Geetish Nayak Purvi Bafna

Sherry Shi Jyoti Yadav

**Assignment 2**

**Objective**

The purpose of this assignment is to track buy and sell prices over time for each stock.

**Input Data**

10 minutes worth of server logs from the old Lehman Brothers trading desks.

**How this is achieved**

2 Map – Reduce programs

1. Purpose of 1st Map – Reduce program

This program reads the log file, filters out the required data by only considering rows that have Stock symbol, Price and Side and then outputs these data into various files.

**Please read below for detailed implementation explanation.**

1. Purpose of 2nd Map – Reduce program

Since the requirement is that the output should be sorted, this program is basically used to sort the output that we obtained from the first program.

**Please read below for detailed implementation explanation.**

**Detailed Explanation**

1. 1st Map – Reduce program

**TransactionInfo.java** – This is a custom class which implements Writable interface and overrides necessary methods. The purpose of this class is to store the Symbol, Timestamp, Price and Side for a specific transaction. It is used as a value which goes of the Mapper.

**CompositeKey.java** – The purpose of this function is to create a custom key so that sorting can be done over it. Since, hadoop only sorts on keys and not on values, this was necessary to do.

**ActualKeyPartitioner.java –** This is to ensure that all the records with the same symbol should go to the same reducer. By this we mean to say that mapper outputs which have symbol = AMZN should go to the same reducer.

**CompositeKeyComparator.java** – This is basically used for sorting on the basis of keys so that the records going to a reducer are first sorted on the basis of Symbol, then timestamp, side and price.

**TraderMapper.java –** This is the mapper class which reads data from the file, processes it and sends it to the reducer. The **ActualPartitioner** class decides to which reducer this record should go to.

**TraderReducer.java –** This is the reducer class where it receives data from the mapper. The purpose of the reducer in this case is only to make sure that the data gets written to the corresponding output file.

**Trader.java –** This is the driver class for the mapper and the reducer. The partitioner, Comparator and other important details are initialized out here.

**Intermediary Step**

Since there are multiple reducers we get multiple output files which are individually sorted but when we merge them it becomes unsorted. So we have to write a second Map – Reduce program for this purpose and make sure that the number of reducers in this case = 1. The purpose of doing this is because we want to sort the entire file.

**2nd Map – Reduce program**

**SortingMapper.java**  - The purpose of this class is to read the file that has been generated by the first Map – Reduce program and the send the data to the reducer. Since there is only 1 reducer all the data that has been generated by various mappers goes to the same reducer. The key in this case is the entire record.

**SortingReducer.java -** The purpose of this reducer is to get the data from various mappers and write it into a file. Since hadoop by default sorts on keys, the input is already sorted. We just have to write it to an output file.

**Sorting.java –**  This is the driver class for the mapper and the reducer. The number of reducers = 1, is set out here.