

# Big Data Tutorial 1: distributed wordcount

Hugues Talbot

November 18, 2019

## 1 Big data tutorial 1

### 1.1 Wordcount on Hadoop using Python

Lesson 1, Introduction to Map/Reduce Module, Running Wordcount with streaming, using Python code

1. Open a Terminal (Right-click on Desktop or click Terminal icon in the top toolbar)
2. Review the following to create the python code Section 1: wordcount\_mapper.py

#### 1.1.1 section 1: mapper

```
[1]: #!/usr/bin/env python
#the above just indicates to use python to intepret this file

# -----
#This mapper code will input a line of text and output <word, 1>
#
# -----

import sys                #a python module with system functions for this OS

# -----
# this 'for loop' will set 'line' to an input line from system
#   standard input file
# -----
for line in sys.stdin:

#-----
#sys.stdin call 'sys' to read a line from standard input,
# note that 'line' is a string object, ie variable, and it has methods that you
→ can apply to it,
# as in the next line
# -----
    line = line.strip()    #strip is a method, ie function, associated
                           # with string variable, it will strip
```

```

# the carriage return (by default)
keys = line.split() #split line at blanks (by default),
# and return a list of keys
for key in keys:    #a for loop through the list of keys
    value = 1
    print('{0}\t{1}'.format(key, value) ) #the {} is replaced by 0th,1st
→items in format list
#also, note that the Hadoop default is 'tab'
→separates key from the value

```

### 1.1.2 Section 2: wordcount\_reducer.py

The reducer code has some basic parts, see the comments in the code. The Lesson 2 assignment will have a similar basic structure.

```

[3]: #!/usr/bin/env python

# -----
#This reducer code will input a line of text and
# output <word, total-count>
# -----
import sys

last_key      = None          #initialize these variables
this_key      = None
running_total = 0

# -----
# Loop thru file
# -----
for input_line in sys.stdin:
    input_line = input_line.strip()

    # -----
    # Get Next Word # -----
    this_key, value = input_line.split("\t", 1) #the Hadoop default is tab
→separates key value
#the split command returns a list of strings, in this
→case into 2 variables
    value = int(value)          #int() will convert a string to integer (this
→program does no error checking)

    # -----
    # Key Check part
    # if this current key is same
    # as the last one Consolidate

```

```

#   otherwise Emit
# -----
if last_key == this_key:      #check if key has changed ('==' is
→                               # logical equality check
    running_total += value    # add value to running total

else:
    if last_key:              #if this key that was just read in
                               # is different, and the previous
                               # (ie last) key is not empty,
                               # then output
                               # the previous <key running-count>
        print( "{0}\t{1}".format(last_key, running_total) )
                               # hadoop expects tab(ie '\t')
                               # separation
    running_total = value     #reset values
    last_key = this_key

if last_key == this_key:
    print( "{0}\t{1}".format(last_key, running_total))

```

None 0

[ ]: NOTE: If you have **not** programmed **with** Python please read the following:

Python notes:

```

#   1 indentations are required to indicate blocks of code,
#   2 all code to be executed as part of some flow control
#       (e.g. if or for statements) must have the same indentation
#       (to be safe use 4 space per indentation level, and don't
#       mix with tabs)
#   3 flow control conditions have a ':' before
#       the corresponding block of code
#

```

You can cut and paste the above into a text file as follows from the terminal prompt in Cloudera VM.

Type in the following to open a text editor, and then cut and paste the above lines for wordcount\_mapper.py into the text editor, save, and exit. Repeat for wordcount\_reducer.py

```
gedit wordcount_mapper.py
```

```
gedit wordcount_reducer.py
```

Enter the following to see that the indentations line up as above

```
more wordcount_mapper.py
```

```
more wordcount_reducer.py
```

Enter the following to make it executable

```
chmod +x wordcount_mapper.py
```

```
chmod +x wordcount_reducer.py
```

Enter the following to see what directory you are in

```
pwd
```

It should be /user/cloudera , or something like that.

### 1.1.3 Section 3. Create some data:

```
echo "A long time ago in a galaxy far far away" > /home/cloudera/testfile1
```

```
echo "Another episode of Star Wars" > /home/cloudera/testfile2
```

### 1.1.4 Section 4. Create a directory on the HDFS file system (if already exists that's OK):

```
hdfs dfs -mkdir /user/cloudera/input
```

### 1.1.5 Section 5. Copy the files from local filesystem to the HDFS filesystem:

```
hdfs dfs -put /home/cloudera/testfile1 /user/cloudera/input
```

```
hdfs dfs -put /home/cloudera/testfile2 /user/cloudera/input
```

### 1.1.6 Section 6. You can see your files on HDFS

```
hdfs dfs -ls /user/cloudera/input
```

### 1.1.7 Section 7. Run the Hadoop WordCount example with the input and output specified.

Note that your file paths may differ. The " just means the command continues on next line.

```
[ ]: hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \  
-input /user/cloudera/input \  
-output /user/cloudera/output_new \  
-mapper /home/cloudera/wordcount_mapper.py \  
-reducer /home/cloudera/wordcount_reducer.py
```

Hadoop prints out a whole lot of logging or error information. If it runs you will see something like the following on the screen scroll by:

```
[ ]: ....

INFO mapreduce.Job: map 0% reduce 0%

INFO mapreduce.Job: map 67% reduce 0%

INFO mapreduce.Job: map 100% reduce 0%

INFO mapreduce.Job: map 100% reduce 100%

INFO mapreduce.Job: Job job_1442937183788_0003 completed successfully

...
```

### 1.1.8 Section 8. Check the output file to see the results:

```
hdfs dfs -cat /user/cloudera/output_new/part-00000
```

### 1.1.9 Section 9. View the output directory:

```
hdfs dfs -ls /user/cloudera/output_new
```

Look at the files there and check out the contents, e.g.:

```
hdfs dfs -cat /user/cloudera/output_new/part-00000
```

### 1.1.10 Section 10. Streaming options:

Try: `hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar -help`

or see [hadoop.apache.org/docs/r1.2.1/](http://hadoop.apache.org/docs/r1.2.1/)

Let's change the number of reduce tasks to see its effects. Setting it to 0 will execute no reducer and only produce the map output. (Note the output directory is changed in the snippet below because Hadoop doesn't like to overwrite output)

```
[ ]: hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
    -input /user/cloudera/input \
    -output /user/cloudera/output_new_0 \
    -mapper /home/cloudera/wordcount_mapper.py \
    -reducer /home/cloudera/wordcount_reducer.py \
    -numReduceTasks 0
```

Get the output file from this run, and then upload it:

```
hdfs dfs -getmerge /user/cloudera/output_new_0/* wordcount_num0_output.txt
```

Try to notice the differences between the output when the reducers are run in Step 9, versus the output when there are no reducers and only the mapper is run in this step. The point of the task is to be aware of what the intermediate results look like. A successful result will have words and counts that are not accumulated (which the reducer performs). Hopefully, this will help you get a sense of how data and tasks are split up in the map/reduce framework, and we will build upon that in the next lesson.

#### 1.1.11 11. Change the number of reducers to 2

When you use 2 reducers instead of 1 reducer, what is the difference in global sort order?

- With 1 reducer, but not 2 reducers, the word counts are in global sort order by word.
- With 2 reducers, but not 1 reducer, the word counts are in global sort order by word.
- With 1 reducer or 2 reducers, the word counts are in global sort order by word.
- With 1 reducer or 2 reducers, the word counts are NOT in global sort order by word.

[ ]: