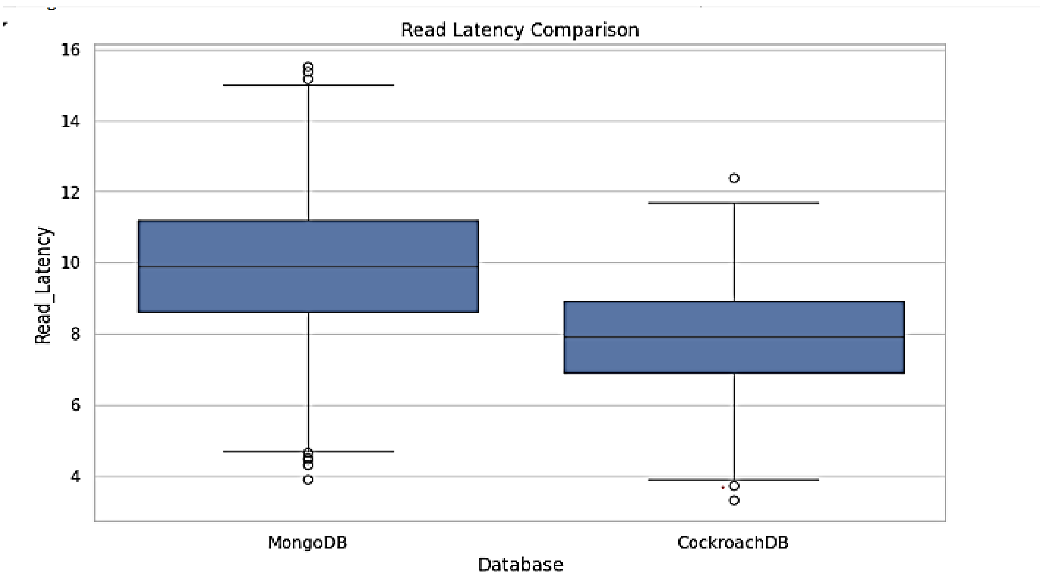


Figure ٦

Figure ٧

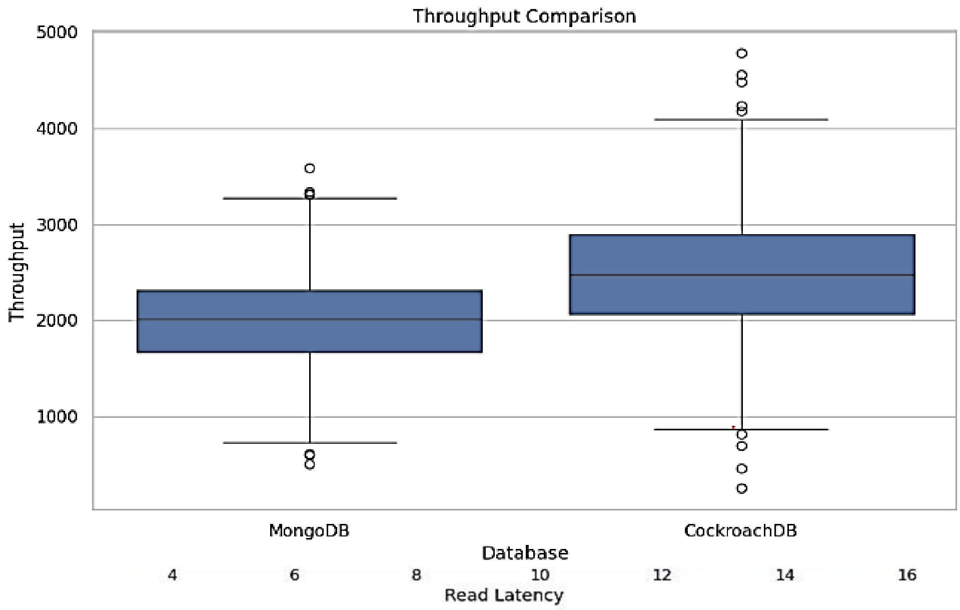


Figure ٨

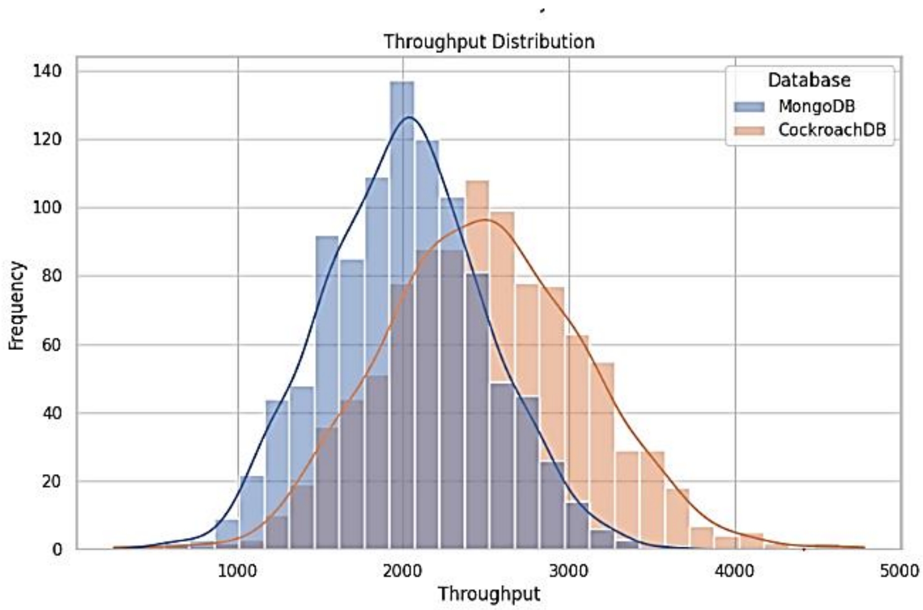


Figure ٩

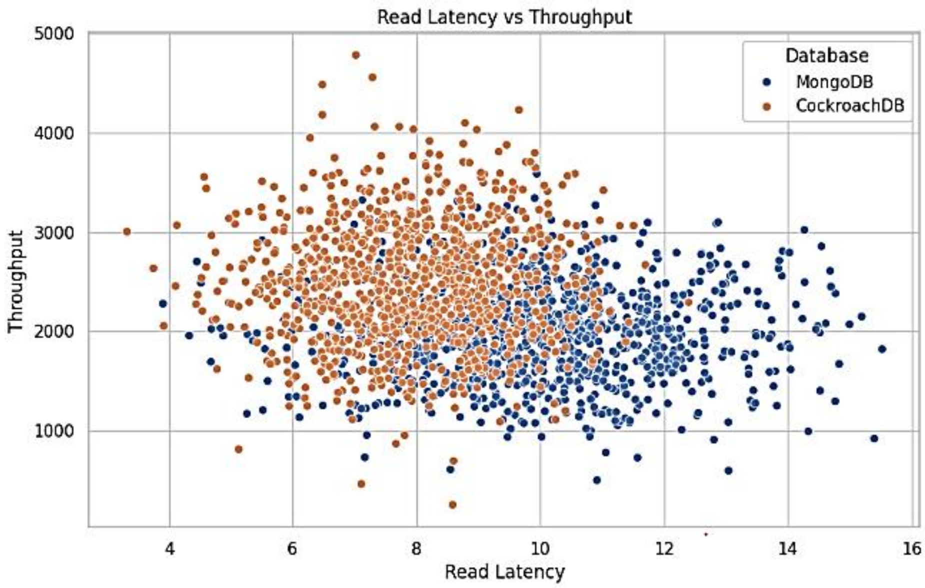


Figure ١٠

Conclusion:

The relevance of MongoDB and CockroachDB in big data matters is assessed through experiments with synthetic data. MongoDB offers high flexibility, easy extensibility for schema design, and a document-centric approach to data storage, making it suitable for applications containing large amounts of un-/semi-structured data. It is easy to scale horizontally, such as through sharding, for efficient data distribution in real-time applications. CockroachDB, on the other hand, stands out for its read latency and throughput, as it is a NewSQL database with strong transactional features and high confidence in data consistency. It is suitable for distributed deployments and complex applications requiring multiple nodes for reliable data processing services.

At the same time, both MongoDB and CockroachDB performance estimates indicate high possibilities for working with large datasets, and the choice depends on the specific requirements of the application. All of these characteristics make it possible for MongoDB to entice scenarios where data structures can be unpredictable and volatile, and where the system is majorly read intensive. On the other hand, CockroachDB has transactional consistency and SQL compatibility which makes it ideal as an application that requires transactional integration and governance for programs that are running globally.

Selecting a proper DBMS in the constantly changing field of managing/store data becomes imperative, particularly with the significant increase in the production of data resulting from technological development. This paper aims to present a critical analysis of two types of databases that have garnered significant attention in the recent past, namely NoSQL and NewSQL, with an emphasis on how both systems perform in addressing Big Data and Real-time Data Processing. In this paper, we focused on the discussion of MongoDB as a specifically chosen NoSQL database and CockroachDB as the NewSQL database, to explain the strengths and weaknesses of them when the requirements of modern data management are expanding.

Finally, MongoDB is revealed to be a highly flexible and highly scalable platform for Big Data. For instance, in terms of data handling capabilities, it features a document model and a schema that is quite tolerant of different types and structures of data. Through sharding, MongoDB remains horizontally scalable for clusters, and this makes the management of big data easy, especially in terms of storage and retrieval. Also, evaluating the company's performance in conditions that call for real-time data processing, namely high throughput and low latency, it is clear that MongoDB fits well into applications that involve the ingestion and processing of data in real time.

On the other hand, CockroachDB, represented a NewSQL DBMS, presents challenging features specifically designed to face Big Data challenges. CockroachDB has distributed SQL engine and provides high level of data consistency and should be preferred for OLTP systems in distributed environments. Geo-partitioning features make it possible to localize and replicate data, which is quite appropriate for applications with a worldwide scale in terms of the amount of data processed. However, CockroachDB has had-built high availability built into the architecture and can scale horizontally allowing it to be used in highly available, mission-critical environments.

Thus, both MongoDB and CockroachDB showcase contrasting positive aspects of addressing Big Data in various tasks and needs. MongoDB has many advantages for complex case with changing data schema and high read operations frequency. However, these applications call for transactional integrity, SQL compatibility, and globally distributed capabilities, which makes CockroachDB a good fit. In light of this comparative analysis, business stakeholders can take informed decisions in choosing an appropriate database solution to meet increasing volumes and kinds of data management requirements for expanding organizations in the new age economy.

Reference:

Kaur, K. and Sachdeva, M. (٢٠١٧). Performance evaluation of NewSQL databases. ٢٠١٧ *International Conference on Inventive Systems and Control (ICISC)*. [online] doi:https://doi.org/١٠.١١٠٩/icisc.٢٠١٧.٨٠٦٨٥٨٥.

Binani, S., Gutti, A., & Upadhyay, S. (٢٠١٦). ISSN : ٢٣٩٤-٤٧١٤ Foundation of Computer Science FCS. Communications on Applied Electronics (CAE), ٦(١). https://www.caeaccess.org /archives/ volume٦/number١/binani-٢٠١٦-cae-٦٥٢٤١٨.pdf

Duggirala, S. (٢٠١٨, January ١). Chapter Two - NewSQL Databases and Scalable In-Memory Analytics (P. Raj & G. C. Deka, Eds.). ScienceDirect; Elsevier. https://www.sciencedirect.com/science/article/abs/pii/S٠٠٦٥٢٤٥٨١٨٣٠ ٠١٣٥

Communications of the ACM. (٢٠٢٤). *SQL databases v. NoSQL databases | Communications of the ACM*. [online] Available at: https://dl.acm.org/doi/abs/١٠.١١٤٥/١٧٢١٦٥٤.١٧٢١٦٥٩

Li, Y. and Manoharan, S. (٢٠١٣). A performance comparison of SQL and NoSQL databases. [online] doi:https://doi.org/١٠.١١٠٩/pacrim.٢٠١٣.٦٦٢٥٤٤١.

ACM Computing Surveys (CSUR). (٢٠١٨). *A Survey on NoSQL Stores | ACM Computing Surveys*. [online] Available at: https://dl.acm.org/doi/abs/١٠.١١٤٥/٣١٥٨٦٦١

Google Books. (٢٠٢٢). *Understanding the New SQL*. [online] Available at: https://books.google.com.pk/books?hl=en&lr=&id=ZOOMSTZ٤T_QC &oi=fnd&pg=PA١&dq=new+sql+&ots=e٣Ie٩HO٦ey&sig=RwhASawqO ٧v٩s٧٤KmzqyNBNf٤oU&redir_esc=y#v=onepage&q=new%٢٠sql&f=fal se

Moniruzzaman, A B M and Hossain, S.A. (٢٠٢٤). *NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison*. [online] arXiv.org. Available at: https://arxiv.org/abs/١٣٠٧.٠١٩١

Katarina Grolinger, Higashino, W.A., Tiwari, A. and Miriam AM Capretz (٢٠١٣). Data management in cloud environments: NoSQL and NewSQL data stores. *Journal of cloud computing*, [online] ٢(١). doi:https://doi.org/١٠.١١٨٦/٢١٩٢-١١٣x-٢-٢٢

María Murazzo, Gómez, P., Rodríguez, N. and Medel, D. (٢٠١٩). Database NewSQL Performance Evaluation for Big Data in the Public Cloud. *Communications in computer and information science*, [online] pp.١١٠–١٢١. doi:https://doi.org/١٠.١٠٠٧/٩٧٨-٣-٠٣٠-٢٧٧١٣-٠_١٠