

**Exp No: 10**

**Date:**

**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

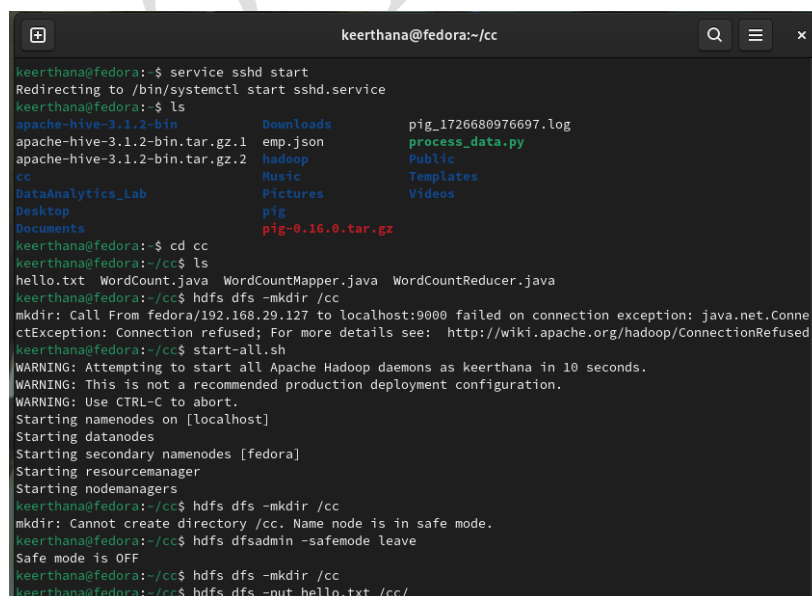
Step 12 – To view results in old Terminal

\$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:**



```
keerthana@fedora:~/cc
keerthana@fedora:~$ service sshd start
Redirecting to /bin/systemctl start sshd.service
keerthana@fedora:~$ ls
apache-hive-3.1.2-bin      Downloads      pig_1726680976697.log
apache-hive-3.1.2-bin.tar.gz.1  emp.json      process_data.py
apache-hive-3.1.2-bin.tar.gz.2  hadoop        Public
cc                           Music         Templates
DataAnalytics_Lab          Pictures      Videos
Desktop                    pig
Documents                  pig-0.16.0.tar.gz
keerthana@fedora:~$ cd cc
keerthana@fedora:~/cc$ ls
hello.txt WordCount.java WordCountMapper.java WordCountReducer.java
keerthana@fedora:~/cc$ hdfs dfs -mkdir /cc
mkdir: Call From fedora/192.168.29.127 to localhost:9000 failed on connection exception: java.net.ConnectionException: Connection refused; For more details see: http://wiki.apache.org/hadoop/ConnectionRefused
keerthana@fedora:~/cc$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as keerthana in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [fedora]
Starting resourcemanager
Starting nodemanagers
keerthana@fedora:~/cc$ hdfs dfs -mkdir /cc
mkdir: Cannot create directory /cc. Name node is in safe mode.
keerthana@fedora:~/cc$ hdfs dfsadmin -safemode leave
Safe mode is OFF
keerthana@fedora:~/cc$ hdfs dfs -mkdir /cc
keerthana@fedora:~/cc$ hdfs dfs -put hello.txt /cc/
```

```
keerthana@fedora:~/cc
43 errors
keerthana@fedora:~/cc$ javac -classpath $HADOOP_HOME/share/hadoop/common/*:$HADOOP_HOME/share/hadoop/mapreduce/*:. -d . WordCountMapper.java WordCountReducer.java WordCount.java
keerthana@fedora:~/cc$ jar cf wordcount.jar WordCount*.class
keerthana@fedora:~/cc$ hadoop jar wordcount.jar WordCount /cc/hello.txt /cc/output
2024-11-17 00:58:37,886 INFO client.DefaultNoHARMAFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-11-17 00:58:38,764 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2024-11-17 00:58:38,849 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/keerthana/.staging/job_1731784864991_0001
2024-11-17 00:58:39,355 INFO input.FileInputFormat: Total input files to process : 1
2024-11-17 00:58:40,023 INFO mapreduce.JobSubmitter: number of splits:1
2024-11-17 00:58:40,761 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1731784864991_0001
2024-11-17 00:58:40,761 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-11-17 00:58:41,227 INFO conf.Configuration: resource-types.xml not found
2024-11-17 00:58:41,227 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-11-17 00:58:42,390 INFO impl.YarnClientImpl: Submitted application application_1731784864991_0001
2024-11-17 00:58:42,508 INFO mapreduce.Job: The url to track the job: http://fedora:8088/proxy/application_1731784864991_0001/
2024-11-17 00:58:42,511 INFO mapreduce.Job: Running job: job_1731784864991_0001
2024-11-17 00:58:47,248 INFO mapreduce.Job: Job job_1731784864991_0001 running in uber mode : false
2024-11-17 00:58:57,254 INFO mapreduce.Job: map 0% reduce 0%
2024-11-17 00:59:04,485 INFO mapreduce.Job: map 100% reduce 0%
2024-11-17 00:59:11,615 INFO mapreduce.Job: map 100% reduce 100%
2024-11-17 00:59:12,663 INFO mapreduce.Job: Job job_1731784864991_0001 completed successfully
2024-11-17 00:59:12,807 INFO mapreduce.Job: Counters: 54
File System Counters
  FILE: Number of bytes read=105
  FILE: Number of bytes written=553005
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=144
```

```
keerthana@fedora:~/cc
HDFS: Number of bytes written=57
HDFS: Number of read operations=8
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
HDFS: Number of bytes read erasure-coded=0
Job Counters
  Launched map tasks=1
  Launched reduce tasks=1
  Data-local map tasks=1
  Total time spent by all maps in occupied slots (ms)=4698
  Total time spent by all reduces in occupied slots (ms)=4966
  Total time spent by all map tasks (ms)=4698
  Total time spent by all reduce tasks (ms)=4966
  Total vcore-milliseconds taken by all map tasks=4698
  Total vcore-milliseconds taken by all reduce tasks=4966
  Total megabyte-milliseconds taken by all map tasks=4810752
  Total megabyte-milliseconds taken by all reduce tasks=5085184
Map-Reduce Framework
  Map input records=1
  Map output records=9
  Map output bytes=81
  Map output materialized bytes=105
  Input split bytes=99
  Combine input records=0
  Combine output records=0
  Reduce input groups=8
  Reduce shuffle bytes=105
  Reduce input records=9
  Reduce output records=8
  Spilled Records=18
  Shuffled Maps =1
  Failed Shuffles=0
```



```
keerthana@fedora:~/cc

Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=214
CPU time spent (ms)=2130
Physical memory (bytes) snapshot=521986048
Virtual memory (bytes) snapshot=5089533952
Total committed heap usage (bytes)=404226048
Peak Map Physical memory (bytes)=326717440
Peak Map Virtual memory (bytes)=2541776896
Peak Reduce Physical memory (bytes)=195268608
Peak Reduce Virtual memory (bytes)=2547757056

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=45
File Output Format Counters
  Bytes Written=57
keerthana@fedora:~/cc$ hdfs dfs -cat /cc/output/*
Hai,      1
Hello,    1
How       2
are        1
health?   1
is         1
you?      1
your      1
keerthana@fedora:~/cc$
```

## **RESULT**

Thus a word count program in java is implemented using Map Reduce.