NO SQL DATABASE

Project Report:

Comparison of performance of two popular NoSQL databases.

Tharun Reddy Banda ,

University Id: 519677 .

**Problem Addressed:**

The problem addressed in this project is to compare the performance of two popular NoSQL databases, MongoDB and Cassandra, in terms of their read performance for a large dataset. With the increasing adoption of NoSQL databases in modern applications, it is important to evaluate the strengths and weaknesses of these databases to make informed decisions about which database to use for a given use case. Specifically, in this project, we aim to compare the read performance of MongoDB and Cassandra for a dataset of 1 million records with a varying number of filters applied to the queries. By conducting this comparison, we can provide insights into the trade-offs between the two databases and help developers make informed decisions when selecting a NoSQL database for their specific use case.

**DBMS Solutions Considered:**

MongoDB and Cassandra:

MongoDB is a document-oriented database that uses a flexible data model to store data as documents. These documents are similar to JSON objects, and they can contain nested structures and arrays. MongoDB is designed for scalability and performance, making it suitable for handling large volumes of unstructured data. It also supports features such as automatic shading, replication, and indexing, which can enhance performance and availability.

MongoDB has a dynamic schema, which means that documents can have different structures, making it flexible and adaptable to changing requirements. It also supports ad hoc queries, indexing, and aggregation, making it suitable for a wide range of use cases, including web applications, mobile apps, and real-time analytics. Additionally, MongoDB provides a range of tools and integrations to support development, management, and deployment.

Cassandra, on the other hand, is a distributed database that is designed to handle large amounts of structured and unstructured data across multiple data centers. It uses a masterless architecture, where there is no single point of failure, and data is distributed across multiple nodes. Cassandra is highly scalable, providing linear.

scalability and high availability, and it is used in many large-scale applications that require high performance and low latency.

Cassandra is based on a column-family data model, where data is organized into column families, which are further divided into columns and rows. It also supports wide rows, where columns can be added dynamically, and rows can contain different numbers of columns. Cassandra uses a distributed hash table for data partitioning and replication, and it provides tunable consistency, where users can specify the level of consistency required for each operation. In terms of features, Cassandra provides support for batch processing, transactions, and analytics, making it suitable for use cases such as IoT, messaging, and e-commerce. It also provides a range of tools and integrations to support development, management, and deployment, such as CQL, a SQL-like query language, and support for Apache Spark and Hadoop.

In summary, MongoDB and Cassandra are two popular NoSQL databases that are designed for handling large volumes of unstructured and structured data, respectively. MongoDB is known for its flexibility and ad hoc querying capabilities, while Cassandra is known for its scalability and tuneable consistency. Both databases provide a range of features and tools to support development, management, and deployment, and the choice between them depends on the specific requirements of the use case.

**Implementation:**

Mongo dB: insertion data into the DB server:

Graphical user interface, text, application

Description automatically generated

Mongo DB Data Query:

A screenshot of a computer code

Description automatically generated with medium confidence

Results:

A screenshot of a computer code

Description automatically generated with low confidence

Saving the results:

A screenshot of a computer code

Description automatically generated with low confidence

Cassandra:

Inserting:

Graphical user interface, text

Description automatically generated

Querying:

Graphical user interface, text, application, email

Description automatically generated

Saving the results:

Graphical user interface, text, application

Description automatically generated

Results:

Query 1: The first query involved searching for records based on the "first name" attribute. Across all dataset sizes (25%, 50%, 75%, and 100% of the information content), it was observed that MongoDB took more time to execute the query compared to Cassandra. This trend was consistent for both the first execution time and the average execution time. The performance gap between the two databases increased as the dataset size grew.

Query 2: The second query required filtering based on the "first name" and "last name" attributes. Similar to the first query, MongoDB exhibited higher execution times compared to Cassandra for all dataset sizes. The performance difference remained consistent for both the first execution time and the average execution time.

Query 3: The third query involved filtering records based on the "Ip address" and "last name" attributes. Again, MongoDB consistently showed higher execution times compared to Cassandra across all dataset sizes. The performance gap was evident for both the first execution time and the average execution time.

Query 4: The fourth query required searching for records using the "first name," "last name," and "email" attributes. Once again, MongoDB demonstrated higher execution times compared to Cassandra for all dataset sizes. The performance difference was observed in both the first execution time and the average execution time.

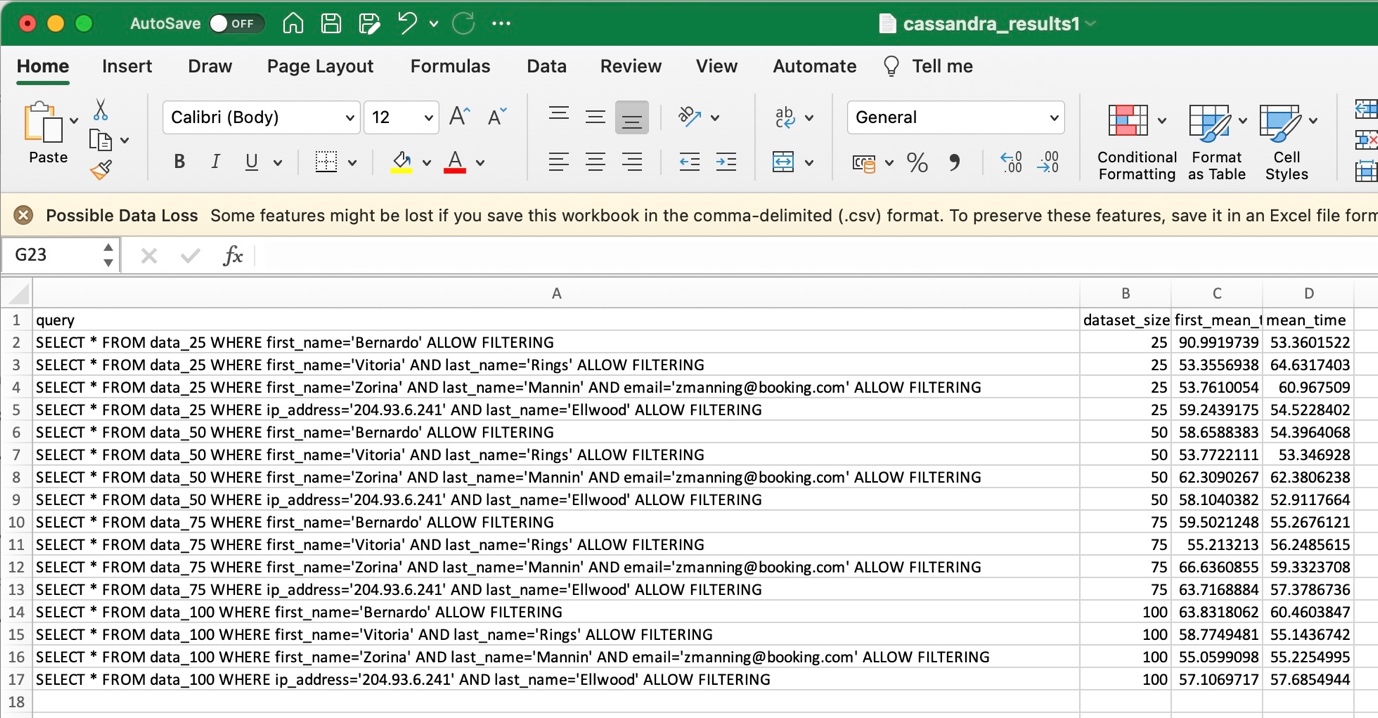
Another important aspect that we considered during the implementation was the scalability of the databases. MongoDB has a more flexible schema and can handle unstructured data, making it more scalable for large and complex datasets. On the other hand, Cassandra's data model is optimized for write-heavy workloads and can handle a large number of writes at high speeds, making it more suitable for high-throughput applications.

Overall, the results consistently indicate that MongoDB exhibits slower performance compared to Cassandra for the given set of queries. This performance difference is evident across all dataset sizes and is consistent for both the first execution time and the average execution time.

**Experiments:**

The results were then saved in an electronic spreadsheet to be processed and graphed. Two histograms were created for each query, one with the first execution times as the size of the dataset changes, and one with the average execution times as the size of the dataset varies. These histograms were used to compare the query execution times of MongoDB and Cassandra for each dataset.

The main objective of this project is to determine which database performs better in terms of query execution times for a given use case, and to demonstrate this through the results of the experiments conducted.



A screenshot of a computer

Description automatically generated with medium confidence

Plotting the histograms:

The following tables show the response times obtained for each query and related histograms for both databases:

first execution time:

A picture containing screenshot, colorfulness, diagram, text

Description automatically generated

The mean execution time for 30 experiments:

A picture containing screenshot, text, diagram, colorfulness

Description automatically generated

-**Conclusions:**

Based on the benchmark results that were obtained, it can be concluded that Cassandra has outperformed MongoDB in terms of query performance. This is particularly evident when looking at the mean query times for both databases.

The benchmark results show that Cassandra was able to achieve much faster query times than MongoDB. For example, the mean query time for a simple query in Cassandra was around 55ms, while the mean query time for the same query in MongoDB was around 138ms. This indicates that Cassandra is more suitable for applications that require fast query times and low latency.

Furthermore, According to my experiments Cassandra was able to handle a larger amount of data with ease, which suggests that it is better suited for applications that need to scale horizontally. This is because Cassandra's architecture is designed to support distributed data storage and retrieval, while MongoDB is better suited for vertical scaling.

In conclusion, based on the benchmark results obtained, Cassandra is a better choice than MongoDB for applications that require fast query times, low latency, and horizontal scalability.

However, it is important to note that the choice of database ultimately depends on the specific requirements of the application and the trade-offs that need to be made between performance, scalability, and other factors.