SMIT R PATEL
19162121031
SEM 5
BIG DATA AND ANALYTICS

PRACTICAL 5

Given below is the data regarding the electrical consumption of an organization. It contains the monthly electrical consumption and the annual average for various years.

	Jan	**Feb**	**Mar**	**Apr**	**May**	**Jun**	**Jul**	**Aug**	**Sep**	**Oct**	**Nov**	**Dec**	**Avg**
1979	23	23	2	43	24	25	26	26	26	26	25	26	25
1980	26	27	28	28	28	30	31	31	31	30	30	30	29
1981	31	32	32	32	33	34	35	36	36	34	34	34	34
1984	39	38	39	39	39	41	42	43	40	39	38	38	40
1985	38	39	39	39	39	41	41	41	00	40	39	39	45

If the above data is given as input, we have to write applications to process it and produce results such as finding the year of maximum usage, year of minimum usage, and so on. This is a walkover for the programmers with finite number of records. They will simply write the logic to produce the required output, and pass the data to the application written.

But, think of the data representing the electrical consumption of all the largescale industries of a particular state, since its formation.

When we write applications to process such bulk data,

• They will take a lot of time to execute.

• There will be a heavy network traffic when we move data from source to network server and so on.

To solve these problems, we have the MapReduce framework.

Input Data

The above data is saved as **sample.txt** and given as input. The input file looks as shown below.

```
1979 23 23 2 43 24 25 26 26 26 26 25 26 25
1980 26 27 28 28 28 30 31 31 31 30 30 30 29
1981 31 32 32 32 33 34 35 36 36 34 34 34 34
1984 39 38 39 39 39 41 42 43 40 39 38 38 40
1985 38 39 39 39 39 41 41 41 00 40 39 39 45
```

Example Program

Given below is the program to the sample data using MapReduce framework.

```
package hadoop;


import java.util.*;

import java.io.IOException;
import java.io.IOException;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class ProcessUnits {
 //Mapper class
 public static class E_EMapper extends MapReduceBase implements
Mapper<LongWritable ,/*Input key Type */
 Text, /*Input value Type*/
 Text, /*Output key Type*/
 IntWritable> /*Output value Type*/
 {
 //Map function
 public void map(LongWritable key, Text value,
OutputCollector<Text, IntWritable> output,
 Reporter reporter) throws IOException {
 String line = value.toString();
 String lasttoken = null;
 StringTokenizer s = new StringTokenizer(line,"\t");  String
year = s.nextToken();

 while(s.hasMoreTokens()) {
 lasttoken = s.nextToken();
 }
 int avgprice = Integer.parseInt(lasttoken);
output.collect(new Text(year), new
IntWritable(avgprice));
 }
 }
 //Reducer class
 public static class E_EReduce extends MapReduceBase implements
Reducer< Text, IntWritable, Text, IntWritable > {
 //Reduce function
 public void reduce( Text key, Iterator <IntWritable>
values,
 OutputCollector<Text, IntWritable> output, Reporter
reporter) throws IOException {
 int maxavg = 30;
 int val = Integer.MIN_VALUE;

 while (values.hasNext()) {
 if((val = values.next().get())>maxavg) {
output.collect(key, new IntWritable(val));   }
 }
 }
 }

 //Main function
 public static void main(String args[])throws Exception {
JobConf conf = new JobConf(ProcessUnits.class);
 conf.setJobName("max_eletricityunits");
 conf.setOutputKeyClass(Text.class);
 conf.setOutputValueClass(IntWritable.class);
conf.setMapperClass(E_EMapper.class);
 conf.setCombinerClass(E_EReduce.class);
 conf.setReducerClass(E_EReduce.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);
 }
}
```

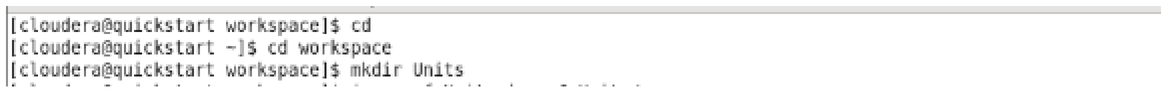
Save the above program as **ProcessUnits.java.** The compilation and execution of the program is explained below.

Compilation and Execution of Process Units Program

Let us assume we are in the home directory of a Hadoop user (e.g. /home/hadoop). Follow the steps given below to compile and execute the above program.

Step 1

The following command is to create a directory to store the compiled java classes. `$ mkdir units`



```
[cloudera@quickstart workspace]$ cd
[cloudera@quickstart ~]$ cd workspace
[cloudera@quickstart workspace]$ mkdir Units
```

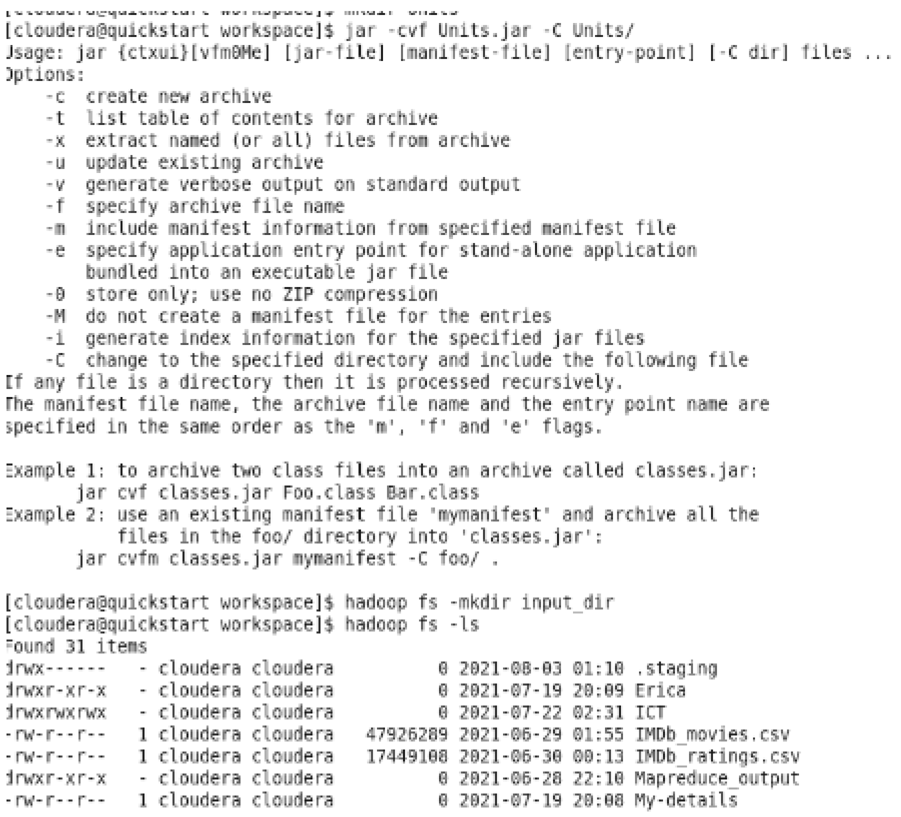
Step 2

Download **Hadoop-core-1.2.1.jar,** which is used to compile and execute the MapReduce program. Visit the following link mvnrepository.com to download the jar. Let us assume the downloaded folder is **/home/hadoop/.**

Step 3

The following commands are used for compiling the **ProcessUnits.java** program and creating a jar for the program.

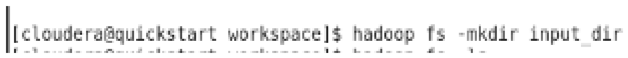
```
$ javac -classpath hadoop-core-1.2.1.jar -d units
ProcessUnits.java
$ jar -cvf units.jar -C units/ .
```



```
[cloudera@quickstart workspace]$ jar -cvf Units.jar -C Units/
Jsage: jar {ctxui}[vfm0Me] [jar-file] [manifest-file] [entry-point] [-C dir] files ...
Dptions:
    -c  create new archive
    -t  list table of contents for archive
    -x  extract named (or all) files from archive
    -u  update existing archive
    -v  generate verbose output on standard output
    -f  specify archive file name
    -m  include manifest information from specified manifest file
    -e  specify application entry point for stand-alone application
        bundled into an executable jar file
    -0  store only; use no ZIP compression
    -M  do not create a manifest file for the entries
    -i  generate index information for the specified jar files
    -C  change to the specified directory and include the following file
If any file is a directory then it is processed recursively.
The manifest file name, the archive file name and the entry point name are
specified in the same order as the 'm', 'f' and 'e' flags.

Example 1: to archive two class files into an archive called classes.jar:
       jar cvf classes.jar Foo.class Bar.class
Example 2: use an existing manifest file 'mymanifest' and archive all the
           files in the foo/ directory into 'classes.jar':
       jar cvfm classes.jar mymanifest -C foo/ .

[cloudera@quickstart workspace]$ hadoop fs -mkdir input_dir
[cloudera@quickstart workspace]$ hadoop fs -ls
Found 31 items
drwx------   - cloudera cloudera          0 2021-08-03 01:10 .staging
drwxr-xr-x   - cloudera cloudera          0 2021-07-19 20:09 Erica
drwxrwxrwx   - cloudera cloudera          0 2021-07-22 02:31 ICT
-rw-r--r--   1 cloudera cloudera   47926289 2021-06-29 01:55 IMDb_movies.csv
-rw-r--r--   1 cloudera cloudera   17449108 2021-06-30 00:13 IMDb_ratings.csv
drwxr-xr-x   - cloudera cloudera          0 2021-06-28 22:10 Mapreduce_output
-rw-r--r--   1 cloudera cloudera          0 2021-07-19 20:08 My-details
```

Step 4

The following command is used to create an input directory in HDFS. `$HADOOP_HOME/bin/hadoop fs -mkdir input_dir`

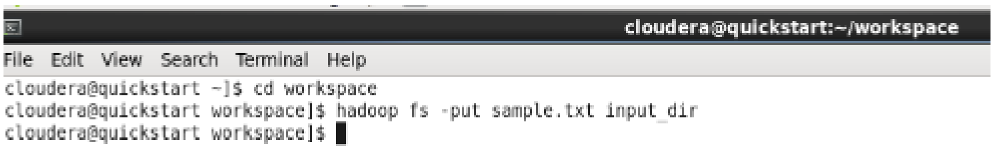


```
[cloudera@quickstart workspace]$ hadoop fs -mkdir input_dir
```

Step 5

The following command is used to copy the input file named **sample.txt**in the input directory of HDFS.

```
$HADOOP_HOME/bin/hadoop fs -put /home/hadoop/sample.txt input_dir
```

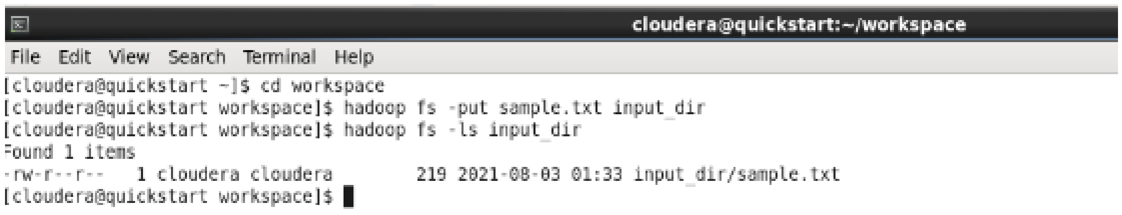


```
cloudera@quickstart ~]$ cd workspace
cloudera@quickstart workspace]$ hadoop fs -put sample.txt input_dir
cloudera@quickstart workspace]$ 
```

Step 6

The following command is used to verify the files in the input

directory. `$HADOOP_HOME/bin/hadoop fs -ls input_dir/`



```
[cloudera@quickstart ~]$ cd workspace
[cloudera@quickstart workspace]$ hadoop fs -put sample.txt input_dir
[cloudera@quickstart workspace]$ hadoop fs -ls input_dir
Found 1 items
-rw-r--r--   1 cloudera cloudera        219 2021-08-03 01:33 input_dir/sample.txt
[cloudera@quickstart workspace]$ 
```

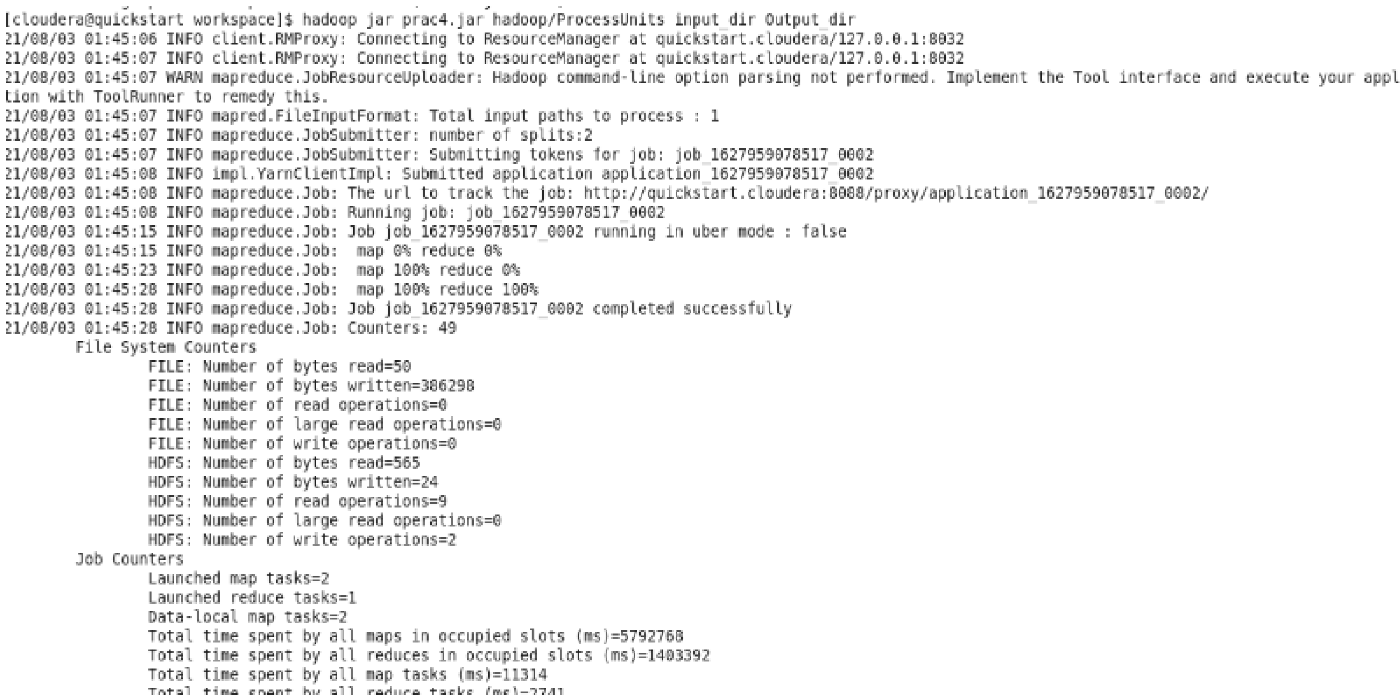
Step 7

The following command is used to run the Eleunit_max application by taking the input files from the input directory.

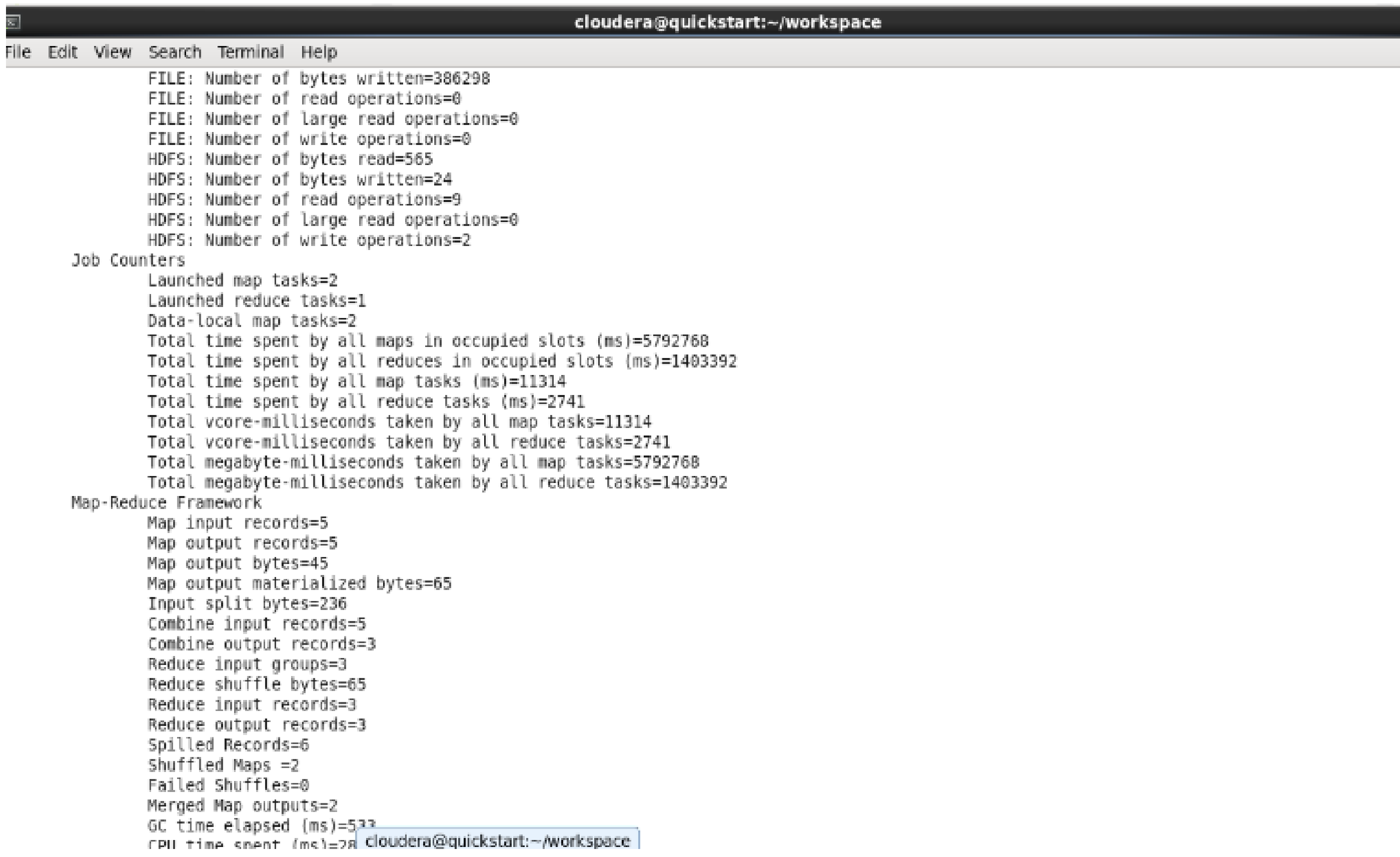
```
$HADOOP_HOME/bin/hadoop jar units.jar hadoop.ProcessUnits
input_dir output_dir
```

Wait for a while until the file is executed. After execution, as shown below, the output will contain the number of input splits, the number of Map tasks, the number of reducer tasks, etc.

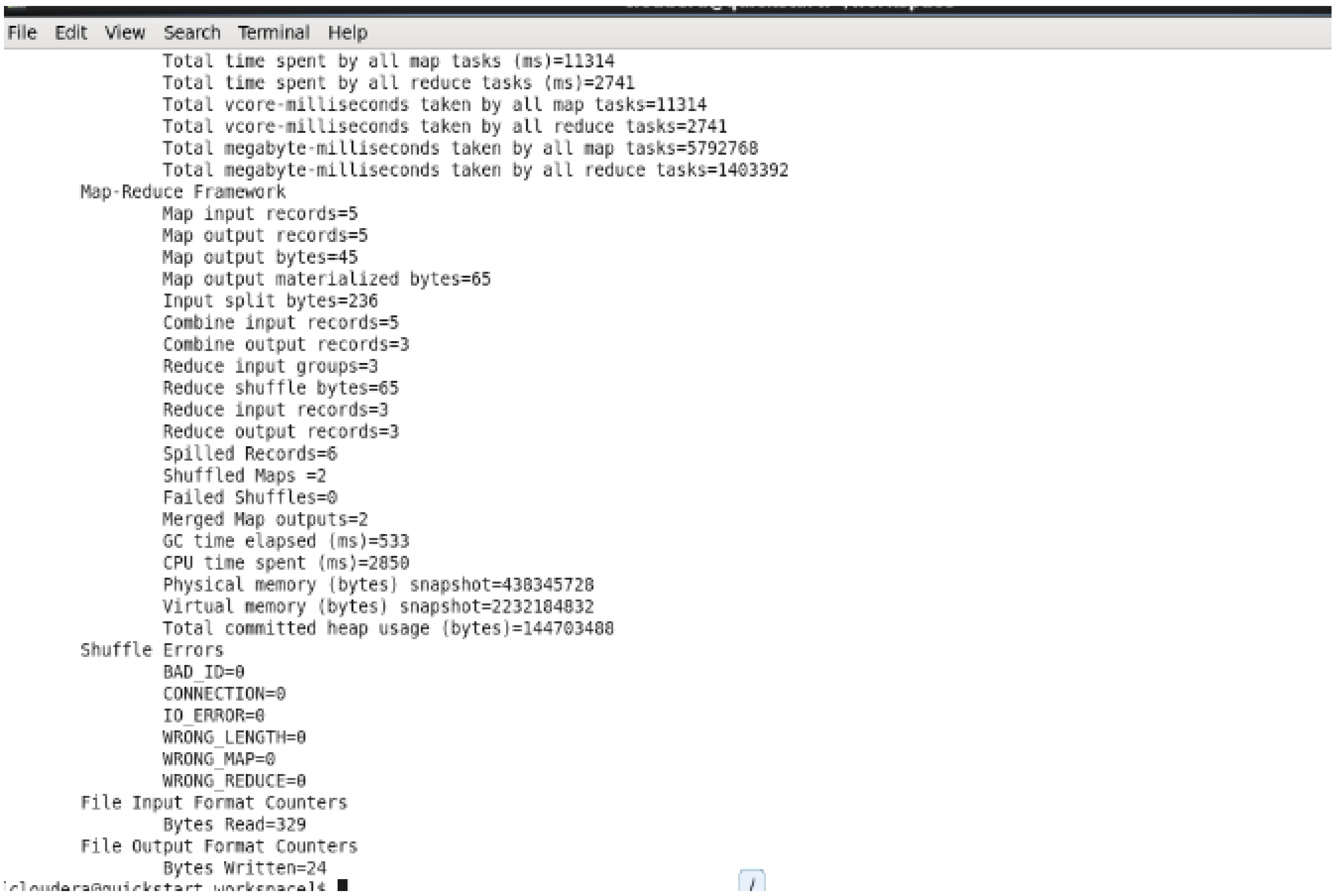
```
INFO mapreduce.Job: Job job_1414748220717_0002
completed successfully
14/10/31 06:02:52
INFO mapreduce.Job: Counters: 49
 File System Counters

FILE: Number of bytes read = 61
FILE: Number of bytes written = 279400
FILE: Number of read operations = 0
FILE: Number of large read operations = 0
FILE: Number of write operations = 0
HDFS: Number of bytes read = 546
HDFS: Number of bytes written = 40
HDFS: Number of read operations = 9
HDFS: Number of large read operations = 0
HDFS: Number of write operations = 2 Job Counters


 Launched map tasks = 2
 Launched reduce tasks = 1
 Data-local map tasks = 2
 Total time spent by all maps in occupied slots (ms) = 146137
Total time spent by all reduces in occupied slots (ms) = 441
Total time spent by all map tasks (ms) = 14613   Total time
spent by all reduce tasks (ms) = 44120   Total vcore-seconds
taken by all map tasks = 146137   Total vcore-seconds taken by
all reduce tasks = 44120   Total megabyte-seconds taken by all
map tasks = 149644288   Total megabyte-seconds taken by all
reduce tasks = 45178880
Map-Reduce Framework

 Map input records = 5
 Map output records = 5
 Map output bytes = 45
 Map output materialized bytes = 67
 Input split bytes = 208
 Combine input records = 5
 Combine output records = 5
 Reduce input groups = 5
 Reduce shuffle bytes = 6
 Reduce input records = 5
 Reduce output records = 5
 Spilled Records = 10
 Shuffled Maps = 2
 Failed Shuffles = 0
 Merged Map outputs = 2
 GC time elapsed (ms) = 948
 CPU time spent (ms) = 5160
  Physical memory (bytes) snapshot = 47749120
 Virtual memory (bytes) snapshot = 2899349504
 Total committed heap usage (bytes) = 277684224

File Output Format Counters

 Bytes Written = 40
```



```
[cloudera@quickstart workspace]$ hadoop jar prac4.jar hadoop/ProcessUnits input_dir Output_dir
21/08/03 01:45:06 INFO client.RMProxy: Connecting to ResourceManager at quickstart.cloudera/127.0.0.1:8032
21/08/03 01:45:07 INFO client.RMProxy: Connecting to ResourceManager at quickstart.cloudera/127.0.0.1:8032
21/08/03 01:45:07 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your appl
tion with ToolRunner to remedy this.
21/08/03 01:45:07 INFO mapred.FileInputFormat: Total input paths to process : 1
21/08/03 01:45:07 INFO mapreduce.JobSubmitter: number of splits:2
21/08/03 01:45:07 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1627959078517_0002
21/08/03 01:45:08 INFO impl.YarnClientImpl: Submitted application application_1627959078517_0002
21/08/03 01:45:08 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1627959078517_0002/
21/08/03 01:45:08 INFO mapreduce.Job: Running job: job_1627959078517_0002
21/08/03 01:45:15 INFO mapreduce.Job: Job job_1627959078517_0002 running in uber mode : false
21/08/03 01:45:15 INFO mapreduce.Job:  map 0% reduce 0%
21/08/03 01:45:23 INFO mapreduce.Job:  map 100% reduce 0%
21/08/03 01:45:28 INFO mapreduce.Job:  map 100% reduce 100%
21/08/03 01:45:28 INFO mapreduce.Job: Job job_1627959078517_0002 completed successfully
21/08/03 01:45:28 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=50
                FILE: Number of bytes written=386298
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=565
                HDFS: Number of bytes written=24
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=5792768
                Total time spent by all reduces in occupied slots (ms)=1403392
                Total time spent by all map tasks (ms)=11314
                Total time spent by all reduce tasks (ms)=2741
```


cloudera@quickstart:~/workspace
File Edit View Search Terminal Help

```
                FILE: Number of bytes written=386298
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=565
                HDFS: Number of bytes written=24
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=5792768
                Total time spent by all reduces in occupied slots (ms)=1403392
                Total time spent by all map tasks (ms)=11314
                Total time spent by all reduce tasks (ms)=2741
                Total vcore-milliseconds taken by all map tasks=11314
                Total vcore-milliseconds taken by all reduce tasks=2741
                Total megabyte-milliseconds taken by all map tasks=5792768
                Total megabyte-milliseconds taken by all reduce tasks=1403392
        Map-Reduce Framework
                Map input records=5
                Map output records=5
                Map output bytes=45
                Map output materialized bytes=65
                Input split bytes=236
                Combine input records=5
                Combine output records=3
                Reduce input groups=3
                Reduce shuffle bytes=65
                Reduce input records=3
                Reduce output records=3
                Spilled Records=6
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=533
                CPU time spent (ms)=28
```

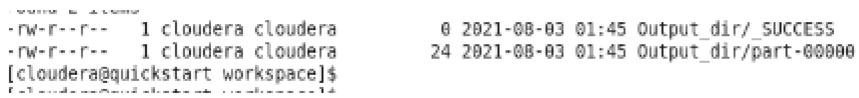

File Edit View Search Terminal Help

```
                Total time spent by all map tasks (ms)=11314
                Total time spent by all reduce tasks (ms)=2741
                Total vcore-milliseconds taken by all map tasks=11314
                Total vcore-milliseconds taken by all reduce tasks=2741
                Total megabyte-milliseconds taken by all map tasks=5792768
                Total megabyte-milliseconds taken by all reduce tasks=1403392
        Map-Reduce Framework
                Map input records=5
                Map output records=5
                Map output bytes=45
                Map output materialized bytes=65
                Input split bytes=236
                Combine input records=5
                Combine output records=3
                Reduce input groups=3
                Reduce shuffle bytes=65
                Reduce input records=3
                Reduce output records=3
                Spilled Records=6
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=533
                CPU time spent (ms)=2850
                Physical memory (bytes) snapshot=438345728
                Virtual memory (bytes) snapshot=2232184832
                Total committed heap usage (bytes)=144703488
        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=329
        File Output Format Counters
                Bytes Written=24
[cloudera@quickstart workspace]$
```

Step 8

The following command is used to verify the resultant files in the output folder.

```
$HADOOP_HOME/bin/hadoop fs -ls output_dir/
```



```
-rw-r--r--   1 cloudera cloudera          0 2021-08-03 01:45 Output_dir/_SUCCESS
-rw-r--r--   1 cloudera cloudera         24 2021-08-03 01:45 Output_dir/part-00000
[cloudera@quickstart workspace]$
```

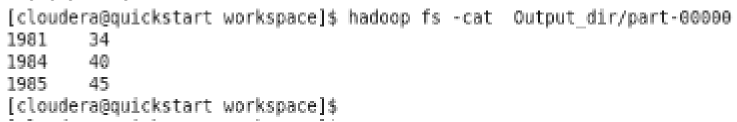
Step 9

The following command is used to see the output in **Part-00000** file. This file is generated by HDFS.

```
$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000
```

Below is the output generated by the MapReduce program.

```
1981 34
1984 40
1985 45
```



```
[cloudera@quickstart workspace]$ hadoop fs -cat  Output_dir/part-00000
1981    34
1984    40
1985    45
[cloudera@quickstart workspace]$
```

Step 10

The following command is used to copy the output folder from HDFS to the local file system for analyzing.

```
$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000/bin/hadoop
dfs get output_dir /home/hadoop
```