# Data Science & Big Data Analytics Laboratory

Assignment No. 10

**Hadoop MapReduce Framework**

Write a code in JAVA for a implement WordCount application that counts the number of occurrence of each word in a given input set using the Hadoop MapReduce framework on local standalone set up.

---

**Pre-requisite:**

o Java Installation: Check whether the Java is installed or not using the following command.

*java -version*

o Hadoop Installation: Check whether the Hadoop is installed or not using the following command.

*hadoop -version*

**Theory:**

Steps to execute MapReduce word count:

Create a text file in your local machine and write some text into it.

*$ nano data.txt*

Check the text written in the data.txt file.

*$ cat data.txt*

**Implementation:**

wordCound.java

```
import java.io.IOException; importjava.util.StringTokenizer;
importorg.apache.hadoop.conf.Configuration; importorg.apache.hadoop.fs.Path;
importorg.apache.hadoop.io.IntWritable; importorg.apache.hadoop.io.Text;
importorg.apache.hadoop.mapreduce.Job; importorg.apache.hadoop.mapreduce.Mapper;
importorg.apache.hadoop.mapreduce.Reducer;
importorg.apache.hadoop.mapreduce.lib.input.FileInputFormat;
importorg.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


 publicclassWordCount {
publicstaticclassTokenizerMapper
extends Mapper<Object,Text,Text,IntWritable>{
private finalstaticIntWritableone= newIntWritable(1); privateTextword= newText();

 publicvoidmap(Object key,Textvalue,Context context
) throwsIOException,InterruptedException { StringTokenizeritr=
newStringTokenizer(value.toString()); while(itr.hasMoreTokens()){
word.set(itr.nextToken()); context.write(word,one);
}
}
}
public staticclassIntSumReducer
```

```
extendsReducer<Text,IntWritable,Text,IntWritable>{ privateIntWritableresult=
newIntWritable();
public voidreduce(Textkey,Iterable<IntWritable> values, Contextcontext
) throws IOException, InterruptedException { int sum=0;
for (IntWritable val : values) { sum +=val.get();
}
result.set(sum);
result.set(sum); context.write(key, result);
}
}

publicstaticvoidmain(String[] args)throwsException { Configurationconf=
newConfiguration();
Job job = Job.getInstance(conf, "word count");
job.setJarByClass(WordCount.class); job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(IntSumReducer.class);
job.setReducerClass(IntSumReducer.class); job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job,newPath(args[0]));
FileOutputFormat.setOutputPath(job,newPath(args[1]));
System.exit(job.waitForCompletion(true)?0:1);
}
}
```

**Input:**

WordCount example reads text files and counts how often words occur. The input is text files, and the output is text files, each line of which contains a word and the count of how often it occurred, separated by a tab.

MapReduce Project that works on weather data and process it, the final outcome of the project can be processed further to find similarities on different weather stations.

input.txt

```
The Shadow was an American pulp magazine published by Street & Smith from 1931 to
1949. Each issue contained a novel about The Shadow, a mysterious crime-fighting
figure who spoke the line "Who knows what evil lurks in the hearts of men? The
Shadow knows "in radio broadcasts of stories from Street & Smith's Detective Story
Magazine. For the first issue, dated April 1931, Walter Gibson wrote the leadnovel,
```

**Output:**


**Word Count Steps to Run:**

1. Starting Hadoop

*$ start all.sh*

2. Make A folder "wordcountexp" and write WordCount.java code.

3. Create new folder for input data.

4. Add input text file in the input data folder.

5. Create new folder to hold java class files.
6. Set HADOOP_CLASSPATH environment variable.

*$ export HADOOP_CLASSPATH=$(hadoop classpath)*

7. Create a directory on HDFS

*$ hdfs dfs mkdir /WordCountTut*

*$ hdfs dfs mkdir /WordCountTut/Input*

8. Upload the input file ( to that directory.

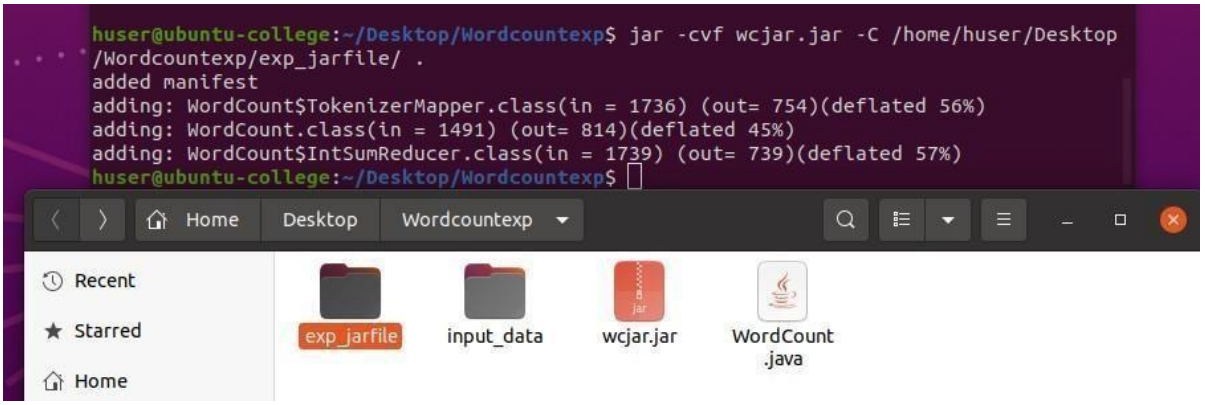
*$ hdfs dfs put '/home/huser/Desktop/Wordcountexp/input_data/input.txt' /WordCountTut/Input*

9. Compile the java code

*$ javac classpath $(HADOOP_ d '/home/huser/Desktop/Wordcountexp/ exp_jarfile' jarfile'/home/huser/Desktop/ Wordcountexp/.*java*

10. Creation of .jar file of classes:

*$ jar-cvfwcjar.jar-C'/home/huser/Desktop/Wordcountexp/exp_jarfile/.*



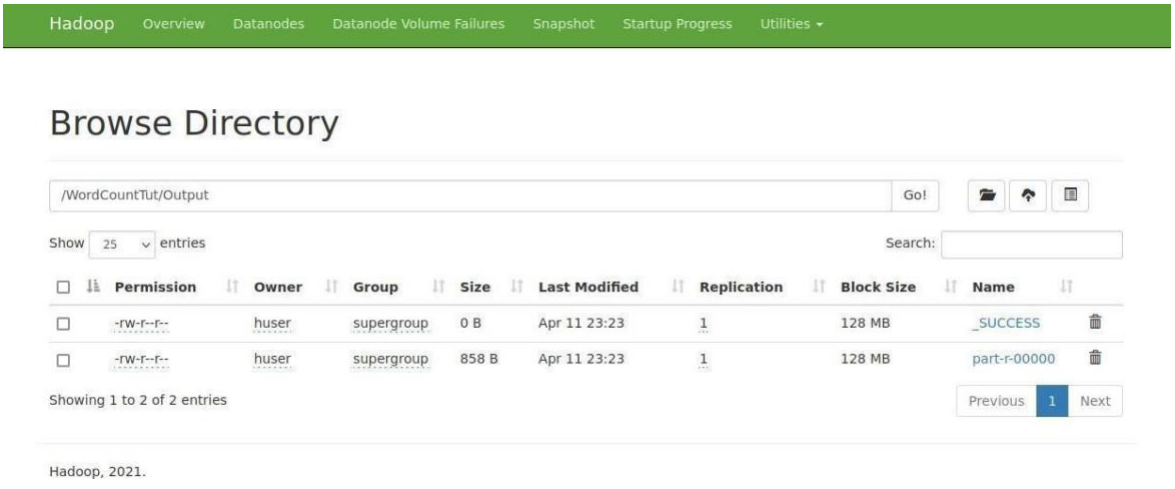

11. Running the jar file on Hadoop

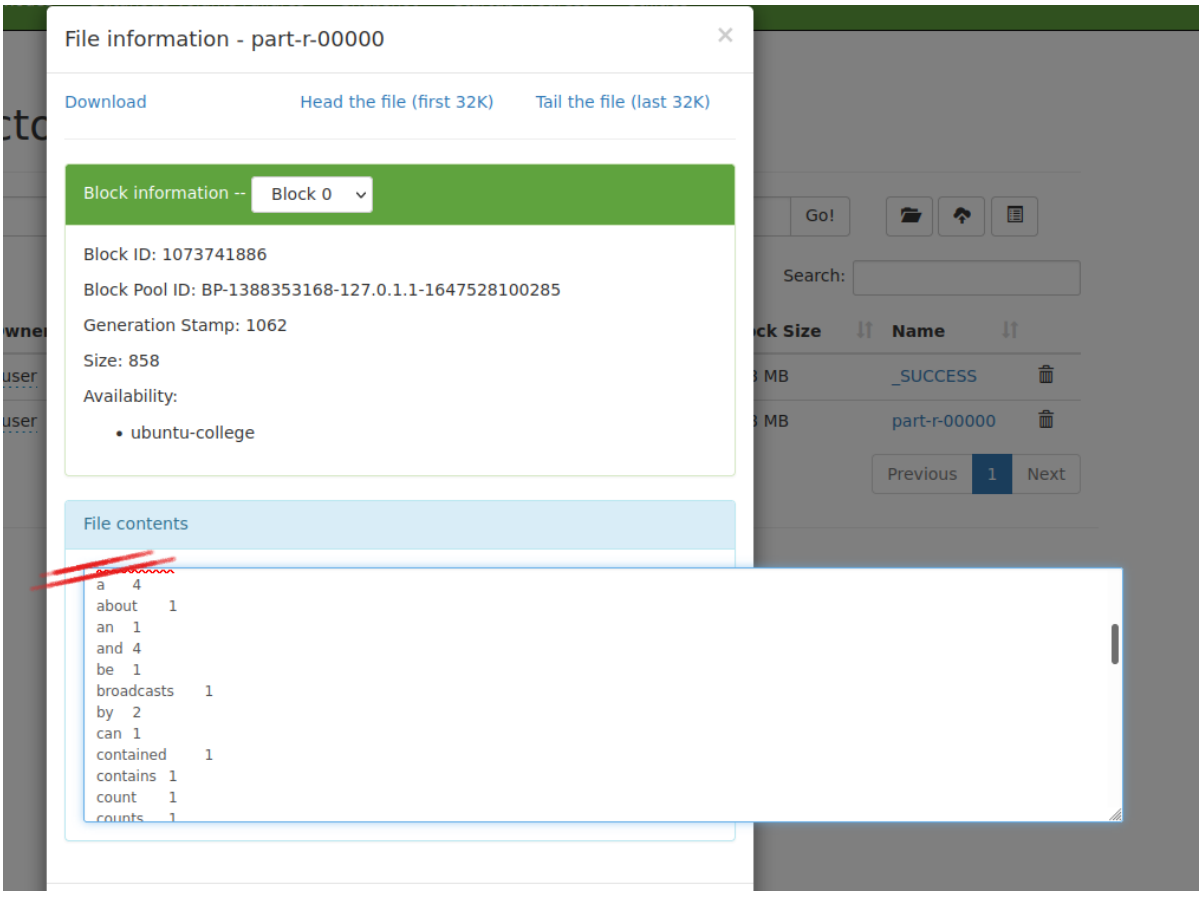
*$ hadoopjarwcjar.jarWordCount/WordCountTut/Input/WordCountTut/Output*

```
huser@ubuntu-college:~/Desktop/Wordcountexp$ hadoop jar wcjar.jar WordCount /WordCount
Tut/Input /WordCountTut/Output
2022-04-11 23:21:14,369 INFO client.RMProxy: Connecting to ResourceManager at /127.0.0
.1:8032
2022-04-11 23:21:17,291 WARN mapreduce.JobResourceUploader: Hadoop command-line option
 parsing not performed. Implement the Tool interface and execute your application with
 ToolRunner to remedy this.
2022-04-11 23:21:17,535 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding f
or path: /tmp/hadoop-yarn/staging/huser/.staging/job_1649697057322_0001
2022-04-11 23:21:18,170 INFO input.FileInputFormat: Total input files to process : 1
2022-04-11 23:21:18,286 INFO mapreduce.JobSubmitter: number of splits:1
2022-04-11 23:21:18,812 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_16
49697057322_0001
2022-04-11 23:21:18,813 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-04-11 23:21:19,296 INFO conf.Configuration: resource-types.xml not found
2022-04-11 23:21:19,296 INFO resource.ResourceUtils: Unable to find 'resource-types.xm
l'.
```

```
                Peak Reduce Physical memory (bytes)=167215104
                Peak Reduce Virtual memory (bytes)=2533072896
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=800
        File Output Format Counters
                Bytes Written=858
huser@ubuntu-college:~/Desktop/Wordcountexp$
```

12. Check the output on localhost:9870/ localhost:50070






**Conclusion:**

Thus, we successfully implemented, WordCount application that counts the number of occurrence of each word in a given input set using the Hadoop MapReduce framework on local standalone set up.