



College Dublin
Computing • IT • Business

CCT College Dublin Continuous Assessment

Group Report

Module Title:	Data Storage Solutions Commercial Solutions Design		
Assessment Type:	Group (3)		
Assessment Title:	CA: 1		
Lecturer Name:	Muhammad Iqbal David McQuaid		
Students Names:	Yuri Ribeiro	Xiaohui Weng	Leisly Pino
Students Numbers:	2020347	2020387	2020303
Assessment Due Date:	07 / May / 2023		
Date of Submission:	12 / May / 2023		
GitHub Depository	https://github.com/lepidu/CSD-DSS		
Google Docs	CSD&DSS.docx		

Declaration

By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.

Index

Dataset	3
Data Storage framework	4
Architecture of the data storage	8
Performance of NoSQL databases	9
Spring Application	11
Contribution	23
References	24

Dataset

As we know that Hadoop works with big data, and also based on the CA requirements we need to use a dataset which should not be older than 5 years. The dataset that was provided 2 years ago is about US air carriers' information, which includes date, time, origin, destination, airline, distance, and delay status of flights from 2016 to 2018. It is estimated that air transportation delays put a 4 billion dollar dent in the country's gross domestic product that year.

We opted to not use all the columns, so the data that will be displayed on our application is:

1. **ID:** BSon ObjectId
2. **Year:** 2016, 2017, 2018
3. **Month:** 1-12
4. **DayOfMonth:** 1-31
5. **DepTime:** Actual departure time (local, hhmm)
6. **CRSDepTime:** Scheduled departure time (local, hhmm)
7. **ArrTime:** Actual arrival time (local, hhmm)
8. **CRSArrTime:** Scheduled arrival time (local, hhmm)
9. **Origin:** Origin IATA airport code
10. **Dest:** Destination IATA airport code
11. **Distance:** In miles

(For more information click here: [Dataset link](#))

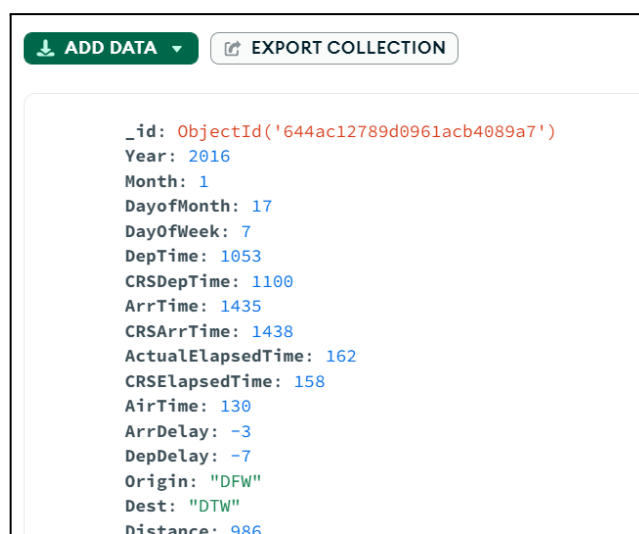


Figure 1. Variables in the Domestic Flight Database.

Data Storage framework

Data Storage Framework is a software which is responsible for computing the data in the system, in another word, it manages the data in a way to process or store the data. Hadoop is one of the most famous open source software for the big data. It provides some services to solve the big data problems, such as massive data storage by using HDFS, resource management, scheduling and allocation by using YARN etc..As far as our knowledge is concerned, that there are 3 important servers for HDFS, the purpose of that to provides storage for the massive amount of data.

- a. NameNode(master node) that maintain the whole file system
- b. Secondary nameNode is a file system Image, when NameNode privodes services for the cluster and sometimes may not provide sufficient resources, then Secondary nameNode will take this part of the job.

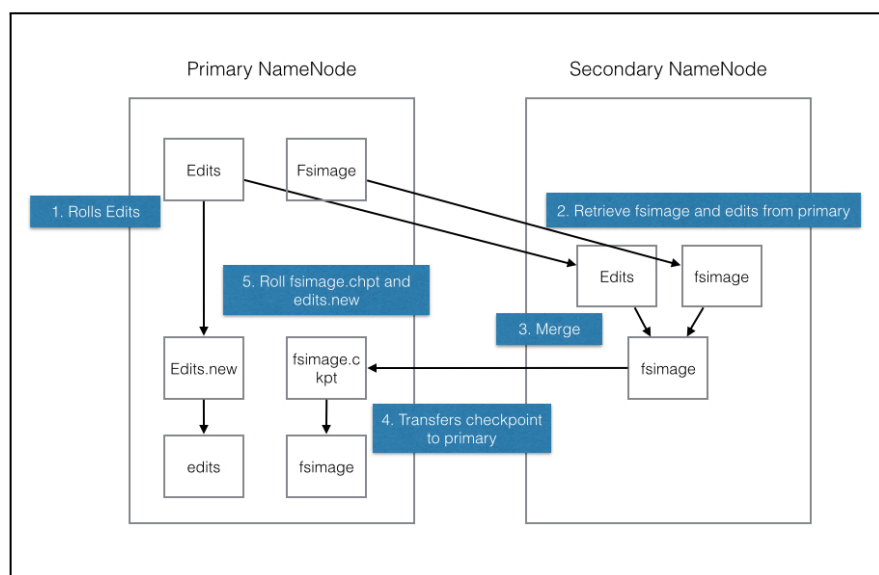


Figure 2. Primary and secondary node diagram.

- c. DataNode (Slave node) that when the client requests to write a file block, those blocks will be copied into multiple dataNodes, in general, the size of each block will be 128MB. And how many copies will be executed will depend on the client. As you can see figures down below that hadoop is running properly.

```
hduser@xiaohui-VirtualBox:~/Desktop/Hadoop$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [xiaohui-VirtualBox]
hduser@xiaohui-VirtualBox:~/Desktop/Hadoop$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hduser@xiaohui-VirtualBox:~/Desktop/Hadoop$
```

Figure 3. Initiation of Hadoop.

On the other hand, MapReducer provides the computing Framework to deal with the data. When MapReduce receives a task, firstly the input from HDFS will split the task into Map task in different nodes, then it will shuffle all those value and reducer them, then get the output.

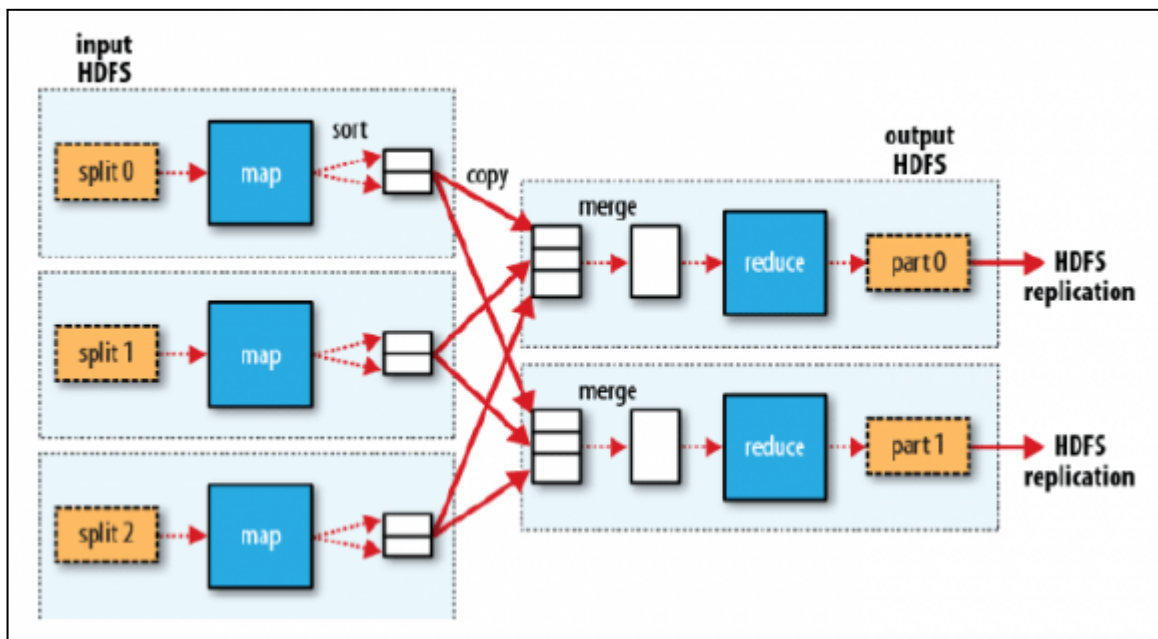


Figure 4. MapReduce Diagram.

As demonstrated, in the mapper.py and reducer.py code we get key-value from the mapper, and reducer uses the key-value pair to process the data.

```

GNU nano 6.2 mapper.py
#!/usr/bin/env python3

import sys

#Read each line from stdin
for line in sys.stdin:

    # Get the words in each line
    words = line.split()

    # Generate the count for each word
    for word in words:

        # Write the key-value pair to stdout to be processed by
        # the reducer.
        # They key is anything before the first tab character and the
        # value is anything after the first tab character.
        print('{0}\t{1}'.format(word,1))

```

Figure 5. Mapper code.

```

GNU nano 6.2 reducer.py
#!/usr/bin/env python3

import sys

curr_word = None
curr_count = 0

# Process each key-value pair from the mapper
for line in sys.stdin:

    # Get the key and value from the current line
    word, count = line.split('\t')

    # Convert the count to an int
    count = int(count)

    # If the current word is the same as the previous word,
    # Increment its count, otherwise print the words count to stdout
    if word == curr_word:
        curr_count += count
    else:
        # Write word and its number of occurrences as key-value
        # pair to stdout
        if curr_word:
            print('{0}\t{1}'.format(curr_word,curr_count))

        curr_word = word
        curr_count = count

# Output the count for the last word
if curr_word == word:
    print('{0}\t{1}'.format(curr_word, curr_count))

```

Figure 6. Reducer code.

As we can tell from the figures, the MapReducer works properly.

```
2023-05-09 22:43:14,773 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2023-05-09 22:43:14,968 ERROR streaming.StreamJob: Error Launching job : Output directory hdfs://localhost:9000/out1 already exists
Streaming Command Failed!
hduser@xiaohui-VirtualBox:~/Desktop/Hadoop$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.2.4.jar -file ./mapper.py -mapper ./mapper.py -file ./reducer.py -reducer
./reducer.py -input /user1/flights.csv -output /out
2023-05-09 22:43:36,702 WARN streaming.StreamJob: -ritte option is deprecated, please use generic option -rites instead.
packageJobJar: [./mapper.py, ./reducer.py] [/tmp/streamjob384334146386699440.jar tmpDir=null
2023-05-09 22:43:37,622 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2023-05-09 22:43:37,768 INFO impl.MetricsSystemImpl: Scheduled metric snapshot period at 10 second(s).
2023-05-09 22:43:37,768 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2023-05-09 22:43:37,800 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2023-05-09 22:43:38,225 INFO mapred.FileInputFormat: Total input files to process : 1
2023-05-09 22:43:38,356 INFO mapreduce.JobSubmitter: number of splits:1
2023-05-09 22:43:38,607 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local208287165_0001 we are streaming flights.csv file
2023-05-09 22:43:38,608 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-05-09 22:43:38,837 INFO mapred.LocalDistributedCacheManager: Localized file:/home/hduser/Desktop/Hadoop/mapper.py as file:/tmp/hadoop-hduser/mapred/local/job_local208287165_0001_
fe56fc6a-1c97-440c-b5b1-42571d7d4cc9/mapper.py
2023-05-09 22:43:38,841 INFO mapred.LocalDistributedCacheManager: Localized file:/home/hduser/Desktop/Hadoop/reducer.py as file:/tmp/hadoop-hduser/mapred/local/job_local208287165_0001_
962077ed-4e52-4aa5-a14d-8da2b887a9da/reducer.py
2023-05-09 22:43:38,969 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2023-05-09 22:43:38,971 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2023-05-09 22:43:38,971 INFO mapreduce.Job: Running job: job_local208287165_0001
2023-05-09 22:43:38,974 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2023-05-09 22:43:38,988 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-05-09 22:43:38,989 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2023-05-09 22:43:39,081 INFO mapred.LocalJobRunner: Waiting for map tasks
2023-05-09 22:43:39,084 INFO mapred.LocalJobRunner: Starting task: attempt_local208287165_0001_m_000000_0
2023-05-09 22:43:39,113 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-05-09 22:43:39,113 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2023-05-09 22:43:39,139 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]
2023-05-09 22:43:39,158 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/user1/flights.csv:0+5232902
2023-05-09 22:43:39,244 INFO mapred.MapTask: numReduceTasks: 1
```

Figure 7. Initiation of MapReducer.

```
HDFS: Number of large read operations=0
HDFS: Number of write operations=4
HDFS: Number of bytes read erasure-coded=0
Map-Reduce Framework
  Map input records=1048270
  Map output records=1045453
  Map output bytes=6272721
  Map output materialized bytes=8363633
  Input split bytes=91
  Combine input records=0
  Combine output records=0
  Reduce input groups=306
  Reduce shuffle bytes=8363633
  Reduce input records=1045453
  Reduce output records=306
  Spilled Records=2090906
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=19
  Total committed heap usage (bytes)=650117120
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=5232902
File Output Format Counters
  Bytes Written=2551
2023-05-09 22:43:44,032 INFO streaming.StreamJob: Output directory: /out
hduser@xiaohui-VirtualBox:~/Desktop/Hadoop$ hadoop fs -ls /
Found 5 items
-rw-r--r-- 1 hduser supergroup 1900385224 2023-04-26 14:54 /hduser
drwxr-xr-x - hduser supergroup 0 2023-05-09 22:43 /out
drwxr-xr-x - hduser supergroup 0 2023-04-26 15:24 /out1
drwx----- - hduser supergroup 0 2023-04-21 22:35 /tmp
drwxr-xr-x - hduser supergroup 0 2023-05-09 22:40 /user1
hduser@xiaohui-VirtualBox:~/Desktop/Hadoop$ hadoop fs -ls /out
Found 2 items
-rw-r--r-- 1 hduser supergroup 0 2023-05-09 22:43 /out/_SUCCESS
-rw-r--r-- 1 hduser supergroup 2551 2023-05-09 22:43 /out/part-00000
hduser@xiaohui-VirtualBox:~/Desktop/Hadoop$ jps
8888 ResourceManager
```

Figure 8. Results of MapReducer.

Architecture of the data storage

The structure of the database was designed by the Bureau of Transportation Statistics of the United States of America to keep track of the delays in the arrival and landing of internal flights within the period 2016-2018 and this data collection helped to estimate close to of a loss of almost 4 million dollars in the country's gross domestic product. The database has a collection Flights which contains date, time, origin, destination, airline, distance, and delay status of flights. This information was organised in a csv file that we were able to get from the Kaggle website. This data was imported into MongoDB which allowed us to query and review the data update that we were able to incorporate during this project.

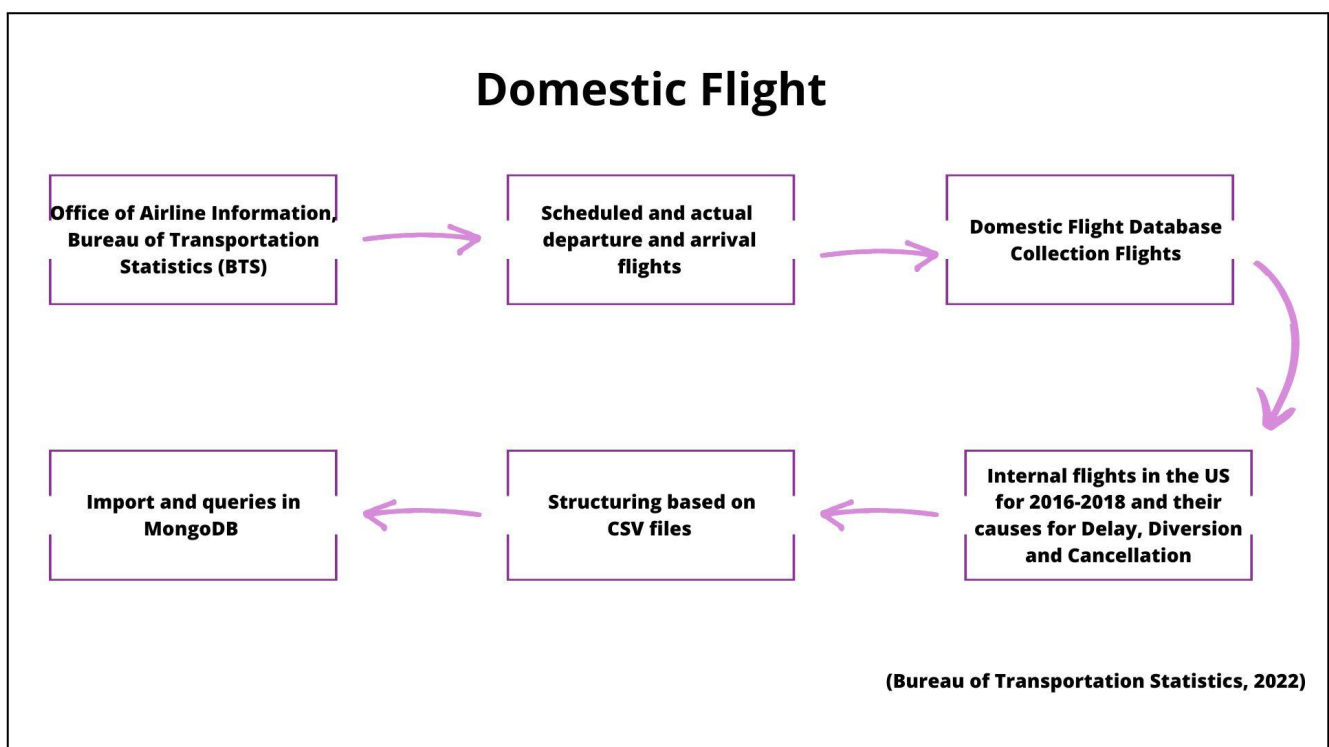


Figure 9. Data Storage Architecture Diagram.

The queries we made to verify that our CRUD program were as follows:

- Consult to find specific information:

```
db.flights.find({ year: 2023 })
```

- Search for a specific data set, this helped us verify data created:


```
db.flights.find({
  year: 2023,
  month: 5,
  dayOfMonth: { $gte: 1, $lte: 10 }
})
```

- Query data in ascending order:

```
db.flights.find().sort({ arrTime: 1 })
```

Performance of NoSQL databases

As we know, there are many types of NoSQL databases, such as MongoDB, HBase, Redis, Memcached etc.. and YCSB (Yahoo! Cloud Serving Benchmark) which is one of the good tools to test the performance of NoSQL databases. It provides RunTime, Throughput, MinLatency, MaxLatency and so on. The test has a lot of detail. In this case we have tested the performance of MongoDB.

```
INSERT-FAILED], 99thPercentileLatency(us), 39391
duser@xiaohui-VirtualBox:~/ycsb-0.17.0
duser@xiaohui-VirtualBox:~/ycsb-0.17.0$ ./bin/ycsb.sh load mongodb -s -P workloads/workloada
usr/bin/java -classpath /home/hduser/ycsb-0.17.0/conf:/home/hduser/ycsb-0.17.0/lib/HdrHistogram-2.1.4.jar:/home/hduser/ycsb-0.17.0/lib/h
race-core4-4.1.0-incubating.jar:/home/hduser/ycsb-0.17.0/lib/jackson-core-asl-1.9.4.jar:/home/hduser/ycsb-0.17.0/lib/jackson-mapper-asl-1.9.4.jar:/home/hduser/ycsb-0.17.0/lib/mysql-c
nector-java-8.0.32.jar:/home/hduser/ycsb-0.17.0/mongodb-binding/lib/logback-classic-1.1.2.jar:/home/hduser/ycsb-0.17.0/mongodb-binding/lib/logback-core-1.1.2.jar:/home/hduser/ycsb-0
17.0/mongodb-binding/lib/mongo-java-driver-3.8.0.jar:/home/hduser/ycsb-0.17.0/mongodb-binding/lib/mongodb-async-driver-2.0.1.jar:/home/hduser/ycsb-0.17.0/mongodb-binding/lib/mongodb-
inding-0.17.0.jar:/home/hduser/ycsb-0.17.0/mongodb-binding/lib/slf4j-api-1.7.25.jar:/home/hduser/ycsb-0.17.0/mongodb-binding/lib/snappy-java-1.1.7.1.jar site.ycsb.Client -load -db si
e.ycsb.db.MongoDbClient -s -P workloads/workloada
ommand line: -load -db site.ycsb.db.MongoDbClient -s -P workloads/workloada
CSB Client 0.17.0

loading workload...
Starting test.
023-05-09 10:28:37:894 0 sec: 0 operations; est completion in 0 second
Mongo client connection created with mongodb://localhost:27017/ycsb:w=1
Wrapper: report latency for each error is false and specific error codes to track for latency are: []
023-05-09 10:28:39:051 1 sec: 1000 operations; 790.51 current ops/sec; [CLEANUP: Count=1, Max=1239, Min=1239, Avg=1239, 90=1239, 99=1239, 99.9=1239, 99.99=1239] [INSERT: Count=1000,
Max=205567, Min=187, Avg=575.12, 90=488, 99=1115, 99.9=27503, 99.99=205567]
OVERALL], RunTime(ms), 1275
OVERALL], Throughput(ops/sec), 784.3137254901961
TOTAL_GC_PS_Scavenge], Count, 5
TOTAL_GC_TIME_PS_Scavenge], Time(ms), 17
TOTAL_GC_TIME_PS_Scavenge], Time(%), 1.3333333333333335
TOTAL_GC_PS_MarkSweep], Count, 0
TOTAL_GC_TIME_PS_MarkSweep], Time(ms), 0
TOTAL_GC_TIME_PS_MarkSweep], Time(%), 0.0
TOTAL_GCs], Count, 3
TOTAL_GC_TIME], Time(ms), 17
TOTAL_GC_TIME_%], Time(%), 1.3333333333333335
CLEANUP], Operations, 1
CLEANUP], AverageLatency(us), 1239.0
CLEANUP], MinLatency(us), 1239
CLEANUP], MaxLatency(us), 1239
CLEANUP], 95thPercentileLatency(us), 1239
CLEANUP], 99thPercentileLatency(us), 1239
INSERT], Operations, 1000
INSERT], AverageLatency(us), 575.121
INSERT], MinLatency(us), 187
INSERT], MaxLatency(us), 205567
INSERT], 95thPercentileLatency(us), 665
INSERT], 99thPercentileLatency(us), 1115
INSERT], Return=OK, 1000
```

Figure 10. YCSB Test.

YCSB workloads A is 50% read and 50% Update, we could tell from the chart, the Runtime is 1275ms, and theThroughput is 784.3137254901961. So throughput can reach about 78.4%. The result returned is correct, 1000. 95thPercentileLatency:665, and 99thPercentileLatency:1115.

```

root@-t -db site.ycsb.db.AsyncMongoDbClient -s -P wo
Command line: -t -db site.ycsb.db.AsyncMongoDbClient -s -P wo
Unable to open the properties file wo
wo (No such file or directory)

rootuser@xiaohui-VirtualBox:~/ycsb-0.17.0$ ls
accumulo1.6-binding      googledatastore-binding  output3.txt
accumulo1.7-binding      griddb-binding           output4.txt
accumulo1.8-binding      hbase098-binding        outputCA.txt
aerospike-binding       hbase10-binding         outputloadCA.txt
arangodb-binding         hbase12-binding         outputload.txt
asynchbase-binding      hbase14-binding         postgresql-binding
azurecosmos-binding     hbase20-binding         rados-binding
azuretablestorage-binding  hypertable-binding      redis-binding
bin                     ignite-binding           rest-binding
cassandra-binding       infinispan-binding       riak-binding
cloudspanner-binding    jdbc-binding            rocksdb-binding
couchbase2-binding      kudu-binding            s3-binding
couchbase-binding       lib                     server-6.0.asc

```

In mongoDB:

Spring Application

We have created a Web Application capable of connecting into a NoSQL (MongoDB) remote database and performing CRUD: CREATE – READ – UPDATE – DELETE methods.

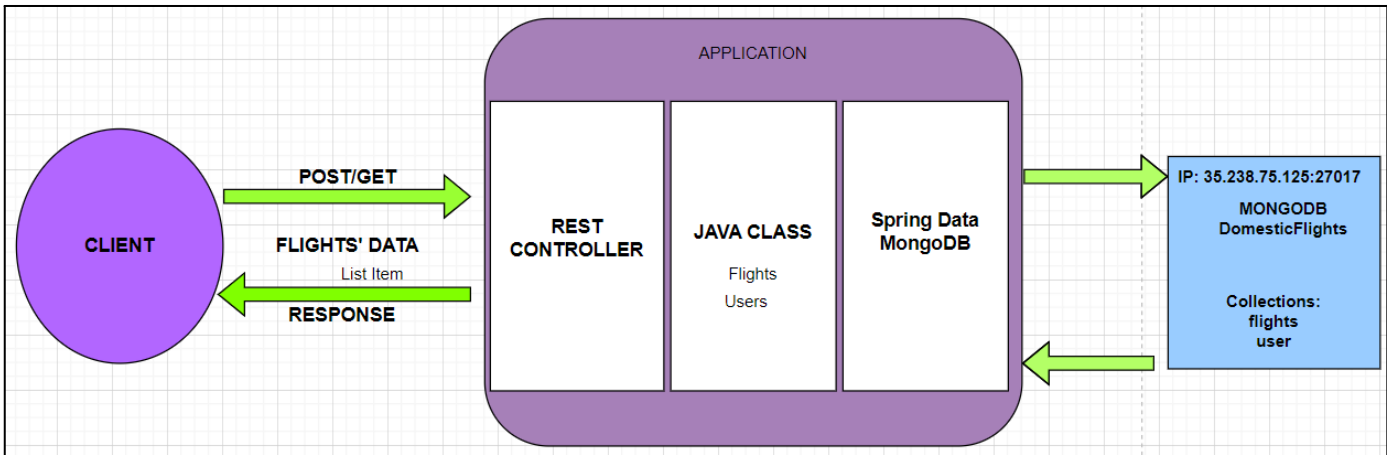


Figure 13. Spring application diagram.

In the diagram above we can find a brief explanation of how the spring application works in this project. A client can request data from a MongoDB based on Google Cloud passing by the REST controller that will provide the data required.

MongoDB

We opted to use a dedicated database instance on Google Cloud, thanks to the credits given by CCT to its students this semester. The mongoDB instance is a Linux machine running MongoDB standalone server 24/7. **The Instance will be UP for testing till June 14.**

INSTANCES

OBSERVABILITY

INSTANCE SCHEDULES

VM instances

Filter

Enter property name or value

<input type="checkbox"/>	Status	Name ↑	Zone	Connect		
<input type="checkbox"/>		dss-vm	us-central1-a	SSH	▼	<div>⋮</div>
<input type="checkbox"/>		mongodb	us-central1-a	SSH	▼	<div>⋮</div>

Figure 14. Linux Virtual Machine.

It was necessary to modify some firewall rules to allow access from anyone running the Web Application.

<input type="checkbox"/>	mongodb	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:27017	Allow	^
--------------------------	-------------------------	---------	--------------	-------------------------	-----------	-------	---

Figure 15. Linux Virtual Machine.

```
hduser@mongodb:~$ sudo systemctl status mongod.service
• mongod.service - MongoDB Database Server
  Loaded: loaded (/lib/systemd/system/mongod.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2023-05-01 20:00:03 UTC; 1 weeks 1 days ago
    Docs: https://docs.mongodb.org/manual
  Main PID: 38869 (mongod)
    Memory: 918.9M
    CGroup: /system.slice/mongod.service
            └─38869 /usr/bin/mongod --config /etc/mongod.conf

May 01 20:00:03 mongodb systemd[1]: Started MongoDB Database Server.
hduser@mongodb:~$
```

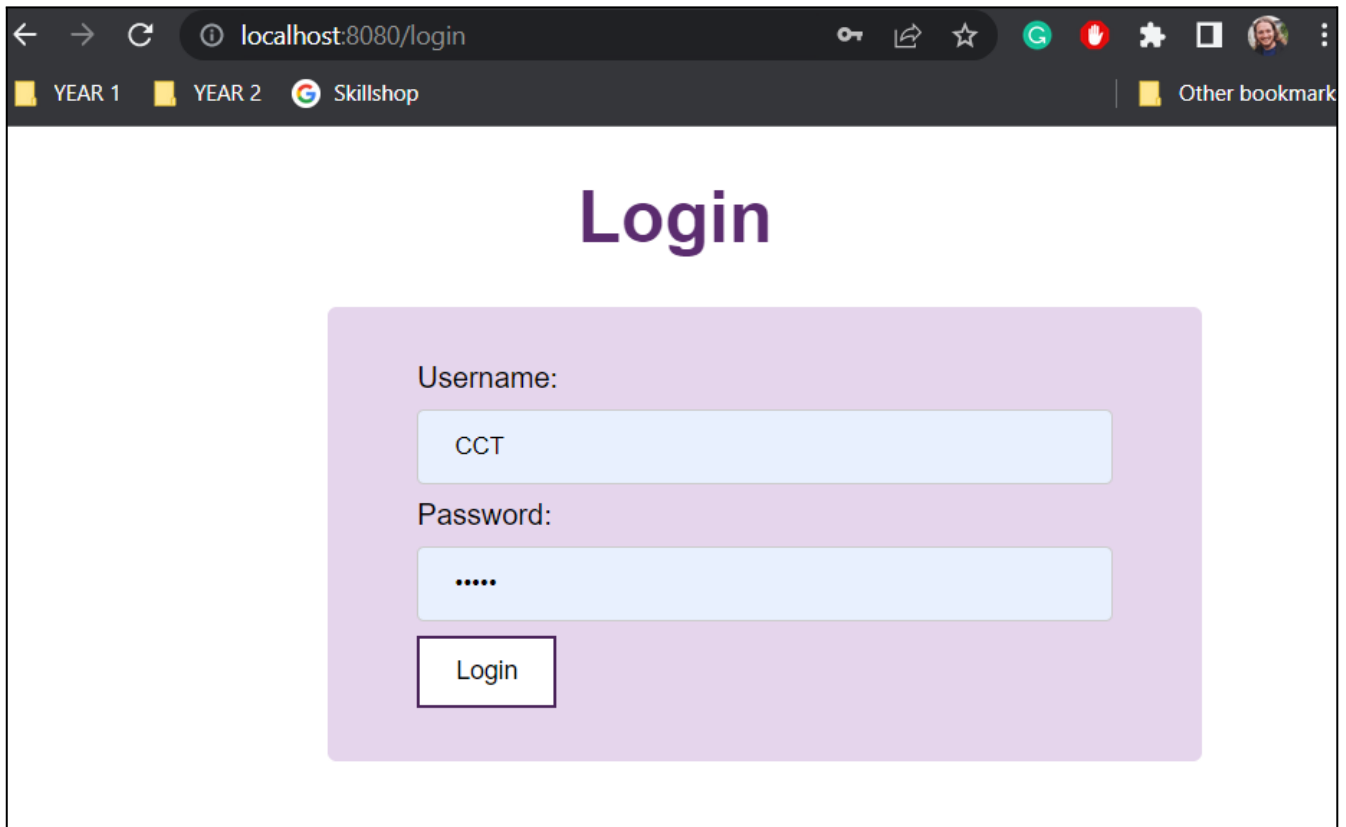
Figure 16. MongoDB Server StandAlone: (IP 35.238.75.125).

```
DomesticFlights> db.flights.find()
[
  {
    _id: ObjectId("644ac12789d0961acb4089a7"),
    Year: 2016,
    Month: 1,
    DayOfMonth: 17,
    DepTime: 1053,
    CRSDepTime: 1100,
    ArrTime: 1435,
    CRSArrTime: 1438,
    Origin: 'DFW',
    Dest: 'DTW',
    Distance: 986,
    _class: 'com.ca.IntegratedCA.model.Flights'
  },
  {
    _id: ObjectId("644ac12789d0961acb4089a8"),
    Year: 2016,
    Month: 1,
    DayOfMonth: 17,
    DepTime: 1055,
    CRSDepTime: 1100,
    ArrTime: 1436,
    CRSArrTime: 1438,
    Origin: 'DFW',
    Dest: 'DTW',
    Distance: 986,
    _class: 'com.ca.IntegratedCA.model.Flights'
  },
  {

```

Figure 17. Database DomesticFlights collection flights.

The REST controller is composed of Methods POST and GET which control the workflow of the data sent and received either by the client or the database able to control multiple queries at the same time. However, the application has a login area to ensure that only authorized access can pass by to its system.



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/login'. The browser's bookmark bar includes 'YEAR 1', 'YEAR 2', 'Skillshop', and 'Other bookmark'. The main content area features a large purple 'Login' heading. Below this heading is a light purple rectangular box containing a login form. The form consists of a 'Username:' label followed by a light blue text input field containing the text 'CCT'. Below the username field is a 'Password:' label followed by a light blue password input field filled with dots. At the bottom of the form is a white 'Login' button with a black border.

Figure 18. Login.html file.

Once the client access is granted it will be displayed in the index.html the first 10 pieces of information from the database Domestic Flights which contains over 16 million documents. As we can see in the picture below:

ID	Year	Month	Day of month	Departure Time	Scheduled Departure Time	Arrival Time	Scheduled Arrival Time	Origin	Destination	Distance	
644ac12789d0961acb4089a7	2016	1	17	1053.0	1100.0	1435.0	1438.0	DFW	DTW	986.0	Update Delete
644ac12789d0961acb4089a8	2016	1	18	1055.0	1100.0	1436.0	1438.0	DFW	DTW	986.0	Update Delete
644ac12789d0961acb4089ad	2016	1	23	1053.0	1100.0	1432.0	1438.0	DFW	DTW	986.0	Update Delete
644ac12789d0961acb4089ae	2016	1	24	1110.0	1100.0	1438.0	1438.0	DFW	DTW	986.0	Update Delete
644ac12789d0961acb4089af	2016	1	25	1059.0	1100.0	1424.0	1438.0	DFW	DTW	986.0	Update Delete
644ac12789d0961acb4089b1	2016	1	27	1101.0	1100.0	1437.0	1438.0	DFW	DTW	986.0	Update Delete
644ac12789d0961acb4089b2	2016	1	28	1100.0	1100.0	1453.0	1438.0	DFW	DTW	986.0	Update Delete

Figure 19. List of flights in index.html file.

On the right side, the client can Update the flight record as shown in the picture below:

ID	Year	Month	Day of month	Departure Time	Scheduled Departure Time	Arrival Time	Scheduled Arrival Time	Origin	Destination	Distance	
644ac12789d0961acb4089a7	2016	1	18	1053.0	1100.0	1435.0	1438.0	DFW	DTW	986.0	Update Delete
644ac12789d0961acb4089a8	2016	1	17	1055.0	1100.0	1436.0	1438.0	DFW	DTW	986.0	Update Delete
644ac12789d0961acb4089ad	2016	1	23	1053.0	1100.0	1432.0	1438.0	DFW	DTW	986.0	Update Delete
644ac12789d0961acb4089ae	2016	1	24	1110.0	1100.0	1438.0	1438.0	DFW	DTW	986.0	Update Delete
644ac12789d0961acb4089af	2016	1	25	1059.0	1100.0	1424.0	1438.0	DFW	DTW	986.0	Update Delete
644ac12789d0961acb4089b1	2016	1	27	1101.0	1100.0	1437.0	1438.0	DFW	DTW	986.0	Update Delete
644ac12789d0961acb4089b2	2016	1	28	1100.0	1100.0	1453.0	1438.0	DFW	DTW	986.0	Update Delete

Figure 20. Update and delete function.

Update Flight

Year: 2016

Month: 1

Day of Month: 17

Departure Time: 1053.0

Scheduled Departure Time: 1100.0

Arrival Time: 1435.0

Scheduled Arrival Time: 1438.0

Figure 21. Update.html file.

Domestic flights 2016 - 2018

The Dataset information of flights between 2016 and 2018

Flight 644ac12789d0961acb4089a7 has been updated.

ID	Year	Month	Day of month	Departure Time	Scheduled Departure Time	Arrival Time	Scheduled Arrival Time	Origin	Destination	Distance	
644ac12789d0961acb4089a7	2016	1	17	1053.0	1100.0	1435.0	1438.0	DFW	DTW	986.0	Update Delete

Figure 22. Notification update successfully.

Also, it is possible to add/create a new flight record just by clicking on the link at the top of the page “New Flight”:

localhost:8080/index/newFlight

YEAR 1 YEAR 2 Skillshop

Home New Flight Search Flight

New Flight

Year: 2023

Month: 5

Day of Month: 10

Departure Time: 1235

Scheduled Departure Time: 1230

Arrival Time: 1545

Scheduled Arrival Time: 1538

Figure 23. NewFlight.html file.

Returning the new ID created automatically by MongoDB :

Home New Flight Search Flight

Domestic flights 2016 - 2018

The Dataset information of flights between 2016 and 2018

Flight 645b834e057d8f074c8a4292 has been added.

Day of	Departure	Scheduled Departure	Arrival	Scheduled Arrival
--------	-----------	---------------------	---------	-------------------

Figure 24. New flight was added successfully.

Using the ID as a reference the client can search for any flight information by clicking at the top of the page link “Search Flight”:

localhost:8080/index/search

YEAR 1 YEAR 2 Skillshop

Home New Flight Search Flight

Search Flight By ID

ID: 645b834e057d8f074c8a4292

Search

Figure 25. Search flight form.

ID	Year	Month	Day of month	Departure Time	Scheduled Departure Time	Arrival Time	Scheduled Arrival Time	Origin	Destination	Distance	
645b834e057d8f074c8a4292	2023	5	10	1235.0	1230.0	1545.0	1538.0	DUB	AMS	350.0	Update Delete

Figure 26. Results of searching flight.

Or delete a flight record:

ID	Year	Month	Day of month	Departure Time	Scheduled Departure Time	Arrival Time	Scheduled Arrival Time	Origin	Destination	Distance	
645b834e057d8f074c8a4292	2023	5	10	1235.0	1230.0	1545.0	1538.0	DUB	AMS	350.0	Update Delete

Figure 27. Delete flight function.

Domestic flights 2016 - 2018	
The Dataset information of flights between 2016 and 2018	
Flight 645b834e057d8f074c8a4292 has been deleted.	

Figure 28. Notification was deleted successfully.

Here goes the core Java code the controller:

```
@Controller

public class FlightsController {

// LOGIN - MontoTemplate used for authentication
```

```
@Autowired
```

```
private MongoTemplate mongoTemplate;
```

```
// Add a new User to the model for authentication
```

```
@GetMapping({ "/login" })
```

```
public String loginForm(Model model) {
```

```
model.addAttribute("user", new User());
```

```
return "login";
```

```
}
```

```
// find the User in the database. If the user exists redirect to index page. Else, return to login page
```

```
@PostMapping("/login")
```

```
public String loginSubmit(@ModelAttribute User user, Model model, HttpSession session) {
```

```
Query query = new Query(
```

```
Criteria.where("username").is(user.getUsername()).and("password").is(user.getPassword());
```

```
User authenticatedUser = mongoTemplate.findOne(query, User.class);
```

```
if (authenticatedUser != null) {
```

```
session.setAttribute("user", authenticatedUser);
```

```
return "redirect:/index";
```

```
} else {
```

```
model.addAttribute("loginError", "Invalid username or password");
```

```
return "login";
```

```
}
```

```
}
```

```
// FLIGHT DATASET - Flights Repository
```

```
@Autowired
```

```
private FlightsRepository repository;
```

```

public FlightsController(FlightsRepository repository) {

this.repository = repository;

}

/* Controller method for displaying the index page.

* Get all flights from the dataset and add to a flight list

* Display only 10 flights per page at the time*/

@GetMapping({ "/index", "/" })

public String home(Model model, @RequestParam(defaultValue = "0") int page) {

int pageSize = 10; // The number of items per page

Pageable pageable = PageRequest.of(page, pageSize);

Page<Flights> flights = repository.findAll(pageable);

model.addAttribute("flights", flights.getContent());

model.addAttribute("currentPage", page);

model.addAttribute("totalPages", flights.getTotalPages());

return "index";

}

/*Controller method that display the form for Add a new Flight data in the database */

@GetMapping("/index/newFlight")

public String formFlight(Model model) {

Flights flights = new Flights();

model.addAttribute("flights", flights);

return "newFlight";

}

/*Save the new flights data to the database

* Returns an message confirming the flight has been created/added*/

```

```

@PostMapping("/index")

public String saveFlights(@ModelAttribute("flights") Flights flights, RedirectAttributes redirectAttributes) {

    repository.save(flights);

    redirectAttributes.addFlashAttribute("AddedMessage", "Flight " + flights.get_id() + " has been added.");

    return "redirect:/index";

}

/*Controller method for Updating the data using an ID as reference*/

@GetMapping("/index/update/{_id}")

public String updateAllFlightById(@PathVariable String _id, Model model) {

    model.addAttribute("flights", repository.findById(_id).orElse(null));

    return "update";

}

/*Retrieves the existing flight data from the database then update all the properties

* Returns an message confirming the flight has been updated*/

@PostMapping("/index/{_id}")

public String updateFlight(@PathVariable String _id, @ModelAttribute("flights") Flights flight, Model model,

RedirectAttributes redirectAttributes) {

    Flights existingFlight = repository.findById(_id).get();

    existingFlight.setYear(flight.getYear());

    existingFlight.setMonth(flight.getMonth());

    existingFlight.setDayofMonth(flight.getDayofMonth());

    existingFlight.setDepTime(flight.getDepTime());

    existingFlight.setCRSDepTime(flight.getCRSDepTime());

    existingFlight.setArrTime(flight.getArrTime());

    existingFlight.setCRSArrTime(flight.getCRSArrTime());

```

```

existingFlight.setOrigin(flight.getOrigin());

existingFlight.setDest(flight.getDest());

repository.save(existingFlight);

redirectAttributes.addFlashAttribute("updateMessage", "Flight " + _id + " has been updated.");

return "redirect:/index";
}

/*Gets the request and creates a new Flights object to the model returning the search method*/

@GetMapping("/index/search")

public String searchFlight(Model model) {

    Flights flights = new Flights();

    model.addAttribute("flights", flights);

    return "search";
}

/*Controller method for Searching the data using an ID as reference

* return an message if flight with the given ID was not found*/

@GetMapping("/search")

public String search(Model model, @RequestParam String _id, RedirectAttributes redirectAttributes) {

    Flights flight = repository.findById(_id).orElse(null);

    if (flight == null) {

        redirectAttributes.addFlashAttribute("searchMessage", "Flight " + _id + " cannot be found!");

        return "redirect:/index";

    }

    Flights flights = new Flights();

    model.addAttribute("flight", flight);

    model.addAttribute("flights", flights);

```

```
return "searchResults";

}

/* Method for deleting the flight data by ID*/

@GetMapping("/index/{_id}")

public String deleteFlightById(@PathVariable("_id") String _id, RedirectAttributes redirectAttributes) {

    repository.deleteByld(_id);

    redirectAttributes.addFlashAttribute("deleteMessage", "Flight " + _id + " has been deleted.");

    return "redirect:/index";

}

}
```

Contribution

At the beginning of our project, we had divided the tasks into 3 individual parts, however, after the first week of working on it we realized that the only way to make it was if we all worked together helping each other and as such the best way to have the work done. As you can see our first approach ended up delaying our delivery of the CA on time. Thus, we all worked together from the choice and deployment of the dataset through the utilisation of the processing environment to the development of the Web Application in Spring. The chart bar below represents the effort and time spent in the CA.

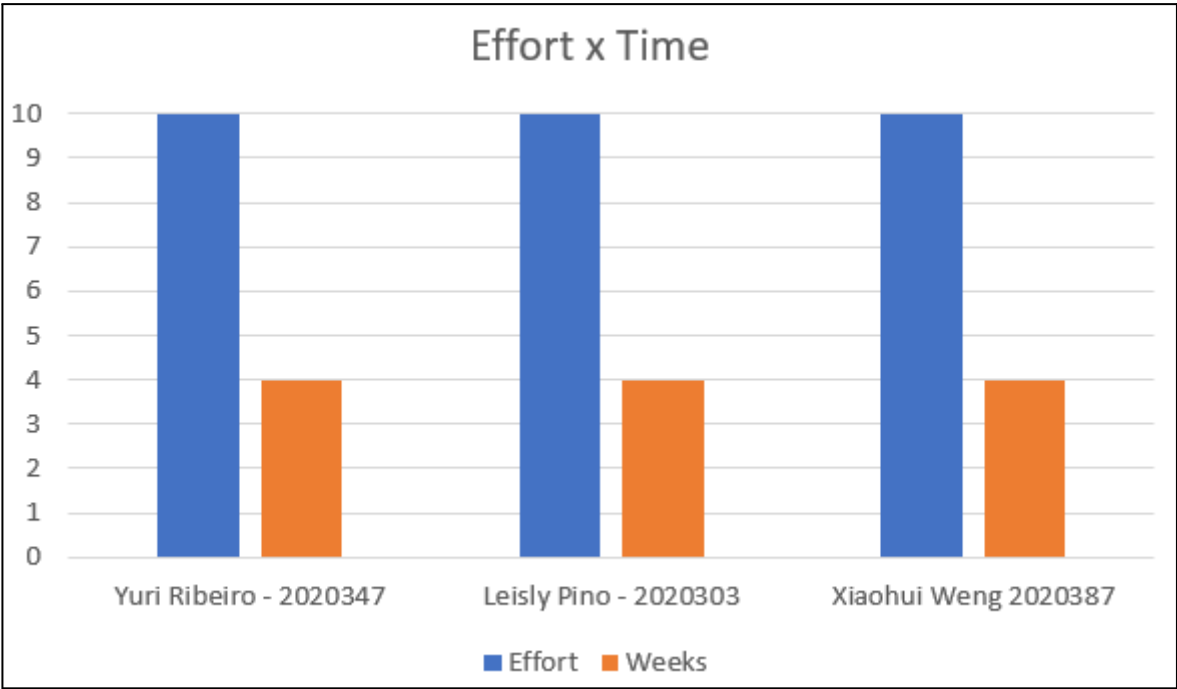


Figure 29. Team contribution diagram.

References

Bureau of Transportation Statistics, (2022). Domestic Flights 2016 - 2018 [online]. Kaggle: Your Machine Learning and Data Science Community. Available at: <https://www.kaggle.com/datasets/rulyjanuarfachmi/domesticusairflight2016-2018>

Spring Boot Thymeleaf CRUD example - BezKoder [online], (2023). BezKoder. Available at: <https://www.bezkoder.com/spring-boot-thymeleaf-example/>

Spring Data, Spring Boot, MongoDB (Example & Tutorial) [online], (2016). Tests4Geeks. Available at: <https://tests4geeks.com/blog/spring-data-boot-mongodb-example/>