PRACTICAL NO.-4

Aim: Run a java program based on parallel programming to implement the concept of Map Reduce Paradigm.

DESCRIPTION:-

The **WordCount** program is the classic example for learning **Hadoop MapReduce**. It counts the frequency of each word in a given input file. This practical demonstrates how to:

1. Prepare input data in **HDFS**.
2. Write **Mapper, Reducer, and Driver** classes in Java.
3. Compile and create a **JAR file**.
4. Run the MapReduce job on Hadoop through **PuTTY**.
5. View results stored in HDFS.

This experiment helps understand the **basic flow of MapReduce** in Hadoop.

STEPS OF EXECUTION: -

Step 1: Prepare Input Data

```
echo "Hadoop is big data Hadoop is Java" > sample.txt
hdfs dfs -mkdir /input
hdfs dfs -put sample.txt /input/
hdfs dfs -ls /input
```

Step 2: Create Java Files

WordMapper.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.mapreduce.Mapper;
public class WordMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();
   public void map(LongWritable key, Text value, Context context)
         throws IOException, InterruptedException {
      String line = value.toString();
      for (String token : line.split("\\s+")) {
         word.set(token);
         context.write(word, one);
      }
   }
```

```
}
```

WordReducer.java

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

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

WordCountDriver.java

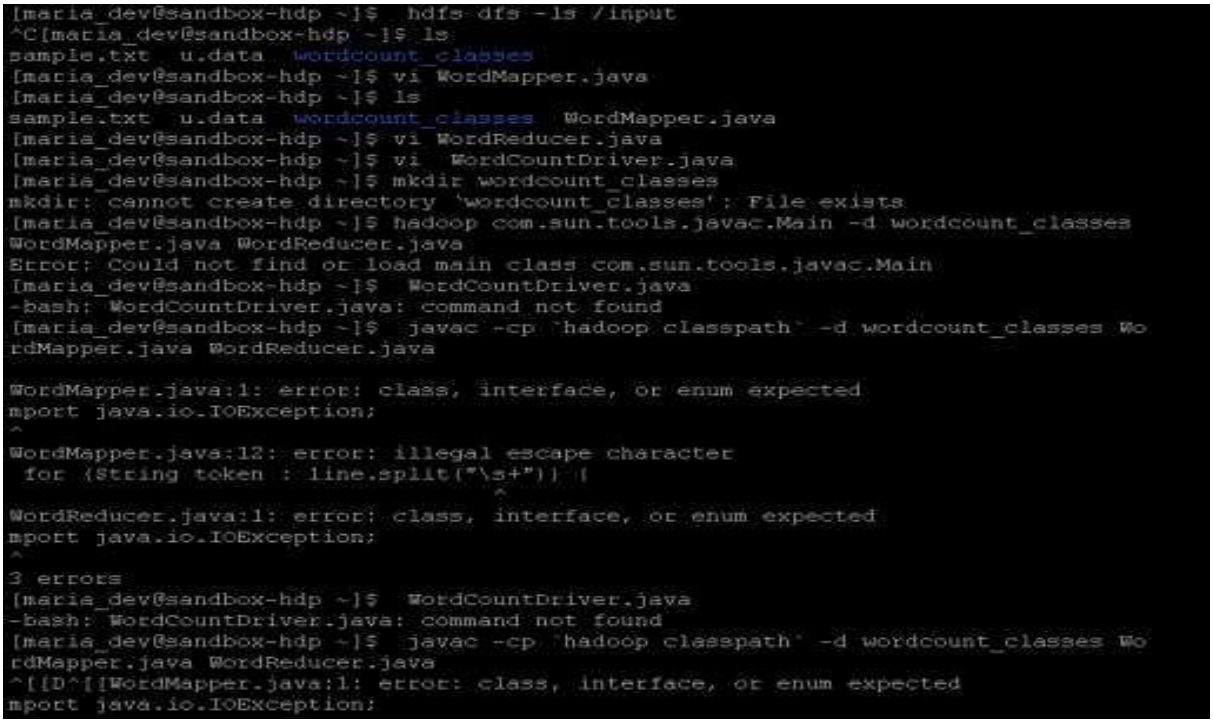
```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

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

      job.setJarByClass(WordCountDriver.class);
      job.setMapperClass(WordMapper.class);
      job.setReducerClass(WordReducer.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);
   }
}
```

Step 3: Compile Java Files
mkdir wordcount_classes
hadoop com.sun.tools.javac.Main -d wordcount_classes WordMapper.java WordReducer.java WordCountDriver.java
👉 If error occurs, use:
javac -cp `hadoop classpath` -d wordcount_classes WordMapper.java WordReducer.java WordCountDriver.java

Step 4: Create JAR File
jar -cvf wordcount.jar -C wordcount_classes/ .

Step 5: Run MapReduce Job
hdfs dfs -rm -r /output
hadoop jar wordcount.jar WordCountDriver /input /output

Step 6: View Output
hdfs dfs -ls /output
hdfs dfs -cat /output/part-r-00000
☑ **Expected Output Example:**
Hadoop 2
Java 1
big 1
data 1
is 2



```
3 errors
[maria_dev@sandbox-hdp ~]$  WordCountDriver.java
-bash: WordCountDriver.java: command not found
[maria_dev@sandbox-hdp ~]$  javac -cp `hadoop classpath` -d wordcount_classes Wo
rdMapper.java WordReducer.java
^[[D^[[WordMapper.java:1: error: class, interface, or enum expected
mport java.io.IOException;
^
WordMapper.java:12: error: illegal escape character
 for (String token : line.split("\s+")) {
                              ^
WordReducer.java:1: error: class, interface, or enum expected
mport java.io.IOException;
^
3 errors                                                                        |
rdMapper.java WordReducer.javaac -cp `hadoop classpath` -d wordcount_classes Wo
 WordCountDriver.java

WordMapper.java:1: error: class, interface, or enum expected
mport java.io.IOException;
^
WordMapper.java:12: error: illegal escape character
 for (String token : line.split("\s+")) {
                              ^
WordReducer.java:1: error: class, interface, or enum expected
mport java.io.IOException;
^
3 errors
[maria_dev@sandbox-hdp ~]$  WordCountDrivet.java
-bash: WordCountDriver.java: command not found
[maria_dev@sandbox-hdp ~]$
[maria_dev@sandbox-hdp ~]$ vi WordReducer.java
[maria_dev@sandbox-hdp ~]$ vi  WordCountDriver.java
[maria_dev@sandbox-hdp ~]$ vi WordMapper.java
[maria_dev@sandbox-hdp ~]$  javac -cp `hadoop classpath` -d wordcount_classes WordMapper.java WordReducer.java
WordMapper.java:12: error: illegal escape character
 for (String token : line.split("\s+")) {
                              ^
1 error
[maria_dev@sandbox-hdp ~]$ vi WordMapper.java
[maria_dev@sandbox-hdp ~]$ ^C
[maria_dev@sandbox-hdp ~]$ vi WordMapper.java
[maria_dev@sandbox-hdp ~]$ vi WordMapper.java
[maria_dev@sandbox-hdp ~]$  javac -cp `hadoop classpath` -d wordcount_classes WordMapper.java WordReducer.java
WordReducer.java:9: error: incompatible types: Object cannot be converted to IntWritable
 for (IntWritable val : values) {
                        ^
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
1 error
[maria_dev@sandbox-hdp ~]$ javac -cp `hadoop classpath` -d wordcount_classes WordMapper.java WordReducer.java WordCountDriver.java
WordReducer.java:9: error: incompatible types: Object cannot be converted to IntWritable
 for (IntWritable val : values) {
                        ^
Note: Some input files use unchecked or unsafe operations.
```

```
Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
1 error
[maria_dev@sandbox-hdp ~]$ ^C
[maria_dev@sandbox-hdp ~]$ vi WordReducer.java
[maria_dev@sandbox-hdp ~]$ javac -cp `hadoop classpath` -d wordcount_classes WordMapper.java WordReducer.java WordCountDriver.java
Note: WordMapper.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
[maria_dev@sandbox-hdp ~]$ ^C
[maria_dev@sandbox-hdp ~]$  jar -cvf wordcount.jar -C wordcount_classes/ .
added manifest
adding: WordMapper.class(in = 1275) (out= 690)(deflated 46%)
adding: WordReducer.class(in = 1592) (out= 663)(deflated 58%)
adding: WordCountDriver.class(in = 1364) (out= 752)(deflated 44%)
[maria_dev@sandbox-hdp ~]$
[maria_dev@sandbox-hdp ~]$  hdfs dfs -rm -r /output

rm: `/output': No such file or directory
[maria_dev@sandbox-hdp ~]$
[maria_dev@sandbox-hdp ~]$ ^C
[maria_dev@sandbox-hdp ~]$ hadoop jar wordcount.jar WordCountDriver /input /output

25/09/01 04:57:03 INFO client.RMProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
25/09/01 04:57:04 INFO client.AHSProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
```

```
File System Counters
        FILE: Number of bytes read=82
        FILE: Number of bytes written=305979
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=155
        HDFS: Number of bytes written=34
        HDFS: Number of read operations=6
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
```

CONCLUSION: -

In this practical, we implemented and executed the **Hadoop WordCount program** on a singlenode Hadoop cluster using PuTTY. We successfully:

- Uploaded input data into HDFS,
- Compiled Java Mapper, Reducer, and Driver classes,
- Packaged them into a JAR,
- Executed the MapReduce job, and
- Verified the word frequency results from HDFS.

This experiment demonstrates the **basic working of MapReduce** in Hadoop.