Hadoop EC2 Assigment

Brent Smith, CSCI5673 Spring 2011

Dataset

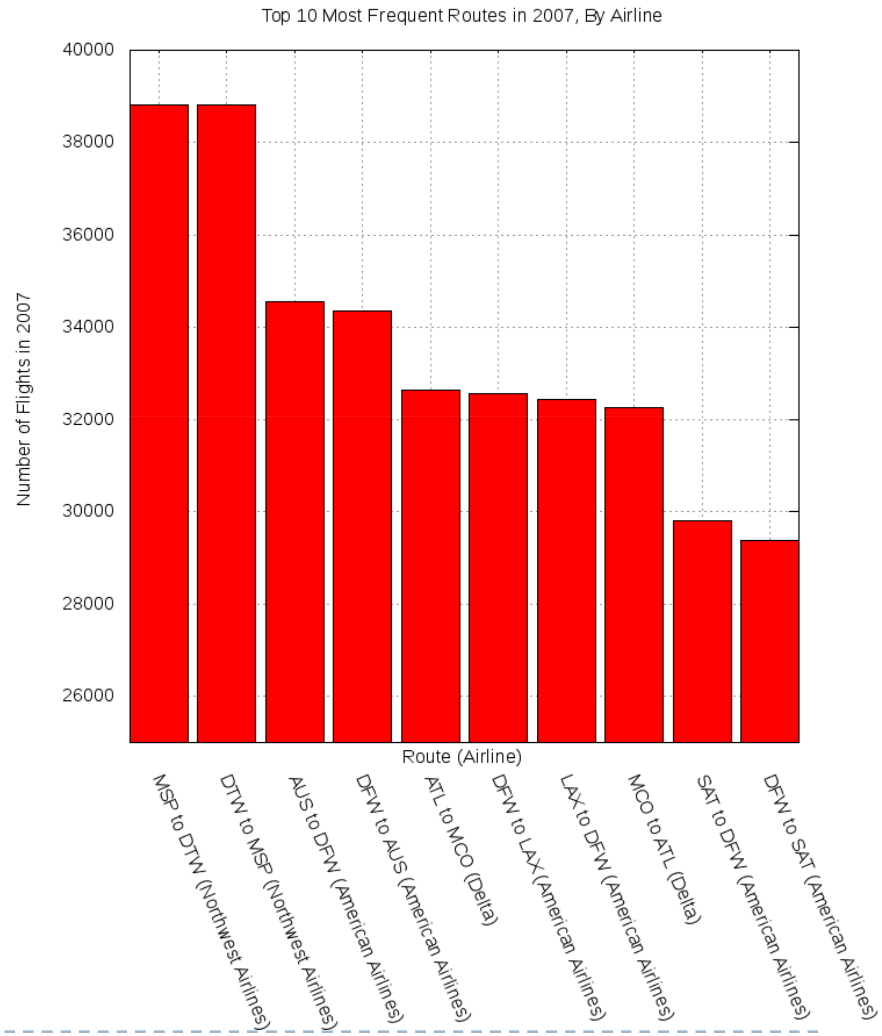
- TransStats Aviation
- Provided by the Bureau of Transportation Statistics to the public
- Older copy of the data is already available via Amazon's public data sets (http://aws.amazon.com/datasets/2289)
- DB1BCoupon table that contains flight data for each quarter.
  - Extracted all quarters in 2007 and concatenated into single file for the year.
  - Roughly 6GB.
  - Copied to 8GB EBS storage volume
- Format is one line per flight, in CSV form

Processing

- Goal was to aggregate all the data by the origin, destination and airline.
- Aggregator.java
  - Map Task
    - Fields to group on is passed on cmd line (also #mappers and #reducers)
    - Input
      - □ Key is ignored
      - □ Value is line from CSV file
    - Output
      - □ Key = grouped fields (origin, destination, airline)
      - □ Value = Literal ONE
  - Reduce Task
    - Sum list of values for each key (origin, destination, airline)
    - Output
      - □ Key = grouped fields (origin, destination, airline)
      - □ Value= sum of all values for given key (# flights with same origin, destination, airline)

Results

- Combination of Origin+Destination = Route
- Total of ~150,000 unique routes+airlines
- Top 10 most frequent routes plotted
- Makes it easy to spot "hubs"
  - Northwest* = MSP (Minneapolis, Minnesota)
  - American Airlines = DFW (Dallas, Ft. Worth)
  - Delta = ATL (Atlanta, Georgia)
  - *Northwest since merged with Delta.

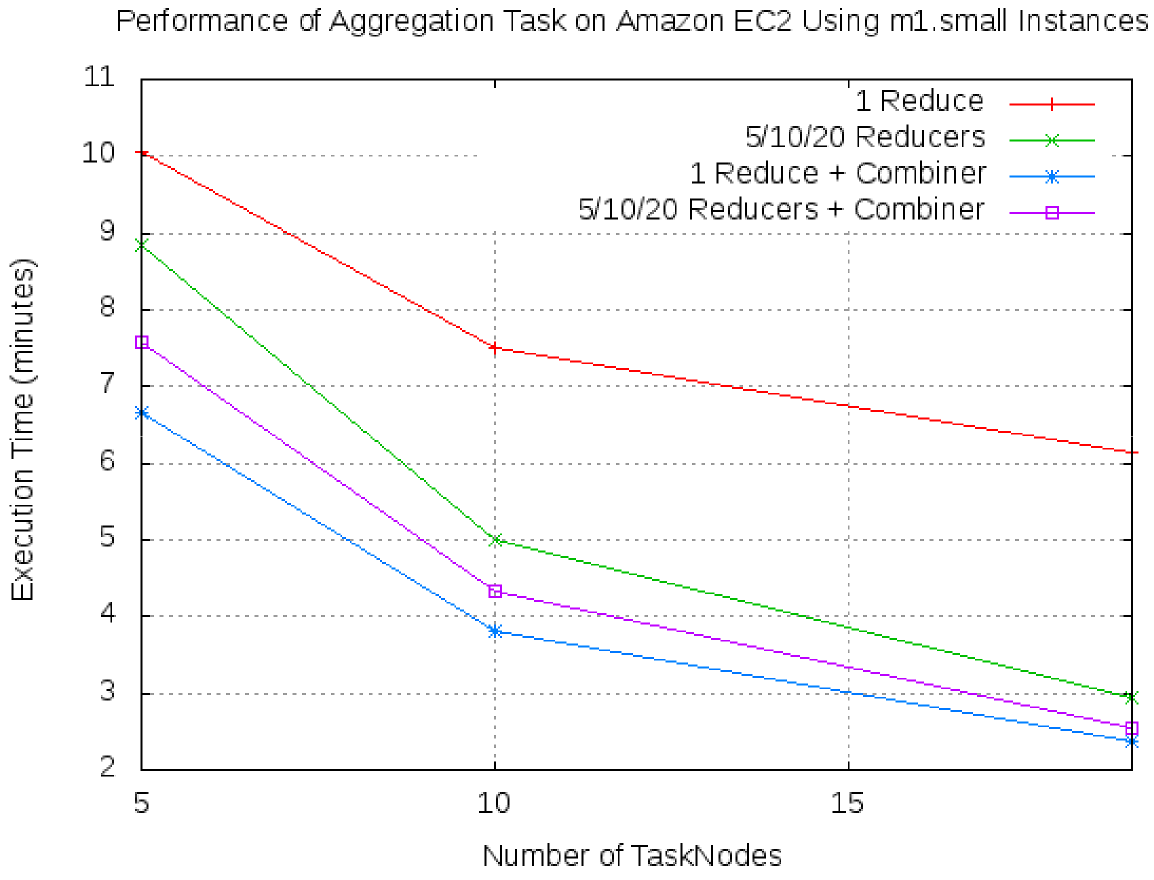


Experimental Setup

- Four tests
  - Default* mappers, One Reducer
    - *6GB / 64MB =~ 92 mappers
  - 100 mappers, #Reducers = #nodes in cluster (5/10/19)
  - Default mappers + Combine stage, One Reducer
  - 100 mappers + Combine stage, #Reducers = #nodes in cluster
- Ran these four tests on clusters with size 5, 10 and 19
  - Excludes the manager node
  - Can only run 20 instances on EC2 so only 19 possible workers
- Ran on both m1.small and c1.medium EC2 instance types
- First test was run with empty buffer cache
  - Subsequent tests (two, three, four) may be affected by OS caching

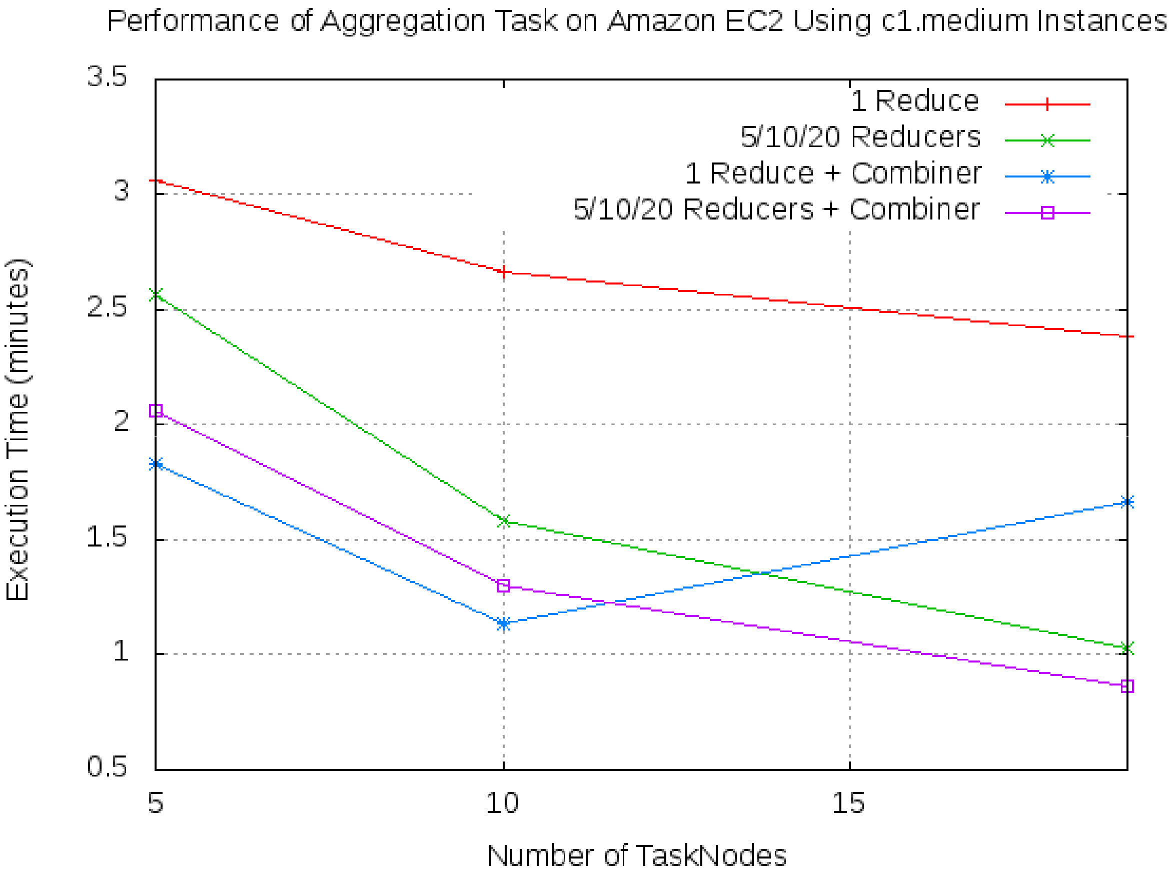
Performance Analysis (m1.small)

- m1.small instance type
- Interesting result: 1 Reducer + Combine stage outperforms many reducers plus combiner
  - Why? Additional overhead to schedule multiple reducers?
  - Difference is quite small.



Performance Analysis (c1.medium)

- c1.medium instance type
- Interesting result: 1 Reducer + Combine stage performs poorly with 20 nodes
  - Why? Bandwidth/data transfer limit? One reducer has to pull all output data from 92 files from each map task?



Extra Credit

- Wanted to do something a little more challenging with the data to understand how difficult it is to do something other than simple aggregation.
- IDEA: Use a graph algorithm to figure out the largest complete subgraph of the graph of all origins and destinations.
  - Maximal clique problem in graph theory.
  - Why? Should tell you the largest set of airports for which every airport in the set has a direct route to every other airport in the set.
    - Is this useful? I don't know. Maybe if you were on the run from the law.

GraphGeneration

- Implemented as a chained MapReduce job.
  - Uses the output from the Aggregator MR job as input.
- GraphGenerator.java
  - Map Task
    - Takes a threshold on # of flights. Don't consider routes with less than 365 flights, e.g. less than one flight per day.
    - Input
      - □ Key is Origin, Destination, Airline
      - □ Value is # of flights
    - Output
      - □ Key = Origin
      - □ Value = Destination
  - Reduce Task
    - Create a java.util.Set of destinations from list of values
    - Output
      - □ Key = Origin
      - □ Value= distinct set of destinations
  - Output creates adjacency list for directed graph

Maximal Clique

- Hard to understand how to do this with MR…
- But, somebody else has already done it
  - XRIME: Cloud-Based Large Scale Social Network Analysis
  - http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?reload=true&arnumber=5557271 (Wei Xue; JuWei Shi; Bo Yang; IBM Res. China, Beijing, China)
- XRIME
  - Implements many graph algorithms using Hadoop MR
  - Implements maximal clique
- Solution: stand on the shoulders of others; e.g., use XRIME.

Chained Maximal Clique

- Input to Maximal Clique MR Job in XRIME is adjacency list
  - Corresponds to output from GraphGenerator MR job
- ChainedMaximalCliqueJob.java
  - Run Aggregation task
  - Run GraphGenerator task
  - Run TextAdjTransformer task
    - Provided by XRIME
    - Converts text based adjacency list into binary form
  - Run MaximalStrongCliqueAlgorithm
    - Outputs value that is set of all nodes in maximal clique
  - Use temporary directories to chain output from one MR job to input of another MR job

Maximal Clique Results

- Many "maximal" cliques of various sizes
  - Because "maximal" clique is defined as largest set of nodes for which adding another node would make subgraph incomplete
  - We want the largest "maximal" clique
  - Use largest_airport_cliques.py to parse output and find these.
- 4 maximal cliques of size 27.
  - Complete subgraphs, so # edges = n(n-1)/2 = 351 (undirected graph)
  - Can reach any airport in set from any other airport in set
  - Useful if you want to increase the work that law enforcement has to do to catch you
- Four sets are available in README.txt.

Questions

- Still not sure how to do maximal clique with MR…
  - Source code is available, study it.
- Maximal clique operates on undirected graphs
  - Ours was directed.
  - Assume that XRIME converts directed to undirected only when there are two directed edges (A→B, and B→A).
    - MR sub-jobs in XRIME called "StrongNeighborhoodGenerate"; I think these have something to do with directed → undirected conversion
  - Read XRIME paper, maybe it talks about this?