Exercise – 4:

Processing WordCount in MapReduce

Driver Class of WordCount: (WordCountJob.java)

```
package com.sample;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.ToolRunner;
import  org.apache.hadoop.util.Tool;

public class WordCountJob implementsTool
{
        private Configuration conf;
        @Override
        public Configuration getConf()
        {
        return conf;
        }
        @Override
        public void setConf(Configuration conf)
        {
                this.conf=conf;
        }
        @Override
        public int run(String []args)throws Exception
        {
                Job wordcountjob=new Job(getConf());
                wordcountjob.setJobName("mat word count");
                wordcountjob.setJarByClass(this.getClass());
                wordcountjob.setMapperClass(WordCountMapper.class);
                wordcountjob.setReducerClass(WordCountReducer.class);
                wordcountjob.setMapOutputKeyClass(Text.class);
                wordcountjob.setMapOutputValueClass(LongWritable.class);
                wordcountjob.setOutputKeyClass(Text.class);
                wordcountjob.setOutputValueClass(LongWritable.class);
                FileInputFormat.setInputPaths(wordcountjob,newPath(args[0]));
                FileOutputFormat.setOutputPath(wordcountjob,new Path(args[1]));
                return wordcountjob.waitForCompletion(true)==true? 0:1;
        }
        public static void main(String []args)throws Exception
        {
               ToolRunner.run(new Configuration(),new WordCountJob(),args);
```

```
    }    }
```

Mapper Class of WordCount: (WordCountMapper.java)

```
package com.sample;
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper extends Mapper<LongWritable, Text, Text, LongWritable>
        {
         private Text temp = new Text();
         private final static LongWritable one = new LongWritable(1);

        @Override
        protected void map(LongWritable key, Text value, Context context)
                                        throws IOException, InterruptedException
                {
                String line = value.toString();
                String[] words = line.split(" ");
                for (int i = 0; i < words.length; i++)
                {
                context.write(new Text(words[i]), one);
                }
                }

        }
```

Reducer Class of WordCount: (WordCountReducer.java)

```
package com.sample;
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.io.Text;

public class WordCountReducer extends Reducer<Text,LongWritable,Text,LongWritable>
{
        @Override
        protected void reduce(Text key,Iterable<LongWritable> value,Context context)throws
                                                                IOException,InterruptedException
        {
                long sum=0; while(value.iterator().hasNext())
                {
                        sum+=value.iterator().next().get();
                }
                context.write(key,new LongWritable(sum));
        }
}
```

Steps to Execute WordCount mapreduce program in hadoop:

1. Create a wordcount project in Eclipse and create driver, mapper and reducer classes in it.

2. Go to project build path and add all the jar files (or) the following jars:
   hadoop-common. jar (/usr/lib/hadoop/hadoop-common.jar)
   hadoop-core. jar (/usr/lib/hadoop-0.20mapreduce/hadoop-core.jar)

3. Now, once the errors are resolved, right click on your project and export the jarfile.(wordcount.jar)

4. Move the input dataset of wordcount (wc1.txt and wc2.txt) from local filesystem to hadoopfilesystem.
   hadoop fs -mkdir /user/cloudera/wordcount_input
   hadoop fs -put wc1.txt /user/cloudera/wordcount_input
   hadoop fs -put wc2.txt/user/cloudera/wordcount_input

5. Execute wordcount mapreduce program in hadoop.
   hadoop jar WordCountJob.jar com.sample.WordCountDriver
   /user/cloudera/WordCount/WC_Input /user/cloudera/wordcount_output

6. Check the output of wordcount in hadoopfilesystem
   hadoop fs -cat /user/cloudera/wordcount_output/part-r-00000

Exercise – 5:

Processing Weather Dataset in MapReduce

Driver Class of Weather Report: (WeatherReport.java)

```
package weather;
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.mapreduce.lib.output.MultipleOutputs;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class WeatherReport {

        public static String caOutputName = "California";
        public static String nyOutputName = "Newyork";
        public static String njOutputName = "Newjersy";

        public static void main(String[] args) throws Exception
        {
                Configuration conf = new Configuration();
                 Job job = new Job(conf, "Weather Report");
                 job.setJarByClass(WeatherReport.class);

                 job.setMapperClass(WeatherMapper.class);
                 job.setReducerClass(WeatherReducer.class);

                 job.setMapOutputKeyClass(Text.class);
                 job.setMapOutputValueClass(FloatWritable.class);
                 job.setOutputKeyClass(Text.class);
                 job.setOutputValueClass(Text.class);
                 MultipleOutputs.addNamedOutput(job, caOutputName, TextOutputFormat.class,
                                                                 Text.class, Text.class);
                 MultipleOutputs.addNamedOutput(job, nyOutputName, TextOutputFormat.class,
                                                                 Text.class, Text.class);
                 MultipleOutputs.addNamedOutput(job, njOutputName, TextOutputFormat.class,
                                                                 Text.class, Text.class);
                 FileInputFormat.addInputPath(job, new Path(args[0]));
                 FileOutputFormat.setOutputPath(job, new Path(args[1]));
                 System.exit(job.waitForCompletion(true) ? 0 : 1);
        }
}
```

Mapper Class of Weather Report: (WeatherMapper.java)

```
package weather;
import java.io.IOException;
importjava.util.StringTokenizer;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WeatherMapper extends Mapper<Object, Text, Text, FloatWritable>
{
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text dayReport, Context context) throws IOException,
                                                                        InterruptedException
        {
                StringTokenizer st2 = new StringTokenizer(dayReport.toString(), "\t");

                 int counter = 0;
                 String cityDateString = "";
                 String maxTempTime = "";
                 String minTempTime = "";
                 String curTime = "";
                 float curTemp = 0;
                 float minTemp = Float.MAX_VALUE;
                 float maxTemp = Float.MIN_VALUE;

                 while (st2.hasMoreElements())
                 {
                        if (counter == 0)
                        {
                                cityDateString = st2.nextToken();
                         }
                         else
                         {
                                if (counter % 2 == 1)
                                {
                                        curTime = st2.nextToken();
                                }
                                else if (counter % 2 == 0)
                                {
                                curTemp =Float.parseFloat(st2.nextToken());
                                if (minTemp > curTemp)
                                {
                                minTemp = curTemp;
                                minTempTime = curTime;
```

```
                    }
                    else if (maxTemp < curTemp)
                    {
                    maxTemp = curTemp;
                    maxTempTime = curTime;
                    }
                    }
                }
                counter++;
            }

            FloatWritable fValue = new FloatWritable();
            Text cityDate = new Text();

            fValue.set(maxTemp);
            cityDate.set(cityDateString);
            context.write(cityDate, fValue);

            fValue.set(minTemp);
            cityDate.set(cityDateString);
            context.write(cityDate, fValue);
        }
}
```

Reducer Class of Weather Report: (WeatherReducer.java)

```
package weather;
import java.io.IOException;
import org.apache.hadoop.io.FloatWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.output.MultipleOutputs;

public class WeatherReducer extends Reducer<Text, FloatWritable, Text, Text>
{
// hadoop,1,1,1,
        MultipleOutputs<Text, Text> mos;
        public void setup(Context context)
        {
                mos = new MultipleOutputs<Text, Text>(context);
        }

        public void reduce(Text key, Iterable<FloatWritable> values, Context context)
                                                        throws IOException, InterruptedException
                        {
                         int counter = 0;
                        float f1 = 0, f2 = 0;
                        Text result = newText();


                        for (FloatWritable value : values)
                        {
                                if (counter == 0)
                                  f1 = value.get();
                                else
                                  f2 = value.get();
                                counter = counter + 1;
                        }
                        if (f1 > f2)
                        {
                         context.write(key, newText(Float.toString(f2)+"\t"+Float.toString(f1)));
                         result = new Text(Float.toString(f2) + "\t" +Float.toString(f1));
                        }
                        else
                        {
                         context.write(key, newText(Float.toString(f1)+"\t"+Float.toString(f2)));
                         result = new Text(Float.toString(f1) + "\t" +Float.toString(f2));
                        }
                String fileName = "";
                if (key.toString().contains("CA"))
                {
```

```
                fileName = WeatherReport.caOutputName;
                }
                else if (key.toString().contains("NY"))
                {
                fileName = WeatherReport.nyOutputName;
                }
                else if (key.toString().contains("NJ"))
                {
                fileName = WeatherReport.njOutputName;
                }

                String strArr[] = key.toString().split("_");
                key.set(strArr[1]);
                mos.write(fileName, key, result);
        }
        @Override
        public void cleanup(Context context) throws IOException, InterruptedException
        {
                mos.close();
        }
}
```


Steps to Execute Weather Report mapreduce program in hadoop:

1. Create a weather project in Eclipse and create driver,mapper and reducer classes in it.
2. Go to project build path and add all the jar files (or) the following jars:
   hadoop-common. jar (/usr/lib/hadoop/hadoop-common.jar)
   hadoop-core. jar (/usr/lib/hadoop-0.20mapreduce/hadoop-core.jar)
3. Now, once the errors are resolved, right click on your project and export the jarfile.
4. Move the input dataset of weather report (wreport.txt) from local filesystem to hadoopfilesystem.

   hadoop fs -mkdir /user/cloudera/weather_input
   hadoop fs -put wreport.txt /user/cloudera/weather_input
5. Execute mapreduce program in hadoop.

   hadoop jar weather.jar weather.WeatherReport
   /user/cloudera/weather_input /user/cloudera/weather_output
6. Check the output of weather report in 3 cities of US in hadoop filesystem

   hadoop fs -cat /user/cloudera/weather_output/California-r-00000
   hadoop fs -cat /user/cloudera/weather_output/Newjersey-r-00000

hadoop fs -cat /user/cloudera/weather_output/Newyork-r-00000

Exercise – 6:

Processing Matrix multiplication in MapReduce

Driver Class of Matrix Multiplication: (MatrixMultiplication.java)

```
package matrix;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
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 MatrixMultiplication
{

        public static void main(String[] args) throws Exception
        {
                Configuration conf = new Configuration();
                // A is an m-by-n matrix; B is an n-by-p matrix. conf.set("m", "2");
                conf.set("n", "5");
                conf.set("p", "3");

                Job job = new Job(conf, "MatrixMultiplication");
                job.setJarByClass(MatrixMultiplication.class);
                job.setOutputKeyClass(Text.class);
                job.setOutputValueClass(Text.class);
                job.setMapperClass(MatrixMapper.class);
                job.setReducerClass(MatrixReducer.class);
                job.setInputFormatClass(TextInputFormat.class);
                job.setOutputFormatClass(TextOutputFormat.class);
                FileInputFormat.addInputPath(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));
                job.waitForCompletion(true);
        }
}
```

Mapper Class of Matrix Multiplication: (MatrixMapper.java)

```
package matrix;
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

    public class MatrixMapper extends Mapper<LongWritable, Text, Text, Text>
    {
      public void map(LongWritable key, Text value, Context context) throws
                                                    IOException, InterruptedException
      {
                Configuration conf = context.getConfiguration();
                int m = Integer.parseInt(conf.get("m"));
                int p = Integer.parseInt(conf.get("p"));
                String line = value.toString();
                String[] indicesAndValue = line.split(",");
                Text outputKey = new Text();
                Text outputValue = new Text();
                if (indicesAndValue[0].equals("A"))
                {
                for (int k = 0; k < p; k++)
                {
                        outputKey.set(indicesAndValue[1] + "," + k);
                        outputValue.set("A," + indicesAndValue[2] + "," + indicesAndValue[3]);
                }
                }
                else
                {
                         context.write(outputKey, outputValue);

                         for (int i = 0; i < m; i++)
                         {
                         outputKey.set(i + "," + indicesAndValue[2]);
                         outputValue.set("B," + indicesAndValue[1] + "," + indicesAndValue[3]);
                          }
                 }
                     context.write(outputKey, outputValue);

       }
       }
```

Reducer Class of Matrix Multiplication: (MatrixReducer.java)

```
package matrix;
import java.io.IOException;
import java.util.HashMap;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

 public class MatrixReducer extends Reducer<Text, Text, Text, Text>
 {
        public void reduce(Text key, Iterable<Text> values, Context context) throws
                                                                IOException, InterruptedException
                {
                String[] value;
                HashMap<Integer, Float> hashA = new HashMap<Integer,Float>();
                HashMap<Integer, Float> hashB = new HashMap<Integer, Float>();
                for (Text val : values)
                {
                value = val.toString().split(",");
                if (value[0].equals("A"))
                {
                hashA.put(Integer.parseInt(value[1]),
                Float.parseFloat(value[2]));
                }
                else
                {
                hashB.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));
                }
                }
                int n = Integer.parseInt(context.getConfiguration().get("n"));
                float result = 0.0f;
                float a_ij;
                float b_jk;
                for (int j = 0; j < n; j++)
                {
                a_ij = hashA.containsKey(j) ? hashA.get(j) : 0.0f;
                b_jk = hashB.containsKey(j) ? hashB.get(j) : 0.0f;
                result += a_ij * b_jk;
                }
                if (result != 0.0f)
                {
                context.write(null, new Text(key.toString() + "," + Float.toString(result)));
                }
                }}
```

Steps to Execute Matrix Multiplication mapreduce program in hadoop:

1. Create a Matrix project in Eclipse and create driver, mapper and reducer classes in it.

2. Go to project build path and add all the jar files (or) the following jars:

   hadoop-common. jar(/usr/lib/hadoop/hadoop-common.jar)
   hadoop-core. jar (/usr/lib/hadoop-0.20mapreduce/hadoop- core.jar)

3. Now, once the errors are resolved, right click on your project and export the jarfile.

4. Move the input dataset of Matrix Multiplication (data1.txt or data2.txt) from local filesystem to hadoopfilesystem.

   hadoop fs -mkdir /user/cloudera/matrix_input hadoop fs -put data.txt/user/cloudera/matrix_input

5. Execute matrix multiplication mapreduce program inhadoop.

   hadoop jar matrix.jar matrix.MatrixMultiplication /user/cloudera/matrix_input /user/cloudera/matrix_output

6. Check the output of Matrix Multiplication in hadoop filesystem

   hadoop fs -cat/user/cloudera/matrix_output/part-r-00000