PRACTICAL-4

AIM:

To implement a word count application using the MapReduce API.

IMPLEMENTATION:

- Firstly, check whether Hadoop is installed or not.

```
hdoop@parth-virtual-machine:~$ hadoop version
Hadoop 3.2.4
Source code repository Unknown -r 7e5d9983b388e372fe640f21f048f2f2ae6e9eba
Compiled by ubuntu on 2022-07-12T11:58Z
Compiled with protoc 2.5.0
From source with checksum ee031c16fe785bbb35252c749418712
This command was run using /home/hdoop/hadoop-3.2.4/share/hadoop/common/hadoop-common-3.2.4.jar
```

- Then, make sure that java compiler is running correctly.

```
hdoop@parth-virtual-machine:~$ javac -version
javac 1.8.0_342
```

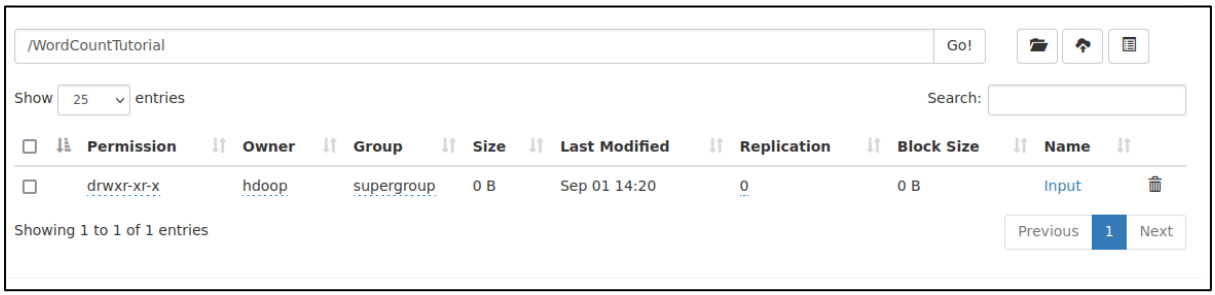
- Now, create a folder and a text file for the input.
- Also, create another folder to store java classes files.
- Now, set Hadoop classpath environment variable.

```
hdoop@parth-virtual-machine:~$ export HADOOP_CLASSPATH=$(hadoop classpath)
hdoop@parth-virtual-machine:~$ echo $HADOOP_CLASSPATH
/home/hdoop/hadoop-3.2.4/etc/hadoop:/home/hdoop/hadoop-3.2.4/share/hadoop/common/lib/*:/home/hdoop/hado
op-3.2.4/share/hadoop/common/*:/home/hdoop/hadoop-3.2.4/share/hadoop/hdfs:/home/hdoop/hadoop-3.2.4/shar
e/hadoop/hdfs/lib/*:/home/hdoop/hadoop-3.2.4/share/hadoop/hdfs/*:/home/hdoop/hadoop-3.2.4/share/hadoop/
mapreduce/lib/*:/home/hdoop/hadoop-3.2.4/share/hadoop/mapreduce/*:/home/hdoop/hadoop-3.2.4/share/hadoop
/yarn:/home/hdoop/hadoop-3.2.4/share/hadoop/yarn/lib/*:/home/hdoop/hadoop-3.2.4/share/hadoop/yarn/*
```

- Create a directory on HDFS.

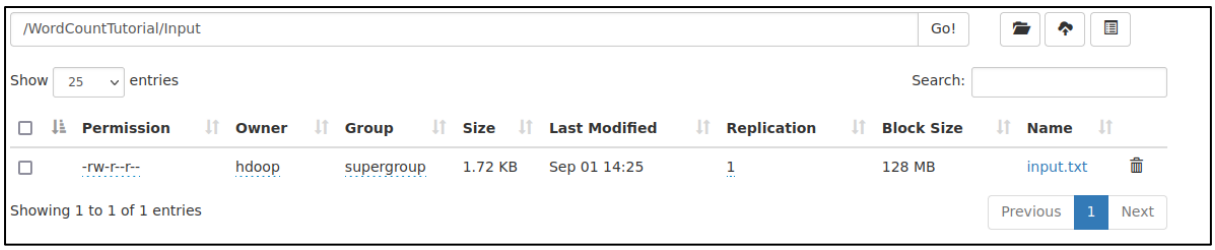
```
hdoop@parth-virtual-machine:~$ hadoop fs -mkdir /WordCountTutorial
hdoop@parth-virtual-machine:~$
```

- Create another directory in WordCountTutorial for the input.



- Upload the input file to that directory.

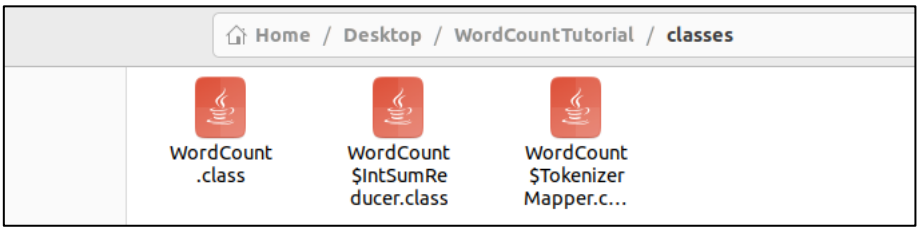
```
hdoop@parth-virtual-machine:~$ hadoop fs -put /home/hdoop/Desktop/WordCountTutorial/input_data/input.txt /WordCountTutorial/Input
hdoop@parth-virtual-machine:~$
```



- Change the directory to the one where all the files are located.
- Then, compile the java code.

```
hdoop@parth-virtual-machine:~/Desktop/WordCountTutorial$ javac -classpath ${HADOOP_CLASSPATH} -d /home/hdoop/Desktop/WordCountTutorial/cla
sses /home/hdoop/Desktop/WordCountTutorial/WordCount.java
hdoop@parth-virtual-machine:~/Desktop/WordCountTutorial$
```

- Class files are generated in the classes folder.

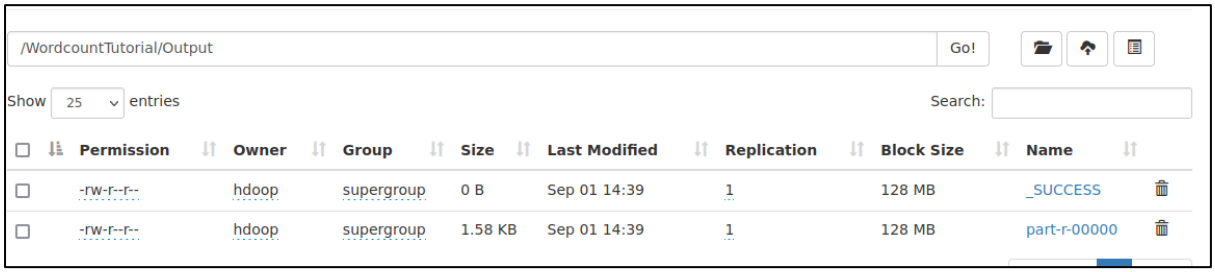


- Put the output files in one jar files.

```
hdoop@parth-virtual-machine:~/Desktop/WordCountTutorial$ jar -cvf wordCount.jar -C classes/ .
added manifest
adding: WordCount$TokenizerMapper.class(in = 1736) (out= 754)(deflated 56%)
adding: WordCount$IntSumReducer.class(in = 1739) (out= 739)(deflated 57%)
adding: WordCount.class(in = 1491) (out= 814)(deflated 45%)
```

- Run the jar file on Hadoop

```
hdoop@parth-virtual-machine:~/Desktop/WordCountTutorial$ hadoop jar /home/hdoop/Desktop/WordCountTutorial/wordCount.jar WordCount /WordCou
ntTutorial/Input /WordcountTutorial/Output
2022-09-01 14:38:17,040 INFO client.RMProxy: Connecting to ResourceManager at /127.0.0.1:8032
2022-09-01 14:38:18,457 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface
 and execute your application with ToolRunner to remedy this.
2022-09-01 14:38:18,484 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hdoop/.staging/job
_1662022089534_0001
2022-09-01 14:38:19,146 INFO input.FileInputFormat: Total input files to process : 1
2022-09-01 14:38:19,375 INFO mapreduce.JobSubmitter: number of splits:1
2022-09-01 14:38:20,088 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1662022089534_0001
2022-09-01 14:38:20,090 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-09-01 14:38:21,222 INFO conf.Configuration: resource-types.xml not found
2022-09-01 14:38:21,223 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-09-01 14:38:24,085 INFO impl.YarnClientImpl: Submitted application application_1662022089534_0001
2022-09-01 14:38:24,169 INFO mapreduce.Job: The url to track the job: http://parth-virtual-machine:8088/proxy/application_1662022089534_00
01/
2022-09-01 14:38:24,171 INFO mapreduce.Job: Running job: job_1662022089534_0001
2022-09-01 14:38:47,590 INFO mapreduce.Job: Job job_1662022089534_0001 running in uber mode : false
2022-09-01 14:38:47,591 INFO mapreduce.Job:  map 0% reduce 0%
```



- Check the output

```
hadoop@parth-virtual-machine:~/Desktop/WordCountTutorial$ hadoop dfs -cat /WordcountTutorial/Output/*
```

PROGRAM CODE:

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
```

```
      word.set(itr.nextToken());
      context.write(word, one);
    }
  }
}

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

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

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(TokenizerMapper.class);
  job.setCombinerClass(IntSumReducer.class);
  job.setReducerClass(IntSumReducer.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);
  }
}
```

OUTPUT:

```
A        1
And      1
He       2
If       1
It       1
It's     1
John     1
No       1
Nobody   1
Only     1
Suitcase         1
That     3
The      2
This     1
Why?Dave         1
a        9
about    1
across   1
actually         1
air,     1
all      2
all,     1
always   1
an       2
and      10
```

CONCLUSION:

In this practical, I learnt to perform wordcount using java, Hadoop and MapReduce Technique.