Sami Emre Erdogan 夏承思
Student ID: 2019280513

BIG DATA HOMEWORK 4
MapReduce

[Tasks]
Read the MapReduce paper attached, answer the following questions:

Question:
a) The results of mapping are shuffled and send to reducers, which could cost a lot of network traffic? Propose a way to significantly reduce the network traffics.
Answer:
We can take the key value pairs and combine it within a list using a function:
(k2,v2) → list (k2,v2).
This method can be seen as some kind of a small batches of reducers to optimize and reduce the network traffic which is done after completing the mapper stage.

Question:
b) If a mapper or reducer task is too slow which would make the whole mapreduce task very slow? Propose an approach to address this issue.
Answer:
To optimise and make the mapreduce job faster we can:
Number of Reducers - Configure appropriate number of reducers
Combiners - Identify if combiner can be used.
Compression - Performance can be improved by compressing the map out

Question:
2. Implement a program that counts the out-degree of all vertices in a graph using MapReduce.
There are 2 graphs to be count. (case1, case2)
Answer:
Implementation

All of the tasks ran successfully. To check:

How to run?
Just execute `run_wc.sh` and `run_od.sh` to compile and run the program

```
You can use hdfs dfs -get wc_res/ {your_local_path} to get the
result file
```

```
You can use hdfs dfs -get wc_res/ {your_local_path} to get the
result file
```

To run the bonus question, just go to the q3_top20 folder and follow the same steps above.

Sami Emre Erdogan 夏承思
Student ID: 2019280513

Mapper

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

import sys

for line in sys.stdin:
    line = line.strip()
    words = line.split(" ")
    for word in words:
        print('%s\t%s' % (word, 1))
```

This is my mapper code for this task. Importing sys allowed me to input the file and check each line. Strip and slip functions allowed me to reduce unnecessary characters in between lines and split the words with the space character. Next I checked every Word with a for loop. In the PRINT line we send the key first (%s is the pointer for it, then tab symbol then the value (second %s)) to the stdout. The key value is divided by the tab symbol. At the end we output all of the keys with the value 1

Reducer

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

import sys

key   = None
total = 0
for line in sys.stdin:
    k, v  = line.split('\t', 1)
    count = int(v.strip())

    if key == k:
        total += count
    else:
        if key:
            print('{}\t{}'.format(key, total))
        key   = k
        total = count

# do not forget this
if key:
    print ('{}\t{}'.format(key, total))
```

I have used the same code given in wcreducer.py
We take the modified file done by mapper. We check each line within our input. It splits and strips the key value pairs from the tab and the value 1 and checks within the for loop how many same keys are there so it can combine them into one and increment the total number which is the value for the pair.

Sami Emre Erdogan 夏承思
Student ID: 2019280513

Question:
3. [Bonus 10pts] Implement a program that find the top-20 biggest out-degree vertices also using MapReduce.
Note:
For case1, just find top-2 biggest out-degree vertices.
Answer:

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

import sys
from heapq import nlargest

counter = {}

for line in sys.stdin:
    k, v  = line.strip().split('\t', 1)
    count = int(v)

    if k in counter:
        counter[k] += count
    else:
        counter[k] = count

Top20 = nlargest(20, counter, key = counter.get)

for val in Top20:
    print(val, ":", counter.get(val))
```

In this bonus question I have used the heapq library to use the nLargest function to call the Top20 keys that has the largest values. I have achieved this by storing the pairs in a dictionary. The mapper is the same I have just modified the reducer file according to this task.