COURSEWORK 2: DATA PROCESSING

MongoDB

Richa Ranjan

University of Southampton

Foundations of Data Science

rr2n17@soton.ac.uk.com

ABSTRACT

This paper provides an insight into using NoSQL databases. The next generation NonSQL (NoSQL) databases are mostly non-relational, distributed and horizontally scalable. The main characteristics of these databases are schema-free, no join, non-relational, easy replication support, simple API and eventually consistent. [5]

INTRODUCTION

The *proceedings* are the details of suitability of different types of databases for Data Science applications. The main limitations with conventional relational database management systems (RDBMS) are that they are hard to scale with Data warehousing, Grid, Web 2.0 and Cloud applications, have non-linear query execution time, have unstable query plans and have static schema. Even though RDBMS's have provided database users with the best mix of simplicity, robustness, flexibility, performance, scalability and compatibility, they are not able to satisfy the present day users and applications for the reasons mentioned above.[5]

The need for Non-Relational Databases

Relational Databases (RDB), based on the relational model have been in use for more than 30 years ago. These were designed mainly to serve business data processing. Since then it has been the best option for storing information that range from financial records, personal data and much more. However, with the advent of web 2.0, many new applications that depend on storing and processing huge amount of data have emerged. These applications need high availably and scalability which added more challenges to the RDB [1]. To handle a huge volume of data like internet, multimedia and social media the use of traditional relational databases is inefficient. The NoSQL term was coined by Carlo Strozzi in 1998 and refers to non-relational databases term, which was later reintroduced in 2009 by Eric Evans. More recently, the term has received another meaning, namely "Not Only SQL". [2] Many Companies have adopted non-relational databases for their applications. For example, Yahoo with **PNUTTS** to meet massively parallel and geographically distributed database system for their web applications, Facebook with **Cassandra** and Google with **BigTable**.

1 RELATIONAL DATABASE VS NOSQL

NoSQL databases disrupted the database market by offering a more flexible, scalable, and less expensive alternative to relational databases. They also were built to better handle the requirements of Big Data applications. [3] NoSQL Databases differ from the older, relational databases in the following main areas:

1.1 Data Models

NoSQL allows storing the data without having to create a schema first. Whereas in case of Relational Databases, the model is very specific, well-organized and has a pre-defined schema. NoSQL databases take many modeling techniques like:

- **Key-value**: These are conceptual distributed dictionaries. E.g: Riak.
- **Document-based**: These databases work with documents of different type, like JSON, BSON, XML and BLOBs. E.g.:Couchbase and MongoDB.
- **Column-oriented**: Databases from BigTable category, such as HBase and Hypertable are columnar type and must have a predefined schema. [2]
- **Graph-oriented**: These databases are meant for complex data queries. E.g.: Neo4j

No predefined schema makes NoSQL databases much easier to update as the data and requirements change. [3]

1.2 Data Structure

Relational databases are designed to handle data that are clearly defined and have a tabular form of storage. On the other hand, NoSQL databases are designed to handle unstructured data. With the growing advent of internet and mass media communications, the need of the hour is to handle huge, unstructured data, like images, audio-visual data, social media posts etc.. NoSQL databases are designed to handle these kind of data. Thus, they implement methods to improve the performance while storing and retrieving data.

1.3 Scalability

NoSQL database requires horizontal scaling (increase in the number of commodity nodes or system units). Relational Databases, on the other hand, require vertical scaling (addition of more hardware i.e. bigger and expensive servers). Thus, scaling a NoSQL database is easier and cost-effective.

1.4 Data Warehouses

Relational Databases are used for typical data warehousing problems. They are used mainly where the data is gathered from various sources and processed and stored finally in warehouses. These warehouses are used for OLTP(On-line Transactional Processing) and OLAP(On-line Analytical Processing) applications. NoSQL databases are not designed for such applications as the main focus in this case is on high performance, scalability and big data storage.

1.5 Cloud Compatibility

Relational databases are not very well-suited for cloud environments, because they do not support full content data search. NoSQL on the other hand are suitable for such applications because the cloud databases are not **ACID** (*Atomicity, Consistency, Isolation* and *Durability*) compliant and provide improved availability, scalability, performance and flexibility.[1]

2 COSTS AND BENEFITS OF USING NOSQL

Efficient storage and retrieval of data has always been an issue due to the growing needs in industry, business and academia. Larger amounts of transactions and experimentation result in massive amounts of data which require organized storage solutions. [4] To understand the benefits of using NoSQL database, we can make a comparison of the performance. The most important factor for an application is the time required for completing a task, and in the case of databases the time required to complete a transaction. According to the **Benchmark Harnessing** process, to measure the performance of the databases and their scalability the metric of total queries per second is used as it allows to extrapolate how well the databases cope with different loads and different numbers of connections. To calculate the queries per second, the following formula can be used:

$$Queriespersecond = \frac{TotalNumberofQueries * TotalNumberofthreads}{AverageQueryTime}$$

[4]

2.1 Database operations

In general, performance tests have been carried out to compare how databases perform while carrying out basic operations like:

- Insert

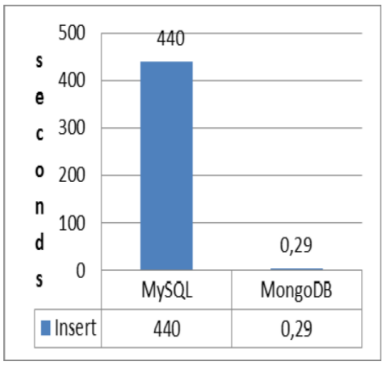

Fig. 1. MySQL vs. MongoDB insert

Figure 1: A typical insertion example of MongoDB and SQL database

[2]

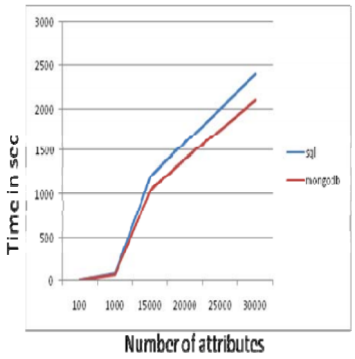

Figure 2: A typical Select query example of MongoDB and MySQL database

[5]

- Update
- Delete

For example, in case of an insertion operation, tests were carried out as mentioned in the article "A Comparative Study: MongoDB vs. MySQL" [2] and the results were cited as shown in the following figure: In the above mentioned article, the result shows that insertions using MongoDB was a lot faster compare to the SQL database. Similar experiments were carried out on other operations as well. As mentioned in the article "Comparison of Relational Database with Document-Oriented Database (MongoDB) for Big Data Applications" [5] , the Select operation results on MongoDB and MySQL were cited, as mentioned in the figure 2: The above examples can serve as evidences that the performance of NoSQL databases are, in general, better than that of RDBMS. Generically speaking, the Non relational databases outperform the RDBMS in the following areas:

- **Complexity:** The storage of data in RDBMS can be quite complex owing to the fact that the data has to be stored strictly in a relational format. If the data does

not fit into those tables, it would be quite complex to work with them. A NoSQL database, is however, capable of storing unstructured and semi-structured data efficiently.
- **Scaling costs:** As discussed already, a relational database requires vertical scaling. Therefore, addition of more hardware is definitely not a preferred approach, when compared to the horizontal scalability of NoSQL databases. Needless to say, NoSQL databases are cost effective in this respect.

2.2 CAP Theorem

Additionally, the CAP theorem must be mentioned. Eric Brewer, in 2000, introduced the idea that there is a fundamental trade-off between *Consistency*, *Availability*, and *Partition tolerance*. The theorem says that only two of these aspects can be guaranteed at the same time in a distributed system. You have to pick two of them.[1] NoSQL has been instrumental in data storage and retrieval with respect to the above theorem.

2.3 NoSQL: A Better fit for the Enron Data?

In my opinion, the NoSQL database has been a great fit for the Enron data used in this coursework. As discussed earlier in this paper, the storage capability of a relational database is only in terms of tables and schemas. MongoDB, on the other hand, provided a better framework to work with the Document-structured data. The benefits observed with MongoDB are as follows:

- **Efficient Storage of Document-based data:** The Enron data was defined in the form of documents, stored in the form of collection, in a JSON format. This is out of scope for a relational DB.
- **Schema-less Model:** MongoDB, being a schema-less model, had no constraints like referential integrities, or unique constraints.
- **Easy retrieval of data:** The retrieval/querying process was efficient, with less time required.
- **Open source:** NoSQL databases are open source whereas any Relational Database would require software installation costs to get started.

3 PREFERRED SELECTION OF DATABASE

MongoDB, as mentioned earlier, has been efficient in handling the Enron data and related operations. In my view, it has been a good choice for this particular dataset. Although, there have been several claims about the benefits of using databases like the Postgre database in such applications, I would still stand by the selection of Mongo DB for the Enron Data Management. Postgre, or any other database that can handle JSON data, could have been used as an alternative for this coursework. However, I would prefer MongoDB for various reasons:

3.1 Distributed Architecture

MongoDB is built around a distributed architecture that can be run within and across geographically distributed data centers and cloud regions, providing levels of availability and scalability. As databases grow in terms of data volume and throughput, MongoDB scales elastically with no downtime, and without application changes. And as an application's performance and availability goals evolve, MongoDB allows users to adapt flexibly, across data centers, with tunable consistency. [6]

3.2 Always-On Availability

MongoDB maintains upto 50 replicas of the data, thus providing a much better availability and recovery management. Also, in the event of failure of these replicas, the recovery process is done within no time, without the intervention of the operator.

3.3 Developer Productivity

On a personal level, this factor has been one of the most important reasons for me to choose MongoDB over any other database. Unlike the syntactical restrictions of any other Relational database systems, MongoDB provides an easy, "dot format" programming approach, which is quite similar to the present day programming languages. Thus, it makes it easier to use even for a novice.

4 CONCLUSION

For this application, the most suitable database was MongoDB, because of its flexible and dynamic structure. Apart from its advantages like availability, recovery, scalability, the most noteworthy factor in my opinion was its ease of access. A huge number of documents were queried and retrieved in seconds. Also, with no unique key constraints and schema restrictions, it was easier to work with the data. Overall, MongoDB has been a good fit for this application.

REFERENCES

[1] Relational vs. NoSQL Databases: A Survey, Mohamed A. Mohamed, Obay G. Altrafi, Mohammed O. Ismail, International Journal of Computer and Information Technology (ISSN: 2279 – 0764)Volume 03 – Issue 03

[2] A Comparative Study: MongoDB vs. MySQL: Cornelia Gyrödi, Robert Gyrödi, George Pecherle, Andrada Olah,2015 13th International Conference on Engineering of Modern Electric Systems (EMES)

[3] https://www.mongodb.com/scale/nosql-vs-relational-databases

[4] RDBMS vs NoSQL: Performance and Scaling Comparison: Christoforos Hadjigeorgiou, RDBMS vs NoSQL: Performance and Scaling Comparison, August 23, 2013

[5] RDBMS vs NoSQL: Comparison of Relational Database with Document-Oriented Database (MongoDB) for Big Data Applications, Satyadhyan Chickerur,Anoop Goudar, Ankita Kinnerkar, 2015 8th International Conference on Advanced Software Engineering And Its Applications

[6] https://www.mongodb.com/