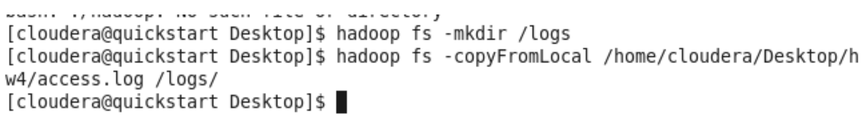
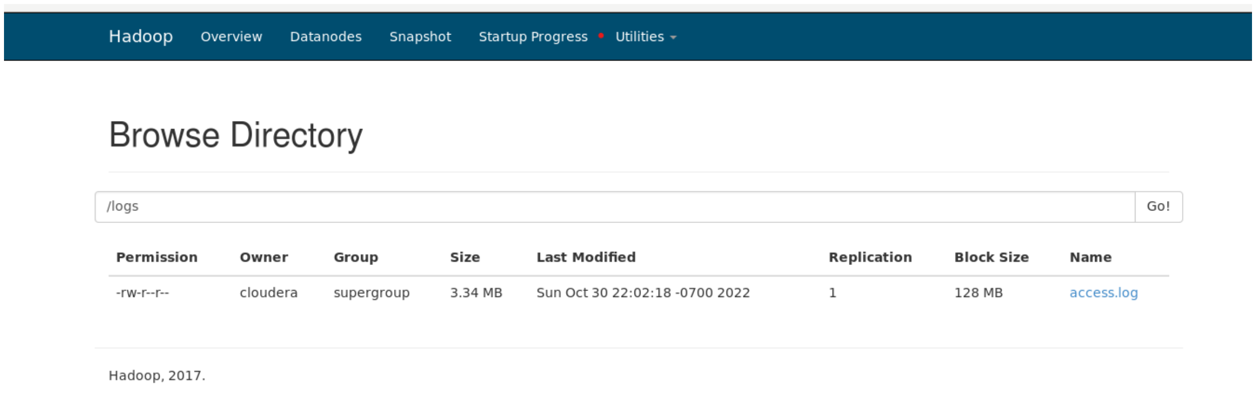
Part-2:

Copy the attached ‘access.log’ file into HDFS under /logs directory.
Using the access.log file stored in HDFS, implement MapReduce in Hadoop to find the number of times each IP accessed the website.
Run the job without a Combiner, and see how long it will take (you may use your clock or smart watch to find the running time)
Then, add the Combiner and see if it will run faster. Since Counting is both commutative and associative operation, you could use the same Reducer as a Combiner.



```
[cloudera@quickstart Desktop]$ hadoop fs -mkdir /logs
[cloudera@quickstart Desktop]$ hadoop fs -copyFromLocal /home/cloudera/Desktop/h
w4/access.log /logs/
[cloudera@quickstart Desktop]$ 
```



Browse Directory

/logs Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	cloudera	supergroup	3.34 MB	Sun Oct 30 22:02:18 -0700 2022	1	128 MB	access.log

Hadoop, 2017.

Mapper Class:

```
import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;


public class IPAddrMapper extends Mapper<Object, Text, Text, IntWritable>{

  private final static IntWritable one = new IntWritable(1);

  Text ipAddressText = new Text();

  //called once for each key/value pair in the input split

  protected void map(Object key, Text value, Mapper<Object, Text, Text,
    IntWritable>.Context context) throws IOException, InterruptedException {
        String line = value.toString();
        String[] tokens = line.split(" ");
        String ipAddress = tokens[0].trim();
        System.out.println("Mapping ipAddress: " + ipAddress);
        ipAddressText.set(ipAddress);
        context.write(ipAddressText, one);   //emit(key,value)
    }
}
```

Reducer Class:

```
public class IPAddrReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    protected void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable,
        Text, IntWritable>.Context context) throws IOException, InterruptedException {
            IntWritable sumObj = new IntWritable(0);
            int sum = 0;
            for(IntWritable val:values) {
                sum += val.get();
        }
        sumObj.set(sum);
        System.out.println("Reducing key: "+ key + " Final count: " + sum);
        context.write(key, sumObj); //returning Hadoop Datatype
    }
}
```

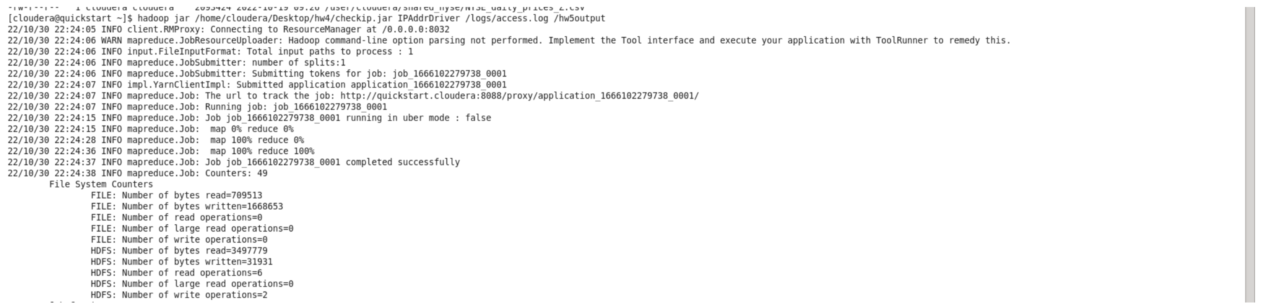
Driver Class:

```
public class IPAddrDriver {
         public static void main(String[] args) throws IOException, InterruptedException,
ClassNotFoundException {
            Configuration configuration = new Configuration();
```

```
        Job job = Job.getInstance(configuration, "IP Address COunter");


        job.setJarByClass(IPAddrDriver.class);
        job.setMapperClass(IPAddrMapper.class);
        job.setReducerClass(IPAddrReducer.class);


        //Reducer Output
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);


        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }


}
```

Run Command:

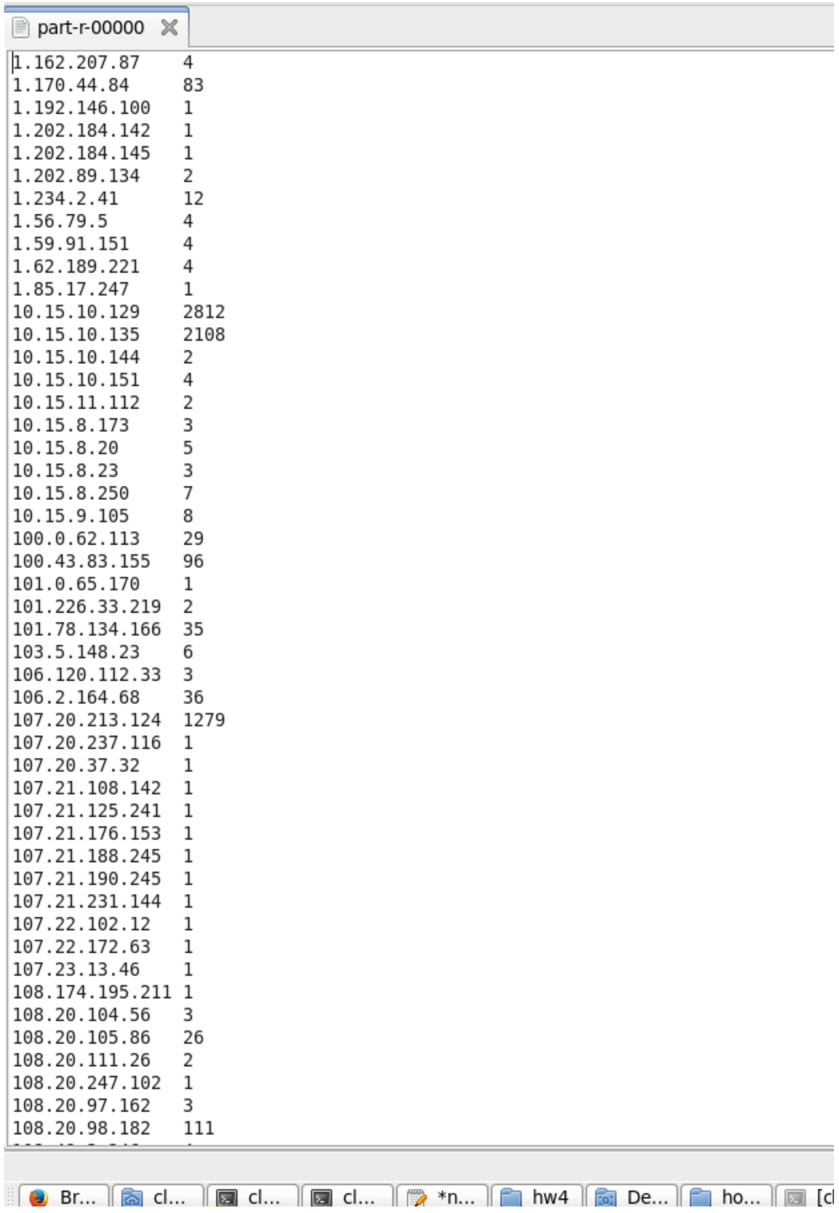
hadoop jar /home/cloudera/Desktop/hw4/checkip.jar IPAddrDriver /logs/access.log /hw5output



```
-rw-r--r--   1 cloudera cloudera    2093424 2022-10-19 09:26 /user/cloudera/shared_nyse/NYSE_daily_prices_Z.csv
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/Desktop/hw4/checkip.jar IPAddrDriver /logs/access.log /hw5output
22/10/30 22:24:05 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
22/10/30 22:24:06 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
22/10/30 22:24:06 INFO input.FileInputFormat: Total input paths to process : 1
22/10/30 22:24:06 INFO mapreduce.JobSubmitter: number of splits:1
22/10/30 22:24:06 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1666102279738_0001
22/10/30 22:24:07 INFO impl.YarnClientImpl: Submitted application application_1666102279738_0001
22/10/30 22:24:07 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1666102279738_0001/
22/10/30 22:24:07 INFO mapreduce.Job: Running job: job_1666102279738_0001
22/10/30 22:24:15 INFO mapreduce.Job: Job job_1666102279738_0001 running in uber mode : false
22/10/30 22:24:15 INFO mapreduce.Job:  map 0% reduce 0%
22/10/30 22:24:28 INFO mapreduce.Job:  map 100% reduce 0%
22/10/30 22:24:36 INFO mapreduce.Job:  map 100% reduce 100%
22/10/30 22:24:37 INFO mapreduce.Job: Job job_1666102279738_0001 completed successfully
22/10/30 22:24:38 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=709513
                FILE: Number of bytes written=1668653
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=3497779
                HDFS: Number of bytes written=31931
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
```

Run Time: 25 seconds

Output:



```
1.162.207.87    4
1.170.44.84     83
1.192.146.100   1
1.202.184.142   1
1.202.184.145   1
1.202.89.134    2
1.234.2.41      12
1.56.79.5       4
1.59.91.151     4
1.62.189.221    4
1.85.17.247     1
10.15.10.129    2812
10.15.10.135    2108
10.15.10.144    2
10.15.10.151    4
10.15.11.112    2
10.15.8.173     3
10.15.8.20      5
10.15.8.23      3
10.15.8.250     7
10.15.9.105     8
100.0.62.113    29
100.43.83.155   96
101.0.65.170    1
101.226.33.219  2
101.78.134.166  35
103.5.148.23    6
106.120.112.33  3
106.2.164.68    36
107.20.213.124  1279
107.20.237.116  1
107.20.37.32    1
107.21.108.142  1
107.21.125.241  1
107.21.176.153  1
107.21.188.245  1
107.21.190.245  1
107.21.231.144  1
107.22.102.12   1
107.22.172.63   1
107.23.13.46    1
108.174.195.211 1
108.20.104.56   3
108.20.105.86   26
108.20.111.26   2
108.20.247.102  1
108.20.97.162   3
108.20.98.182   111
```

<u>With Combiner:</u>

Driver Class

public class IPAddrDriver {

public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

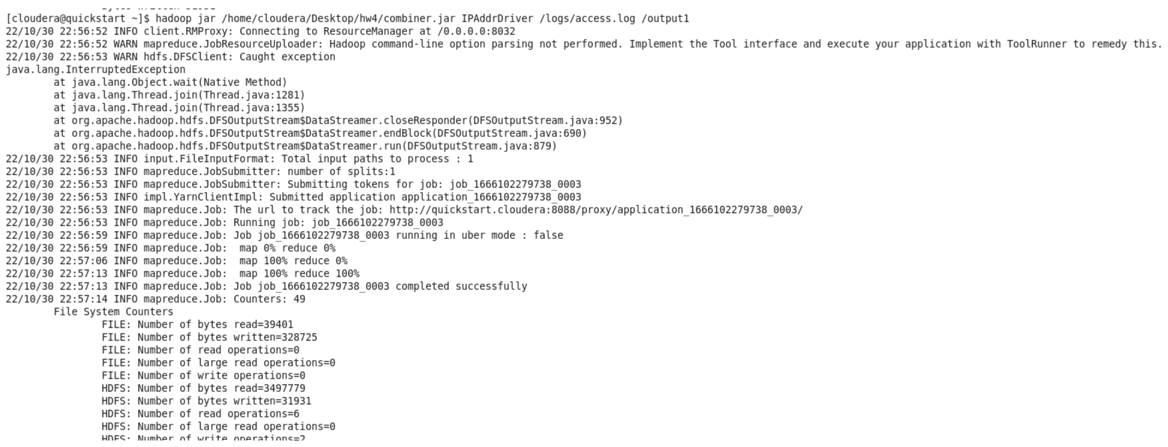
Configuration configuration = new Configuration();

Job job = Job.getInstance(configuration, "IP Address COunter");

```
        job.setJarByClass(IPAddrDriver.class);

        job.setMapperClass(IPAddrMapper.class);

        job.setCombinerClass(IPAddrReducer.class);

        job.setReducerClass(IPAddrReducer.class);

        //Reducer Output

        job.setOutputKeyClass(Text.class);

        job.setOutputValueClass(IntWritable.class);


        FileInputFormat.addInputPath(job, new Path(args[0]));

        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);

    }

}
```

Run Command:

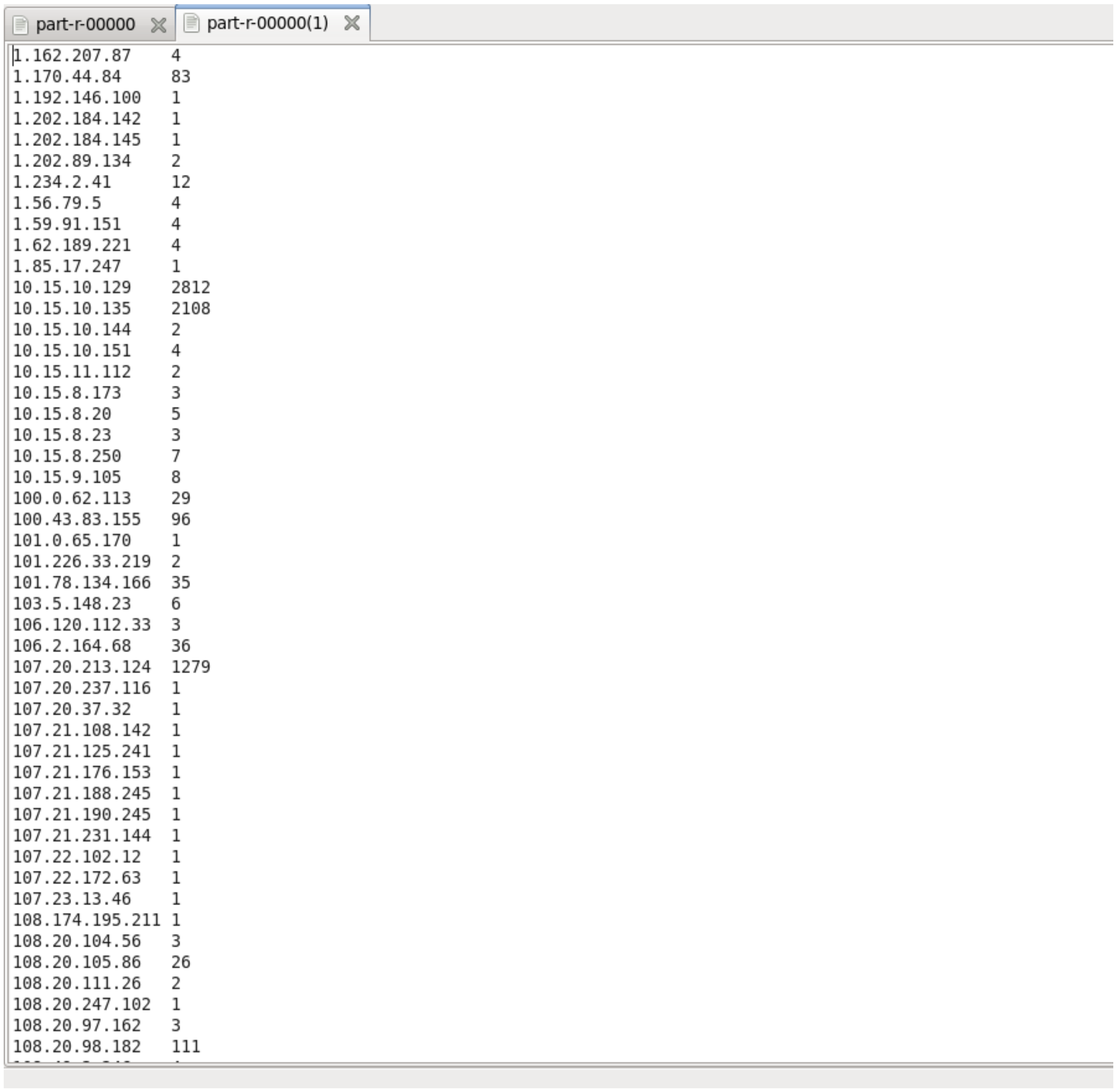
```
hadoop jar /home/cloudera/Desktop/hw4/combiner.jar IPAddrDriver /logs/access.log /output1
```



```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/Desktop/hw4/combiner.jar IPAddrDriver /logs/access.log /output1
22/10/30 22:56:52 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
22/10/30 22:56:52 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
22/10/30 22:56:53 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:952)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:690)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:879)
22/10/30 22:56:53 INFO input.FileInputFormat: Total input paths to process : 1
22/10/30 22:56:53 INFO mapreduce.JobSubmitter: number of splits:1
22/10/30 22:56:53 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1666102279738_0003
22/10/30 22:56:53 INFO impl.YarnClientImpl: Submitted application application_1666102279738_0003
22/10/30 22:56:53 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1666102279738_0003/
22/10/30 22:56:53 INFO mapreduce.Job: Running job: job_1666102279738_0003
22/10/30 22:56:59 INFO mapreduce.Job: Job job_1666102279738_0003 running in uber mode : false
22/10/30 22:56:59 INFO mapreduce.Job:  map 0% reduce 0%
22/10/30 22:57:06 INFO mapreduce.Job:  map 100% reduce 0%
22/10/30 22:57:13 INFO mapreduce.Job:  map 100% reduce 100%
22/10/30 22:57:13 INFO mapreduce.Job: Job job_1666102279738_0003 completed successfully
22/10/30 22:57:14 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=39401
                FILE: Number of bytes written=328725
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=3497779
                HDFS: Number of bytes written=31931
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
```

Run Time: 22 seconds

Output:



```
1.162.207.87     4
1.170.44.84      83
1.192.146.100    1
1.202.184.142    1
1.202.184.145    1
1.202.89.134     2
1.234.2.41       12
1.56.79.5        4
1.59.91.151      4
1.62.189.221     4
1.85.17.247      1
10.15.10.129     2812
10.15.10.135     2108
10.15.10.144     2
10.15.10.151     4
10.15.11.112     2
10.15.8.173      3
10.15.8.20       5
10.15.8.23       3
10.15.8.250      7
10.15.9.105      8
100.0.62.113     29
100.43.83.155    96
101.0.65.170     1
101.226.33.219   2
101.78.134.166   35
103.5.148.23     6
106.120.112.33   3
106.2.164.68     36
107.20.213.124   1279
107.20.237.116   1
107.20.37.32     1
107.21.108.142   1
107.21.125.241   1
107.21.176.153   1
107.21.188.245   1
107.21.190.245   1
107.21.231.144   1
107.22.102.12    1
107.22.172.63    1
107.23.13.46     1
108.174.195.211  1
108.20.104.56    3
108.20.105.86    26
108.20.111.26    2
108.20.247.102   1
108.20.97.162    3
108.20.98.182    111
```

Part-3: Programming Assignment

Download and Copy all the files (http://newton.neu.edu/nyse/ (Links to an external site.)

public class copyNYSE {

    public static void main(String[] args) throws Exception {

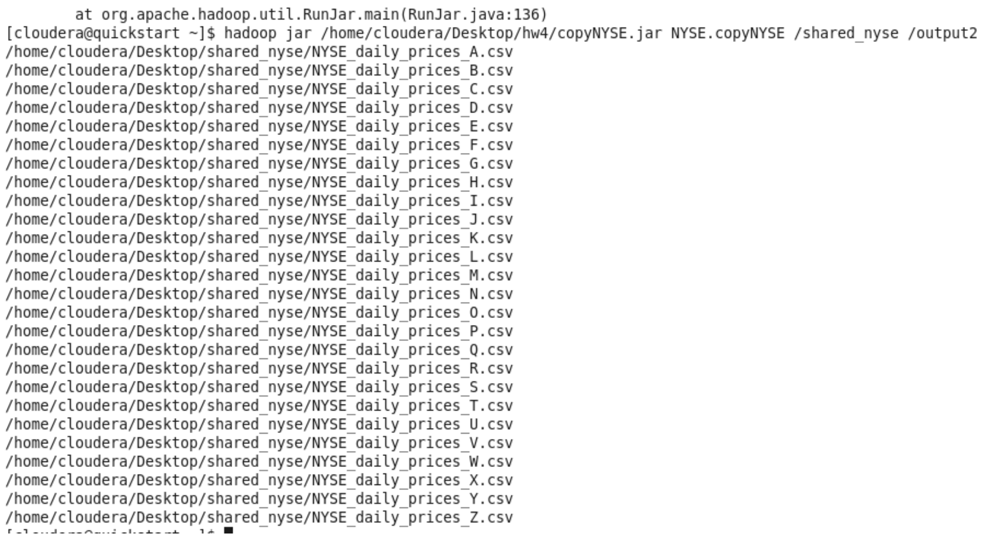
        Configuration config = new Configuration();

        FileSystem hdfs = FileSystem.get(config);

        FileSystem local = FileSystem.getLocal(config);

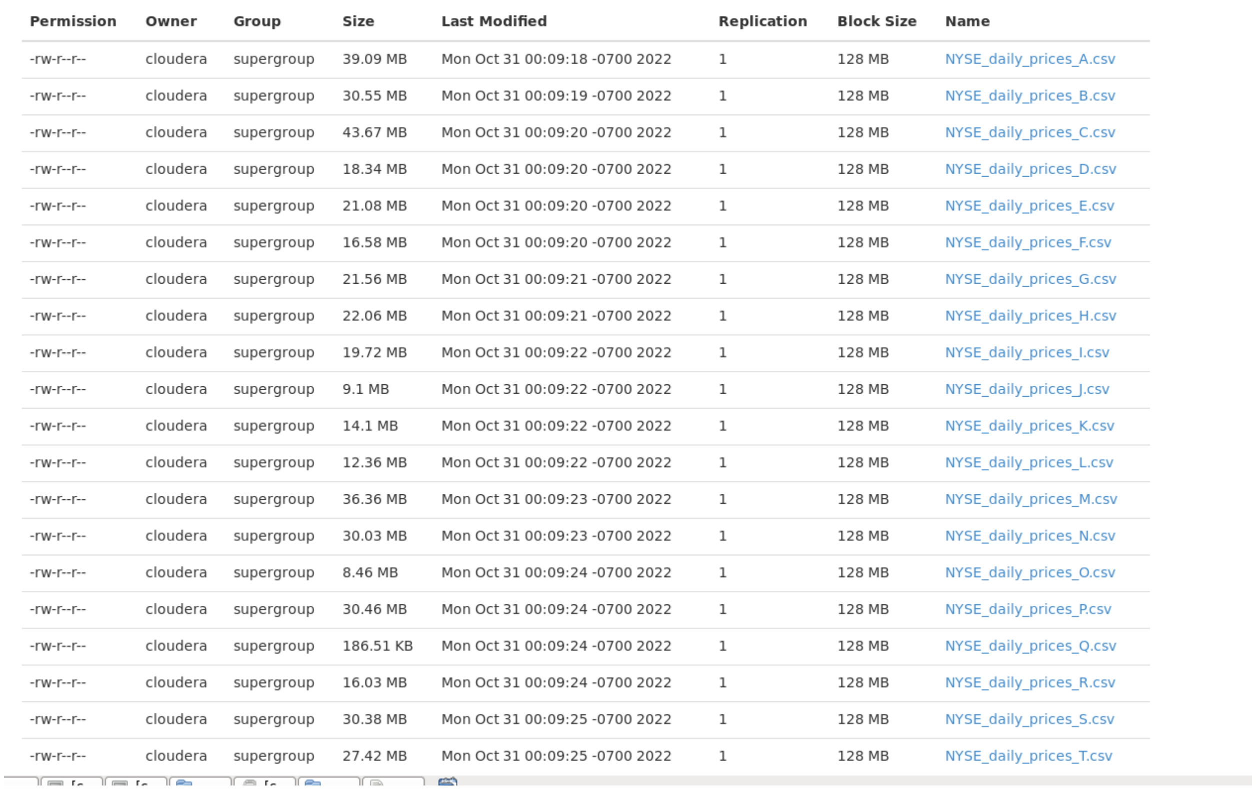
```
        for(char i = 'A' ; i <= 'Z'; i++) {

            String csvPathPrefix = "NYSE_daily_prices_" + i;

            String fullPath =

            "/home/cloudera/Desktop/shared_nyse/" +

            csvPathPrefix + ".csv";

            System.out.println(fullPath);

            hdfs.copyFromLocalFile(new Path(fullPath), new Path("/hw4/NYSE/"));

}

}
```

Run Command: hadoop jar /home/cloudera/Desktop/hw4/copyNYSE.jar NYSE.copyNYSE /shared_nyse /output2



```
        at org.apache.hadoop.util.RunJar.main(RunJar.java:136)
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/Desktop/hw4/copyNYSE.jar NYSE.copyNYSE /shared_nyse /output2
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_A.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_B.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_C.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_D.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_E.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_F.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_G.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_H.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_I.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_J.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_K.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_L.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_M.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_N.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_O.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_P.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_Q.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_R.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_S.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_T.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_U.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_V.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_W.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_X.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_Y.csv
/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_Z.csv
```

Browse Files:



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	cloudera	supergroup	39.09 MB	Mon Oct 31 00:09:18 -0700 2022	1	128 MB	NYSE_daily_prices_A.csv
-rw-r--r--	cloudera	supergroup	30.55 MB	Mon Oct 31 00:09:19 -0700 2022	1	128 MB	NYSE_daily_prices_B.csv
-rw-r--r--	cloudera	supergroup	43.67 MB	Mon Oct 31 00:09:20 -0700 2022	1	128 MB	NYSE_daily_prices_C.csv
-rw-r--r--	cloudera	supergroup	18.34 MB	Mon Oct 31 00:09:20 -0700 2022	1	128 MB	NYSE_daily_prices_D.csv
-rw-r--r--	cloudera	supergroup	21.08 MB	Mon Oct 31 00:09:20 -0700 2022	1	128 MB	NYSE_daily_prices_E.csv
-rw-r--r--	cloudera	supergroup	16.58 MB	Mon Oct 31 00:09:20 -0700 2022	1	128 MB	NYSE_daily_prices_F.csv
-rw-r--r--	cloudera	supergroup	21.56 MB	Mon Oct 31 00:09:21 -0700 2022	1	128 MB	NYSE_daily_prices_G.csv
-rw-r--r--	cloudera	supergroup	22.06 MB	Mon Oct 31 00:09:21 -0700 2022	1	128 MB	NYSE_daily_prices_H.csv
-rw-r--r--	cloudera	supergroup	19.72 MB	Mon Oct 31 00:09:22 -0700 2022	1	128 MB	NYSE_daily_prices_I.csv
-rw-r--r--	cloudera	supergroup	9.1 MB	Mon Oct 31 00:09:22 -0700 2022	1	128 MB	NYSE_daily_prices_J.csv
-rw-r--r--	cloudera	supergroup	14.1 MB	Mon Oct 31 00:09:22 -0700 2022	1	128 MB	NYSE_daily_prices_K.csv
-rw-r--r--	cloudera	supergroup	12.36 MB	Mon Oct 31 00:09:22 -0700 2022	1	128 MB	NYSE_daily_prices_L.csv
-rw-r--r--	cloudera	supergroup	36.36 MB	Mon Oct 31 00:09:23 -0700 2022	1	128 MB	NYSE_daily_prices_M.csv
-rw-r--r--	cloudera	supergroup	30.03 MB	Mon Oct 31 00:09:23 -0700 2022	1	128 MB	NYSE_daily_prices_N.csv
-rw-r--r--	cloudera	supergroup	8.46 MB	Mon Oct 31 00:09:24 -0700 2022	1	128 MB	NYSE_daily_prices_O.csv
-rw-r--r--	cloudera	supergroup	30.46 MB	Mon Oct 31 00:09:24 -0700 2022	1	128 MB	NYSE_daily_prices_P.csv
-rw-r--r--	cloudera	supergroup	186.51 KB	Mon Oct 31 00:09:24 -0700 2022	1	128 MB	NYSE_daily_prices_Q.csv
-rw-r--r--	cloudera	supergroup	16.03 MB	Mon Oct 31 00:09:24 -0700 2022	1	128 MB	NYSE_daily_prices_R.csv
-rw-r--r--	cloudera	supergroup	30.38 MB	Mon Oct 31 00:09:25 -0700 2022	1	128 MB	NYSE_daily_prices_S.csv
-rw-r--r--	cloudera	supergroup	27.42 MB	Mon Oct 31 00:09:25 -0700 2022	1	128 MB	NYSE_daily_prices_T.csv

3.1: Copy NYSE dataset (DailyPrices_A to DailyPrices_Z) to a folder in HDFS. No merging.
Write a MapReduce to find the Max price of stock_price_high for each stock. Capture the running time programmatically (or manually using a wristwatch or smartphone).

Mapper Class:

```
public class MaxStockPriceMap extends Mapper<Object, Text, Text, DoubleWritable> {

    protected void map(Object key, Text value, Mapper<Object, Text, Text,

    DoubleWritable>.Context context) throws IOException, InterruptedException {

    String[] tokens = value.toString().split(",");

    Text stockSymbol = new Text();

    stockSymbol.set(tokens[1]);

    DoubleWritable price = new DoubleWritable();

    Double stock_price_high_value = 0.0;

    try {

      stock_price_high_value = Double.parseDouble(tokens[4]);

      price.set(stock_price_high_value);

    }
```

```
        catch (NumberFormatException nf) {
            System.out.println("Continue without interruption, its the header line");{
    }
            context.write(stockSymbol, price);
        }
    }
}
```

Reducer Class:

```
public class MaxStockPriceReducer extends Reducer<Text, DoubleWritable, Text, DoubleWritable> {
    public void reduce(Text key, Iterable<DoubleWritable> values, Reducer.Context context) throws
IOException, InterruptedException {
        double final_stock_high = Double.MIN_VALUE;
            for (DoubleWritable val: values) {
            final_stock_high = Math.max(val.get(), final_stock_high);
        }
    DoubleWritable final_stock_price_red = new DoubleWritable(final_stock_high);
    context.write(key, final_stock_price_red);
    }
}
```

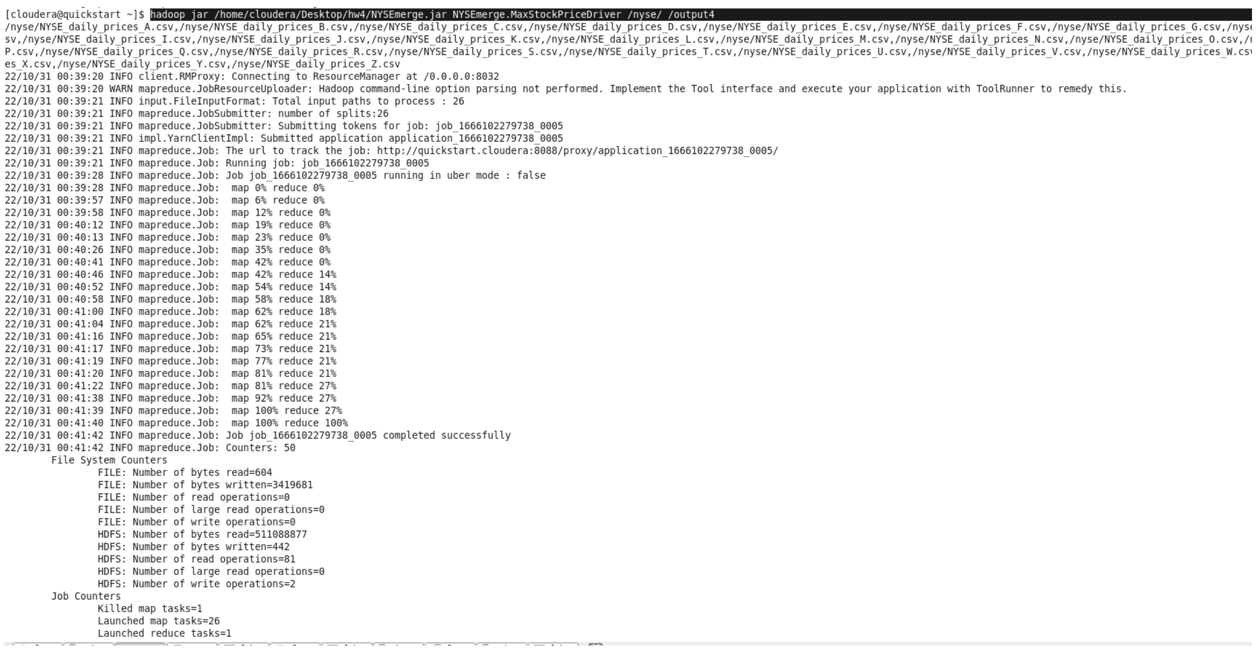
Driver Class:

```
public class MaxStockPriceDriver {
    public static void main(String[] args) throws Exception {
    Configuration configuration = new Configuration();
    Job job = Job.getInstance(configuration, "Calculating max stock price high");
    job.setJarByClass(MaxStockPriceDriver.class);
    job.setMapperClass(MaxStockPriceMap.class);
    job.setCombinerClass(MaxStockPriceReducer.class);
    job.setReducerClass(MaxStockPriceReducer.class);
    //output of the mapper
    job.setOutputKeyClass(Text.class);

    job.setOutputValueClass(DoubleWritable.class);

    StringBuilder fullPath = new StringBuilder("");

    for(char i = 'A' ; i <= 'Z'; i++) {

        String csvPathPrefix = "NYSE_daily_prices_" + i + ".csv";

        if (i == 'Z') {

        fullPath.append(args[0]).append(csvPathPrefix);

}

else {

    fullPath.append(args[0]).append(csvPathPrefix).append(",");

}

}

    String finalcommaSeperatedPath = fullPath.toString();

    System.out.println(finalcommaSeperatedPath);

    FileInputFormat.addInputPaths(job, finalcommaSeperatedPath);

    FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

    }
```

Run Command:

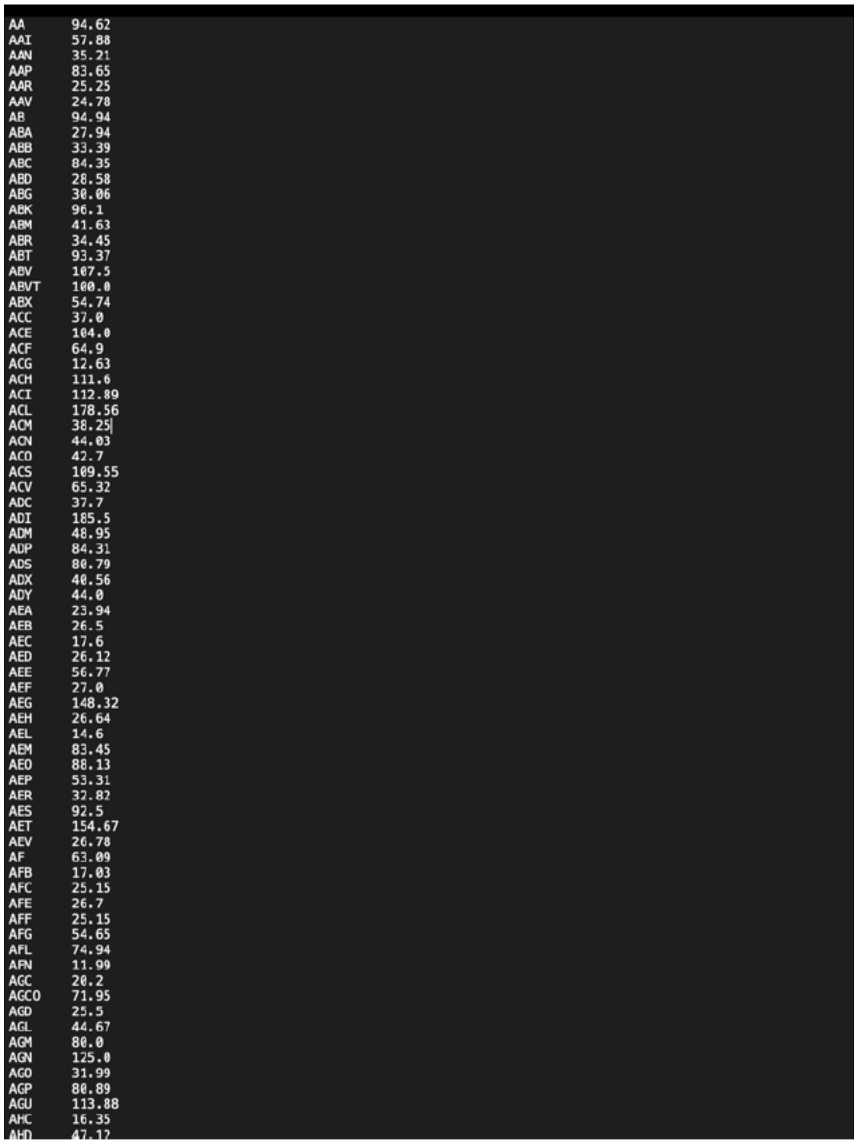
```
hadoop jar /home/cloudera/Desktop/hw4/NYSEmerge.jar NYSEmerge.MaxStockPriceDriver /nyse/
/output4
```

Run Time: 2 minutes 8 seconds



```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/Desktop/hw4/NYSEmerge.jar NYSEmerge.MaxStockPriceDriver /nyse/ /output4
/nyse/NYSE_daily_prices_A.csv,/nyse/NYSE_daily_prices_B.csv,/nyse/NYSE_daily_prices_C.csv,/nyse/NYSE_daily_prices_D.csv,/nyse/NYSE_daily_prices_E.csv,/nyse/NYSE_daily_prices_F.csv,/nyse/NYSE_daily_prices_G.csv,/nyse
sv,/nyse/NYSE_daily_prices_I.csv,/nyse/NYSE_daily_prices_J.csv,/nyse/NYSE_daily_prices_K.csv,/nyse/NYSE_daily_prices_L.csv,/nyse/NYSE_daily_prices_M.csv,/nyse/NYSE_daily_prices_N.csv,/nyse/NYSE_daily_prices_O.csv,/n
P.csv,/nyse/NYSE_daily_prices_Q.csv,/nyse/NYSE_daily_prices_R.csv,/nyse/NYSE_daily_prices_S.csv,/nyse/NYSE_daily_prices_T.csv,/nyse/NYSE_daily_prices_U.csv,/nyse/NYSE_daily_prices_V.csv,/nyse/NYSE_daily_prices_W.csv
es_X.csv,/nyse/NYSE_daily_prices_Y.csv,/nyse/NYSE_daily_prices_Z.csv
22/10/31 00:39:20 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
22/10/31 00:39:20 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
22/10/31 00:39:21 INFO input.FileInputFormat: Total input paths to process : 26
22/10/31 00:39:21 INFO mapreduce.JobSubmitter: number of splits:26
22/10/31 00:39:21 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1666102279738_0005
22/10/31 00:39:21 INFO impl.YarnClientImpl: Submitted application application_1666102279738_0005
22/10/31 00:39:21 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1666102279738_0005/
22/10/31 00:39:21 INFO mapreduce.Job: Running job: job_1666102279738_0005
22/10/31 00:39:28 INFO mapreduce.Job: Job job_1666102279738_0005 running in uber mode : false
22/10/31 00:39:28 INFO mapreduce.Job:  map 0% reduce 0%
22/10/31 00:39:57 INFO mapreduce.Job:  map 6% reduce 0%
22/10/31 00:39:58 INFO mapreduce.Job:  map 12% reduce 0%
22/10/31 00:40:12 INFO mapreduce.Job:  map 19% reduce 0%
22/10/31 00:40:13 INFO mapreduce.Job:  map 23% reduce 0%
22/10/31 00:40:26 INFO mapreduce.Job:  map 35% reduce 0%
22/10/31 00:40:41 INFO mapreduce.Job:  map 42% reduce 0%
22/10/31 00:40:46 INFO mapreduce.Job:  map 42% reduce 14%
22/10/31 00:40:52 INFO mapreduce.Job:  map 54% reduce 14%
22/10/31 00:40:58 INFO mapreduce.Job:  map 58% reduce 18%
22/10/31 00:41:00 INFO mapreduce.Job:  map 62% reduce 18%
22/10/31 00:41:04 INFO mapreduce.Job:  map 62% reduce 21%
22/10/31 00:41:16 INFO mapreduce.Job:  map 65% reduce 21%
22/10/31 00:41:17 INFO mapreduce.Job:  map 73% reduce 21%
22/10/31 00:41:19 INFO mapreduce.Job:  map 77% reduce 21%
22/10/31 00:41:20 INFO mapreduce.Job:  map 81% reduce 21%
22/10/31 00:41:22 INFO mapreduce.Job:  map 81% reduce 27%
22/10/31 00:41:38 INFO mapreduce.Job:  map 92% reduce 27%
22/10/31 00:41:39 INFO mapreduce.Job:  map 100% reduce 27%
22/10/31 00:41:40 INFO mapreduce.Job:  map 100% reduce 100%
22/10/31 00:41:42 INFO mapreduce.Job: Job job_1666102279738_0005 completed successfully
22/10/31 00:41:42 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=604
                FILE: Number of bytes written=3419681
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=511088877
                HDFS: Number of bytes written=442
                HDFS: Number of read operations=81
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Killed map tasks=1
                Launched map tasks=26
                Launched reduce tasks=1
```

Output:



```
AA      94.62
AAI     57.88
AAN     35.21
AAP     83.65
AAR     25.25
AAV     24.78
AB      94.94
ABA     27.94
ABB     33.39
ABC     84.35
ABD     28.58
ABG     30.06
ABK     96.1
ABM     41.63
ABR     34.45
ABT     93.37
ABV     107.5
ABVT    100.0
ABX     54.74
ACC     37.0
ACE     104.0
ACF     64.9
ACG     12.63
ACH     111.6
ACI     112.89
ACL     178.56
ACM     38.25
ACN     44.03
ACO     42.7
ACS     109.55
ACV     65.32
ADC     37.7
ADI     185.5
ADM     48.95
ADP     84.31
ADS     80.79
ADX     40.56
ADY     44.0
AEA     23.94
AEB     26.5
AEC     17.6
AED     26.12
AEE     56.77
AEF     27.0
AEG     148.32
AEH     26.64
AEL     14.6
AEM     83.45
AEO     88.13
AEP     53.31
AER     32.82
AES     92.5
AET     154.67
AEV     26.78
AF      63.09
AFB     17.03
AFC     25.15
AFE     26.7
AFF     25.15
AFG     54.65
AFL     74.94
AFN     11.99
AGC     20.2
AGCO    71.95
AGD     25.5
AGL     44.67
AGM     80.0
AGN     125.0
AGO     31.99
AGP     80.89
AGU     113.88
AHC     16.35
AHD     47.12
```

3.2: Redo 3.1. Merge the NYSE files in a single file (that you may have done in your prev. HW) on HDFS. Now, repeat 3.1 on the single merged-file. Capture the running time. Did MapReduce on a single file run faster than running MapReduce on a bunch of files?

Part-A:

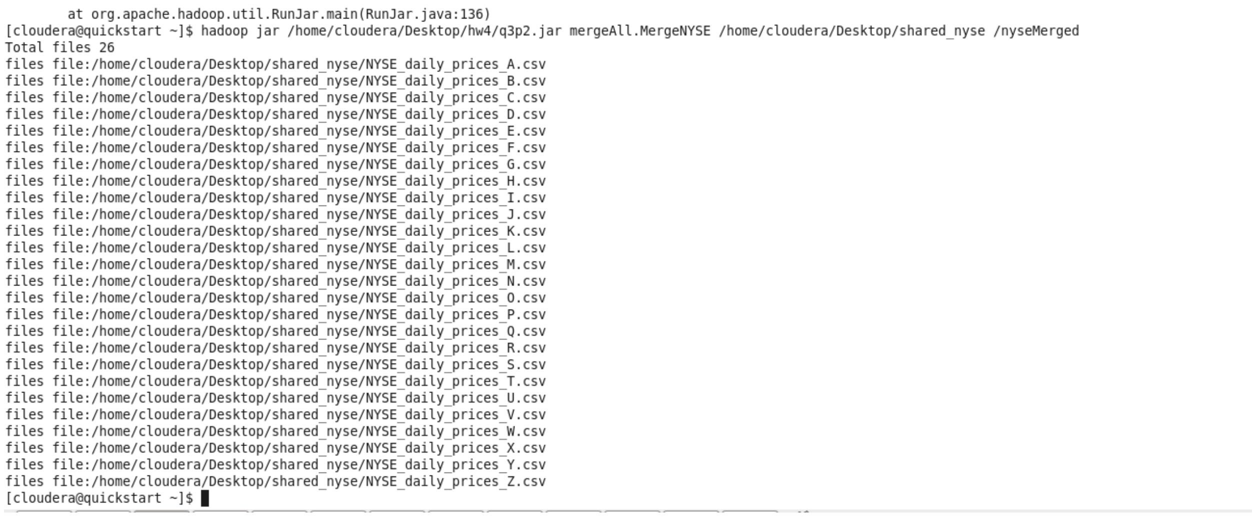
Code for merging NYSE files:

```
public class MergeNYSE {

  public static void main(String args[]) throws Exception {

  Configuration config = new Configuration();

  FileSystem hdfs = FileSystem.get(config);

  FileSystem local = FileSystem.getLocal(config);
```

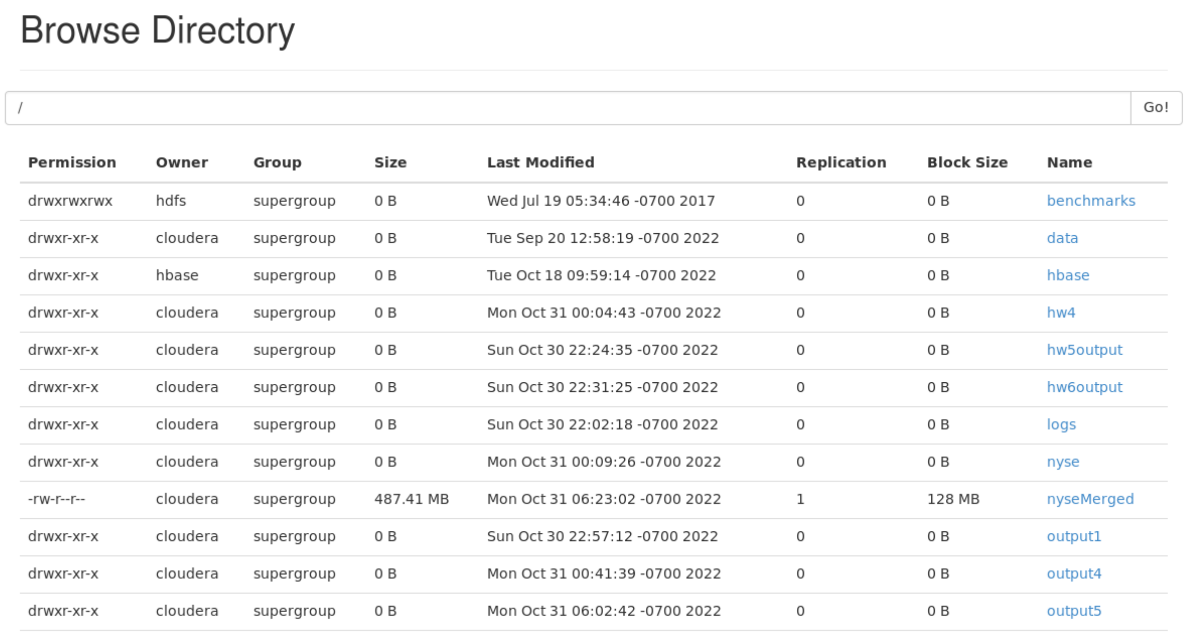
```
   Path op = new Path(args[1]);

   Path ip = new Path(args[0]);

   try {

       FileStatus[] ipFile = local.listStatus(ip);

       FSDataOutputStream outputStream = hdfs.create(op);

       System.out.println("Total files " + ipFile.length);

       for (int i = 0; i < ipFile.length; i++) {

       System.out.println("files " + ipFile[i].getPath());

       FSDataInputStream input = local.open(ipFile[i].getPath());

       byte buffer[] = new byte[256];

       int bytesRead = 0;

       while ((bytesRead = input.read(buffer)) > 0) {

               outputStream.write(buffer, 0, bytesRead);

}

   input.close();

}

   outputStream.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}
```

Run Command:

```
hadoop jar /home/cloudera/Desktop/hw4/q3p2.jar mergeAll.MergeNYSE
/home/cloudera/Desktop/shared_nyse /nyseMerged
```


```
        at org.apache.hadoop.util.RunJar.main(RunJar.java:136)
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/Desktop/hw4/q3p2.jar mergeAll.MergeNYSE /home/cloudera/Desktop/shared_nyse /nyseMerged
Total files 26
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_A.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_B.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_C.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_D.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_E.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_F.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_G.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_H.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_I.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_J.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_K.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_L.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_M.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_N.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_O.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_P.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_Q.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_R.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_S.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_T.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_U.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_V.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_W.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_X.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_Y.csv
files file:/home/cloudera/Desktop/shared_nyse/NYSE_daily_prices_Z.csv
[cloudera@quickstart ~]$ 
```

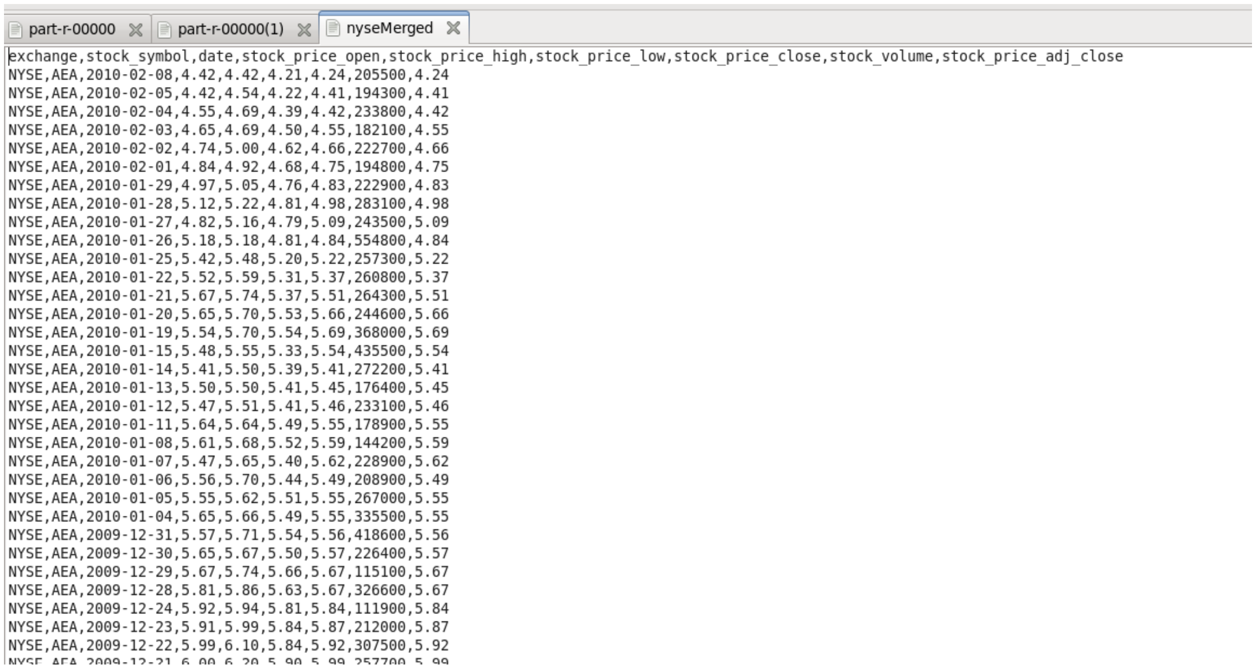
Merged Files:


Browse Directory

/

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxrwx	hdfs	supergroup	0 B	Wed Jul 19 05:34:46 -0700 2017	0	0 B	benchmarks
drwxr-xr-x	cloudera	supergroup	0 B	Tue Sep 20 12:58:19 -0700 2022	0	0 B	data
drwxr-xr-x	hbase	supergroup	0 B	Tue Oct 18 09:59:14 -0700 2022	0	0 B	hbase
drwxr-xr-x	cloudera	supergroup	0 B	Mon Oct 31 00:04:43 -0700 2022	0	0 B	hw4
drwxr-xr-x	cloudera	supergroup	0 B	Sun Oct 30 22:24:35 -0700 2022	0	0 B	hw5output
drwxr-xr-x	cloudera	supergroup	0 B	Sun Oct 30 22:31:25 -0700 2022	0	0 B	hw6output
drwxr-xr-x	cloudera	supergroup	0 B	Sun Oct 30 22:02:18 -0700 2022	0	0 B	logs
drwxr-xr-x	cloudera	supergroup	0 B	Mon Oct 31 00:09:26 -0700 2022	0	0 B	nyse
-rw-r--r--	cloudera	supergroup	487.41 MB	Mon Oct 31 06:23:02 -0700 2022	1	128 MB	nyseMerged
drwxr-xr-x	cloudera	supergroup	0 B	Sun Oct 30 22:57:12 -0700 2022	0	0 B	output1
drwxr-xr-x	cloudera	supergroup	0 B	Mon Oct 31 00:41:39 -0700 2022	0	0 B	output4
drwxr-xr-x	cloudera	supergroup	0 B	Mon Oct 31 06:02:42 -0700 2022	0	0 B	output5

Output:



```
exchange,stock_symbol,date,stock_price_open,stock_price_high,stock_price_low,stock_price_close,stock_volume,stock_price_adj_close
NYSE,AEA,2010-02-08,4.42,4.42,4.21,4.24,205500,4.24
NYSE,AEA,2010-02-05,4.42,4.54,4.22,4.41,194300,4.41
NYSE,AEA,2010-02-04,4.55,4.69,4.39,4.42,233800,4.42
NYSE,AEA,2010-02-03,4.65,4.69,4.50,4.55,182100,4.55
NYSE,AEA,2010-02-02,4.74,5.00,4.62,4.66,222700,4.66
NYSE,AEA,2010-02-01,4.84,4.92,4.68,4.75,194800,4.75
NYSE,AEA,2010-01-29,4.97,5.05,4.76,4.83,222900,4.83
NYSE,AEA,2010-01-28,5.12,5.22,4.81,4.98,283100,4.98
NYSE,AEA,2010-01-27,4.82,5.16,4.79,5.09,243500,5.09
NYSE,AEA,2010-01-26,5.18,5.18,4.81,4.84,554800,4.84
NYSE,AEA,2010-01-25,5.42,5.48,5.20,5.22,257300,5.22
NYSE,AEA,2010-01-22,5.52,5.59,5.31,5.37,260800,5.37
NYSE,AEA,2010-01-21,5.67,5.74,5.37,5.51,264300,5.51
NYSE,AEA,2010-01-20,5.65,5.70,5.53,5.66,244600,5.66
NYSE,AEA,2010-01-19,5.54,5.70,5.54,5.69,368000,5.69
NYSE,AEA,2010-01-15,5.48,5.55,5.33,5.54,435500,5.54
NYSE,AEA,2010-01-14,5.41,5.50,5.39,5.41,272200,5.41
NYSE,AEA,2010-01-13,5.50,5.50,5.41,5.45,176400,5.45
NYSE,AEA,2010-01-12,5.47,5.51,5.41,5.46,233100,5.46
NYSE,AEA,2010-01-11,5.64,5.64,5.49,5.55,178900,5.55
NYSE,AEA,2010-01-08,5.61,5.68,5.52,5.59,144200,5.59
NYSE,AEA,2010-01-07,5.47,5.65,5.40,5.62,228900,5.62
NYSE,AEA,2010-01-06,5.56,5.70,5.44,5.49,208900,5.49
NYSE,AEA,2010-01-05,5.55,5.62,5.51,5.55,267000,5.55
NYSE,AEA,2010-01-04,5.65,5.66,5.49,5.55,335500,5.55
NYSE,AEA,2009-12-31,5.57,5.71,5.54,5.56,418600,5.56
NYSE,AEA,2009-12-30,5.65,5.67,5.50,5.57,226400,5.57
NYSE,AEA,2009-12-29,5.67,5.74,5.66,5.67,115100,5.67
NYSE,AEA,2009-12-28,5.81,5.86,5.63,5.67,326600,5.67
NYSE,AEA,2009-12-24,5.92,5.94,5.81,5.84,111900,5.84
NYSE,AEA,2009-12-23,5.91,5.99,5.84,5.87,212000,5.87
NYSE,AEA,2009-12-22,5.99,6.10,5.84,5.92,307500,5.92
NYSE,AEA,2009-12-21,6.00,6.20,5.90,5.99,257700,5.99
```

Part-B:

Mapper Class:

```
public class MaxStockPriceMap extends Mapper<Object, Text, Text, DoubleWritable> {

    protected void map(Object key, Text value, Mapper<Object, Text, Text,

    DoubleWritable>.Context context) throws IOException, InterruptedException {

    String[] tokens = value.toString().split(",");

    Text stockSymbol = new Text();

    stockSymbol.set(tokens[1]);

    DoubleWritable price = new DoubleWritable();

    Double stock_price_high_value = 0.0;

  try {

    stock_price_high_value = Double.parseDouble(tokens[4]);

    price.set(stock_price_high_value);

  }

  catch (NumberFormatException nf) {

  System.out.println("Continue without interruption, its the header line");
```

```
   }
   context.write(stockSymbol, price);
   }

}
```

Reducer Class:

```
public class MaxStockPriceReduce extends Reducer<Text, DoubleWritable, Text,DoubleWritable> {
   public void reduce(Text key, Iterable<DoubleWritable> values, Context context)
   throws IOException, InterruptedException {
   double final_stock_high = Double.MIN_VALUE;
   for (DoubleWritable val: values) {
   final_stock_high = Math.max(val.get(), final_stock_high);
   }
   DoubleWritable final_stock_price_red = new DoubleWritable(final_stock_high);
   context.write(key, final_stock_price_red);
}

}
```

Driver Class:

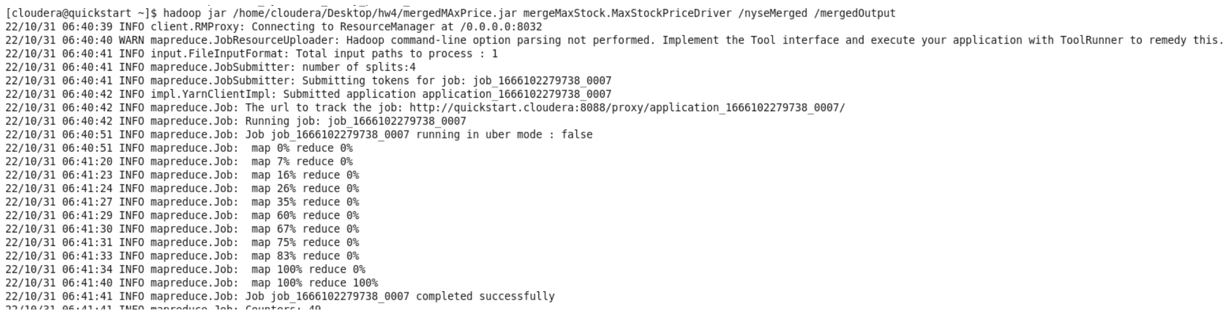
```
public class MaxStockPriceDriver {
public static void main(String[] args) throws Exception {
      Configuration configuration = new Configuration();
      Job job = Job.getInstance(configuration, "Calculating max stock price high");
      job.setJarByClass(MaxStockPriceDriver.class);
      job.setMapperClass(MaxStockPriceMap.class);
      job.setCombinerClass(MaxStockPriceReduce.class);
      job.setReducerClass(MaxStockPriceReduce.class);
      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(DoubleWritable.class);
```

```
        FileInputFormat.addInputPath(job, new Path(args[0]));

        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}
```

Run Command:

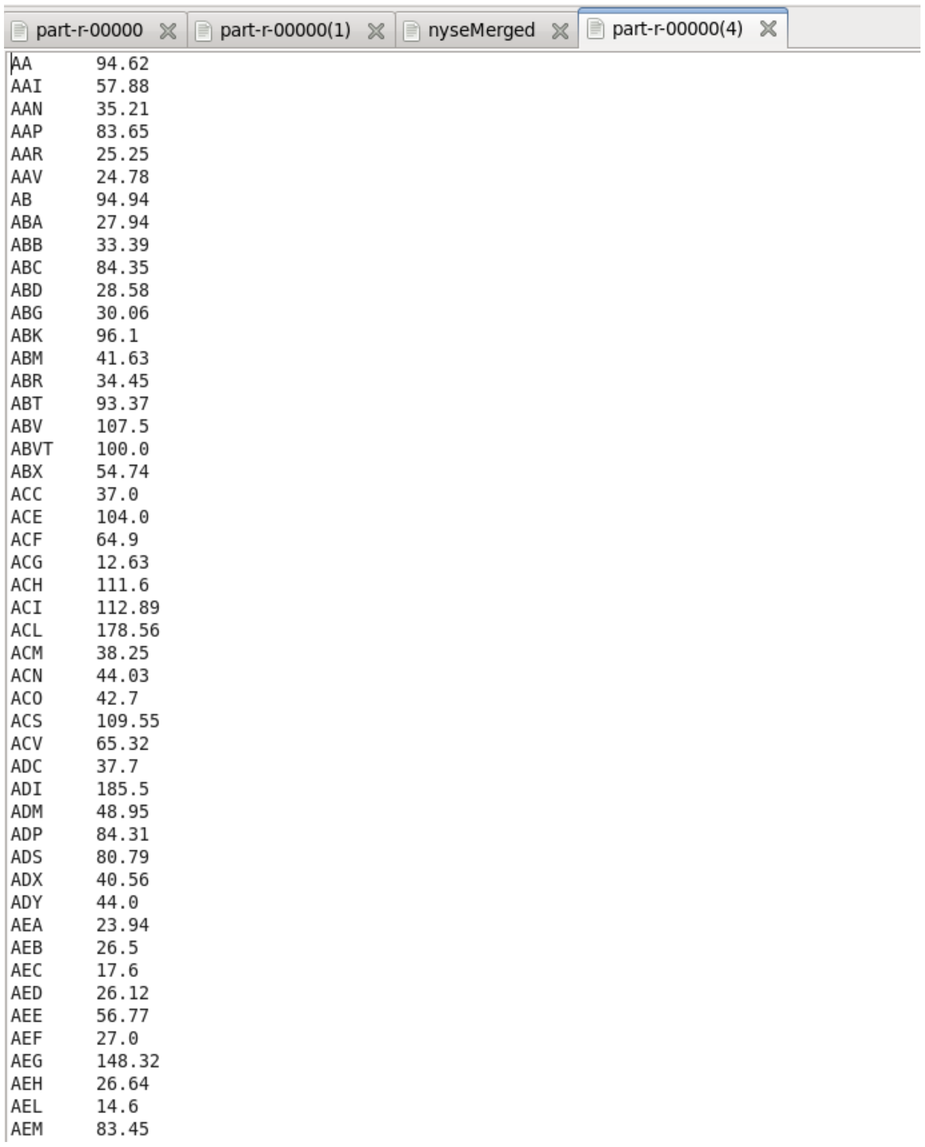
```
hadoop jar /home/cloudera/Desktop/hw4/mergedMAxPrice.jar mergeMaxStock.MaxStockPriceDriver
/nyseMerged /mergedOutput
```

Run time: 1 minute 8 seconds



```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/Desktop/hw4/mergedMAxPrice.jar mergeMaxStock.MaxStockPriceDriver /nyseMerged /mergedOutput
22/10/31 06:40:39 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
22/10/31 06:40:40 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
22/10/31 06:40:41 INFO input.FileInputFormat: Total input paths to process : 1
22/10/31 06:40:41 INFO mapreduce.JobSubmitter: number of splits:4
22/10/31 06:40:41 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1666102279738_0007
22/10/31 06:40:42 INFO impl.YarnClientImpl: Submitted application application_1666102279738_0007
22/10/31 06:40:42 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1666102279738_0007/
22/10/31 06:40:42 INFO mapreduce.Job: Running job: job_1666102279738_0007
22/10/31 06:40:51 INFO mapreduce.Job: Job job_1666102279738_0007 running in uber mode : false
22/10/31 06:40:51 INFO mapreduce.Job:  map 0% reduce 0%
22/10/31 06:41:20 INFO mapreduce.Job:  map 7% reduce 0%
22/10/31 06:41:23 INFO mapreduce.Job:  map 16% reduce 0%
22/10/31 06:41:24 INFO mapreduce.Job:  map 26% reduce 0%
22/10/31 06:41:27 INFO mapreduce.Job:  map 35% reduce 0%
22/10/31 06:41:29 INFO mapreduce.Job:  map 60% reduce 0%
22/10/31 06:41:30 INFO mapreduce.Job:  map 67% reduce 0%
22/10/31 06:41:31 INFO mapreduce.Job:  map 75% reduce 0%
22/10/31 06:41:33 INFO mapreduce.Job:  map 83% reduce 0%
22/10/31 06:41:34 INFO mapreduce.Job:  map 100% reduce 0%
22/10/31 06:41:40 INFO mapreduce.Job:  map 100% reduce 100%
22/10/31 06:41:41 INFO mapreduce.Job: Job job_1666102279738_0007 completed successfully
22/10/31 06:41:41 INFO mapreduce.Job: Counters: 49
```

Merged Output:



```
AA      94.62
AAI     57.88
AAN     35.21
AAP     83.65
AAR     25.25
AAV     24.78
AB      94.94
ABA     27.94
ABB     33.39
ABC     84.35
ABD     28.58
ABG     30.06
ABK     96.1
ABM     41.63
ABR     34.45
ABT     93.37
ABV     107.5
ABVT    100.0
ABX     54.74
ACC     37.0
ACE     104.0
ACF     64.9
ACG     12.63
ACH     111.6
ACI     112.89
ACL     178.56
ACM     38.25
ACN     44.03
ACO     42.7
ACS     109.55
ACV     65.32
ADC     37.7
ADI     185.5
ADM     48.95
ADP     84.31
ADS     80.79
ADX     40.56
ADY     44.0
AEA     23.94
AEB     26.5
AEC     17.6
AED     26.12
AEE     56.77
AEF     27.0
AEG     148.32
AEH     26.64
AEL     14.6
AEM     83.45
```

MapReduce on a single file: 1 minute 8 seconds

MapReduce on a bunch of files: 2 minutes 8 seconds

Hence, MapReduce on a single file runs faster than that of a bunch of files.

Part-4: Programming Assignment

Write one MapReduce program using each of the classes that extend FileInputFormat<k,v>

1. **CombineFileInputFormat**

   An abstract InputFormat that returns CombineFileSplit's in InputFormat.getSplits(JobContext) method. Splits are constructed from the files under the input paths. A split cannot have files from different pools. Each split returned may contain blocks from different files. If a maxSplitSize is specified, then blocks on the same node are combined to form a single split. Blocks that are left over are then combined with other blocks in the same rack. If maxSplitSize is not specified, then blocks from the same rack are combined in a single split; no attempt is made to create node-local splits.

Mapper Class:

```
package CombinedFileInputFormat;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class CombinedMap extends Mapper<Object,DoubleWritable, Text, DoubleWritable> {
   IntWritable one = new IntWritable(1);

   public void map(Object key, DoubleWritable value, Context context) throws IOException,
InterruptedException {
      Text word = new Text();
      context.write(key, one);
   }
}
```

Reducer Class:

```
package CombinedFileInputFormat;

import java.io.IOException;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

/**
 *
 * @author zeeni
 */
public class CombinedReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
   protected void reduce(Text key, Iterable< DoubleWritable > values,
      Reducer<Text, DoubleWritable, Text, DoubleWritable >.Context context) throws IOException,
InterruptedException {
      IntWritable finalSum = new IntWritable(0);
      int sum = 0;
      for(DoubleWritable val:values) {
          sum += val.get();
      }
      finalSum.set(sum);
      System.out.println("Reducing key: "+ key + " final sum: " + sum);
      context.write(key, finalSum);
   }

}
```

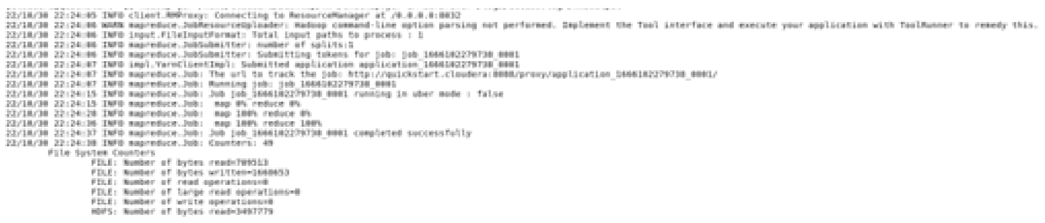
Driver Class:

```
package CombinedFileInputFormat;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.CombineFileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;


/**
 *
 * @author zeeni
 */
public class CombinedDriver {
        public static void main(String[] args) throws Exception {
        Configuration configuration = new Configuration();
        //configuration.setInt(FixedLengthInputFormat.FIXED_RECORD_LENGTH, 10);
        configuration.setInt("fixedlengthinputformat.record.length", 8);
        Job job = Job.getInstance(configuration, "Fixed length Input Format");

        job.setJarByClass(CombinedDriver.class);
        job.setInputFormatClass(CombineFileInputFormat.class);

        job.setMapperClass(CombinedMap.class);
        job.setCombinerClass(CombinedReducer.class);
        job.setReducerClass(CombinedReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
    }
}
```

Run Command:

```
hadoop jar /home/cloudera/Desktop/hw4/fixedLength.jar -D
mapreduce.input.fileinputformat.split.maxsize=55 CombinedFileInputFormat.CombinedDriver
/combinedInputOutput
```

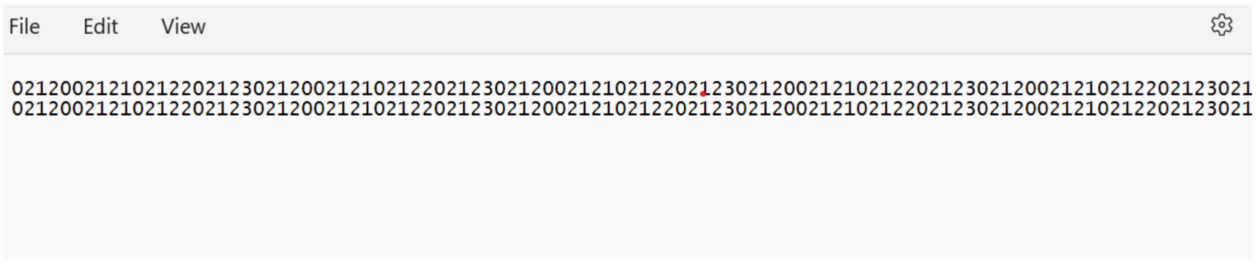
Output:



```
22/10/30 22:24:05 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
22/10/30 22:24:06 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
22/10/30 22:24:06 INFO input.FileInputFormat: Total input paths to process : 1
22/10/30 22:24:06 INFO mapreduce.JobSubmitter: number of splits:1
22/10/30 22:24:06 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1666182279730_0001
22/10/30 22:24:07 INFO impl.YarnClientImpl: Submitted application application_1666182279730_0001
22/10/30 22:24:07 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1666182279730_0001/
22/10/30 22:24:07 INFO mapreduce.Job: Running job: job_1666182279730_0001
22/10/30 22:24:15 INFO mapreduce.Job: Job job_1666182279730_0001 running in uber mode : false
22/10/30 22:24:15 INFO mapreduce.Job:  map 0% reduce 0%
22/10/30 22:24:28 INFO mapreduce.Job:  map 100% reduce 0%
22/10/30 22:24:36 INFO mapreduce.Job:  map 100% reduce 100%
22/10/30 22:24:37 INFO mapreduce.Job: Job job_1666182279730_0001 completed successfully
22/10/30 22:24:38 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=789513
                FILE: Number of bytes written=1660653
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=3497779
```

2. **FixedLengthInputFormat**

Input Data: Text file containing zipcodes without any separator and we try to find the count of each zipcode



```
0212002121021220212302120021210212202123021200212102122021230212002121021220212302120021210212202123021
0212002121021220212302120021210212202123021200212102122021230212002121021220212302120021210212202123021
```

FixedLengthInputFormat is an input format used to read input files which contain fixed-length records. The content of a record need not be text. It can be arbitrary binary data.

Mapper Class:

```
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import org.apache.hadoop.io.BytesWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class FixedLengthInputMap extends Mapper<Object, BytesWritable, Text, IntWritable> {
   IntWritable one = new IntWritable(1);
   public void map(Object key, BytesWritable value, Context context) throws IOException,
InterruptedException {
   Text word = new Text();
   word.set(new String(value.getBytes(), StandardCharsets.UTF_8));
```

```
    context.write(word, one);
}

}
```

Reducer Class:

```
package FixedLengthInputFormat;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class FixedLengthInputReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    protected void reduce(Text key, Iterable<IntWritable> values,
    Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException,
InterruptedException {
    IntWritable finalSum = new IntWritable(0);
    int sum = 0;
    for(IntWritable val:values) {
    sum += val.get();
}
    finalSum.set(sum);
    System.out.println("Reducing key: "+ key + " final sum: " + sum);

}
}
```

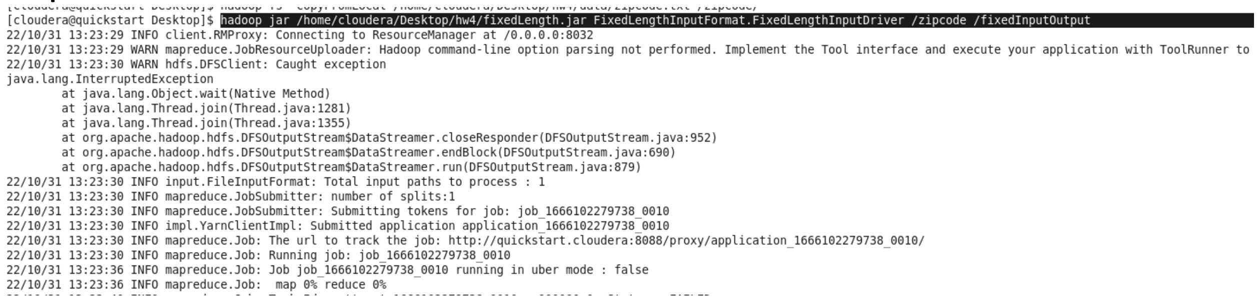
Driver Class:

```
package FixedLengthInputFormat;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.FixedLengthInputFormat;

public class FixedLengthInputDriver {
public static void main(String[] args) throws Exception {
    Configuration configuration = new Configuration();
    configuration.setInt("fixedlengthinputformat.record.length", 5);
    Job job = Job.getInstance(configuration, "Fixed length Input Format");
    job.setJarByClass(FixedLengthInputDriver.class);
    job.setInputFormatClass(FixedLengthInputFormat.class);
    job.setMapperClass(FixedLengthInputMap.class);
    job.setCombinerClass(FixedLengthInputReducer.class);
    job.setReducerClass(FixedLengthInputReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}

}
```

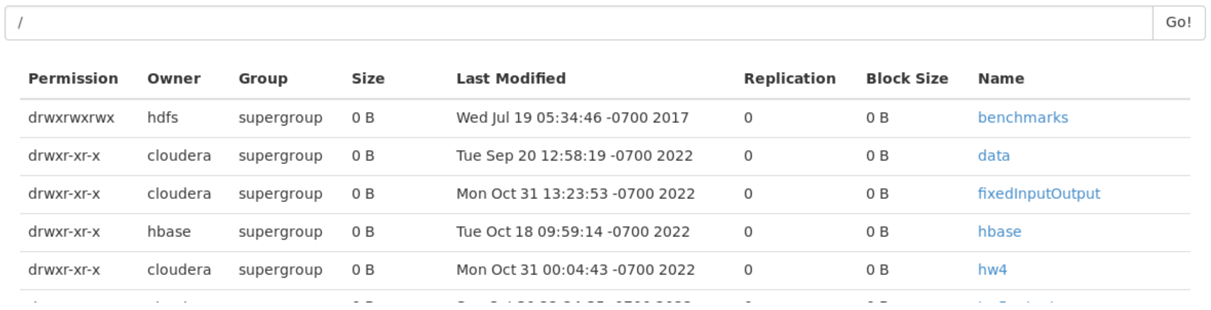
Run Command:

```
hadoop jar /home/cloudera/Desktop/hw4/fixedLength.jar
FixedLengthInputFormat.FixedLengthInputDriver /zipcode /fixedInputOutput
```

Output:



```
[cloudera@quickstart Desktop]$ hadoop jar /home/cloudera/Desktop/hw4/fixedLength.jar FixedLengthInputFormat.FixedLengthInputDriver /zipcode /fixedInputOutput
22/10/31 13:23:29 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
22/10/31 13:23:29 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to
22/10/31 13:23:30 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:952)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:690)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:879)
22/10/31 13:23:30 INFO input.FileInputFormat: Total input paths to process : 1
22/10/31 13:23:30 INFO mapreduce.JobSubmitter: number of splits:1
22/10/31 13:23:30 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1666102279738_0010
22/10/31 13:23:30 INFO impl.YarnClientImpl: Submitted application application_1666102279738_0010
22/10/31 13:23:30 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1666102279738_0010/
22/10/31 13:23:30 INFO mapreduce.Job: Running job: job_1666102279738_0010
22/10/31 13:23:36 INFO mapreduce.Job: Job job_1666102279738_0010 running in uber mode : false
22/10/31 13:23:36 INFO mapreduce.Job:  map 0% reduce 0%
```



/ Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxrwx	hdfs	supergroup	0 B	Wed Jul 19 05:34:46 -0700 2017	0	0 B	benchmarks
drwxr-xr-x	cloudera	supergroup	0 B	Tue Sep 20 12:58:19 -0700 2022	0	0 B	data
drwxr-xr-x	cloudera	supergroup	0 B	Mon Oct 31 13:23:53 -0700 2022	0	0 B	fixedInputOutput
drwxr-xr-x	hbase	supergroup	0 B	Tue Oct 18 09:59:14 -0700 2022	0	0 B	hbase
drwxr-xr-x	cloudera	supergroup	0 B	Mon Oct 31 00:04:43 -0700 2022	0	0 B	hw4

3. KeyValueTextInputFormat

An InputFormat for plain text files. Files are broken into lines. Either line feed or carriage-return are used to signal end of line. Each line is divided into key and value parts by a separator byte. If no such a byte exists, the key will be the entire line and value will be empty. The separator byte can be specified in config file under the attribute name mapreduce.input.keyvaluelinerecordreader.key.value.separator. The default is the tab character ('\t').

Mapper Class:

```
public class KeyValueInputFormatMap extends Mapper<Text, Text, Text,
IntWritable> {
public void map(Text key, Text value, Context context) throws
IOException, InterruptedException {
if (key.toString().length() == 1 || value.toString().length() == 1 )
{
return;
} else {
IntWritable count = new
IntWritable(Integer.parseInt(value.toString()));
context.write(key, count);
}
}
}
```

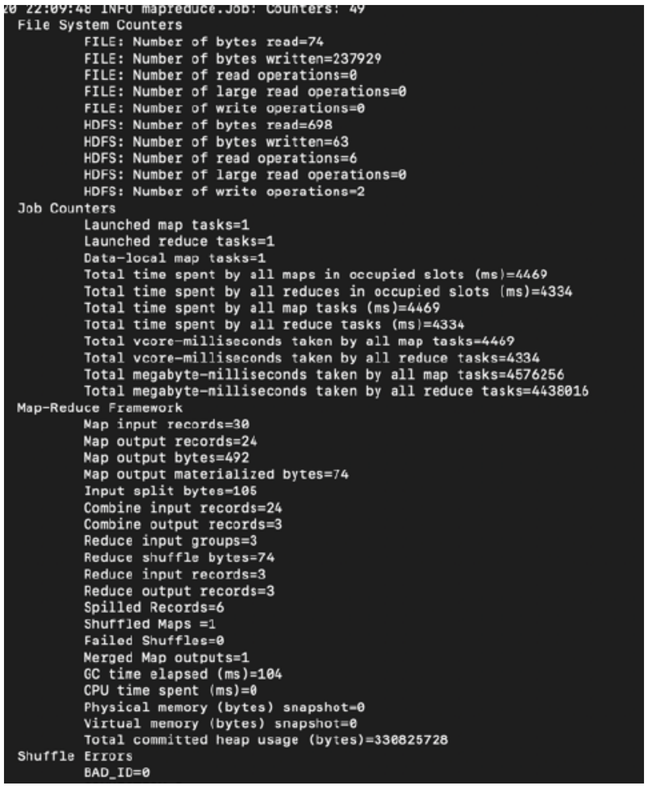
Reducer Class:

```
public class KeyValueInputReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
  protected void reduce(Text key, Iterable<IntWritable> values,
  Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws
  IOException, InterruptedException {
  IntWritable finalSum = new IntWritable(0);
  int sum = 0;
  for(IntWritable val:values) {
  sum += val.get();
}
finalSum.set(sum);
System.out.println("Reducing key: "+ key + " final sum: " + sum);
context.write(key, finalSum);
}

}
```

Driver Class:

```
public class KeyValueInputFormatDriver {
    public static void main(String[] args) throws Exception {
    Configuration configuration = new Configuration();
    configuration.set("mapreduce.input.keyvaluelinerecordreader.key.value.separat
    or", " ");
    Job job = Job.getInstance(configuration, "Key value Input Format");
    job.setJarByClass(KeyValueInputFormatDemoMapper.class);
    job.setInputFormatClass(KeyValueTextInputFormat.class);
```

```
        job.setMapperClass(KeyValueInputFormatDemoMapper.class);
        job.setCombinerClass(KeyValueInputFormatDemoReducer.class);
        job.setReducerClass(KeyValueInputFormatDemoReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
}

}
```



```
20 22:09:48 INFO mapreduce.Job: Counters: 49
 File System Counters
         FILE: Number of bytes read=74
         FILE: Number of bytes written=237929
         FILE: Number of read operations=0
         FILE: Number of large read operations=0
         FILE: Number of write operations=0
         HDFS: Number of bytes read=698
         HDFS: Number of bytes written=63
         HDFS: Number of read operations=6
         HDFS: Number of large read operations=0
         HDFS: Number of write operations=2
 Job Counters
         Launched map tasks=1
         Launched reduce tasks=1
         Data-local map tasks=1
         Total time spent by all maps in occupied slots (ms)=4469
         Total time spent by all reduces in occupied slots (ms)=4334
         Total time spent by all map tasks (ms)=4469
         Total time spent by all reduce tasks (ms)=4334
         Total vcore-milliseconds taken by all map tasks=4469
         Total vcore-milliseconds taken by all reduce tasks=4334
         Total megabyte-milliseconds taken by all map tasks=4576256
         Total megabyte-milliseconds taken by all reduce tasks=4438016
 Map-Reduce Framework
         Map input records=30
         Map output records=24
         Map output bytes=492
         Map output materialized bytes=74
         Input split bytes=105
         Combine input records=24
         Combine output records=3
         Reduce input groups=3
         Reduce shuffle bytes=74
         Reduce input records=3
         Reduce output records=3
         Spilled Records=6
         Shuffled Maps =1
         Failed Shuffles=0
         Merged Map outputs=1
         GC time elapsed (ms)=104
         CPU time spent (ms)=0
         Physical memory (bytes) snapshot=0
         Virtual memory (bytes) snapshot=0
         Total committed heap usage (bytes)=330825728
 Shuffle Errors
         BAD_ID=0
```

4. **NLineInputFormat**

NLineInputFormat which splits N lines of input as one split.

Mapper Class:

```
public class CombinedMyFormat extends CombineFileInputFormat<LongWritable,Text>{

            @Override
   public RecordReader<LongWritable, Text>
         createRecordReader(InputSplit split, TaskAttemptContext context)
               throws IOException {
```

```
    CombineFileRecordReader<LongWritable, Text> reader =
          new CombineFileRecordReader<LongWritable, Text>(
                (CombineFileSplit) split, context, myCombineFileRecordReader.class);
    return reader;
}

public static class myCombineFileRecordReader extends RecordReader<LongWritable, Text> {
    private LineRecordReader lineRecordReader = new LineRecordReader();

    public myCombineFileRecordReader(CombineFileSplit split,
          TaskAttemptContext context, Integer index) throws IOException {

        FileSplit fileSplit = new FileSplit(split.getPath(index),
                                   split.getOffset(index),
                                   split.getLength(index),
                                   split.getLocations());
        lineRecordReader.initialize(fileSplit, context);
    }

    @Override
    public void initialize(InputSplit inputSplit, TaskAttemptContext context)
          throws IOException, InterruptedException {
        //linerecordReader.initialize(inputSplit, context);
    }

    @Override
    public void close() throws IOException {
        lineRecordReader.close();
    }

    @Override
    public float getProgress() throws IOException {
        return lineRecordReader.getProgress();
    }

    @Override
    public LongWritable getCurrentKey() throws IOException,
          InterruptedException {
        return lineRecordReader.getCurrentKey();
    }

    @Override
    public Text getCurrentValue() throws IOException, InterruptedException {
        return lineRecordReader.getCurrentValue();
    }
```

```
        @Override
        public boolean nextKeyValue() throws IOException, InterruptedException {
            return lineRecordReader.nextKeyValue();
        }
    }

}
```

Driver Class:

```
public class InputOutputDriver {

            public static void main(String args[]) throws IllegalArgumentException, IOException,
ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();
        conf.set("mapreduce.input.keyvaluelinerecordreader.key.value.separator", " ");

        Job job = new Job(conf);
        Job job = Job.getInstance();
            job.getConfiguration().setInt("mapreduce.input.lineinputformat.linespermap", 4);
        job.setJarByClass(HitMain.class);

        // Specify various job-specific parameters
        job.setMapperClass(HitCounter.class);
        job.setReducerClass(HitReducer.class);

        job.setInputFormatClass(NLineInputFormat.class);
}
```

Mapper Class:

```
public class InputOutputMapper extends Mapper<LongWritable,Text,Text,IntWritable>{ //-
NLineInputFormat

            private final static IntWritable one = new IntWritable(1);
            private Text ipaddress = new Text();

            public void map(BytesWritable key, Text value,
org.apache.hadoop.mapreduce.Mapper.Context context) throws IOException, InterruptedException
{ //- NLineInputFormat

                              //String line=value.toString();
                              String line = new String(value.get(),StandardCharsets.UTF_8);

                              String[] tokens=line.split(" ");
                              ipaddress.set(tokens[0]);
```

```
                                        context.write(ipaddress, one);
                                        //context gives output key and value
                                }
        }
```

5. **SequenceFileInputFormat**

   An InputFormat for SequenceFiles.

6. **TextInputFormat**

   An InputFormat for plain text files. Files are broken into lines. Either linefeed or carriage-return are used to signal end of line. Keys are the position in the file, and values are the line of text.

Mapper Class:

```
public class MaxStockPriceMap extends Mapper<Object, Text, Text, DoubleWritable> {

    protected void map(Object key, Text value, Mapper<Object, Text, Text,

    DoubleWritable>.Context context) throws IOException, InterruptedException {

    String[] tokens = value.toString().split(",");

    Text stockSymbol = new Text();

    stockSymbol.set(tokens[1]);

    DoubleWritable price = new DoubleWritable();

    Double stock_price_high_value = 0.0;

  try {

    stock_price_high_value = Double.parseDouble(tokens[4]);

    price.set(stock_price_high_value);

  }

  catch (NumberFormatException nf) {

  System.out.println("Continue without interruption, its the header line");

  }

  context.write(stockSymbol, price);

  }


}
```

Reducer Class:

```
public class MaxStockPriceReduce extends Reducer<Text, DoubleWritable, Text,DoubleWritable> {
  public void reduce(Text key, Iterable<DoubleWritable> values, Context context)
  throws IOException, InterruptedException {
  double final_stock_high = Double.MIN_VALUE;
  for (DoubleWritable val: values) {
  final_stock_high = Math.max(val.get(), final_stock_high);
  }
  DoubleWritable final_stock_price_red = new DoubleWritable(final_stock_high);
  context.write(key, final_stock_price_red);
}


}
```

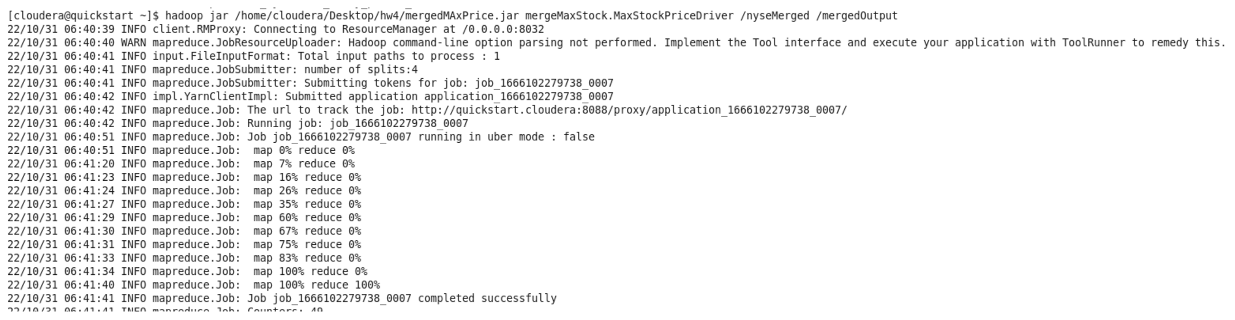
Driver Class:

```
public class MaxStockPriceDriver {
public static void main(String[] args) throws Exception {
    Configuration configuration = new Configuration();
    Job job = Job.getInstance(configuration, "Calculating max stock price high");
    job.setJarByClass(MaxStockPriceDriver.class);
    job.setMapperClass(MaxStockPriceMap.class);
    job.setCombinerClass(MaxStockPriceReduce.class);
    job.setReducerClass(MaxStockPriceReduce.class);


    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(DoubleWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

}

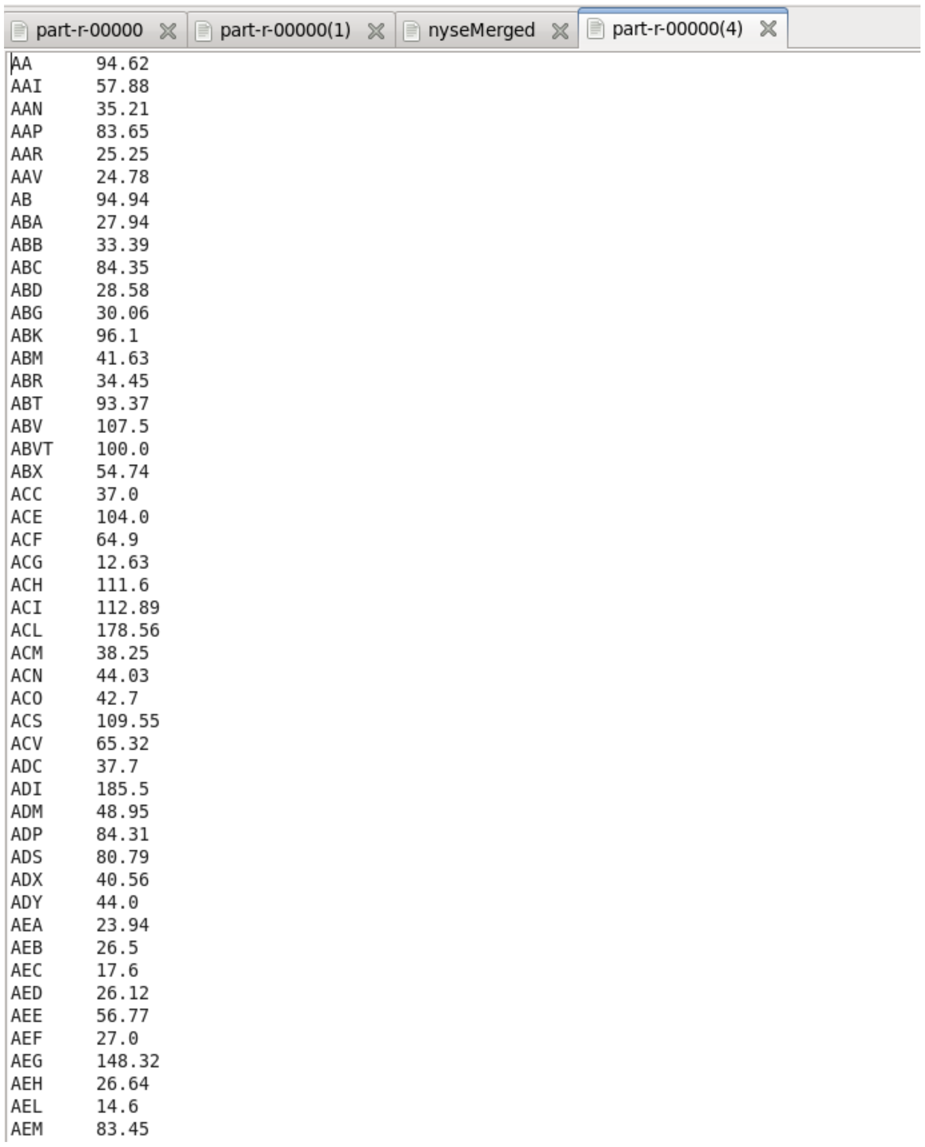
Run Command:

hadoop jar /home/cloudera/Desktop/hw4/mergedMAxPrice.jar mergeMaxStock.MaxStockPriceDriver /nyseMerged /mergedOutput



```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/Desktop/hw4/mergedMAxPrice.jar mergeMaxStock.MaxStockPriceDriver /nyseMerged /mergedOutput
22/10/31 06:40:39 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
22/10/31 06:40:40 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
22/10/31 06:40:41 INFO input.FileInputFormat: Total input paths to process : 1
22/10/31 06:40:41 INFO mapreduce.JobSubmitter: number of splits:4
22/10/31 06:40:41 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1666102279738_0007
22/10/31 06:40:42 INFO impl.YarnClientImpl: Submitted application application_1666102279738_0007
22/10/31 06:40:42 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1666102279738_0007/
22/10/31 06:40:42 INFO mapreduce.Job: Running job: job_1666102279738_0007
22/10/31 06:40:51 INFO mapreduce.Job: Job job_1666102279738_0007 running in uber mode : false
22/10/31 06:40:51 INFO mapreduce.Job:  map 0% reduce 0%
22/10/31 06:41:20 INFO mapreduce.Job:  map 7% reduce 0%
22/10/31 06:41:23 INFO mapreduce.Job:  map 16% reduce 0%
22/10/31 06:41:24 INFO mapreduce.Job:  map 26% reduce 0%
22/10/31 06:41:27 INFO mapreduce.Job:  map 35% reduce 0%
22/10/31 06:41:29 INFO mapreduce.Job:  map 60% reduce 0%
22/10/31 06:41:30 INFO mapreduce.Job:  map 67% reduce 0%
22/10/31 06:41:31 INFO mapreduce.Job:  map 75% reduce 0%
22/10/31 06:41:33 INFO mapreduce.Job:  map 83% reduce 0%
22/10/31 06:41:34 INFO mapreduce.Job:  map 100% reduce 0%
22/10/31 06:41:40 INFO mapreduce.Job:  map 100% reduce 100%
22/10/31 06:41:41 INFO mapreduce.Job: Job job_1666102279738_0007 completed successfully
22/10/31 06:41:41 INFO mapreduce.Job: Counters: 49
```

Merged Output:



part-r-00000 | part-r-00000(1) | nyseMerged | part-r-00000(4)

```
AA      94.62
AAI     57.88
AAN     35.21
AAP     83.65
AAR     25.25
AAV     24.78
AB      94.94
ABA     27.94
ABB     33.39
ABC     84.35
ABD     28.58
ABG     30.06
ABK     96.1
ABM     41.63
ABR     34.45
ABT     93.37
ABV     107.5
ABVT    100.0
ABX     54.74
ACC     37.0
ACE     104.0
ACF     64.9
ACG     12.63
ACH     111.6
ACI     112.89
ACL     178.56
ACM     38.25
ACN     44.03
ACO     42.7
ACS     109.55
ACV     65.32
ADC     37.7
ADI     185.5
ADM     48.95
ADP     84.31
ADS     80.79
ADX     40.56
ADY     44.0
AEA     23.94
AEB     26.5
AEC     17.6
AED     26.12
AEE     56.77
AEF     27.0
AEG     148.32
AEH     26.64
AEL     14.6
AEM     83.45
```

Part-5: Programming Assignment

Create a Writable object that stores some fields from the the NYSE dataset to find
- the date of the max stock_volume
- the date of the min stock_volume
- the max stock_price_adj_close
This will be a custom writable class with the above fields.
Mapper will use this writable object as a value, and Reducer will use this writable object as a value.

Writable Class:

```
public class NyseStock implements Writable {

    private Text maxStockVolDate;

    private Text minStockVolDate;
    private DoubleWritable maxstockVol;
    private DoubleWritable minstockVol;
    private DoubleWritable stock_price_adj_close;

    public NyseStock(Text maxStockVolumeDate, Text
    minStockVolumeDate, DoubleWritable maxstockVolume,DoubleWritable
    minstockVolume, DoubleWritable stock_price_adj_close) {
    this.maxStockVolDate = maxStockVolumeDate;
    this.minStockVolDate = minStockVolumeDate;
    this.maxstockVol = maxstockVolume;
    this.minstockVol = minstockVolume;
    this.stock_price_adj_close = stock_price_adj_close;
}
public NyseStock() {
    this.maxStockVolDate = new Text();
    this.minStockVolDate = new Text();
    this.maxstockVol = new DoubleWritable();
    this.minstockVol = new DoubleWritable();
    this.stock_price_adj_close = new DoubleWritable();
}
public Text getMaxStockVolDate() {
    return maxStockVolDate;
}
public void setMaxStockVolDate(Text maxStockVolumeDate) {
    this.maxStockVolDate = maxStockVolumeDate;
}

public Text getMinStockVolDate() {
    return minStockVolDate;
}

public void setMinStockVolDate(Text minStockVolumeDate) {
   this.minStockVolDate = minStockVolumeDate;
}
public DoubleWritable getMaxstockVol() {
   return maxstockVol;
}

public void setMaxstockVol(DoubleWritable stockVolume) {
   this.maxstockVol = stockVolume;
}

public DoubleWritable getMinstockVol() {
   return minstockVol;
}

public void setMinstockVol(DoubleWritable stockVolume) {
   this.minstockVol = stockVolume;
}

public DoubleWritable getStock_price_adj_close() {
   return stock_price_adj_close;
}

public void setStock_price_adj_close(DoubleWritable stock_price_adj_close) {
   this.stock_price_adj_close = stock_price_adj_close;
}
```

```
@Override
public void write(DataOutput dataOutput) throws IOException {
   minStockVolDate.write(dataOutput);
   maxStockVolDate.write(dataOutput);
   stock_price_adj_close.write(dataOutput);
   maxstockVol.write(dataOutput);
   minstockVol.write(dataOutput);
}

@Override
public void readFields(DataInput dataInput) throws IOException {
   minStockVolDate.readFields(dataInput);
   maxStockVolDate.readFields(dataInput);
   stock_price_adj_close.readFields(dataInput);
   maxstockVol.readFields(dataInput);
   minstockVol.readFields(dataInput);
}

@Override
public String toString() {
return "NyseStockWritable{" +
   "maxStockVolumeDate: " + maxStockVolDate +
   ", minStockVolumeDate: " + minStockVolDate +
   ", maxstockVolume: " + maxstockVol +
   ", minstockVolume: " + minstockVol +
   ", stock_price_adj_close: " + stock_price_adj_close +
   '}';
   }
}
```

Mapper Class:

```
public class NyseStockMap extends Mapper<Object, Text, Text,NyseStock> {

protected void map(Object key, Text value, Mapper<Object, Text, Text, NyseStock>.Context context)

throws IOException, InterruptedException

{

    String[] tokens = value.toString().split(",");

    Text stockSymbol = new Text();

    stockSymbol.set(tokens[1]);

    Text maxStockVolumeDate = new Text(tokens[2]);


try {

    DoubleWritable stockVolume = new

    DoubleWritable(Double.parseDouble(tokens[7]));

    DoubleWritable stock_price_adj_close = new

    DoubleWritable(Double.parseDouble(tokens[8]));

    NyseStock nyseStockWritable = new

    NyseStock(maxStockVolumeDate,maxStockVolumeDate, stockVolume,

    stockVolume, stock_price_adj_close);

    context.write(stockSymbol, nyseStockWritable);

}


catch (NumberFormatException nf) {

    System.out.println("Continue without interruption, its the headerline");

}

}


}
```

Reducer Class:

```
public class NyseStockReduce extends Reducer<Text, NyseStock, Text,NyseStock> {
    public void reduce(Text key, Iterable<NyseStock> values, Context context)
    throws IOException, InterruptedException {
    Text maxDate = new Text();
    Text minDate = new Text();
    DoubleWritable maxStock = new DoubleWritable(Double.MIN_VALUE);
    DoubleWritable minStock = new DoubleWritable(Double.MAX_VALUE);
    DoubleWritable stockClosePrice = new
    DoubleWritable(Double.MIN_VALUE);
    for (NyseStock val: values) {
        if (val.getMaxstockVol().get() > maxStock.get()) {
        maxStock.set(val.getMaxstockVol().get());
        maxDate.set(val.getMaxStockVolDate().toString());
  }

  if (val.getMinstockVol().get() < minStock.get()) {
        minStock.set(val.getMinstockVol().get());
        minDate.set(val.getMinStockVolDate().toString());
  }

  if (val.getStock_price_adj_close().get() > stockClosePrice.get())
  {
    stockClosePrice.set(val.getStock_price_adj_close().get());
    }


  }
    NyseStock nyseStockWritable = new NyseStock(maxDate,
    minDate, maxStock, minStock, stockClosePrice);
    context.write(key, nyseStockWritable);
    }
```

```
}
```

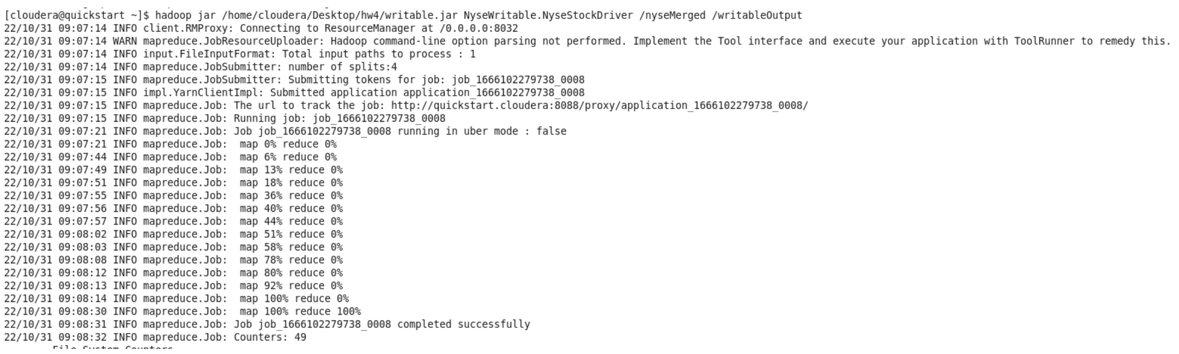
Driver Class:

```
public class NyseStockDriver {

    public static void main(String[] args) throws IOException,InterruptedException,
ClassNotFoundException {

    Configuration configuration = new Configuration();

    Job job = Job.getInstance(configuration, "Custom Writable");

    job.setJarByClass(NyseStockDriver.class);

    job.setMapperClass(NyseStockMap.class);

    job.setReducerClass(NyseStockReduce.class);


    job.setOutputKeyClass(Text.class);

    job.setOutputValueClass(NyseStock.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));

    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    System.exit(job.waitForCompletion(true) ? 0 : 1);


  }

}
```

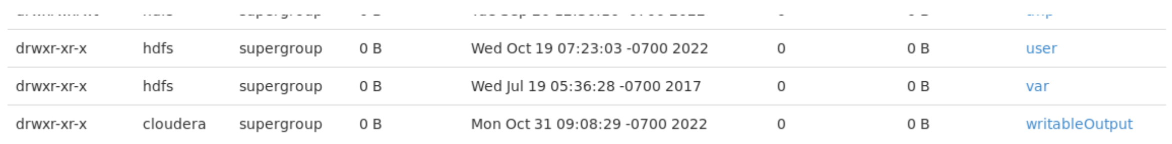
Run Command:

```
hadoop jar /home/cloudera/Desktop/hw4/writable.jar NyseWritable.NyseStockDriver /nyseMerged
/writableOutput
```

Run Time: 1 minute 19 seconds

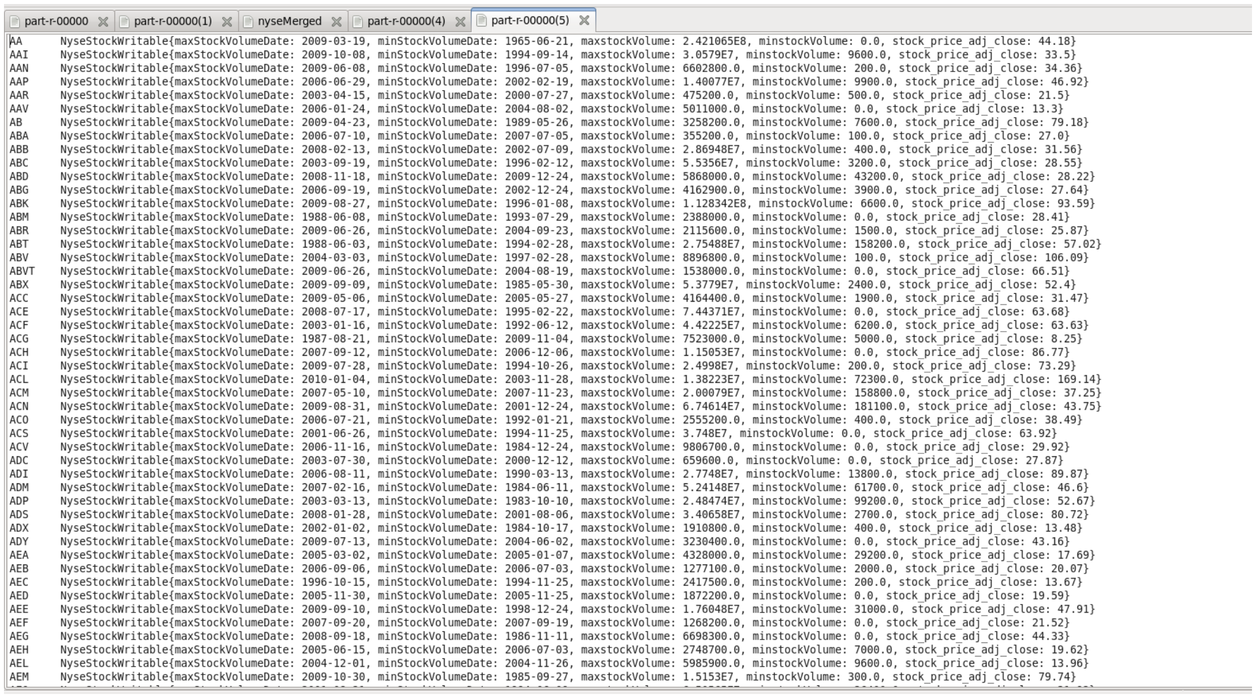


```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/Desktop/hw4/writable.jar NyseWritable.NyseStockDriver /nyseMerged /writableOutput
22/10/31 09:07:14 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
22/10/31 09:07:14 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
22/10/31 09:07:14 INFO input.FileInputFormat: Total input paths to process : 1
22/10/31 09:07:14 INFO mapreduce.JobSubmitter: number of splits:4
22/10/31 09:07:15 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1666102279738_0008
22/10/31 09:07:15 INFO impl.YarnClientImpl: Submitted application application_1666102279738_0008
22/10/31 09:07:15 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1666102279738_0008/
22/10/31 09:07:15 INFO mapreduce.Job: Running job: job_1666102279738_0008
22/10/31 09:07:21 INFO mapreduce.Job: Job job_1666102279738_0008 running in uber mode : false
22/10/31 09:07:21 INFO mapreduce.Job:  map 0% reduce 0%
22/10/31 09:07:44 INFO mapreduce.Job:  map 6% reduce 0%
22/10/31 09:07:49 INFO mapreduce.Job:  map 13% reduce 0%
22/10/31 09:07:51 INFO mapreduce.Job:  map 18% reduce 0%
22/10/31 09:07:55 INFO mapreduce.Job:  map 36% reduce 0%
22/10/31 09:07:56 INFO mapreduce.Job:  map 40% reduce 0%
22/10/31 09:07:57 INFO mapreduce.Job:  map 44% reduce 0%
22/10/31 09:08:02 INFO mapreduce.Job:  map 51% reduce 0%
22/10/31 09:08:03 INFO mapreduce.Job:  map 58% reduce 0%
22/10/31 09:08:08 INFO mapreduce.Job:  map 78% reduce 0%
22/10/31 09:08:12 INFO mapreduce.Job:  map 80% reduce 0%
22/10/31 09:08:13 INFO mapreduce.Job:  map 92% reduce 0%
22/10/31 09:08:14 INFO mapreduce.Job:  map 100% reduce 0%
22/10/31 09:08:30 INFO mapreduce.Job:  map 100% reduce 100%
22/10/31 09:08:31 INFO mapreduce.Job: Job job_1666102279738_0008 completed successfully
22/10/31 09:08:32 INFO mapreduce.Job: Counters: 49
```



drwxr-xr-x	hdfs	supergroup	0 B	Wed Oct 19 07:23:03 -0700 2022	0	0 B	user
drwxr-xr-x	hdfs	supergroup	0 B	Wed Jul 19 05:36:28 -0700 2017	0	0 B	var
drwxr-xr-x	cloudera	supergroup	0 B	Mon Oct 31 09:08:29 -0700 2022	0	0 B	writableOutput

Output:



```
AA      NyseStockWritable{maxStockVolumeDate: 2009-03-19, minStockVolumeDate: 1965-06-21, maxstockVolume: 2.421065E8, minstockVolume: 0.0, stock_price_adj_close: 44.18}
AAI     NyseStockWritable{maxStockVolumeDate: 2009-10-08, minStockVolumeDate: 1994-09-14, maxstockVolume: 3.0579E7, minstockVolume: 9600.0, stock_price_adj_close: 33.5}
AAN     NyseStockWritable{maxStockVolumeDate: 2009-06-08, minStockVolumeDate: 1996-07-05, maxstockVolume: 6602800.0, minstockVolume: 200.0, stock_price_adj_close: 34.36}
AAP     NyseStockWritable{maxStockVolumeDate: 2006-06-29, minStockVolumeDate: 2002-02-19, maxstockVolume: 1.40077E7, minstockVolume: 9900.0, stock_price_adj_close: 46.92}
AAR     NyseStockWritable{maxStockVolumeDate: 2003-04-15, minStockVolumeDate: 2000-07-27, maxstockVolume: 475200.0, minstockVolume: 500.0, stock_price_adj_close: 21.5}
AAV     NyseStockWritable{maxStockVolumeDate: 2006-01-24, minStockVolumeDate: 2004-08-02, maxstockVolume: 5011000.0, minstockVolume: 0.0, stock_price_adj_close: 13.3}
AB      NyseStockWritable{maxStockVolumeDate: 2009-04-23, minStockVolumeDate: 1989-05-26, maxstockVolume: 3258200.0, minstockVolume: 7600.0, stock_price_adj_close: 79.18}
ABA     NyseStockWritable{maxStockVolumeDate: 2006-07-10, minStockVolumeDate: 2007-07-05, maxstockVolume: 355200.0, minstockVolume: 100.0, stock_price_adj_close: 27.0}
ABB     NyseStockWritable{maxStockVolumeDate: 2008-02-13, minStockVolumeDate: 2002-07-09, maxstockVolume: 2.86948E7, minstockVolume: 400.0, stock_price_adj_close: 31.56}
ABC     NyseStockWritable{maxStockVolumeDate: 2003-09-19, minStockVolumeDate: 1996-02-12, maxstockVolume: 5.5356E7, minstockVolume: 3200.0, stock_price_adj_close: 28.55}
ABD     NyseStockWritable{maxStockVolumeDate: 2008-11-18, minStockVolumeDate: 2009-12-24, maxstockVolume: 5868000.0, minstockVolume: 43200.0, stock_price_adj_close: 28.22}
ABG     NyseStockWritable{maxStockVolumeDate: 2006-09-19, minStockVolumeDate: 2002-12-24, maxstockVolume: 4162900.0, minstockVolume: 3900.0, stock_price_adj_close: 27.64}
ABK     NyseStockWritable{maxStockVolumeDate: 2009-08-27, minStockVolumeDate: 1996-01-08, maxstockVolume: 1.128342E8, minstockVolume: 6600.0, stock_price_adj_close: 93.59}
ABM     NyseStockWritable{maxStockVolumeDate: 1988-06-08, minStockVolumeDate: 1993-07-29, maxstockVolume: 2388000.0, minstockVolume: 0.0, stock_price_adj_close: 28.41}
ABR     NyseStockWritable{maxStockVolumeDate: 2009-06-26, minStockVolumeDate: 2004-09-23, maxstockVolume: 2115600.0, minstockVolume: 1500.0, stock_price_adj_close: 25.87}
ABT     NyseStockWritable{maxStockVolumeDate: 1988-06-03, minStockVolumeDate: 1994-02-28, maxstockVolume: 2.75488E7, minstockVolume: 158200.0, stock_price_adj_close: 57.02}
ABV     NyseStockWritable{maxStockVolumeDate: 2004-03-03, minStockVolumeDate: 1997-02-28, maxstockVolume: 8896800.0, minstockVolume: 100.0, stock_price_adj_close: 106.09}
ABVT    NyseStockWritable{maxStockVolumeDate: 2009-06-26, minStockVolumeDate: 2004-08-19, maxstockVolume: 1538000.0, minstockVolume: 0.0, stock_price_adj_close: 66.51}
ABX     NyseStockWritable{maxStockVolumeDate: 2009-09-09, minStockVolumeDate: 1985-05-30, maxstockVolume: 5.3779E7, minstockVolume: 2400.0, stock_price_adj_close: 52.4}
ACC     NyseStockWritable{maxStockVolumeDate: 2009-05-06, minStockVolumeDate: 2005-05-27, maxstockVolume: 4164400.0, minstockVolume: 1900.0, stock_price_adj_close: 31.47}
ACE     NyseStockWritable{maxStockVolumeDate: 2008-07-17, minStockVolumeDate: 1995-02-22, maxstockVolume: 7.44371E7, minstockVolume: 0.0, stock_price_adj_close: 63.68}
ACF     NyseStockWritable{maxStockVolumeDate: 2003-01-16, minStockVolumeDate: 1992-06-12, maxstockVolume: 4.42225E7, minstockVolume: 6200.0, stock_price_adj_close: 63.63}
ACG     NyseStockWritable{maxStockVolumeDate: 1987-08-21, minStockVolumeDate: 2009-11-04, maxstockVolume: 7523000.0, minstockVolume: 5000.0, stock_price_adj_close: 8.25}
ACH     NyseStockWritable{maxStockVolumeDate: 2007-09-12, minStockVolumeDate: 2006-12-06, maxstockVolume: 1.15053E7, minstockVolume: 0.0, stock_price_adj_close: 86.77}
ACI     NyseStockWritable{maxStockVolumeDate: 2009-07-28, minStockVolumeDate: 1994-10-26, maxstockVolume: 2.4998E7, minstockVolume: 200.0, stock_price_adj_close: 73.29}
ACL     NyseStockWritable{maxStockVolumeDate: 2010-01-04, minStockVolumeDate: 2003-11-28, maxstockVolume: 1.38223E7, minstockVolume: 72300.0, stock_price_adj_close: 169.14}
ACM     NyseStockWritable{maxStockVolumeDate: 2007-05-10, minStockVolumeDate: 2007-11-23, maxstockVolume: 2.00079E7, minstockVolume: 158800.0, stock_price_adj_close: 37.25}
ACN     NyseStockWritable{maxStockVolumeDate: 2009-08-31, minStockVolumeDate: 2001-12-24, maxstockVolume: 6.74614E7, minstockVolume: 181100.0, stock_price_adj_close: 43.75}
ACO     NyseStockWritable{maxStockVolumeDate: 2006-07-21, minStockVolumeDate: 1992-01-21, maxstockVolume: 2555200.0, minstockVolume: 400.0, stock_price_adj_close: 38.49}
ACS     NyseStockWritable{maxStockVolumeDate: 2001-06-26, minStockVolumeDate: 1994-11-25, maxstockVolume: 3.748E7, minstockVolume: 0.0, stock_price_adj_close: 63.92}
ACV     NyseStockWritable{maxStockVolumeDate: 2006-11-16, minStockVolumeDate: 1984-12-24, maxstockVolume: 9806700.0, minstockVolume: 0.0, stock_price_adj_close: 29.92}
ADC     NyseStockWritable{maxStockVolumeDate: 2003-07-30, minStockVolumeDate: 2000-12-12, maxstockVolume: 659600.0, minstockVolume: 0.0, stock_price_adj_close: 27.87}
ADI     NyseStockWritable{maxStockVolumeDate: 2006-08-11, minStockVolumeDate: 1990-03-13, maxstockVolume: 2.7748E7, minstockVolume: 13800.0, stock_price_adj_close: 89.87}
ADM     NyseStockWritable{maxStockVolumeDate: 2007-02-16, minStockVolumeDate: 1984-06-11, maxstockVolume: 5.24148E7, minstockVolume: 61700.0, stock_price_adj_close: 46.6}
ADP     NyseStockWritable{maxStockVolumeDate: 2003-03-13, minStockVolumeDate: 1983-10-10, maxstockVolume: 2.48474E7, minstockVolume: 99200.0, stock_price_adj_close: 52.67}
ADS     NyseStockWritable{maxStockVolumeDate: 2008-01-28, minStockVolumeDate: 2001-08-06, maxstockVolume: 3.40658E7, minstockVolume: 2700.0, stock_price_adj_close: 80.72}
ADX     NyseStockWritable{maxStockVolumeDate: 2002-01-02, minStockVolumeDate: 1984-10-17, maxstockVolume: 1910800.0, minstockVolume: 400.0, stock_price_adj_close: 13.48}
ADY     NyseStockWritable{maxStockVolumeDate: 2009-07-13, minStockVolumeDate: 2004-06-02, maxstockVolume: 3230400.0, minstockVolume: 0.0, stock_price_adj_close: 43.16}
AEA     NyseStockWritable{maxStockVolumeDate: 2005-03-02, minStockVolumeDate: 2005-01-07, maxstockVolume: 4328000.0, minstockVolume: 29200.0, stock_price_adj_close: 17.69}
AEB     NyseStockWritable{maxStockVolumeDate: 2006-09-06, minStockVolumeDate: 2006-07-03, maxstockVolume: 1277100.0, minstockVolume: 2000.0, stock_price_adj_close: 20.07}
AEC     NyseStockWritable{maxStockVolumeDate: 1996-10-15, minStockVolumeDate: 1994-11-25, maxstockVolume: 2417500.0, minstockVolume: 200.0, stock_price_adj_close: 13.67}
AED     NyseStockWritable{maxStockVolumeDate: 2005-11-30, minStockVolumeDate: 2005-11-25, maxstockVolume: 1872200.0, minstockVolume: 0.0, stock_price_adj_close: 19.59}
AEE     NyseStockWritable{maxStockVolumeDate: 2009-09-10, minStockVolumeDate: 1998-12-24, maxstockVolume: 1.76048E7, minstockVolume: 31000.0, stock_price_adj_close: 47.91}
AEF     NyseStockWritable{maxStockVolumeDate: 2007-09-20, minStockVolumeDate: 2007-09-19, maxstockVolume: 1268200.0, minstockVolume: 0.0, stock_price_adj_close: 21.52}
AEG     NyseStockWritable{maxStockVolumeDate: 2008-09-18, minStockVolumeDate: 1986-11-11, maxstockVolume: 6698300.0, minstockVolume: 0.0, stock_price_adj_close: 44.33}
AEH     NyseStockWritable{maxStockVolumeDate: 2005-06-15, minStockVolumeDate: 2006-07-03, maxstockVolume: 2748700.0, minstockVolume: 7000.0, stock_price_adj_close: 19.62}
AEL     NyseStockWritable{maxStockVolumeDate: 2004-12-01, minStockVolumeDate: 2004-11-26, maxstockVolume: 5985900.0, minstockVolume: 9600.0, stock_price_adj_close: 13.96}
AEM     NyseStockWritable{maxStockVolumeDate: 2009-10-30, minStockVolumeDate: 1985-09-27, maxstockVolume: 1.5153E7, minstockVolume: 300.0, stock_price_adj_close: 79.74}
```

Part-6: Programming Assignment

Redo Part5 of this assignment, but cram multiple values (max stock_volume, min stock_volume, max stock_price_adj_close) into a Text object with some delimiter. Use a Combiner. Compare the running time of Part 2 to Part 3. You could measure the running time programmatically, or use your smartphone's timer.

Driver Class with Combiner:

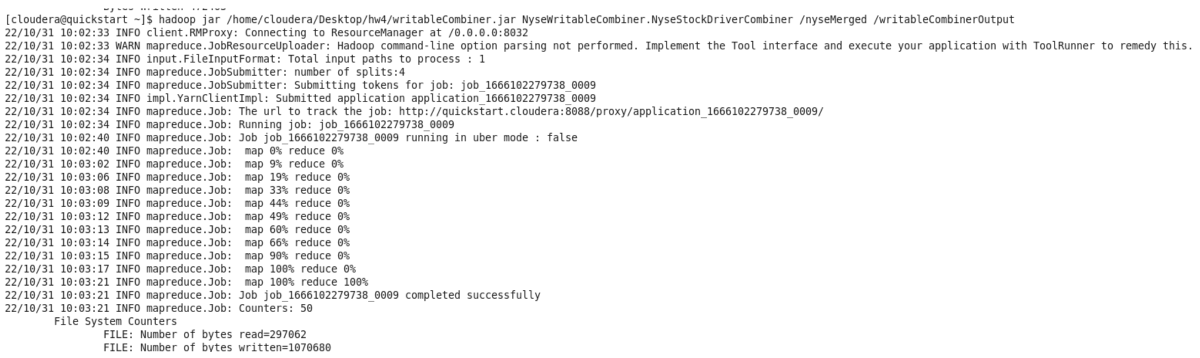
public class NyseStockDriverCombiner {

```
public static void main(String[] args) throws IOException,InterruptedException, ClassNotFoundException
{
    Configuration configuration = new Configuration();
    Job job = Job.getInstance(configuration, "Custom Writable");
    job.setJarByClass(NyseStockDriver.class);
    job.setMapperClass(NyseWritable.NyseStockMap.class);
    job.setCombinerClass(NyseWritable.NyseStockReduce.class);
    job.setReducerClass(NyseWritable.NyseStockReduce.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(NyseWritable.NyseStock.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

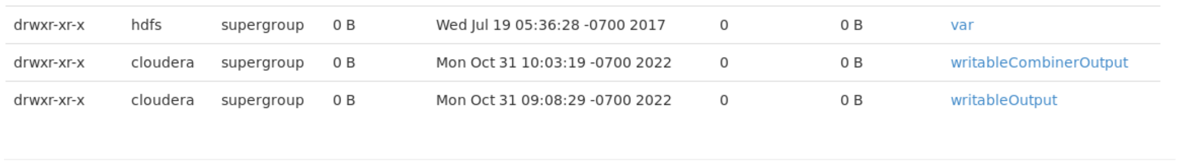
Run Command:

```
hadoop jar /home/cloudera/Desktop/hw4/writableCombiner.jar
NyseWritableCombiner.NyseStockDriverCombiner /nyseMerged /writableCombinerOutput
```

Run Time: 50 seconds

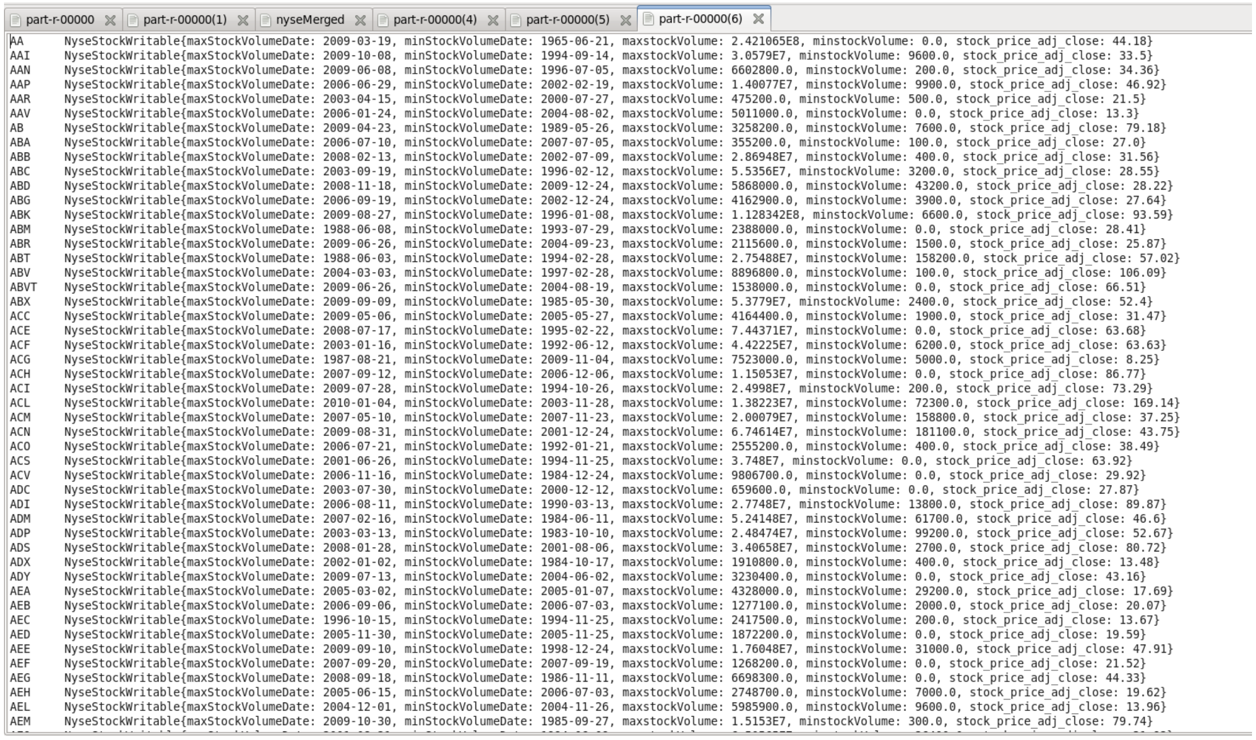


```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/Desktop/hw4/writableCombiner.jar NyseWritableCombiner.NyseStockDriverCombiner /nyseMerged /writableCombinerOutput
22/10/31 10:02:33 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
22/10/31 10:02:33 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
22/10/31 10:02:34 INFO input.FileInputFormat: Total input paths to process : 1
22/10/31 10:02:34 INFO mapreduce.JobSubmitter: number of splits:4
22/10/31 10:02:34 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1666102279738_0009
22/10/31 10:02:34 INFO impl.YarnClientImpl: Submitted application application_1666102279738_0009
22/10/31 10:02:34 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1666102279738_0009/
22/10/31 10:02:34 INFO mapreduce.Job: Running job: job_1666102279738_0009
22/10/31 10:02:40 INFO mapreduce.Job: Job job_1666102279738_0009 running in uber mode : false
22/10/31 10:02:40 INFO mapreduce.Job:  map 0% reduce 0%
22/10/31 10:03:02 INFO mapreduce.Job:  map 9% reduce 0%
22/10/31 10:03:06 INFO mapreduce.Job:  map 19% reduce 0%
22/10/31 10:03:08 INFO mapreduce.Job:  map 33% reduce 0%
22/10/31 10:03:09 INFO mapreduce.Job:  map 44% reduce 0%
22/10/31 10:03:12 INFO mapreduce.Job:  map 49% reduce 0%
22/10/31 10:03:13 INFO mapreduce.Job:  map 60% reduce 0%
22/10/31 10:03:14 INFO mapreduce.Job:  map 66% reduce 0%
22/10/31 10:03:15 INFO mapreduce.Job:  map 90% reduce 0%
22/10/31 10:03:17 INFO mapreduce.Job:  map 100% reduce 0%
22/10/31 10:03:21 INFO mapreduce.Job:  map 100% reduce 100%
22/10/31 10:03:21 INFO mapreduce.Job: Job job_1666102279738_0009 completed successfully
22/10/31 10:03:21 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=297062
                FILE: Number of bytes written=1070680
```



drwxr-xr-x	hdfs	supergroup	0 B	Wed Jul 19 05:36:28 -0700 2017	0	0 B	var
drwxr-xr-x	cloudera	supergroup	0 B	Mon Oct 31 10:03:19 -0700 2022	0	0 B	writableCombinerOutput
drwxr-xr-x	cloudera	supergroup	0 B	Mon Oct 31 09:08:29 -0700 2022	0	0 B	writableOutput

Output:



part-r-00000 | part-r-00000(1) | nyseMerged | part-r-00000(4) | part-r-00000(5) | part-r-00000(6)

```
AA      NyseStockWritable{maxStockVolumeDate: 2009-03-19, minStockVolumeDate: 1965-06-21, maxstockVolume: 2.421065E8, minstockVolume: 0.0, stock_price_adj_close: 44.18}
AAI     NyseStockWritable{maxStockVolumeDate: 2009-10-08, minStockVolumeDate: 1994-09-14, maxstockVolume: 3.0579E7, minstockVolume: 9600.0, stock_price_adj_close: 33.5}
AAN     NyseStockWritable{maxStockVolumeDate: 2009-06-08, minStockVolumeDate: 1996-07-05, maxstockVolume: 6602800.0, minstockVolume: 200.0, stock_price_adj_close: 34.36}
AAP     NyseStockWritable{maxStockVolumeDate: 2006-06-29, minStockVolumeDate: 2002-02-19, maxstockVolume: 1.40077E7, minstockVolume: 9900.0, stock_price_adj_close: 46.92}
AAR     NyseStockWritable{maxStockVolumeDate: 2003-04-15, minStockVolumeDate: 2000-07-27, maxstockVolume: 475200.0, minstockVolume: 500.0, stock_price_adj_close: 21.5}
AAV     NyseStockWritable{maxStockVolumeDate: 2006-01-24, minStockVolumeDate: 2004-08-02, maxstockVolume: 5011000.0, minstockVolume: 0.0, stock_price_adj_close: 13.3}
AB      NyseStockWritable{maxStockVolumeDate: 2009-04-23, minStockVolumeDate: 1989-05-26, maxstockVolume: 3258200.0, minstockVolume: 7600.0, stock_price_adj_close: 79.18}
ABA     NyseStockWritable{maxStockVolumeDate: 2006-07-10, minStockVolumeDate: 2007-07-05, maxstockVolume: 355200.0, minstockVolume: 100.0, stock_price_adj_close: 27.0}
ABB     NyseStockWritable{maxStockVolumeDate: 2008-02-13, minStockVolumeDate: 2002-07-09, maxstockVolume: 2.86948E7, minstockVolume: 400.0, stock_price_adj_close: 31.56}
ABC     NyseStockWritable{maxStockVolumeDate: 2003-09-19, minStockVolumeDate: 1996-02-12, maxstockVolume: 5.5356E7, minstockVolume: 3200.0, stock_price_adj_close: 28.55}
ABD     NyseStockWritable{maxStockVolumeDate: 2008-11-18, minStockVolumeDate: 2009-12-24, maxstockVolume: 5868000.0, minstockVolume: 43200.0, stock_price_adj_close: 28.22}
ABG     NyseStockWritable{maxStockVolumeDate: 2006-09-19, minStockVolumeDate: 2002-12-24, maxstockVolume: 4162900.0, minstockVolume: 3900.0, stock_price_adj_close: 27.64}
ABK     NyseStockWritable{maxStockVolumeDate: 2009-08-27, minStockVolumeDate: 1996-01-08, maxstockVolume: 1.128342E8, minstockVolume: 6600.0, stock_price_adj_close: 93.59}
ABM     NyseStockWritable{maxStockVolumeDate: 1988-06-08, minStockVolumeDate: 1993-07-29, maxstockVolume: 2388000.0, minstockVolume: 0.0, stock_price_adj_close: 28.41}
ABR     NyseStockWritable{maxStockVolumeDate: 2009-06-26, minStockVolumeDate: 2004-09-23, maxstockVolume: 2115600.0, minstockVolume: 1500.0, stock_price_adj_close: 25.87}
ABT     NyseStockWritable{maxStockVolumeDate: 1988-06-03, minStockVolumeDate: 1994-02-28, maxstockVolume: 2.75488E7, minstockVolume: 158200.0, stock_price_adj_close: 57.02}
ABV     NyseStockWritable{maxStockVolumeDate: 2004-03-03, minStockVolumeDate: 1997-02-28, maxstockVolume: 8896800.0, minstockVolume: 100.0, stock_price_adj_close: 106.09}
ABVT    NyseStockWritable{maxStockVolumeDate: 2009-06-26, minStockVolumeDate: 2004-08-19, maxstockVolume: 1538000.0, minstockVolume: 0.0, stock_price_adj_close: 66.51}
ABX     NyseStockWritable{maxStockVolumeDate: 2009-09-09, minStockVolumeDate: 1985-05-30, maxstockVolume: 5.3779E7, minstockVolume: 2400.0, stock_price_adj_close: 52.4}
ACC     NyseStockWritable{maxStockVolumeDate: 2009-05-06, minStockVolumeDate: 2005-05-27, maxstockVolume: 4164400.0, minstockVolume: 1900.0, stock_price_adj_close: 31.47}
ACE     NyseStockWritable{maxStockVolumeDate: 2008-07-17, minStockVolumeDate: 1995-02-22, maxstockVolume: 7.44371E7, minstockVolume: 0.0, stock_price_adj_close: 63.68}
ACF     NyseStockWritable{maxStockVolumeDate: 2003-01-16, minStockVolumeDate: 1992-06-12, maxstockVolume: 4.42225E7, minstockVolume: 6200.0, stock_price_adj_close: 63.63}
ACG     NyseStockWritable{maxStockVolumeDate: 1987-08-21, minStockVolumeDate: 2009-11-04, maxstockVolume: 7523000.0, minstockVolume: 5000.0, stock_price_adj_close: 8.25}
ACH     NyseStockWritable{maxStockVolumeDate: 2007-09-12, minStockVolumeDate: 2006-12-06, maxstockVolume: 1.15053E7, minstockVolume: 0.0, stock_price_adj_close: 86.77}
ACI     NyseStockWritable{maxStockVolumeDate: 2009-07-28, minStockVolumeDate: 1994-10-26, maxstockVolume: 2.4998E7, minstockVolume: 200.0, stock_price_adj_close: 73.29}
ACL     NyseStockWritable{maxStockVolumeDate: 2010-01-04, minStockVolumeDate: 2003-11-28, maxstockVolume: 1.38223E7, minstockVolume: 72300.0, stock_price_adj_close: 169.14}
ACM     NyseStockWritable{maxStockVolumeDate: 2007-05-10, minStockVolumeDate: 2007-11-23, maxstockVolume: 2.00079E7, minstockVolume: 158800.0, stock_price_adj_close: 37.25}
ACN     NyseStockWritable{maxStockVolumeDate: 2009-08-31, minStockVolumeDate: 2001-12-24, maxstockVolume: 6.74614E7, minstockVolume: 181100.0, stock_price_adj_close: 43.75}
ACO     NyseStockWritable{maxStockVolumeDate: 2006-07-21, minStockVolumeDate: 1992-01-21, maxstockVolume: 2555200.0, minstockVolume: 400.0, stock_price_adj_close: 38.49}
ACS     NyseStockWritable{maxStockVolumeDate: 2001-06-26, minStockVolumeDate: 1994-11-25, maxstockVolume: 3.748E7, minstockVolume: 0.0, stock_price_adj_close: 63.92}
ACV     NyseStockWritable{maxStockVolumeDate: 2006-11-16, minStockVolumeDate: 1984-12-24, maxstockVolume: 9806700.0, minstockVolume: 0.0, stock_price_adj_close: 29.92}
ADC     NyseStockWritable{maxStockVolumeDate: 2003-07-30, minStockVolumeDate: 2000-12-12, maxstockVolume: 659600.0, minstockVolume: 0.0, stock_price_adj_close: 27.87}
ADI     NyseStockWritable{maxStockVolumeDate: 2006-08-11, minStockVolumeDate: 1990-03-13, maxstockVolume: 2.7748E7, minstockVolume: 13800.0, stock_price_adj_close: 89.87}
ADM     NyseStockWritable{maxStockVolumeDate: 2007-02-16, minStockVolumeDate: 1984-06-11, maxstockVolume: 5.24148E7, minstockVolume: 61700.0, stock_price_adj_close: 46.6}
ADP     NyseStockWritable{maxStockVolumeDate: 2003-03-13, minStockVolumeDate: 1983-10-10, maxstockVolume: 2.48474E7, minstockVolume: 99200.0, stock_price_adj_close: 52.67}
ADS     NyseStockWritable{maxStockVolumeDate: 2008-01-28, minStockVolumeDate: 2001-08-06, maxstockVolume: 3.40658E7, minstockVolume: 2700.0, stock_price_adj_close: 80.72}
ADX     NyseStockWritable{maxStockVolumeDate: 2002-01-02, minStockVolumeDate: 1984-10-17, maxstockVolume: 1910800.0, minstockVolume: 400.0, stock_price_adj_close: 13.48}
ADY     NyseStockWritable{maxStockVolumeDate: 2009-07-13, minStockVolumeDate: 2004-06-02, maxstockVolume: 3230400.0, minstockVolume: 0.0, stock_price_adj_close: 43.16}
AEA     NyseStockWritable{maxStockVolumeDate: 2005-03-02, minStockVolumeDate: 2005-01-07, maxstockVolume: 4328000.0, minstockVolume: 29200.0, stock_price_adj_close: 17.69}
AEB     NyseStockWritable{maxStockVolumeDate: 2006-09-06, minStockVolumeDate: 2006-07-03, maxstockVolume: 1277100.0, minstockVolume: 2000.0, stock_price_adj_close: 20.07}
AEC     NyseStockWritable{maxStockVolumeDate: 1996-10-15, minStockVolumeDate: 1994-11-25, maxstockVolume: 2417500.0, minstockVolume: 200.0, stock_price_adj_close: 13.67}
AED     NyseStockWritable{maxStockVolumeDate: 2005-11-30, minStockVolumeDate: 2005-11-25, maxstockVolume: 1872200.0, minstockVolume: 0.0, stock_price_adj_close: 19.59}
AEE     NyseStockWritable{maxStockVolumeDate: 2009-09-10, minStockVolumeDate: 1998-12-24, maxstockVolume: 1.76048E7, minstockVolume: 31000.0, stock_price_adj_close: 47.91}
AEF     NyseStockWritable{maxStockVolumeDate: 2007-09-20, minStockVolumeDate: 2007-09-19, maxstockVolume: 1268200.0, minstockVolume: 0.0, stock_price_adj_close: 21.52}
AEG     NyseStockWritable{maxStockVolumeDate: 2008-09-18, minStockVolumeDate: 1986-11-11, maxstockVolume: 6698300.0, minstockVolume: 0.0, stock_price_adj_close: 44.33}
AEH     NyseStockWritable{maxStockVolumeDate: 2005-06-15, minStockVolumeDate: 2006-07-03, maxstockVolume: 2748700.0, minstockVolume: 7000.0, stock_price_adj_close: 19.62}
AEL     NyseStockWritable{maxStockVolumeDate: 2004-12-01, minStockVolumeDate: 2004-11-26, maxstockVolume: 5985900.0, minstockVolume: 9600.0, stock_price_adj_close: 13.96}
AEM     NyseStockWritable{maxStockVolumeDate: 2009-10-30, minStockVolumeDate: 1985-09-27, maxstockVolume: 1.5153E7, minstockVolume: 300.0, stock_price_adj_close: 79.74}
```

The run time for running the MapReduce without a combiner is 1 minute 19 seconds and that of with combiner is 50 seconds.