

Exp No: 09

Date: 13.09.2024

HADOOP

DEMONSTRATE THE MAP REDUCE PROGRAMMING MODEL BY COUNTING THE NUMBER OF WORDS IN A FILE

AIM:

To demonstrate the MAP REDUCE programming model for counting the number of words in a file.

PROCEDURE

Step 1 - Open Terminal

```
$ su hduser
```

Password:

Step 2 - Start dfs and mapreduce services

```
$ cd /usr/local/hadoop/hadoop-2.7.2/sbin
```

```
$ start-dfs.sh
```

```
$ start-yarn.sh
```

```
$ jps
```

Step 3 - Check Hadoop through web UI

// Go to browser type <http://localhost:8088> – All Applications Hadoop Cluster

// Go to browser type <http://localhost:50070> – Hadoop Namenode

Step 4 – Open New Terminal

```
$ cd Desktop/
```

```
$ mkdir inputdata
```

```
$ cd inputdata/
```

```
$ echo "Hai, Hello, How are you? How is your health?" >> hello.txt
```

```
$ cat>> hello.txt
```

Step 5 – Go back to old Terminal

```
$ hadoop fs -copyFromLocal /home/hduser/Desktop/inputdata/hello.txt  
/folder/hduser // Check in hello.txt in Namenode using Web UI
```

Step 6 – Download and open eclipse by creating workspace

Create a new java project.

Step 7 – Add jar to the project

You need to remove dependencies by adding jar files in the hadoop source folder. Now Click on Project tab and go to Properties. Under Libraries tab, click Add External JARs and select all the jars in the folder (click on 1st jar, and Press Shift and Click on last jar to select all jars in between and click ok)

```
/usr/local/hadoop/hadoop-2.7.2/share/hadoop/commonand
```

```
/usr/local/hadoop/hadoop-2.7.2/share/hadoop/mapreduce folders.
```

Step 8 – WordCount Program

Create 3 java files named

- WordCount.java
- WordCountMapper.java
- WordCountReducer.java

WordCount.java

```
import org.apache.hadoop.conf.Configured;  
  
import org.apache.hadoop.fs.Path;  
  
import org.apache.hadoop.io.IntWritable;  
  
import org.apache.hadoop.mapred.FileInputFormat;
```

```
import org.apache.hadoop.mapred.FileOutputFormat;

import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

import org.apache.hadoop.io.Text;

public class WordCount extends Configured implements Tool {

    @Override

    public int run(String[] arg0) throws Exception {

        // TODO Auto-generated method

        stub

        if(arg0.length<2)

        {

            System.out.println("check the command line arguments");

        }

        JobConf conf=new JobConf(WordCount.class);

        FileInputFormat.setInputPaths(conf, new Path(arg0[0]));

        FileOutputFormat.setOutputPath(conf, new Path(arg0[1]));

        conf.setMapperClass(WordMapper.class);

        conf.setReducerClass(WordReducer.class);

        conf.setOutputKeyClass(Text.class);

        conf.setOutputValueClass(IntWritable.class);

        conf.setOutputKeyClass(Text.class);
```

```
        conf.setOutputValueClass(IntWritable.class);

        JobClient.runJob(conf);

        return 0;

    }

    public static void main(String args[]) throws Exception

    {

        int exitcode=ToolRunner.run(new WordCount(),

        args); System.exit(exitcode);

    }

}
```

WordCountMapper.java

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.Mapper;

public class WordCountMapper extends MapReduceBase implements
```

```

Mapper<LongWritable,Text,Text,IntWritable>

{

    @Override

    public void map(LongWritable arg0, Text arg1, OutputCollector<Text,
IntWritable> arg2, Reporter arg3)

        throws IOException {

        // TODO Auto-generated method stub

        String s=arg1.toString();

        for(String word:s.split(" "))

        {

            arg2.collect(new Text(word),new IntWritable(1));

        }

    }

}

```

WordCountReducer.java

```

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

import org.apache.hadoop.io.Text;

```

```

public class WordCountReducer implements
    Reducer<Text,IntWritable,Text,IntWritable> { @Override

    public void configure(JobConf arg0) {

        // TODO Auto-generated method stub

    }

    @Override

    public void close() throws IOException {

        // TODO Auto-generated method stub

    }

    @Override
    public void reduce(Text arg0, Iterator<IntWritable> arg1,
        OutputCollector<Text, IntWritable> arg2, Reporter arg3)

        throws IOException {

        // TODO Auto-generated method

        stub int count=0;

        while(arg1.hasNext())

        {

            IntWritable i=arg1.next();

            count+=i.get();

        }

        arg2.collect(arg0,new IntWritable(count));

    }

}

```

Step 9 - Create JAR file

Now Click on the Run tab and click Run-Configurations. Click on New Configuration button on the left top side and Apply after filling the following properties.

Step 10 - Export JAR file

Now click on File tab and select Export. under Java, select Runnable Jar.

In Launch Config – select the config file you created in Step 9 (WordCountConfig).

➤ Select an export destination (let's say desktop.)

➤ Under Library handling, select Extract Required Libraries into generated JAR and click Finish. ➤ Right-Click the jar file, go to Properties and under Permissions tab, Check Allow executing file as a program. and give Read and Write access to all the users

Step 11 – Go back to old Terminal for Execution of WordCount Program \$hadoop jar wordcount.jar/usr/local/hadoop/input/usr/local/hadoop/output

Step 12 – To view results in old Terminal

\$hdfs dfs -cat /usr/local/hadoop/output/part-r-00000

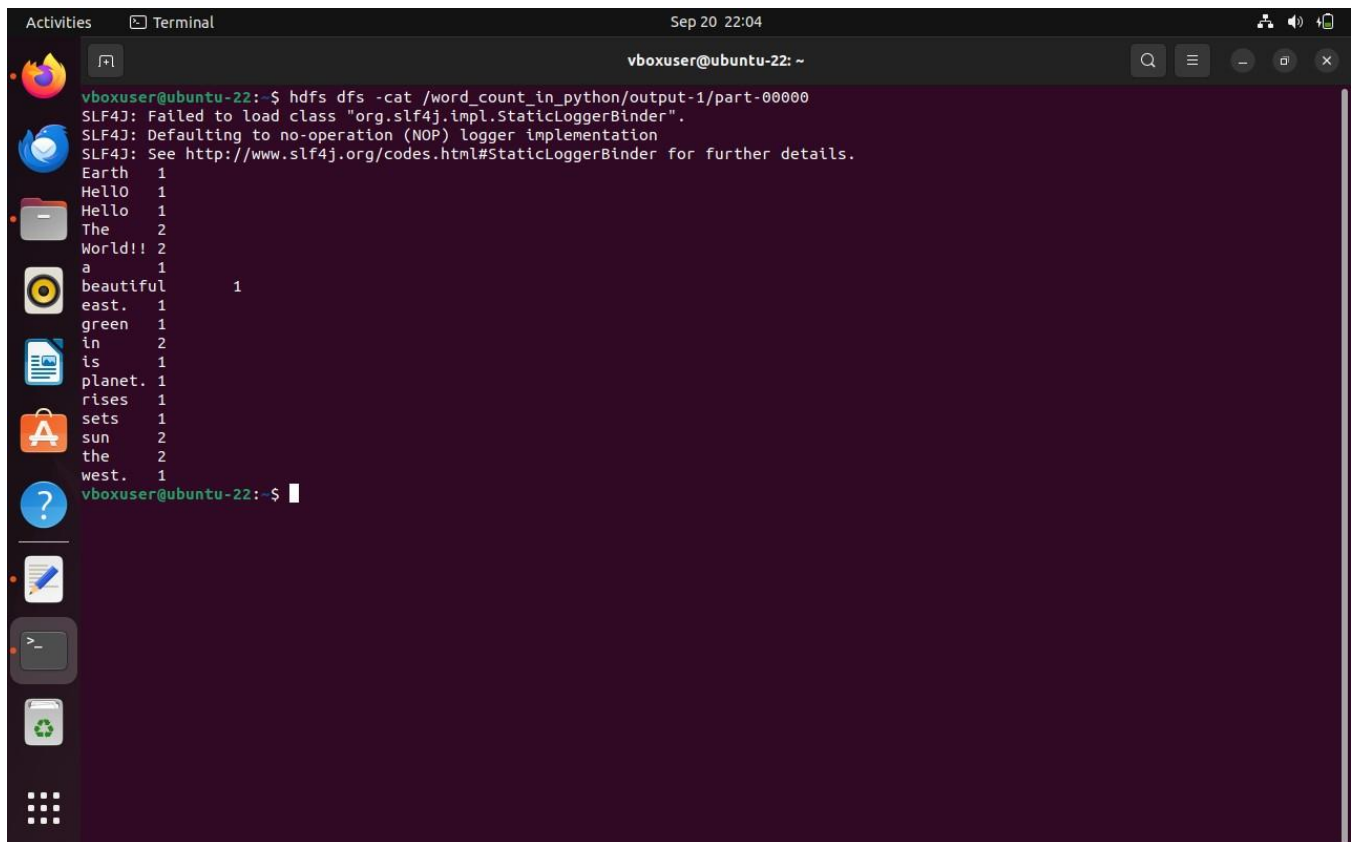
Step 13 - To Remove folders created using hdfs

\$ hdfs dfs -rm -R /usr/local/hadoop/output

OUTPUT:



```
GNU nano 6.2 word_count.txt
Hello World!!
The sun rises in the east.
The sun sets in the west.
Earth is a beautiful green planet.
Hello World!!
```

A terminal window titled 'Terminal' with a dark background. The prompt is 'vboxuser@ubuntu-22: ~'. The command 'hdfs dfs -cat /word_count_in_python/output-1/part-00000' has been executed. The output shows a list of words and their counts: Earth 1, Hello 1, Hello 1, The 2, World!! 2, a 1, beautiful 1, east. 1, green 1, in 2, is 1, planet. 1, rises 1, sets 1, sun 2, the 2, west. 1. There are also SLF4J warning messages at the top of the output.

```
vboxuser@ubuntu-22:~$ hdfs dfs -cat /word_count_in_python/output-1/part-00000
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Earth 1
Hello 1
Hello 1
The 2
World!! 2
a 1
beautiful 1
east. 1
green 1
in 2
is 1
planet. 1
rises 1
sets 1
sun 2
the 2
west. 1
vboxuser@ubuntu-22:~$
```

RESULT:

Thus the MAP REDUCE programming model for counting the number of words in a file is demonstrated.