Hadoop Word Count Assignment – Full Guide with Code and Commands

---

🖍️ Assignment Objective

Implement a Word Count program using the Hadoop MapReduce model on Cloudera VM using the older mapred API.

🔍 Step-by-Step Guide

🔧 Step 1: Create Working Directory (Optional)

mkdir ~/wordcount

cd ~/wordcount

📄 Step 2: Create Java Files

Create the following three files using gedit or any text editor.

1. WCMapper.java

```
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> {

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

```
        String line = value.toString();
        for (String word : line.split(" ")) {
            if (word.length() > 0) {
                output.collect(new Text(word), new IntWritable(1));
            }
        }
    }
}
```

2. WCReducer.java

```
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> {
    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter rep) throws IOException {
        int count = 0;
        while (values.hasNext()) {
            count += values.next().get();
        }
        output.collect(key, new IntWritable(count));
```

```
    }
}
```

3. WCDriver.java

```
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;
    }

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

⚖️ **Step 3: Compile Java Files**

javac -classpath `hadoop classpath` -d . WCMapper.java WCReducer.java WCDriver.java

---

🗓️ **Step 4: Create a JAR File**

jar cf wordcount.jar *.class

📂 **Step 5: Prepare Input Data**

Create and upload a sample file to HDFS:

echo "hadoop mapreduce hadoop hive spark" > input.txt

hadoop fs -mkdir -p /wordcount/input

hadoop fs -put input.txt /wordcount/input/

🏃 Step 6: Run the MapReduce Job

hadoop jar wordcount.jar WCDriver /wordcount/input /wordcount/output

---

📊 Step 7: View the Output

hadoop fs -cat /wordcount/output/part-00000

Expected Output:

hadoop 2

mapreduce 1

hive 1

spark 1

✅ Notes

- Make sure Hadoop daemons are running.
- To re-run, delete output directory first:

hadoop fs -rm -r /wordcount/output

End of Word Count Assignment - Code + Commands