**Exp no 4**

**Date :**

**WORD COUNT ANALYSIS USING MAPREDUDE PROGRAM**

**AIM:** To perform word count operation using map reduce programming.

# Program:

## Main class

import org.apache.hadoop.conf.Configured; import org.apache.hadoop.fs.Path;

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

import org.apache.hadoop.mapred.FileInputFormat; import org.apache.hadoop.mapred.FileOutputFormat; import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf; import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class wordcount extends Configured implements Tool { @Override

public int run(String[] args) throws Exception { if(args.length<2)

{

System.out.println("Plz Give Input Output Directory Correctly"); return -1;

}

JobConf conf = new JobConf(wordcount.class); FileInputFormat.setInputPaths(conf,new Path(args[0])); FileOutputFormat.setOutputPath(conf, new Path(args[1])); conf.setMapperClass(wordmapper.class); conf.setReducerClass(wordreducer.class); conf.setMapOutputKeyClass(Text.class)

conf.setMapOutputValueClass(IntWritable.class); conf.setOutputKeyClass(Text.class); conf.setOutputValueClass(IntWritable.class); JobClient.runJob(conf);

return 0;

}

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

{

int exitcode = ToolRunner.run(new wordcount(), args); System.exit(exitcode);

}

}

## Mapper

import org.apache.hadoop.conf.Configured; import org.apache.hadoop.fs.Path;

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

import org.apache.hadoop.mapred.FileInputFormat; import org.apache.hadoop.mapred.FileOutputFormat; import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf; import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class wordcount extends Configured implements Tool {

@Override

public int run(String[] args) throws Exception {

if(args.length<2)

{

System.out.println("Plz Give Input Output Directory Correctly"); return -1;

}

JobConf conf = new JobConf(wordcount.class); FileInputFormat.setInputPaths(conf,new Path(args[0])); FileOutputFormat.setOutputPath(conf, new Path(args[1])); conf.setMapperClass(wordmapper.class); conf.setReducerClass(wordreducer.class); conf.setMapOutputKeyClass(Text.class); conf.setMapOutputValueClass(IntWritable.class); conf.setOutputKeyClass(Text.class); conf.setOutputValueClass(IntWritable.class); JobClient.runJob(conf);

return 0;

}

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

{

int exitcode = ToolRunner.run(new wordcount(), args); System.exit(exitcode);

}

}

## Reducer

import java.io.IOException;

import java.util.Iterator;

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

import org.apache.hadoop.mapred.MapReduceBase; import org.apache.hadoop.mapred.OutputCollector; import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

public class wordreducer extends MapReduceBase implements Reducer { public void reduce(Text key, Iterator values,

OutputCollector output, Reporter r) throws IOException {

int count=0; while(values.hasNext())

{

IntWritable i= values.next(); count+= i.get();

}

output.collect(key, new IntWritable(count));

}

}

## Execution procedure for Map Reduce program.

1. Creation of input file.
2. Move input file into HDFS.

$cd Desktop

$ls

$hadoop dfs -put cse.txt / cse

1. Create jar file.

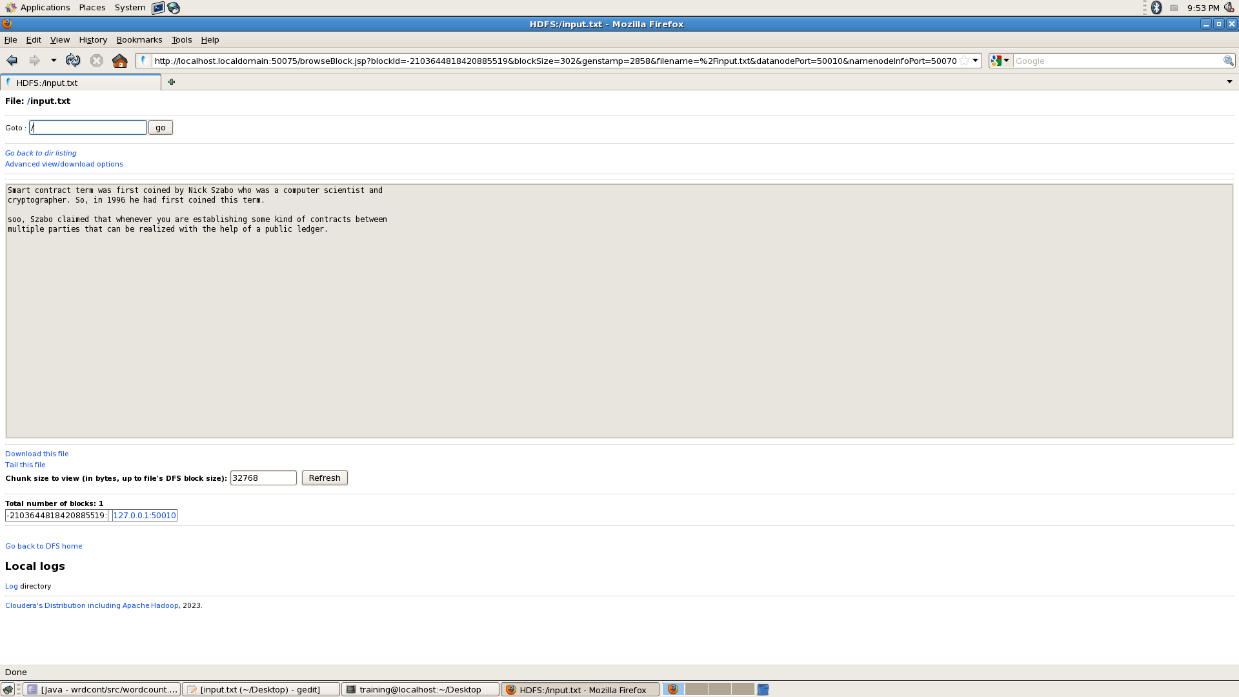
Right click on export option > select export > select java > enable Class Create jar file > browse > set-up-path

1. Execute jar file.

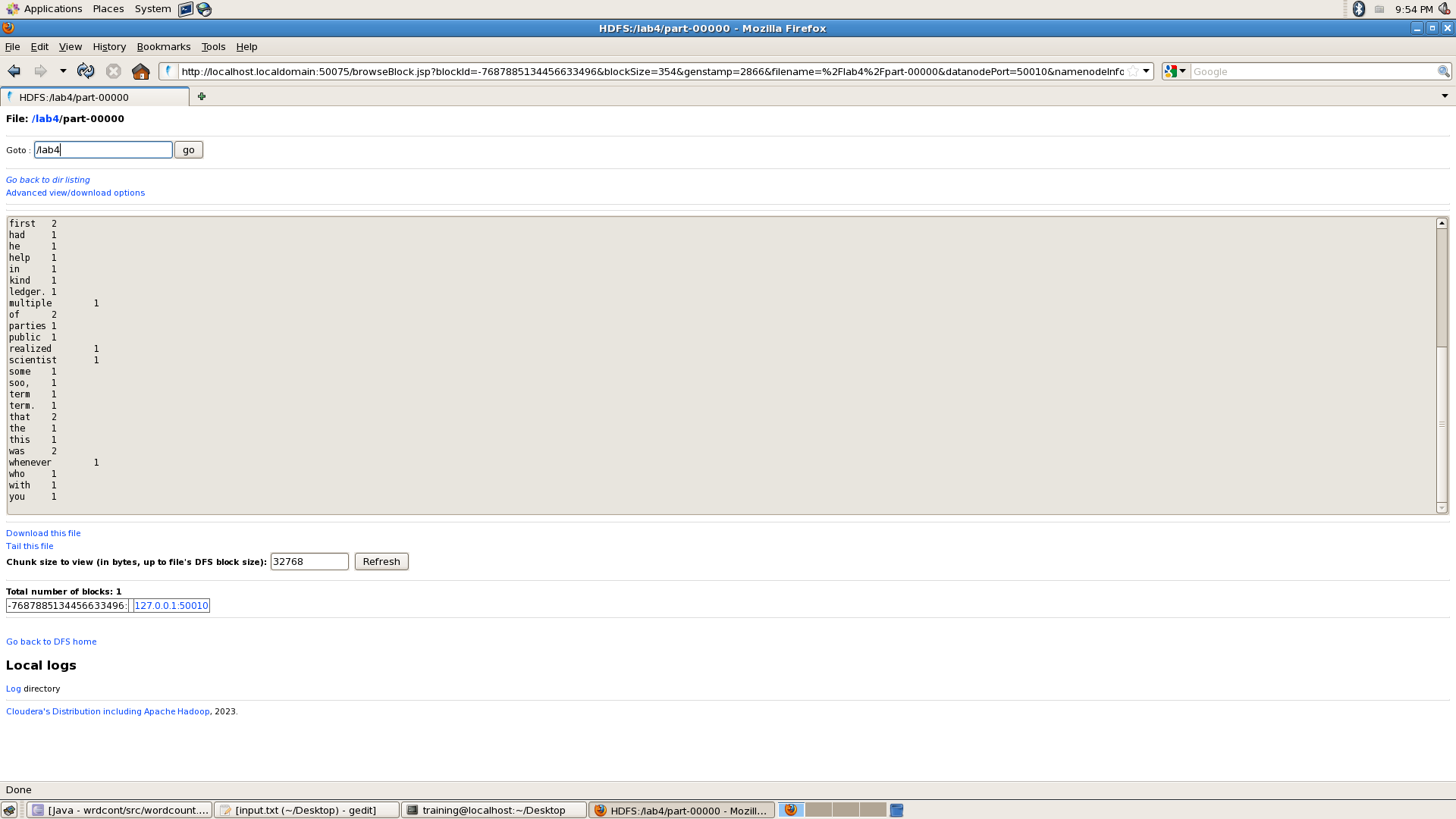
$hadoop jar exp4.jar wordcount / cse.txt / sree

1. See the result in HDFS. GoTo Mozilla Firefox. Browse Output File.

**INPUT FILE:**

****

# Output:

****

**Result :** Map reduce performed for word count using HDFS.

# Exp no 5

# Date :

**STOCK MARKET DATA ANALYSIS DATA USING MAPREDUCE PROGRAMMING**

**AIM:** To work with map reduce data and analyzing stock market data using map reduce program.

# Program:

## Main class

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.input.TextInputFormat; import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class MaxClosePrice {

public static void main(String[] args) throws Exception { if (args.length != 2) {

System.err.println("Usage: MaxClosePrice "); System.exit(-1);

}

// Define MapReduce job Job job = new Job();

job.setJarByClass(MaxClosePrice.class); job.setJobName("MaxClosePrice");

// Set input and output locations FileInputFormat.addInputPath(job, new Path(args[0])); FileOutputFormat.setOutputPath(job, new Path(args[1]));

// Set Input and Output formats job.setInputFormatClass(TextInputFormat.class); job.setOutputFormatClass(TextOutputFormat.class);

// Set Mapper and Reduce classes job.setMapperClass(MaxClosePriceMapper.class); job.setReducerClass(MaxClosePriceReducer.class);

// Output types job.setOutputKeyClass(Text.class); job.setOutputValueClass(FloatWritable.class);

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

}

}

## Mapper

/\*\*

* MaxClosePriceMapper.java
* [www.hadoopinrealworld.com](http://www.hadoopinrealworld.com/)
* This is a Mapper program to calculate Max Close Price from stock dataset using MapReduce

\*/

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 MaxClosePriceMapper extends Mapper { @Override

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

String line = value.toString(); String[] items = line.split(","); String stock = items[1];

Float closePrice = Float.parseFloat(items[6]); context.write(new Text(stock), new FloatWritable(closePrice));

}

}

Reducer

\*\*

* MaxClosePriceReducer.java
* [www.hadoopinrealworld.com](http://www.hadoopinrealworld.com/)
* This is a Reduce program to calculate Max Close Price from stock dataset using MapReduce

\*/

import java.io.IOException;

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

import org.apache.hadoop.mapreduce.Reducer; public class MaxClosePriceReducer

extends Reducer {

@Override

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

float maxClosePrice = Float.MIN\_VALUE;

//Iterate all temperatures for a year and calculate maximum for (FloatWritable value : values) {

maxClosePrice = Math.max(maxClosePrice, value.get());

}

//Write output

context.write(key, new FloatWritable(maxClosePrice));

}

}

## Execution procedure for program.

1. Creation of input file.
2. Move input file into HDFS.

$cd Desktop

$ls

$hadoop dfs -put stock.txt / stock

1. Create jar file.

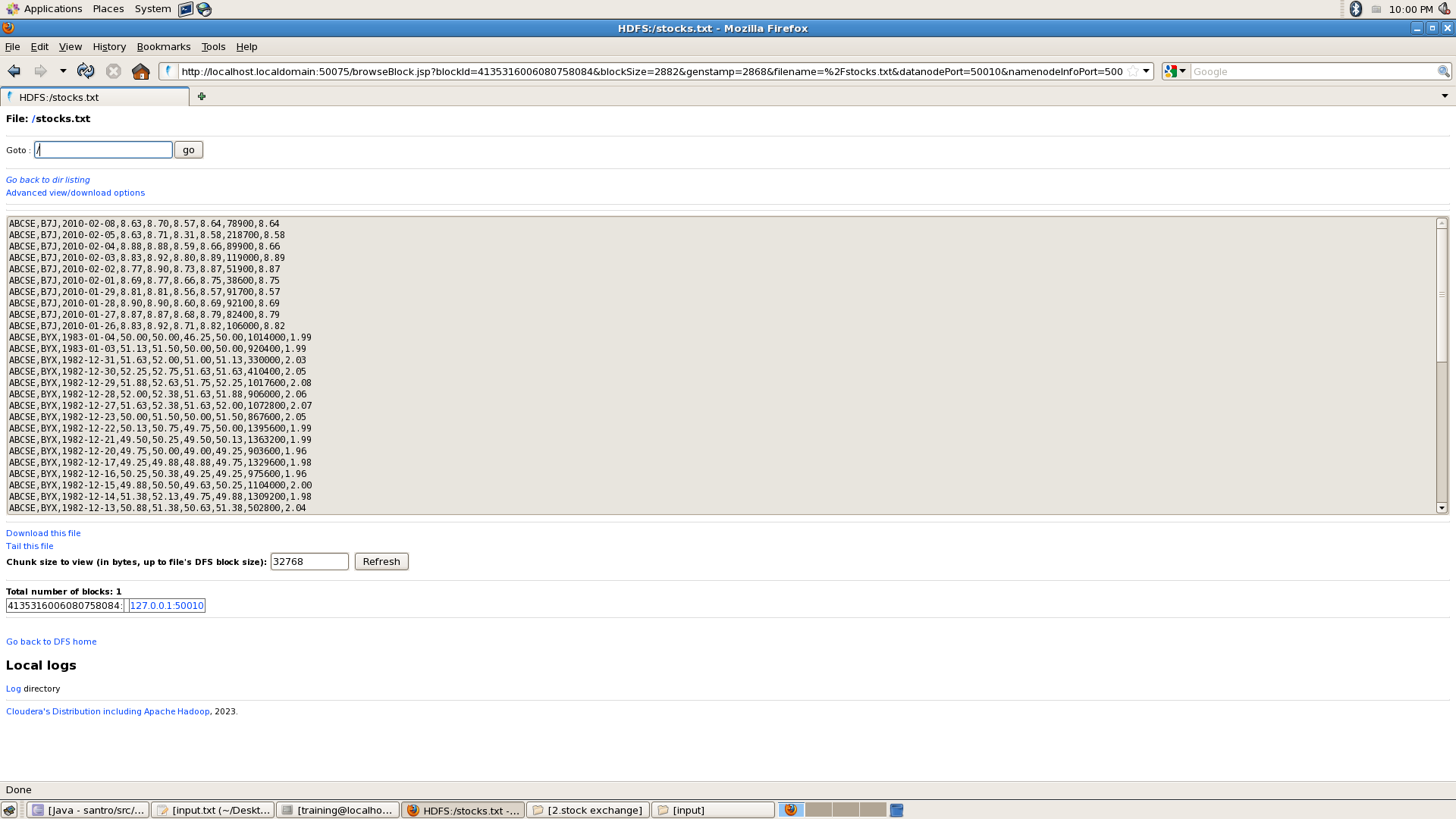
Right click on export option > select export > select java > enable Class Create jar file > browse > set-up-path

1. Execute jar file.

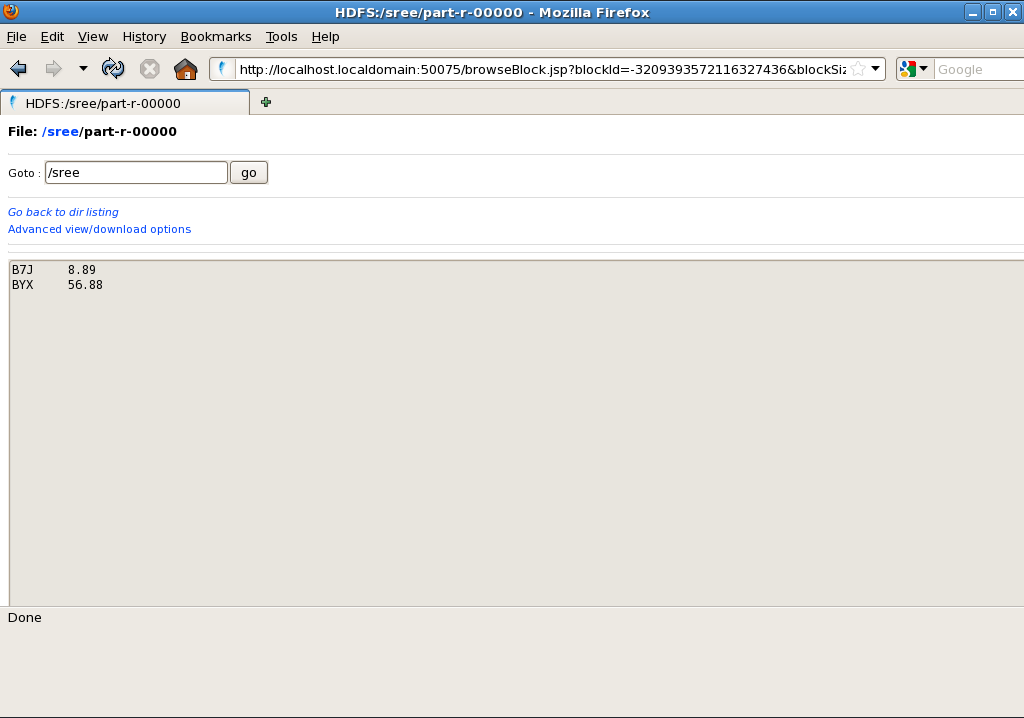
$hadoop jar exp5.jar MaxClosePrice / stocks.txt / sree

1. See the result in HDFS. GoTo Mozilla Firefox. Browse Output File.

**INPUT FILE:**



# Output



## **Result:** Hence, map reduce data and analyzing stock market data using map reduce program.