

EXP.NO:04

DATE:

RUN A BASIC WORD COUNT MAP REDUCE PROGRAM TO UNDERSTAND MAP REDUCE PARADIGM

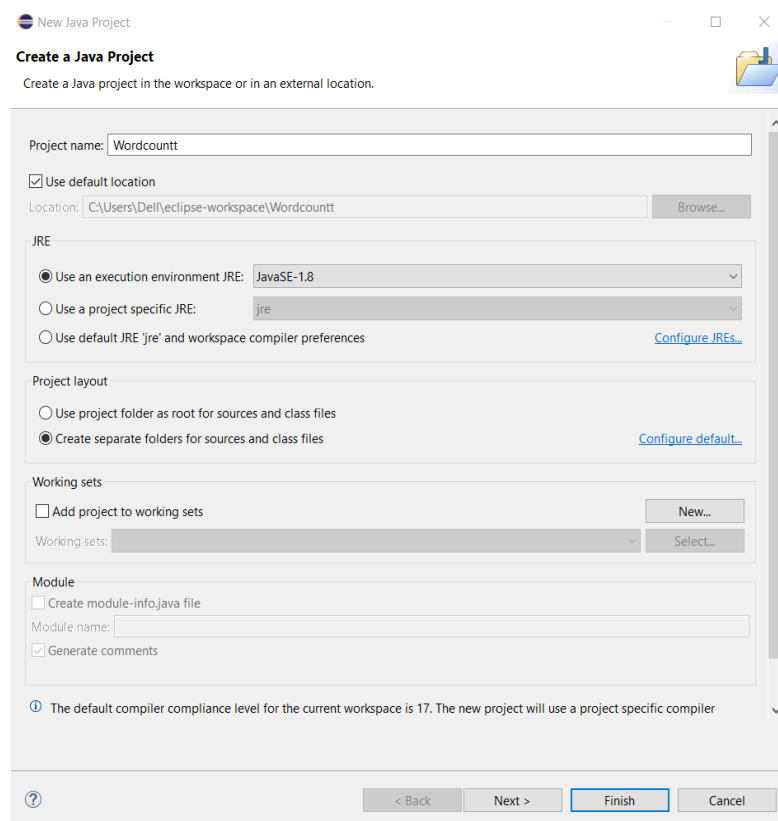
AIM: To Run a Basic Word Count Map Reduce program to understand Map Reduce Paradigm.

STEPS:

STEP 1: Run Eclipse for Java Developers

STEP 2: Create a new Java Project with name “WordCount “

STEP 3: Set the Java Environment Version to your current version of Java (JRE : 1.8)



STEP 4 : Add a Package with name “com.mapreduce.java” and Create three Classes in it.

STEP 5 : Create a New Class With name WC_Mapper.java.

STEP 6: Now write the below program in the “WC_Mapper.java” Class

PROGRAM:

```
package com.mapreduce.java;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reporter;

public class WC_Mapper extends MapReduceBase implements
Mapper<LongWritable,Text,Text,IntWritable>{

    private final static IntWritable one = new IntWritable(1);

    private Text word = new Text();

    public void map(LongWritable key, Text value,OutputCollector<Text,IntWritable> output,

        Reporter reporter) throws IOException{

        String line = value.toString();

        StringTokenizer tokenizer = new StringTokenizer(line);

        while (tokenizer.hasMoreTokens()){

            word.set(tokenizer.nextToken());

            output.collect(word, one);

        }

    }

}
```

```
}  
  
}
```

STEP 7: Now Create another class with name “WC_Reducer.java” and paste the below program in it.

PROGRAM:

```
package com.mapreduce.java;  
  
import java.io.IOException;  
  
import java.util.Iterator;  
  
import org.apache.hadoop.io.IntWritable;  
  
import org.apache.hadoop.io.Text;  
  
import org.apache.hadoop.mapred.MapReduceBase;  
  
import org.apache.hadoop.mapred.OutputCollector;  
  
import org.apache.hadoop.mapred.Reducer;  
  
import org.apache.hadoop.mapred.Reporter;  
  
public class WC_Reducer extends MapReduceBase implements  
Reducer<Text,IntWritable,Text,IntWritable> {  
  
    public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text,IntWritable> output,  
  
        Reporter reporter) throws IOException {  
  
        int sum=0;  
  
        while (values.hasNext()) {  
  
            sum+=values.next().get();  
  
        }  
  
        output.collect(key,new IntWritable(sum));  
  
    }  
}
```

```
}
```

STEP 8: Now, Create another class with name “WC_runner.java” and paste the below program in it.

PROGRAM:

```
package com.mapreduce.java;

import java.io.IOException;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

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.mapred.TextInputFormat;

import org.apache.hadoop.mapred.TextOutputFormat;

public class WC_Runner {

    public static void main(String[] args) throws IOException{

        JobConf conf = new JobConf(WC_Runner.class);

        conf.setJobName("WordCount");

        conf.setOutputKeyClass(Text.class);

        conf.setOutputValueClass(IntWritable.class);

        conf.setMapperClass(WC_Mapper.class);

        conf.setCombinerClass(WC_Reducer.class);

        conf.setReducerClass(WC_Reducer.class);
```

```

conf.setInputFormat(TextInputFormat.class);

conf.setOutputFormat(TextOutputFormat.class);

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

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

JobClient.runJob(conf);

}

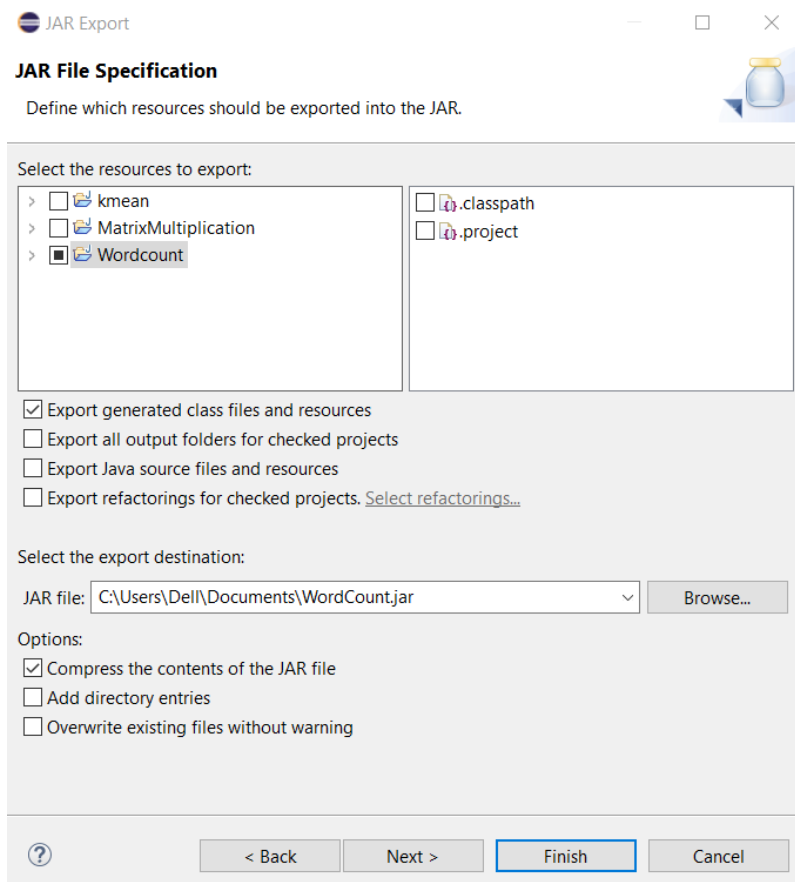
}

```

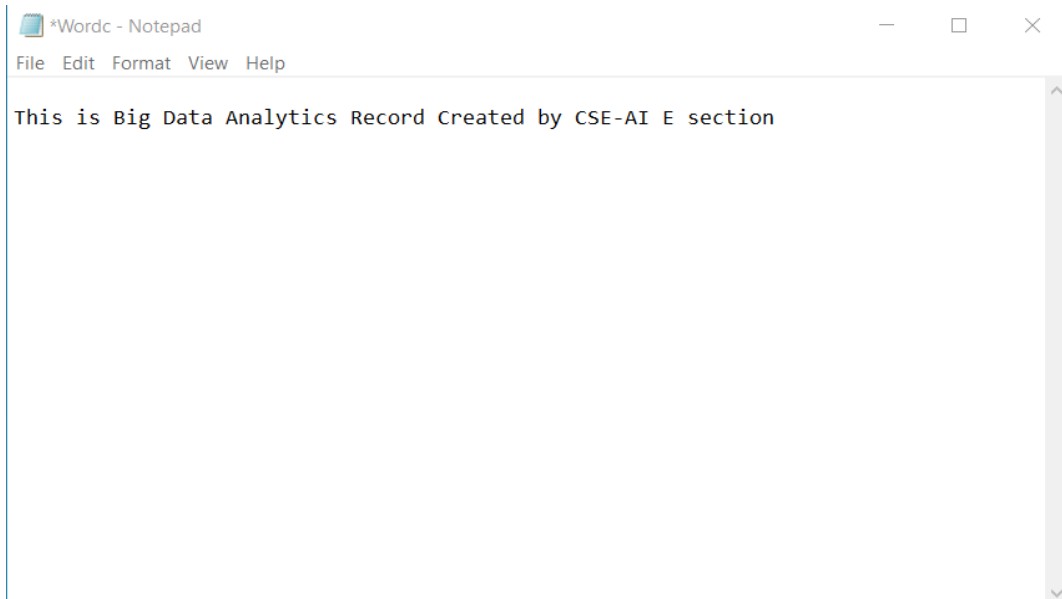
STEP 9: To resolve the errors in the programs we should add two External jar files to it.

- Hadoop_common :2.7.3.jar
- Hadoop_mapreduce:client:core:2.7.1.jar

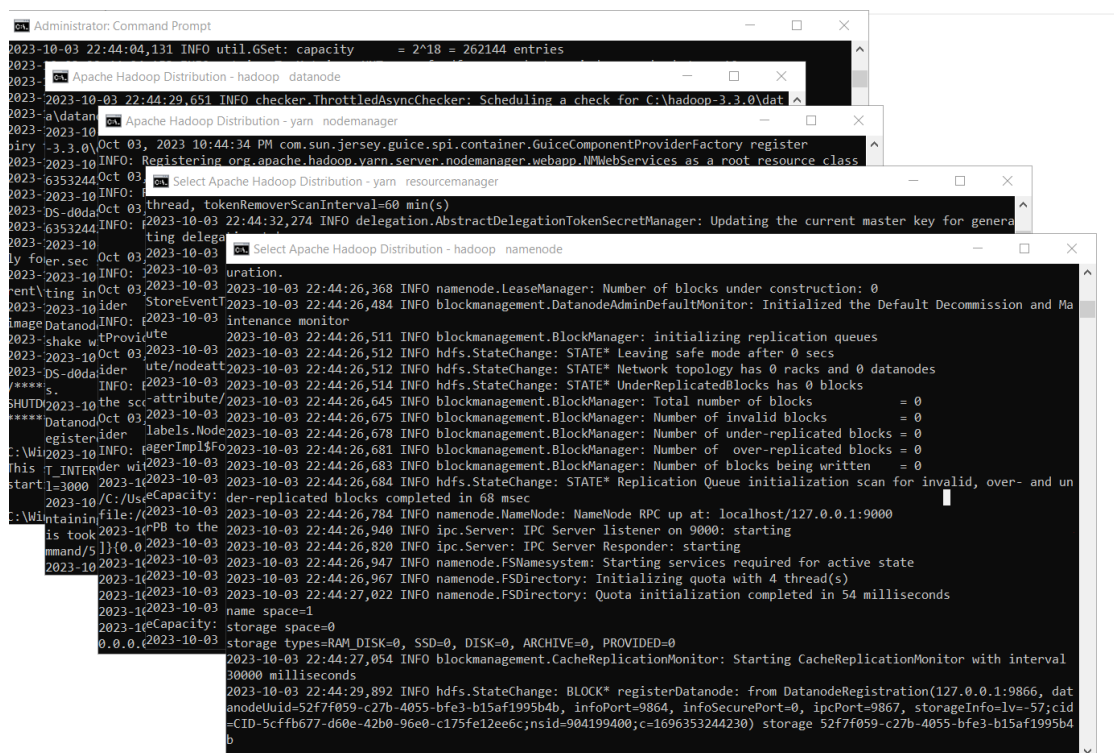
STEP 10: Now export the project into a Jar file and name it as “WordCount.jar”



STEP 11: Now create a Text file in Notepad and name it as “wordc.txt” and write some content inside the text file and save it.



STEP 12: Now run all the daemons in Hadoop.



STEP 13: Create a new input directory named as “inputword”.

By using the command: `hadoop fs :mkdir /inputword`

STEP 14: Now put the “wordc.txt” file to the inputword directory.

By using the command: `hadoop fs :put C:\Users\Dell\Documents\wordc.txt /inputword`

```
Administrator: Command Prompt
2023-10-03 23:20:35,035 INFO util.GSet: Computing capacity for map NameNodeRetryCache
2023-10-03 23:20:35,035 INFO util.GSet: VM type = 64-bit
2023-10-03 23:20:35,036 INFO util.GSet: 0.029999999329447746% max memory 889 MB = 273.1 KB
2023-10-03 23:20:35,036 INFO util.GSet: capacity = 2^15 = 32768 entries
2023-10-03 23:20:35,096 INFO namenode.FSImage: Allocated new BlockPoolId: BP-1025061849-192.168.0.100-1696355435084
2023-10-03 23:20:35,147 INFO common.Storage: Storage directory C:\hadoop-3.3.0\data\namenode1290 has been successfully formatted.
2023-10-03 23:20:35,194 INFO namenode.FSImageFormatProtobuf: Saving image file C:\hadoop-3.3.0\data\namenode1290\current\fsimage.ckpt_00000000000000000000 using no compression
2023-10-03 23:20:35,398 INFO namenode.FSImageFormatProtobuf: Image file C:\hadoop-3.3.0\data\namenode1290\current\fsimage.ckpt_00000000000000000000 of size 397 bytes saved in 0 seconds .
2023-10-03 23:20:35,420 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2023-10-03 23:20:35,427 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2023-10-03 23:20:35,428 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at Azhar/192.168.0.100
*****/

C:\Windows\system32>cd C:\hadoop-3.3.0\sbin

C:\hadoop-3.3.0\sbin>start-all
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\hadoop-3.3.0\sbin>hadoop fs -mkdir /inputword

C:\hadoop-3.3.0\sbin>hadoop fs -put C:\Users\Dell\Documents\wordc.txt /inputword

C:\hadoop-3.3.0\sbin>
```

STEP 15: Run the Jar file created from the project

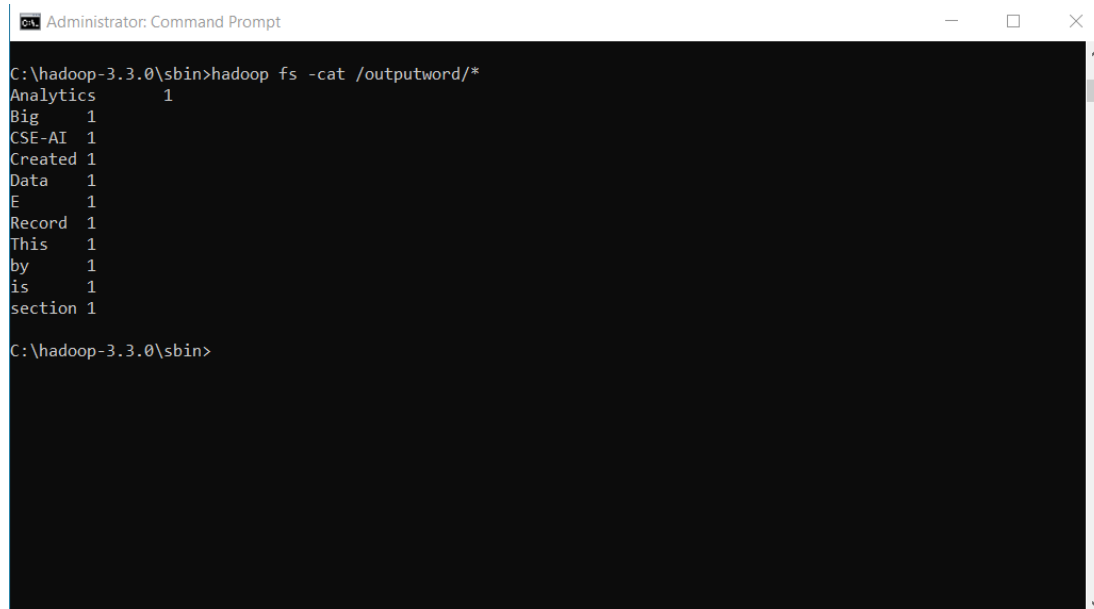
Using the command: `hadoop jar C:\Users\Dell\Documents\Wordcount.jar com.mapreduce.java/WC_Runner /inputword/* /outputword`

```
Administrator: Command Prompt
C:\hadoop-3.3.0\sbin>hadoop jar C:\Users\Dell\Documents\WordCount.jar com.mapreduce.java/WC_Runner /inputword/* /outputword
2023-10-03 23:46:48,672 INFO client.DefaultNoHARMAFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2023-10-03 23:46:48,970 INFO client.DefaultNoHARMAFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2023-10-03 23:46:49,978 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2023-10-03 23:46:50,010 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/AZHAR/.staging/job_1696356523104_0001
2023-10-03 23:46:50,981 INFO mapred.FileInputFormat: Total input files to process : 1
2023-10-03 23:46:51,183 INFO mapreduce.JobSubmitter: number of splits:2
2023-10-03 23:46:51,605 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1696356523104_0001
2023-10-03 23:46:51,605 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-10-03 23:46:51,967 INFO conf.Configuration: resource-types.xml not found
2023-10-03 23:46:51,968 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2023-10-03 23:46:52,672 INFO impl.YarnClientImpl: Submitted application application_1696356523104_0001
2023-10-03 23:46:52,765 INFO mapreduce.Job: The url to track the job: http://Azhar:8088/proxy/application_1696356523104_0001/
2023-10-03 23:46:52,770 INFO mapreduce.Job: Running job: job_1696356523104_0001
2023-10-03 23:47:07,174 INFO mapreduce.Job: Job job_1696356523104_0001 running in uber mode : false
2023-10-03 23:47:07,178 INFO mapreduce.Job: map 0% reduce 0%
2023-10-03 23:47:17,788 INFO mapreduce.Job: map 50% reduce 0%
2023-10-03 23:47:18,807 INFO mapreduce.Job: map 100% reduce 0%
2023-10-03 23:47:26,957 INFO mapreduce.Job: map 100% reduce 100%
2023-10-03 23:47:27,996 INFO mapreduce.Job: Job job_1696356523104_0001 completed successfully
```

STEP 16: At last Print your output for the WordCount text file.

Using the Command : `hadoop fs :cat /outputword/*`

OUTPUT:



```
Administrator: Command Prompt
C:\hadoop-3.3.0\sbin>hadoop fs -cat /outputword/*
Analytics      1
Big            1
CSE-AI         1
Created        1
Data           1
E              1
Record         1
This           1
by             1
is             1
section        1

C:\hadoop-3.3.0\sbin>
```

RESULT: Thus the program to run a basic wordcount mapreduce program to understand mapreduce is executed and output is verified successfully.