ASSIGNMENT– 03

Introduction to Distributed Systems - IS41243
Hadoop - MapReduce

By:
Sayanthiny.R
15APC2383

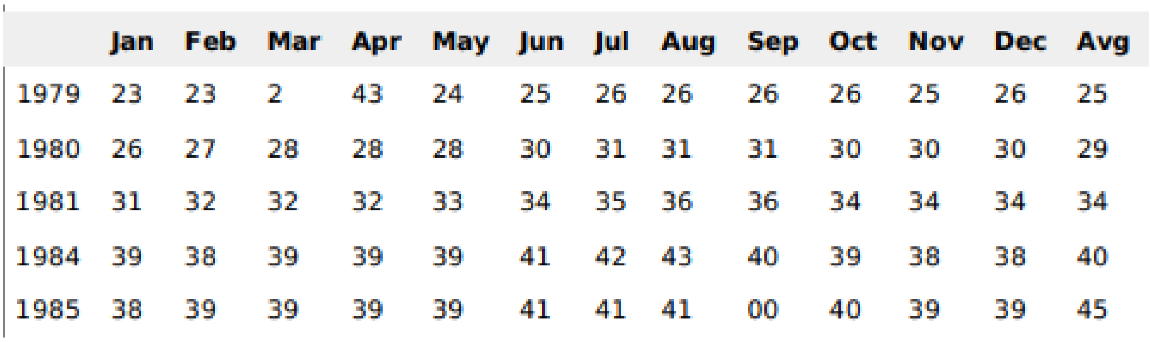
DEPARTMENT OF COMPUTING & INFORMATION SYSTEMS
FACULTY OF APPLIED SCIENCES
SABARAGAMUWA UNIVERSITY OF SRI LANKA

TABLE OF CONTENTS

Table of figures

Example Scenario

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

Figure 1 input data

First navigate to hadoop user (in my case hdoop) using command

```
$ Su - hdoop
```

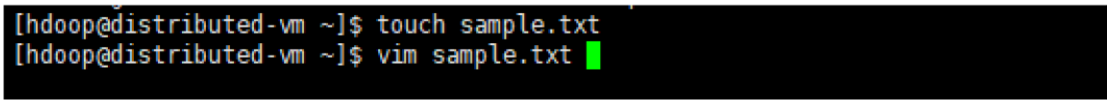
Input Data

Create sample.txt file.

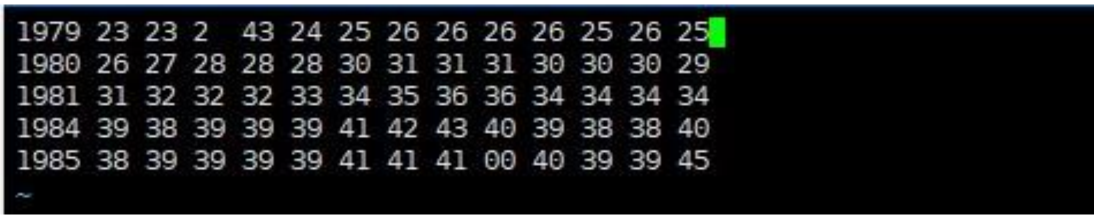
```
$ touch sample.txt
```

Then edit the sample.txt file using this command

```
$ nano sample.txt
```



```
[hdoop@distributed-vm ~]$ touch sample.txt
[hdoop@distributed-vm ~]$ vim sample.txt
```



```
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
~
```

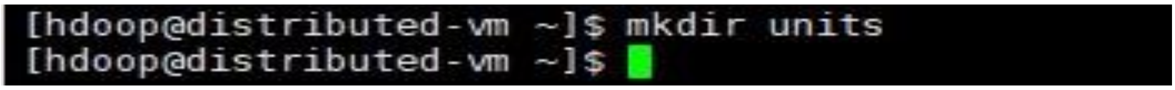
Figure 2 craete input data

Compilation and Execution of Process Units Program

Step 1: establish a directory to store the compiled Java classes

To establish a directory to store the compiled Java classes, use the command below.

```
$ mkdir units
```



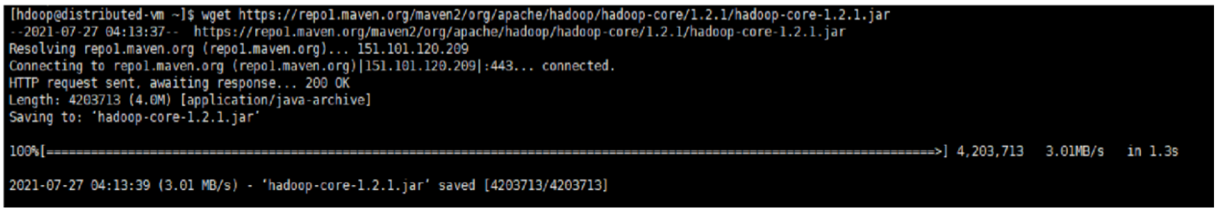
```
[hdoop@distributed-vm ~]$ mkdir units
[hdoop@distributed-vm ~]$ 
```

Figure 1: Create unit directory

Step 2: obtain the Hadoop-core-.2.1.jar file

To obtain the Hadoop-core-.2.1.jar file, go to the following website. The MapReduce software is compiled and executed here.https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-core/1.2.1/hadoop-core-1.2.1.jar

```
$ wget https://repo1.maven.org/maven2/org/apache/hadoop/hadoopcore/1.2.1/hadoop-
core 1.2.1.jar
```



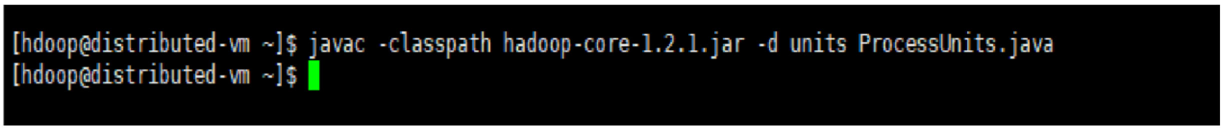
```
[hdoop@distributed-vm ~]$ wget https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-core/1.2.1/hadoop-core-1.2.1.jar
--2021-07-27 04:13:37--  https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-core/1.2.1/hadoop-core-1.2.1.jar
Resolving repo1.maven.org (repo1.maven.org)... 151.101.120.209
Connecting to repo1.maven.org (repo1.maven.org)|151.101.120.209|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4203713 (4.0M) [application/java-archive]
Saving to: ‘hadoop-core-1.2.1.jar’

100%[=============================================================================================================>] 4,203,713   3.01MB/s   in 1.3s

2021-07-27 04:13:39 (3.01 MB/s) - ‘hadoop-core-1.2.1.jar’ saved [4203713/4203713]
```

Figure 3 Download hadoop jar

Seeing if there are any problems or jar was successfully downloaded and working fine.

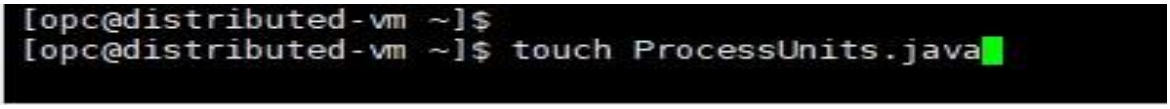


```
[hdoop@distributed-vm ~]$ javac -classpath hadoop-core-1.2.1.jar -d units ProcessUnits.java
[hdoop@distributed-vm ~]$ 
```

Make a file ProcessUnits.java

Make a file called ProcessUnits.java in Java.

```
  $ touch ProcessUnits.java
```

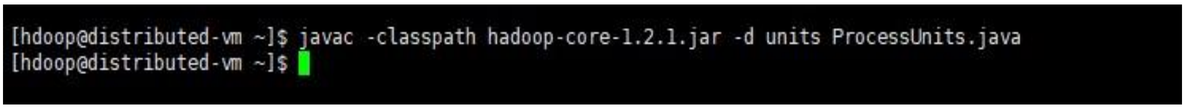


```
[opc@distributed-vm ~]$
[opc@distributed-vm ~]$ touch ProcessUnits.java
```

Figure 4 Make a file ProcessUnits.java

Step 3: Compiling the ProcessUnits.java program and creating a jar for the program.
The following commands are used for compiling the ProcessUnits.java program and creating a jar for the program.

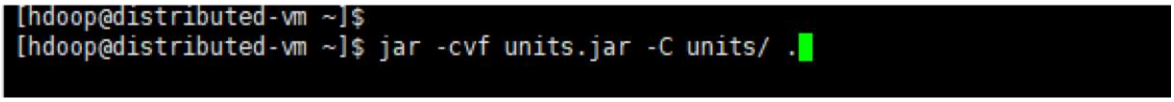
```
$ javac -classpath hadoop-core-1.2.1.jar -d units ProcessUnits.java
```



```
[hdoop@distributed-vm ~]$ javac -classpath hadoop-core-1.2.1.jar -d units ProcessUnits.java
[hdoop@distributed-vm ~]$
```

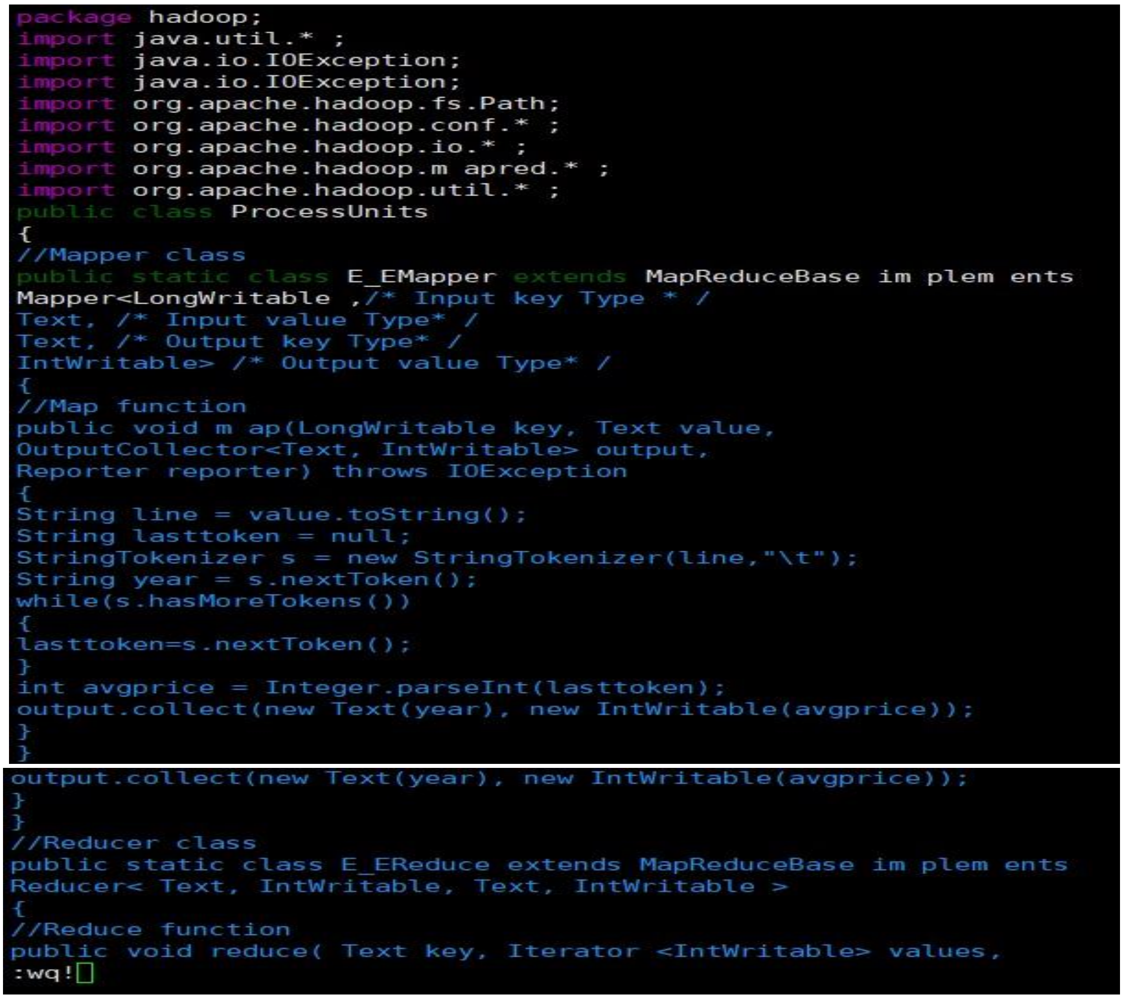
Figure 5 Compiling the ProcessUnits.java program

```
  $ jar -cvf units.jar -C units/ .
```



```
[hdoop@distributed-vm ~]$
[hdoop@distributed-vm ~]$ jar -cvf units.jar -C units/ .
```

Figure 6 creating a jar for the program



```
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.m apred.* ;
import org.apache.hadoop.util.* ;
public class ProcessUnits
{
//Mapper class
public static class E_EMapper extends MapReduceBase im plem ents
Mapper<LongWritable ,/* Input key Type * /
Text, /* Input value Type* /
Text, /* Output key Type* /
IntWritable> /* Output value Type* /
{
//Map function
public void m ap(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));
}
}
```

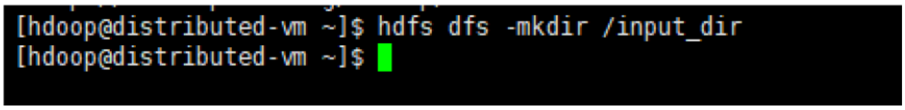
```
output.collect(new Text(year), new IntWritable(avgprice));
}
}
//Reducer class
public static class E_EReduce extends MapReduceBase im plem ents
Reducer< Text, IntWritable, Text, IntWritable >
{
//Reduce function
public void reduce( Text key, Iterator <IntWritable> values,
:wq!
```

Figure 7 program to the sample data using MapReduce framework.

Step 4: create an input directory in HDFS

Run the command below to create an input directory in HDFS.

```
$ hdfs dfs -mkdir /input_dir
```



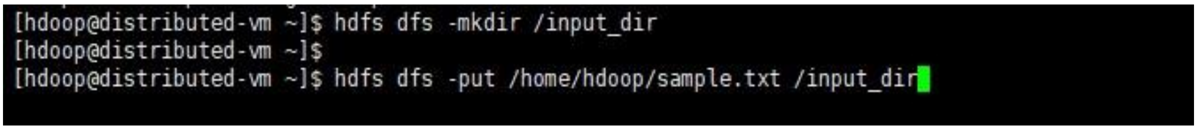
```
[hdoop@distributed-vm ~]$ hdfs dfs -mkdir /input_dir
[hdoop@distributed-vm ~]$ 
```

Figure 8 create an input directory in HDFS

Step 5: copy the input file named sample.txt and put in the input directory of HDFS.

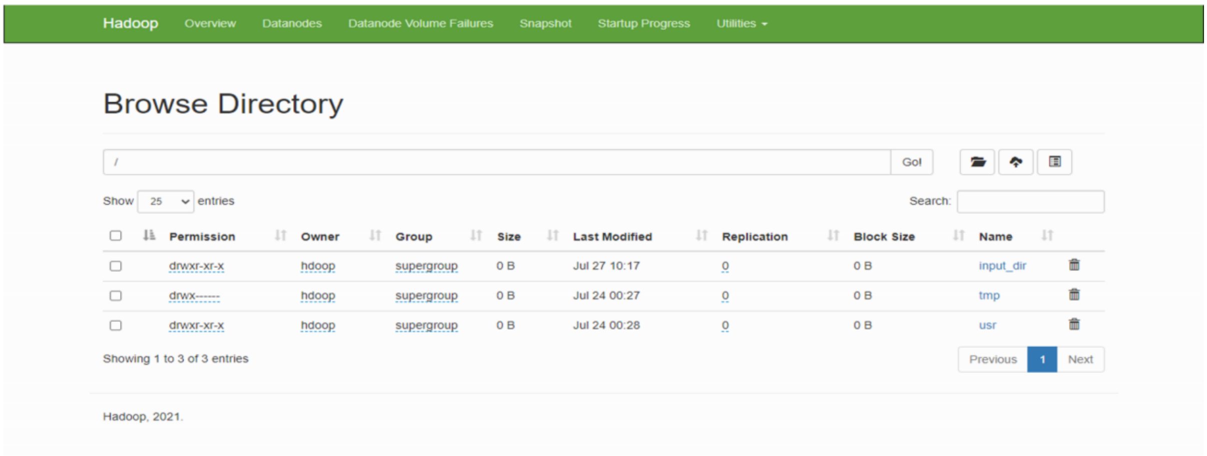
With the following command, copy the above-created input file to the HDFS input directory.

```
$ hdfs dfs -put /home/hdoop/sample.txt /input_dir
```



```
[hdoop@distributed-vm ~]$ hdfs dfs -mkdir /input_dir
[hdoop@distributed-vm ~]$
[hdoop@distributed-vm ~]$ hdfs dfs -put /home/hdoop/sample.txt /input_dir
```

Figure 9 copy the input file and put in the input directory of HDFS



Browse Directory

/

Show 25 entries

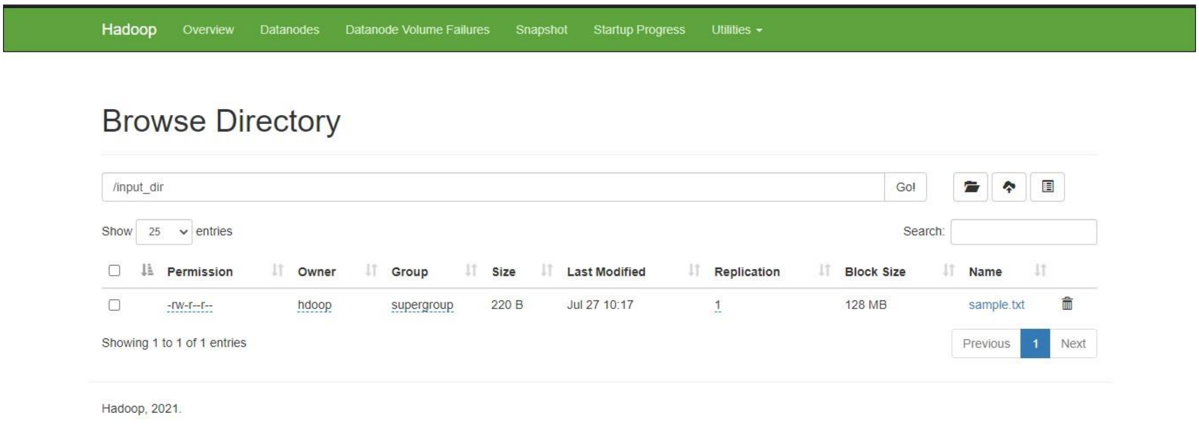
Search:

□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
□	drwxr-xr-x	hdoop	supergroup	0 B	Jul 27 10:17	0	0 B	input_dir
□	drwx------	hdoop	supergroup	0 B	Jul 24 00:27	0	0 B	tmp
□	drwxr-xr-x	hdoop	supergroup	0 B	Jul 24 00:28	0	0 B	usr

Showing 1 to 3 of 3 entries

Hadoop, 2021.

Figure input dir



Browse Directory

/input_dir

Show 25 entries

Search:

□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
□	-rw-r--r--	hdoop	supergroup	220 B	Jul 27 10:17	1	128 MB	sample.txt

Showing 1 to 1 of 1 entries

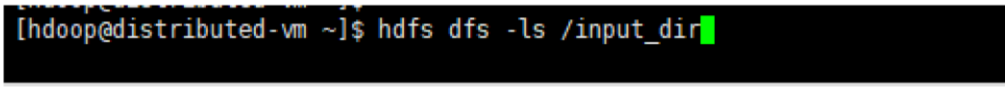
Hadoop, 2021.

Figure sample.txt inside /input.dir

Step 6: verify the files in the input directory

Use the command below to verify the files in the input directory.

```
$ hdfs dfs -ls /input_dir
```



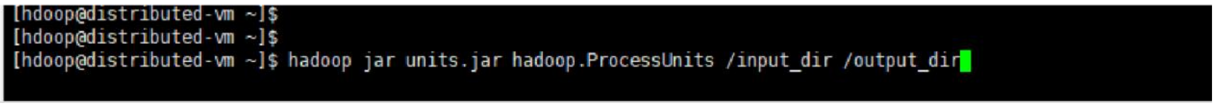
```
[hdoop@distributed-vm ~]$ hdfs dfs -ls /input_dir
```

Figure 14 verify the files in input directory

Step 7: run using the input files

Using the following command, the Eleunit max application is run using the input files from the input directory.

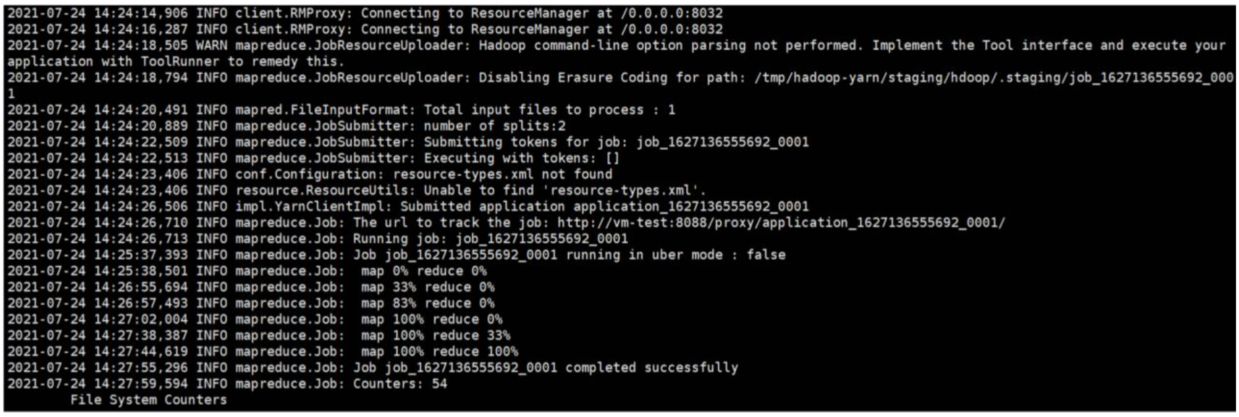
```
$ Hadoop jar units.jar Hadoop.ProcessUnits /input_dir /output_dir
```



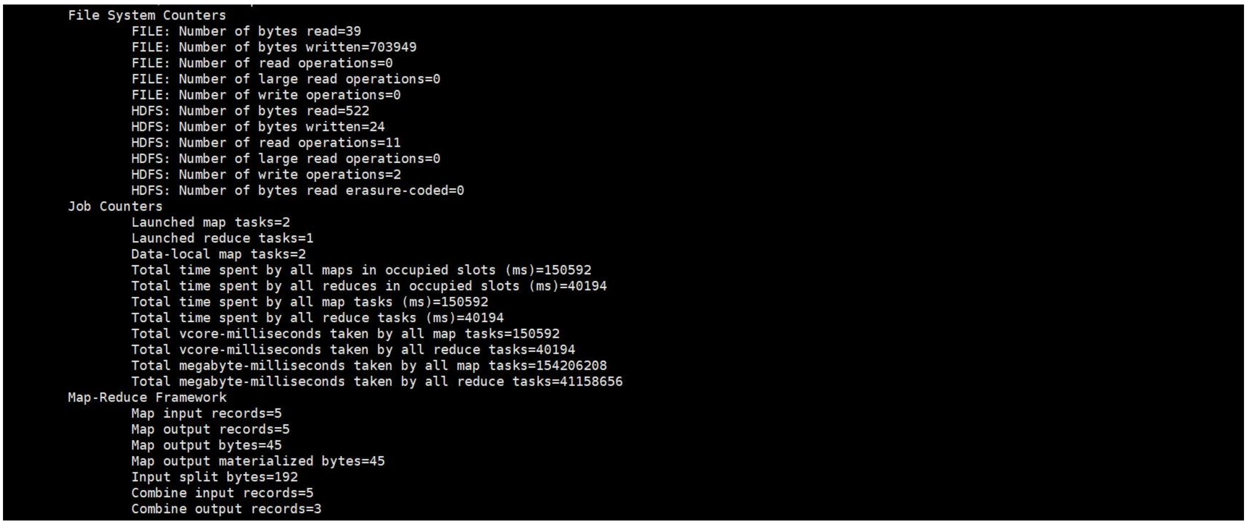
```
[hdoop@distributed-vm ~]$
[hdoop@distributed-vm ~]$
[hdoop@distributed-vm ~]$ hadoop jar units.jar hadoop.ProcessUnits /input_dir /output_dir
```

Figure run the application

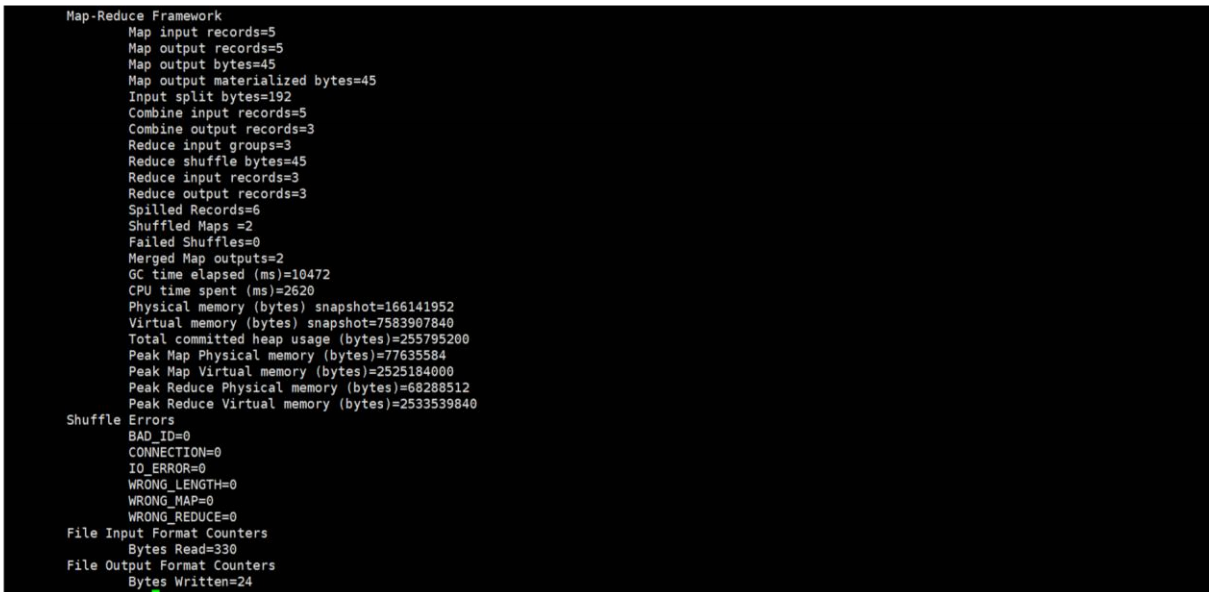
After successful execution, as shown below, the output will contain the number of input splits, the number of Map tasks, the number of reducer tasks, etc.



```
2021-07-24 14:24:14,906 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2021-07-24 14:24:16,287 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2021-07-24 14:24:18,505 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your
application with ToolRunner to remedy this.
2021-07-24 14:24:18,794 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hdoop/.staging/job_1627136555692_000
1
2021-07-24 14:24:20,491 INFO mapred.FileInputFormat: Total input files to process : 1
2021-07-24 14:24:20,889 INFO mapreduce.JobSubmitter: number of splits:2
2021-07-24 14:24:22,509 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1627136555692_0001
2021-07-24 14:24:22,513 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-07-24 14:24:23,406 INFO conf.Configuration: resource-types.xml not found
2021-07-24 14:24:23,406 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-07-24 14:24:26,506 INFO impl.YarnClientImpl: Submitted application application_1627136555692_0001
2021-07-24 14:24:26,710 INFO mapreduce.Job: The url to track the job: http://vm-test:8088/proxy/application_1627136555692_0001/
2021-07-24 14:24:26,713 INFO mapreduce.Job: Running job: job_1627136555692_0001
2021-07-24 14:25:37,393 INFO mapreduce.Job: Job job_1627136555692_0001 running in uber mode : false
2021-07-24 14:25:38,501 INFO mapreduce.Job:  map 0% reduce 0%
2021-07-24 14:26:55,694 INFO mapreduce.Job:  map 33% reduce 0%
2021-07-24 14:26:57,493 INFO mapreduce.Job:  map 83% reduce 0%
2021-07-24 14:27:02,004 INFO mapreduce.Job:  map 100% reduce 0%
2021-07-24 14:27:38,387 INFO mapreduce.Job:  map 100% reduce 33%
2021-07-24 14:27:44,619 INFO mapreduce.Job:  map 100% reduce 100%
2021-07-24 14:27:55,296 INFO mapreduce.Job: Job job_1627136555692_0001 completed successfully
2021-07-24 14:27:59,594 INFO mapreduce.Job: Counters: 54
        File System Counters
```



```
        File System Counters
                FILE: Number of bytes read=39
                FILE: Number of bytes written=703949
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=522
                HDFS: Number of bytes written=24
                HDFS: Number of read operations=11
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        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)=150592
                Total time spent by all reduces in occupied slots (ms)=40194
                Total time spent by all map tasks (ms)=150592
                Total time spent by all reduce tasks (ms)=40194
                Total vcore-milliseconds taken by all map tasks=150592
                Total vcore-milliseconds taken by all reduce tasks=40194
                Total megabyte-milliseconds taken by all map tasks=154206208
                Total megabyte-milliseconds taken by all reduce tasks=41158656
        Map-Reduce Framework
                Map input records=5
                Map output records=5
                Map output bytes=45
                Map output materialized bytes=45
                Input split bytes=192
                Combine input records=5
                Combine output records=3
```



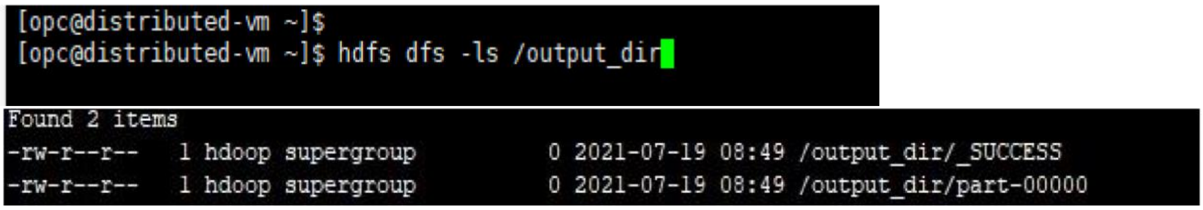
```
        Map-Reduce Framework
                Map input records=5
                Map output records=5
                Map output bytes=45
                Map output materialized bytes=45
                Input split bytes=192
                Combine input records=5
                Combine output records=3
                Reduce input groups=3
                Reduce shuffle bytes=45
                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)=10472
                CPU time spent (ms)=2620
                Physical memory (bytes) snapshot=166141952
                Virtual memory (bytes) snapshot=7583907840
                Total committed heap usage (bytes)=255795200
                Peak Map Physical memory (bytes)=77635584
                Peak Map Virtual memory (bytes)=2525184000
                Peak Reduce Physical memory (bytes)=68288512
                Peak Reduce Virtual memory (bytes)=2533539840
        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=330
        File Output Format Counters
                Bytes Written=24
```

Figure 10 job successfully completed

Step 8: check the result file in the output folder

To check the result file in the output folder, use the command below.

```
$ hdfs dfs -ls /output_dir
```



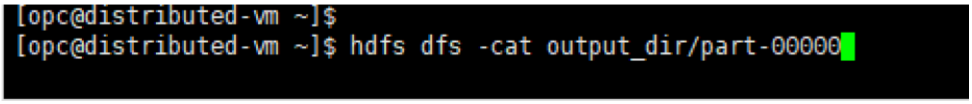
```
[opc@distributed-vm ~]$
[opc@distributed-vm ~]$ hdfs dfs -ls /output_dir
Found 2 items
-rw-r--r--   1 hdoop supergroup          0 2021-07-19 08:49 /output_dir/_SUCCESS
-rw-r--r--   1 hdoop supergroup          0 2021-07-19 08:49 /output_dir/part-00000
```

Figure 11 result file in the output folder

Step 9: copy the output folder from HDFS to the local file system

With the following command, copy the output folder from HDFS to the local file system.

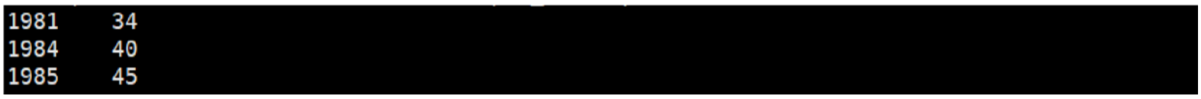
```
$ hdfs dfs -cat output_dir/part-00000
```



```
[opc@distributed-vm ~]$
[opc@distributed-vm ~]$ hdfs dfs -cat output_dir/part-00000
```

Figure 12 see the output in Part-00000 file

The generated output is shown below.



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

Figure 13 output generated by the MapReduce program

Step 10: copy the output folder from HDFS to the local file system

Use the instructions below to copy the output folder from HDFS to the local file system for analysis.

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