An Efficient Aviation Model for Air Transport Regulation using Nosql

Lydia J Ganansigamani, Cynthia Jayapal, H. Parveen Sultana, Monisha K

Abstract—The rapid growth of data in all fields makes NoSQL database spectacular in the recent times. NoSQL gives the efficient storing of the huge volume of information in a data warehouse. The NoSQL server provides horizontal scalability and retrieval of data in the very prominent way. These features make the NoSQL database meaningfully increase in the real-time applications relating to the web services. Handling of extensive data in different areas makes sense of NoSQL. This work particularly uses the aviation data frame describing the operations being taking part in the aircraft systems such as productions and developments. The stable growth in aviation department leads to the global development. Hence risk is analyzed in all aspects. The branches of aviation are public aviation, general aviation, and defense aviation keeps generating extensive records from the various third parties. These files which are in Gigabytes (GB) and Terabytes (TB) are stored in the databases for many practical applications. The data generated in various aviation departments are handled using document store NoSQL. The aviation framework using document store enables the accessing and storing information in the NoSQL database server efficiently with recovering the deleted data from the MongoDB data file structure. This work focuses on using aggregation operation to retrieve the data in an efficient manner and Mongo index B-tree algorithm where insertion, deletion, sequential access, and searches are done in logarithmic time.

Keywords: NoSQL, MongoDB, B-tree algorithm.

I. INTRODUCTION

In general, the relational model based databases are used most commonly today. Due to the rapid growth of data being collected every day, NoSQL (Not Only SQL) has become prominent in recent times(Kaur & Sharma, 2015). These rapid data are not structured generally, doesn’t fit in the relational model. There are more than 120 NoSQL database technologies produced in the latest years. NoSQL databases scale horizontally very well, they have no schema and span multiple nodes. NoSQL databases have been segregated to a different category based on their data model. The data store performance and scalability are influenced by choosing a proper data model. Let us have a brief look at several data models like column-family store, Key-value store, document store and graph databases(Atzeni, Bugiotti, Cabibbo, & Torlone, 2016). In the paper value store, a unique value is identified for each collection in the documents. Distributed data can be efficiently stored in this model, and hence it is scalable. The column store database stores the data in the form of a row and column matrix. Column data store becomes an important factor in analyzing the query performance as a column store reduces the overall disk I/O specification. In Graph database stores the information in the form of nodes and edges.

The data of an object is stored in the node, and the relationship among the nodes is denoted by edges. These edges and nodes also have the properties to describe the graph store functionalities. The key-value store is a data structure is an efficient and robust storage methodology where the values are stored in the form of keys with indexes for retrieving the data efficiently. This kind of association of data with the keys is known as Dictionary or Hashing(Nayak, Poriya, & Poojary, 2013). Many applications use NoSQL as a storage platform, and one among the application is Aviation system. In air transport system massive records are maintained every day. Hence, for such requests NoSQL storage system plays a significant role in sustaining and accessing the files frequently(Chiu & Hsieh, 2016). The aviation is classified based on the purposes and plotted as aviation of general, defence aviation, public aviation. The information is generated each in every department regarding the statistics of the aircraft begin also designed the records made from various third party sources are about the airport details, flight departure, and arrival and giving the trip statistics.

These details are available every month and as a year, where these characteristics can be used for particular needs of retrieving any information about flight statics. Now, the Civil Aviation Authority (CAA) verifies and validates the data for its results and checks for the integrity, reliability, and scalability. But, however, these constraints becomes an issue in aviation as the data becomes misused, wrongly updated, overlapping of data, and sometimes handling of the large data becomes a significant complication(Liang, Lin, Hwang, Wang, & Patterson, 2010). Through this problem, the users, employers, authority are led improperly, and they are updated with improper datasets of flight. This can is overcome with the efficient data management database namely NoSQL.

II. RELATED WORK

In general, the aviation data frames describe the operations took part in the aircraft systems (Insua, Alfaro, Gomez, Hernandez-coronado, & Bernal, 2016).
technique "coherent, integrated structure" is used for allocating the resources with the benefit of accessing the information in the scattered regions. In retrieving the statistics from databases, "stoicism and suppositious" methodology is applied for accuracy and availability. These databases sometimes when interacts with humans, errors can be interpreted which delicately reduces the production of the aviation department. Errors by people in aviation database systems are analyzed with the error analysis methodologies namely, "Human Factor Analysis and Classification System (HFACS) and Cause Analysis (RCA)" (Chiu & Hsieh, 2016). Fuzzy logics are defined with error factor criteria in the analysis and maintenance of the databases for examining the physical states, limitations, coordination and communication. The advantage of this technique is that whenever any operations are interpreted in the databases, then its performance is analyzed using the HFACS and RCA techniques for finding the error rate and limitations. The human error analysis process is developed from historical data for maintenance of aircraft system (MAS) which generates the details of the checklist of the airplane system which contains the regular updates of the plane system. Now, these statistics are handled by the human authorities with the database interference, and these updates are analyzed by applying the HFACS and RCA techniques. Commonly misconceptions are reduced in the aviation departments by using the existing methodologies.

Since our work has an association with the aggregate functions, let us have a brief overview of several related functions in MongoDB (Damaiyanti, Imawan, Indikawati, Choi, & Kwon, 2017). Titus Irma et al. approach the issue in the road traffic big data application by a practical method "SigTrac". This methodology operates on the top of the present MongoDB document based store. The traffic model is designed from the statistic points that are generated from the traffic data sensors. This technique exposes the map functions of MongoDB. SigTrac system constitutes road traffic sensor statistics that has temporal and spatial features as traffic measures. The SigTrac also adopts "Singular Valued Decomposition (SVD) and SVD incremental" algorithms for accessing and solving the user requests. The requests by the users involve the cognition of traffic status, which is a data-driven approach. In general, relational model based databases are used most commonly today. Due to the rapid growth of data being collected every day, NoSQL (Not OnlySQL) has become prominent in the recent times. These early data is not structured, doesn’t fit in the relational model. There are more than 120 NoSQL database technologies created in the past years. NoSQL databases are horizontally scalable, schema-free and distributed. Based on the data models the NoSQL databases are categorized. The data store performance and scalability are influenced by choosing a proper data model. Let us have a brief look at several data models like Key-value store, column-family store, document store and graph databases. In a key-value store, a unique value was identified. Distributed data can be efficiently stored in this model, and hence it is scalable. For relations and structure demanding applications, this model is not suitable. Some of the systems using this model are a big table (Chang et al., 2006), CouchDB, DynamoDB, MemacheDB, and Redis. In Column-family store, the data is composed of many rows. Each row is indexed by unique primary key. It is similar to the key-value store by implementing the logic requiring relations in the client applications. The data are opaque to the database in the key-value store while in the document store model, the data are kept within an internal structure. A higher level of granularity is created by metadata, and a modern programming technique is delivered. A key is used in the document-store model to locate the document within the database. BSON (Binary JSON) or JavaScript Object Notation (JSON) are the most widely used document stores.

The document-store model is well suited for the applications containing text format. Each document should have a unique key in a collection. Similar to key-value stores, the record store is inefficient in multiple-key transactions which involve cross-document operations. The graph database model is to constitute connected data and highly complex structures. The nodes and edges in the chart databases have individual properties comprising of key-value pairs. Our work is mainly concerned with the document store model. The primary purpose of data migration from MySQL to MongoDB is to maintain all data in the old system. The authors (Truică, Boicea, & Trifan, 2013) have examined the key features of MongoDB and focused on basic operations of CRUD. They have compared the MySQL and MongoDB by data creation, selection, insertion, and deletion. Later the authors (Parker, Poe, & Vrbisky, 2013) examined the NoSQL solutions with the traditional database. The performance of the databases was compared regarding runtime when aggregate functions are used. Later the authors (Simanjuntak, Simanjuntak, Situmorang, & Saragih, n.d.) compared the query response time with traditional SQL database and NoSQL MongoDB using statistical methods. The authors (Aboutorabi, Rezapour, Moradi, & Ghadiri, 2015) compared the databases regarding collective aggregated and non-aggregate query performance. This performance analysis was done to the typical e-commerce schema. The question of accessing arbitrary MongoDB documents with SPARQL was discussed by (Michel, Zucker, & Montagnat, 2016). The authors proposed a two-step method for translating SPARQL query to pivot abstract query and then to MongoDB query. In a similar way, the authors (Dede, Govindaraju, Gunter, Canon, & Ramakrishnan, 2013) evaluated the performance, fault-tolerance, and scalability of using MongoDB with Hadoop for scientific data analysis.

III. METHODOLOGY DESCRIPTION & RESULTS

In this paper, MongoDB document store method is used for the aviation datasets. The data sets are manipulated and maintained successfully using the techniques i) Aggregation ii) Indexing and iii) CRUD operations.

3.1 Dataset Overview

The dataset explored was a domestic flights dataset. This model--mainly uses the aviation dataset as the fields are apt
for using the MongoDB aggregation framework.

### Table 1 Aviation Dataset - Overview

The reasons concerning the aviation datasets are mentioned below

1. Find the period of time where the arrival or departure delay can be reduced.
2. Find the different categories of the aircrafts that suffer delays and what are the ages of these aircrafts?
3. Find out how many times a flight delay causes delay to a connecting flight?
4. Find out the effect of a natural disaster on air traffic and also how quickly did the traffic became normal.

The solutions to the mentioned questions (Alabi and Song, 2013,) are given by the collection operations and indexing the obtained aggregated result. These results are furthermore depicted through graph representation for comparing and verifying the best result. This work is also more investigated to find how the flights get canceled? Why the flights get delayed? Which seasons are mostly affected by delays and cancellations? The results obtained from the aggregate queries are used formulated for finding the busiest and worst cities and states.

#### 3.2 Aggregation

Initially, when the collections are imported the CRUD operations are performed. The CRUD operations are inserting, retrieval, updating, and deleting the data is shown. After doing CRUD operations to make even better processing and manipulation of the data, aggregation functions are performed. Collections activities are “project, unwind, sum, match, sort, and group.” The aggregate () method accepts an array of stages as its argument, where each stage is processed sequentially and they describe a data processing step.

##### 3.2.1 Grouping the Documents by Fields and Calculating the Count

The $group stage is used for grouping the records with the specified key value. In this juncture, the fields are arranged with key_id by defining the field path. The function groups the fields in the aviation collection by the State_Origin, City_Origin fields, and $Sum is used for counting the records in each cluster.

##### 3.2.2 Filter and Group Records in each Collection

The $Match groups the similar records by specifying the id field as a key value. The pipeline uses the $match to retrieve the flight's groups for documents with cancelled equal to 0, month equal to 5, 3 and year similar to 2014. The $group stage forms groups with the matching records by State_Origin, City_Origin, Delay_arrival, Delay_Departure and uses the $sum accumulator to calculate the count for teams retrieving the fields by the field path with name field prefixed by the dollar sign $.

##### 3.2.3 The most delayed flights

The delays are found using the arrival delays and departure delays as it matters a lot to the traveller. The following query is used to retrieve the delays where the field can be replaced by the hour, week, month, or any grouping.

```javascript
db.aviation.aggregate({
    "$group": {
        "_id": ,
        "ArrivalDelay": { "$avg": "$Delay_arrival" }
    },
    "$sort": {
        "delays": -1
    }
});
```

![Figure 1 Flight delays for each flight](image1.png)
An Efficient Aviation Model for Air Transport Regulation using NoSQL

The time with least number of arrival delays would be the best time for travel. Which is

Hour, Delay_arrival in Minutes
4, -0.5765124555160143

3.2.4 Years in Service Affecting the Performance

The following query is used for finding if the years of service affect the arrival times:

```
db.aviation.find({"yearsinservice":{"$exists":true},
"_id":1,"Delay_arrival":1,"Delay_Departure":1,
"Brand_Carrier":1,"City_Origin":1,
"City_Destination":1,"yearsinservice":1}).sort(
"Delay_arrival":-1,"Delay_Departure":-1),limit(7);
```

3.2.5 Number of Flights Cancelled

The below aggregation returns the cumulative number of cancellation of flights due to natural disturbances such as bad weather, hurricane or snowfall in the city of New York.

```
db.aviation.aggregate({"$match":{
"Code_Cancellation":2,"Month_Cancellation":10,
"State_Origin_ID":"NY","State_Destination_ID":"NY"}},
{"$group":{"_id":"$Day_Month_Cancellation","sum":{"$sum":1}}},
{"$sort":{"_id":1}});
```

3.2.6 Busiest State

The given query finds the top ten states with the most heavy air traffic both inward and outward:

```
db.aviation.aggregate({"$group":{"_id":"$State_Origin","fly":{"$sum":1}}},
{"$sort":{"fly":-1}},
{"$limit":10});
```

```
Figure 2 The flight delays are displayed with craft span
```

```
Figure 3 Number of Flights cancelled in months
```

```
In this “arrDelay and depDelay, {arrDelay :-1, depDelay :-1}” we use the compound index of the attribute.
```

Retrieval Number: K104009811S19/2019©BEIESP
DOI: 10.35940/ijitee.K1040.09811S19
Published By: Blue Eyes Intelligence Engineering & Sciences Publication

194
3.2.7 Worst States

The below query lists the states with the worst average delays for aircrafts:

```
db.aviation.aggregate([{
  "$group": {
    "_id": "$State_Origin_ID",
    "flight_delay": {"$avg": "$Delay_Departure"}
  }
}, {
  "$sort": {
    "flight_delay": -1
  }
}, {
  "$limit": 10
}]);
```

Figure 4 Busiest states in Flight transportation

45.83% 18.28% 13.98% 31.18% 12.9% 3.23%

Figure 5 Worst delay in Flight transportation

IV. INDEXING

Indexes assess amid MS SQL Server and MongoDB. A databank index is a data structure that advances the speed of data recovery actions. Indexes are essential for excellent presentation read operations for often used queries. Without indexes, databank table/collection will be scanned thoroughly to choice required data that match the query statement. Practice Apache H Base when the real-time read/write access. Just as Big table influences the distributed data storage delivered by the Google File System, Apache H Base provides Big table-like abilities on top of Hadoop and HDFS. …A hyper table is a great act, open source, immensely scalable database exhibited after Big table, Google’s brand, massively scalable database. The accumulator is built on Google’s Big Table project and is constructed on top of Apache Hadoop, Zookeeper, and Thrift. The importance of indexing in MongoDB was experienced first-hand during the execution of the queries. It was clear that some of the data needed indexing as the query execution became slower and slower as the complexity of the query increased. Indexes were especially needed for Delay_arrival, Delay_Departure.

Each index took about 2 to 3 minutes to create. They ranged from 120MB to 200MB in size. MongoDB is a worthwhile trade-off since all our queries run much faster after creating the indexes.

Explaining the above given statement a compound index is created on the fields, Delay_arrival and Delay_Departure ((Delay_arrival : -1, Delay_Departure : -1)). Also, if the index is created an index on depDelay only but not on Delay_arrival because the prefixes of the compound index can satisfy queries on any prefix of its fields. Based on the machine and the usage
of relevant indices the query performance time varied significantly. If a query is written to find the flights that start on time but arrive delayed or late would take about 20 minutes to on precise. Creation of that compound index takes about 20 minutes and 15 seconds. But with the index in place the query execution takes only 5 seconds, which is a very big improvement for all the future queries.

db.aviation.aggregate( {
  "match": { "$or": [ {
    $and": [ {
      "$gt": 0,
      "$lte": 0
    }]
  }]
}, {
  "$group": { 
    "$id": 1,
    "$State_Origin_Id": 1,
    "$TotalNumberofFlights": 1,
    "$sum": 1,
    "$limit": 5
  }
});

V. CONCLUSION AND FUTURE WORK

This paper shows the use of NOSQL databases in real-time applications. The use of document store NoSQL marks the obtainability of data over different NoSQL servers. The method also uses the mongo index B-tree algorithm which has logarithmic time complexity for searching, insertion and deletion. It stores data in sorted order. B-trees are usually used in systems that access extensive blocks of data for a single read or write such as relational database systems. When using B-tree with MongoDB storing and accessing of the BSON data becomes more scalable and easy. The future aspect would be trying to create join statements in MongoDB between two collections and using B-Tree internally to make it more efficient. Also shortening the field names makes the storage more efficient but it affects the readability of the data. Hence we have use it with caution. Some of the other parameters like number of servers and the number of copies it maintains and where the server looks for the duplicate copies can be more fine-tuned to get an optimized performance.

VI. REFERENCES


