Big Data HW2 - Solution

GROUP 24
*IDs: **304997067, 311132468, 204502926***
Ofer Gross, Yuval Katz, Ofir Nesher

S3 bucket:
https://s3.console.aws.amazon.com/s3/buckets/afeka-big-data-course?region=us-east-1&prefix=Data%2F

1. Shell commands:

```
BUCKET = 's3://afeka-big-data-course/Data'

# First check you can see the source S3 data
!hadoop fs -ls {BUCKET}

# Copy the data
!hadoop distcp {BUCKET} .

# See files in your hdfs
!hadoop fs -ls .

!chmod ugo+x *.py

# reducer test 2
!hadoop fs -cat Gutenberg/* | /home/hadoop/mapper.py | sort -k1,1 |
/home/hadoop/reducer.py | sort -k2 -g -r > Gutenberg.out

# One example for the main M/R job and runtime data to fill the table.
# This one is for Gutenberg with 1 slave (“instance” / “core”), and the
rest is similar after changing the number of slaves in the cluster’s
configuration
!time \
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
-file /home/hadoop/mapper.py \
-mapper /home/hadoop/mapper.py \
-file /home/hadoop/reducer.py \
-reducer /home/hadoop/reducer.py \
-input Gutenberg/* \
-output 'Gutenberg_1_instance'
```

2. Python code for map and reduce. (map.py and reduce.py files):

Mapper function:

```
#!/usr/bin/python
"""mapper.py"""

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
        print('%s\t%s' % (word, 1))
```

Reducer function:

```
#!/usr/bin/python
"""reducer.py"""

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)
```

```
    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue

    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # write result to STDOUT
            print('%s\t%s' % (current_word, current_count))
        current_count = count
        current_word = word

# do not forget to output the last word if needed!
if current_word == word:
    print('%s\t%s' % (current_word, current_count))
```

3. Results:

Input:	Gutenberg (3.5 MB)	Britannica (27.8 MB)	Wikipedia (2.2 GB)
1 master + 1 slave	**map tasks = 9** **reduce tasks = 3** **Run time = 01:19 mm:ss** **CPU time = 24960 ms** • Physical memory (bytes) snapshot=5235978240 • Virtual memory (bytes) snapshot=43689955328 • Total committed heap usage (bytes)=4169138176	**map tasks = 24** **reduce tasks = 3** **Run time = 02: 46 mm:ss** **CPU time = 79140 ms** • Physical memory (bytes) snapshot=12482588672 • Virtual memory (bytes) snapshot=93217808384 • Total committed heap usage (bytes)=10150739968	**map tasks = 22** **reduce tasks = 3** **Run time = 29:59 mm:ss** **CPU time = 2901550 ms** • Physical memory (bytes) snapshot=15893684224 • Virtual memory (bytes) snapshot=86471282688 • Total committed heap usage (bytes)=14915469312
1 master + 5 slaves	**map tasks = 9** **reduce tasks = 3** **Run time = 00:41 mm:ss** **CPU time = 26050 ms** • Physical memory (bytes) snapshot=5059424256 • Virtual memory (bytes) snapshot=43685306368 • Total committed heap usage (bytes)=4080009216	**map tasks = 25** **reduce tasks = 3** **Run time = 00:55 mm:ss** **CPU time = 84480 ms** • Physical memory (bytes) snapshot=12260253696 • Virtual memory (bytes) snapshot=93153951744 • Total committed heap usage (bytes)=10054795264	**map tasks = 23** **reduce tasks = 3** **Run time = 11:30 mm:ss** **CPU time = 3092400 ms** • Physical memory (bytes) snapshot=14499246080 • Virtual memory (bytes) snapshot=86442504192 • Total committed heap usage (bytes)=13443792896
1 master + 8 slaves	**map tasks = 9** **reduce tasks = 3** **Run time = 00:41 mm:ss** **CPU time = 25890 ms** • Physical memory (bytes) snapshot=5102166016 • Virtual memory (bytes) snapshot=43674800128 • Total committed heap usage (bytes)=4129816576	**map tasks = 25** **reduce tasks = 4** **Run time = 00:49 mm:ss** **CPU time = 83650 ms** • Physical memory (bytes) snapshot=12267225088 • Virtual memory (bytes) snapshot=93153239040 • Total committed heap usage (bytes)=10024910848	**map tasks = 26** **reduce tasks = 3** **Run time = 07:30 mm:ss** **CPU time = 2883660 ms** • Physical memory (bytes) snapshot=16456634368 • Virtual memory (bytes) snapshot=86454009856 • Total committed heap usage (bytes)=15899033600
1 master + 8 slaves 30 reducers			**map tasks = 22** **reduce tasks = 32** **Run time = 05:59 mm:ss** **CPU time = 3181760 ms**

			● Physical memory (bytes) snapshot=23333691392 ● Virtual memory (bytes) snapshot=211909259264 ● Total committed heap usage (bytes)=21879586816

- **We've added (in bullet points) what we think is responsible for having 3 reduce tasks.**