Quantitative Analysis of Document Stored Databases

Pradeep Soni
(Research Scholar)
Department of Computer Science & Engineering
SBCET, Jaipur, Rajasthan, India

Narendra Singh Yadav, PhD
(Assoc. Prof. & Head)
Department of Computer Science & Engineering
SBCET, Jaipur, Rajasthan, India

ABSTRACT
So far relational databases are used for storing the data for the applications but now there is need to store huge amount of data to store and manage which cannot stored by relational databases. NoSQL technology over comes this problem. This research paper provides a brief introduction to NoSQL database working and comparative study between MongodB and Cassandra, Which are mostly used for big data application. The operations are performed on Ubuntu system to explore the results as distinguish between both NoSql databases. This paper shows the performance of Mongodb and Cassandra. Results proves that Cassandra is more powerful than Mongodb to load and process on big data and processing very fast as compare to Mongodb. This paper describes the functionality of Mongodb and Cassandra over the large dataset.

Index Terms— NoSql Databases, Mongodb, Cassandra, Big Data.

1. INTRODUCTION
NoSql Technology: NoSQL stands for Not only SQL. It provides the different kind of databases like document databases, graph databases, key value databases which are used for massive dataset such as big data applications. This technology is very easy to use in conventional load balanced clusters and persistent data. This is easy to scale at very massive level to available memory. NoSQL does not have any type of fixed schema and allows schema migration without downtime. Relational databases are not able to scale at very large scale as compare to NoSQL databases.

1.1 Document Database: Document Database objects manage data types easily and can be embedded documents and arrays to reduce need for joins. These databases use the dynamic schema to make polymorphism easier. These databases have the strength to perform on web application and give the best results when the incomplete data is given.

1.1.1 Mongodb: it is a document database that provides high performance, high availability, and easy scalability. Mongodb database is easy to embed that makes reads and writes fast. This database uses the indexes that include keys from documents and arrays. This provides the high availability for higher performance and very easy to scale and easy to manage the operations.

MongoDB stores the data into documents and collections instead of storing data in table as rows and columns. Collections allow representation of complex relationships easily. Is has the capability to handle the large volume of data and can load data across a cluster. It can perform many operations which relational database cannot do.

There are some following features of MongoDB

- Map reduce and Aggregation Tools are supported by Mongo DB.
- Java Scripts can be used instead of Procedures
- Mongo DB is a schema less Document based database.
- Mongo DB provide the facility to use secondary indexes and geospatial indexes.
- Easy to handle the Mongo DB in cases of failures
- Mongo DB designed to provide High Performance
- MongoDB stores files of any size without down to failure of memory.

1.1.2 Apache Cassandra: Cassandra a type of NoSQL database which is massively scalable. As technical aspects Cassandra can be found at companies recognized for their ability to manage big data effectively –Amazon, Google and Facebook.

In today's environment Cassandra is used for modern businesses to handle their critical data infrastructure, and known for being the solution for the technical professionals when they require a NoSQL database that gives high performance at massive scale, that never degrades the performance of operations. Cassandra is used for unstructured data as big data application, which are mostly used across nearly every industry.

This model is a partitioned in row store with consistency. [2] These are arranged into tables, primary key is assigned always as first component and rows are clustered in the remaining fields of the key. [3] Columns are indexed through primary key. [4]

Tables may be structured, deleted, and modified at runtime without blocking updates and queries. [5]

Joins and sub queries are not supported by the Cassandra except for batch analysis via Hadoop, rather it performs denormalization through features like collections. [6]

Features:

Decentralized
Data are distributed across the cluster and each node contains different data so there is no single point of failure.

Supports for replication and multi data center replication
These strategies can be configurable. [7] Cassandra is designed as a distributed system, for deployment of large numbers of nodes across multiple data centers.

Scalability
It provides the reads and writes on the large dataset with no downtime to applications.

Fault-tolerant
In Cassandra data is replicated to multiple nodes automatically that make it faul-taulerance. Replication across multiple data centers is supported.

MapReduce support
MapReduce can be perform on Cassandra also support for Apache Pig and Apache Hive.

1.2 JSON: JSON stands for JavaScript Object Notation. It is extended from the JavaScript scripting language. It is very easy to read and write.it is lightweight text based interchangeable format. This is Language independent. It is used as document file for Mongodb database.

```
{
cust_id:"001",
cust_name:"Mukul",
price:200,
status:"1"
}
{
cust_id:"002",
cust_name:"Pradeep",
price:256,
status:"1"
}.
```

1.3 MapReduce: MapReduce is a framework for effectively processing the analysis of big data on several servers. It was developed by the Google for the back end of Google's search engine to enable a large number of commodity servers to efficiently process the analysis of huge numbers of webpages collected from all over the world. Apache developed a project to implement MapReduce, which was published as open source software (OSS), this enabled many organizations, such as businesses and universities, to tackle big data analysis.

It was originally developed by Google and built on well-known principles in parallel and distributed processing. Since then Map Reduce was extensively adopted for analyzing large data sets in its open source flavor Hadoop.

MapReduce [8] is a simple programming model for processing huge data sets in parallel. MapReduce have master/slave architecture. The basic notion of MapReduce is to divide a task into subtasks, handle the sub-tasks in parallel, and aggregate the results of the subtasks to form the final output.

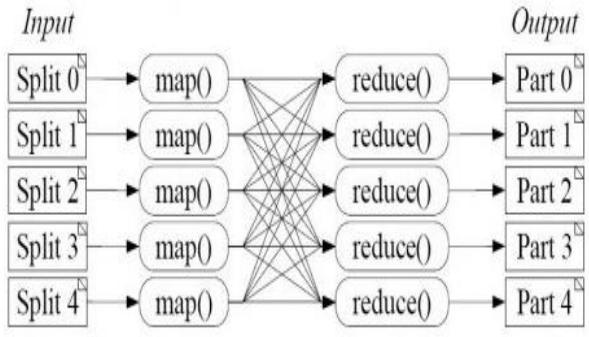

Figure 1: MapReduce Architecture

In most computation related to high data volumes, it is observed that two main phases are commonly used in most data processing components this is shown in above figure 3. Map Reduce created an abstraction phases of Map Reduce model called 'mappers' and 'reducers' (Original idea was inspired from programming languages such as Lisp).

1.3.1 MapReduce using MongoDB
Consider the following document structure storing customer information. The document stores cust_id, price and status of the customer.

```
{
cust_id:"002",
cust_name:"Pradeep",
price:256,
status:"1"
}
```

Now, we will perform a mapReduce function on data collection to select all the active status, group them on the basis of cust_id and then find the sum of price of data by each user using the following code:

```
db.data.mapReduce
(
 function()
 {
 emit (this.cust_id, this.price);
 },
 function(key, values) {return Array.sum(values)},
 {
 query: { status:"1"},
 out:"total"
 }
)
```

This operation shows the following output.

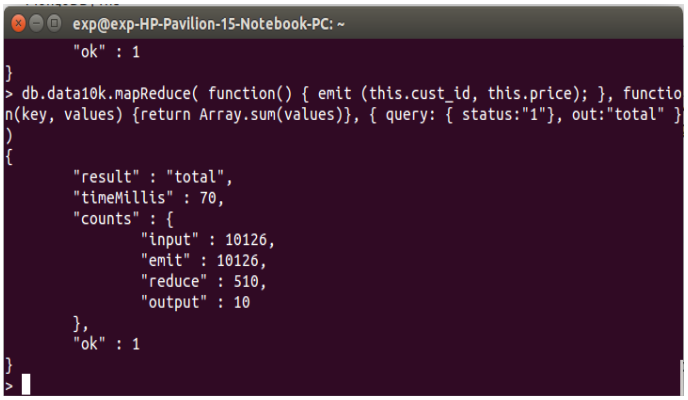

Figure 2: Output Screen of MapReduce Operation

2. LITERATURE SURVEY

In 2012, University of Toronto researchers studying NoSQL systems concluded that Cassandra is most powerful in terms of scalability throughout the experiments. It achieves the maximum throughput for the number of nodes. "It comes at the price of high write and read latencies.[1]" Mongodb database stores the large amount of data in JSON format and performs the operations that will give results quickly.

3. PROPOSED APPROACH

Mongodb and Cassandra both are NoSQL databases which are used when data is huge. Here JSON file is used to store large amount of data. On which Mongodb operations are performed such as MapReduce, Insertion, Deletion and Updation.

In case of Cassandra there is database created by using the CQL (Cassandra Query Language). This database contains the same data as JSON file has.

There are for collections created in Mongodb. First collection contains 50k records, second collection contains 100k records, third collection contains 500k reocrds and fourth collection contains the 1000k records in it.

Cassandra follows the same scenario that uses four tables and having same amount of records as Mongodb has.

Open Source (Ubuntu) platform is used to perform the operations on Mongodb and Cassandra. These databases provide the high speed and high throughput as compare to relational databases.

4. EXPERIMENTAL SETUP

In this research, all the tests are performed under following specifications:

1) **Host System:** Intel i5 processor with 6 GB RAM and 500 GB Hard disk.
2) **Operating Environment:** Ubuntu 14.10 LTS
3) **Mongo DB**
4) **Cassandra**

a) **Execution Time:** Execution time can be defined in terms of time consumed by an algorithm in order to solve a problem using processor p.

5. RESULTS AND ANALYSIS

Experiemnt-1: Performing mapReduce function on data collection to retrieve all the active status, group as cust_id and then calculate the sum of price of data by each user using the following code:

```
db.data50k.mapReduce
(
 function()
 {
 emit (this.cust_id, this.price);
 },
 function(key, values) {return Array.sum(values)},
 {
 query: { status:"1"},
 out:"total"
 }
)
```

This operation can be perform in Cassandra as following query

Select cust_id, SUM(price) from data50k group by cust_id

Table: 1 Execution Time for MongoDB & Cassandra for Data Retrieval

Records	MongoDB	Cassandra
50k	0.322	0.245
100k	0.915	0.469
500k	3.536	2.47
1000k	6.235	4.356

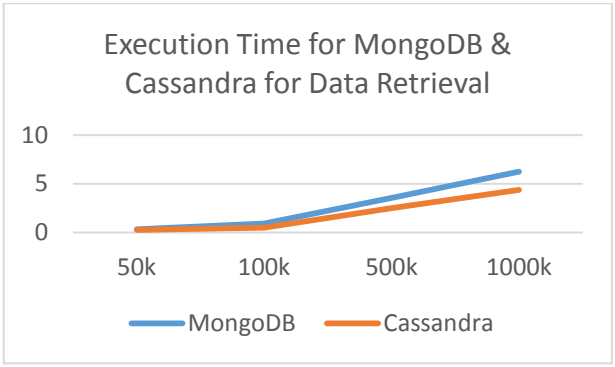

Figure 5.1: Execution Time for MongoDB & Cassandra for Data Retrieval

From the figure 5.1 it is clear that execution time taken by Cassandra is better than MongoDB for different numbers of records. As the number of records increases performance of Cassandra is also increased for the data retrieval operation in comparison to MongoDB.

Experiment-2:- to perform the update operation in Mongodb, the query is like to update the data50k collection where cust_name is 'pradeep'. So code as follows

db.data50k.update({'title':'pradeep'},{$set:{'title':'mukul'}}, {multi:true})

In Cassandra CQL code as follows

Update data50k set cust_name="mukul" where cust_name="pradeep"

This code will changes the cust_name "pradeep" to "mukul".

Table: 2 Execution Time for MongoDB & Cassandra for Data Updation

Records	MongoDB	Cassandra
50k	0.253	0.214
100k	0.623	0.365
500k	0.899	0.478
1000k	1.785	0.898

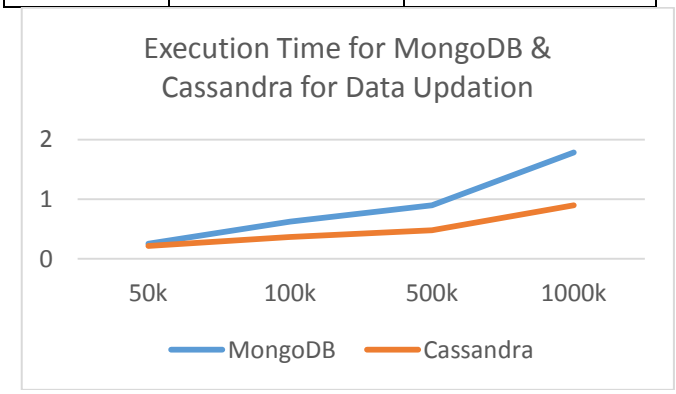

Figure 5.2: Execution Time for MongoDB & Cassandra for Data Updation

From the figure 5.2 we can analysis that execution time taken by Cassandra is better than MongoDB. As the number of records increases performance of Cassandra is also increased for the data updation operations in comparison to MongoDB.

Experiment-3:- the following code is used to delete the data in Mongodb and cassandra databases.

db.data50k.remove()

in Cassandra CQL code will be as

delete from data50k

this query will delete all the data from the data50k table.

Table: 3 Execution Time for MongoDB & Cassandra for Data Deletion

Records	MongoDB	Cassandra
50k	0.144	0.124
100k	0.259	0.196
500k	0.596	0.248
1000k	1.235	0.685

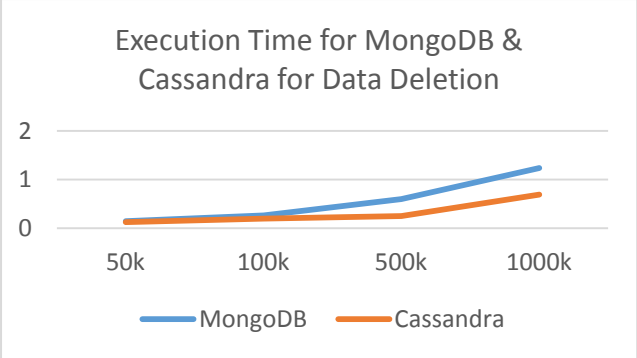

Figure 5.3: Execution Time for MongoDB & Cassandra for Data Deletion

From the figure 5.3 we can analysis that for less number of records the execution time for Cassandra and MongoDB is not very different but as the number of records increases performance of Cassandra is also increased for the data updation operations in comparison to MongoDB.

Experiment-4:- the following code is used to insert the data in collection of Mongodb database.

```
db.data50k.insert(
        {
        cust_id:"001",
        cust_name:"Mukul",
        price:200,
        status:"1"
}
)
```

in Cassandra CQL code will be as

insert into data50k (cust_id, cust _name, price, status) values(001,'mukul', 200,'1');

this query will insert the data into data50k table.

Table: 4 Execution Time for MongoDB & Cassandra for Data Creation

Records	MongoDB	Cassandra
50k	0.175	0.136
100k	0.247	0.211
500k	0.356	0.314
1000k	0.869	0.742

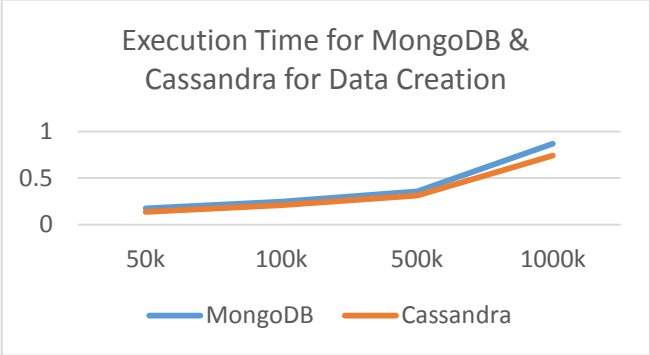

Figure 5.4: Execution Time for MongoDB & Cassandra for Data Creation

From the figure 5.4 we can analysis that execution time taken by Cassandra and mongoDB is almost similar. As the number of records increases performance of Cassandra is also increased for the data creation operations in comparison to MongoDB.

6. CONCLUSION

As the number of records in database increases, the difference between the execution time taken by Cassandra for the computation of different database operations is better in camparison to MongoDB.

For the data retrieval operation, the performance of Cassandra is about 50% better in comparison with MongoDB, for the different numbers of records.

For data updation operation as the number of records increases the performance of cassandra is also increases in comparison with MongoDB significantly. For data updation cassandra is almost 70% faster than MongoDB.

For data deletion cassandra is almost 70% faster than MongoDB for the different numbers of records.

While performing data creation operation cassandra is about 18% better than MongoDB, which is the least performance of cassandra over MongoDB.

Collectively we can say that for all database operations cassandra is much better than MongoDB, even the number of records are less or large.

7. FUTURE SCOPE

The present and future of NoSQL technologies are bright, and full of opportunities and great challenges as it processes big data.

In future we can compare these two document based databases for the different types of documents such as xml, json and csv. We can also compare them with other NoSQL document databases as couchDB and RavenDB.

8. REFERENCES

[1] Rabl, Tilmann; Sadoghi, Mohammad; Jacobsen, Hans-Arno; Villamor, Sergio Gomez-; Mulero -, Victor Muntes; Mankovskii, Serge (2012-08-27). "Solving Big Data Challenges for Enterprise Application Performance Management". VLDB. Retrieved 2013-07-25. In terms of scalability, there is a clear winner throughout our experiments. Cassandra achieves the highest throughput for the maximum number of nodes in all experiments... this comes at the price of high write and read latencies

[2] DataStax (2013-01-15). "About data consistency". Retrieved 2013-07-25.

[3] Ellis, Jonathan (2012-02-15). "Schema in Cassandra 1.1". DataStax. Retrieved 2013-07-25.

[4] Ellis, Jonathan (2010-12-03). "What's new in Cassandra 0.7: Secondary indexes". DataStax. Retrieved 2013-07-25.

[5] Ellis, Jonathan (2012-03-02). "The Schema Management Renaissance in Cassandra 1.1". DataStax. Retrieved 2013-07-25.

[6] Lebresne, Sylvain (2012-08-05). "Coming in 1.2: Collections support in CQL3". DataStax. Retrieved 2013-07-25.

[7] "Deploying Cassandra across Multiple Data Centers". DataStax. Retrieved 11 December 2014.

[8] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters", Commun. ACM, Pages:107-113, 2008.

9. ABOUT AUTHORS

Mr. Pradeep Soni is a Microsoft Certified Technology Specialist. He has 2.5+ years' experience in development with Microsoft Technologies. He is Pursuing his Master of Technology Degree in Computer Science. His area of research are Parallel Programming, Networking and DSA.

Dr. Narendra Singh Yadav received M.Tech. degree in Computer Science from Birla Institute of Technology, Ranchi, India in 2002 and completed Ph.D from Malaviya National Institute of Technology, Jaipur, India in 2011. He is an Associate Professor and Head of Department of Computer Science & Engineering in SBCET, Jaipur. He is an active member of various professional bodies. His research interests include Clustering, Routing and Security in ad hoc wireless networks, wireless sensor and wireless hybrid network.