Assignment 11

File: WC_Mapper.java

```
package com.javatpoint;

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class WC_Mapper extends MapReduceBase implements Mapper<LongWritable,Text,Text,IntWritable>{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(LongWritable key, Text value,OutputCollector<Text,IntWritable> output,
           Reporter reporter) throws IOException{
        String line = value.toString();
        StringTokenizer  tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()){
            word.set(tokenizer.nextToken());
            output.collect(word, one);
        }
    }

}
```

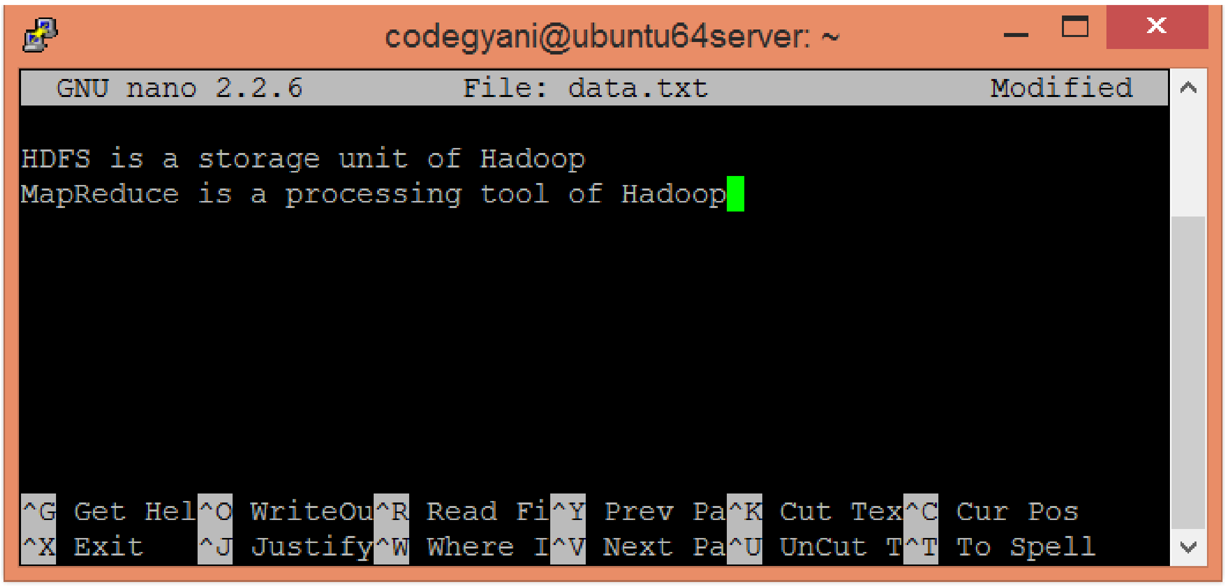
File: WC_Reducer.java

```
package com.javatpoint;
    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 WC_Reducer  extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text,IntWritable> output,
     Reporter reporter) throws IOException {
    int sum=0;
    while (values.hasNext()) {
    sum+=values.next().get();
    }
    output.collect(key,new IntWritable(sum));
    }
    }
```

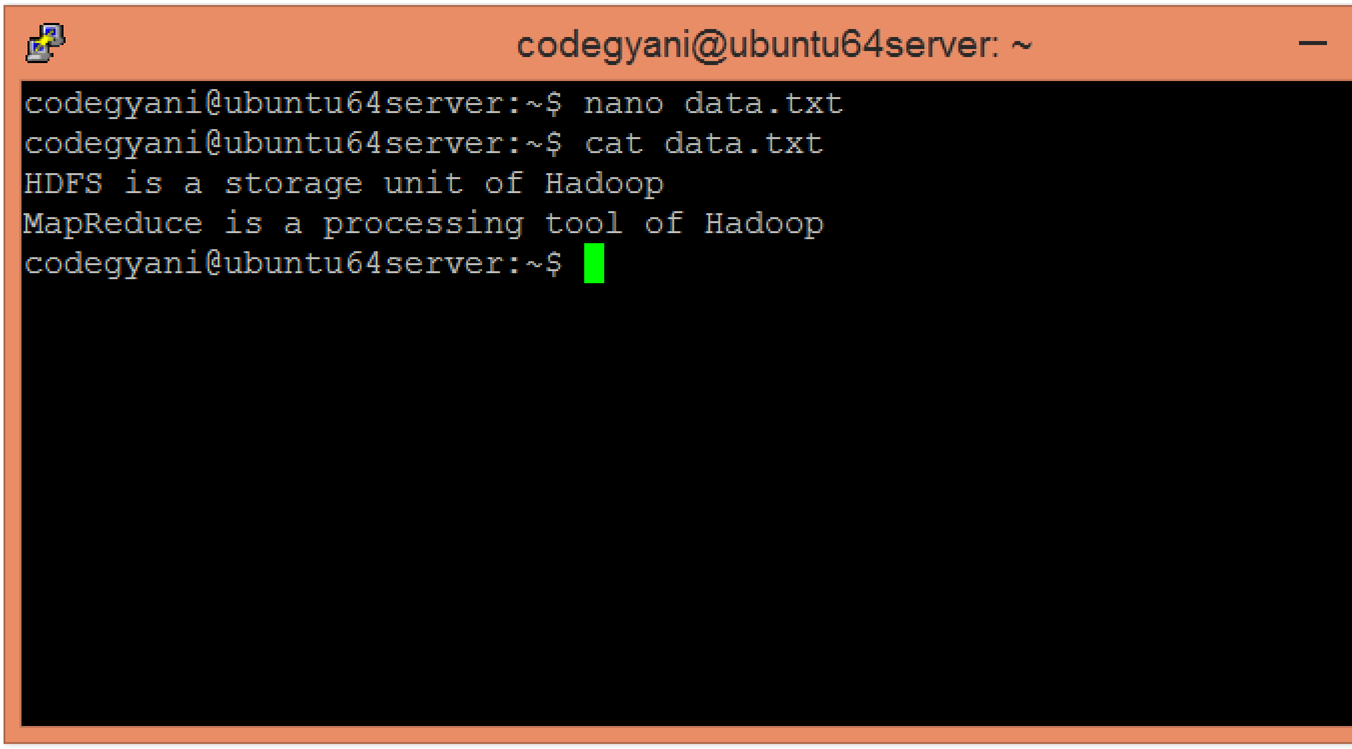
```
package com.javatpoint;

    import java.io.IOException;
    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;
```

```java
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
public class WC_Runner {
    public static void main(String[] args) throws IOException{
        JobConf conf = new JobConf(WC_Runner.class);
        conf.setJobName("WordCount");
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        conf.setMapperClass(WC_Mapper.class);
        conf.setCombinerClass(WC_Reducer.class);
        conf.setReducerClass(WC_Reducer.class);
        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);
        FileInputFormat.setInputPaths(conf,new Path(args[0]));
        FileOutputFormat.setOutputPath(conf,new Path(args[1]));
        JobClient.runJob(conf);
    }
}
```

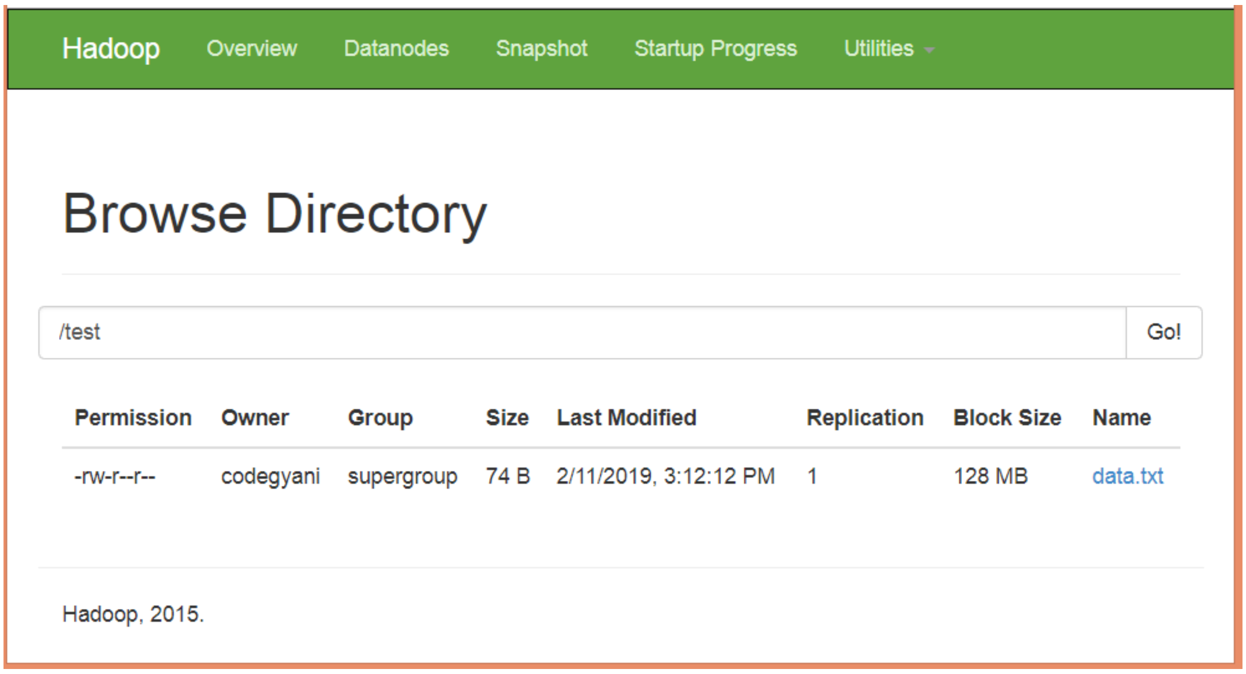
Output:



```
codegyani@ubuntu64server: ~
  GNU nano 2.2.6            File: data.txt                       Modified

HDFS is a storage unit of Hadoop
MapReduce is a processing tool of Hadoop









^G Get Hel^O WriteOu^R Read Fi^Y Prev Pa^K Cut Tex^C Cur Pos
^X Exit   ^J Justify^W Where I^V Next Pa^U UnCut T^T To Spell
```



```
codegyani@ubuntu64server:~$ nano data.txt
codegyani@ubuntu64server:~$ cat data.txt
HDFS is a storage unit of Hadoop
MapReduce is a processing tool of Hadoop
codegyani@ubuntu64server:~$
```

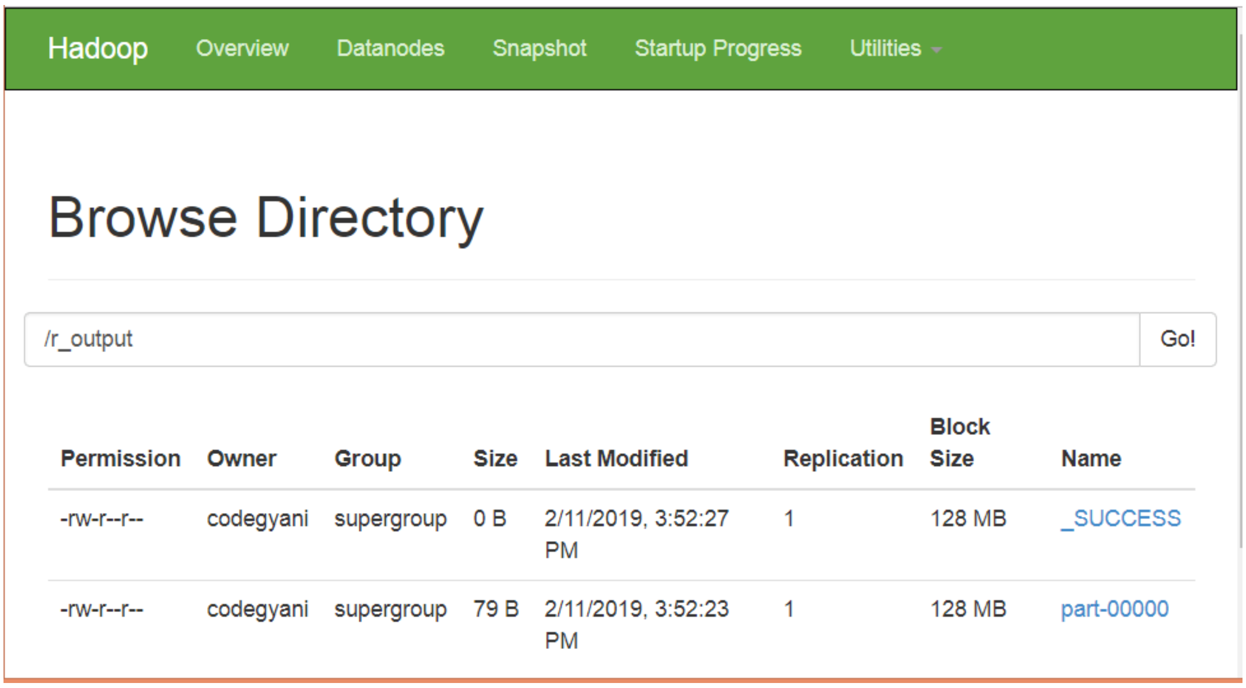


Browse Directory

/test Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	codegyani	supergroup	74 B	2/11/2019, 3:12:12 PM	1	128 MB	data.txt

Hadoop, 2015.

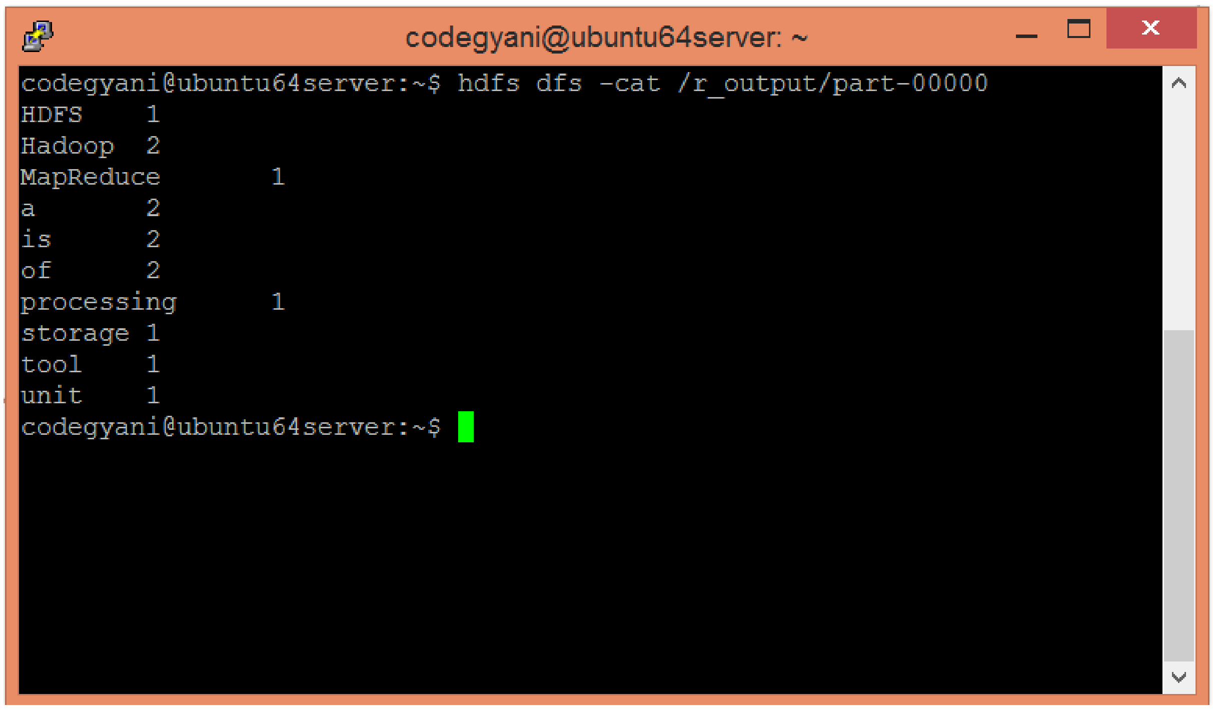


Browse Directory

/r_output

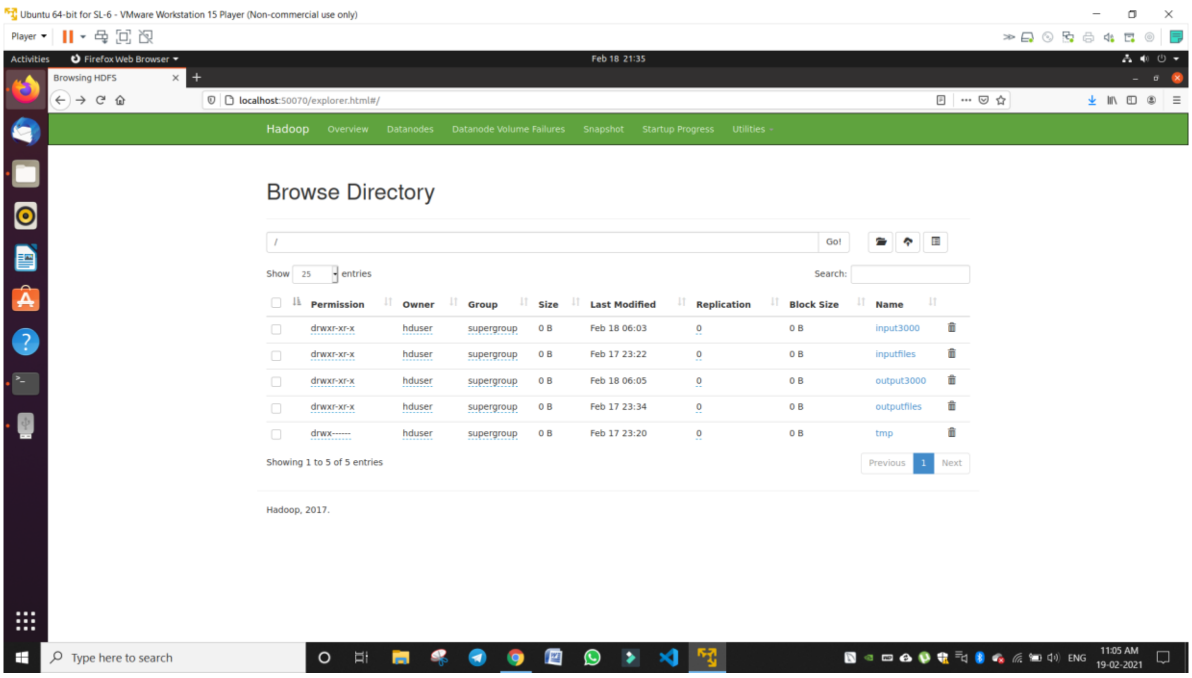
Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	codegyani	supergroup	0 B	2/11/2019, 3:52:27 PM	1	128 MB	_SUCCESS
-rw-r--r--	codegyani	supergroup	79 B	2/11/2019, 3:52:23 PM	1	128 MB	part-00000



```
codegyani@ubuntu64server:~$ hdfs dfs -cat /r_output/part-00000
HDFS    1
Hadoop  2
MapReduce       1
a       2
is      2
of      2
processing      1
storage 1
tool    1
unit    1
codegyani@ubuntu64server:~$
```

Assignment 12



Ubuntu 64-bit for SL-6 - VMware Workstation 15 Player (Non-commercial use only)

localhost:50070/explorer.html#/

Browse Directory

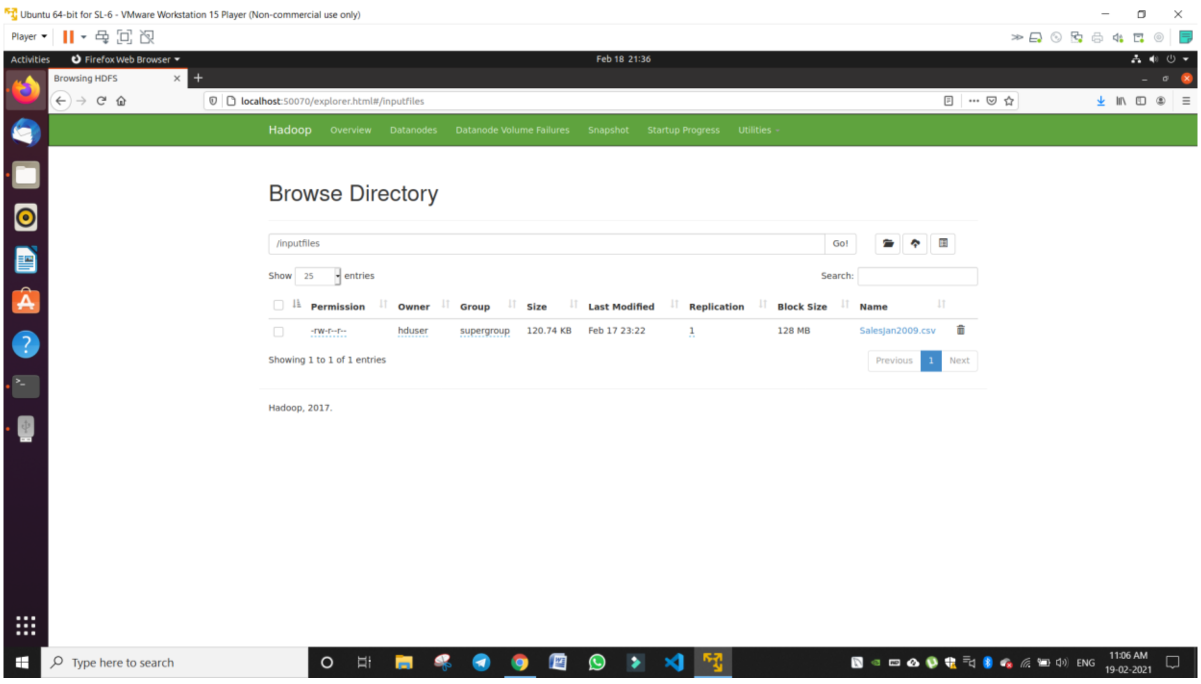
/

Show 25 entries

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
	drwxr-xr-x	hduser	supergroup	0 B	Feb 18 06:03	0	0 B	input3000
	drwxr-xr-x	hduser	supergroup	0 B	Feb 17 23:22	0	0 B	inputfiles
	drwxr-xr-x	hduser	supergroup	0 B	Feb 18 06:05	0	0 B	output3000
	drwxr-xr-x	hduser	supergroup	0 B	Feb 17 23:34	0	0 B	outputfiles
	drwx------	hduser	supergroup	0 B	Feb 17 23:20	0	0 B	tmp

Showing 1 to 5 of 5 entries

Hadoop, 2017.



Ubuntu 64-bit for SL-6 - VMware Workstation 15 Player (Non-commercial use only)

localhost:50070/explorer.html#/inputfiles

Browse Directory

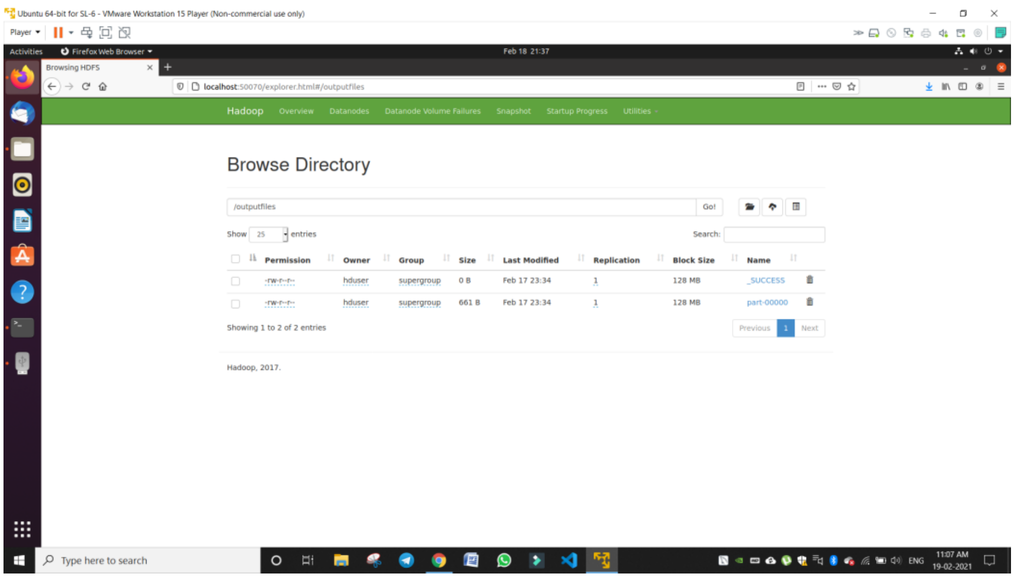
/inputfiles

Show 25 entries

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
	-rw-r--r--	hduser	supergroup	120.74 KB	Feb 17 23:22	1	128 MB	SalesJan2009.csv

Showing 1 to 1 of 1 entries

Hadoop, 2017.



Browse Directory

/outputfiles Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	supergroup	0 B	Feb 17 23:34	1	128 MB	_SUCCESS
-rw-r--r--	hduser	supergroup	661 B	Feb 17 23:34	1	128 MB	part-00000

Showing 1 to 2 of 2 entries

Previous 1 Next

Hadoop, 2017.

Assignment 13

Program :

```
package com.org.vasanth.weather;


import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
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.mapred.KeyValueTextInputFormat;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

/**
 *
 *
 * This is an Hadoop Map/Reduce application for Working on weather data It reads
 * the text input files, breaks each line into stations weather data and finds
 * average for temperature , dew point , wind speed. The output is a locally
 * sorted list of stations and its 12 attribute vector of average temp , dew ,
 * wind speed of 4 sections for each month.
 *
 * To run: bin/hadoop jar target/weather-1.0.jar [-m <i>maps</i>] [-r
 * <i>reduces</i>] <i>in-dir for job 1</i> <i>out-dir for job 1</i> <i>out-dir
 * for job 2</i>
 */
public class Weather extends Configured implements Tool {
            final long DEFAULT_SPLIT_SIZE = 128 * 1024 * 1024;

            /**
             * Map Class for Job 1
             *
             * For each line of input, emits key value pair with
             * station_yearmonth_sectionno as key and 3 attribute vector with
             * temperature , dew point , wind speed as value.Map method will strip the
             * day and hour from field and replace it with section no (
             * <b>station_yearmonth_sectionno</b>, <b><temperature,dew point , wind
             * speed></b>).
             */
            public static class MapClass extends MapReduceBase
```

```
		implements Mapper<LongWritable, Text, Text, Text> {

	private Text word = new Text();
	private Text values = new Text();

	public void map(LongWritable key, Text value,
				OutputCollector<Text, Text> output,
				Reporter reporter) throws IOException {
		String line = value.toString();
		StringTokenizer itr = new StringTokenizer(line);
		int counter = 0;
		String key_out = null;
		String value_str = null;
		boolean skip = false;
		loop:while (itr.hasMoreTokens() && counter<13) {
			String str = itr.nextToken();
			switch (counter) {
			case 0:
				key_out = str;
				if(str.contains("STN")){//Ignoring rows where station id is
all 9
					skip = true;
					break loop;
				}else{
					break;
				}
			case 2:
				int hour =
Integer.valueOf(str.substring(str.lastIndexOf("_")+1, str.length()));
				str = str.substring(4,str.lastIndexOf("_")-2);
				/*if(hour<=5){
					str = str.concat("_section4");
				}else if(hour>5 && hour<=11){
					str = str.concat("_section1");
				}else if(hour>11 && hour<=17){
					str = str.concat("_section2");
				}else if(hour>17 && hour<=23){
					str = str.concat("_section3");
				}*/

				if(hour>4 && hour<=10){
					str = str.concat("_section1");
				}else if(hour>10 && hour<=16){
					str = str.concat("_section2");
				}else if(hour>16 && hour<=22){
					str = str.concat("_section3");
				}else{
					str = str.concat("_section4");
				}



				key_out = key_out.concat("_").concat(str);
				break;
			case 3://Temperature
				if(str.equals("9999.9")){//Ignoring rows where
temperature is all 9
					skip = true;
```

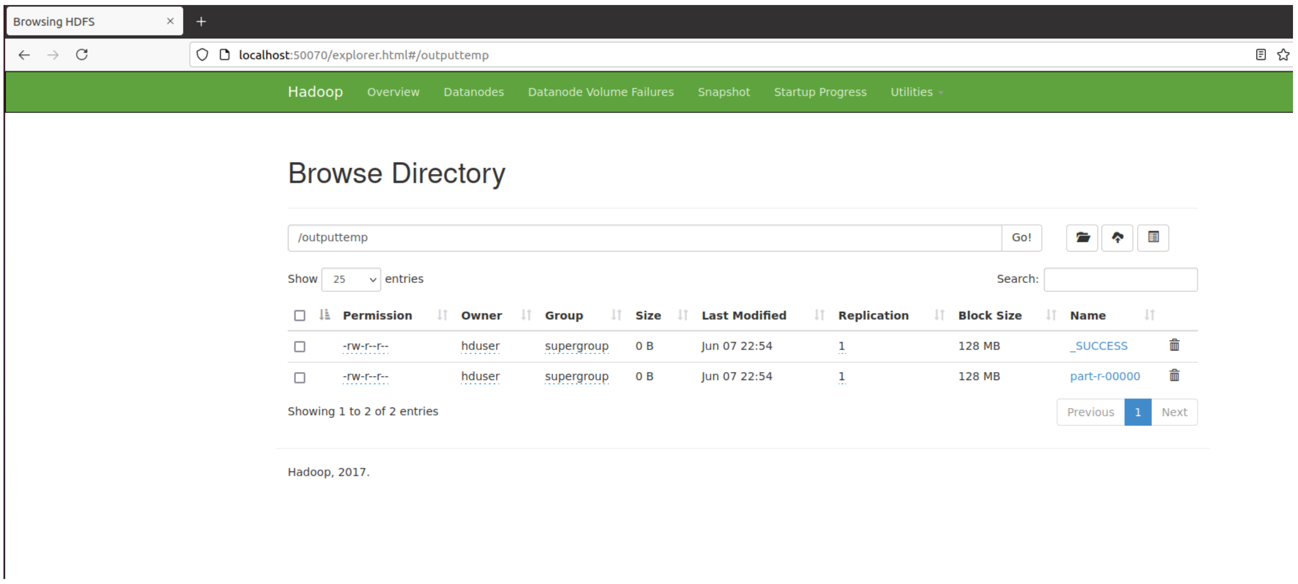
```
							break loop;
						}else{
							value_str = str.concat(" ");
							break;
						}
					case 4://Dew point
						if(str.equals("9999.9")){//Ignoring rows where dew point
is all 9
							skip = true;
							break loop;
						}else{
							value_str = value_str.concat(str).concat(" ");
							break;
						}
					case 12://Wind speed
						if(str.equals("999.9")){//Ignoring rows where wind speed
is all 9
							skip = true;
							break loop;
						}else{
							value_str = value_str.concat(str).concat(" ");
							break;
						}
					default:
						break;
					}
					counter++;
				}
				if(!skip){
					word.set(key_out);
					values.set(value_str);
					output.collect(word, values);
				}
			}
		}

		/**
		 * Map Class for Job 2
		 *
		 * For each input, emits key value pair with station_yearmonth as key and 3
		 * attribute vector with temperature , dew point , wind speed as value by
		 * stripping the section no from key and adding section no into vector value
		 * ( <b>station_yearmonth</b>, <b><temperature,dew point , wind speed></b>).
		 */
		public static class MapClassForJob2 extends MapReduceBase
				implements Mapper<Text, Text, Text, Text> {
			private Text key_text = new Text();
			private Text value_text = new Text();
			@Override
			public void map(Text key, Text value,
					OutputCollector<Text, Text> output, Reporter reporter) throws
IOException {

				String str = key.toString();
				String station = str.substring(str.lastIndexOf("_")+1, str.length());
				str = str.substring(0,str.lastIndexOf("_"));
				key_text.set(str);
```

```
                        StringTokenizer itr = new StringTokenizer(value.toString());
                        String str_out = station.concat("<");
                        while (itr.hasMoreTokens()) {
                                String nextToken = itr.nextToken(" ");
                                str_out = str_out.concat(nextToken);
                                str_out = ((itr.hasMoreTokens()) ? str_out.concat(",") :
str_out.concat(">"));
                        }
                        value_text.set(str_out);
                        output.collect(key_text,value_text);
                }
        }

        /**
         * Reducer Class for Job 1
         *
         * A reducer class that just emits 3 attribute vector with average
         * temperature , dew point , wind speed for each of the section of the month
         * for each input
         */
        public static class Reduce extends MapReduceBase
                        implements Reducer<Text, Text, Text, Text> {
                private Text value_out_text = new Text();
                public void reduce(Text key, Iterator<Text> values,
                                OutputCollector<Text, Text> output, Reporter reporter) throws
IOException {
                        double sum_temp = 0;
                        double sum_dew = 0;
                        double sum_wind = 0;
                        int count = 0;

                        while (values.hasNext()) {

                                String str = values.next().toString();

                                StringTokenizer itr = new StringTokenizer(str);
                                int count_vector = 0;
                                while (itr.hasMoreTokens()) {
                                        String nextToken = itr.nextToken(" ");
                                        if(count_vector==0){
                                                sum_temp += Double.valueOf(nextToken);
                                        }
                                        if(count_vector==1){
                                                sum_dew += Double.valueOf(nextToken);
                                        }
                                        if(count_vector==2){
                                                sum_wind += Double.valueOf(nextToken);
                                        }
                                        count_vector++;
                                }
                                count++;
                        }
                        double avg_tmp = sum_temp / count;
                        double avg_dew = sum_dew / count;
                        double avg_wind = sum_wind / count;
```

```
				System.out.println(key.toString()+" count is "+count+" sum of temp is
"+sum_temp+" sum of dew is "+sum_dew+" sum of wind is "+sum_wind+"\n");

				String value_out = String.valueOf(avg_tmp).concat("
").concat(String.valueOf(avg_dew)).concat(" ").concat(String.valueOf(avg_wind));
				value_out_text.set(value_out);
				output.collect(key, value_out_text);
			}
	}

	/**
	 * Reducer Class for Job 2
	 *
	 * A reducer class that just emits 12 attribute vector with average
	 * temperature , dew point , wind speed for all section of the month
	 * for each input
	 */
	public static class ReduceForJob2 extends MapReduceBase
				implements Reducer<Text, Text, Text, Text> {
		private Text value_out_text = new Text();
		public void reduce(Text key, Iterator<Text> values,
					OutputCollector<Text, Text> output, Reporter reporter) throws
IOException {
				String value_out = "";
				while (values.hasNext()) {
					value_out = value_out.concat(values.next().toString()).concat(" ");
				}
				value_out_text.set(value_out);
				output.collect(key, value_out_text);
		}
	}

	static int printUsage() {
		System.out.println("weather [-m <maps>] [-r <reduces>] <job_1 input> <job_1
output> <job_2 output>");
		ToolRunner.printGenericCommandUsage(System.out);
		return -1;
	}

	/**
	 * The main driver for weather map/reduce program.
	 * Invoke this method to submit the map/reduce job.
	 * @throws IOException When there is communication problems with the
	 *                 job tracker.
	 */
	public int run(String[] args) throws Exception {
		Configuration config = getConf();


		// We need to lower input block size by factor of two.
		/*config.setLong(
		    FileInputFormat.SPLIT_MAXSIZE,
		    config.getLong(
		        FileInputFormat.SPLIT_MAXSIZE, DEFAULT_SPLIT_SIZE) / 2);*/

		JobConf conf = new JobConf(config, Weather.class);
		conf.setJobName("Weather Job1");
```

```
// the keys are words (strings)
conf.setOutputKeyClass(Text.class);
// the values are counts (ints)
conf.setOutputValueClass(Text.class);

conf.setMapOutputKeyClass(Text.class);
conf.setMapOutputValueClass(Text.class);

conf.setMapperClass(MapClass.class);
//conf.setCombinerClass(Combiner.class);
conf.setReducerClass(Reduce.class);
List<String> other_args = new ArrayList<String>();
for(int i=0; i < args.length; ++i) {
        try {
                if ("-m".equals(args[i])) {
                        conf.setNumMapTasks(Integer.parseInt(args[++i]));
                } else if ("-r".equals(args[i])) {
                        conf.setNumReduceTasks(Integer.parseInt(args[++i]));
                } else {
                        other_args.add(args[i]);
                }
        } catch (NumberFormatException except) {
               System.out.println("ERROR: Integer expected instead of " + args[i]);
                return printUsage();
        } catch (ArrayIndexOutOfBoundsException except) {
                System.out.println("ERROR: Required parameter missing from " +
                                args[i-1]);
                return printUsage();
        }
}
// Make sure there are exactly 2 parameters left.
/*if (other_args.size() != 3) {
        System.out.println("ERROR: Wrong number of parameters: " +
                        other_args.size() + " instead of 3.");
        return printUsage();
}*/
FileInputFormat.setInputPaths(conf, other_args.get(0));
FileOutputFormat.setOutputPath(conf, new Path(other_args.get(1)));

JobClient.runJob(conf);

JobConf conf2 = new JobConf(config, Weather.class);
conf2.setJobName("Weather Job 2");

// the keys are words (strings)
conf2.setOutputKeyClass(Text.class);
// the values are counts (ints)
conf2.setOutputValueClass(Text.class);

conf2.setInputFormat(KeyValueTextInputFormat.class);

conf2.setMapOutputKeyClass(Text.class);
conf2.setMapOutputValueClass(Text.class);

conf2.setMapperClass(MapClassForJob2.class);
//conf.setCombinerClass(Combiner.class);
```

```
                conf2.setReducerClass(ReduceForJob2.class);


                FileInputFormat.setInputPaths(conf2, new Path(other_args.get(1)));
                FileOutputFormat.setOutputPath(conf2, new Path(other_args.get(2)));

                JobClient.runJob(conf2);
                return 0;
        }


        public static void main(String[] args) throws Exception {
                int res = ToolRunner.run(new Configuration(), new Weather(), args);
                System.exit(res);
        }

}
```



Browse Directory

/outputtemp

Show 25 entries

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	supergroup	0 B	Jun 07 22:54	1	128 MB	_SUCCESS
-rw-r--r--	hduser	supergroup	0 B	Jun 07 22:54	1	128 MB	part-r-00000

Showing 1 to 2 of 2 entries

Hadoop, 2017.