Data Storage Management
Project B
Mongodb and Cassandra

I. Introduction

MongoDB: It is an open source database where high performance, automatic scaling and high availability is provided. The data is stored in the key-value pairs form. The mechanism of this database is, document-oriented storage. Being the cross-platform database, it can be installed on different platforms. MongoDB is a NoSQL database [1].

Document based storage: Document is a data structure with value pairs and fields which is same as JSON objects.

Cassandra: Similar to MongoDB, Cassandra is also an open source column family database which is highly scalable and high-performance database. Large data can be handled by the commodity servers to deliver high availability without single point of failure. This database is also a NoSQL database [2].

II. Key Characteristics

MongoDB:

- High performance- With the document data model input/output operations are less as compared to relational databases. Due to indexing, select queries are quite fast as embedded documents keys are included to indexes.

- Rich Query Language- Major CRUD operations are supported by this language. Apart from CRUD operations this language also delivers feature as data aggregation and text search on string content.

- Replication- MongoDB's replication ability is called as replica set where automatic failover and redundancy are part of it. On a server failure data can be restored using replica. As the replica set have many MongoDB servers, that have the similar set of data which increases the high availability.

- Horizontal Scalability- To distribute the data across multiple servers sharding is the method. So, to deploy a large set of data MongoDB uses sharding. In horizontal scaling the dataset is divided and loaded on the several servers. Individual server handles the load which provides better efficiency as compared to single high capacity server. This is possible only because of sharding.

- Multiple Storage Engine- Multiple storage engines are supported by MongoDB. They manage the data storage on disk as well as memory. As different engines execute better for a particular workload, application impact on the performance should be considered, that is why choosing the engine is an important task [1].

Cassandra:

- High Scalability- Cassandra has a master less design (data can be read and written on any of the node). This provides operational easiness and to scale-out easily.

- Linear scale performance- Performance of Cassandra increases as the nodes are increased.

- Fault detection and recovery- Restoring and recovering the failed nodes can be easily done in Cassandra.

- Continuous availability- As the data is replicated on different nodes, it eliminated the chances of single-point failure and provides the continuous availability.

- Data compression- Data can be compressed up to 80% without any overhead (performance overhead).

- Multi Datacentre replication- Data can be replicated across different data centres.

- Data protection- Cassandra gives strong data protection. It has commit log design and backup-restore method. With this mechanism the data is protected [3][4].

III. Database Architecture

MongoDB:

MongoDB stores the data in the document as extended JSON which is known as BSON (Binary JSON). The document which are having same structure are ordered as collections. All the data from one document can be provided by MongoDB documents.

Schema in MongoDB is dynamic as each of the document have different fields. Thus, the modelling of the data is flexible. MongoDB services the dynamic schema in which, without updating the central system catalog new fields can be added to a

document. Other document in the system can be unaffected without any downtime. Schema should be designed according to user requirements.

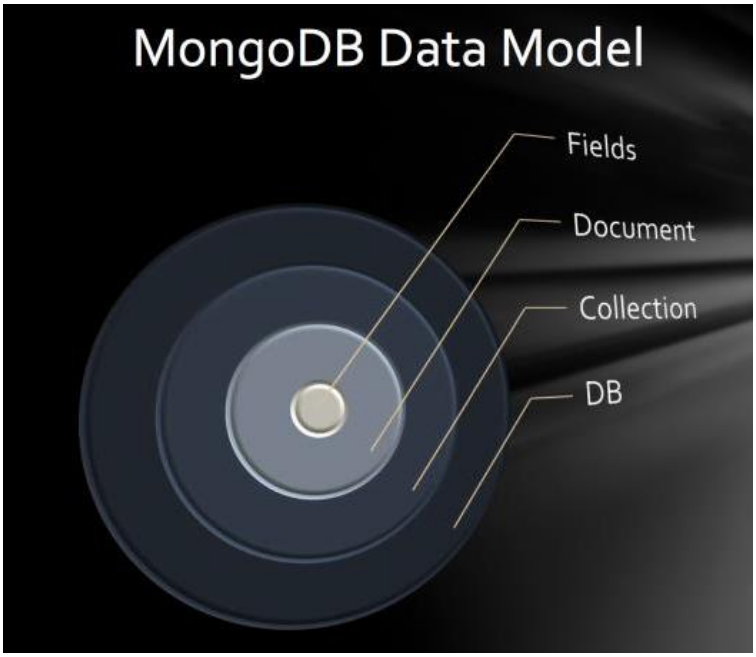


Figure 1. MongoDB Model

MongoDB Shell is a JavaScript shell which is interactive. All the commands that are supported including administrative functions can be used on the shell.

Sharding- For horizontal scaling of database, sharding is employed by MongoDB. Data is balanced across the multiple physical partition. These partitions are known as shards. This makes the database highly scalable. There are two strategies for sharding-

Range based sharding – Using Shared key value documents are distributed.
Hash based sharding – Using MD5 hash of the shard key value documents are distributed.

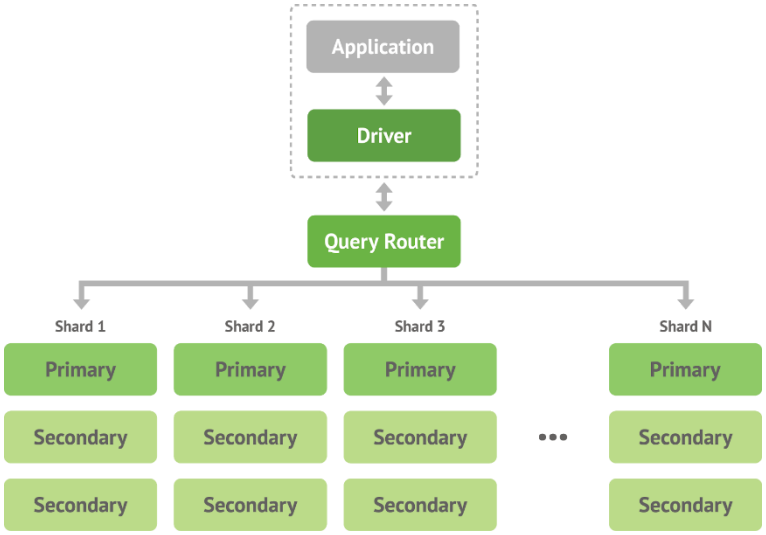


Figure 2. MongoDB Overview

From the figure above, the data is acrossed with the multiple shards. Each shard is having primary and secondary copy which are replicas [1][5].

Cassandra:

Cassandra has masterless ring architecture. This architecture is easy to set up and maintain. In the cluster the nodes acts like replicas. Most recent value is returned to the requesting application. If some of the node not having the latest value, in the background ‘read repair’ is performed and the value on that node is updated.

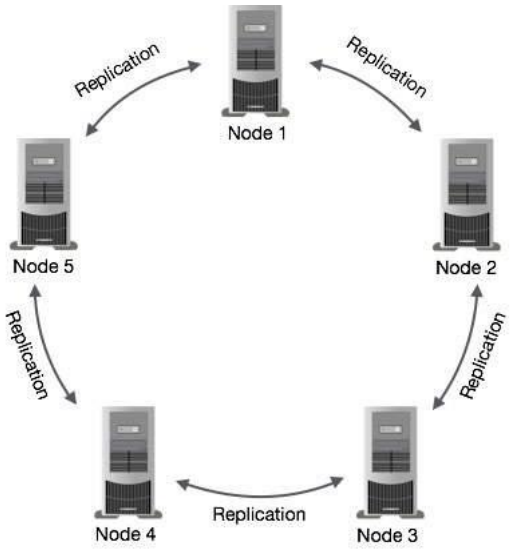


Figure 3. Cassandra Architecture

Components:

Node- Data is stored on the nodes.

Data centre- Collection of the nodes is a data centre.

Cluster- Cluster has one or multiple data centres.

Commit log- It is a crash recovery method. Commit log maintains all the write operation log.
Mem table and SST table- After writing commit log data is stored on mem tables. When the mem table reaches threshold, the data is flushed to the SST table. SST is nothing but the disk file. There can be multiple mem tables also.
Bloom filter- Bloom filter is the cache used for the quick access. For any read request this component is checked first for having the requested data.

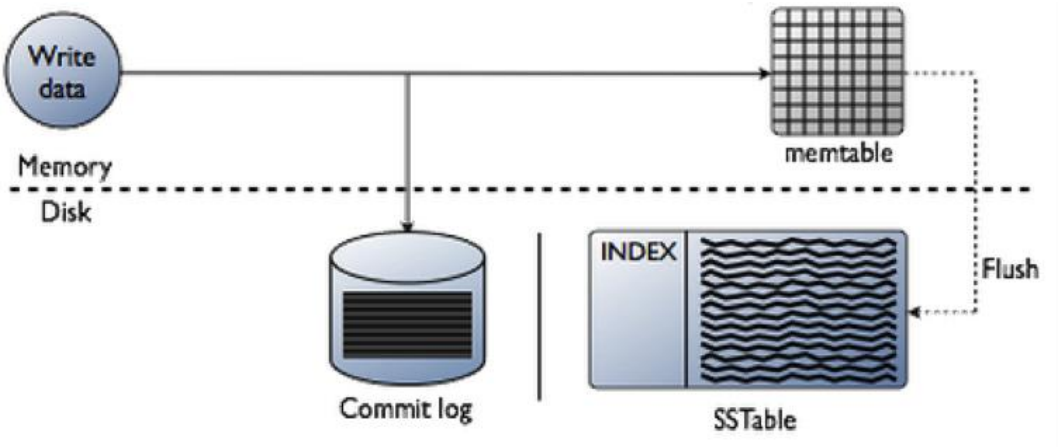


Figure 4. Cassandra Read/Write

With this mechanism many(MB) disk I/O operations occur at the same time which gives Cassandra fabulous writing performance.

Data distribution- Partitioner is an internal component of Cassandra which distributes the data across the nodes.
Replication- This mechanism is simple to configure and maintain in Cassandra. Cassandra can replicate the data multiple nodes in a cluster. This guarantees availability, reliability and fast input/output operations.

Cassandra Query Language(CQL)- This language is similar to SQL. DDL and DML operations are supported. There are different command line tools like cqlsh are used to interact with the cluster.

IV. Security

MongoDB

Security features of MongoDB:

- Authentication: The most important and vital security measure in any database implementation is authentication to determine what the user is allowed for authorization. MongoDB has authentication in integration with external security mechanism including LDAP, Windows active directory, Kerberos and x.509 PKI certificates.

- Authorization: Authentication is requirement for authorization. Each user is assigned roles. These roles define it can configure granular permissions for a user or application based on job.
- Encryption: Data on MongoDB can be encrypted on the network and on disk. There are two different types of encryption which MongoDB supports. MongoDB uses AES256-CBC which is Advanced Encryption Standard.
- Auditing: The most important part of security which allows database administrators to track history of database is Auditing. To track access and operations executed on the database that runs for regulatory compliance, a native audit log lets you track it.
- Network Exposure: Network exposure should be limited to run MongoDB, it is achieved by running MongoDB on trusted network and allowing only clients which are trusted to link with the network [7][8].

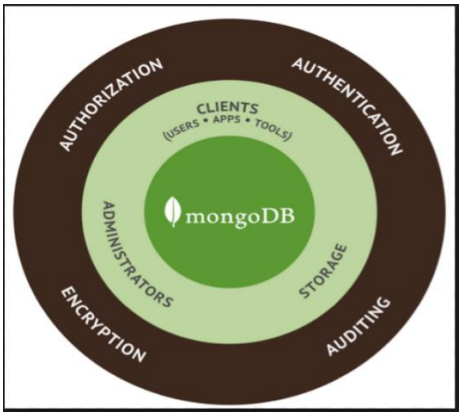


Figure 5. Security in MongoDB

Cassandra

Security in Cassandra:

Cassandra provides security using following three components:

- TLS/SSL Encryption

  TLS (Transport layer security) and SSL (Secure sockets layer) are the way of communication methods between the end user and database engine. Cassandra ensures secure association between end user and cluster containing database. Encryption can take place between nodes and cluster of the database and it can be managed by configuring each of the encryption manually depending upon the requirement. Encryption takes place with the help of cipher suite of JVM supported protocols which can be changed in settings of cassandra.yaml file. For inter-node encryption, the settings must be changed in server_encryption_options of Cassandra.yaml file. For client node encryption settings must be changed in client_encryption_options of Cassandra.yaml file.

- Authentication

  Authentication is achieved in Cassandra with the help of authenticator config file in Cassandra.yaml. By default, Cassandra authenticates user with AllowAllAuthenticator which doesn't authenticate any user and don't require any credentials to login. This file is configured to disable the authentication. It also

consists of PasswordAuthenticator which requires username and password credentials to login to database.

- Authorization
  Authorization security in Cassandra allows user to access only the necessary files. It is configured by changing the settings of authorizer in Cassandra.yaml file. Just like authentication, authorization consists of two methods such as AllowAllAuthorizer which doesn't check any credentials and authorizes each and every role to access all of the permissions. Second default authorization includes CassandraAuthorizer and authorizes role for given set of permission.

- Roles
  It also gives clients the traditional security feature of creating the database roles. Based upon the particular role in database, administrator assigns set of permission and authority to given user. User cannot access the revoked commands from it. Roles can be assigned using role_manager file in Cassandra.yaml file [6].

V. Performance Test Plan

Install YCSB

1. Switch to hduser-
   sudo su – hduser

2. Download YCSB-curl -O –location
   https://github.com/brianfrankcooper/YCSB/releases/download/0.11.0/ycsb-0.11.0.tar.gz

3. Untar the zip file-
   tar xfvz ycsb-0.11.0.tar.gz

Configure Testharness tool

1. Using pscp, copy the testharness zip from local to the server-

   pscp -i /path/to/the/.ppk/file \path\of\testharness\ ubuntu@ip:/home/hduser/

2. After copying, in the runtest.sh file modify the YCSB_HOME path to-
   YCSB_HOME=/home/hduser/ycsb-0.11.0

3. Ensure the output directory exist under YCSB_HOME, if not create a directory
   cd /home/hduser/ycsb-0.11.0
   mkdir output

4. For MongoDB, in the file usertableClear.js use lowercase ycsb.

Opcounts-
For this test we will be using following wordcounts-
200000, 400000, 600000,800000

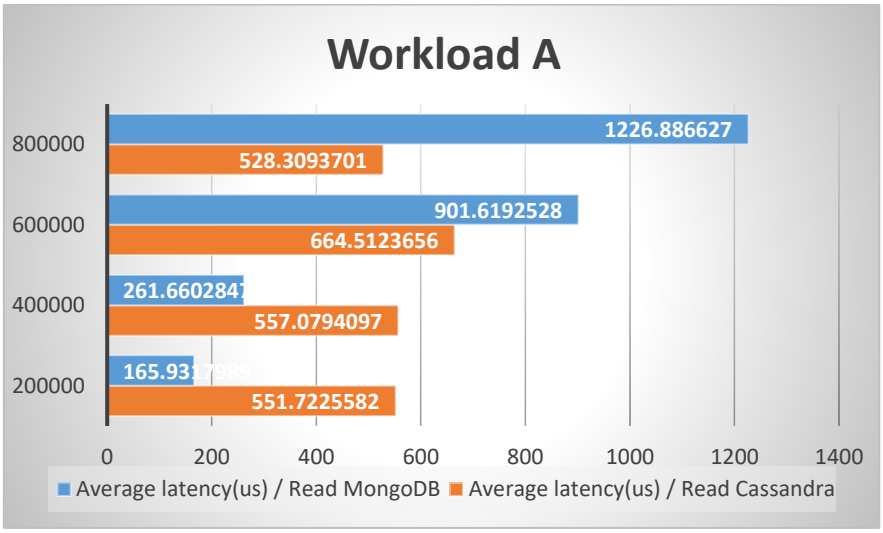
Workloads-
We will use workloada and workloadf. Workloada is having 50% read operations and 50% update operations where workloadf is having 50% read operations and 50% read/write/modify operations.

TestDbs-
Modify the file testdbs.txt with the databases on which we want to run the test. In this project we will add mongodb and cassandra2-cql databases.

VI. Evaluation and Results

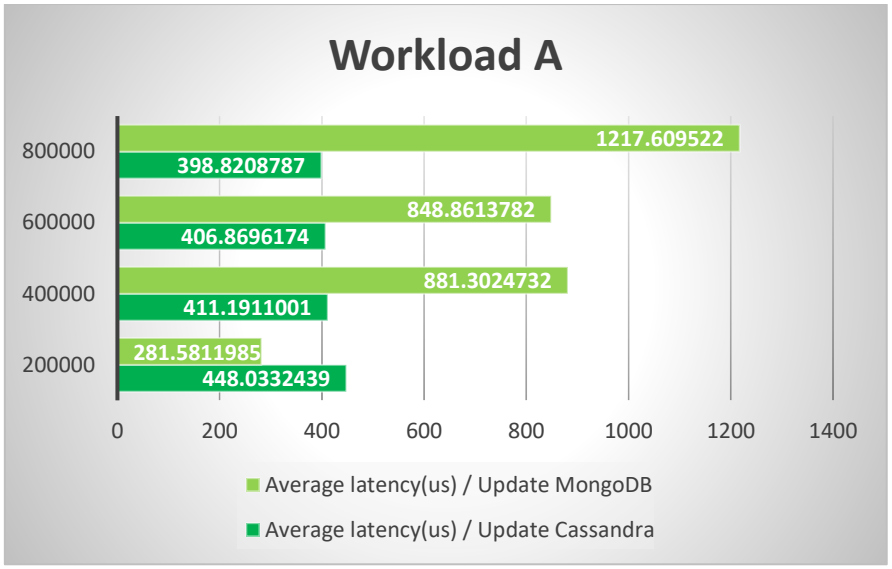
We will compare the results of databases on basis of average latency for various operations.

1. Workload A- Average latency(us) for read operations



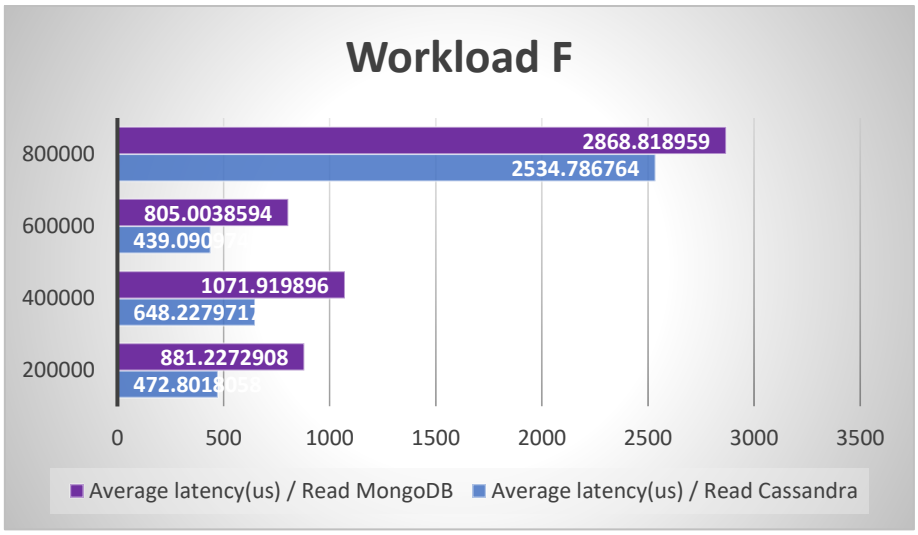
For the opcount 200000 MongoDB performs well than the Cassandra. But for the large opcount 800000 Cassandra performs so better than the MongoDB. MongoDB's aveage latency increases as we increase the opcount but in Cassandra the average latency is almost same for all the opcounts.

2. Workload F- Average latency(us) for update operations



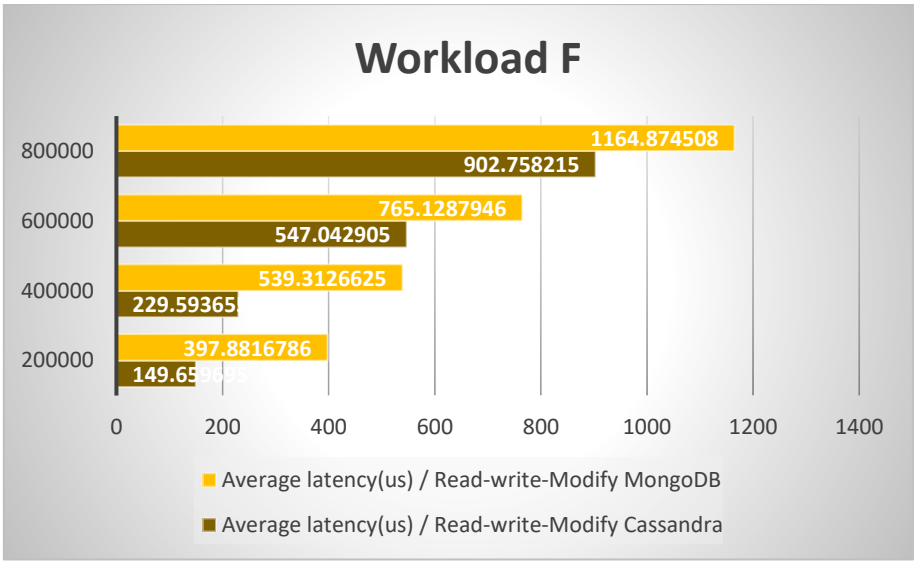
For the update operations, like read operations MongoDB performs well in case of small opcount 200000. But the average latency increase as the opcount is increased. Cassandra's latency time is quite similar in all workloads.

3. Workload F- Average latency(us) for read operations



In workload F the read operation count is more than the workload A read operations. In this test also, MongoDB performs well than the Cassandra if the opcount is small. For the 8 million opcount Cassandra is better than MongoDB.

4. Workload F- Average latency(us) for Read/write/modify operations.



In Workload F 50% operations are read/write/Modify operations. In this combination of operations Cassandra is better in all the opcount test compared to MongoDB.

VII. Conclusion

We compared the architecture, components, security and key characteristics of MongoDB and Cassandra. Both the systems are easy to use and configure. We have run the testharness tool on YCSB to compare the average latency for different operations. If the domain needs good data model, MongoDB has an expressive data model. For high availability Cassandra is the DB system to choose as it provides no downtime on failure of a node. For higher opcounts Cassandra performed better in each operation than the MongoDB so for scalability and where large data needs to be maintained Cassandra will be a good choice.

References

[1] https://docs.mongodb.com/manual/introduction/

[2] https://academy.datastax.com/resources/brief-introduction-apache-cassandra

[3] https://www.javatpoint.com/cassandra-features

[4] https://www.tutorialspoint.com/cassandra/cassandra_introduction.htm

[5] A Critical Comparison of NOSQL Databases in the Context of Acid and Base

[6] http://cassandra.apache.org/doc/latest/operating/security.html

[7] https://www.mongodb.com/scale/nosql-database-security

[8] http://www.cs.rochester.edu/courses/261/fall2017/termpaper/submissions/07/Paper.pdf