Big Data Analysis using Hadoop

for Proof of Concept

Version	1.0
Last Modified Date	February, 24th 2014
Status	*In Progress*
Author	Venkat Voruganti
File Name	Big Data Analysis using Hadoop - Venkat Voruganti.docx

Change Log:

Version	Date	Author	Comments
0.1	Nov 30th 2013	Venkat Voruganti	Initial Creation
1.0	Feb 24th 2014	Venkat Voruganti	Final Version

Table of Contents

1 Objective

Objective of this project is to demonstrate the ability and usability of the Hadoop framework for analyzing large volume of data (Big Data). This project was done as part of the Hadoop Training.

2 Work Statement

1. Lending Club is an online financial community that brings together creditworthy borrowers and savvy investors to arrange loans. Since 2007, Lending Club has funded $3 Billion in loans. It has published loan database on their website (https://www.lendingclub.com). (Map Reduce)

   Based on the information available in 'Declined Loan Data' file, calculate following metrics

   1. Summarize loans by State, Credit Rating and Loan Title
   2. Identify top 10 cities with maximum number of loans
   3. Calculate total loan amount for each loan title in the state of New Jersey
   4. Number of loans and loan amount in each month.

2. Find out the top 10 stocks that gained most value in New York Stock Exchange between Aug 2009 & August 2010. (Pig)

3. Identify the schools with most SAT takers in New York city (Hive)

4. Transfer top 10 stocks, created in step 2, from Hadoop file system into a RDBMS table. (SQOOP)

3 Solution Requirements

The solution to gather required metrics will be developed with the use of Hadoop API on a system with Linux/Unix operating system. The solution will demonstrate key features of Hadoop API, such as,

1. Map Reduce for filtering and categorizing the data
2. Pig for filtering and creating subsets of data
3. Hive for sorting and finding records with top values.
4. Sqoop for exporting summarized data from HDFS into RDMS System
5. Optionally; performance of cloud based cluster with 2 data nodes will compared with the performance of a single machine.

4 Architecture:

Operating System	Standalone ubuntu-server-12.04 running in VMWare on Windows 7, 64bit
Number of Processors	2
Physical Memory	6GB.
RAM Allocated for Ubuntu	3.5GB
IDE	Eclipse, Kepler
Java	1.6
Hadoop Release	Hadoop-1.0.3
Other Tools	Putty, WinSCP

5 Map Reduce

5.1 Objective:

1. Summarize loans by State, Credit Rating and Loan Title
2. Identify top 10 cities with maximum number of loans
3. Calculate total loan amount for each loan title in the state of New Jersey
4. Number of loans and loan amount in each month.

5.2 Additional Requirements

1. Omit the records with zero credit score as they are bad records.
2. Derive credit ratings as per below table.

FICO Score Range	Credit Rating
0-619	B- Bad
620-659	F – Fair
660-719	G- Good
720+	E- Excellent

5.3 Dataset:

Source of Data https://www.lendingclub.com/info/download-data.action

Number of records: 666,003, Dataset size: 44MB

Sample Data Records

Amount Requested	Application Date	Loan Title	FICO-Score	Debt-To-Income Ratio	City	State	Employment Length	Policy Code
30000	1/1/2013	debt_consolidation	754	32.01%	Lyle	MN	< 1 year	0
15000	1/1/2013	medical	728	27.01%	hayward	CA	< 1 year	0
2500	1/1/2013	other	723	0.92%	RIDLEY PARK	PA	7 years	0
4000	1/1/2013	car	563	24.92%	Greenwood	SC	< 1 year	0
22000	1/1/2013	debt_consolidation	0	26.91%	grandfield	OK	< 1 year	0
6000	1/1/2013	debt_consolidation	640	11.82%	Lubbock	TX	< 1 year	0
10000	1/1/2013	debt_consolidation	606	22.15%	greenville	SC	< 1 year	0
35000	1/1/2013	debt_consolidation	670	66.31%	Mattawan	MI	< 1 year	0
10000	1/1/2013	debt_consolidation	558	37.10%	san antonio	TX	< 1 year	0

Data Structure

Reject Stats File	Description
Amount Requested	The total amount requested by the borrower
Application Date	The date which the borrower applied
Loan Title	The loan title provided by the borrower
FICO_Score	The borrower's Vantage score
Debt-To-Income Ratio	The borrower's debt to income ratio, calculated using the monthly payments on the total debt obligations, excluding mortgage, divided by self-reported monthly income.
City	The address city provided by the borrower during loan application.
State	The address state provided by the borrower during loan application
Employment Length	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
Policy Code	publicly available policy_code=1

5.4 Approach:

New API of MapReduce will be used to summarize (reduce) the data.

Mapper:

1. Omit the records that have zero credit scores
2. Calculate credit rating as mentioned in section 2.1
3. Emit State + Credit Rating + Loan Title as key, loan amount as Values

Reduce:

1. Add loan amounts
2. Count records

5.5 MapReduce program code

Mapper Class : LoanSummaryMapper.java

```
package com.venkatvorugantipoc;

import java.io.IOException;

import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class LoanSummaryMapper extends
   Mapper<LongWritable, Text, Text, FloatWritable> {
  private Text outKey = new Text();
  private FloatWritable outValue = new FloatWritable();

  public void map(LongWritable key, Text value, Context context)
     throws IOException, InterruptedException {
    String line = value.toString();

    String[] lineFields = line.split(",(?=([^\"]*\"[^\"]*\")*[^\"]*$)");
    for (int i = 0; i < lineFields.length; i++)
      lineFields[i] = lineFields[i].replaceAll("^\"|\"$", "");

    if (lineFields.length > 1)
```

```
    // ignore header row
    if (!lineFields[0].equalsIgnoreCase("Amount Requested")) {
      float amountRequested = (new Float(lineFields[0])).floatValue();
      String loanTitle = lineFields[2];
      int FICO_Score = 0;
      if (!"".equals(lineFields[3]))
        try {
          FICO_Score = (new Integer(lineFields[3])).intValue();
        } catch (NumberFormatException e) {
          System.out.println("Bad FICO Score. Mapper Input. Key= "
              + key + " Input Line# = " + value);
        }
      String state = lineFields[6];
      if (FICO_Score > 0) // Omit loans with zero FICO score
        if (state.equalsIgnoreCase("NJ")) // Process NJ only
        {
          String creditRating;
          if (0 <= FICO_Score && FICO_Score <= 610)
            creditRating = "B-Bad";
          else if (620 <= FICO_Score && FICO_Score <= 659)
            creditRating = "F-Fair";
          else if (660 <= FICO_Score && FICO_Score <= 719)
            creditRating = "G-Good";
          else
            // if (720 <= FICO_Score)
            creditRating = "E-Excellent";

          outKey = new Text(state + "\t" + creditRating + "\t"
              + loanTitle);
          outValue = new FloatWritable(amountRequested);
          context.write(outKey, outValue);
        }
    }
  }
} // end of LoanSummaryMapper class
```

Reduce Class: LoanSummaryReducer.java

```
package com.venkatvorugantipoc;

import java.io.IOException;

import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class LoanSummaryReducer extends
   Reducer<Text, FloatWritable, Text, FloatWritable> {

  public void reduce(Text key, Iterable<FloatWritable> values, Context context)
     throws IOException, InterruptedException {

   float sum = 0;
   float count = 0;
   for (FloatWritable val : values) {
     sum += val.get();
     count++;
   }
   context.write(new Text(key.toString() + "\t" + count), new FloatWritable(sum));
  }
} // end of class LoanSummaryReducer
```

Main Class: LoanSummary.java

```
package com.venkatvorugantipoc;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.FloatWritable;
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.util.GenericOptionsParser;

public class LoanSummary {

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

    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();

    Job job = new Job(conf, "Loan Summary");
    job.setJarByClass(LoanSummary.class);
    job.setMapperClass(LoanSummaryMapper.class);
    job.setReducerClass(LoanSummaryReducer.class);

    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(FloatWritable.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(FloatWritable.class);

    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

    System.exit(job.waitForCompletion(true) ? 0 : 1);
  } // end of main

}
```

Test Class: LoanSummaryUnitTestCase.java

```
package com.venkatvorugantipoc;

import java.util.ArrayList;
import java.util.List;

import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mrunit.mapreduce.MapDriver;
import org.apache.hadoop.mrunit.mapreduce.MapReduceDriver;
import org.apache.hadoop.mrunit.mapreduce.ReduceDriver;
import org.junit.Before;
import org.junit.Test;

public class LoanSummaryUnitTestCase {
  MapDriver<LongWritable, Text, Text, FloatWritable> mapDriver;
  ReduceDriver<Text, FloatWritable, Text, FloatWritable> reduceDriver;

  MapReduceDriver<LongWritable, Text, Text, FloatWritable, Text, FloatWritable> mapReduceDriver;

  @Before
  public void setUp() {
    LoanSummaryMapper mapper = new LoanSummaryMapper();
    LoanSummaryReducer reducer = new LoanSummaryReducer();

    mapDriver = new MapDriver<LongWritable, Text, Text, FloatWritable>();
```

```
    mapDriver.setMapper(mapper);

    reduceDriver = new ReduceDriver<Text, FloatWritable, Text, FloatWritable>();
    reduceDriver.setReducer(reducer);

    mapReduceDriver = new MapReduceDriver<LongWritable, Text, Text, FloatWritable, Text,
FloatWritable>();
    mapReduceDriver.setMapper(mapper);
    mapReduceDriver.setReducer(reducer);
  }

  public void testMapper() {
    mapDriver
        .withInput(
           new LongWritable(1),
           new Text(
              "30000.00,2013-01-01,debt_consolidation,754,32.01%,Lyle,MN,< 1 year,0"));
    mapDriver.withOutput(new Text("MN, E-Excellent, debt_consolidation"),
        new FloatWritable((float) 30000.00));
    mapDriver.runTest();
  }

  @Test
  public void testMapper2() {
    mapDriver
        .withInput(
           new LongWritable(1),
           new Text(
              "\"30000.00\",\"2013-01-02\",\"Paying 4-5 credit cards, closing them
\",\"694\",\"21.25%\",\"Round Rock\",\"TX\",\"4 years\",\"0\""));
    mapDriver.withOutput(new Text(
        "TX, G-Good, Paying 4-5 credit cards, closing them"),
        new FloatWritable((float) 30000.00));
    mapDriver.runTest();
  }

  public void testReducer() {
    List<FloatWritable> values = new ArrayList<FloatWritable>();
    values.add(new FloatWritable((float) 10.0));
    values.add(new FloatWritable((float) 15.0));
    reduceDriver.withInput(new Text("SANDBOXES"), values);
    reduceDriver.withOutput(new Text("SANDBOXES"), new FloatWritable(25));
    reduceDriver.runTest();
  }

  public void testMapReduceSingleProduct() {
    mapReduceDriver
        .withInput(
           new LongWritable(1),
           new Text(
              "30000.00,2013-01-01,debt_consolidation,754,32.01%,Lyle,MN,< 1 year,0"));
    mapReduceDriver
        .withInput(
           new LongWritable(1),
           new Text(
              "30000.00,2013-01-01,debt_consolidation,754,32.01%,Lyle,MN,< 1 year,0"));

    mapReduceDriver.addOutput(new Text(
        "MN, E-Excellent, debt_consolidation"), new FloatWritable(
        (float) 60000.00));
    mapReduceDriver.runTest();
  }

  public void testMapReduceMultipleProduct() {
    mapReduceDriver
        .withInput(
           new LongWritable(1),
           new Text(
              "30000.00,2013-01-01,debt_consolidation,754,32.01%,Lyle,MN,< 1 year,0"));
```

```
    mapReduceDriver
        .withInput(
            new LongWritable(1),
            new Text(
                "15000.00,2013-01-01,medical,728,27.01%,hayward,CA,< 1 year,0"));
    mapReduceDriver
        .withInput(
            new LongWritable(1),
            new Text(
                "Amount Requested,Application Date,Loan Title,FICO-Score,Debt-To-Income
Ratio,City,State,Employment Length,Policy Code"));

    mapReduceDriver.addOutput(new Text("CA, E-Excellent, medical"),
        new FloatWritable((float) 15000.00));
    mapReduceDriver.addOutput(new Text(
        "MN, E-Excellent, debt_consolidation"), new FloatWritable(
        (float) 30000.00));
    mapReduceDriver.runTest();
  }

}
```

5.6 Execution

1. Copy the data file to HDFS. Input data file location in HDFS: /input/LSinputFull.dat
2. Execute the MapReduce

```
$ hadoop jar LoanSummary.jar com.venkatvorugantipoc.LoanSummary /input/LSinputFull.dat
/output/LoanSummaryFull
```

```
14/02/23 03:02:58 INFO input.FileInputFormat: Total input paths to process : 1
14/02/23 03:02:58 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/02/23 03:02:58 WARN snappy.LoadSnappy: Snappy native library not loaded
14/02/23 03:02:58 INFO mapred.JobClient: Running job: job_201402230221_0002
14/02/23 03:02:59 INFO mapred.JobClient:  map 0% reduce 0%
14/02/23 03:03:21 INFO mapred.JobClient:  map 19% reduce 0%
14/02/23 03:03:24 INFO mapred.JobClient:  map 33% reduce 0%
14/02/23 03:03:27 INFO mapred.JobClient:  map 48% reduce 0%
14/02/23 03:03:31 INFO mapred.JobClient:  map 62% reduce 0%
14/02/23 03:03:34 INFO mapred.JobClient:  map 76% reduce 0%
14/02/23 03:03:37 INFO mapred.JobClient:  map 90% reduce 0%
14/02/23 03:03:40 INFO mapred.JobClient:  map 100% reduce 0%
14/02/23 03:03:55 INFO mapred.JobClient:  map 100% reduce 100%
14/02/23 03:04:03 INFO mapred.JobClient: Job complete: job_201402230221_0002
14/02/23 03:04:03 INFO mapred.JobClient: Counters: 29
14/02/23 03:04:03 INFO mapred.JobClient:   Job Counters
14/02/23 03:04:03 INFO mapred.JobClient:     Launched reduce tasks=1
14/02/23 03:04:03 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=41764
14/02/23 03:04:03 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving
slots (ms)=0
14/02/23 03:04:03 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots
(ms)=0
14/02/23 03:04:03 INFO mapred.JobClient:     Launched map tasks=1
14/02/23 03:04:03 INFO mapred.JobClient:     Data-local map tasks=1
14/02/23 03:04:03 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=12837
14/02/23 03:04:03 INFO mapred.JobClient:   File Output Format Counters
14/02/23 03:04:03 INFO mapred.JobClient:     Bytes Written=35900
14/02/23 03:04:03 INFO mapred.JobClient:   FileSystemCounters
14/02/23 03:04:03 INFO mapred.JobClient:     FILE_BYTES_READ=631725
14/02/23 03:04:03 INFO mapred.JobClient:     HDFS_BYTES_READ=44957942
14/02/23 03:04:03 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=1307025
14/02/23 03:04:03 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=35900
14/02/23 03:04:03 INFO mapred.JobClient:   File Input Format Counters
14/02/23 03:04:03 INFO mapred.JobClient:     Bytes Read=44957834
14/02/23 03:04:03 INFO mapred.JobClient:   Map-Reduce Framework
```

```
14/02/23 03:04:03 INFO mapred.JobClient:     Map output materialized bytes=631725
14/02/23 03:04:03 INFO mapred.JobClient:     Map input records=666003
14/02/23 03:04:03 INFO mapred.JobClient:     Reduce shuffle bytes=0
14/02/23 03:04:03 INFO mapred.JobClient:     Spilled Records=41020
14/02/23 03:04:03 INFO mapred.JobClient:     Map output bytes=590699
14/02/23 03:04:03 INFO mapred.JobClient:     CPU time spent (ms)=27800
14/02/23 03:04:03 INFO mapred.JobClient:     Total committed heap usage (bytes)=273022976
14/02/23 03:04:03 INFO mapred.JobClient:     Combine input records=0
14/02/23 03:04:03 INFO mapred.JobClient:     SPLIT_RAW_BYTES=108
14/02/23 03:04:03 INFO mapred.JobClient:     Reduce input records=20510
14/02/23 03:04:03 INFO mapred.JobClient:     Reduce input groups=926
14/02/23 03:04:03 INFO mapred.JobClient:     Combine output records=0
14/02/23 03:04:03 INFO mapred.JobClient:     Physical memory (bytes) snapshot=357683200
14/02/23 03:04:03 INFO mapred.JobClient:     Reduce output records=926
14/02/23 03:04:03 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=3493900288
14/02/23 03:04:03 INFO mapred.JobClient:     Map output records=20510
```

5.7 Output

Output of the MapReduce gets stored in folder, **`hdfs://output/LoanSummaryFull/part-r-00000`**

Sample output file contents (actual file is a delimited text file. For readability it is shown here in tabular form and column heading were added manually)

State	Credit Rating	Loan Title	No Loans	Total Loan Amount
NJ	B-Bad	Home Improvement	1	$ 2,000.00
NJ	B-Bad	Other	1	$ 5,000.00
NJ	B-Bad	car	217	$ 1,944,750.00
NJ	B-Bad	credit card	1	$ 15,000.00
NJ	B-Bad	credit_card	319	$ 3,776,400.00
NJ	B-Bad	debt_consolidation	1877	$ 21,281,624.00
NJ	B-Bad	home_improvement	271	$ 3,179,025.00
NJ	B-Bad	house	56	$ 975,000.00
NJ	B-Bad	major_purchase	225	$ 1,846,275.00
NJ	B-Bad	medical	197	$ 1,172,300.00
NJ	B-Bad	moving	224	$ 1,223,200.00
NJ	B-Bad	other	1191	$ 7,998,125.00
NJ	B-Bad	renewable_energy	13	$ 114,500.00
NJ	B-Bad	small_business	203	$ 3,715,775.00
NJ	B-Bad	vacation	76	$ 430,800.00
NJ	B-Bad	wedding	82	$ 699,825.00
NJ	E-Excellent	PAYOFF A HIGHER APR LOAN	1	$ 15,000.00

6 Pig

6.1 Objective:

Find out the top 10 stocks that gained most value in New York Stock Exchange between August 2009 and August 2010. (Pig)

6.2 Dataset:

Data Source: http://kumo.swcp.com/stocks/

Data Format: csv

Number of Records: 122,574. File size: 5.2MB

Sample Data:

Date	StockSymbol	dayOpen	DayHigh	DayLow	DayClose	Volume
20090821	A	25.6	25.61	25.22	25.55	34758
20090824	A	25.64	25.74	25.33	25.5	22247
20090825	A	25.5	25.7	25.225	25.34	30891
20090826	A	25.32	25.6425	25.145	25.48	33334
20090827	A	25.5	25.57	25.23	25.54	70176
20090828	A	25.67	26.05	25.63	25.83	39694
20090831	A	25.45	25.74	25.31	25.68	51064
20090901	A	25.51	26.33	25.48	25.85	66422
20090902	A	25.97	25.97	24.96	25.22	64614

6.3 Approach:

1. MapReduce method of Pig is required as the dataset is large.
2. Import the Daily Stocks file into HDFS
3. LOAD the file into PIG
4. Group records by stock symbol, calculate max & minimum values across the dataset for each stock symbol. Also calculate the difference between max and minimum values.
5. Sort the output on price difference in descending order. Limit the output to 10 records.
6. Write the sort output to a file.

6.4 PIG program code

Pig: commands

```
DailyPrices = LOAD '/input/stocks.sp500hst.txt' using PigStorage(',') AS
  (dpDate:chararray,
  dpStock:chararray,
  dpOpen:double,
  dpHigh:double,
  dpLow:double,
  dpClose:double,
  dpVolume:long);

SockSymbols = LOAD '/input/StockSymbols.dat'  using PigStorage(';') AS
  (ssSymbol:chararray,
  ssName:chararray);

DailyPrices_grouped = GROUP DailyPrices BY dpStock;
DailyPrices_highlows = FOREACH DailyPrices_grouped GENERATE group,
  MAX(DailyPrices.dpHigh), MIN(DailyPrices.dpLow),
  MAX(DailyPrices.dpHigh) - MIN(DailyPrices.dpLow);
DailyPrices_highlows_sorted = ORDER DailyPrices_highlows BY $3 DESC;
DailyPrices_highlows_top10 = LIMIT DailyPrices_highlows_sorted 10;
```

```
STORE DailyPrices_highlows_top10 INTO '/output/pigStocksTopTen';

STORE DailyPrices_highlows INTO '/output/pigStocks';
```

6.5 Execution

Pig: Execution

```
notroot@ubuntu:~$  hadoop fs -copyFromLocal lab/data/StockSymbols.dat /input/StockSymbols.dat
notroot@ubuntu:~$ pig
2014-02-24 02:10:45,807 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/notroot/pig_1393207845783.log
2014-02-24 02:10:46,335 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://localhost/
2014-02-24 02:10:47,529 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to map-reduce job tracker at: localhost:8021
grunt> DailyPrices = LOAD '/input/stocks.sp500hst.txt' using PigStorage(',') AS
>> (dpDate:chararray,
>> dpStock:chararray,
>> dpOpen:double,
>> dpHigh:double,
>> dpLow:double,
>> dpClose:double,
>> dpVolume:long);
grunt>
grunt> SockSymbols = LOAD '/input/StockSymbols.dat'  using PigStorage(';') AS
>> (ssSymbol:chararray,
>> ssName:chararray);
grunt> DailyPrices_grouped = GROUP DailyPrices BY dpStock;
grunt> DailyPrices_highlows = FOREACH DailyPrices_grouped GENERATE group, MAX(DailyPrices.dpHigh), MIN(DailyPrices.dpLow), MAX(DailyPrices.dpHigh) - MIN(DailyPrices.dpLow);
grunt> DailyPrices_highlows_sorted = ORDER DailyPrices_highlows BY $3 DESC;
grunt> DailyPrices_highlows_top10 = LIMIT DailyPrices_highlows_sorted 10;
grunt>
grunt> STORE DailyPrices_highlows_top10 INTO '/output/pigStocksTopTen';
2014-02-24 02:13:11,260 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: GROUP_BY,ORDER_BY,LIMIT
2014-02-24 02:13:11,760 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2014-02-24 02:13:12,013 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.CombinerOptimizer - Choosing to move algebraic foreach to combiner
2014-02-24 02:13:12,114 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 3
2014-02-24 02:13:12,114 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 3
2014-02-24 02:13:12,353 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig script settings are added to the job
2014-02-24 02:13:12,413 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
2014-02-24 02:13:12,429 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - creating jar file Job8331703627879804263.jar
2014-02-24 02:13:15,489 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - jar file Job8331703627879804263.jar created
2014-02-24 02:13:15,540 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting up single store job
2014-02-24 02:13:15,644 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - BytesPerReducer=1000000000 maxReducers=999 totalInputFileSize=5326536
2014-02-24 02:13:15,644 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Neither PARALLEL nor default parallelism is set for this job. Setting number of reducers to 1
2014-02-24 02:13:15,807 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 1 map-reduce job(s) waiting for submission.
2014-02-24 02:13:16,314 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 0% complete
2014-02-24 02:13:16,542 [Thread-8] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2014-02-24 02:13:16,543 [Thread-8] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
```

```
2014-02-24 02:13:16,618 [Thread-8] INFO  org.apache.hadoop.util.NativeCodeLoader - Loaded the native-hadoop library
2014-02-24 02:13:16,619 [Thread-8] WARN  org.apache.hadoop.io.compress.snappy.LoadSnappy - Snappy native library not loaded
2014-02-24 02:13:16,628 [Thread-8] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths
(combined) to process : 1
2014-02-24 02:13:18,075 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher -
HadoopJobId: job_201402240210_0001
2014-02-24 02:13:18,075 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - More
information at: http://localhost:50030/jobdetails.jsp?jobid=job_201402240210_0001
2014-02-24 02:13:45,737 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 12%
complete
2014-02-24 02:13:48,869 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 16%
complete
2014-02-24 02:14:04,195 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 33%
complete
2014-02-24 02:14:18,126 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig script settings are added to the job
2014-02-24 02:14:18,129 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler -
mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
2014-02-24 02:14:18,131 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - creating
jar file Job8612457952553597871.jar
2014-02-24 02:14:21,089 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - jar file
Job8612457952553597871.jar created
2014-02-24 02:14:21,096 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting
up single store job
2014-02-24 02:14:21,179 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 1 map-
reduce job(s) waiting for submission.
2014-02-24 02:14:21,486 [Thread-18] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2014-02-24 02:14:21,490 [Thread-18] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to
process : 1
2014-02-24 02:14:21,496 [Thread-18] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths
(combined) to process : 1
2014-02-24 02:14:22,329 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher -
HadoopJobId: job_201402240210_0002
2014-02-24 02:14:22,329 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - More
information at: http://localhost:50030/jobdetails.jsp?jobid=job_201402240210_0002
2014-02-24 02:14:40,753 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 50%
complete
2014-02-24 02:14:47,415 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 50%
complete
2014-02-24 02:14:56,075 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 66%
complete
2014-02-24 02:15:07,277 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig script settings are added to the job
2014-02-24 02:15:07,280 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler -
mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
2014-02-24 02:15:07,308 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - creating
jar file Job5705748916303126622.jar
2014-02-24 02:15:10,245 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - jar file
Job5705748916303126622.jar created
2014-02-24 02:15:10,251 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting
up single store job
2014-02-24 02:15:10,389 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 1 map-
reduce job(s) waiting for submission.
2014-02-24 02:15:10,708 [Thread-29] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2014-02-24 02:15:10,711 [Thread-29] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to
process : 1
2014-02-24 02:15:10,740 [Thread-29] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths
(combined) to process : 1
2014-02-24 02:15:11,670 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher -
HadoopJobId: job_201402240210_0003
2014-02-24 02:15:11,671 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - More
information at: http://localhost:50030/jobdetails.jsp?jobid=job_201402240210_0003
2014-02-24 02:15:35,054 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 83%
complete
2014-02-24 02:16:01,871 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 100%
complete
2014-02-24 02:16:01,915 [main] INFO  org.apache.pig.tools.pigstats.SimplePigStats - Script Statistics:
```

```
HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
1.0.3   0.9.2   notroot 2014-02-24 02:13:12     2014-02-24 02:16:01     GROUP_BY,ORDER_BY,LIMIT

Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTIme      AvgMapTime      MaxReduceTime   MinReduceTime   AvgReduceTime   Alias
Feature Outputs
job_201402240210_0001   1       1       15      15      15      15      15      15      DailyPrices,DailyPrices_grouped,DailyPrices_highlows
GROUP_BY,COMBINER
job_201402240210_0002   1       1       9       9       9       15      15      15      DailyPrices_highlows_sorted     SAMPLER
job_201402240210_0003   1       1       9       9       9       15      15      15      DailyPrices_highlows_sorted     ORDER_BY,COMBINER
/output/pigStocksTopTen,

Input(s):
Successfully read 122574 records (5326897 bytes) from: "/input/stocks.sp500hst.txt"

Output(s):
Successfully stored 10 records (307 bytes) in: "/output/pigStocksTopTen"

Counters:
Total records written : 10
Total bytes written : 307
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_201402240210_0001   ->      job_201402240210_0002,
job_201402240210_0002   ->      job_201402240210_0003,
job_201402240210_0003


2014-02-24 02:16:02,030 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher -
Success!
grunt>
```

6.6 Output

File: /output/pigStocksTopTen/part-r-00000

output file contents (actual file is a tab delimited text file. For readability it is shown here in tabular form and column heading were added manually)

Symbol	MaxPrice	MinPrice	PriceChange
WPO	547.58	295.56	252.02
GOOG	629.51	433.63	195.88
ISRG	393.92	211	182.92
PCLN	308.95	166.67	142.28
CME	353.029	236.58	116.449
AAPL	279.01	164.11	114.9
MIL	106.99	26.73	80.26
AZO	213.65	135.68	77.97
MA	269.88	193	76.88
AMZN	151.09	77.51	73.58

7 Hive

Objective:

7.1 Objective:

Identify the schools with most SAT takers in New York city (Hive)

7.2 Dataset:

Source: NYC Open data. URL: https://data.cityofnewyork.us/Education/SAT-Results/f9bf-2cp4

Data format: csv, Number of records: 470, File size 28kb

7.3 Approach:

1. Copy the file to Ubuntu file system using WinScp
2. In HIVE, create a table to match the dataset
3. Load the file data into the table
4. Run a HIVE Query to select the required data
5. Write the output to a file

7.4 Hive Commands

Hive: commands

```
create database SAT;
use SAT;
create table SATScores (schoolCode STRING, schoolName STRING, testTakers INT, ReadingAvg INT, MathAvg
INT, WritingAvg INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
load data local inpath '/home/notroot/lab/data/SAT_Results.csv' OVERWRITE INTO TABLE SATScores;
SELECT schoolCode, schoolName, testTakers, ReadingAvg, MathAvg, WritingAvg from SATScores ORDER BY
testTakers DESC LIMIT 15;
```

7.5 Execution

Hive: Execution

```
notroot@ubuntu:~$ cd hive
notroot@ubuntu:~/hive$ hive
WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated. Please use
org.apache.hadoop.log.metrics.EventCounter in all the log4j.properties files.
Logging initialized using configuration in jar:file:/home/notroot/lab/software/hive-0.9.0-bin/lib/hive-
common-0.9.0.jar!/hive-log4j.properties
Hive history file=/tmp/notroot/hive_job_log_notroot_201402240503_2114241397.txt
hive> create database SAT;
OK
Time taken: 8.304 seconds
hive> use SAT;
OK
Time taken: 0.027 seconds
hive> create table SATScores (schoolCode STRING, schoolName STRING, testTakers INT, ReadingAvg INT,
MathAvg INT, WritingAvg INT)
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
OK
Time taken: 1.34 seconds
hive> load data local inpath '/home/notroot/lab/data/SAT_Results.csv' OVERWRITE INTO TABLE SATScores;
Copying data from file:/home/notroot/lab/data/SAT_Results.csv
Copying file: file:/home/notroot/lab/data/SAT_Results.csv
Loading data to table sat.satscores
Deleted hdfs://localhost:8020/user/hive/warehouse/sat.db/satscores
OK
Time taken: 1.112 seconds
hive> SELECT schoolCode, schoolName, testTakers, ReadingAvg, MathAvg, WritingAvg from SATScores ORDER
BY testTakers DESC LIMIT 15;
Total MapReduce jobs = 1
```

```
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201402240210_0015, Tracking URL =
http://localhost:50030/jobdetails.jsp?jobid=job_201402240210_0015
Kill Command = /home/notroot/lab/software/hadoop-1.0.3/libexec/../bin/hadoop job  -
Dmapred.job.tracker=localhost:8021 -kill job_201402240210_0015
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2014-02-24 05:04:29,545 Stage-1 map = 0%,  reduce = 0%
2014-02-24 05:04:38,839 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.03 sec
2014-02-24 05:04:39,920 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.03 sec
2014-02-24 05:04:40,983 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.03 sec
2014-02-24 05:04:42,009 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.03 sec
2014-02-24 05:04:43,028 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.03 sec
2014-02-24 05:04:44,048 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.03 sec
2014-02-24 05:04:45,077 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.03 sec
2014-02-24 05:04:46,105 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.03 sec
2014-02-24 05:04:47,125 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.03 sec
2014-02-24 05:04:48,144 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.03 sec
2014-02-24 05:04:49,156 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.03 sec
2014-02-24 05:04:50,179 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.03 sec
2014-02-24 05:04:51,199 Stage-1 map = 100%,  reduce = 33%, Cumulative CPU 2.03 sec
2014-02-24 05:04:52,216 Stage-1 map = 100%,  reduce = 33%, Cumulative CPU 2.03 sec
2014-02-24 05:04:53,229 Stage-1 map = 100%,  reduce = 33%, Cumulative CPU 2.03 sec
2014-02-24 05:04:54,261 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.61 sec
2014-02-24 05:04:55,283 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.61 sec
2014-02-24 05:04:56,319 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.61 sec
2014-02-24 05:04:57,341 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.61 sec
2014-02-24 05:04:58,365 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.61 sec
2014-02-24 05:04:59,383 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.61 sec
2014-02-24 05:05:00,403 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.61 sec
2014-02-24 05:05:01,435 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.61 sec
2014-02-24 05:05:02,510 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.61 sec
MapReduce Total cumulative CPU time: 4 seconds 610 msec
Ended Job = job_201402240210_0015
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 4.61 sec   HDFS Read: 28567 HDFS Write: 777 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 610 msec
OK
13K430  BROOKLYN TECHNICAL HIGH SCHOOL  1277    587     659     587
26Q430  FRANCIS LEWIS HIGH SCHOOL       934     468     539     467
26Q415  BENJAMIN N. CARDOZO HIGH SCHOOL 888     480     545     489
02M475  STUYVESANT HIGH SCHOOL  832     679     735     682
22K405  MIDWOOD HIGH SCHOOL     824     478     519     476
31R455  TOTTENVILLE HIGH SCHOOL 807     462     486     470
28Q440  FOREST HILLS HIGH SCHOOL        762     456     497     454
10X445  BRONX HIGH SCHOOL OF SCIENCE    731     632     688     649
21K525  EDWARD R. MURROW HIGH SCHOOL    727     468     496     467
26Q495  BAYSIDE HIGH SCHOOL     708     462     523     464
20K490  FORT HAMILTON HIGH SCHOOL       694     417     478     411
25Q425  JOHN BOWNE HIGH SCHOOL  558     397     451     395
31R460  SUSAN E. WAGNER HIGH SCHOOL     535     455     474     459
03M485  FIORELLO H. LAGUARDIA HIGH SCHOOL OF MUSIC & ART AND PERFORMING ARTS    531     566     564
577
22K425  JAMES MADISON HIGH SCHOOL       518     436     475     439
Time taken: 46.777 seconds
hive>
```

7.6 Output

Top 15 schools with most test takers.

Hive: Output

```
13K430  BROOKLYN TECHNICAL HIGH SCHOOL  1277    587     659     587
26Q430  FRANCIS LEWIS HIGH SCHOOL       934     468     539     467
26Q415  BENJAMIN N. CARDOZO HIGH SCHOOL 888     480     545     489
02M475  STUYVESANT HIGH SCHOOL  832     679     735     682
22K405  MIDWOOD HIGH SCHOOL     824     478     519     476
31R455  TOTTENVILLE HIGH SCHOOL 807     462     486     470
28Q440  FOREST HILLS HIGH SCHOOL        762     456     497     454
10X445  BRONX HIGH SCHOOL OF SCIENCE    731     632     688     649
21K525  EDWARD R. MURROW HIGH SCHOOL    727     468     496     467
26Q495  BAYSIDE HIGH SCHOOL     708     462     523     464
20K490  FORT HAMILTON HIGH SCHOOL       694     417     478     411
25Q425  JOHN BOWNE HIGH SCHOOL  558     397     451     395
31R460  SUSAN E. WAGNER HIGH SCHOOL     535     455     474     459
03M485  FIORELLO H. LAGUARDIA HIGH SCHOOL OF MUSIC & ART AND PERFORMING ARTS    531     566     564
577
22K425  JAMES MADISON HIGH SCHOOL       518     436     475     439
```

8 Sqoop

8.1 Objective:

Load the stocks that gained most value in 2009-2010 period into a RDBMS database (MySQL).

8.2 Additional Requirements

Use the output created by the Pig commands in section 6.

8.3 Dataset:

Dataset : /output/pigStocksTopTen/part-r-00000 created in section 6.

8.4 Approach:

1. Create a RDBMS table called TopStocks
2. Export the HDFS file into the table TopStocks using SQOOP

8.5 SQOOP code

SQOOP: Code

```
//Create a table in MYSQL

CREATE database stocks;

USE stocks;

CREATE TABLE TopStocks (
    Symbol VARCHAR(20) NOT NULL,
    MaxPrice FLOAT NOT NULL,
    MinPrice FLOAT NOT NULL,
    PriceChange FLOAT NOT NULL);

//Export data into above table using stokcs
sqoop export --connect jdbc:mysql://localhost/stocks --table TopStocks --username root --password
hadoop123 --export-dir /output/pigStocksTopTen/part-r-00000 --input-fields-terminated-by '\t'
```

8.6 Execution

SQOOP: Execution

```
notroot@ubuntu:~$ sqoop export --connect jdbc:mysql://localhost/stocks --table TopStocks --username
root --password hadoop123 --export-dir /output/pigStocksTopTen/part-r-00000 --input-fields-terminated-
by '\t'
Warning: /usr/lib/hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
14/02/25 01:07:40 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure.
Consider using -P instead.
14/02/25 01:07:45 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
14/02/25 01:07:45 INFO tool.CodeGenTool: Beginning code generation
14/02/25 01:07:52 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `TopStocks` AS t
LIMIT 1
14/02/25 01:07:52 INFO orm.CompilationManager: HADOOP_HOME is /home/notroot/lab/software/hadoop-
1.0.3/libexec/..
Note: /tmp/sqoop-notroot/compile/4597911249aaf7f80749063b02e02065/TopStocks.java uses or overrides a
deprecated API.
Note: Recompile with -Xlint:deprecation for details.
14/02/25 01:08:57 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-
notroot/compile/4597911249aaf7f80749063b02e02065/TopStocks.jar
14/02/25 01:08:58 INFO mapreduce.ExportJobBase: Beginning export of TopStocks
14/02/25 01:10:43 INFO input.FileInputFormat: Total input paths to process : 1
14/02/25 01:10:43 INFO input.FileInputFormat: Total input paths to process : 1
14/02/25 01:10:43 INFO util.NativeCodeLoader: Loaded the native-hadoop library
14/02/25 01:10:43 WARN snappy.LoadSnappy: Snappy native library not loaded
14/02/25 01:11:15 INFO mapred.JobClient: Running job: job_201402250052_0002
14/02/25 01:11:16 INFO mapred.JobClient:  map 0% reduce 0%
14/02/25 01:17:55 INFO mapred.JobClient:  map 25% reduce 0%
14/02/25 01:20:30 INFO mapred.JobClient:  map 75% reduce 0%
14/02/25 01:21:32 INFO mapred.JobClient:  map 100% reduce 0%
14/02/25 01:23:02 INFO mapred.JobClient: Job complete: job_201402250052_0002
14/02/25 01:23:02 INFO mapred.JobClient: Counters: 18
14/02/25 01:23:02 INFO mapred.JobClient:   Job Counters
14/02/25 01:23:02 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=587134
14/02/25 01:23:02 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving
slots (ms)=0
14/02/25 01:23:02 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots
(ms)=0
14/02/25 01:23:02 INFO mapred.JobClient:     Launched map tasks=4
14/02/25 01:23:02 INFO mapred.JobClient:     Data-local map tasks=4
14/02/25 01:23:02 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=0
14/02/25 01:23:02 INFO mapred.JobClient:   File Output Format Counters
14/02/25 01:23:02 INFO mapred.JobClient:     Bytes Written=0
14/02/25 01:23:02 INFO mapred.JobClient:   FileSystemCounters
14/02/25 01:23:02 INFO mapred.JobClient:     HDFS_BYTES_READ=1396
14/02/25 01:23:02 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=120292
14/02/25 01:23:02 INFO mapred.JobClient:   File Input Format Counters
14/02/25 01:23:02 INFO mapred.JobClient:     Bytes Read=0
14/02/25 01:23:02 INFO mapred.JobClient:   Map-Reduce Framework
14/02/25 01:23:02 INFO mapred.JobClient:     Map input records=10
14/02/25 01:23:02 INFO mapred.JobClient:     Physical memory (bytes) snapshot=325398528
14/02/25 01:23:02 INFO mapred.JobClient:     Spilled Records=0
14/02/25 01:23:02 INFO mapred.JobClient:     CPU time spent (ms)=45370
14/02/25 01:23:02 INFO mapred.JobClient:     Total committed heap usage (bytes)=265682944
14/02/25 01:23:02 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=6974132224
14/02/25 01:23:02 INFO mapred.JobClient:     Map output records=10
14/02/25 01:23:02 INFO mapred.JobClient:     SPLIT_RAW_BYTES=565
14/02/25 01:23:02 INFO mapreduce.ExportJobBase: Transferred 1.3633 KB in 807.8873 seconds (1.728
bytes/sec)
14/02/25 01:23:02 INFO mapreduce.ExportJobBase: Exported 10 records.
notroot@ubuntu:~$
```

8.7 Output

SQOOP: Results

```
notroot@ubuntu:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 55
```

```
Server version: 5.5.28-0ubuntu0.12.04.2 (Ubuntu)

Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from stocks.TopStocks;
+--------+----------+----------+-------------+
| Symbol | MaxPrice | MinPrice | PriceChange |
+--------+----------+----------+-------------+
| WPO    |   547.58 |   295.56 |      252.02 |
| GOOG   |   629.51 |   433.63 |      195.88 |
| ISRG   |   393.92 |      211 |      182.92 |
| AZO    |   213.65 |   135.68 |       77.97 |
| MA     |   269.88 |      193 |       76.88 |
| AMZN   |   151.09 |    77.51 |       73.58 |
| PCLN   |   308.95 |   166.67 |      142.28 |
| CME    |  353.029 |   236.58 |     116.449 |
| AAPL   |   279.01 |   164.11 |       114.9 |
| MIL    |   106.99 |    26.73 |       80.26 |
+--------+----------+----------+-------------+
10 rows in set (0.00 sec)
```

Contents of stocks.TopStocks table matches with the content of the input file, thus shows successful execution of SQOOP export process.

9 Conclusion

With this project, I successfully demonstrated the ability and usability of the Hadoop framework for analyzing data that is in various forms. Requirements were met using various analytical platforms (Pig, Hive, SQOOP) that were built on Hadoop MapReduce framework. Even though the solution was tested with datasets that are considered small compared real world Big Data datasets, no additional changes are required to process large datasets. The program code developed in this effort can be used in a Hadoop environment running on a large cluster with tens or hundreds of nodes.