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
```

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/ .
```

Step 4

The following command is used to create an input directory in HDFS.

```
$HADOOP_HOME/bin/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
```

Step 6

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

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

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
```

Step 8

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

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

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
```

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
```