HADOOP VERSION 3.3.5

Word Count DSBDA

WordCount.java

```
import org.apache.hadoop.conf.*;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.*;
import org.apache.hadoop.mapreduce.lib.output.*;

public class WordCount {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");

        job.setJarByClass(WordCount.class);
        job.setMapperClass(WordCountMapper.class);
        job.setCombinerClass(WordCountReducer.class);
        job.setReducerClass(WordCountReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

WordCountMapper.java

```
import java.io.IOException;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;

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

    @Override
    public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
        String line = value.toString();
        String[] words = line.split("\\s+"); // Splitting by whitespace
        for (String word : words) {
            this.word.set(word);
            context.write(this.word, one);
        }
    }
}
```

WordCountReduce.java

```
import java.io.IOException;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;

public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    private IntWritable result = new IntWritable();

    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {
        int sum = 0;
        for (IntWritable value : values) {
            sum += value.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}
```

Step 1: Start Hadoop using

```
1. start-dfs.sh
 2. start-yarn.sh
```

Step 2: Making Class files
syntax:
javac -classpath **path_to/*hadoop-common-3.3.5.jar***:**path_to/*hadoop-mapreduce-client-core-3.3.5.jar***
-d ***path_to/all .java files***

ex.

```
javac -classpath /home/hadoop/hadoop-3.3.5/share/hadoop/common/hadoop-common-
3.3.5.jar:/home/hadoop/hadoop-3.3.5/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.3.5.jar
-d /home/hadoop/Documents/ WordCount.java WordCountMapper.java WordCountReducer.java
```

Step 3: Maing .jar file using all classfiles

```
jar cvf WordCount.jar *.class
```

Note: *.class considers all class file from the current directory to combine and make jarfile.

Step 4: Copying your input file to hdfs

```
hadoop fs -copyFromLocal wordinput.txt  /user/hadoop/wordinput.txt
```

Note:
red colored file is yout file from local system. Blue colored path will be path in hdfs where file is stored.
If you get file already exsist just change name of file in blue path it will copy flocal file to hdfs with name changed
ex hadoop fs -copyFromLocal wordinput.txt /user/hadoop/wordinput.txt

Input.txt

```
Hello world
This is a sample input file
It contains multiple lines of text
Each line contains words
Some words may appear multiple times
This file is used for testing purposes
```

Step 5: Running the program

```
hadoop jar path_to..jar WordCount /user/hadoop/input.txt /user/hadoop/output
```

```
hadoop jar WordCount.jar WordCount /user/hadoop/wordinput.txt /user/hadoop/output
```

`hadoop jar` is used to run a Hadoop job using a jar file.

- `WordCount.jar` is the name of the jar file that contains the `WordCount` class.
- `WordCount` is the name of the class that contains the **main method** for the Hadoop job.
- `/user/hadoop/input.txt` is the path to the input file for the Hadoop job.
- `/user/hadoop/output` is the path to the output directory for the Hadoop job.

Fo output if run more than twice ake sure to change the name ex output1

Step 6 : Displaying the output

hadoop fs -cat /user/hadoop/wordoutput/part-r-00000

wordoutput: is the name of outfile u used in prev command