CS 488P/588P

Yufei Tao, Mi Yon Kim

Term Project Part 1 Report

In this term project, after we investigated among multiple database systems, we decided to research MongoDB. The MongoDB is an open source horizontal scale-out architecture. The mongo database uses NoSQL design so it gives some alternatives other than the traditional relational databases. MongoDB does not restrict the type of the document that is stored in the system. The structure of MongoDB is different from the traditional database. Mongo uses collections instead of tables, and each collection is made of documents instead of rows. They also call their “columns” fields and “table joins” as embedded documents. One of the key differences also include how these two databases handle primary keys. In MongoDB, the default key _id provided by MongoDB itself.

Compared to the traditional relational databases, MongoDB has many advantages. The first is how flexible the structure is. MongoDB does not require a specific schema. An example of disadvantage of the traditional database is if some fields in the rows of the traditional relational database were optional, which happens a lot of time. The more data in that database, the more potential wasted empty cells would it have. It’s really not space efficient. In MongoDB, programmers can store different kinds and different numbers of information in each document because there’s no schema restriction. This is a great way to solve fixed numbers of columns or multiple data in one cell. For example, if we are trying to store customers’ information, and if some customers have more than one addresses or phone numbers. In MongoDB, we can put those addresses into one field instead of creating a new table or more columns in traditional relational databases. Another advantage of MongoDB is the Document oriented storage. It uses JSON style documents, which allows more data types. This is the reason why we can store different kinds of data into each field.

The data model we chose for this project is the freeway information. The data have different sets depending on the length of the time and other various standards. Among all those datasets. They all have 7 attributes for each data. The id of the detector, the start time of the 20 seconds interval when the detector was running. Volume represents how many cars passed though. The occupancy, status and the data quality of the detector. An example data of the loop data looks like this: 1345,2011-09-18 14:08:40-07,7,83,6,2,0. It means the detector 1345 started detecting at 2011-09-18 14:08:40-07 for 20 seconds. During that time, there were 7 cars passed through with an average speed of 83 mph, 6 occupancy and the quality of the data is inhibited. If we insert a new data into this system, we could use the mongo command:

```
db.products.insert(
      {detectorid: 20201,
      starttime: 2011-09-01 04:39:40-07,
      volume: 50,
      speed: 65,
      occupancy: 4,
      status: 2,
      dqflags: 0 }
)
```

There are also some other datasets like freeway information. They all have really similar structure. MongoDB lets us have a really easy time inserting new data. It also has many other built-in functions to help us manipulate data.

The UI we used to analyze the freeway data is MongoDB Compass. Some examples of interacting with the database using mongo. For example when we try to find the data that matches certain conditions. We used : `db.Freeway.find({status : "0"}).limit(20)`. This lets us find the data that their status is 0, and we are only showing the first 20 of them.

Another example of how we interacted with the database is we could see the schema from the mongodb client. It analyzes our dataset and shows us the distribution of different kinds of data such as how many data that are collected by this specific detector with graphs inside of the client.

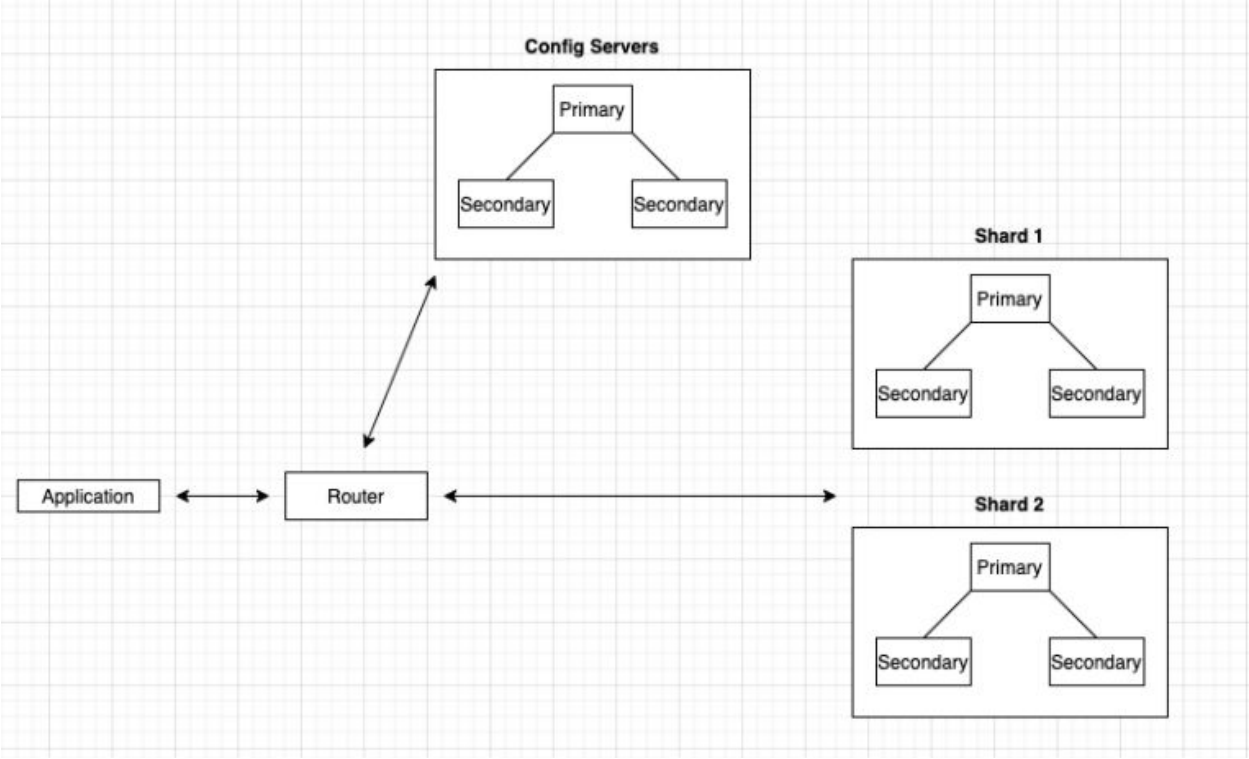
MongoDB automatically creates a unique index on the _id field which is the primary index during the creation of the collection. In addition, MongoDB supports secondary indices to execute the query more efficiently. For example, we are able to create an index on the detectorid field to query by the detectorid. We are able to create indices through createIndex() method.

```
db.collection.createIndex( <key and index type specification>, <options> )
```

MongoDB has Write Concern and Read Concern levels to achieve the tunable consistency. Write Concern can be specified either as a numeric value or a “majority”. When we choose the {w: numeric value(N)} option, the server will send the acknowledgement to the client after the data has been written to a specific number(N) of nodes of the replica set. In comparison, when we choose the {w:”majority”} option, the server will send the acknowledgement to the client after the data has been written to the majority of the server. Read Concern has available, linearizable, local, majority, and snapshot options. When we choose “local”, the client will receive the data from the local server. This will give the client the newest data from the server. In comparison, when we choose the “majority” level, the client will receive the data after the majority nodes of replica sets are committed with that data. With this option, the client can receive the stale data. MongoDB has a snapshot level of Read Concern which provides transactions with snapshot isolation guarantees. This option provides durability guarantee that any data read or written is deferred until transaction commit time.

MongoDB supports sharding and replication to support deployments with very large data sets and increase the availability of data sets. We must specify the shard key to shard the

collections, using sh.shardcollection() operation. At this point, we can think of the database system architecture that Amee(TA) suggested in the MongoDB_GCP_Manual document. However, we can tune the system while implementing and analyzing the datasets.



This is from the MongoDB_GCP_Manual document.