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 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

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 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:

```
harithaah@fedora:~/CC/exp2$ ls
Mapper1.java  Reducer1.java  Runner1.java  s.txt
harithaah@fedora:~/CC/exp2$ hdfs dfs -ls /
Found 7 items
drwxr-xr-x   - harithaah supergroup          0 2024-10-23 19:57 /cc
drwxr-xr-x   - harithaah supergroup          0 2024-10-10 20:38 /exp1
drwxr-xr-x   - harithaah supergroup          0 2024-10-10 20:47 /exp2
drwxr-xr-x   - harithaah supergroup          0 2024-10-10 21:02 /exp4
drwxr-xr-x   - harithaah supergroup          0 2024-10-10 21:13 /exp6
drwxr-xr-x   - harithaah supergroup          0 2024-10-10 21:13 /home
drwxr-xr-x   - harithaah supergroup          0 2024-10-10 21:08 /tmp
harithaah@fedora:~/CC/exp2$ hdfs dfs -mkdir /CC
harithaah@fedora:~/CC/exp2$ hdfs dfs -put s.txt /CC
```

```
harithaah@fedora:~/CC/exp2$ javac -classpath $HADOOP_HOME/share/hadoop/common/*:$HADOOP_HOME/share/hadoop/mapreduce/*:. -d . Mapper1.java Reducer1.java Runner1.java
harithaah@fedora:~/CC/exp2$ jar -cvf wordcount.jar -C . .
added manifest
adding: Mapper1.java(in = 1036) (out= 369)(deflated 64%)
adding: Reducer1.java(in = 871) (out= 368)(deflated 57%)
adding: Runner1.java(in = 1433) (out= 487)(deflated 66%)
adding: s.txt(in = 158) (out= 114)(deflated 27%)
adding: Mapper1.class(in = 1858) (out= 759)(deflated 59%)
adding: Reducer1.class(in = 1527) (out= 604)(deflated 60%)
adding: Runner1.class(in = 1432) (out= 709)(deflated 50%)
harithaah@fedora:~/CC/exp2$ hadoop jar wordcount.jar Runner1 /CC/s.txt /CC/output
2024-10-26 14:00:08,752 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-10-26 14:00:09,469 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-10-26 14:00:11,224 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application
with ToolRunner to remedy this.
2024-10-26 14:00:11,493 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/harithaah/.staging/job_1729930878688_0001
2024-10-26 14:00:13,797 INFO mapred.FileInputFormat: Total input files to process : 1
2024-10-26 14:00:15,965 INFO mapreduce.JobSubmitter: number of splits:2
2024-10-26 14:00:18,874 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1729930878688_0001
2024-10-26 14:00:18,875 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-10-26 14:00:19,491 INFO conf.Configuration: resource-types.xml not found
2024-10-26 14:00:19,495 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-10-26 14:00:21,006 INFO impl.YarnClientImpl: Submitted application application_1729930878688_0001
2024-10-26 14:00:21,385 INFO mapreduce.Job: The url to track the job: http://fedora:8088/proxy/application_1729930878688_0001/
2024-10-26 14:00:21,390 INFO mapreduce.Job: Running job: job_1729930878688_0001
2024-10-26 14:00:59,125 INFO mapreduce.Job: Job job_1729930878688_0001 running in uber mode : false
2024-10-26 14:00:59,130 INFO mapreduce.Job:  map 0% reduce 0%
2024-10-26 14:01:26,297 INFO mapreduce.Job:  map 100% reduce 0%
2024-10-26 14:01:46,586 INFO mapreduce.Job:  map 100% reduce 100%
2024-10-26 14:01:48,821 INFO mapreduce.Job: Job job_1729930878688_0001 completed successfully
2024-10-26 14:01:48,756 INFO mapreduce.Job: Counters: 54
```

```
harithaah@fedora:~/CC/exp2$ hdfs dfs -cat /CC/output/part-00000
B       1
CSE     1
From    1
Hello   1
Hey     1
We      1
a       1
an      1
are     1
awesome 1
be      1
ever    1
found   1
get     1
girl    2
happy   1
have    2
her     2
here    1
i       3
if      1
is      1
like    1
never   1
```

RESULT

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