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

210701262

$ 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 jat 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

<u>WordCount.java</u>

```
import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

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

210701262

```
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);
```


21070126 2

```
                    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
```

21070126 2

```
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;
```

210701262

```
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

210701262

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 fie 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

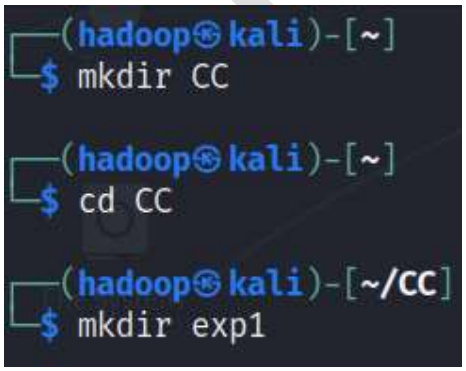
Step 11 – Go back to old Terminal for Execution of WordCount Program $hadoop jar wordcount.jar/usr/local/hadoop/input/usr/local/hadoop/output

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:



10701262

```
┌──(hadoop㉿kali)-[~/CC/exp1]
└─$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoop 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 [kali]
Starting resourcemanager
Starting nodemanagers
```

```
┌──(hadoop㉿kali)-[~/CC/exp1]
└─$ nano Mapper1.java

┌──(hadoop㉿kali)-[~/CC/exp1]
└─$ nano Reducer1.java

┌──(hadoop㉿kali)-[~/CC/exp1]
└─$ nano Runner1.java
```

```
┌──(hadoop㉿kali)-[~/CC/exp1]
└─$ nano word_count.txt
```

```
┌──(hadoop㉿kali)-[~/CC/exp1]
└─$ javac -classpath $HADOOP_HOME/share/hadoop/common/*:$HADOOP_HOME/share/hadoop/mapreduce/*:. -d . Mapper1.java Reducer1.java Runne
r1.java

┌──(hadoop㉿kali)-[~/CC/exp1]
└─$ jar -cvf wordcount.jar -C . .
added manifest
adding: Mapper1.class(in = 1858) (out= 759)(deflated 59%)
adding: Reducer1.class(in = 1527) (out= 604)(deflated 60%)
adding: Runner1.java(in = 1433) (out= 487)(deflated 66%)
adding: Runner1.class(in = 1432) (out= 709)(deflated 50%)
adding: Reducer1.java(in = 870) (out= 369)(deflated 57%)
adding: Mapper1.java(in = 1034) (out= 368)(deflated 64%)
```

```
┌──(hadoop㉿kali)-[~/CC/exp1]
└─$ hdfs dfs -mkdir /CC1

┌──(hadoop㉿kali)-[~/CC/exp1]
└─$ hdfs dfs -put word_count.txt /CC1

┌──(hadoop㉿kali)-[~/CC/exp1]
└─$ hadoop jar wordcount.jar Runner1 /CC1/word_count.txt /CC1/output
2024-11-17 05:47:17,122 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-11-17 05:47:17,691 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-11-17 05:47:18,848 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool inte
rface and execute your application with ToolRunner to remedy this.
2024-11-17 05:47:18,922 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.stagi
ng/job_1731839544220_0001
2024-11-17 05:47:19,736 INFO mapred.FileInputFormat: Total input files to process : 1
2024-11-17 05:47:19,976 INFO mapreduce.JobSubmitter: number of splits:2
2024-11-17 05:47:20,528 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1731839544220_0001
2024-11-17 05:47:20,529 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-11-17 05:47:21,076 INFO conf.Configuration: resource-types.xml not found
2024-11-17 05:47:21,077 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-11-17 05:47:21,786 INFO impl.YarnClientImpl: Submitted application application_1731839544220_0001
2024-11-17 05:47:21,966 INFO mapreduce.Job: The url to track the job: http://kali:8088/proxy/application_1731839544220_0001/
2024-11-17 05:47:21,979 INFO mapreduce.Job: Running job: job_1731839544220_0001
```

```
2024-11-17 05:47:21,979 INFO mapreduce.Job: Running job: job_1731839544220_0001
2024-11-17 05:47:41,589 INFO mapreduce.Job: Job job_1731839544220_0001 running in uber mode : false
2024-11-17 05:47:41,593 INFO mapreduce.Job:  map 0% reduce 0%
2024-11-17 05:47:57,000 INFO mapreduce.Job:  map 50% reduce 0%
2024-11-17 05:47:58,148 INFO mapreduce.Job:  map 100% reduce 0%
2024-11-17 05:48:10,378 INFO mapreduce.Job:  map 100% reduce 100%
2024-11-17 05:48:11,454 INFO mapreduce.Job: Job job_1731839544220_0001 completed successfully
2024-11-17 05:48:11,764 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=207
                FILE: Number of bytes written=829553
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=339
                HDFS: Number of bytes written=133
                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)=27588
```

```
                Total time spent by all maps in occupied slots (ms)=27588
                Total time spent by all reduces in occupied slots (ms)=9089
                Total time spent by all map tasks (ms)=27588
                Total time spent by all reduce tasks (ms)=9089
                Total vcore-milliseconds taken by all map tasks=27588
                Total vcore-milliseconds taken by all reduce tasks=9089
                Total megabyte-milliseconds taken by all map tasks=28250112
                Total megabyte-milliseconds taken by all reduce tasks=9307136
        Map-Reduce Framework
                Map input records=5
                Map output records=18
                Map output bytes=174
                Map output materialized bytes=213
                Input split bytes=184
                Combine input records=18
                Combine output records=17
                Reduce input groups=17
                Reduce shuffle bytes=213
                Reduce input records=17
                Reduce output records=17
                Spilled Records=34
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=650
```

```
                GC time elapsed (ms)=650
                CPU time spent (ms)=4340
                Physical memory (bytes) snapshot=800653312
                Virtual memory (bytes) snapshot=7771283456
                Total committed heap usage (bytes)=621805568
                Peak Map Physical memory (bytes)=301854720
                Peak Map Virtual memory (bytes)=2588606464
                Peak Reduce Physical memory (bytes)=199397376
                Peak Reduce Virtual memory (bytes)=2594144256
        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=155
        File Output Format Counters
                Bytes Written=133
```

```
┌──(hadoop㉿kali)-[~/CC/exp1]
└─$ hdfs dfs -ls /CC1/output
Found 2 items
-rw-r--r--   1 hadoop supergroup          0 2024-11-17 05:48 /CC1/output/_SUCCESS
-rw-r--r--   1 hadoop supergroup        133 2024-11-17 05:48 /CC1/output/part-00000

┌──(hadoop㉿kali)-[~/CC/exp1]
└─$ hdfs dfs -cat /CC1/output/part-00000
Everyday        1
Everything      1
Live    1
Today   1
a       1
an      1
beginning       1
change  1
day     1
in      1
is      2
moment  1
new     1
one     1
opportunity     1
the     1
will    1
```

RESULT

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