

**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:

```
kaarokki@fedora:~$ hadoop jar $HADOOP_STREAMING -input /exp1/s.txt -output /exp1/output1 -mapper ~/exp2/mapper.py -reducer ~/exp2/reducer.py
packageJobJar: [/tmp/hadoop-unjar4345487188727211757/] [] /tmp/streamjob2795676177018606624.jar tmpDir=null
2024-10-20 10:09:27,083 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-10-20 10:09:27,397 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-10-20 10:09:28,201 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/kaarokki/.staging/job_1729433151498_0001
2024-10-20 10:09:29,400 INFO mapred.FileInputFormat: Total input files to process : 1
2024-10-20 10:09:29,995 INFO mapreduce.JobSubmitter: number of splits:2
2024-10-20 10:09:30,314 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1729433151498_0001
2024-10-20 10:09:30,314 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-10-20 10:09:30,606 INFO conf.Configuration: resource-types.xml not found
2024-10-20 10:09:30,607 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-10-20 10:09:31,218 INFO impl.YarnClientImpl: Submitted application application_1729433151498_0001
2024-10-20 10:09:31,300 INFO mapreduce.Job: The url to track the job: http://fedora:8088/proxy/application_1729433151498_0001/
2024-10-20 10:09:31,302 INFO mapreduce.Job: Running job: job_1729433151498_0001
2024-10-20 10:09:42,667 INFO mapreduce.Job: Job job_1729433151498_0001 running in uber mode : false
2024-10-20 10:09:42,670 INFO mapreduce.Job: map 0% reduce 0%
2024-10-20 10:09:50,910 INFO mapreduce.Job: map 100% reduce 0%
2024-10-20 10:09:57,012 INFO mapreduce.Job: map 100% reduce 100%
2024-10-20 10:09:58,067 INFO mapreduce.Job: Job job_1729433151498_0001 completed successfully
2024-10-20 10:09:58,237 INFO mapreduce.Job: Counters: 54
File System Counters
FILE: Number of bytes read=278
```

```

2024-10-20 10:09:58,237 INFO mapreduce.Job: Counters: 54
  File System Counters
    FILE: Number of bytes read=278
    FILE: Number of bytes written=835554
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=396
    HDFS: Number of bytes written=175
    HDFS: Number of read operations=11
    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=2
    Launched reduce tasks=1
    Data-local map tasks=2
    Total time spent by all maps in occupied slots (ms)=11182
    Total time spent by all reduces in occupied slots (ms)=4117
    Total time spent by all map tasks (ms)=11182
    Total time spent by all reduce tasks (ms)=4117
    Total vcore-milliseconds taken by all map tasks=11182
    Total vcore-milliseconds taken by all reduce tasks=4117
    Total megabyte-milliseconds taken by all map tasks=11450368
    Total megabyte-milliseconds taken by all reduce tasks=4215808
  Map-Reduce Framework
    Map input records=7

```

```

kaaroKki@fedora:~$ hdfs dfs -cat /exp1/output/part-00000
Callin 1
Finally 1
LA 2
Lookin 1
Lost 1
Made 1
Maria 2
Might 1
Trynnna 1
dive 1
dough 1
for 2
in 2
it 1
make 1
marina 1
my 1
own 1
the 2
though 1
to 1
weed 1
without 1
yeah 2

```

## **RESULT**

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