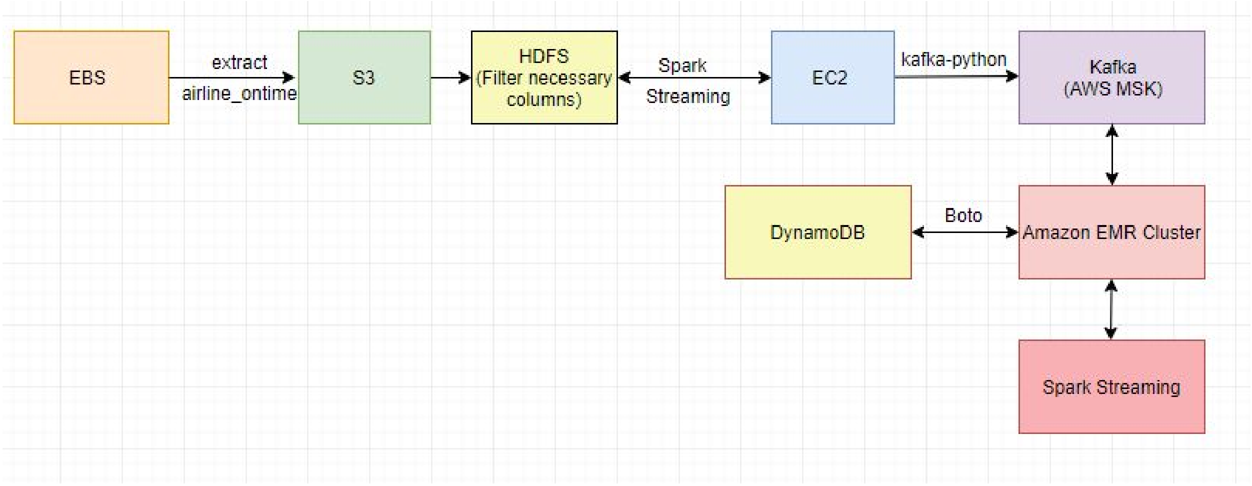
## Overall Design



- For Task 2, I have adopted Kafka + Spark Streaming
- Airline_ontime folder is unzipped to S3.
- Relevant columns from CSV are filtered and stored in HDFS which is then streamed to EC2. EC2 instance then writes the data to Kafka topic
- AWS MSK is used to set up a Kafka Cluster with replication factor = 3 one m5.x2large node in each of the 3 regions. Kafka version used in 2.1.0
- Data in Kafka topic is consumed by EMR cluster, task2 queries are processed and the results stored in DynamoDB using boto python library.

## Data Extraction

EBS Volume is mounted on EC2 and then only the required folder (airline_ontime) is unzipped to S3.

```
./ebs_to_s3.sh $EBS_MOUNTED_FOLDER $S3_BUCKET
```

Source
Extraction Script

Data in S3 is copied to HDFS and relevant columns filtered using spark streaming

Source
streamFiltering.py (discards cancelled flights)
streamFilteringWithCancelled.py (includes cancelled flights)
streamFiltering_2008.py (includes info only for 2008 flights and discards cancelled flights)

- Filtered data is moved to kafka cluster under different topics. Kafka-python library is used for this

  Some optimization flags used to prevent data drop and speed up write

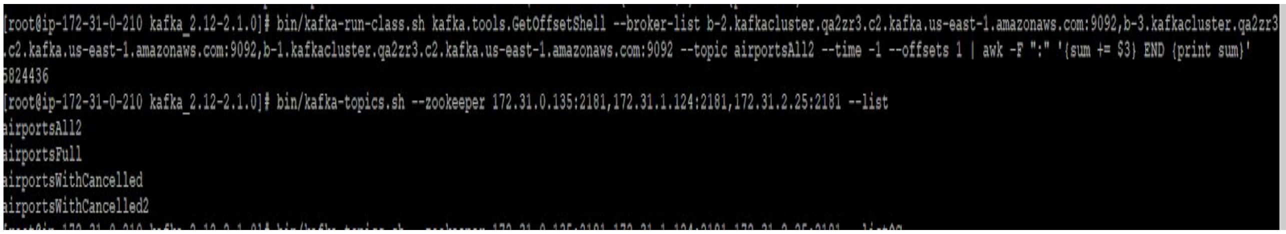
  ```
  batch_size=98304 (default batch size is 16k)
  linger_ms=100
  acks='all' (acks set to false by default)
  ```

  Source

  kafkaProducer.py

  kafkaProducer_2008.py

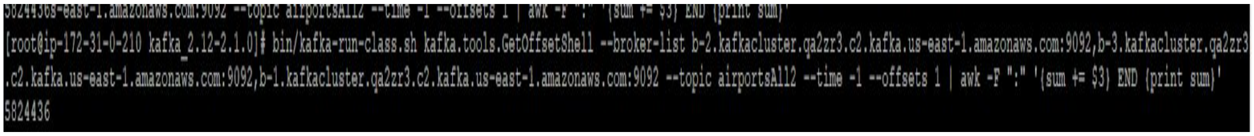
  Snapshot of kafka topics created



```
[root@ip-172-31-0-210 kafka_2.12-2.1.0]# bin/kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list b-2.kafkacluster.qa2zr3.c2.kafka.us-east-1.amazonaws.com:9092,b-3.kafkacluster.qa2zr3
.c2.kafka.us-east-1.amazonaws.com:9092,b-1.kafkacluster.qa2zr3.c2.kafka.us-east-1.amazonaws.com:9092 --topic airportsAll2 --time -1 --offsets 1 | awk -F ":" '{sum += $3} END {print sum}'
5824436
[root@ip-172-31-0-210 kafka_2.12-2.1.0]# bin/kafka-topics.sh --zookeeper 172.31.0.135:2181,172.31.1.124:2181,172.31.2.25:2181 --list
airportsAll2
airportsFull
airportsWithCancelled
airportsWithCancelled2
```

  Number of messages in kafka topic for year 2008 (not including cancelled flights)

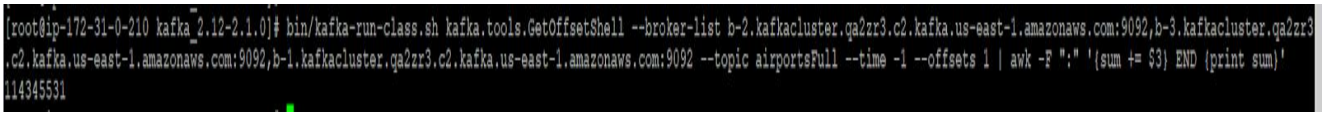
  5824436



```
5824436s-east-1.amazonaws.com:9092 --topic airportsAll2 --time -1 --offsets 1 | awk -F ":" '{sum += $3} END {print sum}'
[root@ip-172-31-0-210 kafka_2.12-2.1.0]# bin/kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list b-2.kafkacluster.qa2zr3.c2.kafka.us-east-1.amazonaws.com:9092,b-3.kafkacluster.qa2zr3
.c2.kafka.us-east-1.amazonaws.com:9092,b-1.kafkacluster.qa2zr3.c2.kafka.us-east-1.amazonaws.com:9092 --topic airportsAll2 --time -1 --offsets 1 | awk -F ":" '{sum += $3} END {print sum}'
5824436
```

  Number of messages in kafka topic for all years (not including cancelled flights)

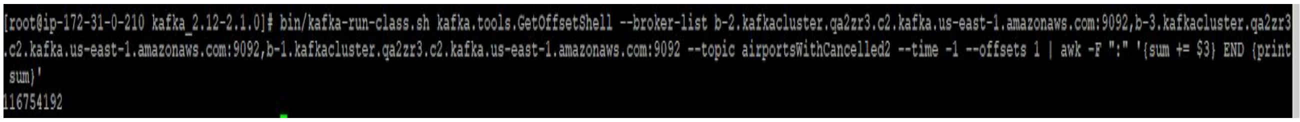
  114345531



```
[root@ip-172-31-0-210 kafka_2.12-2.1.0]# bin/kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list b-2.kafkacluster.qa2zr3.c2.kafka.us-east-1.amazonaws.com:9092,b-3.kafkacluster.qa2zr3
.c2.kafka.us-east-1.amazonaws.com:9092,b-1.kafkacluster.qa2zr3.c2.kafka.us-east-1.amazonaws.com:9092 --topic airportsFull --time -1 --offsets 1 | awk -F ":" '{sum += $3} END {print sum}'
114345531
```

  Number of messages in kafka topic for all years (including cancelled flights)

  116754192



```
[root@ip-172-31-0-210 kafka_2.12-2.1.0]# bin/kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list b-2.kafkacluster.qa2zr3.c2.kafka.us-east-1.amazonaws.com:9092,b-3.kafkacluster.qa2zr3
.c2.kafka.us-east-1.amazonaws.com:9092,b-1.kafkacluster.qa2zr3.c2.kafka.us-east-1.amazonaws.com:9092 --topic airportsWithCancelled2 --time -1 --offsets 1 | awk -F ":" '{sum += $3} END {print
 sum}'
116754192
```

- Kafka Consumer is EMR cluster with Spark 2.4.2 which reads from the topics and stores final answers in dynaoDB

## Problem 1.1

Strategy:

1. map() each airport_id to (airport_id,1)
2. reduce() count for each airport_id to get (airport_id,count)
3. filter out top 10 airports with highest counts

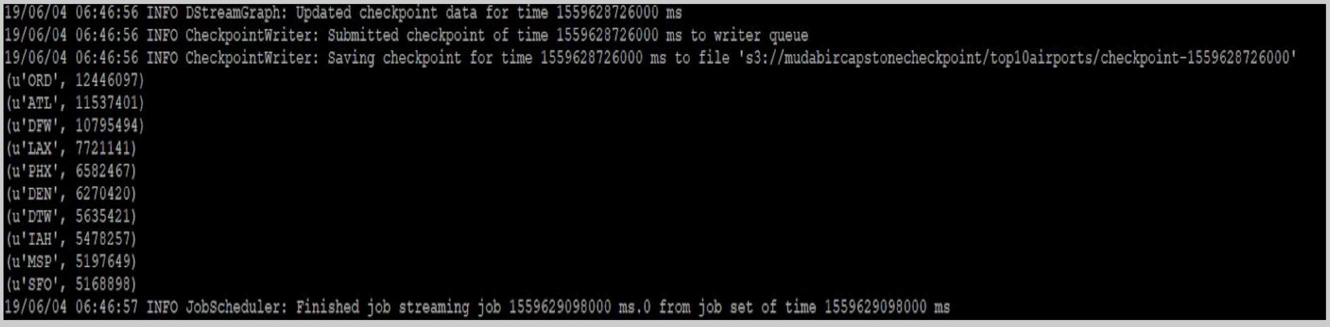
```
spark-submit --executor-memory 6g --packages
org.apache.spark:spark-streaming-kafka-0-8_2.11:2.1.0 --conf
spark.streaming.backpressure.enabled=true --conf
spark.streaming.kafka.maxRatePerPartition=250000 g1_1.py
```

Source:

g1_1.py

Results:

Format: (AirportID, Count)



```
19/06/04 06:46:56 INFO DStreamGraph: Updated checkpoint data for time 1559628726000 ms
19/06/04 06:46:56 INFO CheckpointWriter: Submitted checkpoint of time 1559628726000 ms to writer queue
19/06/04 06:46:56 INFO CheckpointWriter: Saving checkpoint for time 1559628726000 ms to file 's3://mudabircapstonecheckpoint/top10airports/checkpoint-1559628726000'
(u'ORD', 12446097)
(u'ATL', 11537401)
(u'DFW', 10795494)
(u'LAX', 7721141)
(u'PHX', 6582467)
(u'DEN', 6270420)
(u'DTW', 5635421)
(u'IAH', 5478257)
(u'MSP', 5197649)
(u'SFO', 5168898)
19/06/04 06:46:57 INFO JobScheduler: Finished job streaming job 1559629098000 ms.0 from job set of time 1559629098000 ms
```

## Problem 1.2

Strategy

1) Filter for all non cancelled flights and map (flightID, ArrivalDelay)
2) Get (flightID, (`delaySum,count,avgArrivalDelay`) //see updateStateByKey
3) Filter top 10 flights with best Arrival delay

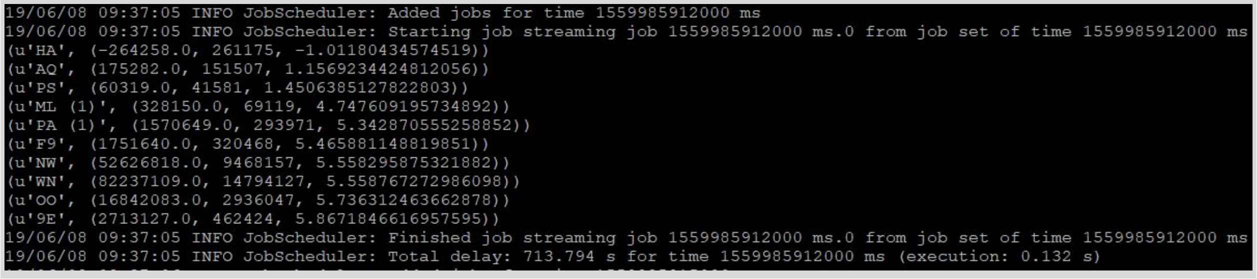
Source:

g1_2.py

Results:

Format: FlightID [totalDepDelay, totalCount, AvgArrivalDelay]

```
(u'HA', (-264258.0, 261175, -1.01180434574519))
(u'AQ', (175282.0, 151507, 1.1569234424812056))
(u'PS', (60319.0, 41581, 1.4506385127822803))
(u'ML (1)', (328150.0, 69119, 4.747609195734892))
(u'PA (1)', (1570649.0, 293971, 5.342870555258852))
(u'F9', (1751640.0, 320468, 5.465881148819851))
(u'NW', (52626818.0, 9468157, 5.558295875321882))
(u'WN', (82237109.0, 14794127, 5.558767272986098))
(u'OO', (16842083.0, 2936047, 5.736312463662878))
(u'9E', (2713127.0, 462424, 5.8671846616957595))
```



```
19/06/08 09:37:05 INFO JobScheduler: Added jobs for time 1559985912000 ms
19/06/08 09:37:05 INFO JobScheduler: Starting job streaming job 1559985912000 ms.0 from job set of time 1559985912000 ms
(u'HA', (-264258.0, 261175, -1.01180434574519))
(u'AQ', (175282.0, 151507, 1.1569234424812056))
(u'PS', (60319.0, 41581, 1.4506385127822803))
(u'ML (1)', (328150.0, 69119, 4.747609195734892))
(u'PA (1)', (1570649.0, 293971, 5.342870555258852))
(u'F9', (1751640.0, 320468, 5.465881148819851))
(u'NW', (52626818.0, 9468157, 5.558295875321882))
(u'WN', (82237109.0, 14794127, 5.558767272986098))
(u'OO', (16842083.0, 2936047, 5.736312463662878))
(u'9E', (2713127.0, 462424, 5.8671846616957595))
19/06/08 09:37:05 INFO JobScheduler: Finished job streaming job 1559985912000 ms.0 from job set of time 1559985912000 ms
19/06/08 09:37:05 INFO JobScheduler: Total delay: 713.794 s for time 1559985912000 ms (execution: 0.132 s)
```

**Problem 2.1**

```
'''
        The incoming data format is
        Year|Month|date|DayofWeek|UniqueCarrier|FlightNum|Origin|Dest|CRSDeptime|DepDelay|ArrDelay
        '''
```

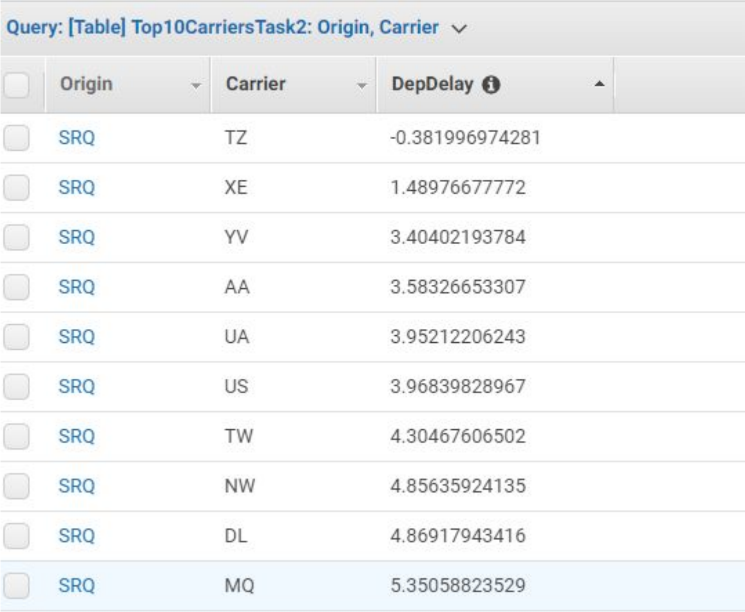
(For each origin airport X, rank top 10 carriers in decreasing order of ontime departure)
Strategy:

1) Filter for all non cancelled flights and map ((OriginAirport,carrier), DepDelay)
2) Get ((OriginAirpor,carrier), (depDelaySum,count, avgDepDelay)) // see `flightsDelay.updateStateByKey(updateFunction)`
3) For each OriginAirport, filter 10 best carriers in terms of Dep delay
4) Store results in dynamoDB
    - Main DB has OriginAirport as Partition key and Carrier as Sort key
    - Also created a Secondary index mapped to main DB with Partition key as OriginAirport and sort key as DepDelay. Hence querying the Secondary index returns results sorted in increasing order of DepDelay.

Source:

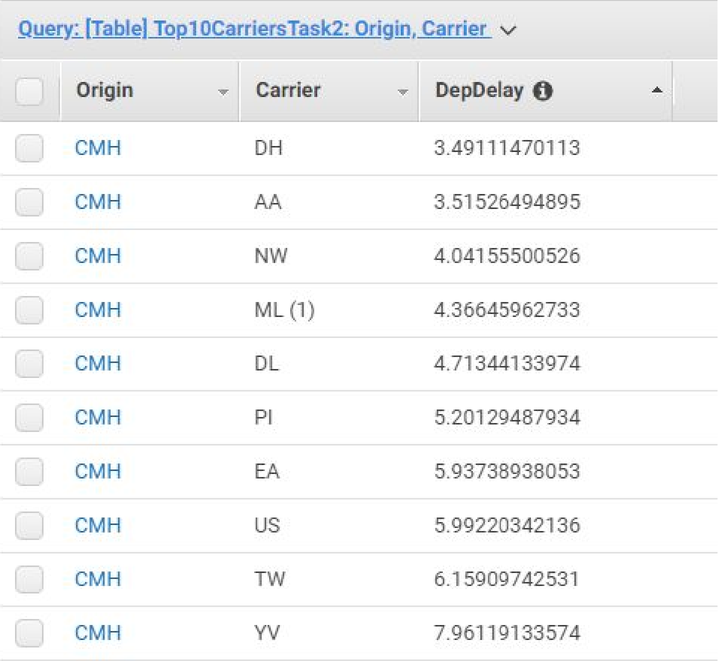
g2_1.py

Results:



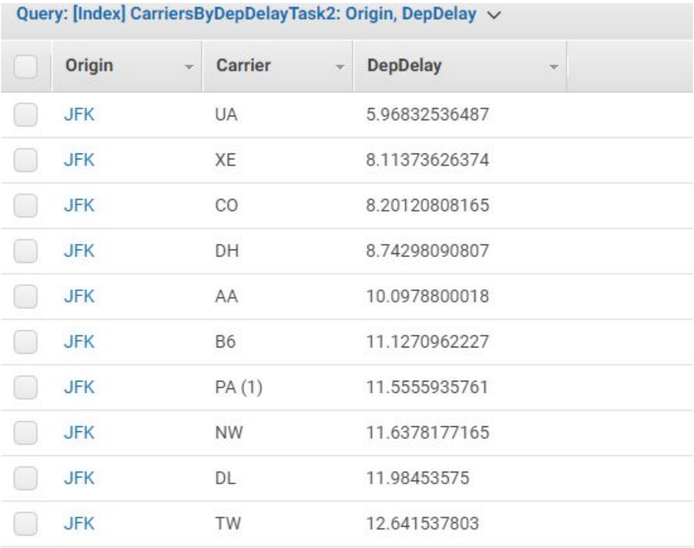
Query: [Table] Top10CarriersTask2: Origin, Carrier

| □ | Origin | Carrier | DepDelay |
|---|---|---|---|
| □ | SRQ | TZ | -0.381996974281 |
| □ | SRQ | XE | 1.48976677772 |
| □ | SRQ | YV | 3.40402193784 |
| □ | SRQ | AA | 3.58326653307 |
| □ | SRQ | UA | 3.95212206243 |
| □ | SRQ | US | 3.96839828967 |
| □ | SRQ | TW | 4.30467606502 |
| □ | SRQ | NW | 4.85635924135 |
| □ | SRQ | DL | 4.86917943416 |
| □ | SRQ | MQ | 5.35058823529 |



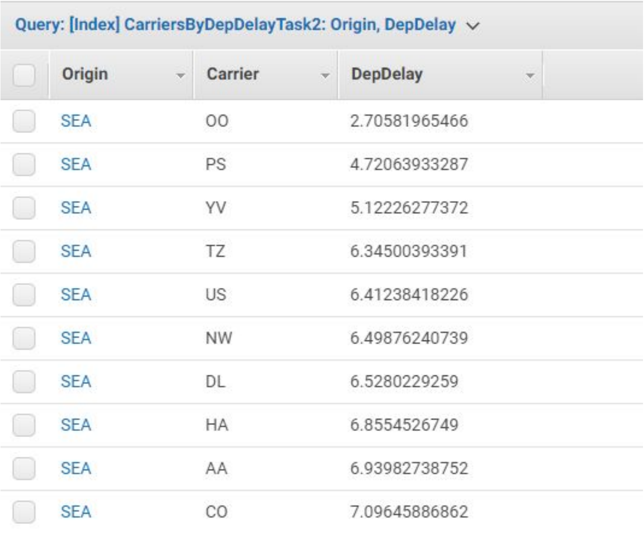
Query: [Table] Top10CarriersTask2: Origin, Carrier

| □ | Origin | Carrier | DepDelay |
|---|---|---|---|
| □ | CMH | DH | 3.49111470113 |
| □ | CMH | AA | 3.51526494895 |
| □ | CMH | NW | 4.04155500526 |
| □ | CMH | ML (1) | 4.36645962733 |
| □ | CMH | DL | 4.71344133974 |
| □ | CMH | PI | 5.20129487934 |
| □ | CMH | EA | 5.93738938053 |
| □ | CMH | US | 5.99220342136 |
| □ | CMH | TW | 6.15909742531 |
| □ | CMH | YV | 7.96119133574 |



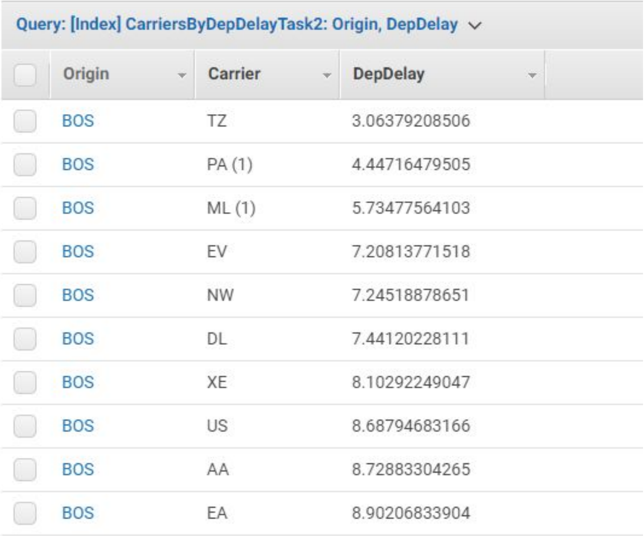
Query: [Index] CarriersByDepDelayTask2: Origin, DepDelay

| Origin | Carrier | DepDelay |
|---|---|---|
| JFK | UA | 5.96832536487 |
| JFK | XE | 8.11373626374 |
| JFK | CO | 8.20120808165 |
| JFK | DH | 8.74298090807 |
| JFK | AA | 10.0978800018 |
| JFK | B6 | 11.1270962227 |
| JFK | PA (1) | 11.5555935761 |
| JFK | NW | 11.6378177165 |
| JFK | DL | 11.98453575 |
| JFK | TW | 12.641537803 |



Query: [Index] CarriersByDepDelayTask2: Origin, DepDelay

| Origin | Carrier | DepDelay |
|---|---|---|
| SEA | OO | 2.70581965466 |
| SEA | PS | 4.72063933287 |
| SEA | YV | 5.12226277372 |
| SEA | TZ | 6.34500393391 |
| SEA | US | 6.41238418226 |
| SEA | NW | 6.49876240739 |
| SEA | DL | 6.5280229259 |
| SEA | HA | 6.8554526749 |
| SEA | AA | 6.93982738752 |
| SEA | CO | 7.09645886862 |



Query: [Index] CarriersByDepDelayTask2: Origin, DepDelay

| Origin | Carrier | DepDelay |
|---|---|---|
| BOS | TZ | 3.06379208506 |
| BOS | PA (1) | 4.44716479505 |
| BOS | ML (1) | 5.73477564103 |
| BOS | EV | 7.20813771518 |
| BOS | NW | 7.24518878651 |
| BOS | DL | 7.44120228111 |
| BOS | XE | 8.10292249047 |
| BOS | US | 8.68794683166 |
| BOS | AA | 8.72883304265 |
| BOS | EA | 8.90206833904 |

**Problem 2.2**

Strategy:

Same as problem 2.1 except replace carrier with DestAirport

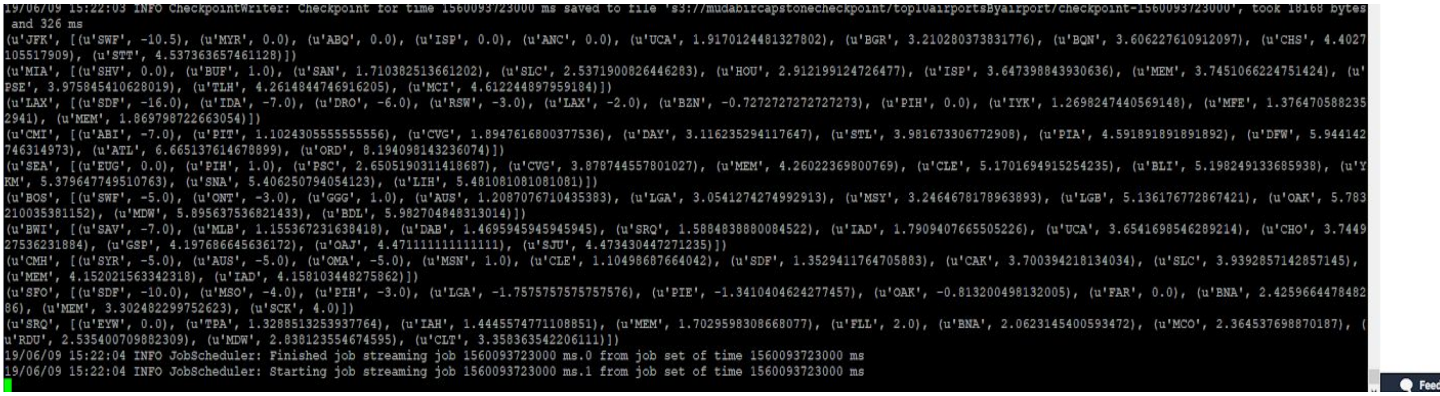
Source:

g2_2.py

Result:

Output format:

Origin, ((dest1, delay), (dest2,delay)....)



```
19/06/09 15:22:03 INFO CheckpointWriter: Checkpoint for time 1560093723000 ms saved to file 's3://mudabircapstonecheckpoint/top10airportsByairport/checkpoint-1560093723000', took 18168 bytes
 and 326 ms
(u'JFK', [(u'SWF', -10.5), (u'MYR', 0.0), (u'ABQ', 0.0), (u'ISP', 0.0), (u'ANC', 0.0), (u'UCA', 1.9170124481327802), (u'BGR', 3.210280373831776), (u'BQN', 3.606227610912097), (u'CHS', 4.4027
105517909), (u'STT', 4.537363657461128)])
(u'MIA', [(u'SHV', 0.0), (u'BUF', 1.0), (u'SAN', 1.710382513661202), (u'SLC', 2.5371900826446283), (u'HOU', 2.912199124726477), (u'ISP', 3.647398843930636), (u'MEM', 3.7451066224751424), (u'
PSE', 3.975845410628019), (u'TLH', 4.2614844746916205), (u'MCI', 4.612244897959184)])
(u'LAX', [(u'SDF', -16.0), (u'IDA', -7.0), (u'DRO', -6.0), (u'RSW', -3.0), (u'LAX', -2.0), (u'BZN', -0.7272727272727273), (u'PIH', 0.0), (u'IYK', 1.2698247440569148), (u'MFE', 1.376470588235
2941), (u'MEM', 1.869798722663054)])
(u'CMI', [(u'ABI', -7.0), (u'PIT', 1.1024305555555556), (u'CVG', 1.8947616800377536), (u'DAY', 3.116235294117647), (u'STL', 3.981673306772908), (u'PIA', 4.591891891891892), (u'DFW', 5.944142
746314973), (u'ATL', 6.665137614678899), (u'ORD', 8.194098143236074)])
(u'SEA', [(u'EUG', 0.0), (u'PIH', 1.0), (u'PSC', 2.6505190311418687), (u'CVG', 3.878744557801027), (u'MEM', 4.26022369800769), (u'CLE', 5.1701694915254235), (u'BLI', 5.198249133685938), (u'Y
KM', 5.379647749510763), (u'SNA', 5.406250794054123), (u'LIH', 5.481081081081081)])
(u'BOS', [(u'SWF', -5.0), (u'ONT', -3.0), (u'GGG', 1.0), (u'AUS', 1.2087076710435383), (u'LGA', 3.0541274274992913), (u'MSY', 3.2464678178963893), (u'LGB', 5.136176772867421), (u'OAK', 5.783
210035381152), (u'MDW', 5.895637536821433), (u'BDL', 5.982704848313014)])
(u'BWI', [(u'SAV', -7.0), (u'MLB', 1.155367231638418), (u'DAB', 1.4695945945945945), (u'SRQ', 1.5884838880084522), (u'IAD', 1.7909407665505226), (u'UCA', 3.6541698546289214), (u'CHO', 3.7449
27536231884), (u'GSP', 4.197686645636172), (u'OAJ', 4.471111111111111), (u'SJU', 4.473430447271235)])
(u'CMH', [(u'SYR', -5.0), (u'AUS', -5.0), (u'OMA', -5.0), (u'MSN', 1.0), (u'CLE', 1.10498687664042), (u'SDF', 1.3529411764705883), (u'CAK', 3.700394218134034), (u'SLC', 3.9392857142857145),
(u'MEM', 4.152021563342318), (u'IAD', 4.158103448275862)])
(u'SFO', [(u'SDF', -10.0), (u'MSO', -4.0), (u'PIH', -3.0), (u'LGA', -1.7575757575757576), (u'PIE', -1.3410404624277457), (u'OAK', -0.813200498132005), (u'FAR', 0.0), (u'BNA', 2.4259664478482
86), (u'MEM', 3.302482299752623), (u'SCK', 4.0)])
(u'SRQ', [(u'EYW', 0.0), (u'TPA', 1.3288513253937764), (u'IAH', 1.4445574771108851), (u'MEM', 1.7029598308668077), (u'FLL', 2.0), (u'BNA', 2.0623145400593472), (u'MCO', 2.364537698870187), (
u'RDU', 2.535400709882309), (u'MDW', 2.838123554674595), (u'CLT', 3.358363542206111)])
19/06/09 15:22:04 INFO JobScheduler: Finished job streaming job 1560093723000 ms.0 from job set of time 1560093723000 ms
19/06/09 15:22:04 INFO JobScheduler: Starting job streaming job 1560093723000 ms.1 from job set of time 1560093723000 ms
```

From logs:

 and 322 ms

(u'JFK', [(u'SWF', -10.5), (u'MYR', 0.0), (u'ABQ', 0.0), (u'ISP', 0.0), (u'ANC', 0.0), (u'UCA', 1.9170124481327802), (u'BGR', 3.210280373831776), (u'BQN', 3.606227610912097), (u'CHS', 4.4027105517909), (u'STT', 4.537363657461128)])

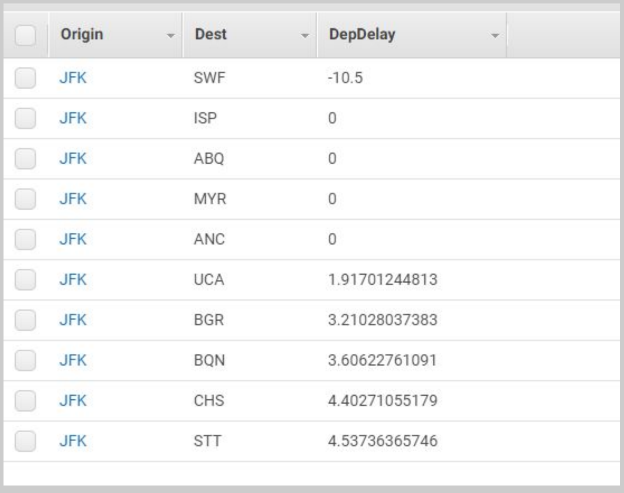
(u'MIA', [(u'SHV', 0.0), (u'BUF', 1.0), (u'SAN', 1.710382513661202), (u'SLC', 2.5371900826446283), (u'HOU', 2.912199124726477), (u'ISP', 3.647398843930636), (u'MEM', 3.7451066224751424), (u'PSE', 3.975845410628019), (u'TLH', 4.2614844746916205), (u'MCI', 4.612244897959184)])

(u'LAX', [(u'SDF', -16.0), (u'IDA', -7.0), (u'DRO', -6.0), (u'RSW', -3.0), (u'LAX', -2.0), (u'BZN', -0.7272727272727273), (u'PIH', 0.0), (u'IYK', 1.2698247440569148), (u'MFE', 1.3764705882352941), (u'MEM', 1.869798722663054)])

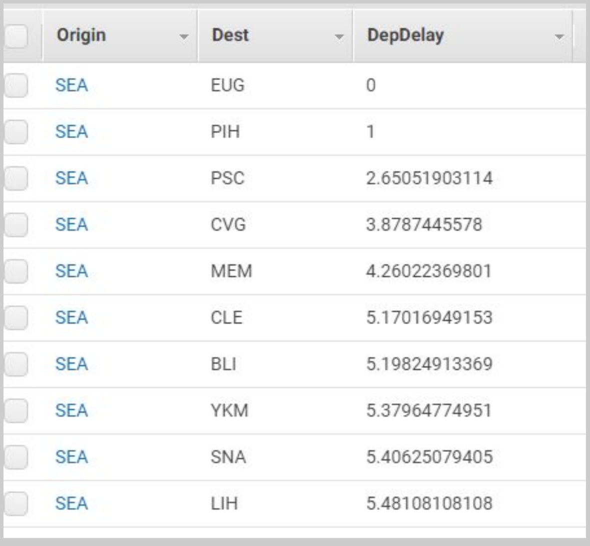
(u'CMI', [(u'ABI', -7.0), (u'PIT', 1.1024305555555556), (u'CVG', 1.8947616800377536), (u'DAY', 3.116235294117647), (u'STL', 3.981673306772908), (u'PIA', 4.591891891891892), (u'DFW', 5.944142746314973), (u'ATL', 6.665137614678899), (u'ORD', 8.194098143236074)])

(u'SEA', [(u'EUG', 0.0), (u'PIH', 1.0), (u'PSC', 2.6505190311418687), (u'CVG', 3.878744557801027), (u'MEM', 4.26022369800769), (u'CLE', 5.1701694915254235), (u'BLI', 5.198249133685938), (u'YKM', 5.379647749510763), (u'SNA', 5.406250794054123), (u'LIH', 5.481081081081081)])

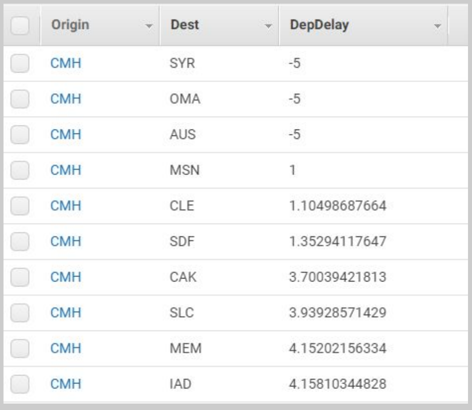
```
(u'BOS', [(u'SWF', -5.0), (u'ONT', -3.0), (u'GGG', 1.0), (u'AUS', 1.2087076710435383), (u'LGA',
3.0541274274992913), (u'MSY', 3.2464678178963893), (u'LGB', 5.136176772867421),
(u'OAK', 5.783210035381152), (u'MDW', 5.895637536821433), (u'BDL', 5.982704848313014)])
(u'BWI', [(u'SAV', -7.0), (u'MLB', 1.155367231638418), (u'DAB', 1.4695945945945945),
(u'SRQ', 1.5884838880084522), (u'IAD', 1.7909407665505226), (u'UCA',
3.6541698546289214), (u'CHO', 3.744927536231884), (u'GSP', 4.197686645636172), (u'OAJ',
4.4711111111111111), (u'SJU', 4.473430447271235)])
(u'CMH', [(u'SYR', -5.0), (u'AUS', -5.0), (u'OMA', -5.0), (u'MSN', 1.0), (u'CLE',
1.10498687664042), (u'SDF', 1.3529411764705883), (u'CAK', 3.700394218134034), (u'SLC',
3.9392857142857145), (u'MEM', 4.152021563342318), (u'IAD', 4.158103448275862)])
(u'SFO', [(u'SDF', -10.0), (u'MSO', -4.0), (u'PIH', -3.0), (u'LGA', -1.7575757575757576), (u'PIE',
-1.3410404624277457), (u'OAK', -0.813200498132005), (u'FAR', 0.0), (u'BNA',
2.425966447848286), (u'MEM', 3.302482299752623), (u'SCK', 4.0)])
(u'SRQ', [(u'EYW', 0.0), (u'TPA', 1.3288513253937764), (u'IAH', 1.4445574771108851),
(u'MEM', 1.7029598308668077), (u'FLL', 2.0), (u'BNA', 2.0623145400593472), (u'MCO',
2.364537698870187), (u'RDU', 2.535400709882309), (u'MDW', 2.838123554674595), (u'CLT',
3.358363542206111)])
19/06/09 15:21:24 INFO JobScheduler: Finished job streaming job 1560093684000 ms.0 from
job set of time 1560093684000 ms
19/06/09 15:21:24 INFO JobScheduler: Starting job streaming job 1560093684000 ms.1 from
job set of time 1560093684000 ms
19/06/09 15:21:25 INFO JobScheduler: Finished job streaming job 1560093684000 ms.1 from
job set of time 1560093684000 ms
19/06/09 15:21:25 INFO JobScheduler: Total delay: 1.318 s for time 1560093684000 ms
(execution: 1.206 s)
19/06/09 15:21:25 INFO KafkaRDD: Removing RDD 21220 from persistence list
19/06/09 15:21:25 INFO JobGenerator: Checkpointing graph for time 1560093684000 ms
19/06/09 15:21:25 INFO DStreamGraph: Updating checkpoint data for time 1560093684000 ms
19/06/09 15:21:25 INFO DStreamGraph: Updated checkpoint data for time 1560093684000 ms
```



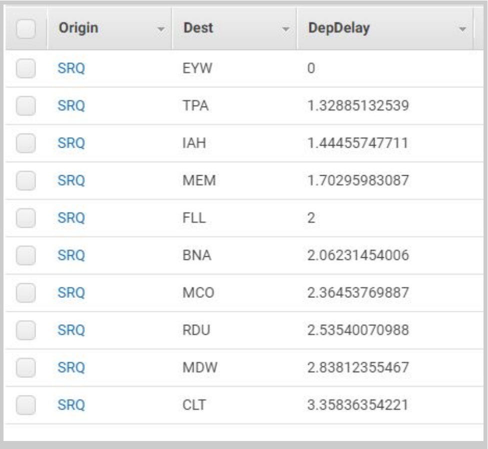
| □ | Origin | Dest | DepDelay |
|---|---|---|---|
| □ | JFK | SWF | -10.5 |
| □ | JFK | ISP | 0 |
| □ | JFK | ABQ | 0 |
| □ | JFK | MYR | 0 |
| □ | JFK | ANC | 0 |
| □ | JFK | UCA | 1.91701244813 |
| □ | JFK | BGR | 3.21028037383 |
| □ | JFK | BQN | 3.60622761091 |
| □ | JFK | CHS | 4.40271055179 |
| □ | JFK | STT | 4.53736365746 |



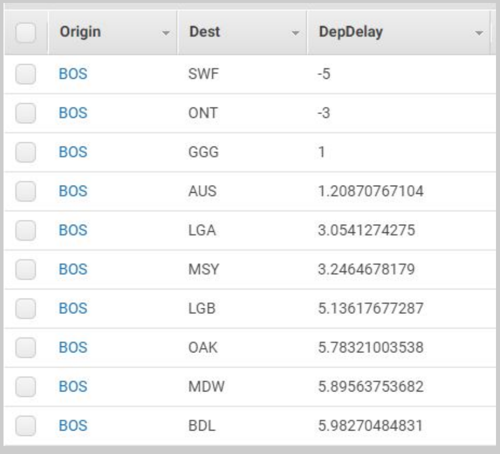
| □ | Origin | Dest | DepDelay |
|---|---|---|---|
| □ | SEA | EUG | 0 |
| □ | SEA | PIH | 1 |
| □ | SEA | PSC | 2.65051903114 |
| □ | SEA | CVG | 3.8787445578 |
| □ | SEA | MEM | 4.26022369801 |
| □ | SEA | CLE | 5.17016949153 |
| □ | SEA | BLI | 5.19824913369 |
| □ | SEA | YKM | 5.37964774951 |
| □ | SEA | SNA | 5.40625079405 |
| □ | SEA | LIH | 5.48108108108 |



| □ | Origin | Dest | DepDelay |
|---|---|---|---|
| □ | CMH | SYR | -5 |
| □ | CMH | OMA | -5 |
| □ | CMH | AUS | -5 |
| □ | CMH | MSN | 1 |
| □ | CMH | CLE | 1.10498687664 |
| □ | CMH | SDF | 1.35294117647 |
| □ | CMH | CAK | 3.70039421813 |
| □ | CMH | SLC | 3.93928571429 |
| □ | CMH | MEM | 4.15202156334 |
| □ | CMH | IAD | 4.15810344828 |



| □ | Origin | Dest | DepDelay |
|---|---|---|---|
| □ | SRQ | EYW | 0 |
| □ | SRQ | TPA | 1.32885132539 |
| □ | SRQ | IAH | 1.44455747711 |
| □ | SRQ | MEM | 1.70295983087 |
| □ | SRQ | FLL | 2 |
| □ | SRQ | BNA | 2.06231454006 |
| □ | SRQ | MCO | 2.36453769887 |
| □ | SRQ | RDU | 2.53540070988 |
| □ | SRQ | MDW | 2.83812355467 |
| □ | SRQ | CLT | 3.35836354221 |



| □ | Origin | Dest | DepDelay |
|---|---|---|---|
| □ | BOS | SWF | -5 |
| □ | BOS | ONT | -3 |
| □ | BOS | GGG | 1 |
| □ | BOS | AUS | 1.20870767104 |
| □ | BOS | LGA | 3.0541274275 |
| □ | BOS | MSY | 3.2464678179 |
| □ | BOS | LGB | 5.13617677287 |
| □ | BOS | OAK | 5.78321003538 |
| □ | BOS | MDW | 5.89563753682 |
| □ | BOS | BDL | 5.98270484831 |

**Problem 2.4**

(determine the mean arrival delay (in minutes) for a flight from X to Y)

Strategy:

1. Filter for all non cancelled flights and map (origin,dest) -> (ArrDelay)
2. Get (origin, dest) -> `(ArrDelaySum,count, avgArrDelay) //` See flightsDelay.updateStateByKey(updateFunction)
3. Save to DB

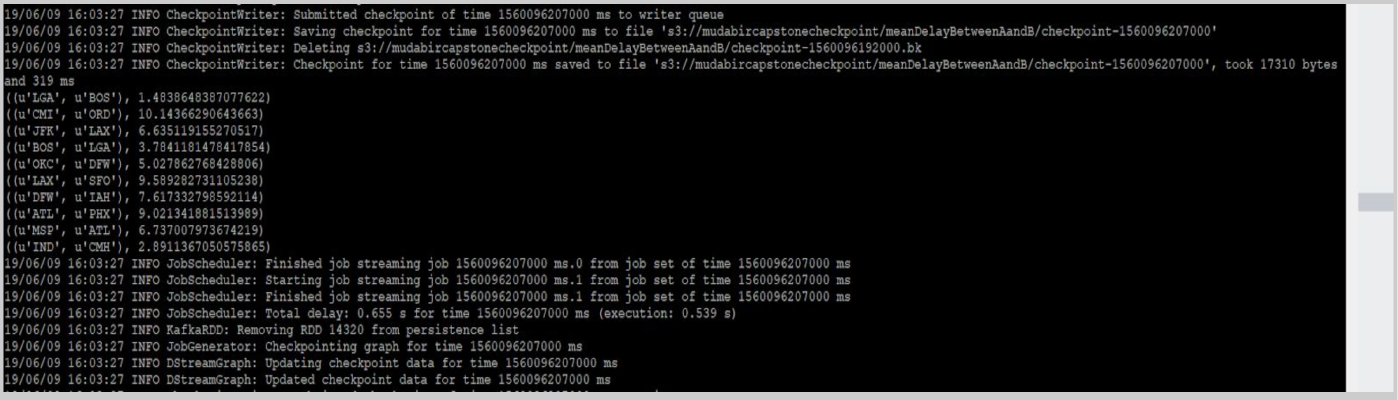
Source:

g2_4.py

Result:

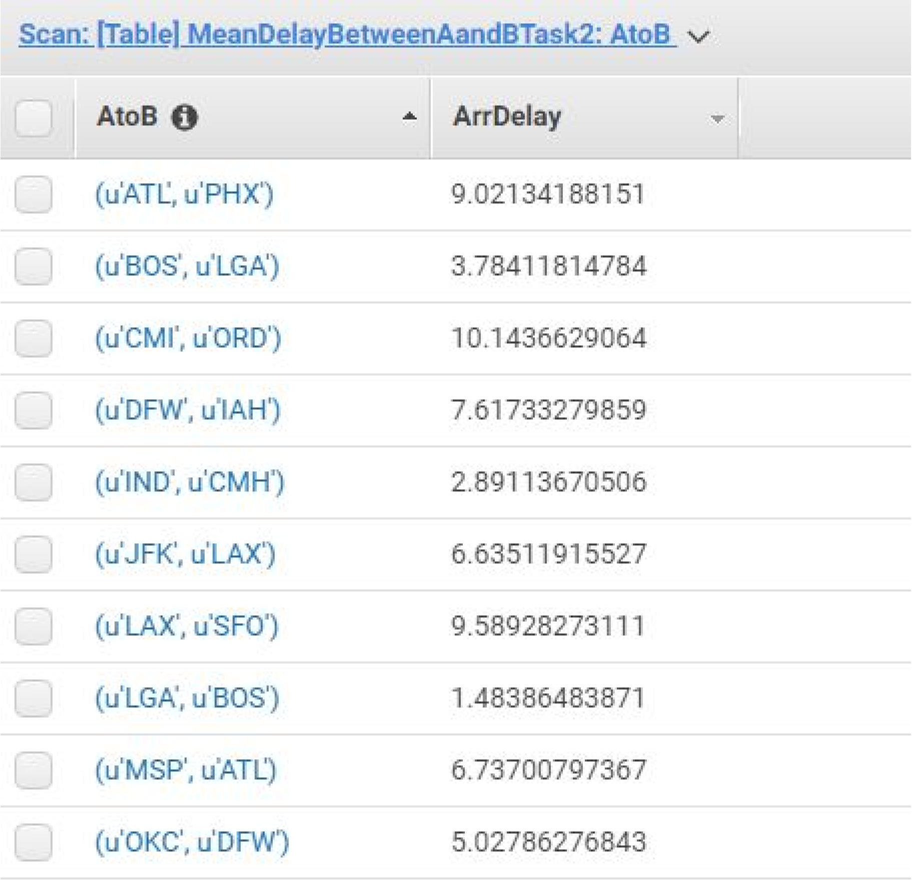
Format: { (<origin>, <dest>) ,<delay value> }

```
((u'LGA', u'BOS'), 1.4838648387077622)
((u'CMI', u'ORD'), 10.14366290643663)
((u'JFK', u'LAX'), 6.635119155270517)
((u'BOS', u'LGA'), 3.7841181478417854)
((u'OKC', u'DFW'), 5.027862768428806)
((u'LAX', u'SFO'), 9.589282731105238)
((u'ATL', u'PHX'), 9.021341881513989)
((u'DFW', u'IAH'), 7.617332798592114)
((u'MSP', u'ATL'), 6.737007973674219)
((u'IND', u'CMH'), 2.8911367050575865)
```



```
19/06/09 16:03:27 INFO CheckpointWriter: Submitted checkpoint of time 1560096207000 ms to writer queue
19/06/09 16:03:27 INFO CheckpointWriter: Saving checkpoint for time 1560096207000 ms to file 's3://mudabircapstonecheckpoint/meanDelayBetweenAandB/checkpoint-1560096207000'
19/06/09 16:03:27 INFO CheckpointWriter: Deleting s3://mudabircapstonecheckpoint/meanDelayBetweenAandB/checkpoint-1560096192000.bk
19/06/09 16:03:27 INFO CheckpointWriter: Checkpoint for time 1560096207000 ms saved to file 's3://mudabircapstonecheckpoint/meanDelayBetweenAandB/checkpoint-1560096207000', took 17310 bytes
and 319 ms
((u'LGA', u'BOS'), 1.4838648387077622)
((u'CMI', u'ORD'), 10.14366290643663)
((u'JFK', u'LAX'), 6.635119155270517)
((u'BOS', u'LGA'), 3.7841181478417854)
((u'OKC', u'DFW'), 5.027862768428806)
((u'LAX', u'SFO'), 9.589282731105238)
((u'DFW', u'IAH'), 7.617332798592114)
((u'ATL', u'PHX'), 9.021341881513989)
((u'MSP', u'ATL'), 6.737007973674219)
((u'IND', u'CMH'), 2.8911367050575865)
19/06/09 16:03:27 INFO JobScheduler: Finished job streaming job 1560096207000 ms.0 from job set of time 1560096207000 ms
19/06/09 16:03:27 INFO JobScheduler: Starting job streaming job 1560096207000 ms.1 from job set of time 1560096207000 ms
19/06/09 16:03:27 INFO JobScheduler: Finished job streaming job 1560096207000 ms.1 from job set of time 1560096207000 ms
19/06/09 16:03:27 INFO JobScheduler: Total delay: 0.655 s for time 1560096207000 ms (execution: 0.539 s)
19/06/09 16:03:27 INFO KafkaRDD: Removing RDD 14320 from persistence list
19/06/09 16:03:27 INFO JobGenerator: Checkpointing graph for time 1560096207000 ms
19/06/09 16:03:27 INFO DStreamGraph: Updating checkpoint data for time 1560096207000 ms
19/06/09 16:03:27 INFO DStreamGraph: Updated checkpoint data for time 1560096207000 ms
```

Arrival delay of some origin-dest pairs from dynamoDB



Scan: [Table] MeanDelayBetweenAandBTask2: AtoB

| □ | AtoB | ArrDelay |
|---|---|---|
| □ | (u'ATL', u'PHX') | 9.02134188151 |
| □ | (u'BOS', u'LGA') | 3.78411814784 |
| □ | (u'CMI', u'ORD') | 10.1436629064 |
| □ | (u'DFW', u'IAH') | 7.61733279859 |
| □ | (u'IND', u'CMH') | 2.89113670506 |
| □ | (u'JFK', u'LAX') | 6.63511915527 |
| □ | (u'LAX', u'SFO') | 9.58928273111 |
| □ | (u'LGA', u'BOS') | 1.48386483871 |
| □ | (u'MSP', u'ATL') | 6.73700797367 |
| □ | (u'OKC', u'DFW') | 5.02786276843 |

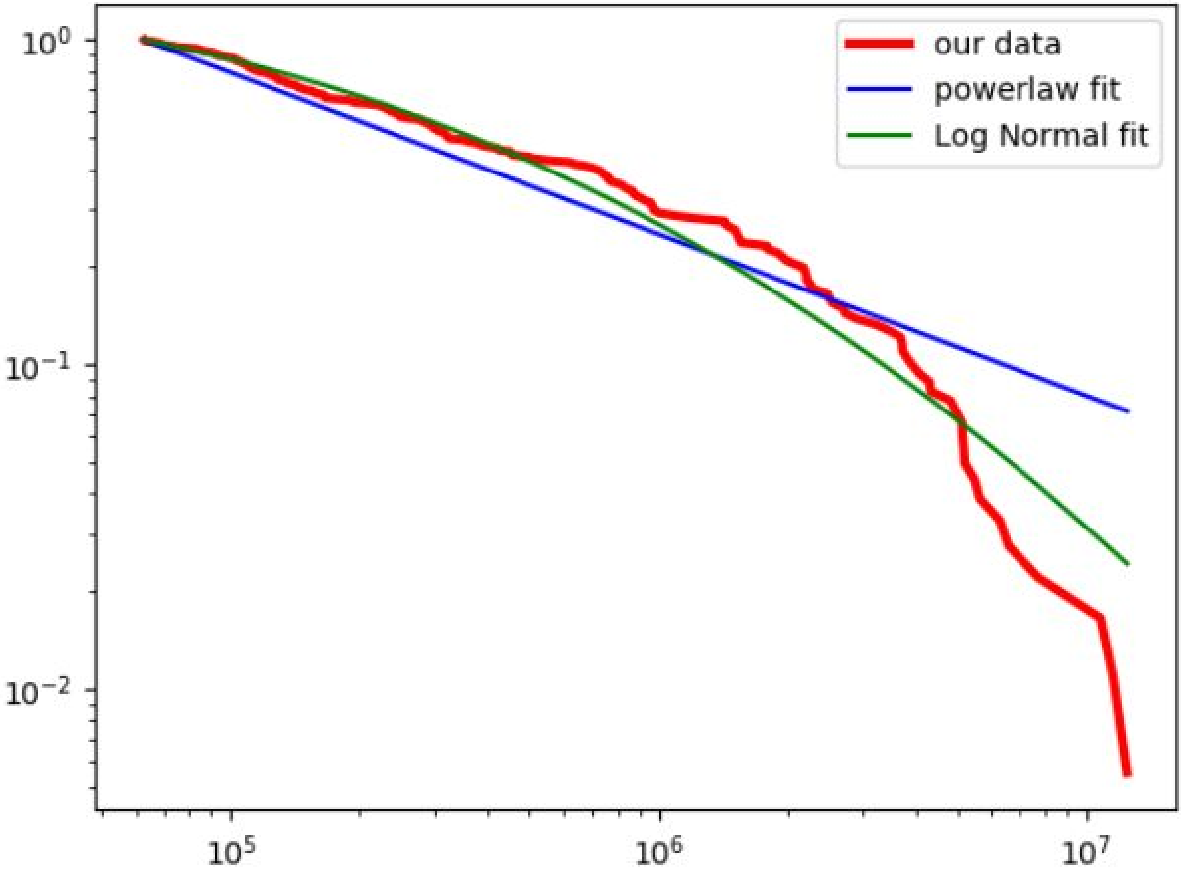
**Problem 3.1**
**(copy pasting my results from Task1 for this problem)**
Strategy:

1) Collected ranking of all airports into a file using Spark
2) Used powerlaw library to plot the CCDF of the power law distribution and the lognormal distribution of the data collected in 1)

Source:
g3_1.py
pythonplot.py

Result:
g3_1.log (Ranking of airports with their count)

Our data resembles the log Normal fit better than the powerlaw fit. We can conclude that the popularity of airports doesn't follow Zipf distribution



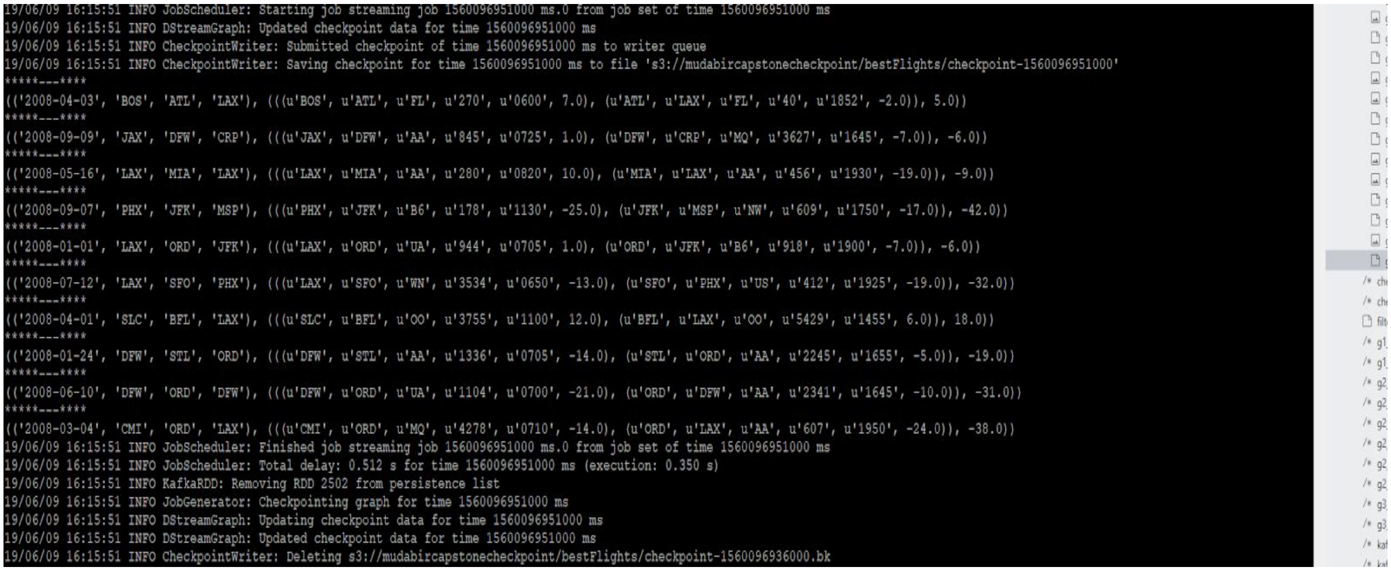
**Problem 3.2**
Strategy:

1. Filter non cancelled flights

2. Create RDD for all flights which fly before noon and dest as Y. key is (date, Y) -> (flight info)
3. Create RDD for all flights which fly after noon with origin as Y and departure date subtracted by 2 days
4. Join is done on (date,Y) as key. This gives all flights landing in Y before noon and all flights departing from Y two days later
5. For X->Y->Z route, filter the flight combo with minimum arrival delay
6. Write data to DynamoDB. Since a huge set of data was to be written, increased the write capacity of DynamoDB to 1500 to transfer the data faster.

Source:
g3_2.py

Result:



```
19/06/09 16:15:51 INFO JobScheduler: Starting job streaming job 1560096951000 ms.0 from job set of time 1560096951000 ms
19/06/09 16:15:51 INFO DStreamGraph: Updated checkpoint data for time 1560096951000 ms
19/06/09 16:15:51 INFO CheckpointWriter: Submitted checkpoint of time 1560096951000 ms to writer queue
19/06/09 16:15:51 INFO CheckpointWriter: Saving checkpoint for time 1560096951000 ms to file 's3://mudabircapstonecheckpoint/bestFlights/checkpoint-1560096951000'
*****___****
(('2008-04-03', 'BOS', 'ATL', 'LAX'), (((u'BOS', u'ATL', u'FL', u'270', u'0600', 7.0), (u'ATL', u'LAX', u'FL', u'40', u'1852', -2.0)), 5.0))
*****___****
(('2008-09-09', 'JAX', 'DFW', 'CRP'), (((u'JAX', u'DFW', u'AA', u'845', u'0725', 1.0), (u'DFW', u'CRP', u'MQ', u'3627', u'1645', -7.0)), -6.0))
*****___****
(('2008-05-16', 'LAX', 'MIA', 'LAX'), (((u'LAX', u'MIA', u'AA', u'280', u'0820', 10.0), (u'MIA', u'LAX', u'AA', u'456', u'1930', -19.0)), -9.0))
*****___****
(('2008-09-07', 'PHX', 'JFK', 'MSP'), (((u'PHX', u'JFK', u'B6', u'178', u'1130', -25.0), (u'JFK', u'MSP', u'NW', u'609', u'1750', -17.0)), -42.0))
*****___****
(('2008-01-01', 'LAX', 'ORD', 'JFK'), (((u'LAX', u'ORD', u'UA', u'944', u'0705', 1.0), (u'ORD', u'JFK', u'B6', u'918', u'1900', -7.0)), -6.0))
*****___****
(('2008-07-12', 'LAX', 'SFO', 'PHX'), (((u'LAX', u'SFO', u'WN', u'3534', u'0650', -13.0), (u'SFO', u'PHX', u'US', u'412', u'1925', -19.0)), -32.0))
*****___****
(('2008-04-01', 'SLC', 'BFL', 'LAX'), (((u'SLC', u'BFL', u'OO', u'3755', u'1100', 12.0), (u'BFL', u'LAX', u'OO', u'5429', u'1455', 6.0)), 18.0))
*****___****
(('2008-01-24', 'DFW', 'STL', 'ORD'), (((u'DFW', u'STL', u'AA', u'1336', u'0705', -14.0), (u'STL', u'ORD', u'AA', u'2245', u'1655', -5.0)), -19.0))
*****___****
(('2008-06-10', 'DFW', 'ORD', 'DFW'), (((u'DFW', u'ORD', u'UA', u'1104', u'0700', -21.0), (u'ORD', u'DFW', u'AA', u'2341', u'1645', -10.0)), -31.0))
*****___****
(('2008-03-04', 'CMI', 'ORD', 'LAX'), (((u'CMI', u'ORD', u'MQ', u'4278', u'0710', -14.0), (u'ORD', u'LAX', u'AA', u'607', u'1950', -24.0)), -38.0))
19/06/09 16:15:51 INFO JobScheduler: Finished job streaming job 1560096951000 ms.0 from job set of time 1560096951000 ms
19/06/09 16:15:51 INFO JobScheduler: Total delay: 0.512 s for time 1560096951000 ms (execution: 0.350 s)
19/06/09 16:15:51 INFO KafkaRDD: Removing RDD 2502 from persistence list
19/06/09 16:15:51 INFO JobGenerator: Checkpointing graph for time 1560096951000 ms
19/06/09 16:15:51 INFO DStreamGraph: Updating checkpoint data for time 1560096951000 ms
19/06/09 16:15:51 INFO DStreamGraph: Updated checkpoint data for time 1560096951000 ms
19/06/09 16:15:51 INFO CheckpointWriter: Deleting s3://mudabircapstonecheckpoint/bestFlights/checkpoint-1560096936000.bk
```

Format: ((startdate,X,Y,Z), ((XYdetails,YZdetails),totalArrivalDelay)

*****___****

(('2008-04-03', 'BOS', 'ATL', 'LAX'), (((u'BOS', u'ATL', u'FL', u'270', u'0600', 7.0), (u'ATL', u'LAX', u'FL', u'40', u'1852', -2.0)), 5.0))

*****___****

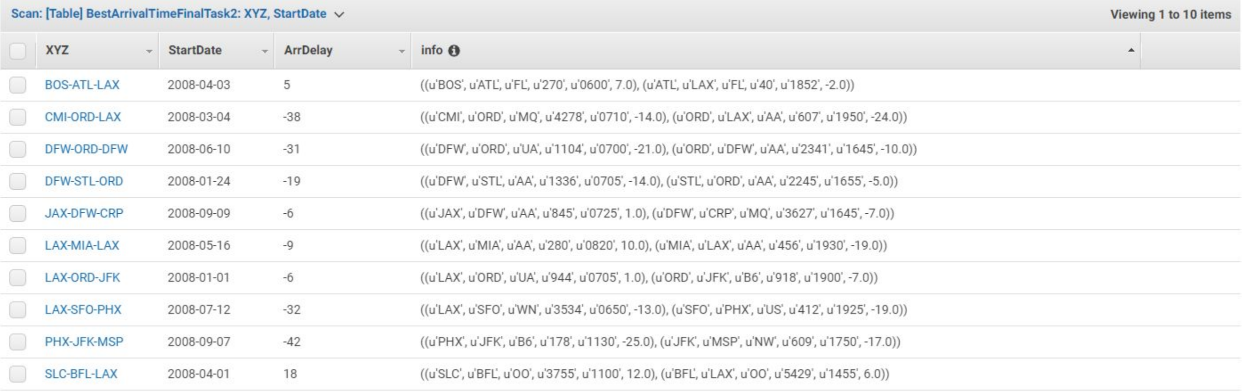
(('2008-09-09', 'JAX', 'DFW', 'CRP'), (((u'JAX', u'DFW', u'AA', u'845', u'0725', 1.0), (u'DFW', u'CRP', u'MQ', u'3627', u'1645', -7.0)), -6.0))

*****___****

(('2008-05-16', 'LAX', 'MIA', 'LAX'), (((u'LAX', u'MIA', u'AA', u'280', u'0820', 10.0), (u'MIA', u'LAX', u'AA', u'456', u'1930', -19.0)), -9.0))

```
*****___****
(('2008-09-07', 'PHX', 'JFK', 'MSP'), (((u'PHX', u'JFK', u'B6', u'178', u'1130', -25.0), (u'JFK', u'MSP', u'NW', u'609', u'1750', -17.0)), -42.0))
*****___****
(('2008-01-01', 'LAX', 'ORD', 'JFK'), (((u'LAX', u'ORD', u'UA', u'944', u'0705', 1.0), (u'ORD', u'JFK', u'B6', u'918', u'1900', -7.0)), -6.0))
*****___****
(('2008-07-12', 'LAX', 'SFO', 'PHX'), (((u'LAX', u'SFO', u'WN', u'3534', u'0650', -13.0), (u'SFO', u'PHX', u'US', u'412', u'1925', -19.0)), -32.0))
*****___****
(('2008-04-01', 'SLC', 'BFL', 'LAX'), (((u'SLC', u'BFL', u'OO', u'3755', u'1100', 12.0), (u'BFL', u'LAX', u'OO', u'5429', u'1455', 6.0)), 18.0))
*****___****
(('2008-01-24', 'DFW', 'STL', 'ORD'), (((u'DFW', u'STL', u'AA', u'1336', u'0705', -14.0), (u'STL', u'ORD', u'AA', u'2245', u'1655', -5.0)), -19.0))
*****___****
(('2008-06-10', 'DFW', 'ORD', 'DFW'), (((u'DFW', u'ORD', u'UA', u'1104', u'0700', -21.0), (u'ORD', u'DFW', u'AA', u'2341', u'1645', -10.0)), -31.0))
*****___****
(('2008-03-04', 'CMI', 'ORD', 'LAX'), (((u'CMI', u'ORD', u'MQ', u'4278', u'0710', -14.0), (u'ORD', u'LAX', u'AA', u'607', u'1950', -24.0)), -38.0))
```

## Snapshot from DynamoDB



Scan: [Table] BestArrivalTimeFinalTask2: XYZ, StartDate

Viewing 1 to 10 items

| □ | XYZ | StartDate | ArrDelay | info |
|---|---|---|---|---|
| □ | BOS-ATL-LAX | 2008-04-03 | 5 | ((u'BOS', u'ATL', u'FL', u'270', u'0600', 7.0), (u'ATL', u'LAX', u'FL', u'40', u'1852', -2.0)) |
| □ | CMI-ORD-LAX | 2008-03-04 | -38 | ((u'CMI', u'ORD', u'MQ', u'4278', u'0710', -14.0), (u'ORD', u'LAX', u'AA', u'607', u'1950', -24.0)) |
| □ | DFW-ORD-DFW | 2008-06-10 | -31 | ((u'DFW', u'ORD', u'UA', u'1104', u'0700', -21.0), (u'ORD', u'DFW', u'AA', u'2341', u'1645', -10.0)) |
| □ | DFW-STL-ORD | 2008-01-24 | -19 | ((u'DFW', u'STL', u'AA', u'1336', u'0705', -14.0), (u'STL', u'ORD', u'AA', u'2245', u'1655', -5.0)) |
| □ | JAX-DFW-CRP | 2008-09-09 | -6 | ((u'JAX', u'DFW', u'AA', u'845', u'0725', 1.0), (u'DFW', u'CRP', u'MQ', u'3627', u'1645', -7.0)) |
| □ | LAX-MIA-LAX | 2008-05-16 | -9 | ((u'LAX', u'MIA', u'AA', u'280', u'0820', 10.0), (u'MIA', u'LAX', u'AA', u'456', u'1930', -19.0)) |
| □ | LAX-ORD-JFK | 2008-01-01 | -6 | ((u'LAX', u'ORD', u'UA', u'944', u'0705', 1.0), (u'ORD', u'JFK', u'B6', u'918', u'1900', -7.0)) |
| □ | LAX-SFO-PHX | 2008-07-12 | -32 | ((u'LAX', u'SFO', u'WN', u'3534', u'0650', -13.0), (u'SFO', u'PHX', u'US', u'412', u'1925', -19.0)) |
| □ | PHX-JFK-MSP | 2008-09-07 | -42 | ((u'PHX', u'JFK', u'B6', u'178', u'1130', -25.0), (u'JFK', u'MSP', u'NW', u'609', u'1750', -17.0)) |
| □ | SLC-BFL-LAX | 2008-04-01 | 18 | ((u'SLC', u'BFL', u'OO', u'3755', u'1100', 12.0), (u'BFL', u'LAX', u'OO', u'5429', u'1455', 6.0)) |

## Optimizations

- Filtered only necessary fields from all the csv reducing the total data size to 4.5GB from 15 GB
- In the kafka producer side, increased batch size and linger_ms to dispatch larger amounts of data at a time to the kafka cluster without waiting or ack. Also make ack= True to make sure no data is dropped/lost while writing to kafka topic

- Made replication factor =3 to make the kafka cluster reliable. Kept the number of partitions to 1. Increasing the number of partitions seemed to make the read from consumer side slower

**Streaming versus Batch processing**

- For the given data, batch processing seemed a better option. Since we had to compute top 10 aiports/carriers etc based on entire data, doing stream processing as the data arrives seems like an overkill.
- However, stream processing is useful in analysing live how the parameters are changing over time.

**Comparing Stacks**

Since I used Spark for task1 and Spark Streaming for task2, the stacks were more of less similar. Using hadoop Map-reduce for task 1 would have made the process very slow.

-------------------------------------------