Final Project Report

Deepen Mehta, Abhijeet Sharma, Afan Ahmad Khan, Akshay Raje

April 22, 2016

Introduction

Our program implements Map Reduce paradigm in a server-client system. Our program consist of 1 Server and multiple Client nodes(sort nodes).The Server and each client runs on individual EC2 instances, and the Server issues tasks to each client. The objective of the program is to design and implement a streamlined API built from first principles. The source code can be found here.

Design Implementation

Our Map-Reduce API design can be broady categorized into 3 phases :

-> Mapper phase

1. Our automation script spins up 1 Server and N Client EC2 instances.
2. Our program reads input files from S3 input bucket and performs a map operation on the file contents.
3. The mapper output(list of keys and values) is stored in the S3 in a temp directory.

-> Sample and Shuffle phase

1. The Server asks the clients to begin the sample tasks.
2. The data from the temp directory(mapped output) are distributed uniformly (by size) to each client.
3. Once data is read by the client, each client provides samples of data to server.
4. The server calculates pivot points from all the samples and distributes the pivots to each client.
5. Each client partitions its data based on the pivot points and stores partitions in a common bucket in S3.
6. Each client reads it's repective partitions from S3 and merges the data.
7. The merged data is read by the reducer.

-> Reducer phase

1. Our program takes in data from local directory which is to be read by the Reducer class.
2. The class reads each file and performs an Iterable of values to the reduce method.
3. The final reducer output is stored in the user provided S3 output directory.

Code Implementation

com.aws package

This package contains classes to connect to AWS from a java program and handle all request to S3.

com.examples package

This package contains Word Count implementation using out Map Reduce.

com.main package

This package contain classes which contain the 'main' function to run the program.

com.map package

This package contain classes to perform map operation on the input records.

com.net package

This package contains the classes for server and client nodes and the communication between them.

com.reduce package

This package contains the classes to perform reduce operation based on the each key generated by the mapper class.

com.sort package

This package consists the implementation of Distributed Sort algorithm as found in [http://web.cs.dal.ca/~arc/teaching/CSci6702/2013/Assignment2/SampleSort.pdf].

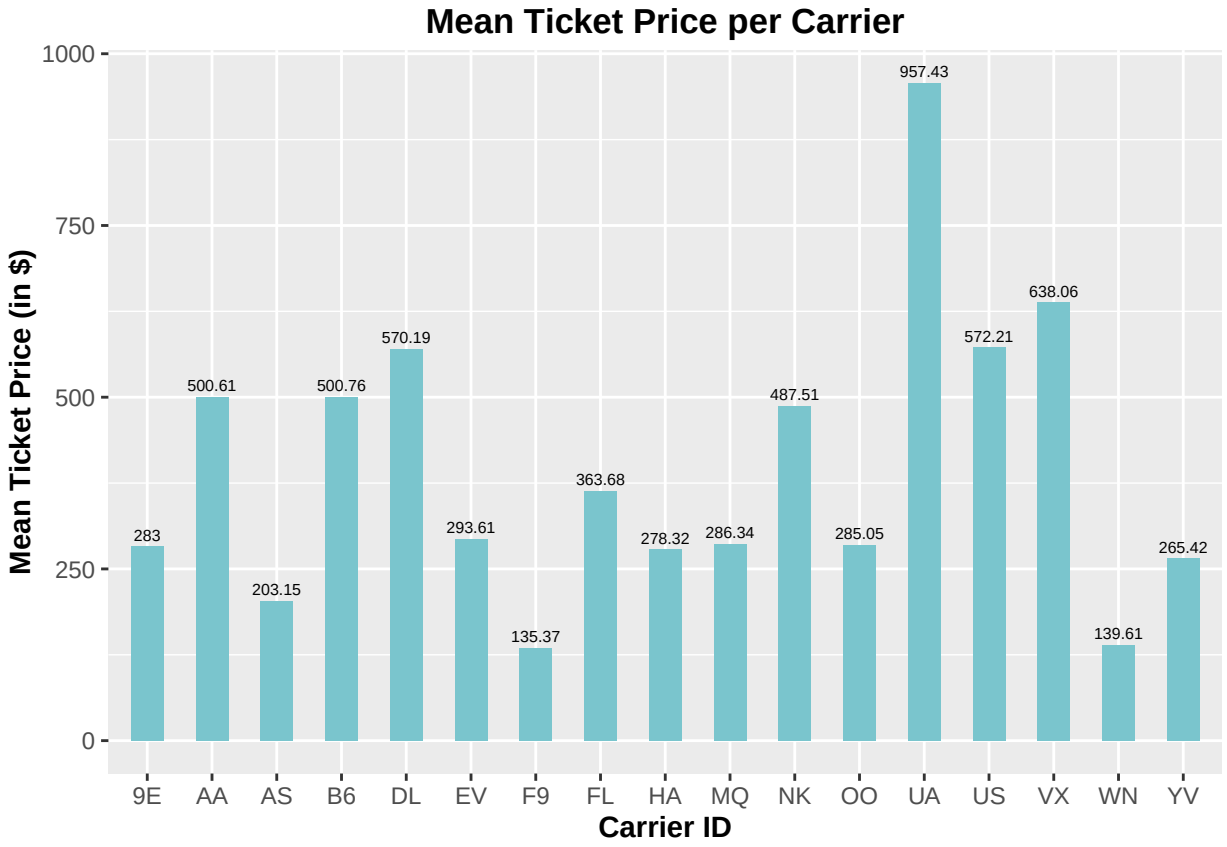
com.utils

This package consists of miscellaneous utility methods.

Example Implementation - Mean Flight

This is an implementation of Assignment 2 where we calculate the mean average ticket price of unique carriers. Data is calculated from 2 input files.

```
##   Unique_Carrier Mean_Ticket_Price
## 1             9E            283.00
## 2             AA            500.61
## 3             AS            203.15
## 4             B6            500.76
## 5             DL            570.19
## 6             EV            293.61
```



Job Execution

1 Server, 2 Client (t2.medium)

We executed Word Count program using our Map Reduce API on AWS EC2 with 1 Server and 2 Client nodes in `t2.medium` configuration. Total running time was ~4 minutes.

Pseudo

We executed Mean flight program using our Map Reduce API on local machine configuration. Total running time was ~16 minutes for 653.8MB of compressed data.

1 Server, 5 Client (m4.xlarge)

We executed Mean flight program using our Map Reduce API on local machine configuration. Total running time was ~2 minutes, 15 seconds for 653.8MB of compressed data.

Assumptions

The framework assumes the keys and values to be in String format.

Task Distribution:

Server-client Network: Abhijeet Sharma, Deepen Mehta *Mapper-Reducer:* Abhijeet Sharma, Deepen Mehta *EC2 Automation:* Afan Khan, Akshay Raje *Packaging:* Akshay Raje, Afan Khan *Code Maintainence:* Afan Khan
Report: Afan Khan *ReadMe:* Akshay Raje