Omkar Gurav
TE IT Batch T3
Roll No:- 8048

DATE:

Assignment - 2

- Aim:- Design a distributed application using MapReduce which process a text file. List out the count of each word occuring in the file.

- Theory:-

Q.1. Explain the need & concept of MapReduce?

→ MapReduce is a software framework which is used to write applications to process huge amounts of data simultaneously on a large number of nodes in reliable way.

MapReduce is required when we need to process huge amounts of data that cannot be handled with regular frameworks.

MapReduce primarly refers to two tasks

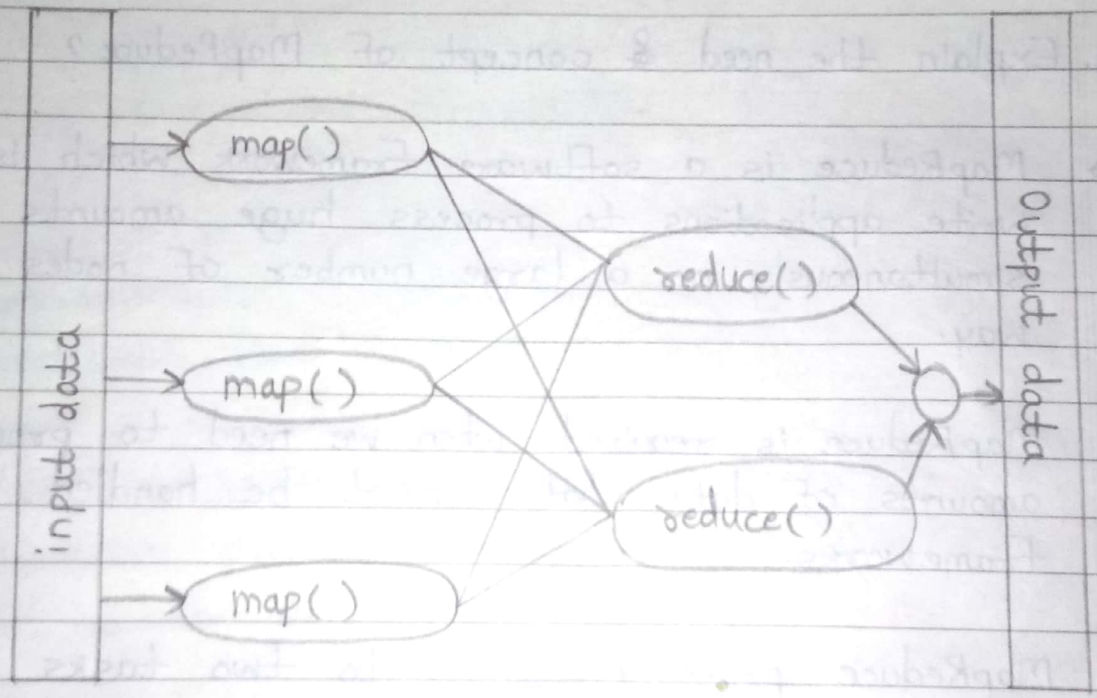
① Map: Here one set of data is converted into another set of data in which the individual elements are broken down into tuples i.e. into key/value pairs.

② Reduce: This task takes the output of the map task as its input & combine the data tuples into a smaller tuples. It is always performed after the map task.

DATE:

Advantages of MapReduce Framework:

① Easy to scale
② Handles task scheduling, monitoring, failure & execution
③ Fault tolerant
④ Simple & easy to understand
⑤ Has support for unstructed data



Q.2. Explain the following

① Job Tracker

→ Job Tracker is the master for job management, scheduling & execution in the Hadoop framework.

Initially the user copies files into HDFS with the -put or -copyFromLocal commands.

DATE:

The job is submitted via the job tracker. It runs on the same node which runs other jobs on data nodes.

The job is initialized in the job queue & the job tracker creates maper & reduces. The map & reduce tasks will depend on the input programs that user provides

Job tracker primarily performs 4 tasks-

① Resource management
② Resource Availability
③ Monitoring
④ Scheduling

② Task Tracker

→ The task tracker accepts tasks assigned by job tracker on the master node while itself running on slave nodes.

It divides the JVM (Java Virtual Machine) processes & threads to run these tasks. The task tracker reports the progress of these tasks & health status

Hadoop maintains 3 lists for task trackers

① Blacklist : Used to blacklist a task tracker if performance is not optimal or unstable.

② Grey list: a list of potentially problematic nodes

DATE:

③ Excluded list: list of excluded task trackers

- Conclusion: Using the MapReduce Framework, application to design a distributed app to process a text file & list out word count has been successfully designed.

Driver class : Driver.java

```
package words;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.conf.Configuration;

public class Driver {

       @SuppressWarnings("deprecation")
    public static void main(String[] args) throws Exception{
    //creating object of configuration class
    Configuration c = new Configuration();

    //Assigning job to new configuration object
    Job job = new Job(c);

    //setting jar class
    job.setJarByClass(words.Driver.class);

    job.setMapperClass(words.WordMapper.class);

    job.setReducerClass(words.WordReducer.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    //Adding a Path to the list of inputs
    FileInputFormat.addInputPath(job, new Path(args[0]));

    //Setting the Path of the output directory
    FileOutputFormat.setOutputPath(job,new Path(args[1]));

    //wait till job is completed
    System.exit(job.waitForCompletion(true)?0:1);
}

}
```

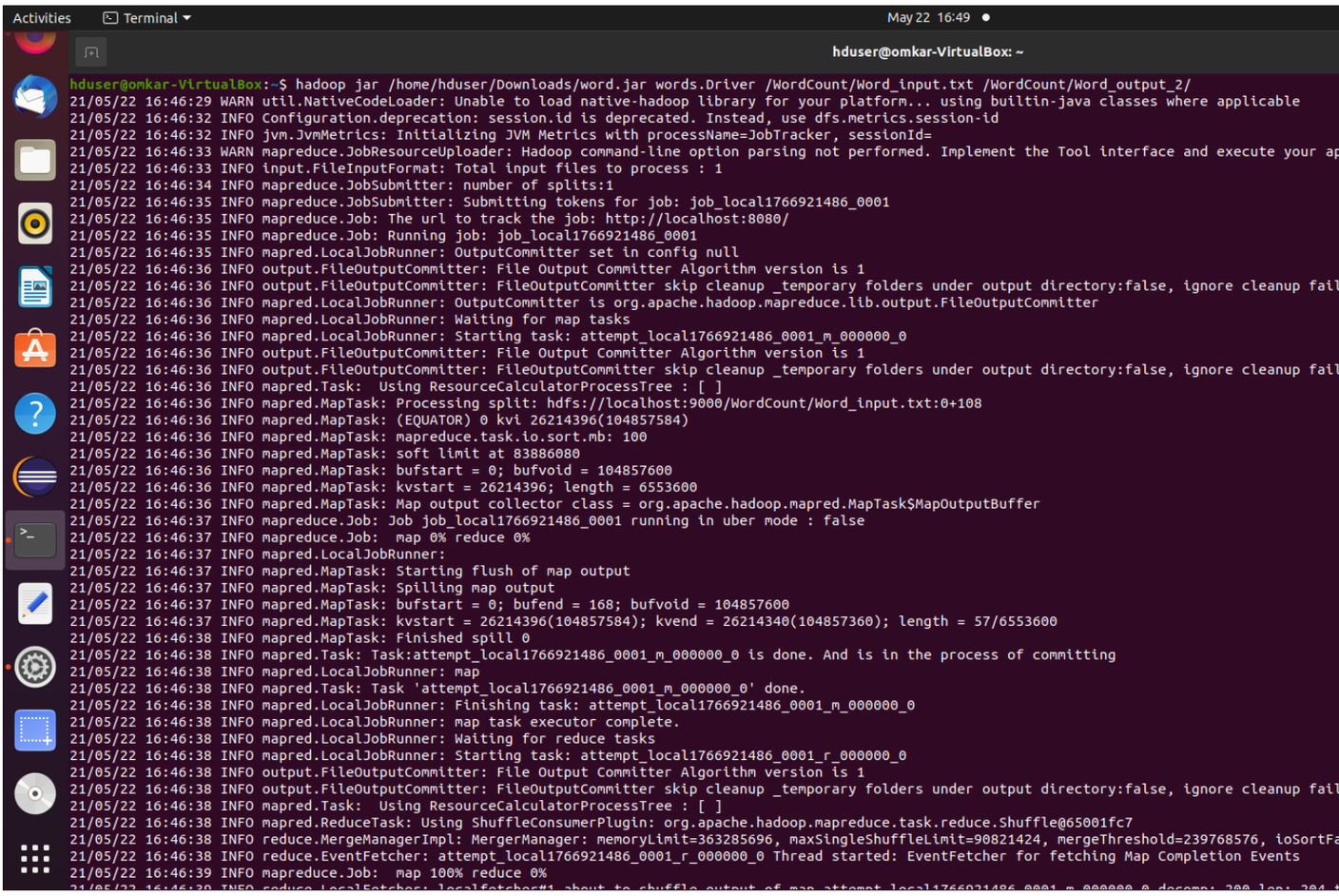
Mapper class : WordMapper.java

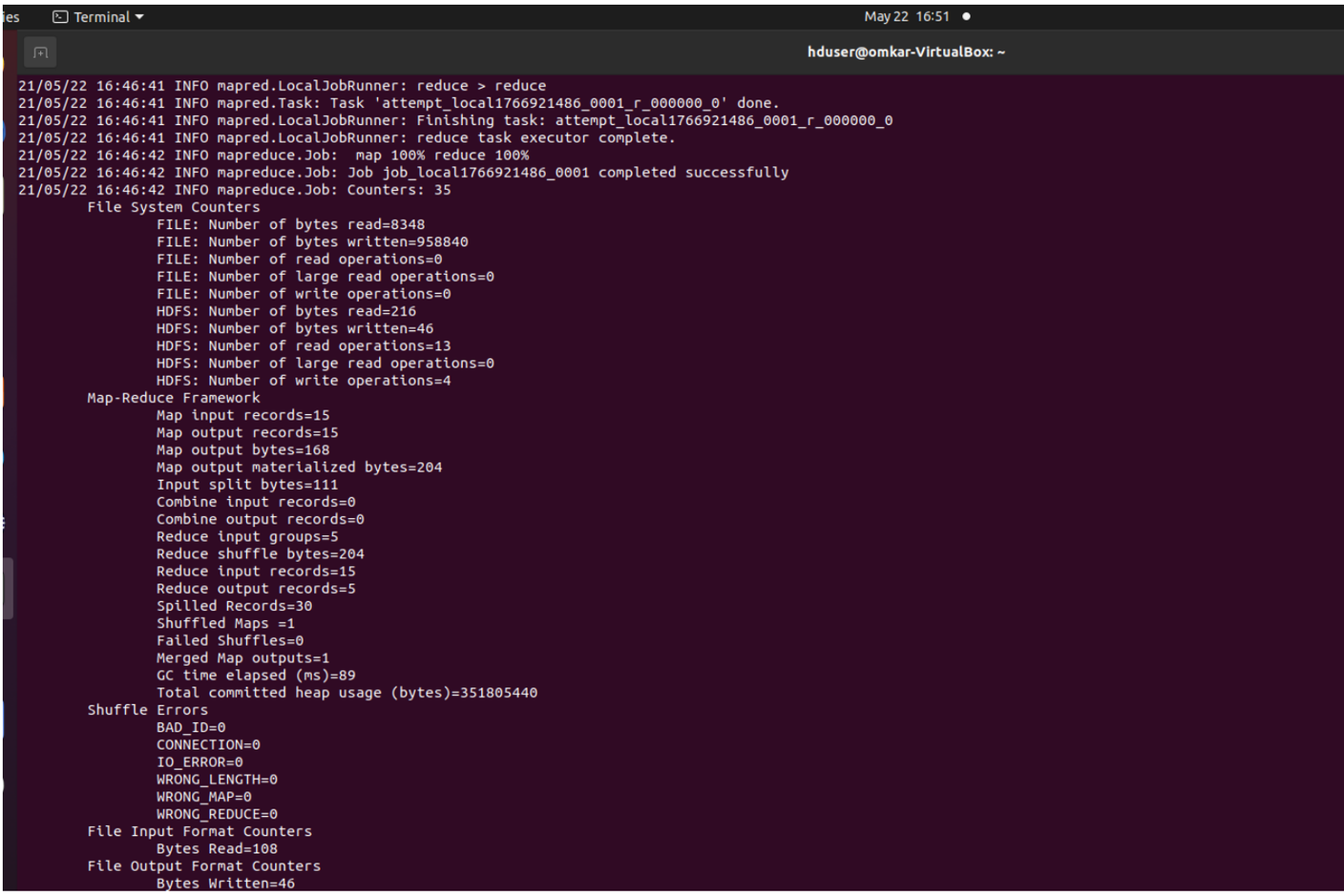
```
package words;

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.mapreduce.Mapper;

public class WordMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

       public void map(LongWritable key, Text value, Context con) throws IOException,
InterruptedException{

              String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);

        while (tokenizer.hasMoreTokens()) {
              con.write(new Text((tokenizer.nextToken())),new IntWritable(1));
        }
       }
}
```

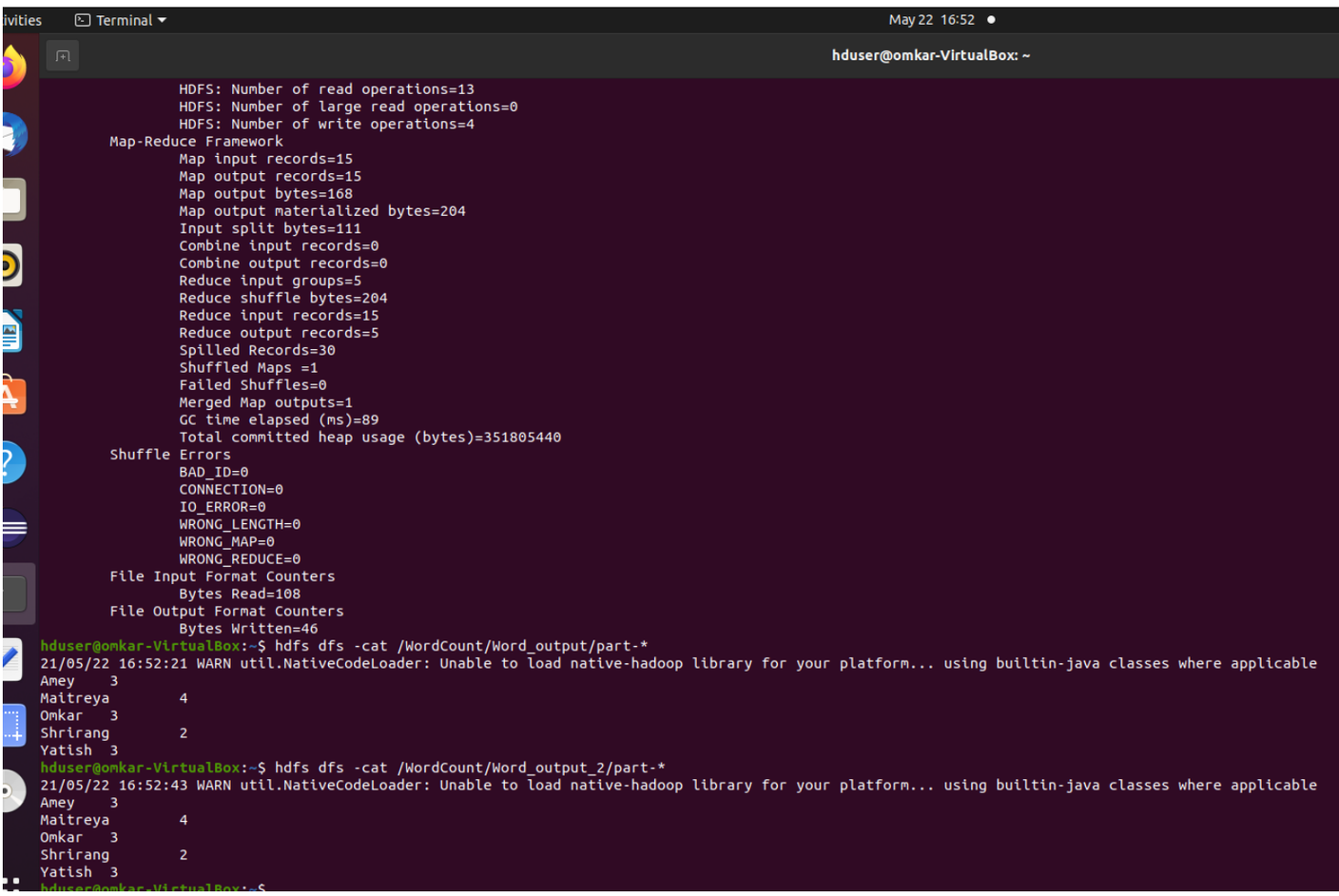
Reducer class : WordReducer.java

```
package words;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class WordReducer extends Reducer<Text,IntWritable,Text,IntWritable> {

          public void reduce(Text word, Iterable<IntWritable> values, Context con)
throws IOException, InterruptedException {

                        int sum=0;

                            for(IntWritable value : values)
                            {
                                sum += value.get();
                            }

                            con.write(word, new IntWritable(sum));
                }
  }
```

Output Screenshots




Terminal
May 22 16:51
hduser@omkar-VirtualBox: ~
21/05/22 16:46:41 INFO mapred.LocalJobRunner: reduce > reduce
21/05/22 16:46:41 INFO mapred.Task: Task 'attempt_local1766921486_0001_r_000000_0' done.
21/05/22 16:46:41 INFO mapred.LocalJobRunner: Finishing task: attempt_local1766921486_0001_r_000000_0
21/05/22 16:46:41 INFO mapred.LocalJobRunner: reduce task executor complete.
21/05/22 16:46:42 INFO mapreduce.Job: map 100% reduce 100%
21/05/22 16:46:42 INFO mapreduce.Job: Job job_local1766921486_0001 completed successfully
21/05/22 16:46:42 INFO mapreduce.Job: Counters: 35
File System Counters
FILE: Number of bytes read=8348
FILE: Number of bytes written=958840
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=216
HDFS: Number of bytes written=46
HDFS: Number of read operations=13
HDFS: Number of large read operations=0
HDFS: Number of write operations=4
Map-Reduce Framework
Map input records=15
Map output records=15
Map output bytes=168
Map output materialized bytes=204
Input split bytes=111
Combine input records=0
Combine output records=0
Reduce input groups=5
Reduce shuffle bytes=204
Reduce input records=15
Reduce output records=5
Spilled Records=30
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=89
Total committed heap usage (bytes)=351805440
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=108
File Output Format Counters
Bytes Written=46


Terminal
May 22 16:52
hduser@omkar-VirtualBox: ~
HDFS: Number of read operations=13
HDFS: Number of large read operations=0
HDFS: Number of write operations=4
Map-Reduce Framework
Map input records=15
Map output records=15
Map output bytes=168
Map output materialized bytes=204
Input split bytes=111
Combine input records=0
Combine output records=0
Reduce input groups=5
Reduce shuffle bytes=204
Reduce input records=15
Reduce output records=5
Spilled Records=30
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=89
Total committed heap usage (bytes)=351805440
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=108
File Output Format Counters
Bytes Written=46
hduser@omkar-VirtualBox:~$ hdfs dfs -cat /WordCount/Word_output/part-*
21/05/22 16:52:21 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Amey 3
Maitreya 4
Omkar 3
Shrirang 2
Yatish 3
hduser@omkar-VirtualBox:~$ hdfs dfs -cat /WordCount/Word_output_2/part-*
21/05/22 16:52:43 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Amey 3
Maitreya 4
Omkar 3
Shrirang 2
Yatish 3