coreservlets.com – Hadoop Course

Input and Output

In this exercise, you will practice implementing and executing jobs with different InputFormats as well as compare and contrast split generation between the formats. You will also get a chance to implement MapReduce job that reads and writes to/from HBase.

Approx. Time: 60 minutes

Perform

1. Utilize TextInputFormat then switch to NlineInputFormat. mapRed.inputAndOutput. CountTokens MapReduce job is already implemented to use TextInputFormat, the job is responsible for taking a text file and producing total count of tokens in the file. You can run the job with the following command:

```
$ yarn jar $PLAY_AREA/Exercises.jar \
     mapRed.inputAndOutput.CountTokens \
     /training/data/hamlet.txt \
     /training/playArea/InputOutput/CountTokens
```

How many splits did the `TextInputFormat` produce and why?

Now switch to using `NlineInputFormat` with 2500 lines per split. (*WARNING:* If you don't specify number of lines per split the job will produce thousands of splits and the job may cause your VM to be briefly unresponsive.)

Re-run the job. How many splits did the `NlineInputFormat` produce and why?

2. Implement and test a job that counts each unique token in HBase table and outputs distinct tokens and corresponding counts to another HBase table. Read tokens from table 'Exercise_InputAndOutput_Tokens' column 'token:value'. There will be a single token per row. You can explore the table via 'hbase shell' command:

```
$ hbase shell

     hbase(main):028:0> scan 'Exercise_InputAndOutput_Tokens'
     ROW        COLUMN+CELL
      001       column=token:value, timestamp=1347990678225, value=apple
      002       column=token:value, timestamp=1347990678255, value=banana
      003       column=token:value, timestamp=1347990678279, value=peach
      004       column=token:value, timestamp=1347990678299, value=peach
      005       column=token:value, timestamp=1347990678325, value=banana
      006       column=token:value, timestamp=1347990678368, value=peach
      007       column=token:value, timestamp=1347990678387, value=apple
      008       column=token:value, timestamp=1347990678408, value=peach
      009       column=token:value, timestamp=1347990678429, value=banana
      010       column=token:value, timestamp=1347990679016, value=apple
     10 row(s) in 0.0560 seconds
```

Output results into the table 'Exercise_InputAndOutput_Result' where row id is the distinct

token and count stored in the ‘counts:count’ column. Verify the result by scanning through the table; it should look something like this:

```
hbase(main):030:0> scan 'Exercise_InputAndOutput_Result'
ROW         COLUMN+CELL
 apple      column=counts:count, timestamp=1347997340886, value=3
 banana     column=counts:count, timestamp=1347997340886, value=3
 peach      column=counts:count, timestamp=1347997340886, value=4
3 row(s) in 0.0220 seconds
```

Exercises project has a template set up for this job, you’ll need to fill in the blanks:

```
mapRed.inputAndOutput.UniqueCounterTool.java
mapRed.inputAndOutput.UniqueCounterMapper.java
mapRed.inputAndOutput.UniqueCounterReducer.java
```

Solution

1. The code can be found in the Solution's project:
   mapRed.inputAndOutput.CountTokens.java

   `TextInputFormat` produced 1 split because it generates a split per HDFS block. The default HDFS block is 64MB and the input file is 206.3k therefore only a single split is created. In the case of `NlineInputFormat`, a split is produced per N lines, in our case a split per 2,500 lines. Input file, hamlet.txt, contains 5,159 lines therefore 3 splits are generated. First two splits will process 2,500 lines and the third 159 reminder lines.

2. The code can be found in the Solution's project:
   `mapRed.inputAndOutput.UniqueCounterTool.java`
   `mapRed.inputAndOutput.UniqueCounterMapper.java`
   `mapRed.inputAndOutput.UniqueCounterReducer.java`