PRACTICAL 2 - Implementing Map-Reduce Program for Word Count problem.

SERVICES TO START-

HDFS

HBASE

If name name error occurs , restart HDFS

Steps to Perform the Practical:

Step 1: Open Cloudera

- Start your Cloudera environment and log in.

Step 2: Go to Eclipse

- Launch Eclipse from the Cloudera environment.

Step 3: Create a new Java Project

- Go to `File > New > Java Project`.

Step 4: Name the Project and Configure Libraries

- Give the project a name, for example, "WordCount."
- Click "Next" and navigate to the "Libraries" tab.
- Click "Add External JARs."

Step 5: Add Hadoop JARs

- Navigate to the Hadoop library directory, typically located at `/usr/lib/hadoop`.
- Select all the JAR files and click "OK."

Step 6: Add Hadoop Client JARs

- Click "Add External JARs" again.
- Navigate to `/usr/lib/hadoop/client` and select all the JAR files.
- Click "OK."

Step 7: Add Hadoop Client 0.20 JARs

- Once more, click "Add External JARs."
- Navigate to `/usr/lib/hadoop/client0.20` and select all the JAR files.
- Click "OK."

Step 8: Finish Project Configuration

- Click "Finish" to complete the project setup.

Step 9: Create a New Java Class

- Under the newly created Java project, right-click on `src > New > Class`.
- Give the class the same name as the project, e.g., "WordCount," and click "Finish."

Step 10: Write Word Count Code

- Write the Java code for WordCount in the newly created class.

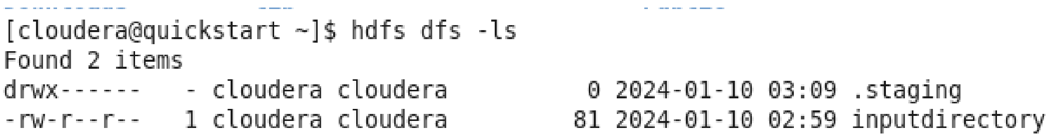
```
import java.io.IOException;
import java.util.StringTokenizer;

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

public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

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

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

- Save the file using `Ctrl + S`.

Step 11: Navigate to the project name, right-click on it, then choose "Export" > "Java" > "JAR f
Click "Next," set the path for the JAR file (e.g., cloudera), and provide a name matching the pro
Finally, click "Finish" to complete the export process.

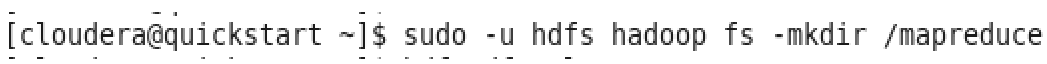
OPEN CLOUDERA TERMINAL

Step 1: Check existing files in HDFS



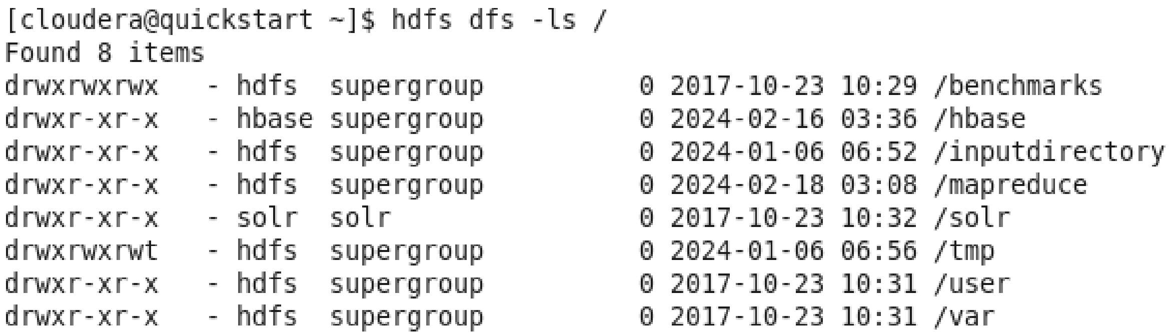
```
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 2 items
drwx------   - cloudera cloudera          0 2024-01-10 03:09 .staging
-rw-r--r--   1 cloudera cloudera         81 2024-01-10 02:59 inputdirectory
```

Step 2: Create a directory /mapreduce in HDFS



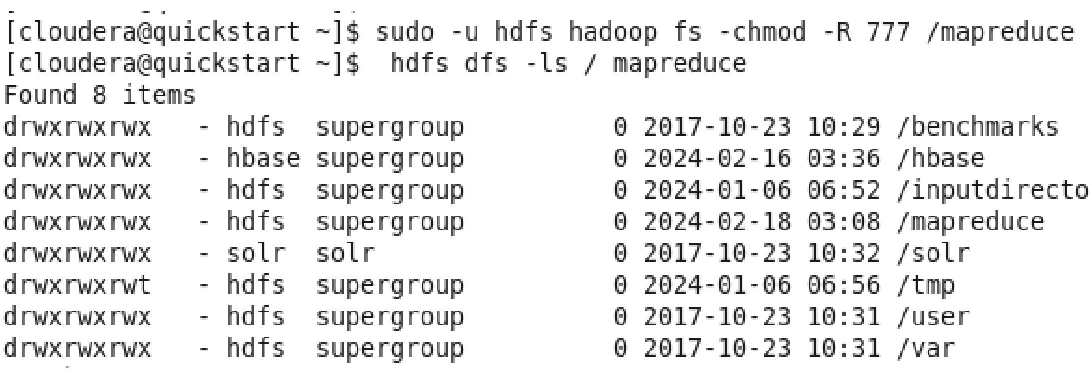
```
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -mkdir /mapreduce
```

Step 3: Check the contents of the root directory in HDFS



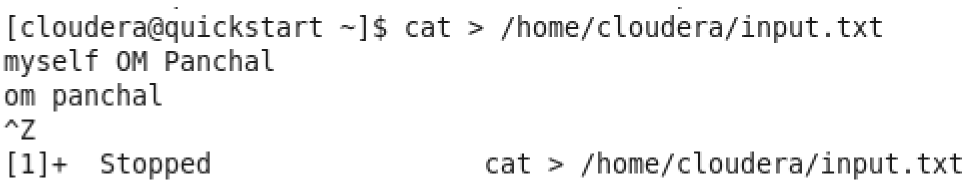
```
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 8 items
drwxrwxrwx   - hdfs  supergroup          0 2017-10-23 10:29 /benchmarks
drwxr-xr-x   - hbase supergroup          0 2024-02-16 03:36 /hbase
drwxr-xr-x   - hdfs  supergroup          0 2024-01-06 06:52 /inputdirectory
drwxr-xr-x   - hdfs  supergroup          0 2024-02-18 03:08 /mapreduce
drwxr-xr-x   - solr  solr                0 2017-10-23 10:32 /solr
drwxrwxrwt   - hdfs  supergroup          0 2024-01-06 06:56 /tmp
drwxr-xr-x   - hdfs  supergroup          0 2017-10-23 10:31 /user
drwxr-xr-x   - hdfs  supergroup          0 2017-10-23 10:31 /var
```

Step 4: Change permissions of the /mapreduce directory to 777



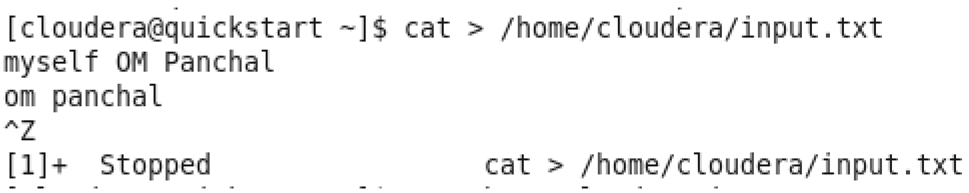
```
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -chmod -R 777 /mapreduce
[cloudera@quickstart ~]$  hdfs dfs -ls / mapreduce
Found 8 items
drwxrwxrwx   - hdfs  supergroup          0 2017-10-23 10:29 /benchmarks
drwxrwxrwx   - hbase supergroup          0 2024-02-16 03:36 /hbase
drwxrwxrwx   - hdfs  supergroup          0 2024-01-06 06:52 /inputdirectory
drwxrwxrwx   - hdfs  supergroup          0 2024-02-18 03:08 /mapreduce
drwxrwxrwx   - solr  solr                0 2017-10-23 10:32 /solr
drwxrwxrwt   - hdfs  supergroup          0 2024-01-06 06:56 /tmp
drwxrwxrwx   - hdfs  supergroup          0 2017-10-23 10:31 /user
drwxrwxrwx   - hdfs  supergroup          0 2017-10-23 10:31 /var
```

Step 5: Create a local file /home/cloudera/input.txt



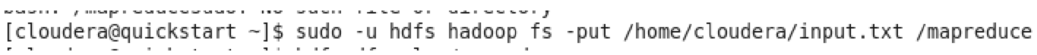
```
[cloudera@quickstart ~]$ cat > /home/cloudera/input.txt
myself OM Panchal
om panchal
^Z
[1]+  Stopped                 cat > /home/cloudera/input.txt
```

Step 6: Verify the content of the local file



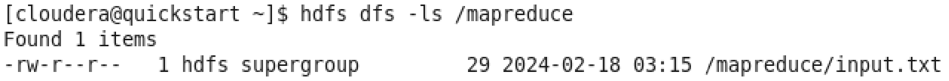
```
[cloudera@quickstart ~]$ cat > /home/cloudera/input.txt
myself OM Panchal
om panchal
^Z
[1]+  Stopped                 cat > /home/cloudera/input.txt
```

Step 7: Upload the local file to HDFS under the directory /mapreduce



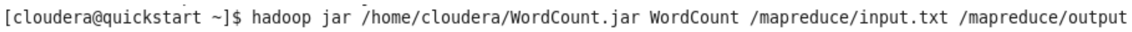
```
bash: /mapreducesudo: No such file or directory
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -put /home/cloudera/input.txt /mapreduce
```

Step 8: Verify the uploaded file in HDFS

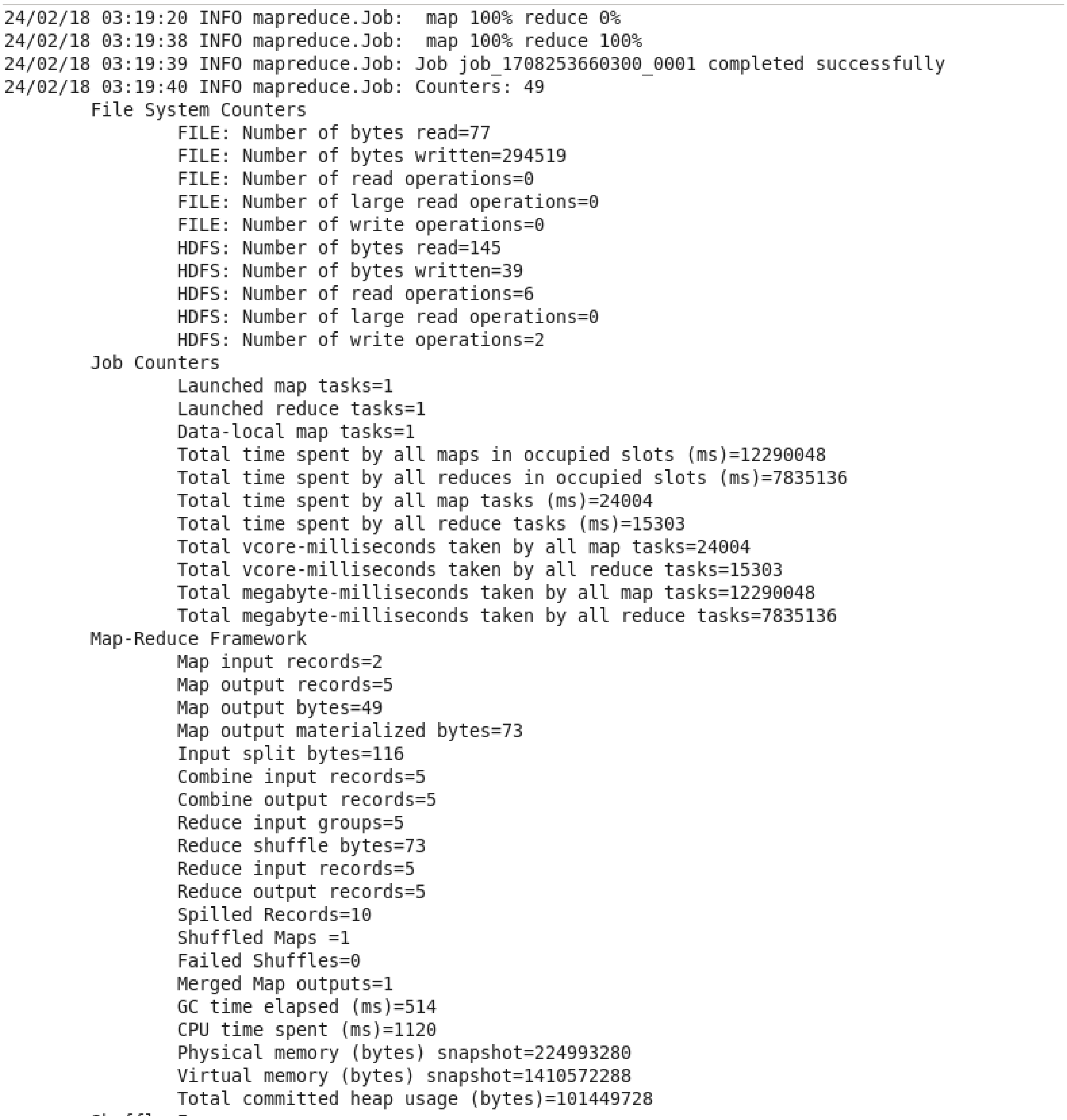


```
[cloudera@quickstart ~]$ hdfs dfs -ls /mapreduce
Found 1 items
-rw-r--r--   1 hdfs supergroup         29 2024-02-18 03:15 /mapreduce/input.txt
```

Step 9: Run the WordCount job

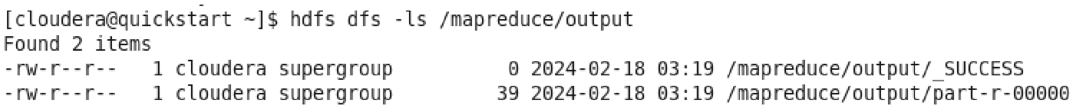


```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/WordCount.jar WordCount /mapreduce/input.txt /mapreduce/output
```



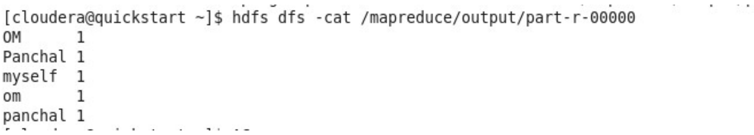
```
24/02/18 03:19:20 INFO mapreduce.Job:  map 100% reduce 0%
24/02/18 03:19:38 INFO mapreduce.Job:  map 100% reduce 100%
24/02/18 03:19:39 INFO mapreduce.Job: Job job_1708253660300_0001 completed successfully
24/02/18 03:19:40 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=77
                FILE: Number of bytes written=294519
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=145
                HDFS: Number of bytes written=39
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=12290048
                Total time spent by all reduces in occupied slots (ms)=7835136
                Total time spent by all map tasks (ms)=24004
                Total time spent by all reduce tasks (ms)=15303
                Total vcore-milliseconds taken by all map tasks=24004
                Total vcore-milliseconds taken by all reduce tasks=15303
                Total megabyte-milliseconds taken by all map tasks=12290048
                Total megabyte-milliseconds taken by all reduce tasks=7835136
        Map-Reduce Framework
                Map input records=2
                Map output records=5
                Map output bytes=49
                Map output materialized bytes=73
                Input split bytes=116
                Combine input records=5
                Combine output records=5
                Reduce input groups=5
                Reduce shuffle bytes=73
                Reduce input records=5
                Reduce output records=5
                Spilled Records=10
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=514
                CPU time spent (ms)=1120
                Physical memory (bytes) snapshot=224993280
                Virtual memory (bytes) snapshot=1410572288
                Total committed heap usage (bytes)=101449728
```

Step 10: Check the output in HDFS



```
[cloudera@quickstart ~]$ hdfs dfs -ls /mapreduce/output
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2024-02-18 03:19 /mapreduce/output/_SUCCESS
-rw-r--r--   1 cloudera supergroup         39 2024-02-18 03:19 /mapreduce/output/part-r-00000
```

Step 11: View the contents of the output file



```
[cloudera@quickstart ~]$ hdfs dfs -cat /mapreduce/output/part-r-00000
OM      1
Panchal 1
myself  1
om      1
panchal 1
```