Cloud Compute Project Task 1 Report

Chenghao Shi Oct, 2017

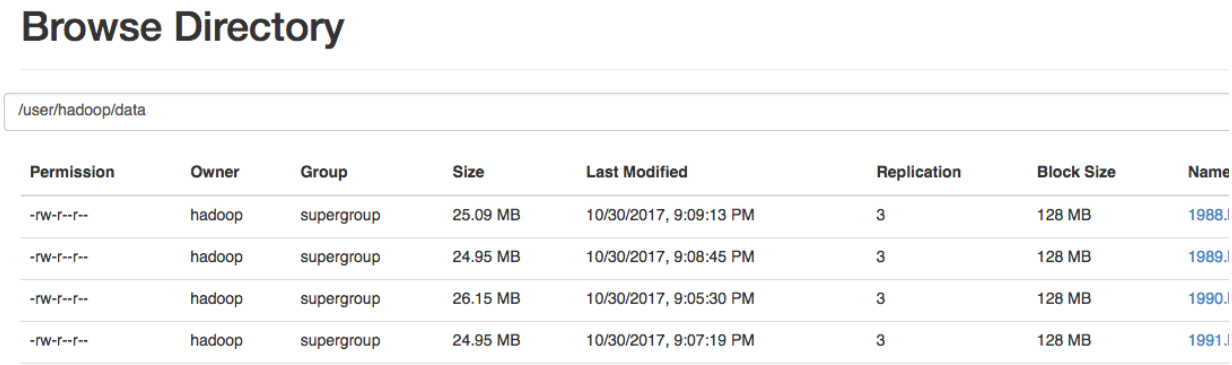
System setup

Name	Instance ID	Instance Type	Availability Zone	Instance State
master	i-066ba3748e09c86f1	c4.xlarge	us-east-1a	running
slave1	i-00e29941bd53eb6…	c4.large	us-east-1a	running
slave2	i-0308f31820f35e6a8	c4.large	us-east-1a	running
slave3	i-099544425424d18…	c4.large	us-east-1a	running

1. Amazon linux AMI is used (with aws CLI pre-installed for querying dynamo db)
2. Create hadoop users on all hosts, grant sudo access and password-less ssh login between master and slaves
3. Enable PubkeyAuthentication and PasswordAuthentication on all hosts and restart sshd services
4. Run aws configure on all hosts
5. Install hadoop-2.7.4 on master, configure slaves, hdfs, mapred, yarn related configuration files, format namenode then scp Hadoop folder to all slaves
6. Start yarn and dfs
7. Attach transportation snapshot volume to master and mount to /data folder
8. Create empty dynamo db tables for Group 2 and 3's results

Data cleanup and import

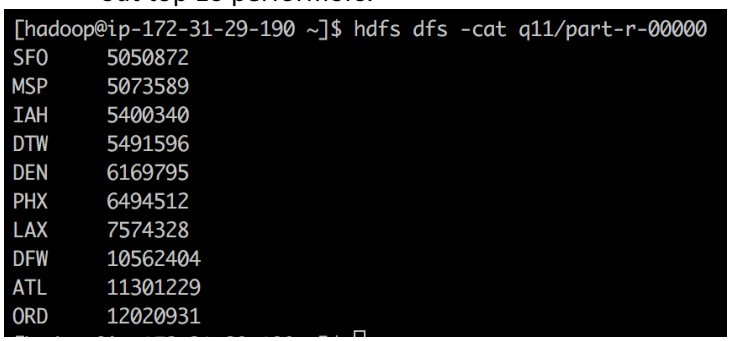
1. Examine /data folder, only aviation/aviation-ontime folder is needed for task 1
2. Unzip and open any CSV file, determined only FlightDate, UniqueCarrier, FlightNum, Origin, Dest, DepTime, DepDelayMins, ArrTime, ArrDelaysMins fields are needed.
3. Wrote mapreduce job to unzip each year's ontime data folder, filter needed fields to generate new CSV file, re-compress and then upload to HDFS



(Pic : Part of the uploaded HDFS directory)

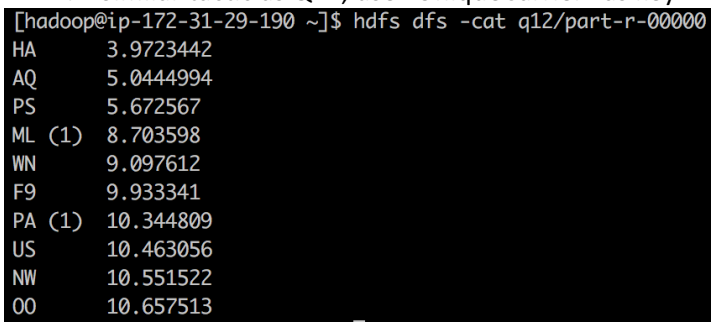
Q11 method and result

1. Simple MapReduce job similar with wordcount example included in every Hadoop package
2. One optimization used here is utilizing reducer class also as combiner, so that to relieve the network traffic between datanodes.
3. Concatenate 2 Mapreduce job, one for count the popularity and the second one for ranking out top 10 performors.



Q12 method and result

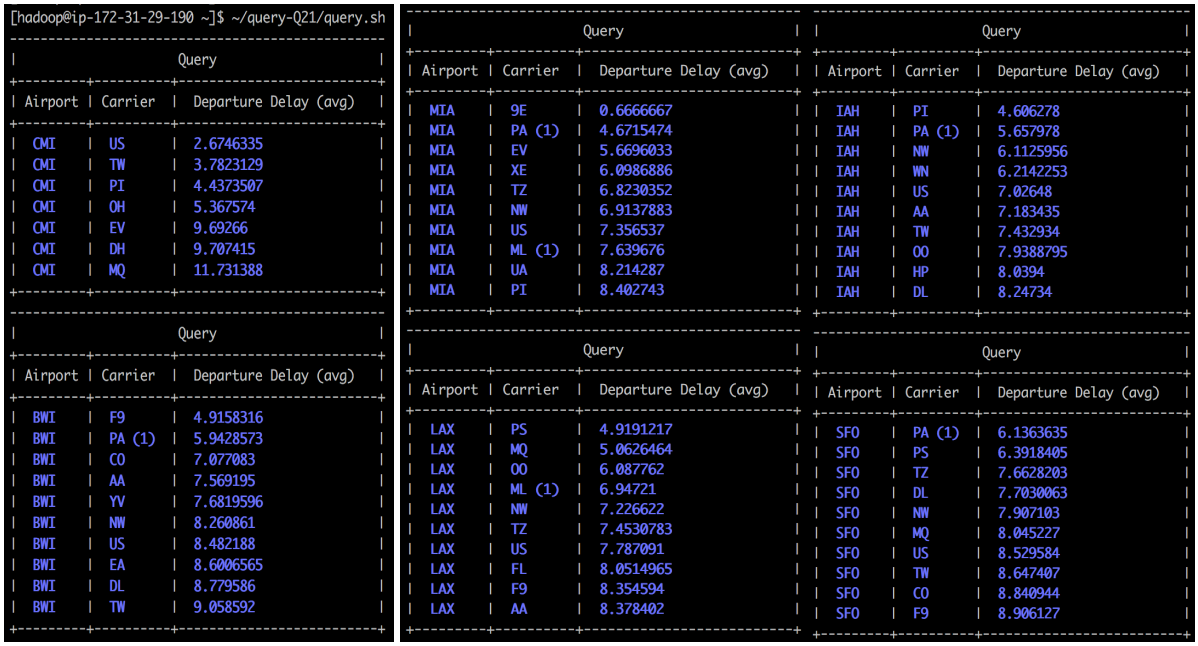
1. Simliar tactic as Q11, use “UniqueCarrier” as key



Q21, Q22, Q23 method

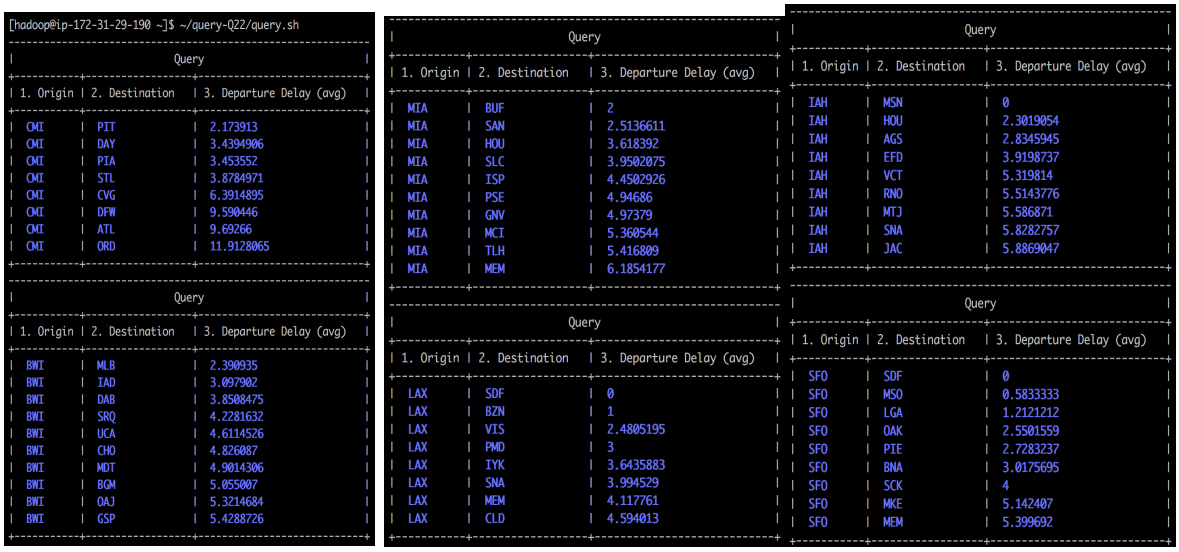
1. Multi-phases MapReduce jobs are developed and concatenated as a workflow
2. Airport+Carrier are connected together as the key for first MapReduce job to count the performance data
3. Optimization used by Second MapReduce job is using MapReduce framework’s own shuffle and sort feature so that our job itself only needs to count the top 10 performers
4. AWS dynamodb java SDK is utilized for saving data into pre-created tables

Q21 result



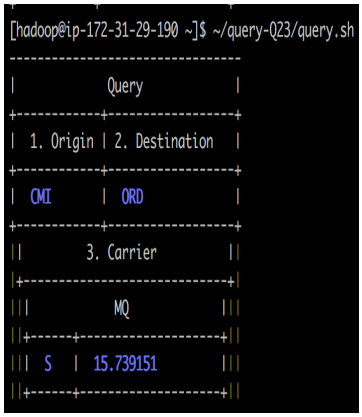
```
[hadoop@ip-172-31-29-190 ~]$ ~/query-Q21/query.sh
Query
| Airport | Carrier | Departure Delay (avg) |
| CMI | US | 2.6746335 |
| CMI | TW | 3.7823129 |
| CMI | PI | 4.4373507 |
| CMI | OH | 5.367574 |
| CMI | EV | 9.69266 |
| CMI | DH | 9.707415 |
| CMI | MQ | 11.731388 |
Query
| Airport | Carrier | Departure Delay (avg) |
| BWI | F9 | 4.9158316 |
| BWI | PA (1) | 5.9428573 |
| BWI | CO | 7.077083 |
| BWI | AA | 7.569195 |
| BWI | YV | 7.6819596 |
| BWI | NW | 8.260861 |
| BWI | US | 8.482188 |
| BWI | EA | 8.6006565 |
| BWI | DL | 8.779586 |
| BWI | TW | 9.058592 |
Query
| Airport | Carrier | Departure Delay (avg) |
| MIA | 9E | 0.6666667 |
| MIA | PA (1) | 4.6715474 |
| MIA | EV | 5.6696033 |
| MIA | XE | 6.0986886 |
| MIA | TZ | 6.8230352 |
| MIA | NW | 6.9137883 |
| MIA | US | 7.356537 |
| MIA | ML (1) | 7.639676 |
| MIA | UA | 8.214287 |
| MIA | PI | 8.402743 |
Query
| Airport | Carrier | Departure Delay (avg) |
| LAX | PS | 4.9191217 |
| LAX | MQ | 5.0626464 |
| LAX | OO | 6.087762 |
| LAX | ML (1) | 6.94721 |
| LAX | NW | 7.226622 |
| LAX | TZ | 7.4530783 |
| LAX | US | 7.787091 |
| LAX | FL | 8.0514965 |
| LAX | F9 | 8.354594 |
| LAX | AA | 8.378402 |
Query
| Airport | Carrier | Departure Delay (avg) |
| IAH | PI | 4.606278 |
| IAH | PA (1) | 5.657978 |
| IAH | NW | 6.1125956 |
| IAH | WN | 6.2142253 |
| IAH | US | 7.02648 |
| IAH | AA | 7.183435 |
| IAH | TW | 7.432934 |
| IAH | OO | 7.9388795 |
| IAH | HP | 8.0394 |
| IAH | DL | 8.24734 |
Query
| Airport | Carrier | Departure Delay (avg) |
| SFO | PA (1) | 6.1363635 |
| SFO | PS | 6.3918405 |
| SFO | TZ | 7.6628203 |
| SFO | DL | 7.7030063 |
| SFO | NW | 7.907103 |
| SFO | MQ | 8.045227 |
| SFO | US | 8.529584 |
| SFO | TW | 8.647407 |
| SFO | CO | 8.840944 |
| SFO | F9 | 8.906127 |
```

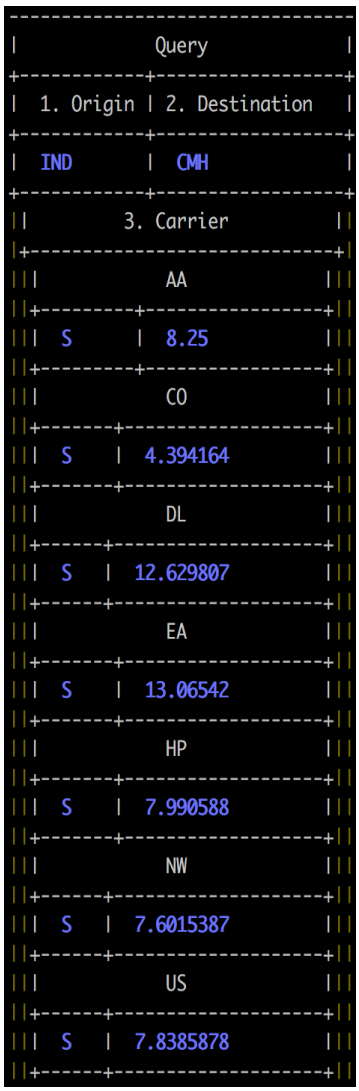
Q22 result

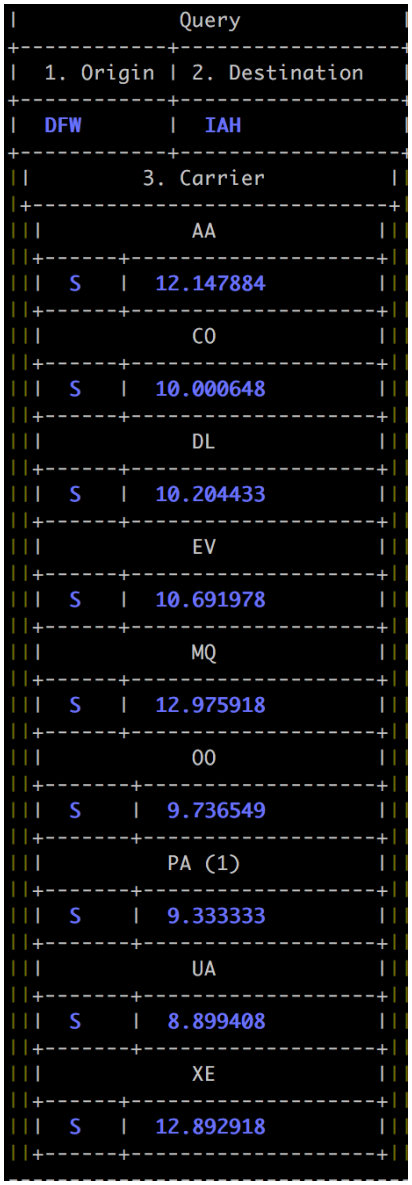


```
[hadoop@ip-172-31-29-190 ~]$ ~/query-Q22/query.sh
Query
| 1. Origin | 2. Destination | 3. Departure Delay (avg) |
| CMI | PIT | 2.173913 |
| CMI | DAY | 3.4394906 |
| CMI | PIA | 3.453552 |
| CMI | STL | 3.8784971 |
| CMI | CVG | 6.3914895 |
| CMI | DFW | 9.590446 |
| CMI | ATL | 9.69266 |
| CMI | ORD | 11.9128065 |
Query
| 1. Origin | 2. Destination | 3. Departure Delay (avg) |
| BWI | MLB | 2.390935 |
| BWI | IAD | 3.097902 |
| BWI | DAB | 3.8508475 |
| BWI | SRQ | 4.2281632 |
| BWI | UCA | 4.6114526 |
| BWI | CHO | 4.826087 |
| BWI | MDT | 4.9014306 |
| BWI | BGM | 5.055007 |
| BWI | OAJ | 5.3214684 |
| BWI | GSP | 5.4288726 |
Query
| 1. Origin | 2. Destination | 3. Departure Delay (avg) |
| MIA | BUF | 2 |
| MIA | SAN | 2.5136611 |
| MIA | HOU | 3.618392 |
| MIA | SLC | 3.9502075 |
| MIA | ISP | 4.4502926 |
| MIA | PSE | 4.94686 |
| MIA | GNV | 4.97379 |
| MIA | MCI | 5.360544 |
| MIA | TLH | 5.416809 |
| MIA | MEM | 6.1854177 |
Query
| 1. Origin | 2. Destination | 3. Departure Delay (avg) |
| LAX | SDF | 0 |
| LAX | BZN | 1 |
| LAX | VIS | 2.4805195 |
| LAX | PMD | 3 |
| LAX | IYK | 3.6435883 |
| LAX | SNA | 3.994529 |
| LAX | MEM | 4.117761 |
| LAX | CLD | 4.594013 |
Query
| 1. Origin | 2. Destination | 3. Departure Delay (avg) |
| IAH | MSN | 0 |
| IAH | HOU | 2.3019054 |
| IAH | AGS | 2.8345945 |
| IAH | EFD | 3.9198737 |
| IAH | VCT | 5.319814 |
| IAH | RNO | 5.5143776 |
| IAH | MTJ | 5.586871 |
| IAH | SNA | 5.8282757 |
| IAH | JAC | 5.8869047 |
Query
| 1. Origin | 2. Destination | 3. Departure Delay (avg) |
| SFO | SDF | 0 |
| SFO | MSO | 0.5833333 |
| SFO | LGA | 1.2121212 |
| SFO | OAK | 2.5501559 |
| SFO | PIE | 2.7283237 |
| SFO | BNA | 3.0175695 |
| SFO | SCK | 4 |
| SFO | MKE | 5.142407 |
| SFO | MEM | 5.399692 |
```

Q23 result


[hadoop@ip-172-31-29-190 ~]$ ~/query-Q23/query.sh
Query
1. Origin | 2. Destination
CMI | ORD
3. Carrier
MQ
S | 15.739151


Query
1. Origin | 2. Destination
IND | CMH
3. Carrier
AA
S | 8.25
CO
S | 4.394164
DL
S | 12.629807
EA
S | 13.06542
HP
S | 7.990588
NW
S | 7.6015387
US
S | 7.8385878


Query
1. Origin | 2. Destination
DFW | IAH
3. Carrier
AA
S | 12.147884
CO
S | 10.000648
DL
S | 10.204433
EV
S | 10.691978
MQ
S | 12.975918
OO
S | 9.736549
PA (1)
S | 9.333333
UA
S | 8.899408
XE
S | 12.892918

1. Origin	2. Destination
LAX	SFO

3. Carrier	
AA	
S	12.465499
CO	
S	14.0017395
DL	
S	13.4838505
EV	
S	13.398714
F9	
S	6.96531
MQ	
S	10.933456
NW	
S	12.790287
PS	
S	5.830403
TZ	
S	6.2380953
US	
S	10.821993

Query

1. Origin	2. Destination
JFK	LAX

3. Carrier	
AA	
S	15.044821
DL	
S	16.631231
HP	
S	14.865142
PA (1)	
S	17.093708
TW	
S	18.287762
UA	
S	11.469386

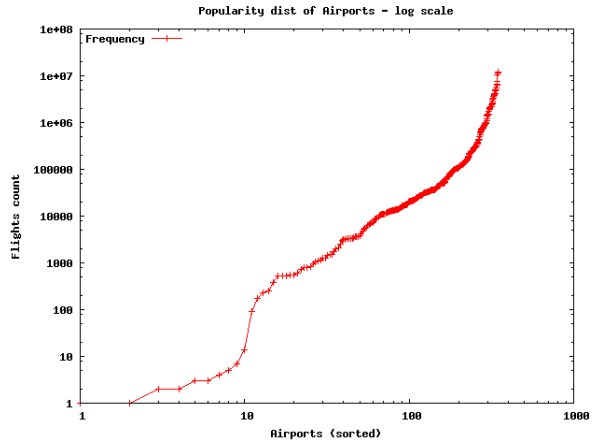
Query

1. Origin	2. Destination
ATL	PHX

3. Carrier	
DL	
S	13.867261
EA	
S	14.008674
FL	
S	12.61
HP	
S	13.367141
US	
S	12.687395

Q31 method and result

1. Use Same code as Q11 but remove the top 10 ranking job
2. Use output <airport, popularity> key-value pairs for plotting
3. From https://en.wikipedia.org/wiki/Zipf%27s_law, log-log plot shall be straight-line
4. Our data's log-log plot is not, so it does not follow Zipf distribution but more like a log-normal distribution instead



Q32 method and result

1. 2 MapReduce jobs are implemented
2. First one to find out the best performer as part of the requirement (least delay)
3. Second job divides the best performers into 2 sub-set, one before 12pm and one after;
4. Second job matches the 2 sub-set to find out the best 2-leg options for given date and airports
5. The resulting datasets are tremendous, around 59 million lines, it does take a while to save all into dynamo db and default “5 reads, 5 writes” setting is not adequate, “writes” shall be increased to 10000 at least