Ex No – 11	COMPILING AND RUNNING A WORD COUNT PROGRAM

Aim:

To write, compile and execute a Java Program for word count in a Hadoop Single Node Cluster using map reduce concept.

Procedure:

1. Login With hduser in the ubuntu machine which has the hadoop single node cluster setup.
2. Navigate into the Hadoop Home Directory.
3. Check if the hadoop daemons are running properly.
4. If Hadoop daemons are not running, start them all.

```
cloudlab@cloudlab-OptiPlex-9020:~$ su hduser
Password:
hduser@cloudlab-OptiPlex-9020:/home/cloudlab$ cd $HADOOP_HOME
hduser@cloudlab-OptiPlex-9020:/usr/local/hadoop$ jps
31061 Jps
hduser@cloudlab-OptiPlex-9020:/usr/local/hadoop$ start-dfs.sh
hduser@cloudlab-OptiPlex-9020:/usr/local/hadoop$ start-yarn.sh
hduser@cloudlab-OptiPlex-9020:/usr/local/hadoop$ jps
31468 DataNode
31657 SecondaryNameNode
31212 NameNode
32140 ResourceManager
32515 NodeManager
32747 Jps
```

Compiling Map Reduce Java Program and Creating the JAR file for WordCount Operation :

5. Make a new directory and navigate into the directory.
6. Create a new java file “WordCount.java”

```
hduser@cloudlab-OptiPlex-9020:/usr/local/hadoop$ mkdir mywordcount
hduser@cloudlab-OptiPlex-9020:/usr/local/hadoop$ cd mywordcount/ hduser@cloudlab-
OptiPlex-9020:/usr/local/hadoop/mywordcount$ nano WordCount.java
```

7. Type the following Map Reduce program in the file “WordCount.java”.
8. After finished typing, Save and Quit the file.

```
//Java program for word count operation
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);
  }
}
//Press ctrl + o and enter to save and ctrl + x to quit the file.
```

9. Navigate back to the Hadoop Home Directory.
10. Compile "WordCount.java"

```
hduser@clo...9020:/usr/local/hadoop/mywordcount$ cd ..
hduser@...hadoop$ bin/hadoop com.sun.tools.javac.Main mywordcount/WordCount.java
```

11. Again Navigate into the directory in which "WordCount.java" is located.
12. List the files in the directory. Now, the directory will contain the compiled Map Reduce class files of "WordCount.java"
13. Create the jar file "wc.jar" with all the compiled classes of "WordCount.java"
14. Again list the files in the directory, to check if the newly created "wc.jar" is located.

```
hduser@cloudlab-OptiPlex-9020:/usr/local/hadoop$ cd mywordcount/
hduser@clo...9020:/usr/local/hadoop/mywordcount$ ls
WordCount$IntSumReducer.class WordCount$TokenizerMapper.class
WordCount.class WordCount.java
hduser@clo....9020:/usr/local/hadoop/mywordcount$ jar cf wc.jar WordCount*.class
hduser@clo...9020:/usr/local/hadoop/mywordcount$ ls
wc.jar       WordCount$IntSumReducer.class WordCount$TokenizerMapper.class
WordCount.class WordCount.java
hduser@cloudlab-OptiPlex-9020:/usr/local/hadoop/mywordcount$ cd ..
```

<u>Running the Word Count Program in Hadoop :</u>

15. Create a text file "sample.txt" and type some sample contents for Word Count Processing.
16. After finished typing, Save and Quit the file.
17. Navigate back to the Hadoop Home Directory.

```
hduser@cloudlab-OptiPlex-9020:/usr/local/hadoop/mywordcount$ nano
sample.txt //Sample Contents for Word Count Processing
This is a sample text file that is going to be used to test
the word count program. It is important that this
program work correctly, since it will be used on the
exam. I need to know that all is fine with the code I am
providing. The program keeps track of all the words in
this file and how many times each has occurred.
//Press ctrl + o and enter to save and ctrl + x to quit the file.
```

```
hduser@cloudlab-OptiPlex-9020:/usr/local/hadoop/mywordcount$ cd ..
hduser@cloudlab-OptiPlex-9020:/usr/local/hadoop$
```

18. Create a new directory in the hdfs.
19. List the files in hdfs, to check if the directory is created.
20. Now, move "sample.txt" located in our local filesystem to the newly created directory in hdfs.

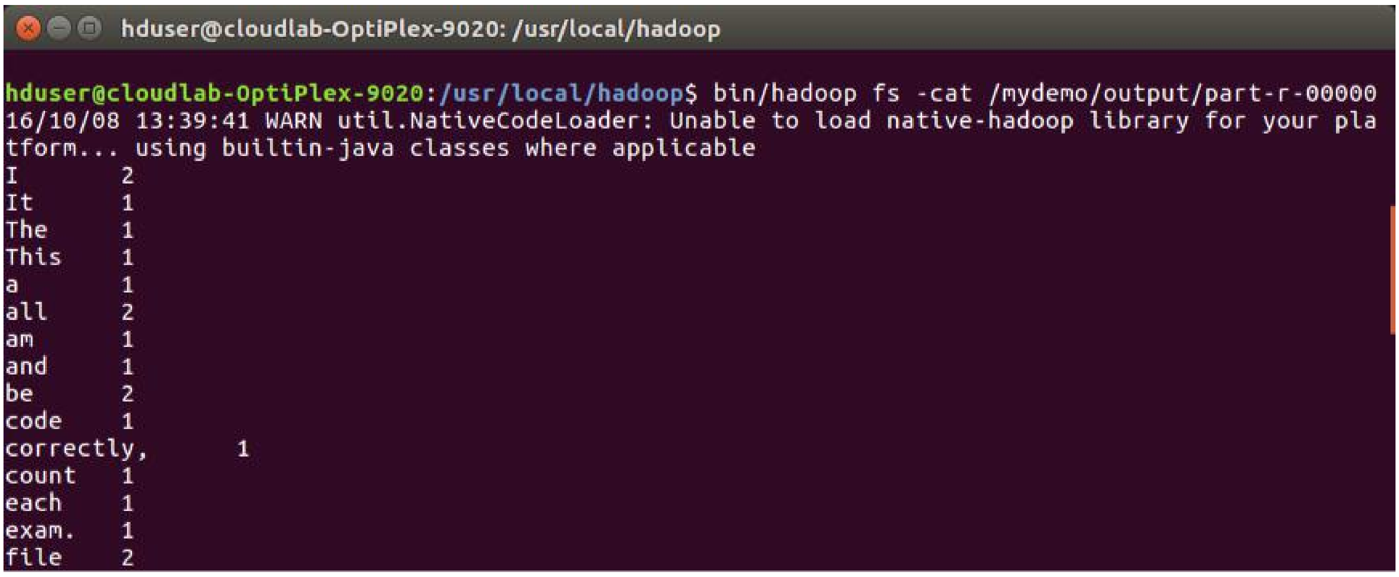
```
hduser@cloudlab-OptiPlex-9020:/usr/local/hadoop$ bin/hadoop fs -mkdir
/mydemo hduser@cloudlab-OptiPlex-9020:/usr/local/hadoop$ bin/hadoop fs -ls /
Found 2 items
drwxr-xr-x  - hduser supergroup    0 2016-09-30 16:06 /demo
drwxr-xr-x  - hduser supergroup    0 2016-10-08 10:57 /mydemo
hduser@clo....../hadoop$ bin/hdfs dfs -moveFromLocal mywordcount/sample.txt /mydemo
```

21. Now, execute the below hadoop command for processing "sample.txt" using "wc.jar".

```
bin/hadoop jar mywordcount/wc.jar WordCount /mydemo/sample.txt /mydemo/output
```

22. In hdfs, a new directory named "output" will be created in the location /mydemo and the output file "part-r-00000" will be located in it, which contains the output of the Word Count operation.
23. Display the output using hadoop -cat command.
24. Stop.

OUTPUT:



hduser@cloudlab-OptiPlex-9020: /usr/local/hadoop

```
hduser@cloudlab-OptiPlex-9020:/usr/local/hadoop$ bin/hadoop fs -cat /mydemo/output/part-r-00000
16/10/08 13:39:41 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your pla
tform... using builtin-java classes where applicable
I       2
It      1
The     1
This    1
a       1
all     2
am      1
and     1
be      2
code    1
correctly,      1
count   1
each    1
exam.   1
file    2
```

Result:

Thus the Java Program for performing word count operation using map reduce concept have been created, compiled and executed successfully..