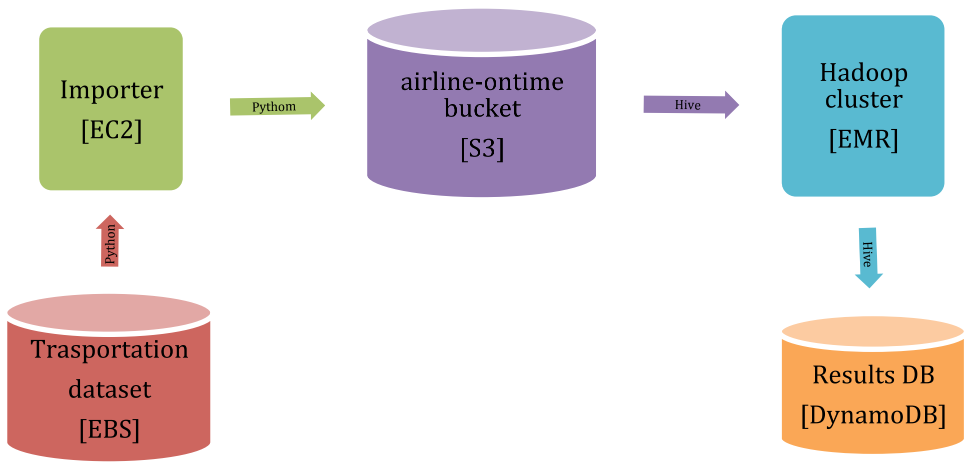
Cloud Computing Capstone

Task 1
Data Extraction, Batch Processing with Hadoop

This document describes the solution adopted to solve the first task of the Coursera Cloud Computing Capstone.

The next figure gives a brief overview of the architecture, which is explained in the following sections of the document:



The stack of technologies used for this task is:

- EC2: machine used to clean and import the data
- Python: script to clean and import the data
- S3: persistent storage of imported data
- EMR: managed Hadoop cluster to perform calculations on data
- HDFS: temporary storage of data to perform the queries
- Hive: query language to perform queries against the data
- DynamoDB: persistent key-value storage of task results.

Data extraction and cleaning

After analysing the provided dataset, I determined the only DB needed to answer task 1 questions is *airline-ontime*, which includes flight, origin, destination and delay info of every flight in the required period.

In order to import data, an Amazon EC2 instance (labelled *importer*) was launched with the transportation dataset EBS volume mounted on it. This data was loaded into a S3 bucket, named *airline-ontime*, using a Python script.

To optimize data storage and transmission and to make queries more efficient, only the minimum rows required to solve the problems were imported:

Year - Month - DayofMonth - DayOfWeek - UniqueCarrier - FlightNum - Origin - Dest -

CRSDepTime - DepDelay - CRSArrTime - ArrDelay - Cancelled

The imported data is organized into S3 using *Date* as the key, so it is more efficiently loaded into HDFS by Hive. Each folder inside the bucket contains data from a single day in CSV format, and is named with the following syntax: *date=yyyy-mm-dd*.

Systems integration

In order to answer the questions from task 1, an EMR cluster (Amazon managed Hadoop service) was launched with one master and two core nodes. Its integrated Hue console allows running Hive queries against the data.

After node was launched, data was loaded from S3 into HDFS using the following Hive command, which loads CSV data partitioned by date:

```
CREATE EXTERNAL TABLE airline_ontime (year INT, month INT, day INT,
weekday INT, carrier STRING, flight_num STRING, origin STRING, dest
STRING, deptime STRING, depdelay INT, arrtime STRING, arrdelay INT,
cancelled INT)
PARTITIONED BY (date string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ","
LOCATION 's3n://airline-ontime/';
```

DynamoDB result tables (*group2_ex1*, *group2_ex2*, *group2_ex4* and *group3_ex2*) were also created and mapped to HDFS using commands such as:

```
CREATE EXTERNAL TABLE group2_ex1 (airport STRING, carrier STRING,
mean_delay BIGINT)
    STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
    TBLPROPERTIES (
  "dynamodb.table.name" = "group2_ex1",
  "dynamodb.region" = "us-east-1",
  "dynamodb.throughput.write.percent" = "1",
  "dynamodb.column.mapping" = "airport:airport, carrier:carrier,
mean_delay:mean_delay");
```

With these initial commands, clean data is integrated into the Hadoop cluster and Hive queries can be run against it, storing results into DynamoDB tables.

Algorithms to answer each question

Each of the questions was answered using Hive queries, which use HQL (an SQL-like language) to query HDFS tables. Each HQL query is translated to one or more map-reduce tasks, which are run against data in HDFS.

Some examples of these queries are:

Group 1 ex 2:

```
select carrier, sum(arrdelay)/count(arrdelay) as mean_delay
from airline_ontime
where cancelled = 0 group by carrier order by mean_delay asc limit 10;
```

Group 2 ex 4:

```
insert overwrite table group2_ex4
select origin, dest as destination, sum(arrdelay)/count(arrdelay) as
mean_delay
from airline_ontime
where cancelled = 0 group by origin, dest;
```

Group 3 ex 1:

```
select o.origin as airport, o.flight_nr + d.flight_nr as popularity
from
  (select origin, count(origin) as flight_nr from airline_ontime
  where cancelled = 0 group by origin) as o,
  (select dest, count(dest) as flight_nr from airline_ontime
  where cancelled = 0 group by dest) as d
where o.origin = d.dest order by popularity desc;
```

Questions results

Question 1.1

Airport	Total flights
ORD	12051796
ATL	11323515
DFW	10591818
LAX	7586304
PHX	6505078
DEN	6183518
DTW	5504120
IAH	5416653
MSP	5087036
SFO	5062339

Question 1.2

Carrier	Mean delay (min)
HA	-1.01
AQ	1.15
PS	1.45
ML(1)	4.74
PA (1)	5.32
F9	5.46
NW	5.55
WN	5.56
OO	5.73
9E	5.86

Question 1.3

Weekday	Mean delay (min)
6	4.30
2	5.99
7	6.61
1	6.71
3	7.20

4	9.09
5	9.72

Question 2.1

Origin	Top 10 carriers by on time departure from origin
CMI	OH, US, PI, TW, EV, DH, MQ
BWI	F9, PA(1), NW, CO, YV, AA, US, UA, FL, DL
MIA	9E, EV, XE, TZ, PA(1), NW, US, UA, ML(1), FL
LAX	MQ, OO, PS, FL, TZ, HA, NW, F9, US, YV
IAH	PA(1), NW, PI, US, AA, F9, OO, HP, XE, MQ
SFO	TZ, MQ, PA(1), NW, F9, PS, DL, US, AA, CO

Question 2.2

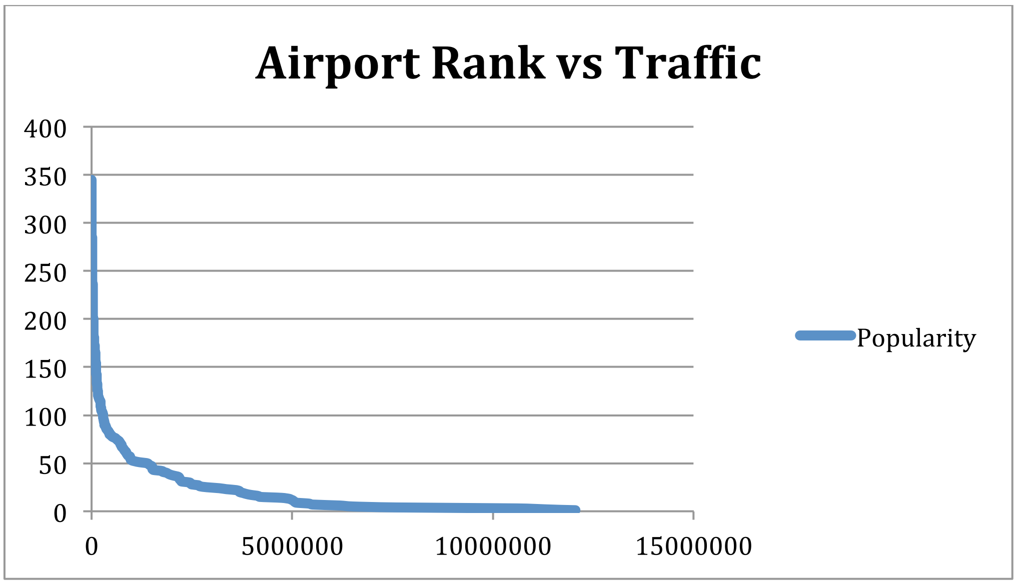
Origin	Top 10 destinations by on time departure from origin
CMI	ABI, PIT, CVG, DAY, STL, PIA, DFW, ATL, ORD
BWI	SAV, SRQ, DAB, IAD, MLB, UCA, CHO, DCA, IAH, OAJ
MIA	SHV, BUF, SAN, SLC, HOU, ISP, MEM, PSE, GNV, TLH
LAX	GRR, AZO, MSP, DTW, DAY, PIT, CVG, CLE, IAD, ATL
IAH	MSN, MLI, AGS, EFD, JAC, HOU, MTJ, VCT, RNO, BPT
SFO	SDF, MSO, PIH, LGA, PIE, FAR, OAK, BNA, MEM, SCK

Question 2.4

Origin	Destination	Mean arrival delay
CMI	ORD	10 min
IND	CMH	2 min
DFW	IAH	7 min
LAX	SFO	9 min
JFK	LAX	6 min
ATL	PHX	9 min

Question 3.1

The following table shows the relationship between airport rank and its popularity (sum of incoming and outgoing flights) using a linear scale.



The distribution followed by airport popularity has the characteristics of a Zipf one:

- A high rise near 0, meaning a huge number of airports (rank 100 to 350) with very low popularity (close to 0 flights).
- A small curved area in the low end of the scale, meaning a moderate number of airports (rank 100 to 10) with medium to low popularity.
- A long tail close to 0, meaning a very low number of airports (rank 10 to 0) with very high popularity (more than 5M flights).

Question 3.2

X	**Y**	**Z**	**DATE**	**Flight X-> Y**	**Flight Y->Z**
CMI	ORD	LAX	04/03/2008	MQ4401	AA1345
JAX	DFW	CRP	09/09/2008	AA845	MQ3627
SLC	BFL	LAX	01/04/2008	OO3755	OO5429
LAX	SFO	PHX	12/07/2008	WN3534	US412
DFW	ORD	DFW	10/06/2008	UA1104	OO6119
LAX	ORD	JFK	01/01/2008	UA944	B6918

Employed optimizations

In order to improve performance, lower network traffic and obtain results faster, several optimisations have been employed on the process:

- Removal of unused data columns: only needed columns have been imported from the original dataset, thus lowering the needs for data transfer and storage capacity.
- Data ordering and partitioning: the import script orders and partitions by date the data from the original dataset. This way the import process into HDFS is faster.
- Data preload into HDFS: data stored into S3 is loaded to HDFS before starting to process it. This way, we can achieve persistent storage in S3 and fast access from HDFS.
- Use of DynamoDB ranges and indexes: result data can be queried and ordered directly in DynamoDB via the use of range keys and local secondary indexes.

Results analysis

Results obtained from task 1 exercises do make sense and can be useful, as they transform a huge amount of information into particular answers to specific questions.

These answers summarize data that might be useful for individuals to plan routes or make travelling decisions and for companies to improve flights and airport connections performance.

Video report

https://youtu.be/RuH4cJrV_Jc