IEMS5730 Spring 2019 Homework 1

I declare that the assignment submitted on Elearning system is original except for source material explicitly acknowledged, and that the same or related material has not been previously submitted for another course. I also acknowledge that I am aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the website http://www.cuhk.edu.hk/policy/academichonesty/.

Signed (Student Wenli SONG) Date: 23 Dec 2018

Name Wenli SONG SID 1155114524

Question a.

Download datasets

```
$ wget --user bigdata --password spring2019bigdata \
http://mobitec.ie.cuhk.edu.hk/iems5730Spring2019/homework/hw1_small_dataset.zip
```

Upzip it

```
$ upzip hw1_small_dataset.zip
```

Put datasets into HDFS

```
$ hdfs dfs -copyFromLocal hw1_small_dataset hw1_small_dataset
```

For this matrix multiplication question, I use 2 MapReduce jobs to implement.

The codes are shown as below:

Mapper 1.py

```
#!/home/ubuntu/miniconda2/bin/python
import sys
import os


def read_input(file):
    for line in file:
        yield line.rstrip().split()


def main(separator='\t'):
    try:
        file_path = os.environ['mapreduce_map_input_file']
```

```
    except KeyError:
        file_path = os.environ['map_input_file']
    file_name = os.path.split(file_path)[-1]

    data = read_input(sys.stdin)
    if file_name in ["M_small.dat","M_median.dat","M_large.dat"]:
        for line in data:
            # (j M,i,v)
            print "%d\t%s,%d,%f" % (int(line[1]), 'M', int(line[0]),
float(line[2]))
    elif file_name in ["N_small.dat","N_median.dat","N_large.dat"]:
        for line in data:
            # (j N,k,w)
            print "%d\t%s,%d,%f" % (int(line[0]), 'N', int(line[1]),
float(line[2]))


if __name__ == "__main__":
    main()
```

Reducer 1.py

```
#!/home/ubuntu/miniconda2/bin/python
import sys
from itertools import groupby
from operator import itemgetter


def read_mapper_out(file, separator=None):
    for line in file:
        yield line.rstrip().split(separator)


def main(separator='\t'):
    data = read_mapper_out(sys.stdin, separator=separator)
    for key, group in groupby(data, itemgetter(0)):
        try:
            M_list = []
            N_list = []
            for key, val in group:
                val = val.split(',')
                if val[0] == 'M':
                    M_list.append([int(val[1]), float(val[2])])
                else:
                    N_list.append([int(val[1]), float(val[2])])
```

```
            for i, v in M_list:
                for k, w in N_list:
                    print "%d,%d\t%f" % (i, k, v * w)
        except:
            pass


if __name__ == "__main__":
    main()
```

Mapper 2.py

```
#!/home/ubuntu/miniconda2/bin/python
import sys


def read_reduce_output(file, separator=None):
    for line in file:
        yield line.rstrip().split(separator)


def main(separator='\t'):
    data = read_reduce_output(sys.stdin, separator=separator)
    for line in data:
        print "%s\t%f" % (line[0], float(line[1]))


if __name__ == "__main__":
    main()
```

Reducer 2.py

```
#!/home/ubuntu/miniconda2/bin/python
import sys
from itertools import groupby
from operator import itemgetter


def read_mapper_out(file, separator=None):
    for line in file:
        yield line.rstrip().split(separator)


def main(separator='\t'):
    data = read_mapper_out(sys.stdin, separator=separator)
    for key, group in groupby(data, itemgetter(0)):
```

```
        try:
            total = sum((float(val) for key, val in group))
            coordinate = key.split(',')
            print "%d\t%d\t%f" % (int(coordinate[0]), int(coordinate[1]), total)
        except:
            pass


if __name__ == "__main__":
    main()
```

Run

```
$ hadoop jar ~/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar \
-file ~/mapper1.py -mapper mapper1.py -file ~/reducer1.py -reducer reducer1.py \
-input hw1_small_dataset -output small-output/output-1

$ hadoop jar ~/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar \
-file ~/mapper2.py -mapper mapper2.py -file ~/reducer2.py -reducer reducer2.py \
-input small-output/output-1 -output small-output/output-2
```

Time Cost

	#Mapper	#Reducer	Time Cost
Map&Reduce stage1	2	1	17 s
Map&Reduce stage2	2	1	21 s

Check result, my student No. is 1155114524, hence I have to get the 24th rows, 124rows, 224rows, 324th rows, …

```
$ hdfs dfs -cat output-2/part* | awk '{if(($1-24)%100==0) print}' | \
sort -V > small-result.txt
```

Part of the results are shown as below

24 1 20.250000
24 2 2.250000
24 5 8.250000
24 6 15.000000
24 7 20.250000
24 10 12.000000
24 11 30.250000
24 14 9.000000
24 15 25.750000
24 16 24.750000
…

(To see entire result, please refer to '1155114524_result.pdf')

Question b.

In this section, we need to perform MapReduce on the median and large datasets.

Download datasets

```
$ wget wget --user bigdata --password spring2019bigdata \
http://mobitec.ie.cuhk.edu.hk/iems5730Spring2019/homework/hw1_median_dataset.zip
$ wget wget --user bigdata --password spring2019bigdata \
http://mobitec.ie.cuhk.edu.hk/iems5730Spring2019/homework/hw1_large_dataset.zip
```

Unzip datasets

```
$ upzip hw1_median_dataset.zip
$ upzip hw1_large_dataset.zip
```

Copy them to HDFS

```
$ hdfs dfs -copyFromLocal hw1_median_dataset hw1_median_dataset
$ hdfs dfs -copyFromLocal hw1_large_dataset hw1_large_dataset
```

Run using median dataset

```
$ hadoop jar ~/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar \
-file ~/mapper1.py -mapper mapper1.py -file ~/reducer1.py -reducer reducer1.py \
-input hw1_median_dataset -output median-output/output-1

$ hadoop jar ~/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar \
-file ~/mapper2.py -mapper mapper2.py -file ~/reducer2.py -reducer reducer2.py \
-input median-output/output-1 -output median-output/output-2
```

Time Cost

	#Mapper	#Reducer	Time Cost
Map&Reduce stage1	2	1	1 mins 49 s
Map&Reduce stage2	16	1	4 mins 48 s

Check result

```
$ hdfs dfs -cat median-output/output-2/part* | awk '{if(($1-524)%1000==0) print}' |
\
sort -V > median-result.txt
```

Part of the result are shown as below

524 1 321.500000
524 2 219.000000
524 3 149.500000
524 4 37.500000
524 5 120.250000
524 6 192.250000
524 7 169.750000
524 8 24.000000
524 9 3.750000

524 10 239.000000
…
(To see entire result, please refer to '1155114524_result.pdf')

Run using large dataset

```
$ hadoop jar ~/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar \
-file ~/mapper1.py -mapper mapper1.py -file ~/reducer1.py -reducer reducer1.py \
-input hw1_large_dataset -output large-output/output-1

$ hadoop jar ~/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar \
-file ~/mapper2.py -mapper mapper2.py -file ~/reducer2.py -reducer reducer2.py \
-input large-output/output-1 -output large-output/output-2
```

Time Cost

	#Mapper	#Reducer	Time Cost
Map&Reduce stage1	2	1	12 mins 57 s
Map&Reduce stage2	72	1	21 mins 44 s

Check result

```
$ hdfs dfs -cat large-output/output-2/part* | awk '{if(($1-524)%1000==0) print}' |
\
sort -V  > large-result.txt
```

Part of the results are shown as below
524 1 3011.500000
524 2 1773.000000
524 3 1348.750000
524 4 438.500000
524 5 1106.750000
524 6 1724.750000
524 7 1324.250000
524 8 266.750000
524 9 443.500000
524 10 1874.250000
…
(To see entire result, please refer to '1155114524_result.pdf')

Question c.

In this section, we will try running the MapReduce job with different number of mappers and reducers and observe the cost time in the 2dn MapReduce stage.

Config the amount of memory to request from the scheduler for each map and reduce task in mapred-site.xml (128MB, 256MB, 512MB respectively)

```
    <property>
            <name>mapreduce.input.fileinputformat.split.minsize</name>
            <value>134217728</value>
    </property>
    <property>
            <name>mapreduce.input.fileinputformat.split.minsize</name>
            <value>268435456</value>
    </property>
    <property>
            <name>mapreduce.input.fileinputformat.split.minsize</name>
            <value>536870912</value>
    </property>
```

Run

```
$ hadoop jar ~/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar \
-D mapreduce.job.maps=18 -D mapreduce.job.reduces=16 \
-file ~/mapper2.py -mapper mapper2.py -file ~/reducer2.py -reducer reducer2.py \
-input large-output/output-1 -output questionC/output-18-16

$ hadoop jar ~/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar \
-D mapreduce.job.maps=18 -D mapreduce.job.reduces=32 \
-file ~/mapper2.py -mapper mapper2.py -file ~/reducer2.py -reducer reducer2.py \
-input large-output/output-1 -output questionC/output-18-32

$ hadoop jar ~/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar \
-D mapreduce.job.maps=18 -D mapreduce.job.reduces=64 \
-file ~/mapper2.py -mapper mapper2.py -file ~/reducer2.py -reducer reducer2.py \
-input large-output/output-1 -output questionC/output-18-64

$ hadoop jar ~/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar \
-D mapreduce.job.maps=36 -D mapreduce.job.reduces=16 \
-file ~/mapper2.py -mapper mapper2.py -file ~/reducer2.py -reducer reducer2.py \
-input large-output/output-1 -output questionC/output-36-16

$ hadoop jar ~/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar \
-D mapreduce.job.maps=36 -D mapreduce.job.reduces=32 \
-file ~/mapper2.py -mapper mapper2.py -file ~/reducer2.py -reducer reducer2.py \
-input large-output/output-1 -output questionC/output-36-32

$ hadoop jar ~/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar \
-D mapreduce.job.maps=36 -D mapreduce.job.reduces=64 \
-file ~/mapper2.py -mapper mapper2.py -file ~/reducer2.py -reducer reducer2.py \
-input large-output/output-1 -output questionC/output-36-64
```

```
$ hadoop jar ~/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar \
-D mapreduce.job.maps=72 -D mapreduce.job.reduces=16 \
-file ~/mapper2.py -mapper mapper2.py -file ~/reducer2.py -reducer reducer2.py \
-input large-output/output-1 -output questionC/output-72-16

$ hadoop jar ~/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar \
-D mapreduce.job.maps=72 -D mapreduce.job.reduces=32 \
-file ~/mapper2.py -mapper mapper2.py -file ~/reducer2.py -reducer reducer2.py \
-input large-output/output-1 -output questionC/output-72-32

$ hadoop jar ~/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.2.jar \
-D mapreduce.job.maps=72 -D mapreduce.job.reduces=64 \
-file ~/mapper2.py -mapper mapper2.py -file ~/reducer2.py -reducer reducer2.py \
-input large-output/output-1 -output questionC/output-72-64
```

Time Cost

#Mapper	#Reducer	Mapper Time	Reducer Time	Total job time
18	16	9 mins 44 s	11 mins 40 s	13 mins 59 s
18	32	9 mins 26 s	10 mins 2 s	12 mins 26 s
18	64	10 mins 21 s	7 mins 8 s	13 mins 33 s
36	16	8 mins 34 s	7 mins 37 s	11 mins 50 s
36	32	7 mins 23 s	4 mins 51 s	10 mins 1 s
36	64	7 mins 17 s	5 mins 25 s	10 mins 31 s
72	16	7 mins 33 s	8 mins 23 s	11 mins 26 s
72	32	7 mins 21 s	6 mins 58 s	10 mins 5 s
72	64	7 mins 53 s	8 mins 1 s	11 mins 12 s

Maximum mapper time	10 mins 21 s
Minimum mapper time	7 mins 17 s
Average mapper time	8 mins 24 s
Maximum reducer time	11 mins 40 s
Minimum reducer time	4 mins 51 s
Average reducer time	7 mins 47 s
Maximum Total job time	13 mins 59 s
Minimum Total job time	10 mins 1 s
Average Total job time	11 mins 40 s