Experiment No 6 :Implementing Word Count using MapReduce

AIM: To implement a word count program using hadoop map reduce.

Procedure:
1.To implement word count using map reduce we first need to open eclipse > File >New >Java project >Name it as WordCount >then Finish

2.We then need to reference 2 libraries stored in the system ,we can do that by adding external jar files by Configuring build path
Those two jar files are stored in
1./usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.6.0-mr1-cdh5.13.0.jar
2. /usr/lib/hadoop/hadoop-common-2.6.0-cdh5.13.0.jar

3.After referencing the libraries we are now going to create three programs that activates the driver ,maps and reduces
We will create three programs as follows
File > New > Class > WCDriver,WCMapper,WCReducer>Finish

4.We will Now add code to the three classes as follows
WCDriver.java

```
// Importing libraries
import java.io.IOException;
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 WCDriver extends Configured implements Tool {

	public int run(String args[]) throws IOException
	{
		if (args.length < 2)
		{
			System.out.println("Please give valid inputs");
			return -1;
		}
		JobConf conf = new JobConf(WCDriver.class);
		FileInputFormat.setInputPaths(conf, new Path(args[0]));
		FileOutputFormat.setOutputPath(conf, new Path(args[1]));
		conf.setMapperClass(WCMapper.class);
		conf.setReducerClass(WCReducer.class);
		conf.setMapOutputKeyClass(Text.class);
		conf.setMapOutputValueClass(IntWritable.class);
		conf.setOutputKeyClass(Text.class);
		conf.setOutputValueClass(IntWritable.class);
		JobClient.runJob(conf);
		return 0;
	}

	// Main Method
	public static void main(String args[]) throws Exception
	{
		int exitCode = ToolRunner.run(new WCDriver(), args);
		System.out.println(exitCode);
	}
}
```

WDMapper.java

```
// Importing libraries
import java.io.IOException;
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 WCMapper extends MapReduceBase implements
Mapper<LongWritable,

Text, Text, IntWritable> {

	// Map function
	public void map(LongWritable key, Text value, OutputCollector<Text,
				IntWritable> output, Reporter rep) throws IOException
	{

		String line = value.toString();

		// Splitting the line on spaces
		for (String word : line.split(" "))
		{
			if (word.length() > 0)
			{
```

```
                                        output.collect(new Text(word), new IntWritable(1));
                              }
                    }
          }
}
```

WDReducer.java

```
// Importing libraries
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 WCReducer extends MapReduceBase implements Reducer<Text,
                                                                                IntWritable, Text,
IntWritable> {

          // Reduce function
          public void reduce(Text key, Iterator<IntWritable> value,
                                        OutputCollector<Text, IntWritable> output,
                                                                      Reporter rep) throws IOException
          {

                    int count = 0;

                    // Counting the frequency of each words
                    while (value.hasNext())
                    {
                              IntWritable i = value.next();
                              count += i.get();
                    }

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

5.we will now create a text file containing the sentence we want to run the word count on and then we will transfer the file to hdfs using command hadoop fs -put /user/cloudera/WCFIle.txt /user/cloudera

6.Now we will export the project as a jar file and then run it as follows

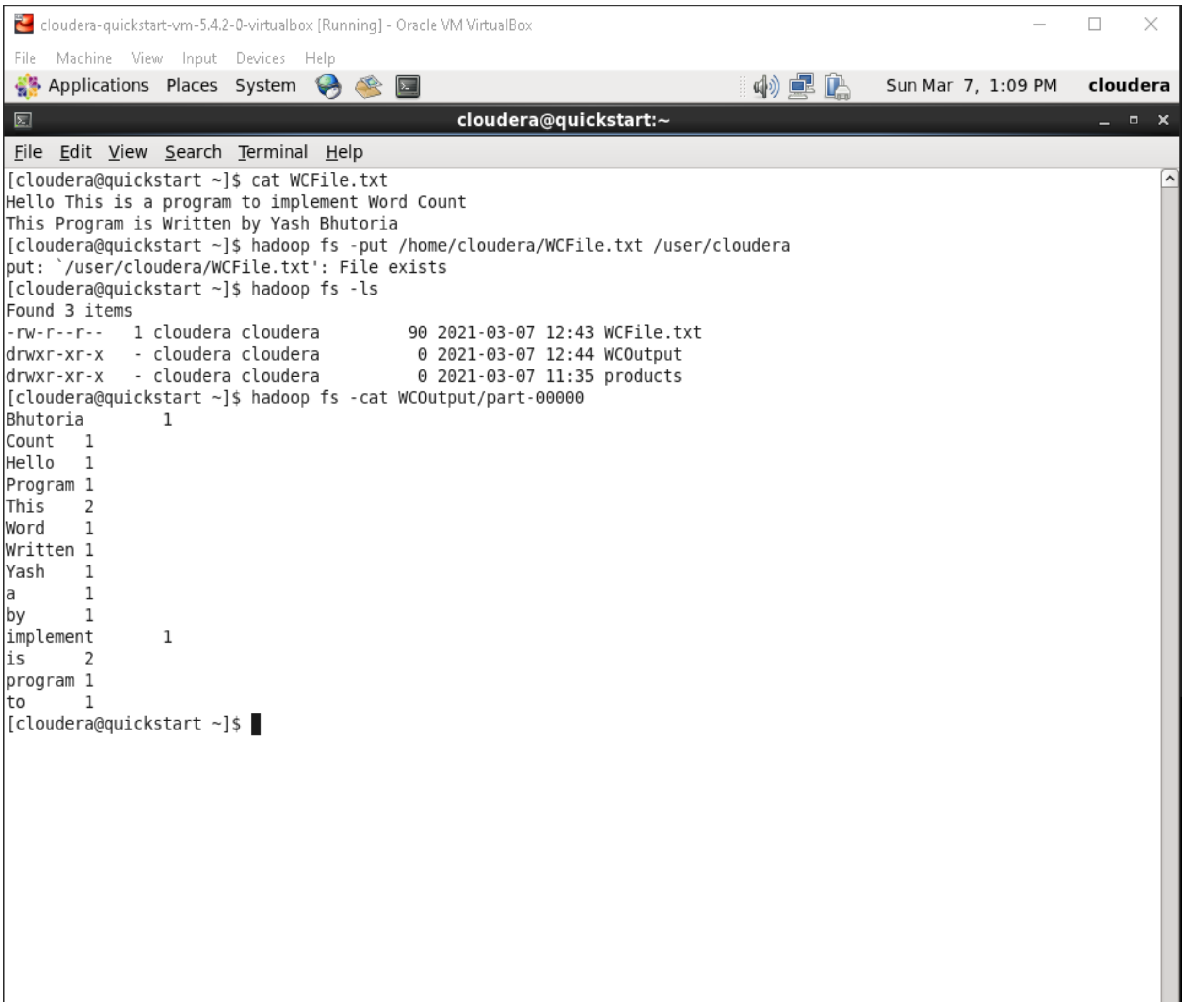
```
Hadoop jar WordCount.jar WCDriver WCFile.txt WCOutput
```

7.To check the output we will use the command

```
hadoop fs -cat WCOutput/part-00000
```

Result : You have successfully executed the word count program using map reduce .

Screenshot:


cloudera-quickstart-vm-5.4.2-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Sun Mar 7, 1:09 PM
cloudera
cloudera@quickstart:~
File Edit View Search Terminal Help

```
[cloudera@quickstart ~]$ cat WCFile.txt
Hello This is a program to implement Word Count
This Program is Written by Yash Bhutoria
[cloudera@quickstart ~]$ hadoop fs -put /home/cloudera/WCFile.txt /user/cloudera
put: `/user/cloudera/WCFile.txt': File exists
[cloudera@quickstart ~]$ hadoop fs -ls
Found 3 items
-rw-r--r--   1 cloudera cloudera         90 2021-03-07 12:43 WCFile.txt
drwxr-xr-x   - cloudera cloudera          0 2021-03-07 12:44 WCOutput
drwxr-xr-x   - cloudera cloudera          0 2021-03-07 11:35 products
[cloudera@quickstart ~]$ hadoop fs -cat WCOutput/part-00000
Bhutoria        1
Count   1
Hello   1
Program 1
This    2
Word    1
Written 1
Yash    1
a       1
by      1
implement       1
is      2
program 1
to      1
[cloudera@quickstart ~]$ 
```