EXPERIMENT NO: 4

AIM: Experiment on word counting using Hadoop map-reduce

Theory:

Word counting using Hadoop MapReduce is a fundamental example of how distributed computing can process vast amounts of data efficiently. This experiment demonstrates how to leverage the power of Hadoop, a popular big data framework, to analyse and count the occurrences of words within large datasets. Below is a comprehensive explanation of this experiment:

Introduction to Word Counting with Hadoop MapReduce:

Hadoop MapReduce is a programming model and processing framework that allows users to perform distributed data processing tasks. One of the most common introductory exercises in Hadoop MapReduce is word counting, which involves determining the frequency of each word in a given set of documents or text data.

Experiment Setup:

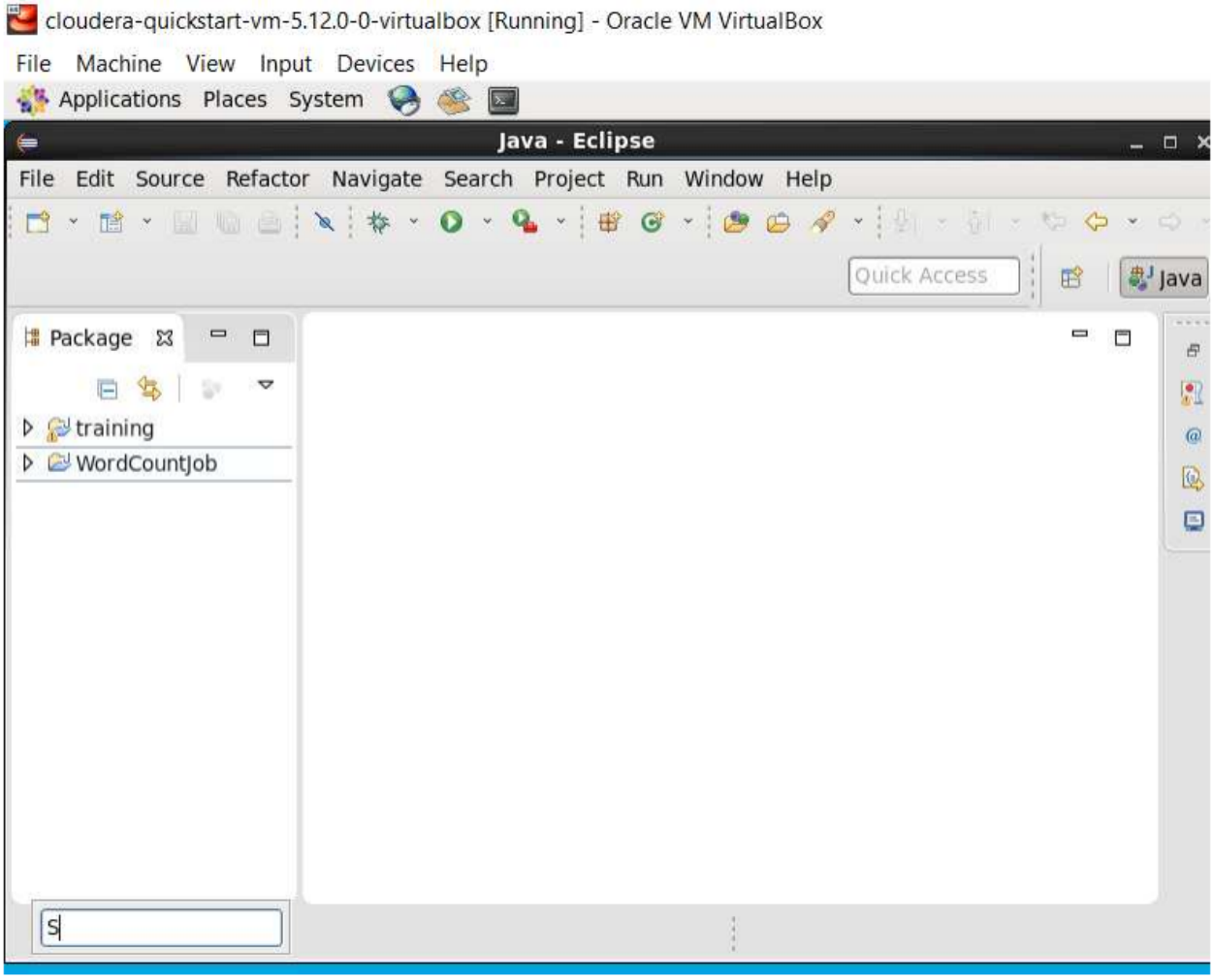
1. **Data Input**: The first step in this experiment is to have a set of input text documents. These documents could be stored in Hadoop HDFS (Hadoop Distributed File System), and they are divided into smaller chunks called blocks.
2. **Map Phase**: The MapReduce process starts with the "Map" phase, where each Mapper task processes a portion of the input data. Mappers read the text, tokenize it into words, and emit key-value pairs where the key is the word and the value is usually 1, signifying the occurrence of that word.
3. **Shuffle and Sort**: After the Map phase, the framework groups all the emitted key-value pairs by keys and sorts them. This phase ensures that all occurrences of the same word are grouped together.
4. **Reduce Phase**: The "Reduce" phase takes the grouped and sorted key-value pairs and applies a "Reducer" function to aggregate the counts. In the context of word counting, the Reducer sums up the values for each word key, giving the final count of occurrences.

STEP 1 :

Create New Java Project in Eclipse

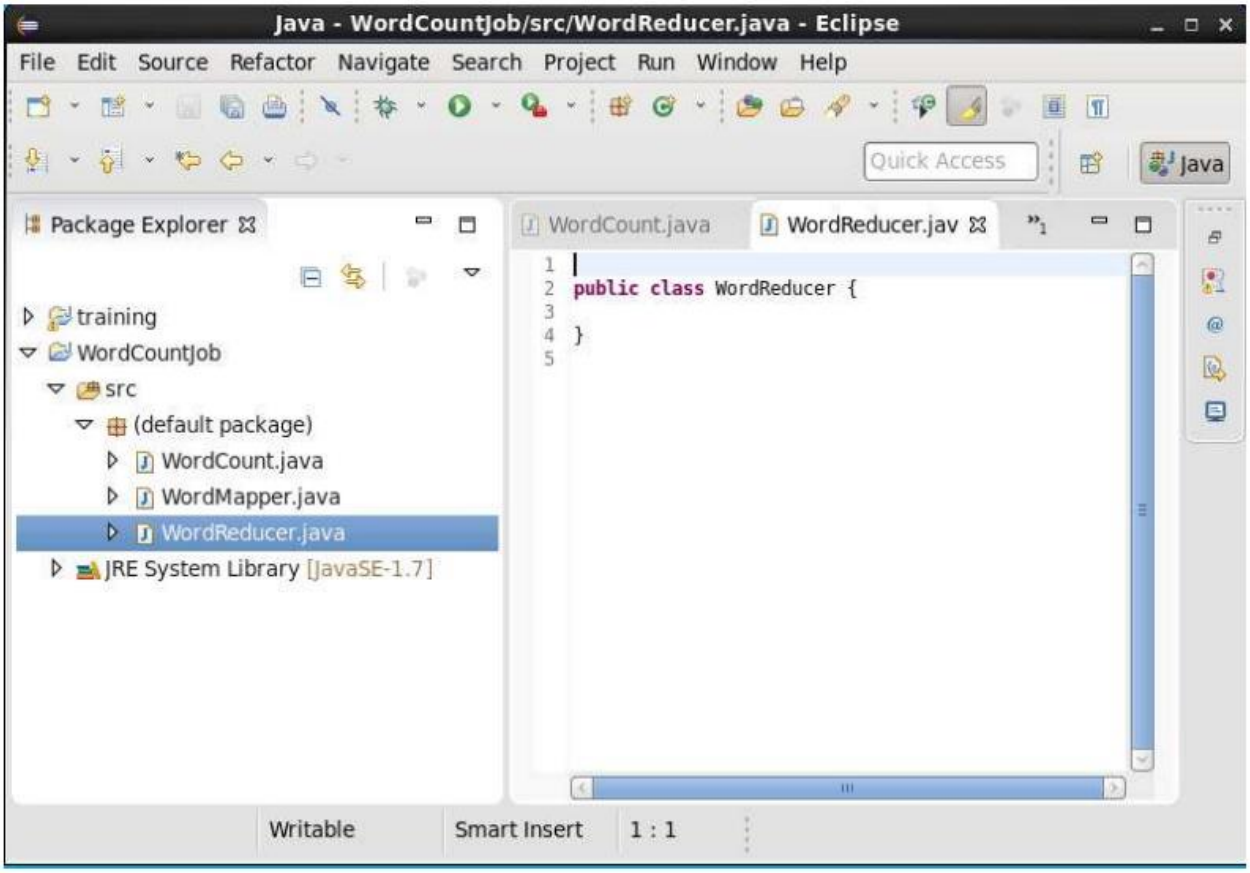
Name project :

WordCounJob click : finish



STEP 2:

Right Click on project -> New -> Class create class files as Name : WordCount Name : WordMapper Name : WordReducer



Code for all the files is :

Mapper:

```
// Importing libraries

import java.io.IOException;

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 WCMapper extends MapReduceBase implements Mapper<LongWritable,

                                                                Text,

Text, IntWritable> {
```

```
// Map function
public void map(LongWritable key, Text value, OutputCollector<Text,
                                IntWritable> output, Reporter rep) throws IOException
{

        String line = value.toString();

        // Splitting the line on spaces
        for (String word : line.split(" "))
        {
                if (word.length() > 0)
                {
                        output.collect(new Text(word), new IntWritable(1));
```

reducer:

```
// Importing libraries
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 WCReducer extends MapReduceBase implements Reducer<Text,
                                                        IntWritable, Text, IntWritable> {

        // Reduce function
        public void reduce(Text key, Iterator<IntWritable> value,
                                OutputCollector<Text, IntWritable> output,
                                                        Reporter rep) throws IOException
        {

                int count = 0;
```

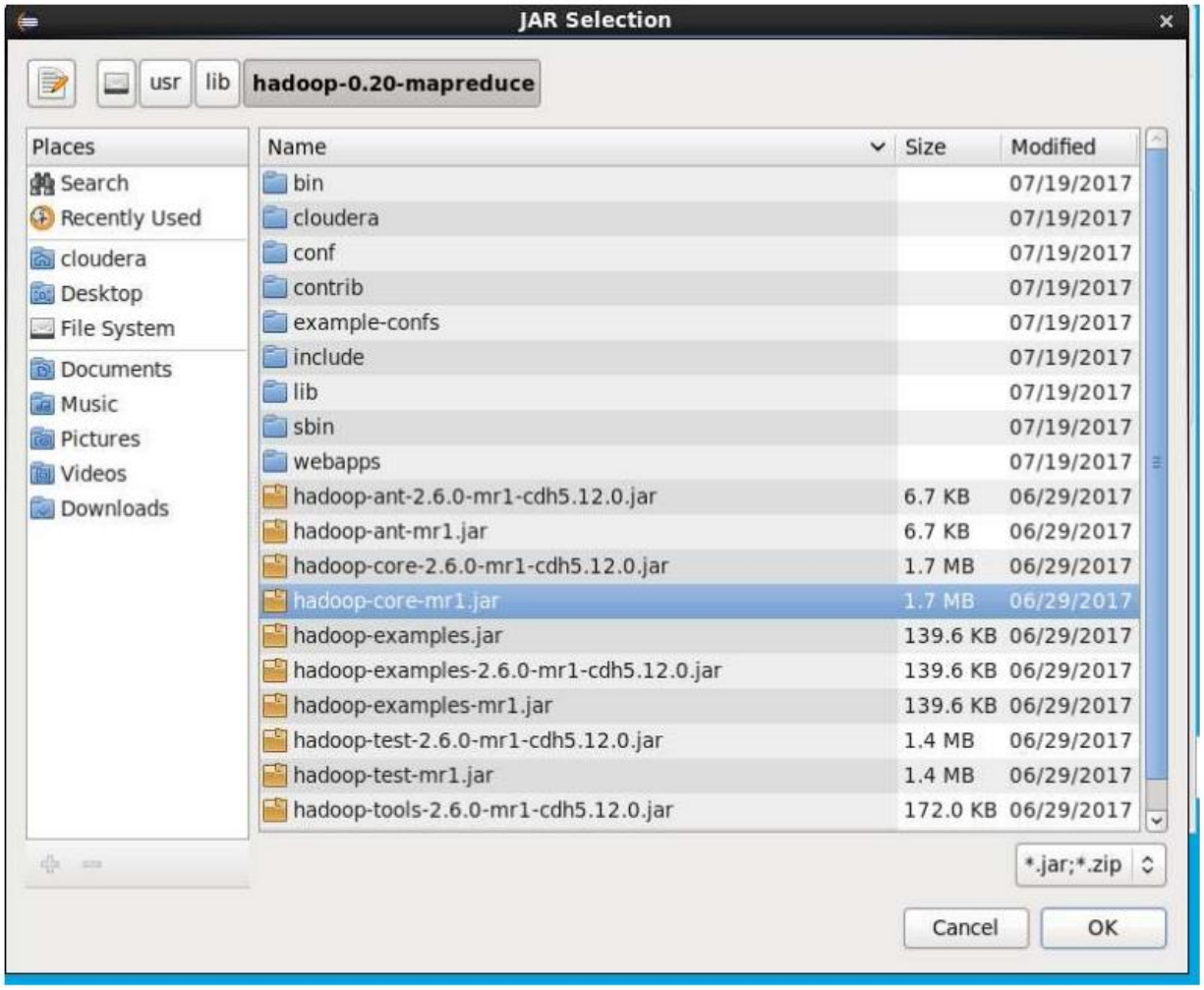
```
                // Counting the frequency of each words
                while (value.hasNext())
                {
                        IntWritable i = value.next();
                        count += i.get();
                }

                output.collect(key, new IntWritable(count));
        }
}
```

Word count:

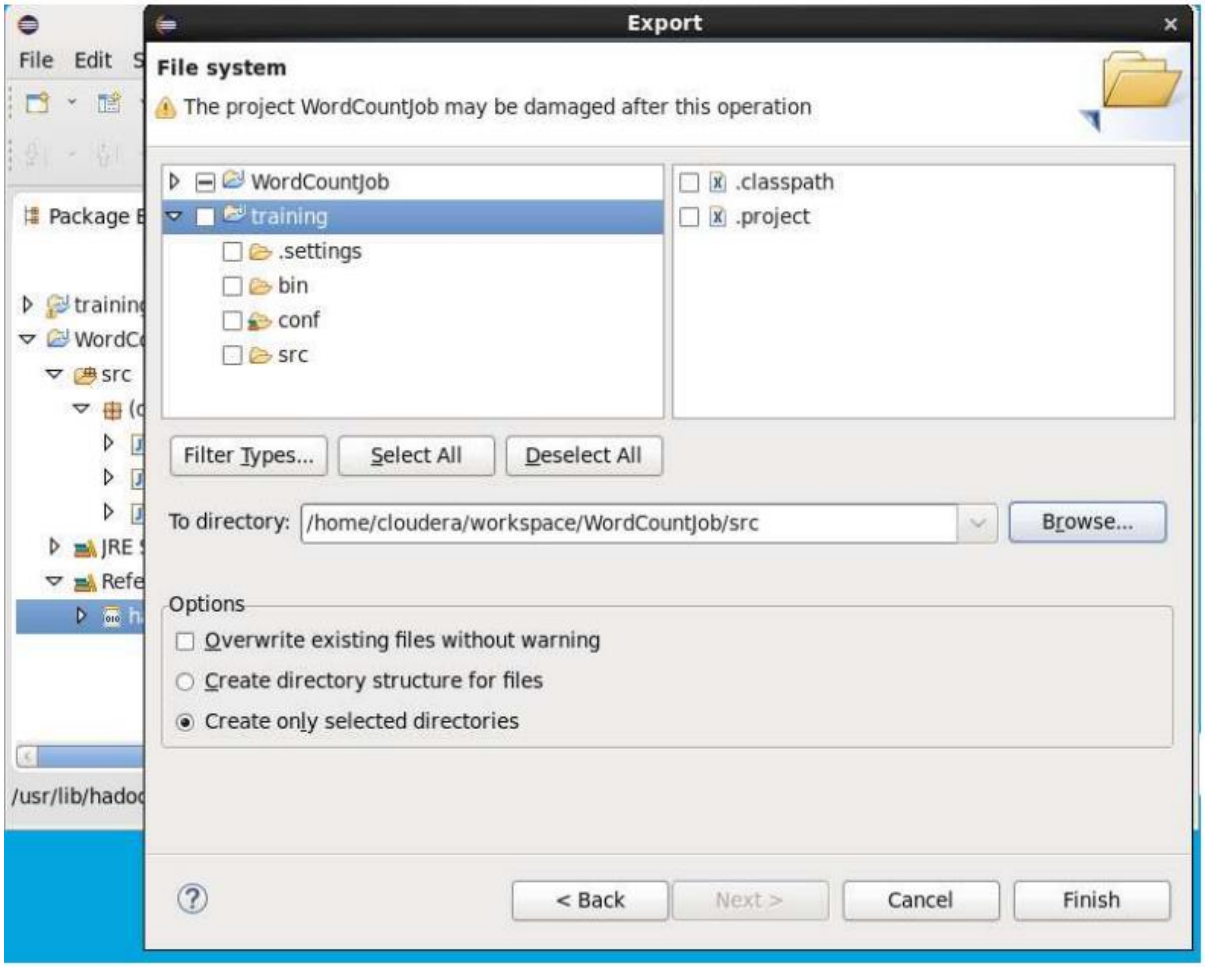
```
// Importing libraries
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool {

        public int run(String args[]) throws IOException
        {
                if (args.length < 2)
                {
                        System.out.println("Please give valid inputs");
                        return -1;
                }

                JobConf conf = new JobConf(WCDriver.class);
```

```
                FileInputFormat.setInputPaths(conf, new Path(args[0]));
                FileOutputFormat.setOutputPath(conf, new Path(args[1]));
                conf.setMapperClass(WCMapper.class);
                conf.setReducerClass(WCReducer.class);
                conf.setMapOutputKeyClass(Text.class);
                conf.setMapOutputValueClass(IntWritable.class);
                conf.setOutputKeyClass(Text.class);
                conf.setOutputValueClass(IntWritable.class);
                JobClient.runJob(conf);
                return 0;
        }

        // Main Method
        public static void main(String args[]) throws Exception
        {
                int exitCode = ToolRunner.run(new WCDriver(), args);
                System.out.println(exitCode);
        }
}
```

STEP 3 : Right Click Project -> build path -> add expertnal archievs -> filesystem -> usr -> lib -> hadoop-0.20 -> hadoop-core.jar click add



STEP 4 : export the jar file in same folder
(training/workspace/WordCountJob/src)



STEP 5: open terminal [training@localhost ~]$ ls -l

```
[cloudera@quickstart ~]$ ls -l
total 252
drwxrwxr-x 2 cloudera cloudera  4096 Aug 26 03:29 BDApracs
-rwxrwxr-x 1 cloudera cloudera  5387 Jul 19  2017 cloudera-manager
-rwxrwxr-x 1 cloudera cloudera  9964 Jul 19  2017 cm_api.py
drwxrwxr-x 2 cloudera cloudera  4096 Aug 28 01:24 data
-rw-rw-r-- 1 cloudera cloudera    67 Aug 27 00:31 demo.txt
drwxrwxr-x 3 cloudera cloudera  4096 Aug 26 03:44 Desktop
drwxrwxr-x 4 cloudera cloudera  4096 Jul 19  2017 Documents
drwxr-xr-x 2 cloudera cloudera  4096 Aug 26 03:21 Downloads
drwxrwsr-x 9 cloudera cloudera  4096 Aug 29 04:02 eclipse
-rw-rw-r-- 1 cloudera cloudera 53655 Jul 19  2017 enterprise-deployment.json
-rw-rw-r-- 1 cloudera cloudera 50515 Jul 19  2017 express-deployment.json
-rwxrwxr-x 1 cloudera cloudera  5007 Jul 19  2017 kerberos
drwxrwxr-x 2 cloudera cloudera  4096 Jul 19  2017 lib
drwxr-xr-x 2 cloudera cloudera  4096 Aug 26 03:21 Music
drwxrwxr-x 2 cloudera cloudera  4096 Aug 26 23:29 myfirstdata
-rwxrwxr-x 1 cloudera cloudera  4228 Jul 19  2017 parcels
drwxr-xr-x 2 cloudera cloudera  4096 Aug 26 03:21 Pictures
drwxr-xr-x 2 cloudera cloudera  4096 Aug 26 03:21 Public
-rw-rw-r-- 1 cloudera cloudera 18353 Aug 26 23:46 registercopy.java
-rw-rw-r-- 1 cloudera cloudera 18281 Aug 26 23:36 register.java
drwxr-xr-x 2 cloudera cloudera  4096 Aug 26 03:21 Templates
drwxrwxr-x 2 cloudera cloudera  4096 Aug 29 00:11 test
drwxrwxr-x 2 cloudera cloudera  4096 Aug 27 00:33 training
drwxr-xr-x 2 cloudera cloudera  4096 Aug 26 03:21 Videos
drwxrwxr-x 6 cloudera cloudera  4096 Aug 29 05:26 workspace
[cloudera@quickstart ~]$
```

STEP 6: create sample file [training@localhost ~]$ cat WCFile.txt Hello we are doing BDA pracs Hello I’m a student Put the file in Hadoop

```
[cloudera@quickstart workspace]$ cat WCFile.txt
Hello we are doing BDA pracs
Hello I'm a student
[cloudera@quickstart workspace]$
```

STEP 7 : Put the content of sample.txt to samplehadoop.txt
[training@localhost ~]$ hadoop fs -put WCFile.txt WCFile.txt
[training@localhost ~]$ hadoop fs -ls

```
[cloudera@quickstart workspace]$ hadoop fs -put WCFile.txt WCFile.txt
[cloudera@quickstart workspace]$ hadoop fs -ls
Found 4 items
-rw-r--r--   1 cloudera cloudera         49 2022-08-29 05:32 WCFile.txt
drwxr-xr-x   - cloudera cloudera          0 2022-08-26 03:41 hadoop
drwxr-xr-x   - cloudera cloudera          0 2022-08-26 04:08 raunak
drwxr-xr-x   - cloudera cloudera          0 2022-08-26 04:11 test.txt
[cloudera@quickstart workspace]$
```

STEP 8 : change the directory to [training@localhost ~]$ cd /home/training/workspace/WordCount/src

```
[cloudera@quickstart ~]$ cd workspace
[cloudera@quickstart workspace]$ cd WordCount
[cloudera@quickstart WordCount]$ cd src
[cloudera@quickstart src]$ ls
WCDriver.java  WCMapper.java  WCReducer.java
[cloudera@quickstart src]$
```

STEP 9: Listing the contents of that directory [cloudera@quickstart src]$ ls WCDriver.java WCMapper.java WCReducer.java

STEP 10 : Run the wordcount program and collect the output in WCOutput [cloudera@quickstart workspace]$ hadoop jar WordCount.jar WCDriver WCFile.txt WCOutput

```
[cloudera@quickstart workspace]$ hadoop jar WordCount.jar WCDriver WCFile.txt WC
Output
22/08/29 05:37:49 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
22/08/29 05:37:50 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
22/08/29 05:37:50 WARN mapreduce.JobResourceUploader: Hadoop command-line option
 parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
22/08/29 05:37:51 INFO mapred.FileInputFormat: Total input paths to process : 1
22/08/29 05:37:51 INFO mapreduce.JobSubmitter: number of splits:2
22/08/29 05:37:51 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_16
61770860486_0001
22/08/29 05:37:52 INFO impl.YarnClientImpl: Submitted application application_16
61770860486_0001
22/08/29 05:37:53 INFO mapreduce.Job: The url to track the job: http://quickstar
t.cloudera:8088/proxy/application_1661770860486_0001/
22/08/29 05:37:53 INFO mapreduce.Job: Running job: job_1661770860486_0001
22/08/29 05:38:08 INFO mapreduce.Job: Job job_1661770860486_0001 running in uber
 mode : false
22/08/29 05:38:08 INFO mapreduce.Job:  map 0% reduce 0%
22/08/29 05:38:21 INFO mapreduce.Job:  map 50% reduce 0%
22/08/29 05:38:22 INFO mapreduce.Job:  map 100% reduce 0%
22/08/29 05:38:28 INFO mapreduce.Job:  map 100% reduce 100%
22/08/29 05:38:29 INFO mapreduce.Job: Job job_1661770860486_0001 completed succe
ssfully
22/08/29 05:38:30 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=115
                FILE: Number of bytes written=375712
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=290
```

STEP 11 : chk the output file [cloudera@quickstart workspace]$ hadoop fs -ls WCOutput

STEP 12: Displaying the Output [cloudera@quickstart workspace]$ hadoop fs -cat WCOutput/part-00000

```
[cloudera@quickstart workspace]$ hadoop fs -cat WCOutput/part-00000
BDA     1
Hello   2
I'm     1
a       1
are     1
doing   1
pracs   1
student 1
we      1
[cloudera@quickstart workspace]$
```