BTS Aviation Dataset - Big Data Batch Processing

We use batch processing in Apache Hadoop to process and extract relevant information from an aviation dataset provided by the Bureau of Transportation Statistics (BTS). The dataset contains records of flights in the United States from 1988 to 2008, including flight origin and destination and departure and arrival delays. There are 116,754,606 observations in the aviation selection.

Data Extraction and Cleaning

The dataset is stored in an Elastic Block Store (EBS) snapshot (snap-e1608d88) on AWS in the us-east-1 availability zone. We extract the data by creating an EBS volume based on the snapshot. Using an EC2 instance and the AWS Command Line Interface we copy the snapshot to a S3 bucket. We then download the "airline_ontime" folder, which contains the relevant dataset, using the AWS CLI on the local machine.

The dataset consists of 240 tables, each representing a month from 1988 to 2008. We observe that 2002 is missing the data from May to December. We also observe that the files for October 2008 to December 2008 are corrupted, which are excluded.

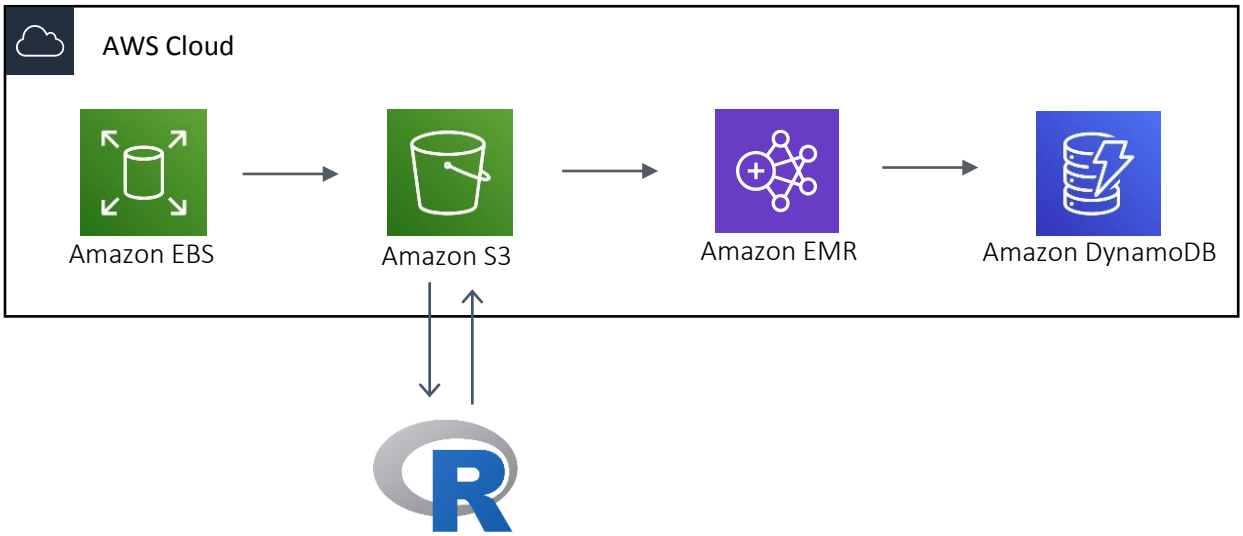
We load the tables in R and clean the dataset by replacing blanks with "NULL" and extracting only 7 columns from the tables. The extracted columns are:

```
"DayOfWeek","FlightDate","UniqueCarrier","Origin","Dest","DepDelay","ArrDelay"
```

The R script (included) loads each table, performs the two operations and writes a new table.

System Integration

The new tables are uploaded to S3 and are then integrated in the Hadoop Distribution File System (HDFS) through an AWS Elastic MapReduce (EMR) cluster, which copies the files from S3 to HDFS using the S3DistCp tool. The EMR cluster uses a m5.xlarge instance type, which has 4 vCPUs, 16 GB of memory and up to 10 Gbps of dedicated bandwidth. The cluster uses 1 master and 2 core nodes. Finally, the cluster is also configured with Hive 2.3.6, which will process the data queries. Selected queries will be stored in AWS DynamoDB, a NoSQL database. The data pipeline architecture can be seen below:



Approaches and Algorithms

We create a table in our cluster to move the data from S3 into the Hive file system as a table:

```
CREATE TABLE aviation
(DayOfWeek BIGINT,
FlightDate STRING,
UniqueCarrier STRING,
Origin STRING,
Dest STRING,
DepDelay BIGINT,
ArrDelay BIGINT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION 's3://aviation-database2/aviation-data/data/';
```

We use HiveQL to query the dataset and store the results of some of our queries in DynamoDB. Tables are created beforehand in DynamoDB in preparation for the storage of our selected queries. We then create tables in HDFS for our queries and map them to the DynamoDB tables. Finally, DynamoDB tables can also be queried using HiveQL, streamlining our table creations and relevant queries.

Results

Using HiveQL, we extract relevant information about the characteristics of commercial aviation in the United States. We display below the results, indicating performance in seconds, and produce the queries used in the appendix.

Top 10 most popular airports by numbers of flights to/from the airport

"ORD" 12449354
"ATL" 11540422
"DFW" 10799303
"LAX" 7723596
"PHX" 6585534
"DEN" 6273787
"DTW" 5636622
"IAH" 5480734
"MSP" 5199213
"SFO" 5171023

Time taken: 53.111 seconds

Days of the week by on-time arrival performance

6 4.3
2 5.99
7 6.61
1 6.72
3 7.2
4 9.09
5 9.72

Time taken: 80.806 seconds

For each airport X: top-10 carriers in decreasing order of on-time departure performance from X

1 "CMI" "OH" 0.61
2 "CMI" "US" 2.03
3 "CMI" "TW" 4.12
4 "CMI" "PI" 4.46
5 "CMI" "DH" 6.03
6 "CMI" "EV" 6.67
7 "CMI" "MQ" 8.02

1 "BWI" "F9" 0.76
2 "BWI" "PA (1)" 4.76
3 "BWI" "CO" 5.18
4 "BWI" "YV" 5.5
5 "BWI" "NW" 5.71
6 "BWI" "AA" 6.0
7 "BWI" "9E" 7.24
8 "BWI" "US" 7.49
9 "BWI" "DL" 7.68
10 "BWI" "UA" 7.74

1 "MIA" "9E" -3.0
2 "MIA" "EV" 1.2
3 "MIA" "TZ" 1.78
4 "MIA" "XE" 1.87
5 "MIA" "PA (1)" 4.2
6 "MIA" "NW" 4.5
7 "MIA" "US" 6.09
8 "MIA" "UA" 6.87
9 "MIA" "ML (1)" 7.5
10 "MIA" "FL" 8.57

1 "LAX" "MQ" 2.41
2 "LAX" "OO" 4.22
3 "LAX" "FL" 4.73
4 "LAX" "TZ" 4.76
5 "LAX" "PS" 4.86
6 "LAX" "NW" 5.12
7 "LAX" "F9" 5.73
8 "LAX" "HA" 5.81
9 "LAX" "YV" 6.02
10 "LAX" "US" 6.75

1 "IAH" "NW" 3.56
2 "IAH" "PA (1)" 3.98
3 "IAH" "PI" 3.99

4 "IAH" "US" 5.06
5 "IAH" "F9" 5.55
6 "IAH" "AA" 5.7
7 "IAH" "TW" 6.05
8 "IAH" "WN" 6.23
9 "IAH" "OO" 6.59
10 "IAH" "MQ" 6.71

1 "SFO" "TZ" 3.95
2 "SFO" "MQ" 4.85
3 "SFO" "F9" 5.16
4 "SFO" "PA (1)" 5.29
5 "SFO" "NW" 5.76
6 "SFO" "PS" 6.3
7 "SFO" "DL" 6.56
8 "SFO" "CO" 7.08
9 "SFO" "US" 7.53
10 "SFO" "TW" 7.79

For each source-destination pair X-Y: top-10 carriers in decreasing order of on-time arrival performance at Y from X

1 "CMI"-"ORD" "MQ" 10.14

1 "IND"-"CMH" "CO" -2.55
2 "IND"-"CMH" "AA" 5.5
3 "IND"-"CMH" "HP" 5.7
4 "IND"-"CMH" "NW" 5.76
5 "IND"-"CMH" "US" 6.88
6 "IND"-"CMH" "DL" 10.69
7 "IND"-"CMH" "EA" 10.81

1 "DFW"-"IAH" "PA (1)" -1.6
2 "DFW"-"IAH" "EV" 5.09
3 "DFW"-"IAH" "UA" 5.41
4 "DFW"-"IAH" "CO" 6.49
5 "DFW"-"IAH" "OO" 7.56
6 "DFW"-"IAH" "XE" 8.09
7 "DFW"-"IAH" "AA" 8.38
8 "DFW"-"IAH" "DL" 8.6
9 "DFW"-"IAH" "MQ" 9.1

1 "LAX"-"SFO" "TZ" -7.62
2 "LAX"-"SFO" "PS" -2.15
3 "LAX"-"SFO" "F9" -2.03
4 "LAX"-"SFO" "EV" 6.96
5 "LAX"-"SFO" "AA" 7.39

6 "LAX"-"SFO" "MQ" 7.81
7 "LAX"-"SFO" "US" 7.96
8 "LAX"-"SFO" "WN" 8.79
9 "LAX"-"SFO" "CO" 9.35
10 "LAX"-"SFO" "NW" 9.85

1 "JFK"-"LAX" "UA" 3.31
2 "JFK"-"LAX" "HP" 6.68
3 "JFK"-"LAX" "AA" 6.9
4 "JFK"-"LAX" "DL" 7.93
5 "JFK"-"LAX" "PA (1)" 11.02
6 "JFK"-"LAX" "TW" 11.7

1 "ATL"-"PHX" "FL" 4.55
2 "ATL"-"PHX" "US" 6.29
3 "ATL"-"PHX" "HP" 8.48
4 "ATL"-"PHX" "EA" 8.95
5 "ATL"-"PHX" "DL" 9.81

Time taken: 0.299 seconds

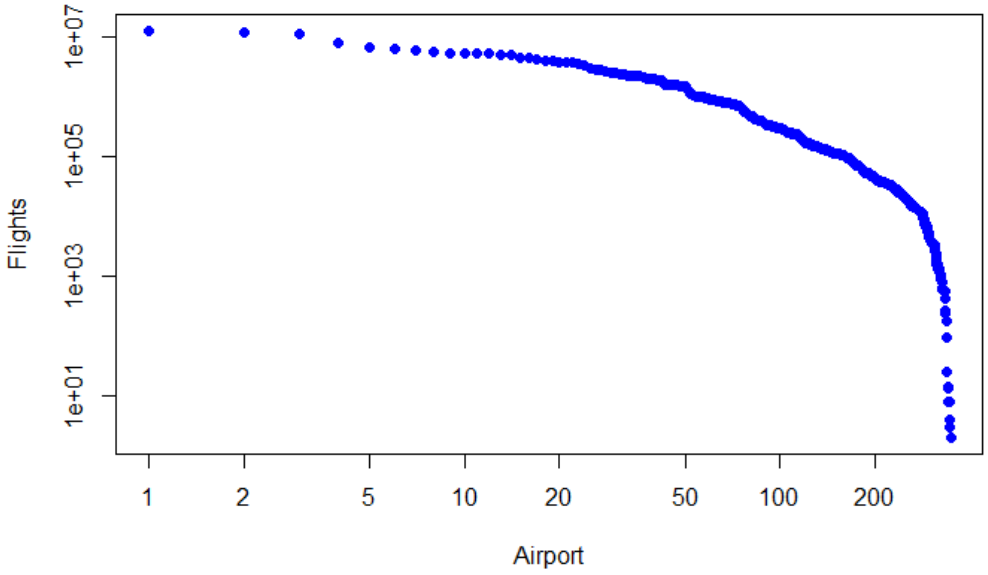
For each source-destination pair X-Y: mean arrival delay (in minutes) for a flight from X to Y

"CMI"-"ORD" 10.14
"IND"-"CMH" 2.9
"DFW"-"IAH" 7.65
"LAX"-"SFO" 9.59
"JFK"-"LAX" 6.64
"ATL"-"PHX" 9.02

Time taken: 0.274 seconds

Flight Distribution

Plotting the airports' "popularity" on a log-log axis table, we do not see a Zipf distribution as hypothesized. A Zipf distribution would show a decreasing straight line on a log-log plot, which is not demonstrated here. In fact, we see a log-normal distribution, which fits in well with our analysis, where a few airports are extremely popular with a subsequent exponential decrease in popularity.



Appendix

R Cleaning:

```r
file_names <- list.files(path = ".", pattern = "csv")
```

```r
for(i in 1:length(file_names)){
  file <- read.csv(file_names[i])
  file <- file[,c(5:7,12,18,26,37)]
  name <- paste("on_time", i,".csv", sep = '')
  write.csv(file, name, row.names = FALSE, na = 'NULL')
}
```

HiveQL Queries:

Group 1

Rank the top 10 most popular airports by numbers of flights to/from the airport.

```
SELECT origin, popular1+popular2 AS MostPopular
FROM
(SELECT origin, count(*) AS popular1 FROM aviation GROUP BY origin) AS
ori
INNER JOIN
(SELECT dest, count(*) AS popular2 FROM aviation GROUP BY dest) AS des
ON ori.origin = des.dest
ORDER BY MostPopular
DESC Limit 10;
```

Rank the days of the week by on-time arrival performance.

```
SELECT DayOfWeek, round(AVG(ArrDelay),2) AS perform
FROM aviation
WHERE DayOfWeek IS NOT NULL
GROUP BY DayOfWeek
ORDER BY perform;
```

Group 2

For each airport X, rank the top-10 carriers in decreasing order of on-time departure performance from X.

```
--Create DynamoDB Table

CREATE EXTERNAL TABLE ddb_topcarriersdepart
    (origin STRING,
    uniquecarrier STRING,
    avgdelay DOUBLE,
    rank BIGINT)
    STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
    TBLPROPERTIES(
```

```
    "dynamodb.table.name" = "Top_Carriers_Depart",

"dynamodb.column.mapping"="origin:Airport,uniquecarrier:Airline,avgdel
ay:Delay,rank:Rank"
    );

--Load Data to DynamoDB Table

INSERT OVERWRITE TABLE ddb_topcarriersdepart
SELECT *
FROM (
SELECT origin, uniquecarrier, avgdelay, rank() over(PARTITION BY
origin ORDER BY avgdelay, origin) AS airline_rank
FROM (
SELECT origin, uniquecarrier, round(AVG(depdelay),2) AS avgdelay
FROM aviation
GROUP BY origin, uniquecarrier
) AS no_rank
) AS rank
WHERE rank.airline_rank <= 10;

--Query DynamoDB Table

SELECT rank, origin, uniquecarrier, avgdelay
FROM ddb_topcarriersarrive
WHERE origin IN ('CMI', 'BWI', 'MIA', 'LAX', 'IAH', 'SFO');
```

For each source-destination pair X-Y, rank the top-10 carriers in decreasing order of on-time arrival performance at Y from X.

```
--Create DynamoDB Table

CREATE EXTERNAL TABLE ddb_topcarriersarrive
    (route STRING,
    uniquecarrier STRING,
    avgarrival DOUBLE,
    rank BIGINT)
    STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
    TBLPROPERTIES(
    "dynamodb.table.name" = "Top_Carriers_Arrival",

"dynamodb.column.mapping"="route:Route,uniquecarrier:Airline,avgarriva
l:Delay,rank:Rank"
    );

--Load Data to DynamoDB Table

INSERT OVERWRITE TABLE ddb_topcarriersarrive
SELECT concat(rank.origin, "-", rank.dest) AS route,
rank.uniquecarrier,
rank.avgdelay,
```

```
rank.airline_rank
FROM (
SELECT origin, dest, uniquecarrier, avgdelay, rank() over(PARTITION BY
origin, dest ORDER BY avgdelay, uniquecarrier) AS airline_rank
FROM (
SELECT origin, dest, uniquecarrier, round(AVG(arrdelay),2) AS avgdelay
FROM aviation
WHERE arrdelay IS NOT NULL
GROUP BY origin, dest, uniquecarrier
) AS no_rank
) AS rank
WHERE rank.airline_rank <= 10;


--Query DynamoDB Table

SELECT rank, route, uniquecarrier, avgarrival
FROM ddb_topcarriersarrive
WHERE route IN ('"CMI"-"ORD"','"IND"-"CMH"','"DFW"-"IAH"','"LAX"-
"SFO"','"JFK"-"LAX"','"ATL"-"PHX"');
```

For each source-destination pair X-Y, determine the mean arrival delay (in minutes) for a flight from X to Y.

```
--Create DynamoDB Table

CREATE EXTERNAL TABLE ddb_meandelay
    (route STRING,
    avgdelay DOUBLE)
    STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
    TBLPROPERTIES(
    "dynamodb.table.name" = "Mean_Delay",
    "dynamodb.column.mapping"="route:Route,avgdelay:Delay"
    );

--Load Data to DynamoDB Table

INSERT OVERWRITE TABLE ddb_meandelay
SELECT concat(origin, "-", dest) AS route,
round(AVG(ArrDelay),2) AS avgdelay
FROM aviation
WHERE arrdelay IS NOT NULL
GROUP BY Origin, Dest;


--Query DynamoDB Table

SELECT *
FROM ddb_meandelay
WHERE route IN ('"CMI"-"ORD"','"IND"-"CMH"','"DFW"-"IAH"','"LAX"-
"SFO"','"JFK"-"LAX"','"ATL"-"PHX"');
```

Group 3

Does the popularity distribution of airports follow a Zipf distribution? If not, what distribution does it follow?

```
CREATE EXTERNAL TABLE s3_aviation(Origin string, Flights bigint)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION 's3://aviation-database2/aviation-data/data/';

INSERT OVERWRITE TABLE s3_aviation
SELECT origin, departures+arrivals AS MostPopular
FROM
(SELECT origin, count(*) AS departures FROM aviation GROUP BY origin)
AS O
INNER JOIN
(SELECT dest, count(*) AS arrivals FROM aviation GROUP BY dest) AS D
ON O.origin = D.dest
ORDER BY MostPopular DESC;
```