

Homework-1 [Big Data Programming]

Varaprasad Kurra

Panther ID:002430487

1. Source Code

```
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 HadoopWordCount
{
    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);
        }
    }
}
```

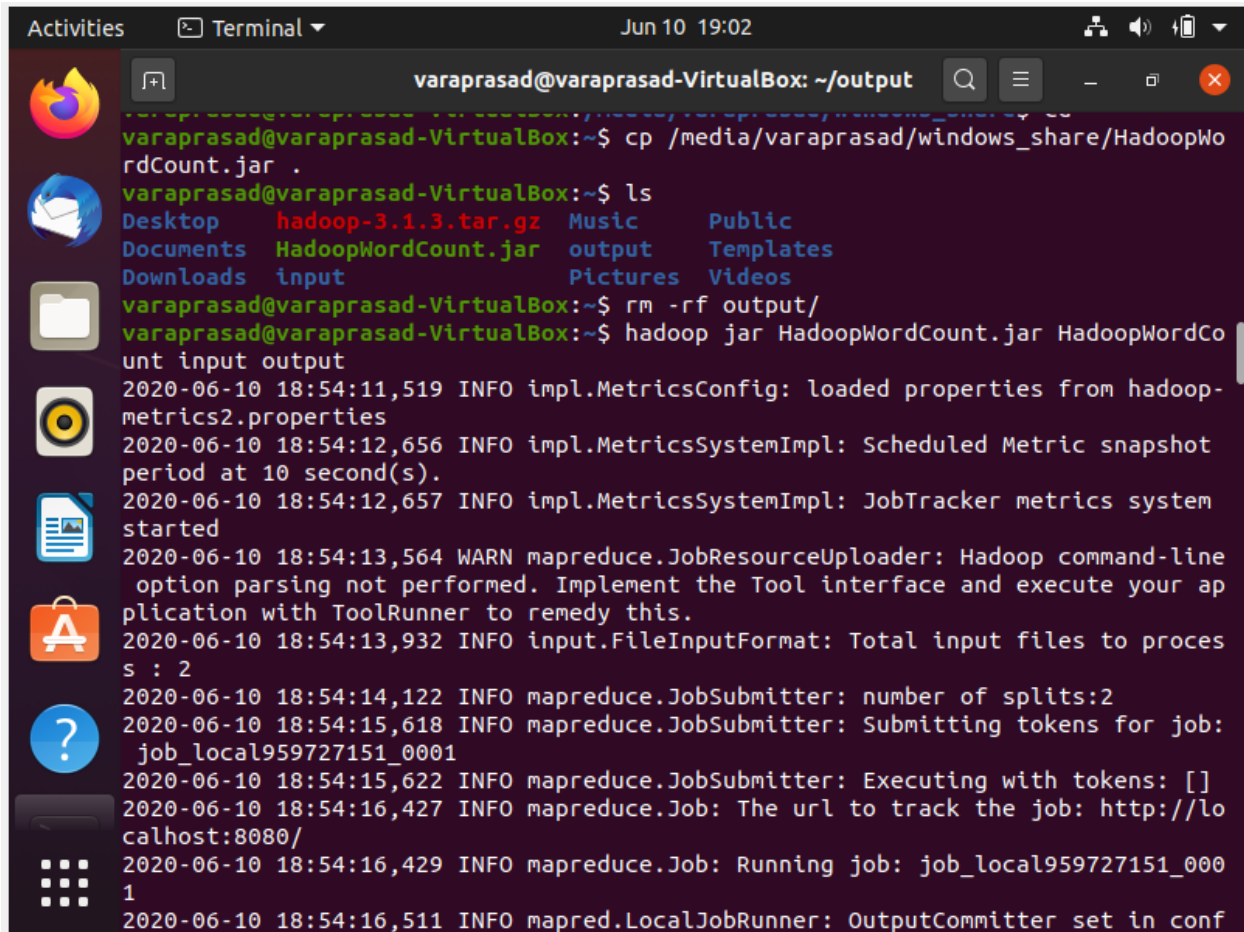
```

}

public static void main(String[] args) throws Exception
{
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(HadoopWordCount.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);
}
}

```

2. Screen shot showing command used to start a job



The screenshot shows a terminal window titled "varaprasad@varaprasad-VirtualBox: ~/output". The user has executed several commands to set up and run a Hadoop job:

- `cp /media/varaprasad/windows_share/HadoopWordCount.jar .`
- `ls` (listing files: Desktop, hadoop-3.1.3.tar.gz, Music, Public, Documents, HadoopWordCount.jar, output, Templates, Downloads, input, Pictures, Videos)
- `rm -rf output/`
- `hadoop jar HadoopWordCount.jar HadoopWordCount input output`

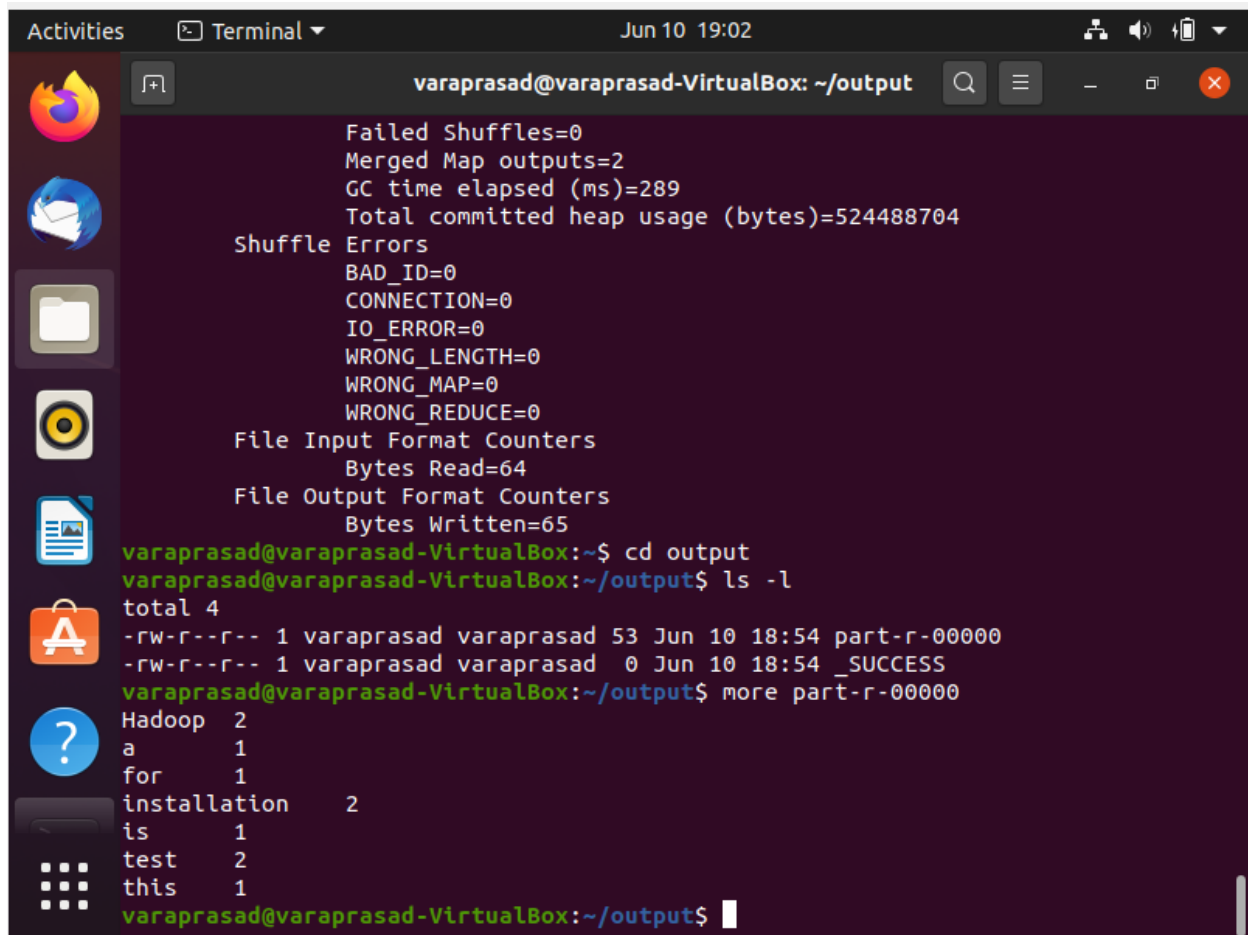
The output shows the Hadoop job execution progress, including logging of metrics, warnings about command-line option parsing, and status updates from the JobTracker and LocalJobRunner.

```

varaprasad@varaprasad-VirtualBox:~/output$ cp /media/varaprasad/windows_share/HadoopWordCount.jar .
varaprasad@varaprasad-VirtualBox:~$ ls
Desktop      hadoop-3.1.3.tar.gz  Music      Public
Documents    HadoopWordCount.jar  output     Templates
Downloads     input                Pictures    Videos
varaprasad@varaprasad-VirtualBox:~$ rm -rf output/
varaprasad@varaprasad-VirtualBox:~$ hadoop jar HadoopWordCount.jar HadoopWordCount input output
2020-06-10 18:54:11,519 INFO impl.MetricsConfig: loaded properties from hadoop-metrics2.properties
2020-06-10 18:54:12,656 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2020-06-10 18:54:12,657 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2020-06-10 18:54:13,564 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2020-06-10 18:54:13,932 INFO input.FileInputFormat: Total input files to process : 2
2020-06-10 18:54:14,122 INFO mapreduce.JobSubmitter: number of splits:2
2020-06-10 18:54:15,618 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local959727151_0001
2020-06-10 18:54:15,622 INFO mapreduce.JobSubmitter: Executing with tokens: []
2020-06-10 18:54:16,427 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2020-06-10 18:54:16,429 INFO mapreduce.Job: Running job: job_local959727151_0001
2020-06-10 18:54:16,511 INFO mapred.LocalJobRunner: OutputCommitter set in conf

```

3. Screenshot showing successful completion and Result



The screenshot shows a terminal window titled "varaprasad@varaprasad-VirtualBox: ~/output" with a search bar and window controls. The terminal output displays Hadoop job statistics: Failed Shuffles=0, Merged Map outputs=2, GC time elapsed (ms)=289, and Total committed heap usage (bytes)=524488704. It lists Shuffle Errors (BAD_ID=0, CONNECTION=0, IO_ERROR=0, WRONG_LENGTH=0, WRONG_MAP=0, WRONG_REDUCE=0) and File Input/Output Format Counters (Bytes Read=64, Bytes Written=65). The user then navigates to the output directory and lists files, showing "part-r-00000" and "_SUCCESS". Finally, the user uses the "more" command to view the contents of "part-r-00000", which lists Hadoop-related terms and their counts: Hadoop (2), a (1), for (1), installation (2), is (1), test (2), and this (1).

```
Failed Shuffles=0
Merged Map outputs=2
GC time elapsed (ms)=289
Total committed heap usage (bytes)=524488704
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=64
File Output Format Counters
Bytes Written=65
varaprasad@varaprasad-VirtualBox:~$ cd output
varaprasad@varaprasad-VirtualBox:~/output$ ls -l
total 4
-rw-r--r-- 1 varaprasad varaprasad 53 Jun 10 18:54 part-r-00000
-rw-r--r-- 1 varaprasad varaprasad  0 Jun 10 18:54 _SUCCESS
varaprasad@varaprasad-VirtualBox:~/output$ more part-r-00000
Hadoop 2
a      1
for    1
installation 2
is     1
test   2
this   1
varaprasad@varaprasad-VirtualBox:~/output$
```