CS6240 Final Project

Purushottam Jha
Rushil Patel
Tushar Bhandari
Nishant Agarwal

DATA

Name: Airline On-Time Performance and Causes of Flight Delays

Size: 4.34GB with 4336074861 records.

Fields: 55 Columns (such as FlightNo, Date, DepDelayMins, etc.)

View of sample dataset: (with few columns)

flightdate	flightnum	origin	origincityname	dest	destcityname	deptime	arrtime	arrdelayminutes
2007-10-02	415	IND	Indianapolis, IN	MDW	Chicago, IL	1102	1100	30.00
2007-10-02	917	IND	Indianapolis, IN	MDW	Chicago, IL	0728	0712	0.00
2007-10-02	1022	IND	Indianapolis, IN	MDW	Chicago, IL	1855	1842	0.00
2007-10-02	2081	IND	Indianapolis, IN	MDW	Chicago, IL	1530	1522	0.00
2007-10-02	1033	IND	Indianapolis, IN	PHX	Phoenix, AZ	1500	1544	0.00

Link to the dataset: https://explore.data.gov/Transportation/Airline-On-Time-Performance-and-Causes-of-Flight-D/ar4r-an9z

TASKS

Task 1	Plain MapReduce	Finding Hubs and Spokes.
Task 2	HBase	a) On Time Arrival Performance of Airlines b) Average delay at each airport.
Task 3	Plain MapReduce	Calculate PageRank of each Airport.
Task 4	Hive and Pig Latin	Three legged round trip flights from Boston.

TASK 1 : Hubs and Spokes

- Understanding HITS Algorithm
  - Iterative - 30 iterations [5- Small Machines : 1hr 8mins].
  - Each Node [Airport] has HubValue and SpokeValue.
  - Normalization is done at each airport as follows
    - $NewHubVal = HubVal/\sqrt{\sum(HubVals\ at\ Each\ Airport)^2}$
- 2 MR Jobs:
  - Convert file Data into Node Structure.
  - Perform HITS Algorithm iteratively.

HITS in MapReduce

Job 2:

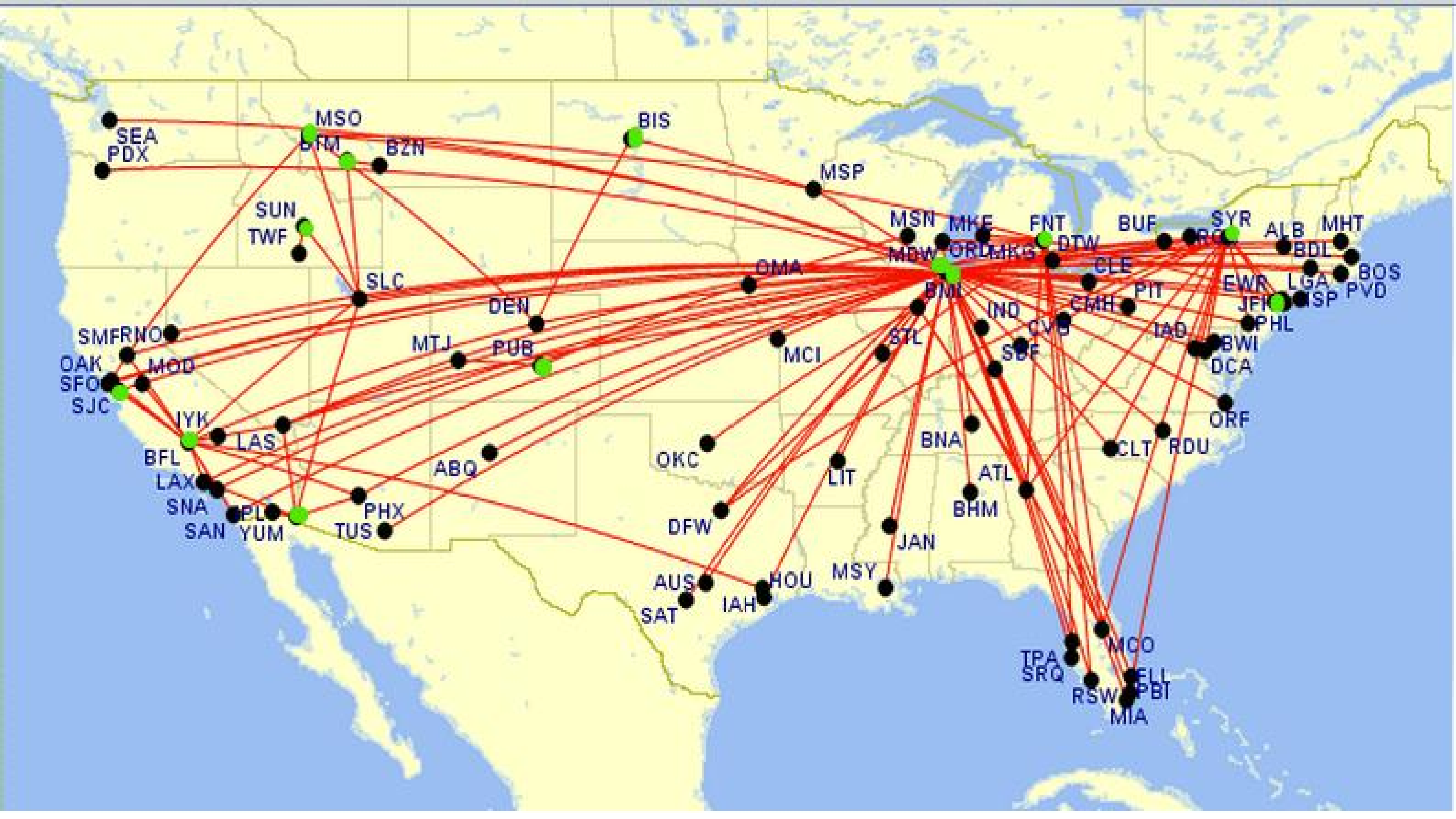
MAPPER:

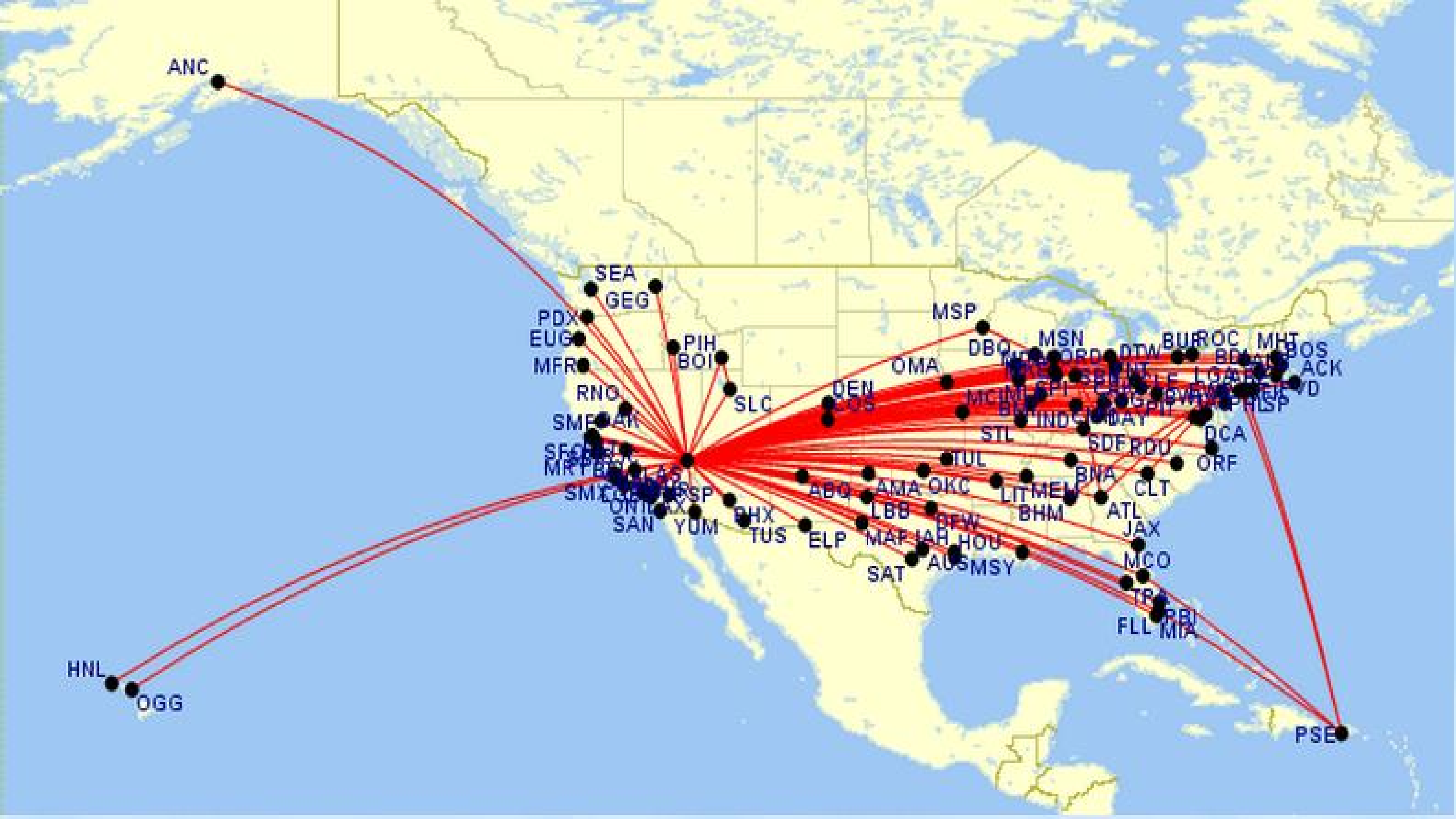
```
map(Key k, Value v) // Here Key: Object Value: Text
{       Node n = parse(v)
        emit(n.nid, N )
        for(Node p in n.getInList()){
                n.setIsIn(“YES”)
                emit(p.nid,n)
        }
        for(Nope p in n.getOutList()){
                n.setIsIn(“NO”)
                emit(p.nid,n)
        }
}
```

REDUCER

```
setup()
{
        map = new Map(); // initialise hashmap
}
```

```
reduce(Key k, Values[v1,v2,v3,....  ]) //  Key : Text Values:[ Node n1. Node n2,......]
{
        Node M = null
        HubVal = 0
        SpokeVal = 0
        for(Node n in values)
        {
                if(isNode(N))
                        M=n
                else
                        if(n.isInList())
                                SpokeVal +=n.getHubVal()
                        else
                                HubVal += n.getSpokeVal()
        }
        map.put(n)
}
cleanup()
{       for(val in  map)
                HubNorm += Math.pow(val.Hubval,2)
                SpokeNorm +=Math.pow(val.SpokeVal,2)
        for(val in map)
                val.HubVal = val.HubVal/HubNorm
                val.SpokeVal = val.getSpokeVal/SpokeNorm
                emit(val,””)
}
```


MSO
BIS
SEA
PDX
BTM
BZN
MSP
SUN
TWF
MSN
MKE
FNT
BUF
SYR
ALB
MHT
MDW
ORD
MKG
DTW
ROC
BDL
BOS
OMA
CLE
LGA
PVD
SLC
BMI
EWR
PIT
ISP
JFK
DEN
CMH
IND
CVG
IAD
PHL
SMF
RNO
MTJ
PUB
STL
MCI
SDF
BWI
OAK
MOD
DCA
SFO
SJC
ORF
IYK
LAS
BNA
CLT
RDU
BFL
OKC
ABQ
LIT
ATL
LAX
SNA
IPL
PHX
BHM
DFW
SAN
YUM
TUS
JAN
MSY
AUS
HOU
IAH
SAT
MCO
TPA
SRQ
FLL
PBI
RSW
MIA


ANC
SEA
GEG
PDX
EUG
MFR
PIH
BOI
SLC
RNO
SMF
OAK
SFO
MRY
SMX
ONT
SAN
LAS
PSP
YUM
PHX
TUS
ELP
ABQ
AMA
LBB
MAF
SAT
AUS
IAH
HOU
MSY
DFW
OKC
TUL
LIT
MEM
BHM
ATL
BNA
STL
MCI
OMA
DEN
COS
MSP
DBQ
MSN
ORD
SPI
IND
DTW
DAY
SDF
BUF
ROC
MHT
BOS
BDL
ACK
PVD
ISP
DCA
RDU
ORF
CLT
JAX
MCO
TPA
PBI
FLL
MIA
HNL
OGG
PSE

TASK 2 : HBase

1. On time arrival performance of airlines.

MR Job 1:

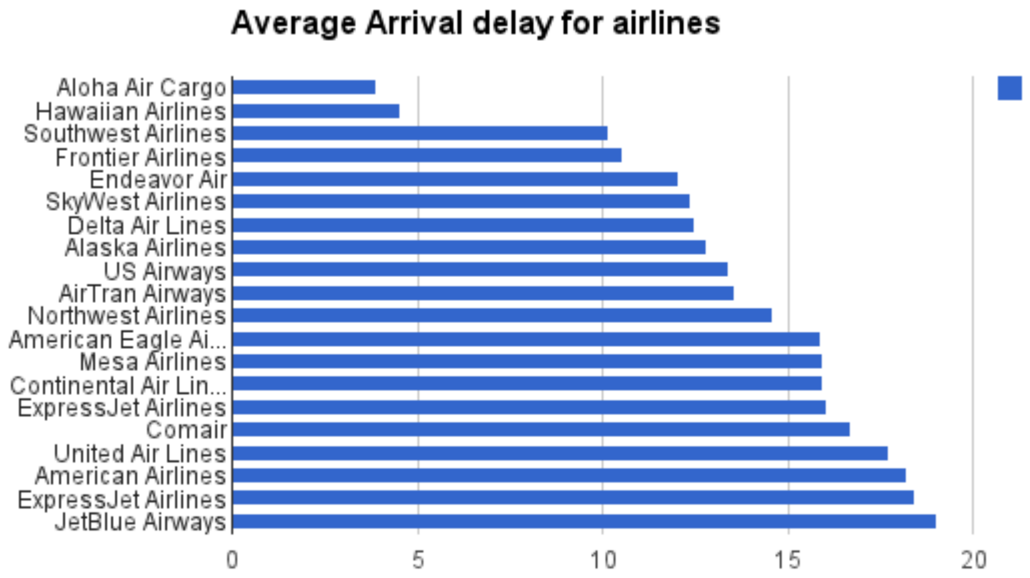
```
// key = line offset , value = each record as String
map(key, value) {
          airlineId = value.getAirlineID();
         arrDelayMins = value.getArrDelayMinutes();
         emit(airlineId, arrDelayMins);
 }

reduce(key, List[values]) {
          sum, total = 0,
          for delayValue in values:
                  total ++;
                  sum += delayValue;
         //compute average and emit in reducer
         averageDelayMins =(sum / total);
         // HBaseConnection is a utility class
         HBaseConnection.addRecord(tablename,
averageDelayMins , "airlineID", "", key.toString());
}
```

MR Job 2: (Map-Only)

```
// key is averageDelay
// value is airlineID
map( key, value) {

         //convert average delay
         // from Bytes to DoubleWritable
          averageDelay = row.get()

         // get airline ID from result
         airlineID = value.getValue(CF, ATTR)

          emit(averageArrivalDelay, airlineID);
 }
```

HBase

1. On-time arrival performance of airlines.



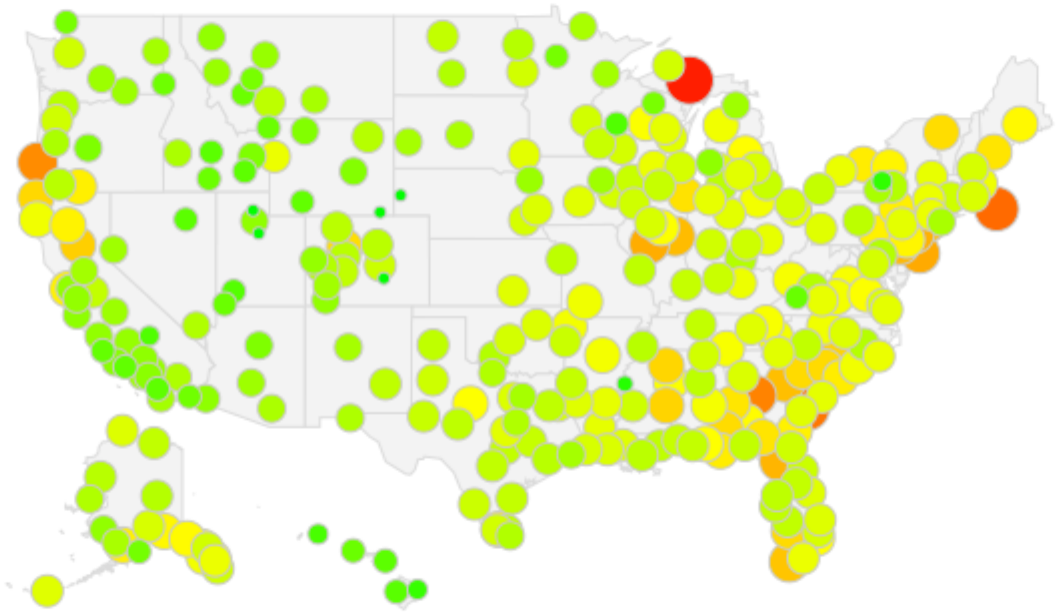
HBase

Performance numbers on EMR

Program	No. of Worker Machines	Machine Type	Time Taken
HBase1 (Task 1)	2	large	8 minutes
HBase 1 (Task 1)	2	large	9 minutes
HBase2 (Task 2)	5	large	4 minutes
HBase2 (Task 2)	5	large	4 minutes

HBase

2. Average delay at each airport



Task 3: Pagerank

- Multiple flights having same source and destination does affect the pagerank value
- However, the data transfer can be minimized by using in-mapper aggregation and/or combiner
- This way if there are 10 flights from BOS -> JFK, instead of emitting (JFK, [BOS,BOS,BOS,BOS……..],

  emit(JFK,[(BOS,10),(MIA,3),(SEA,5)])

Pagerank discrepancy

CLE and LAX switch their position in the order

	Without count	**With count**
BOS	0.03509803921568627	0.03509803921568627
CLE	0.05751257812154845	0.06333333333333332
ORD	0.03509803921568627	0.03509803921568627
LAX	0.06777777777777777	0.03333333333333333

Aggregation Vs No-Aggregation

It took double the resources to achieve the same runtime

	Aggregation	**No-Aggregation**
# of machines	10	20
type	small	small
map output records	163749	26790152
time taken (secs)	295	297

Same data, more workers

More workers can sometimes decrease the performance

Pagerank	10 machines	20 machines
Iteration #1	140	170
Iteration #2	143	174
Iteration #3	141	172
Iteration #4	142	171

TASK 4 : Hive and PigLatin

Shortest three-legged round trip journey.

Hive Code:

```
INSERT OVERWRITE DIRECTORY 's3n://finalproj-puru/hiveoutput'
Select a.origin, a.dest, b.dest, c.dest, a.flightdate, a.DepTime, a.ArrTime, b.DepTime,
b.ArrTime, c.DepTime, c.ArrTime, (a.actualelapsedtime + b.actualelapsedtime + c.
actualelapsedtime + (b.DepTime-a.ArrTime) + (c.DepTime-b.ArrTime)) AS TotalTime from
flight a JOIN flight b on (a.dest = b.origin) JOIN flight c on (b.dest = c.origin) where
a.flightdate ='2008-01-01' AND b.flightdate = '2008-01-01' AND c.flightdate = '2008-01-
01' AND a.origin = 'BOS' AND c.dest = 'BOS'
AND a.DepTime < a.ArrTime
AND b.DepTime < b.ArrTime
AND c.DepTime < c.ArrTime
AND b.DepTime > a.ArrTime
AND c.DepTime > b.ArrTime AND b.DepTime - a.ArrTime > 100 AND c.DepTime - b.ArrTime >
100
AND a.cancelled != 1 AND b.cancelled != 1 AND b.cancelled != 1
Order by TotalTime LIMIT 20;
```

Hive and PigLatin

PigLatin Code:

```
Flights1_Data = FILTER Flights1_Data by (orig1 == 'BOS') AND (flightDate1 == '2008-01-01') AND (cancelled1 != 1);
Flights2_Data = FILTER Flights2_Data by (flightDate2 == '2008-01-01') AND (cancelled2 != 1);
Flights3_Data = FILTER Flights3_Data by  (dest3 == 'BOS') AND (flightDate3 == '2008-01-01') AND (cancelled3 != 1);

f1f2 = JOIN Flights1_Data BY (dest1), Flights2_Data BY (orig2);
f1f2 = FILTER f1f2 BY depTime2 > arrTime1;

f1f2f3 = JOIN f1f2 BY (dest2), Flights3_Data BY (orig3);
f1f2f3 = FILTER f1f2f3 BY  depTime3 > arrTime2;

f1f2f3 = FILTER f1f2f3 BY ((depTime2-arrTime1) > 100) AND ((depTime3-arrTime2) > 100);

final = FOREACH f1f2f3 GENERATE orig1,dest1,dest2,dest3,flightDate1,depTime1,arrTime1,depTime2,arrTime2,depTime3,
arrTime3,  (actualElapsedTime1 + actualElapsedTime2 + actualElapsedTime3 + (depTime2 - arrTime1) + (depTime3 -
arrTime2)) AS totalTripTime;

final = ORDER final BY totalTripTime;
final = limit final 20;
STORE final INTO '$OUTPUT';
```

Hive and PigLatin

Origin	Dest 1	Dest 2	End	Flight Date	F1: DepTime	F1: ArrTime	F2: DepTime	F2: ArrTime	F3: DepTime	F3: ArrTime	TotalTripTime
BOS	DCA	LGA	BOS	1/1/2008	847	1028	1131	1223	1328	1435	428
BOS	DCA	LGA	BOS	1/1/2008	847	1028	1131	1223	1339	1442	435
BOS	DCA	LGA	BOS	1/1/2008	1602	1735	1852	2004	2107	2200	438
BOS	IAD	LGA	BOS	1/1/2008	1021	1211	1320	1420	1528	1625	444
BOS	DCA	LGA	BOS	1/1/2008	847	1028	1131	1223	1354	1452	445
BOS	JFK	PHL	BOS	1/1/2008	1434	1542	1645	1832	1938	2041	447
BOS	LGA	DCA	BOS	1/1/2008	1128	1229	1352	1502	1619	1738	450
BOS	IAD	LGA	BOS	1/1/2008	1021	1211	1320	1420	1530	1632	451
BOS	IAD	LGA	BOS	1/1/2008	1021	1211	1320	1420	1556	1647	466

Hive and PigLatin

Performance Numbers on EMR

Program	No. of Machines	Machine Type	Time Taken
HIVE	11	small	32 Mins
HIVE	20	small	15 Mins
PigLatin	11	small	14 Mins 49 Secs
PigLatin	20	small	13 Mins 23 Secs

THANK YOU