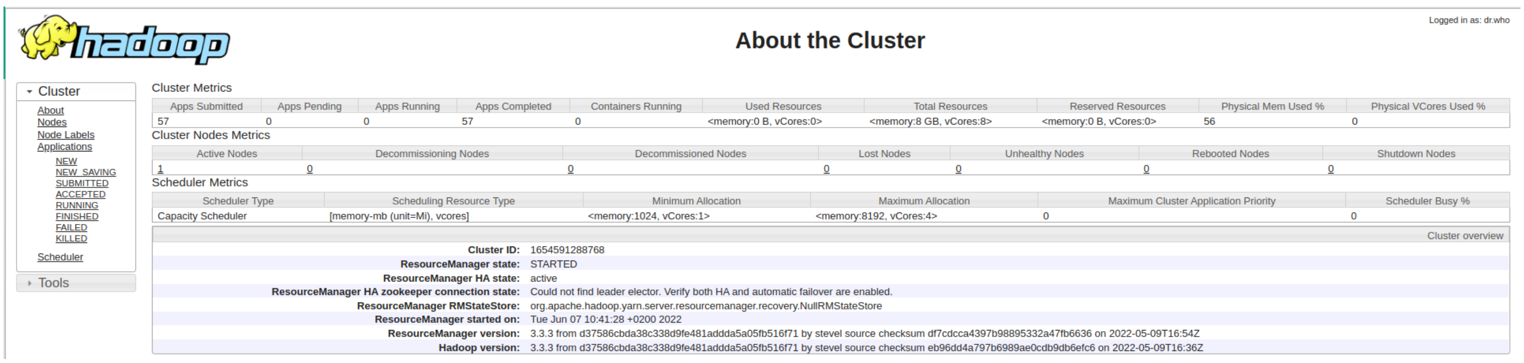
DDA Exercise 05

Farjad Ahmed - 1747371
First Semester - Group 1

System Information

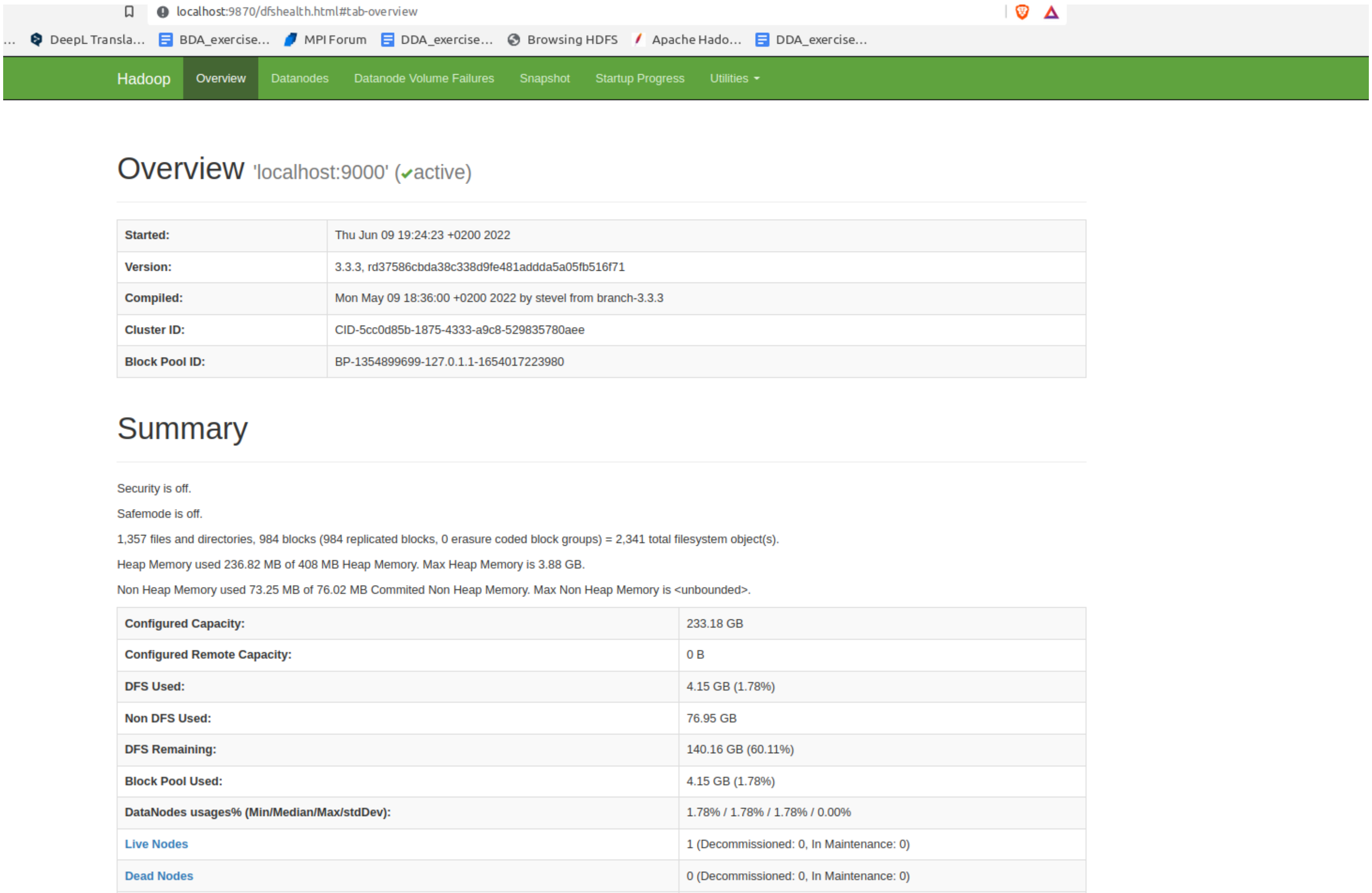
CPU op-mode(s): 32-bit, 64-bit
  Address sizes: 39 bits physical, 48 bits virtual
  Byte Order: Little Endian
CPU(s): 8
  On-line CPU(s) list: 0-7
Vendor ID: GenuineIntel
  Model name: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
    CPU family: 6
    Model: 142
    Thread(s) per core: 2
    Core(s) per socket: **4**



The report contains answers to questions, then code explanations and Performance Analysis at the very end.

MapReduce 1.3

Word Count Example:


localhost:9870/dfshealth.html#tab-overview

... DeepL Transla... BDA_exercise... MPI Forum DDA_exercise... Browsing HDFS Apache Hado... DDA_exercise...

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Overview 'localhost:9000' (✔active)

Started:	Thu Jun 09 19:24:23 +0200 2022
Version:	3.3.3, rd37586cbda38c338d9fe481addda5a05fb516f71
Compiled:	Mon May 09 18:36:00 +0200 2022 by stevel from branch-3.3.3
Cluster ID:	CID-5cc0d85b-1875-4333-a9c8-529835780aee
Block Pool ID:	BP-1354899699-127.0.1.1-1654017223980

Summary

Security is off.

Safemode is off.

1,357 files and directories, 984 blocks (984 replicated blocks, 0 erasure coded block groups) = 2,341 total filesystem object(s).

Heap Memory used 236.82 MB of 408 MB Heap Memory. Max Heap Memory is 3.88 GB.

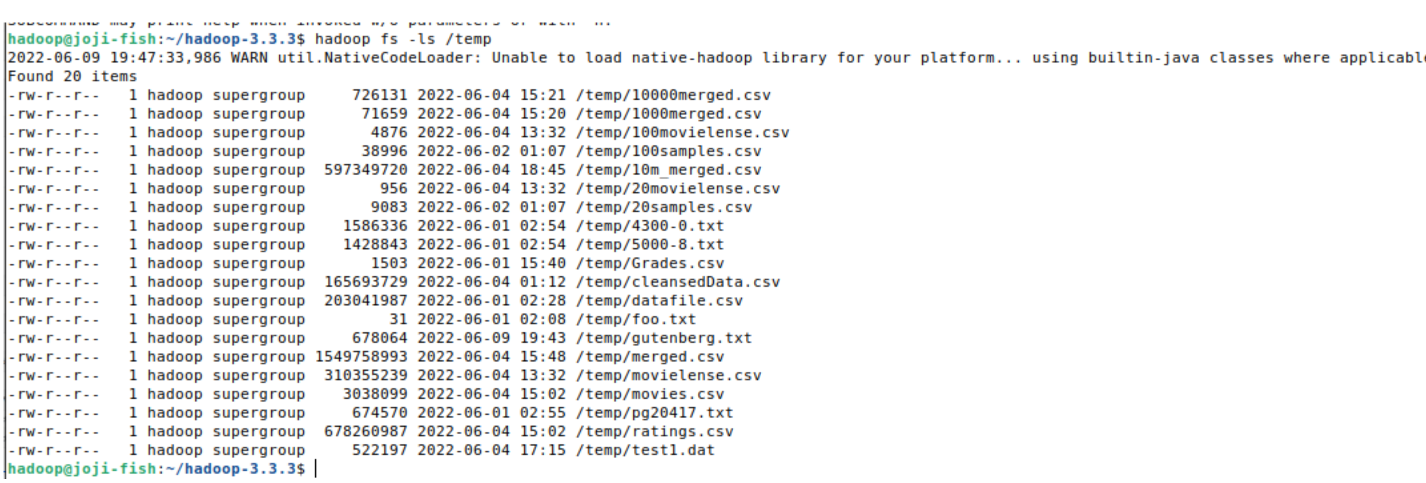
Non Heap Memory used 73.25 MB of 76.02 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	233.18 GB
Configured Remote Capacity:	0 B
DFS Used:	4.15 GB (1.78%)
Non DFS Used:	76.95 GB
DFS Remaining:	140.16 GB (60.11%)
Block Pool Used:	4.15 GB (1.78%)
DataNodes usages% (Min/Median/Max/stdDev):	1.78% / 1.78% / 1.78% / 0.00%
Live Nodes	1 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)

1. This example starts with downloading the gutenberg dataset and putting it into hadoop directory using the command.

hdfs dfs -put /mnt/Farjad_Ahmed/Masters/Distributed_Data_Analytics/Exercise_5/gutenberg.txt /temp/gutenberg.txt

2. Checking hadoop directory


```
hadoop@joji-fish:~/hadoop-3.3.3$ hadoop fs -ls /temp
2022-06-09 19:47:33,986 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 20 items
-rw-r--r--   1 hadoop supergroup     726131 2022-06-04 15:21 /temp/10000merged.csv
-rw-r--r--   1 hadoop supergroup      71659 2022-06-04 15:20 /temp/1000merged.csv
-rw-r--r--   1 hadoop supergroup       4876 2022-06-04 13:32 /temp/100movielense.csv
-rw-r--r--   1 hadoop supergroup      38996 2022-06-02 01:07 /temp/100samples.csv
-rw-r--r--   1 hadoop supergroup  597349720 2022-06-04 18:45 /temp/10m_merged.csv
-rw-r--r--   1 hadoop supergroup        956 2022-06-04 13:32 /temp/20movielense.csv
-rw-r--r--   1 hadoop supergroup       9083 2022-06-02 01:07 /temp/20samples.csv
-rw-r--r--   1 hadoop supergroup    1586336 2022-06-01 02:54 /temp/4300-0.txt
-rw-r--r--   1 hadoop supergroup    1428843 2022-06-01 02:54 /temp/5000-8.txt
-rw-r--r--   1 hadoop supergroup       1503 2022-06-01 15:40 /temp/Grades.csv
-rw-r--r--   1 hadoop supergroup  165693729 2022-06-04 01:12 /temp/cleansedData.csv
-rw-r--r--   1 hadoop supergroup  203041987 2022-06-01 02:28 /temp/datafile.csv
-rw-r--r--   1 hadoop supergroup         31 2022-06-01 02:08 /temp/foo.txt
-rw-r--r--   1 hadoop supergroup     678064 2022-06-09 19:43 /temp/gutenberg.txt
-rw-r--r--   1 hadoop supergroup 1549758993 2022-06-04 15:48 /temp/merged.csv
-rw-r--r--   1 hadoop supergroup  310355239 2022-06-04 13:32 /temp/movielense.csv
-rw-r--r--   1 hadoop supergroup    3038099 2022-06-04 15:02 /temp/movies.csv
-rw-r--r--   1 hadoop supergroup     674570 2022-06-01 02:55 /temp/pg20417.txt
-rw-r--r--   1 hadoop supergroup  678260987 2022-06-04 15:02 /temp/ratings.csv
-rw-r--r--   1 hadoop supergroup     522197 2022-06-04 17:15 /temp/test1.dat
hadoop@joji-fish:~/hadoop-3.3.3$ |
```

3. Mapper for this example is simple, it reads the designated file given in the hadoop command, The line is split by default by the whitespaces using the spli() function in the for loop which returns a list of tokens of the sentence. Another, nested for loop iterates over 'words' and print a key value combination, where each word

is the key while value is its count, since each token is just one word, hence value for each token would be 1, which is printed as <word><separator><value>. In this case the separator is a tab.

mapper.py
admin:///home/hadoop/scripts

```
#!/usr/bin/python3
import sys
for line in sys.stdin:
  words = line.split()
  for word in words:
    print('{0}\t{1}'.format(word, 1))
```

2. The reducer for this example starts with initiating 3 variables which will be used for this algorithm. It receives the input through the std.in, which was the printed information from the mapper. It starts with iterating over each print received and strips the received data and removes the leading and trailing spaces to avoid issues in splitting. The data is a key value pair hence it always breaks in a list of a key and a value when split based on the tab separator which it is in this case. In order to avoid invalid data a try except block is used by converting the count to int which it should be converted if the data is appropriate. If the count value is not invalid, it is ignored by continuing that line through ‘continue’ which skips that loop where it is called. The data received by reducer is sorted by key by hadoop, hence for the conditional each new word first goes into the else and if current_word is not false, it prints the output as the current_word and the current_count. It sets the current word as the last word received and the count to the count till this point. Now, if the same word comes in again, it goes to the first if condition in line 17 and the count is increased, and later printed when all similar instances are received before counting the new word. Lastly, for the last word it could be a singular instance, this case is handled separately in line 25.

```python
#!/usr/bin/python3

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)
    try:
        count = int(count)
    except ValueError:
        continue
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print('%s\t%s' % (current_word, current_count))
        current_count = count
        current_word = word

if current_word == word:
    print('%s\t%s' % (current_word, current_count))
```

3. Running this in hadoop.

bin/hadoop jar /home/hadoop/hadoop-3.3.3/share/hadoop/tools/lib/hadoop-streaming-3.3.3.jar -file ~/scripts/mapper.py -mapper ~/scripts/mapper.py -file ~/scripts/reducer.py -reducer ~/scripts/reducer.py -input /temp/gutenberg.txt -output /myoutputs/GUTENBERG_OUT

```
adoop@joji-fish:~/hadoop-3.3.3$ bin/hadoop jar /home/hadoop/hadoop-3.3.3/share/hadoop/tools/lib/hadoop-streaming-3.3.3.jar -file ~/scripts/mapper.py    -mapper ~/scripts/mapper.py -file ~/scripts/reducer.py   -reducer ~/scripts/reducer.py -input /temp/gutenberg.txt  -out
ut /myoutputs/GUTENBERG_OUT
022-06-09 20:18:52,390 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
022-06-09 20:18:52,670 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
ackageJobJar: [/home/hadoop/scripts/mapper.py, /home/hadoop/scripts/reducer.py, /tmp/hadoop-unjar17490105470731602069/] [] /tmp/streamjob16508320388373607374.jar tmpDir=null
022-06-09 20:18:54,316 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
022-06-09 20:18:54,650 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
022-06-09 20:18:55,012 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging/job_1654795488707_0007
022-06-09 20:18:55,779 INFO mapred.FileInputFormat: Total input files to process : 1
022-06-09 20:18:56,376 INFO mapreduce.JobSubmitter: number of splits:2
022-06-09 20:18:56,730 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1654795488707_0007
022-06-09 20:18:56,730 INFO mapreduce.JobSubmitter: Executing with tokens: []
022-06-09 20:18:57,027 INFO conf.Configuration: resource-types.xml not found
022-06-09 20:18:57,027 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
022-06-09 20:18:57,124 INFO impl.YarnClientImpl: Submitted application application_1654795488707_0007
022-06-09 20:18:57,190 INFO mapreduce.Job: The url to track the job: http://joji-fish:8088/proxy/application_1654795488707_0007/
022-06-09 20:18:57,193 INFO mapreduce.Job: Running job: job_1654795488707_0007
022-06-09 20:19:06,378 INFO mapreduce.Job: Job job_1654795488707_0007 running in uber mode : false
022-06-09 20:19:06,379 INFO mapreduce.Job:  map 0% reduce 0%
022-06-09 20:19:15,501 INFO mapreduce.Job:  map 100% reduce 0%
022-06-09 20:19:24,621 INFO mapreduce.Job:  map 100% reduce 100%
022-06-09 20:19:25,645 INFO mapreduce.Job: Job job_1654795488707_0007 completed successfully
022-06-09 20:19:25,925 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=1078912
                FILE: Number of bytes written=2995437
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=682344
                HDFS: Number of bytes written=233636
                HDFS: Number of read operations=11
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=13232
                Total time spent by all reduces in occupied slots (ms)=7024
                Total time spent by all map tasks (ms)=13232
                Total time spent by all reduce tasks (ms)=7024
                Total vcore-milliseconds taken by all map tasks=13232
                Total vcore-milliseconds taken by all reduce tasks=7024
                Total megabyte-milliseconds taken by all map tasks=13549568
                Total megabyte-milliseconds taken by all reduce tasks=7192576
        Map-Reduce Framework
                Map input records=13006
                Map output records=105488
                Map output bytes=867930
                Map output materialized bytes=1078918
                Input split bytes=184
                Combine input records=0
                Combine output records=0
                Reduce input groups=21438
                Reduce shuffle bytes=1078918
                Reduce input records=105488
                Reduce output records=21438
                Spilled Records=210976
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=220
                CPU time spent (ms)=10020
                Physical memory (bytes) snapshot=833597440
                Virtual memory (bytes) snapshot=8227811328
                Total committed heap usage (bytes)=792723456
                Peak Map Physical memory (bytes)=298692608
                Peak Map Virtual memory (bytes)=2741698560
```


```
        Map-Reduce Framework
                Map input records=13006
                Map output records=105488
                Map output bytes=867930
                Map output materialized bytes=1078918
                Input split bytes=184
                Combine input records=0
                Combine output records=0
                Reduce input groups=21438
                Reduce shuffle bytes=1078918
                Reduce input records=105488
                Reduce output records=21438
                Spilled Records=210976
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=220
                CPU time spent (ms)=10020
                Physical memory (bytes) snapshot=833597440
                Virtual memory (bytes) snapshot=8227811328
                Total committed heap usage (bytes)=792723456
                Peak Map Physical memory (bytes)=298692608
                Peak Map Virtual memory (bytes)=2741698560
```


```
                Peak Reduce Physical memory (bytes)=237047808
                Peak Reduce Virtual memory (bytes)=2746961920
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=682160
        File Output Format Counters
                Bytes Written=233636
2022-06-09 20:19:25,927 INFO streaming.StreamJob: Output directory: /myoutputs/GUTENBERG_OUT
hadoop@joji-fish:~/hadoop-3.3.3$
```

localhost:9870/explorer.html#/

DeepL Transla… BDA_exercise… MPI Forum DDA_exercise… Browsing HDFS Apache Hado… DDA_exercise…

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Browse Directory

/ Go!

Show 25 entries Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
	drwxr-xr-x	hadoop	supergroup	0 B	Jun 01 03:03	0	0 B	hadoop
	drwxr-xr-x	hadoop	supergroup	0 B	May 31 19:17	0	0 B	hadoopdemo
	drwxr-xr-x	hadoop	supergroup	0 B	Jun 09 20:19	0	0 B	myoutputs
	drwxr-xr-x	hadoop	supergroup	0 B	Jun 09 19:43	0	0 B	temp
	drwxr-xr-x	hadoop	supergroup	0 B	Jun 01 13:24	0	0 B	tmp

Showing 1 to 5 of 5 entries

Previous 1 Next

Hadoop, 2022.

localhost:9870/explorer.html#/myoutputs

DeepL Transla… BDA_exercise… MPI Forum DDA_exercise… Browsing HDFS Apache Hado… DDA_exercise…

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Browse Directory

/myoutputs Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hadoop	supergroup	0 B	Jun 07 12:47	0	0 B	3.3_tm_810
drwxr-xr-x	hadoop	supergroup	0 B	Jun 07 12:24	0	0 B	3.3_tm_82
drwxr-xr-x	hadoop	supergroup	0 B	Jun 07 12:31	0	0 B	3.3_tm_84
drwxr-xr-x	hadoop	supergroup	0 B	Jun 07 12:40	0	0 B	3.3_tm_88
drwxr-xr-x	hadoop	supergroup	0 B	Jun 09 20:19	0	0 B	GUTENBERG_OUT
drwxr-xr-x	hadoop	supergroup	0 B	Jun 04 00:51	0	0 B	farjad
drwxr-xr-x	hadoop	supergroup	0 B	Jun 09 19:44	0	0 B	myoutput1122
drwxr-xr-x	hadoop	supergroup	0 B	Jun 09 19:51	0	0 B	myoutput11222
drwxr-xr-x	hadoop	supergroup	0 B	Jun 05 15:00	0	0 B	myoutput_2.1
drwxr-xr-x	hadoop	supergroup	0 B	Jun 05 15:34	0	0 B	myoutput_2.2
drwxr-xr-x	hadoop	supergroup	0 B	Jun 04 18:23	0	0 B	myoutput_3.1a
drwxr-xr-x	hadoop	supergroup	0 B	Jun 06 21:49	0	0 B	myoutput_3.1a1
drwxr-xr-x	hadoop	supergroup	0 B	Jun 06 21:53	0	0 B	myoutput_3.1a2
drwxr-xr-x	hadoop	supergroup	0 B	Jun 05 21:15	0	0 B	myoutput_3.1ae
drwxr-xr-x	hadoop	supergroup	0 B	Jun 06 21:54	0	0 B	myoutput_3.1b1
drwxr-xr-x	hadoop	supergroup	0 B	Jun 06 21:55	0	0 B	myoutput_3.1c1
drwxr-xr-x	hadoop	supergroup	0 B	Jun 05 13:28	0	0 B	myoutput_3.1e

localhost:9870/explorer.html#/myoutputs/GUTENBERG_OUT

DeepL Transla... BDA_exercise... MPI Forum DDA_exercise... Browsing HDFS Apache Hado... DDA_exercise...

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Browse Directory

/myoutputs/GUTENBERG_OUT Go!

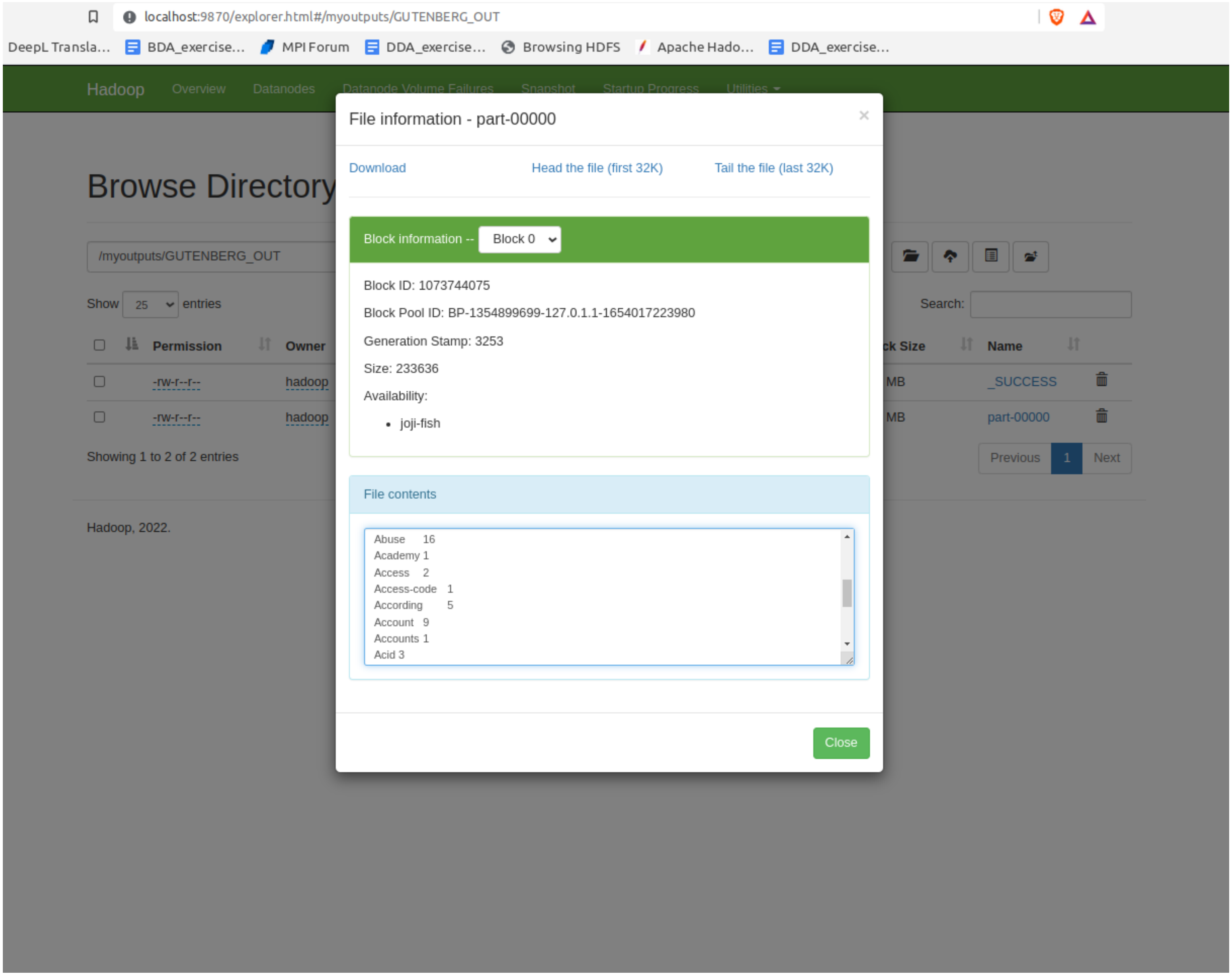
Show 25 entries Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
	-rw-r--r--	hadoop	supergroup	0 B	Jun 09 20:19	1	128 MB	_SUCCESS
	-rw-r--r--	hadoop	supergroup	228.16 KB	Jun 09 20:19	1	128 MB	part-00000

Showing 1 to 2 of 2 entries

Previous 1 Next

Hadoop, 2022.

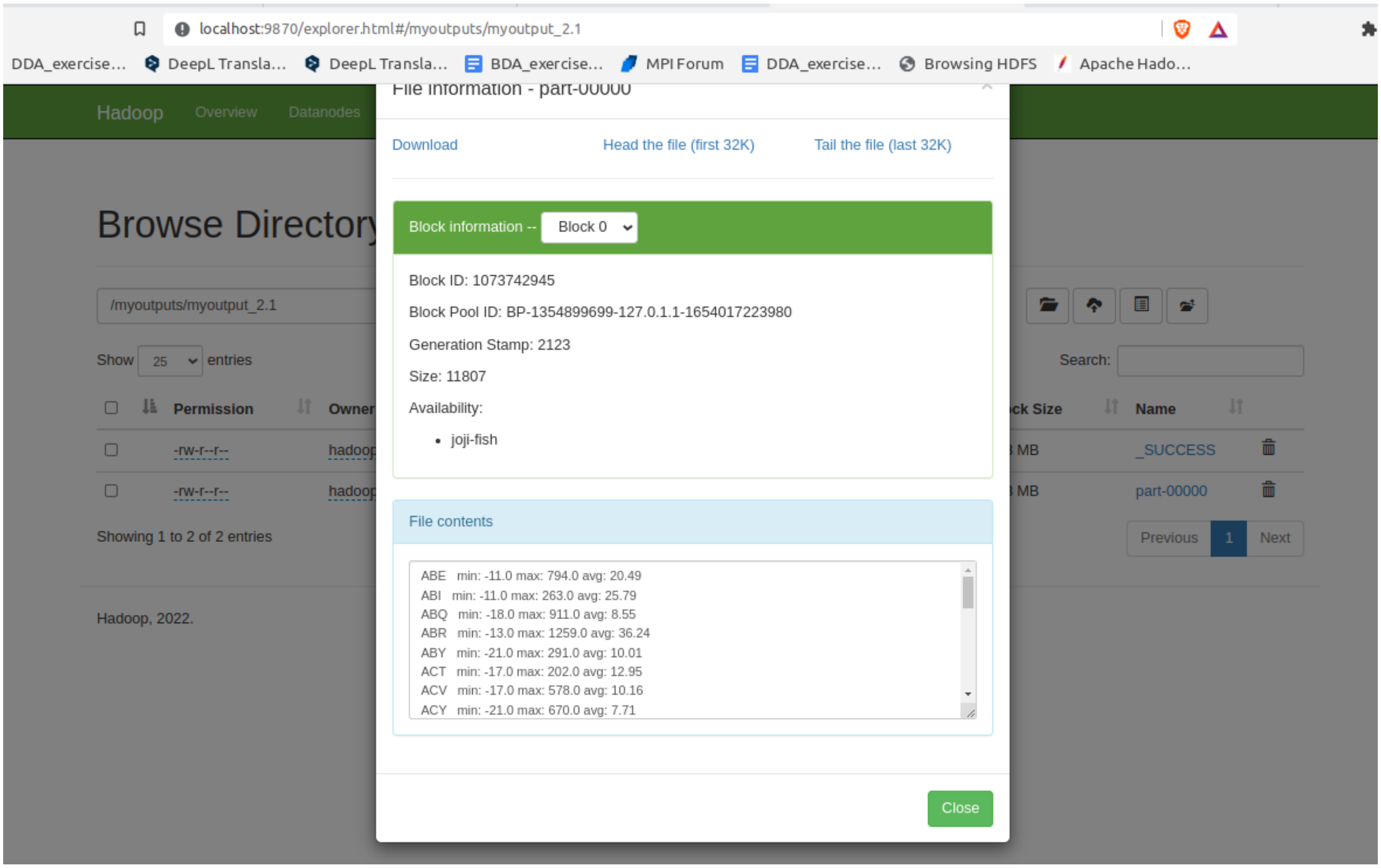


It can be seen in the output the word count is calculated for the text input to the mapreduce framework successfully.

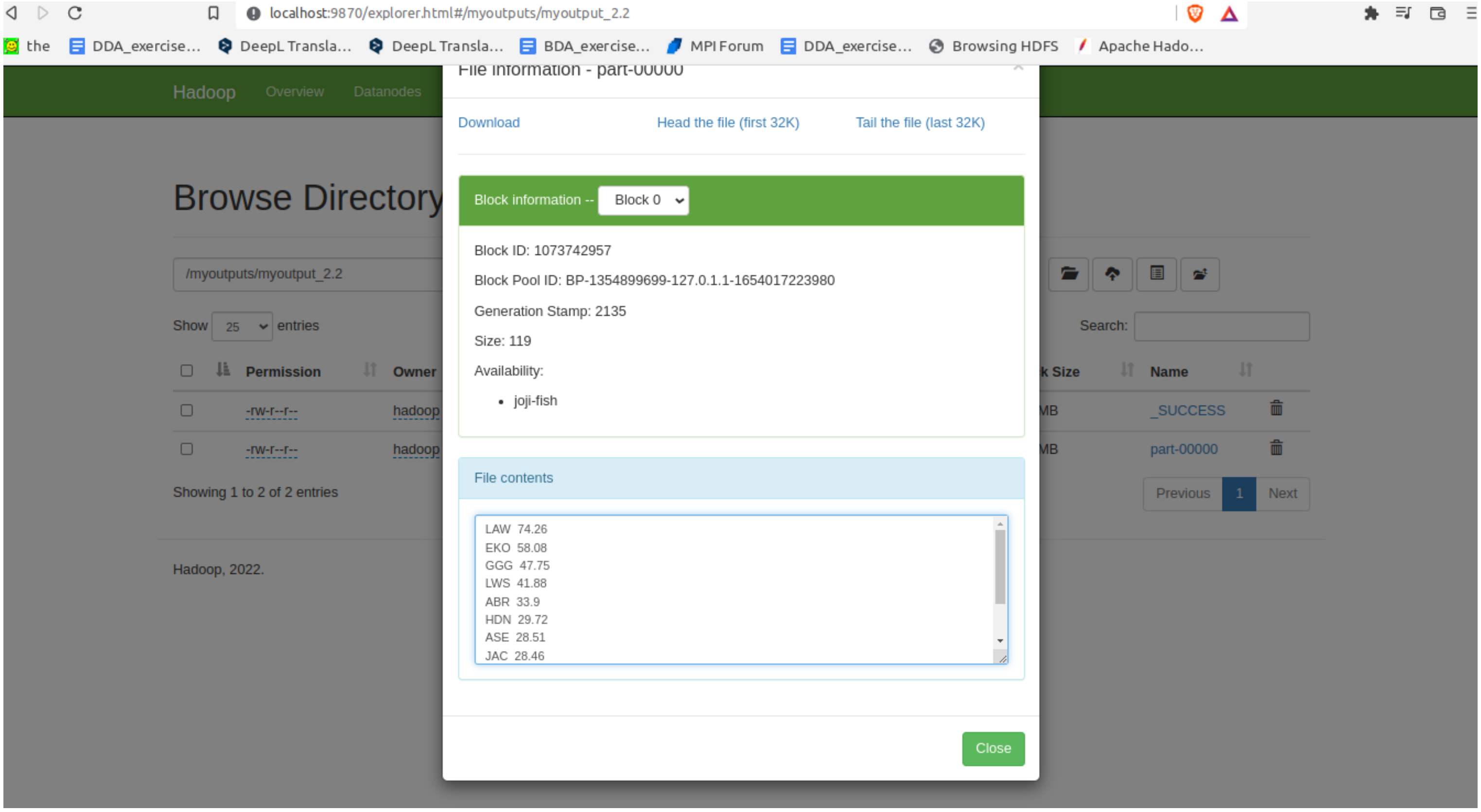
Question 2

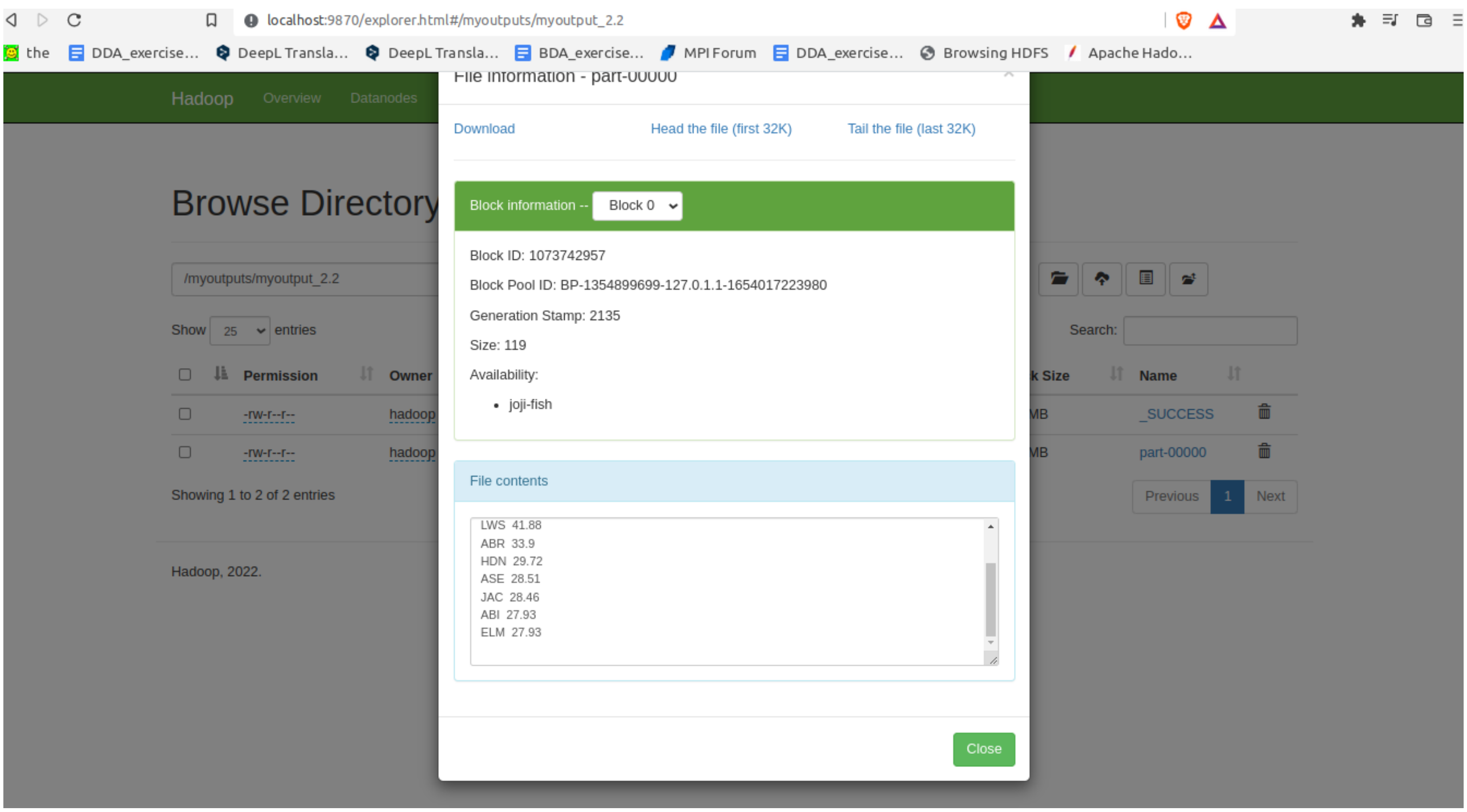
1. Computing the maximum, minimum, and average departure delay for each airport.[Hint: you are required to find max, min and avg in a single map reduce job]

The list is large, attaching the results from the Hadoop UI at **http://localhost:9870/**



2. Computing a ranking list that contains top 10 airports by their average Arrival delay.





3. What are your mapper.py and reduce.py solutions?
4. Describe step-by-step how you apply them and the outputs during this procedure.
Both of these questions are answered in the code explanation of 2.1 and 2.2 below.

Question 2.1 Code Explanation

1. Question 2 starts with downloading the required data and performing some preprocessing on it. The data contains missing values hence, for simplicity we replace missing values with 0 in the columns of interest.

mnt > Farjad_Ahmed > Masters > Distributed_Data_Analytics > Exercise_5 > scripts > 2.1 > datacleanse.py > ...

```python
import pandas as pd
def read_file(file):
    df = pd.read_csv(file, low_memory=False)
    return df

df = read_file('/mnt/Farjad_Ahmed/Masters/Distributed_Data_Analytics/Exercise_5/datafile.csv')
df['DepDelayMinutes'] = df['DepDelayMinutes'].fillna(0)
df['ArrDelayMinutes'] = df['ArrDelayMinutes'].fillna(0)
df['ArrDelay'] = df['ArrDelay'].fillna(0)
df['DepDelay'] = df['DepDelay'].fillna(0)
df.to_csv('cleansedData')
```

2. This csv is saved and then exported to hadoop using the command below.\

hdfs dfs -put -f /mnt/Farjad_Ahmed/Masters/Distributed_Data_Analytics/Exercise_5/scripts/2.1/cleansedData.csv /temp/cleansedData.csv

3. Now, we define the mapper script for this question. The script is simple: since the first line contains headers, hence next(sys.stdin) iterates to the next line skipping the header. Next, it takes the input from stdin and iterates over each line printing the desired columns to the output. In my case the columns used for this task, 2.1 were the **'Origin'** and **'DepDelay'**. Based on this the column numbers are printed from the data list that contains stripped and split rows by ',' since it is a comma separated file.

amapper2.1.py
Farjad_Ahmed /mnt/Farjad_Ahmed/Masters/Dist…buted_Data_Analytics/Ex…

```python
#!/usr/bin/python3

import sys
import re

next(sys.stdin)

for line in sys.stdin:
    data = line.strip().split(',')
    print(data[15],'\t',data[34])
```

4. Next, the reducer for this task is defined. The reducer receives stream from the mapper, it works by creating a dictionary and storing each new origin, while the dep_delay goes into the value as a list. For each line incoming the dictionary is checked if that key already exists, if it does then the dep_delay is appended. Else, a new list is created with dep_delay as the first element, and the value for that key is set as this list.

```python
#!/usr/bin/python3
# -*- coding: utf-8 -*-
import sys

mydict1 = {}
for line in sys.stdin:
    origin, dep_delay = line.strip().split('\t')
    dep_delay = float(dep_delay)
    #print(origin, dep_delay)

    if origin in mydict1:
        mydict1[origin].append(dep_delay)
    else:
        ddelay_list = [dep_delay]
        mydict1[origin] = ddelay_list
for key in mydict1:
    print( key,' min:', min(mydict1[key]), 'max:', max(mydict1[key]), 'avg:', round(sum(mydict1[key])/len(mydict1[key]), 2))
```

5. Next, the dictionary contains each unique origin value as the key and all the associated dep_delay as the 'value' in the list container. Now, we can iterate over all elements in the dictionary using a for loop and use python's min and max functions to find the min and maximum departure delay respectively and for the avg departure delay, we sum this list and divide by its length. Lastly, for the ease of readability, the values are rounded to 2 decimal places using the round() function.

6. Now that the scripts are ready we can execute them. First we do it locally using cat command, then in hadoop.

7. Running through cat command we get the results as shown below.

```
hadoop@joji-fish:~$ hdfs dfs -cat /temp/cleansedData.csv | ~/scripts/amapper2.1.py | sort | ~/scripts/areducer2.1.py
2022-06-05 14:59:12,504 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
ABE   min: -11.0 max: 794.0 avg: 20.49
ABI   min: -11.0 max: 263.0 avg: 25.79
ABQ   min: -18.0 max: 911.0 avg: 8.55
ABR   min: -13.0 max: 1259.0 avg: 36.24
ABY   min: -21.0 max: 291.0 avg: 10.01
ACT   min: -17.0 max: 202.0 avg: 12.95
ACV   min: -17.0 max: 578.0 avg: 10.16
ACY   min: -21.0 max: 670.0 avg: 7.71
ADK   min: -34.0 max: 41.0 avg: 0.67
ADQ   min: -28.0 max: 73.0 avg: -6.07
AEX   min: -20.0 max: 354.0 avg: 11.55
AGS   min: -11.0 max: 1130.0 avg: 20.97
ALB   min: -21.0 max: 981.0 avg: 11.75
AMA   min: -15.0 max: 298.0 avg: 4.42
ANC   min: -48.0 max: 1195.0 avg: 3.62
APN   min: -19.0 max: 278.0 avg: 19.04
ASE   min: -20.0 max: 1110.0 avg: 31.64
ATL   min: -22.0 max: 1470.0 avg: 15.07
ATW   min: -16.0 max: 1065.0 avg: 24.85
AUS   min: -22.0 max: 1246.0 avg: 8.72
AVL   min: -14.0 max: 425.0 avg: 14.63
AVP   min: -18.0 max: 783.0 avg: 32.04
AZO   min: -20.0 max: 268.0 avg: 9.33
BDL   min: -20.0 max: 549.0 avg: 6.65
BET   min: -27.0 max: 72.0 avg: -2.48
BFL   min: -20.0 max: 290.0 avg: -2.52
BGM   min: -15.0 max: 216.0 avg: 9.02
BHM   min: -18.0 max: 1068.0 avg: 13.75
BIL   min: -19.0 max: 340.0 avg: 7.03
BIS   min: -18.0 max: 189.0 avg: 10.41
BJI   min: -15.0 max: 65.0 avg: 0.37
BLI   min: -21.0 max: 324.0 avg: 8.17
BMI   min: -18.0 max: 1458.0 avg: 29.52
BNA   min: -20.0 max: 1447.0 avg: 8.7
BOI   min: -22.0 max: 889.0 avg: 14.53
BOS   min: -28.0 max: 1545.0 avg: 9.02
BPT   min: -6.0 max: 129.0 avg: 22.86
BQK   min: -9.0 max: 343.0 avg: 23.52
BQN   min: -23.0 max: 351.0 avg: 9.96
```

8. Running this in mapreduce in hadoop using the command.

bin/hadoop jar /home/hadoop/hadoop-3.3.3/share/hadoop/tools/lib/hadoop-streaming-3.3.3.jar -file ~/scripts/amapper2.1.py -mapper ~/scripts/amapper2.1.py -file ~/scripts/areducer2.1.py -reducer ~/scripts/areducer2.1.py -input /temp/cleansedData.csv -output /myoutputs/myoutput_2.1

```
Activities   Terminal   So Jun 5 15:08
hadoop@joji-fish: ~/hadoop-3.3.3
2022-06-05 15:00:22,899 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
packageJobJar: [/home/hadoop/scripts/amapper2.1.py, /home/hadoop/scripts/areducer2.1.py, /tmp/hadoop-unjar1765097024505880352 8/] [] /tmp/streamjob1201611387506722 6836.jar tmpDir=null
2022-06-05 15:00:23,479 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
2022-06-05 15:00:23,620 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
2022-06-05 15:00:23,792 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging/job_1654430975230_0006
2022-06-05 15:00:24,019 INFO mapred.FileInputFormat: Total input files to process : 1
2022-06-05 15:00:24,029 INFO net.NetworkTopology: Adding a new node: /default-rack/127.0.0.1:9866
2022-06-05 15:00:24,063 INFO mapreduce.JobSubmitter: number of splits:2
2022-06-05 15:00:24,245 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1654430975230_0006
2022-06-05 15:00:24,245 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-05 15:00:24,386 INFO conf.Configuration: resource-types.xml not found
2022-06-05 15:00:24,386 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-06-05 15:00:24,434 INFO impl.YarnClientImpl: Submitted application application_1654430975230_0006
2022-06-05 15:00:24,460 INFO mapreduce.Job: The url to track the job: http://joji-fish:8088/proxy/application_1654430975230_0006/
2022-06-05 15:00:24,461 INFO mapreduce.Job: Running job: job_1654430975230_0006
2022-06-05 15:00:29,522 INFO mapreduce.Job: Job job_1654430975230_0006 running in uber mode : false
2022-06-05 15:00:29,523 INFO mapreduce.Job:  map 0% reduce 0%
2022-06-05 15:00:35,571 INFO mapreduce.Job:  map 50% reduce 0%
2022-06-05 15:00:36,577 INFO mapreduce.Job:  map 100% reduce 0%
2022-06-05 15:00:40,596 INFO mapreduce.Job:  map 100% reduce 100%
2022-06-05 15:00:40,602 INFO mapreduce.Job: Job job_1654430975230_0006 completed successfully
2022-06-05 15:00:40,666 INFO mapreduce.Job: Counters: 55
        File System Counters
                FILE: Number of bytes read=5802897
                FILE: Number of bytes written=12443513
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=165698015
                HDFS: Number of bytes written=11807
                HDFS: Number of read operations=11
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=1
                Rack-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=7546
                Total time spent by all reduces in occupied slots (ms)=2708
                Total time spent by all map tasks (ms)=7546
                Total time spent by all reduce tasks (ms)=2708
                Total vcore-milliseconds taken by all map tasks=7546
                Total vcore-milliseconds taken by all reduce tasks=2708
                Total megabyte-milliseconds taken by all map tasks=7727104
                Total megabyte-milliseconds taken by all reduce tasks=2772992
        Map-Reduce Framework
                Map input records=450018
                Map output records=450016
                Map output bytes=4902859
                Map output materialized bytes=5802903
                Input split bytes=190
                Combine input records=0
                Combine output records=0
                Reduce input groups=298
                Reduce shuffle bytes=5802903
                Reduce input records=450016
                Reduce output records=298
                Spilled Records=900032
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=92
                CPU time spent (ms)=5820
                Physical memory (bytes) snapshot=866848768
                Virtual memory (bytes) snapshot=8228757504
                Total committed heap usage (bytes)=792723456
                Peak Map Physical memory (bytes)=316878848
                Peak Map Virtual memory (bytes)=2741772288
                Peak Reduce Physical memory (bytes)=240762880
                Peak Reduce Virtual memory (bytes)=2745769984
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=165697825
        File Output Format Counters
                Bytes Written=11807
2022-06-05 15:00:40,666 INFO streaming.StreamJob: Output directory: /myoutputs/myoutput_2.1
```

Question 2.2 Code Explanation

1. For this question the logic remains the same for the mapper, the columns considered here are 'Origin' and 'ArrDelay'.

```
#!/usr/bin/python3

import sys
import re

next(sys.stdin)

for line in sys.stdin:
  data = line.strip().split(',')
  print(data[15],'\t',data[45])
```

2. For this question the reducer is similar to the one in 2.1 but some parts are different. The reducer receives stream from the mapper, it works by creating a dictionary and storing each new origin, while the arr_delay goes into the value as a list. For each line incoming the dictionary is checked if that key already exists, if it does then the arr_delay is appended. Else, a new list is created with arr_delay as the first element, and the value for that key is set as this list. Next, when the dictionary is populated fully, a food loop is used to iterate over each key of the dictionary and replace the value which was a list, by the average of that list, as a result we get average arrival delay times for each origin value. This is rounded to 2 decimal places for better

readability. Now since we are supposed to rank the airports, we need to sort them. DIctionaries cannot be sorted, so a tuple is created using the key. Value pairs with the tuple containing (key, value), and this is appended to a tuple list called tuplist. This list can be sorted using the sort() function of python by feeding the lambda parameter and specifying the tuple element index to be considered for sorting, through y:y[1]. The list is sorted in descending order by specifying reverse = True, this was the largest arrival delay will be the first element of this list and the smallest being the last one. Lastly, a for loop prints the top 10 elements of this list.

```python
#!/usr/bin/python3
# -*- coding: utf-8 -*-
import sys

mydict1 = {}
tuplist = []

for line in sys.stdin:
    origin, arr_delay = line.strip().split('\t')
    arr_delay = float(arr_delay)

    if origin in mydict1:
        mydict1[origin].append(arr_delay)
    else:
        ddelay_list = []
        ddelay_list.append(arr_delay)
        mydict1[origin] = ddelay_list
for key in mydict1:
    mydict1[key] = round(sum(mydict1[key])/len(mydict1[key]), 2)
    tuplist.append(tuple((key, mydict1[key])))
    tuplist.sort(key=lambda y: y[1], reverse=True)

for i in range(10):
    print(tuplist[i][0], tuplist[i][1])
```

Saving file "admin:///home/hadoop/scripts/areducer2.2.py"... Python Tab Width: 4 Ln 8, Col 23 INS

3. Running this locally using the command below.

hdfs dfs -cat /temp/cleansedData.csv | ~/scripts/amapper2.2.py | sort | ~/scripts/areducer2.2.py

```
hadoop@joji-fish:~$ hdfs dfs -cat /temp/cleansedData.csv | ~/scripts/amapper2.2.py | sort | ~/scripts/areducer2.2.py
2022-06-05 15:42:33,665 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
LAW  74.26
EKO  58.08
GGG  47.75
LWS  41.88
ABR  33.9
HDN  29.72
ASE  28.51
JAC  28.46
ABI  27.93
ELM  27.93
```

4. Running this in hadoop, using the command below.

bin/hadoop jar /home/hadoop/hadoop-3.3.3/share/hadoop/tools/lib/hadoop-streaming-3.3.3.jar -file ~/scripts/amapper2.2.py -mapper ~/scripts/amapper2.2.py -file ~/scripts/areducer2.2.py -reducer ~/scripts/areducer2.2.py -input /temp/cleansedData.csv -output /myoutputs/myoutput_2.2

```
Activities    Terminal                So Jun 5 15:44                en    100 %
hadoop@joji-fish: ~/hadoop-3.3.3
2022-06-05 15:34:12,951 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
packageJobJar: [/home/hadoop/scripts/amapper2.2.py, /home/hadoop/scripts/areducer2.2.py, /tmp/hadoop-unjar17197627393712617779/] [] /tmp/streamjob16931052026614734265.jar tmpDir=null
2022-06-05 15:34:13,580 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
2022-06-05 15:34:13,719 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
2022-06-05 15:34:13,884 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging/job_1654430975230_0007
2022-06-05 15:34:14,135 INFO mapred.FileInputFormat: Total input files to process : 1
2022-06-05 15:34:14,145 INFO net.NetworkTopology: Adding a new node: /default-rack/127.0.0.1:9866
2022-06-05 15:34:14,181 INFO mapreduce.JobSubmitter: number of splits:2
2022-06-05 15:34:14,333 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1654430975230_0007
2022-06-05 15:34:14,333 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-05 15:34:14,469 INFO conf.Configuration: resource-types.xml not found
2022-06-05 15:34:14,469 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-06-05 15:34:14,522 INFO impl.YarnClientImpl: Submitted application application_1654430975230_0007
2022-06-05 15:34:14,548 INFO mapreduce.Job: The url to track the job: http://joji-fish:8088/proxy/application_1654430975230_0007/
2022-06-05 15:34:14,549 INFO mapreduce.Job: Running job: job_1654430975230_0007
2022-06-05 15:34:19,659 INFO mapreduce.Job: Job job_1654430975230_0007 running in uber mode : false
2022-06-05 15:34:19,660 INFO mapreduce.Job:  map 0% reduce 0%
2022-06-05 15:34:25,709 INFO mapreduce.Job:  map 50% reduce 0%
2022-06-05 15:34:26,714 INFO mapreduce.Job:  map 100% reduce 0%
2022-06-05 15:34:31,736 INFO mapreduce.Job:  map 100% reduce 100%
2022-06-05 15:34:31,744 INFO mapreduce.Job: Job job_1654430975230_0007 completed successfully
2022-06-05 15:34:31,843 INFO mapreduce.Job: Counters: 55
        File System Counters
                FILE: Number of bytes read=5959322
                FILE: Number of bytes written=12756363
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=165698015
                HDFS: Number of bytes written=119
                HDFS: Number of read operations=11
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=1
                Rack-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=7554
                Total time spent by all reduces in occupied slots (ms)=2715
                Total time spent by all map tasks (ms)=7554
                Total time spent by all reduce tasks (ms)=2715
                Total vcore-milliseconds taken by all map tasks=7554
                Total vcore-milliseconds taken by all reduce tasks=2715
                Total megabyte-milliseconds taken by all map tasks=7735296
                Total megabyte-milliseconds taken by all reduce tasks=2780160
        Map-Reduce Framework
                Map input records=450018
                Map output records=450016
                Map output bytes=5059284
                Map output materialized bytes=5959328
                Input split bytes=190
                Combine input records=0
                Combine output records=0
                Reduce input groups=298
                Reduce shuffle bytes=5959328
                Reduce input records=450016
                Reduce output records=10
                Spilled Records=900032
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=67
                CPU time spent (ms)=6160
                Physical memory (bytes) snapshot=869199872
                Virtual memory (bytes) snapshot=8222535680
                Total committed heap usage (bytes)=792723456
                Peak Map Physical memory (bytes)=314388480
                Peak Map Virtual memory (bytes)=2740137984
                Peak Reduce Physical memory (bytes)=242741248
                Peak Reduce Virtual memory (bytes)=2743242752
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=165697825
        File Output Format Counters
                Bytes Written=119
2022-06-05 15:34:31,843 INFO streaming.StreamJob: Output directory: /myoutputs/myoutput_2.2
```

Question 3

This question will contain the answers to the question along with results screenshots, followed by a detailed explanation of mapper and reducer scripts for each part.

Since the data contained two files that needed to be merged, movies.dat and ratings.dat. I will first describe the preprocessing on this data. The .dat files were read using pd.read_csv. Columns of interest for this exercise are selected as a subset of the dataframe in df1 and df2 for movies and ratings respectively.
Next, pd.merge is used to join both the data frames using the ‘inner’ option in order to remove redundant columns with missing keys. The key used is ‘movieId’. This dataframe is then written to a csv as ‘10m_merged.csv’. This data will be used in all parts of this question.

mnt > Farjad_Ahmed > Masters > Distributed_Data_Analytics > Exercise_5 > scripts > 3.1 > 10merged.py > ...

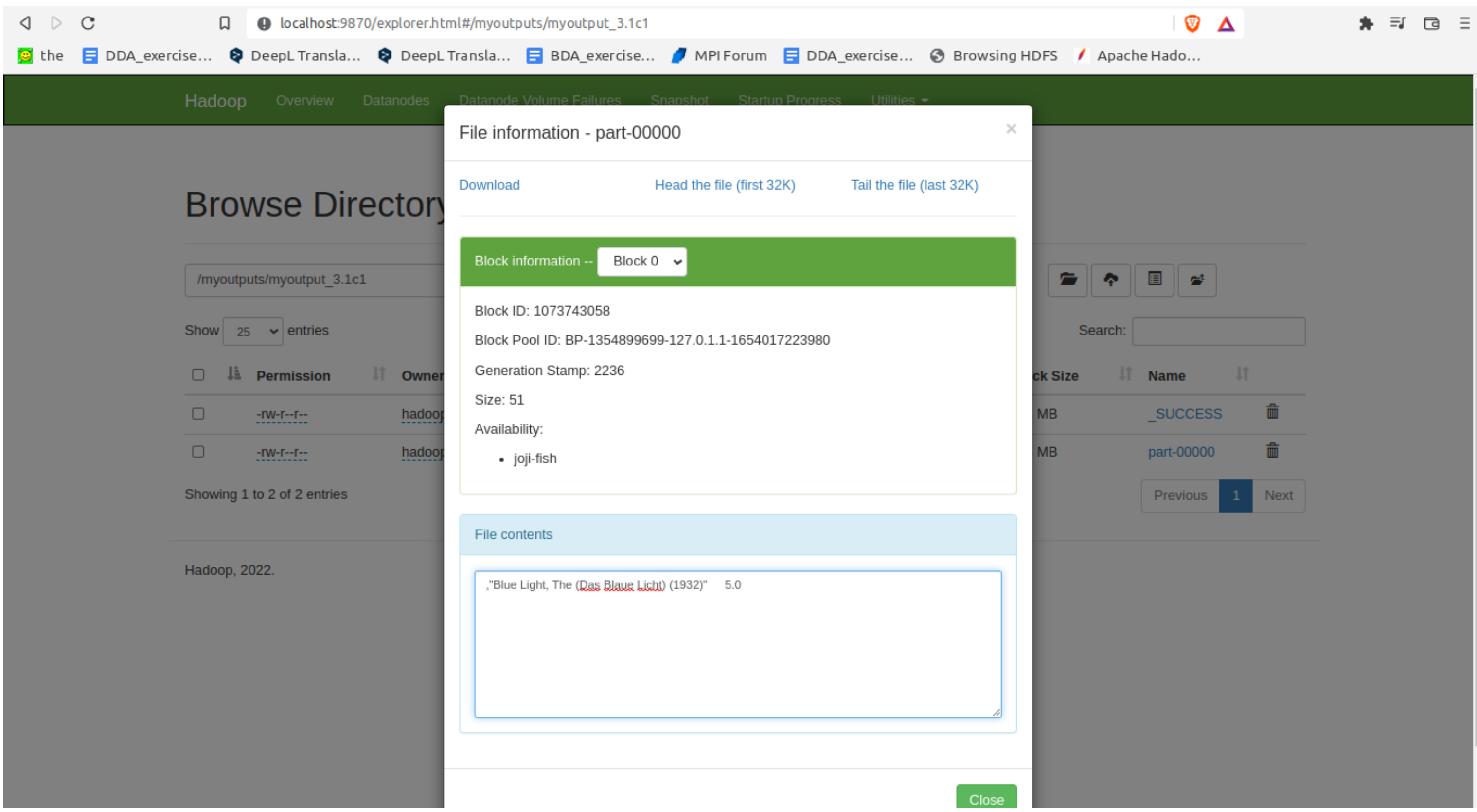
```python
from turtle import home
import pandas as pd

df1 = pd.read_csv('/mnt/Farjad_Ahmed/Masters/Distributed_Data_Analytics/Exercise_5/ml-10m/ml-10M100K/movies.dat', sep='::', engine='python')
df1 = pd.DataFrame({'movieId':df1.iloc[:,0],'title':df1.iloc[:,1], 'genres':df1.iloc[:,2]})
df2 = pd.read_csv('/mnt/Farjad_Ahmed/Masters/Distributed_Data_Analytics/Exercise_5/ml-10m/ml-10M100K/ratings.dat',sep='::', engine='python')
df2 = pd.DataFrame({'userId':df2.iloc[:,0],'movieId':df2.iloc[:,1], 'rating':df2.iloc[:,2], 'timestamp':df2.iloc[:,3]})
df2.drop(['timestamp'], axis=1, inplace=True)
df3 = pd.merge(df1, df2, on=['movieId'], how='inner')
df3 = df3[['title','genres', 'movieId', 'rating', 'userId']]
df3.to_csv('10m_merged.csv', index=False, sep=',')
```

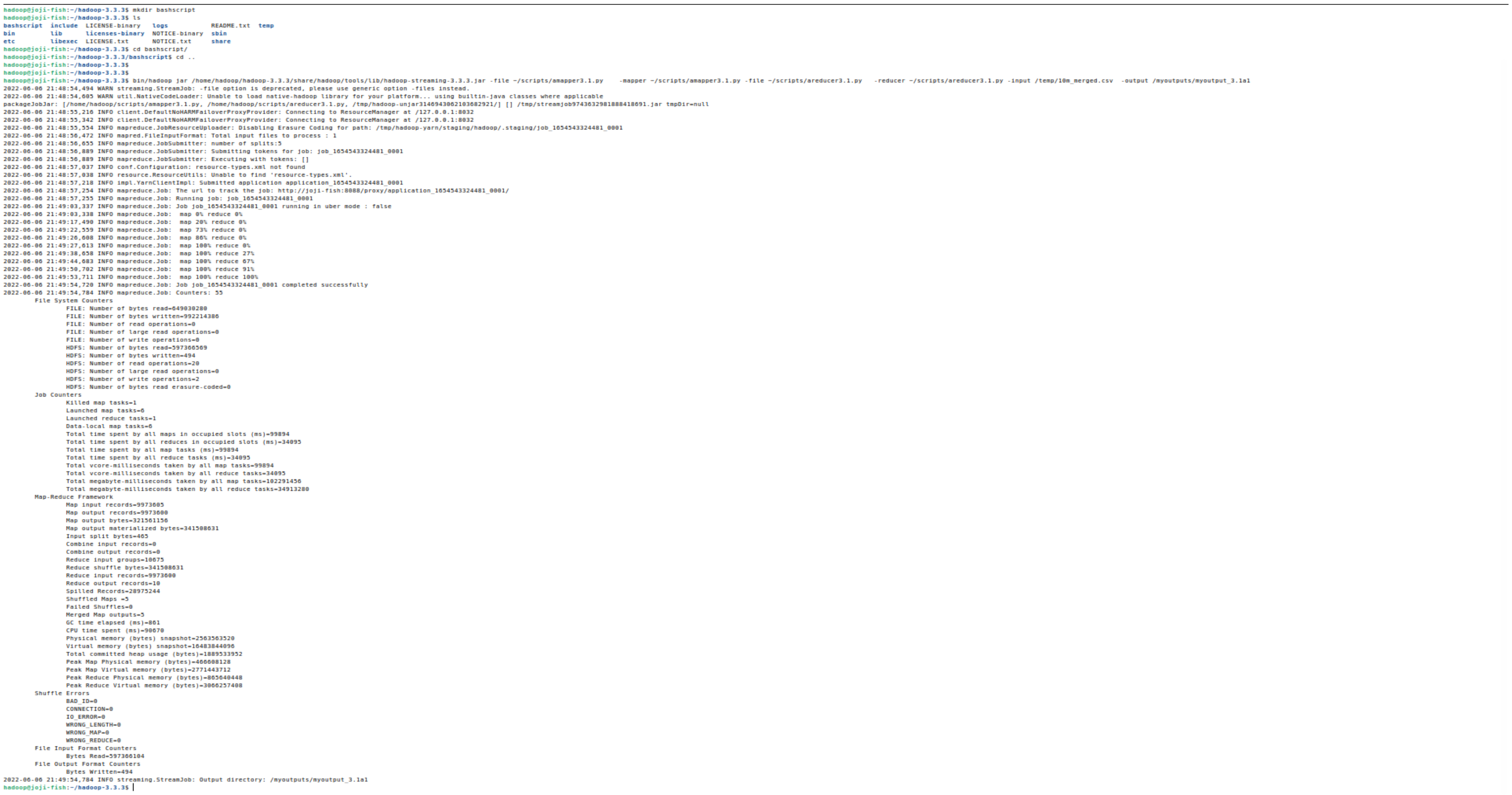
From this the acquired data set contains repeating movie titles for each user’s rating, which would be required later in a question to be grouped.
The dataset looks like:

title	genres	movieId	rating	userId
Jumanji (1995)	Adventure|Children|Fantasy	2	2.5	8
Jumanji (1995)	Adventure|Children|Fantasy	2	3	13
Jumanji (1995)	Adventure|Children|Fantasy	2	3	14
Jumanji (1995)	Adventure|Children|Fantasy	2	3	18
Jumanji (1995)	Adventure|Children|Fantasy	2	3	34
Jumanji (1995)	Adventure|Children|Fantasy	2	3	36
Jumanji (1995)	Adventure|Children|Fantasy	2	3.5	47
Jumanji (1995)	Adventure|Children|Fantasy	2	2.5	56
Jumanji (1995)	Adventure|Children|Fantasy	2	3	65
Jumanji (1995)	Adventure|Children|Fantasy	2	3	77
Jumanji (1995)	Adventure|Children|Fantasy	2	4	91
Jumanji (1995)	Adventure|Children|Fantasy	2	4	101
Jumanji (1995)	Adventure|Children|Fantasy	2	5	116
Jumanji (1995)	Adventure|Children|Fantasy	2	2.5	126

1. Find the movie title which has the maximum average rating?

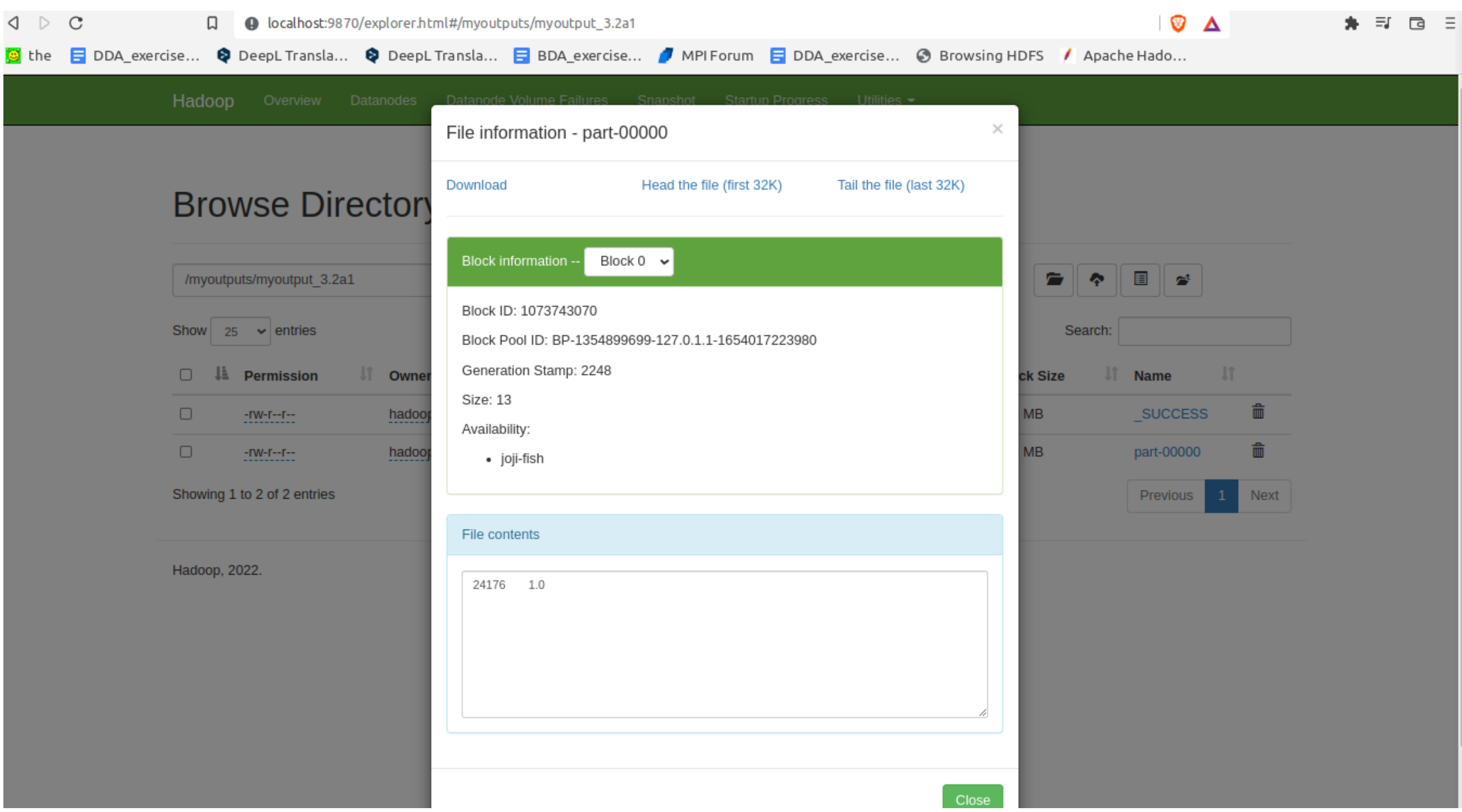
bin/hadoop jar /home/hadoop/hadoop-3.3.3/share/hadoop/tools/lib/hadoop-streaming-3.3.3.jar -file ~/scripts/amapper3.1.py -mapper ~/scripts/amapper3.1.py -file ~/scripts/areducer3.1.py -reducer ~/scripts/areducer3.1.py -input /temp/10m_merged.csv -output /myoutputs/myoutput_3.1a1





2. Find the user who has assign lowest average rating among all the users who rated more than 40 Times?

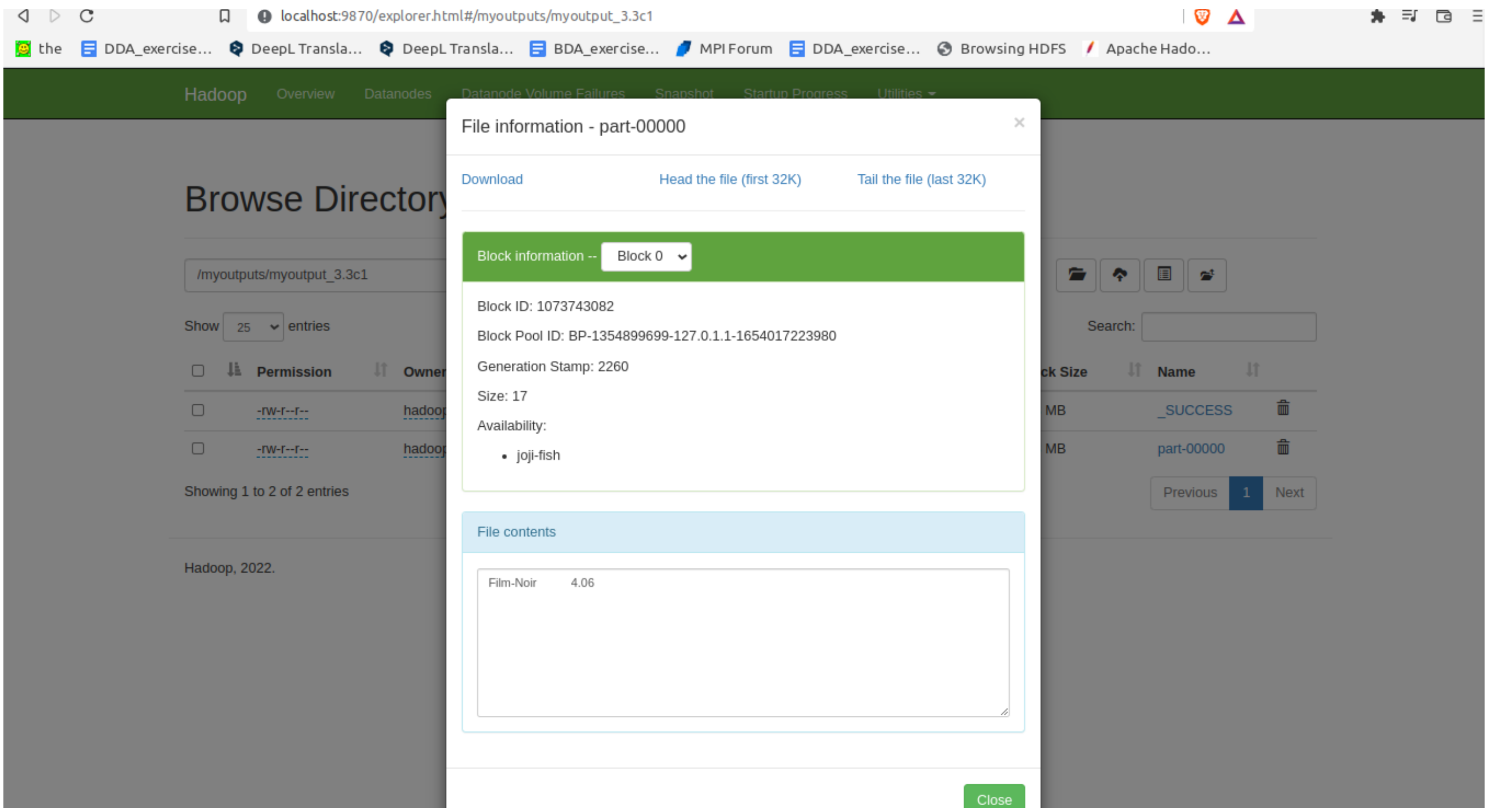
bin/hadoop jar /home/hadoop/hadoop-3.3.3/share/hadoop/tools/lib/hadoop-streaming-3.3.3.jar -file ~/scripts/amapper3.2.py   -mapper ~/scripts/amapper3.2.py -file ~/scripts/areducer3.2.py   -reducer ~/scripts/areducer3.2.py -input /temp/10m_merged.csv -output /myoutputs/myoutput_3.2a1



```
hadoop@joji-fish:~/hadoop-3.3.3$ bin/hadoop jar /home/hadoop/hadoop-3.3.3/share/hadoop/tools/lib/hadoop-streaming-3.3.3.jar -file ~/scripts/amapper3.2.py    -mapper ~/scripts/amapper3.2.py -file ~/scripts/areducer3.2.py   -reducer ~/scripts/areducer3.2.py -input /temp/10m_merged.csv  -output /myoutputs/myoutput_3.2a1
2022-06-06 22:27:46,505 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
2022-06-06 22:27:46,648 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
packageJobJar: [/home/hadoop/scripts/amapper3.2.py, /home/hadoop/scripts/areducer3.2.py, /tmp/hadoop-unjar1056927679655940898 5/] [] /tmp/streamjob7075506791891686064.jar tmpDir=null
2022-06-06 22:27:47,227 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
2022-06-06 22:27:47,351 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
2022-06-06 22:27:47,530 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging/job_1654543324481_0005
2022-06-06 22:27:48,063 INFO mapred.FileInputFormat: Total input files to process : 1
2022-06-06 22:27:48,215 INFO mapreduce.JobSubmitter: number of splits:5
2022-06-06 22:27:48,387 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1654543324481_0005
2022-06-06 22:27:48,387 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-06 22:27:48,499 INFO conf.Configuration: resource-types.xml not found
2022-06-06 22:27:48,499 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-06-06 22:27:48,544 INFO impl.YarnClientImpl: Submitted application application_1654543324481_0005
2022-06-06 22:27:48,567 INFO mapreduce.Job: The url to track the job: http://joji-fish:8088/proxy/application_1654543324481_0005/
2022-06-06 22:27:48,568 INFO mapreduce.Job: Running job: job_1654543324481_0005
2022-06-06 22:27:53,625 INFO mapreduce.Job: Job job_1654543324481_0005 running in uber mode : false
2022-06-06 22:27:53,625 INFO mapreduce.Job:  map 0% reduce 0%
2022-06-06 22:28:06,788 INFO mapreduce.Job:  map 20% reduce 0%
2022-06-06 22:28:11,894 INFO mapreduce.Job:  map 33% reduce 0%
2022-06-06 22:28:12,913 INFO mapreduce.Job:  map 73% reduce 0%
2022-06-06 22:28:13,922 INFO mapreduce.Job:  map 93% reduce 0%
2022-06-06 22:28:14,926 INFO mapreduce.Job:  map 100% reduce 0%
2022-06-06 22:28:26,969 INFO mapreduce.Job:  map 100% reduce 84%
2022-06-06 22:28:31,987 INFO mapreduce.Job:  map 100% reduce 100%
2022-06-06 22:28:31,993 INFO mapreduce.Job: Job job_1654543324481_0005 completed successfully
2022-06-06 22:28:32,054 INFO mapreduce.Job: Counters: 55
        File System Counters
                FILE: Number of bytes read=138129196
                FILE: Number of bytes written=277933891
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=597366569
                HDFS: Number of bytes written=13
                HDFS: Number of read operations=20
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Killed map tasks=1
                Launched map tasks=6
                Launched reduce tasks=1
                Data-local map tasks=6
                Total time spent by all maps in occupied slots (ms)=86884
                Total time spent by all reduces in occupied slots (ms)=22751
                Total time spent by all map tasks (ms)=86884
                Total time spent by all reduce tasks (ms)=22751
                Total vcore-milliseconds taken by all map tasks=86884
                Total vcore-milliseconds taken by all reduce tasks=22751
                Total megabyte-milliseconds taken by all map tasks=88969216
                Total megabyte-milliseconds taken by all reduce tasks=23297024
        Map-Reduce Framework
                Map input records=9973605
                Map output records=9973600
                Map output bytes=118181990
                Map output materialized bytes=138129220
                Input split bytes=465
                Combine input records=0
                Combine output records=0
                Reduce input groups=69878
                Reduce shuffle bytes=138129220
                Reduce input records=9973600
                Reduce output records=1
                Spilled Records=19947200
                Shuffled Maps =5
                Failed Shuffles=0
                Merged Map outputs=5
                GC time elapsed (ms)=909
                CPU time spent (ms)=63380
                Physical memory (bytes) snapshot=2392104960
                Virtual memory (bytes) snapshot=16494272512
                Total committed heap usage (bytes)=1694498816
                Peak Map Physical memory (bytes)=445325312
                Peak Map Virtual memory (bytes)=2764386304
                Peak Reduce Physical memory (bytes)=562900992
                Peak Reduce Virtual memory (bytes)=3004375040
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=597366104
        File Output Format Counters
                Bytes Written=13
2022-06-06 22:28:32,054 INFO streaming.StreamJob: Output directory: /myoutputs/myoutput_3.2a1
```

3. Find the highest average rated movie genre? [Hint: you may need to merge/combine movie.dat and rating.dat in a preprocessing step.]

bin/hadoop jar /home/hadoop/hadoop-3.3.3/share/hadoop/tools/lib/hadoop-streaming-3.3.3.jar -file ~/scripts/amapper3.3.py   -mapper ~/scripts/amapper3.3.py -file ~/scripts/areducer3.3.py   -reducer ~/scripts/areducer3.3.py -input /temp/10m_merged.csv -output /myoutputs/myoutput_3.3c1



```
hadoop@joji-fish:~/hadoop-3.3.3$ bin/hadoop jar /home/hadoop/hadoop-3.3.3/share/hadoop/tools/lib/hadoop-streaming-3.3.3.jar -file ~/scripts/amapper3.2.py    -mapper ~/scripts/amapper3.2.py -file ~/scripts/areducer3.2.py   -reducer ~/scripts/areducer3.2.py -input /temp/10m_merged.csv  -output /myoutputhadoop@joji-fish:~/hadoop-3.3.3$ bin/hadoop jar /home/hadoop/hadoop-3.3.3/share
/hadoop/tools/lib/hadoop-streaming-3.3.3.jar -file ~/scripts/amapper3.3.py    -mapper ~/scripts/amapper3.3.py -file ~/scripts/areducer3.3.py   -reducer ~/scripts/areducer3.3.py -input /temp/10m_merged.csv  -output /myoutputs/myoutput_3.3c1
2022-06-06 22:38:08,923 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
2022-06-06 22:38:09,016 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
packageJobJar: [/home/hadoop/scripts/amapper3.3.py, /home/hadoop/scripts/areducer3.3.py, /tmp/hadoop-unjar14478246454473697769/] [] /tmp/streamjob3524751719979576715.jar tmpDir=null
2022-06-06 22:38:09,550 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
2022-06-06 22:38:09,688 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /127.0.0.1:8032
2022-06-06 22:38:09,876 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging/job_1654543324481_0006
2022-06-06 22:38:10,317 INFO mapred.FileInputFormat: Total input files to process : 1
2022-06-06 22:38:10,472 INFO mapreduce.JobSubmitter: number of splits:5
2022-06-06 22:38:10,671 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1654543324481_0006
2022-06-06 22:38:10,671 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-06 22:38:10,790 INFO conf.Configuration: resource-types.xml not found
2022-06-06 22:38:10,790 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-06-06 22:38:10,834 INFO impl.YarnClientImpl: Submitted application application_1654543324481_0006
2022-06-06 22:38:10,856 INFO mapreduce.Job: The url to track the job: http://joji-fish:8088/proxy/application_1654543324481_0006/
2022-06-06 22:38:10,857 INFO mapreduce.Job: Running job: job_1654543324481_0006
2022-06-06 22:38:16,920 INFO mapreduce.Job: Job job_1654543324481_0006 running in uber mode : false
2022-06-06 22:38:16,921 INFO mapreduce.Job:  map 0% reduce 0%
2022-06-06 22:38:30,088 INFO mapreduce.Job:  map 20% reduce 0%
2022-06-06 22:38:36,201 INFO mapreduce.Job:  map 47% reduce 0%
2022-06-06 22:38:37,208 INFO mapreduce.Job:  map 74% reduce 0%
2022-06-06 22:38:40,230 INFO mapreduce.Job:  map 93% reduce 0%
2022-06-06 22:38:41,234 INFO mapreduce.Job:  map 100% reduce 0%
2022-06-06 22:38:52,281 INFO mapreduce.Job:  map 100% reduce 47%
2022-06-06 22:38:58,305 INFO mapreduce.Job:  map 100% reduce 75%
2022-06-06 22:39:04,332 INFO mapreduce.Job:  map 100% reduce 88%
2022-06-06 22:39:10,356 INFO mapreduce.Job:  map 100% reduce 100%
2022-06-06 22:39:11,365 INFO mapreduce.Job: Job job_1654543324481_0006 completed successfully
2022-06-06 22:39:11,442 INFO mapreduce.Job: Counters: 55
        File System Counters
                FILE: Number of bytes read=518150248
                FILE: Number of bytes written=792574341
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=597366569
                HDFS: Number of bytes written=17
                HDFS: Number of read operations=20
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Killed map tasks=1
                Launched map tasks=6
                Launched reduce tasks=1
                Data-local map tasks=6
                Total time spent by all maps in occupied slots (ms)=103150
                Total time spent by all reduces in occupied slots (ms)=37103
                Total time spent by all map tasks (ms)=103150
                Total time spent by all reduce tasks (ms)=37103
                Total vcore-milliseconds taken by all map tasks=103150
                Total vcore-milliseconds taken by all reduce tasks=37103
                Total megabyte-milliseconds taken by all map tasks=105625600
                Total megabyte-milliseconds taken by all reduce tasks=37993472
        Map-Reduce Framework
                Map input records=9973605
                Map output records=9973600
                Map output bytes=252801388
                Map output materialized bytes=272748618
                Input split bytes=465
                Combine input records=0
                Combine output records=0
                Reduce input groups=797
                Reduce shuffle bytes=272748618
                Reduce input records=9973600
                Reduce output records=1
                Spilled Records=28975244
                Shuffled Maps =5
                Failed Shuffles=0
                Merged Map outputs=5
                GC time elapsed (ms)=960
                CPU time spent (ms)=82020
                Physical memory (bytes) snapshot=2354372608
                Virtual memory (bytes) snapshot=16459198464
                Total committed heap usage (bytes)=1673527296
                Peak Map Physical memory (bytes)=404701184
                Peak Map Virtual memory (bytes)=2754449408
                Peak Reduce Physical memory (bytes)=790032384
                Peak Reduce Virtual memory (bytes)=3112448000
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=597366104
        File Output Format Counters
                Bytes Written=17
2022-06-06 22:39:11,442 INFO streaming.StreamJob: Output directory: /myoutputs/myoutput_3.3c1
```

Code Explanation

3.1

1. For this question the movie titles and their respective rating were required. This dataset contained an interesting challenge in the movie titles. First the header is skipped using next(sys.stdin). Then we need to read each title and its associated rating. Since it is a csv and we are splitting based on ‘,’ the title containing ‘,’ would also be separated. As per the file it contains 5 columns, wherever ‘,’s exist the length of ‘data’ will be more than 5. This case only exists for the title column. Hence, we will use negative indexing for rating for such cases while for data being of length 5 we can use either. While in case any data length is less than 5, it means there is a missing value so we skip it. For rows containing length > 5, we concatenate these titles and print the title along with rating at -2 index.

```
#!/usr/bin/python3

import sys
import re
next(sys.stdin)
title = ''
for line in sys.stdin:
    data = line.strip().split(',')
    length = len(data)
    if length < 5:
        next(sys.stdin)
    elif length == 5:
        print(data[0],'\t',data[3])
    elif length >5:
        for i in range((length-5)+1):
            title = title+','+data[i]
        print(title,'\t',data[-2])
        title = ''
```

The reducer is similar to as explained in question 2, except that instead of printing the list of top ranked n items, the algorithm takes in the ratings and movie titles and does the same calculations and sorting on the list of tuples but at the end, prints the 0th element of the list.

```python
#!/usr/bin/python3
# -*- coding: utf-8 -*-
import sys

last_movie = None
mydict1 = {}
tuplist = []

for line in sys.stdin:
    movie, rating = line.strip().split('\t')
    rating = float(rating)
    if movie in mydict1:
        mydict1[movie].append(rating)
    else:
        rating_list = []
        rating_list.append(rating)
        mydict1[movie] = rating_list
for key in mydict1:
    mydict1[key] = round(sum(mydict1[key])/len(mydict1[key]), 2)
    tuplist.append(tuple((key, mydict1[key])))
tuplist.sort(key=lambda y: y[1], reverse=True)

print(tuplist[0][0], '\t', tuplist[0][1])
```

3.2

1. The mapper for this question is simpler since it requires only the userId and the ratings, we can simply do this by using negative indices -1 and -2 for these items in the line, for the data list generated as a result of splitting based on ‘,’.

areducer3.1.py × amapper3.2.py ×

```python
#!/usr/bin/python3


import sys
import re
next(sys.stdin)

for line in sys.stdin:
    data = line.strip().split(',')
    print(data[-1],'\t',data[-2])
```

2. For this reducer the dictionary creation remains the same as preceding questions, i.e userId as key and ratings as a list in the value of the associated key. Next, we need to take only those users which have rated movies more than 40 times. Hence we condition out tuple creation and appending process for userId and

ratings based on the the len(<dictionary_value>) > 40, where dictionary value is a list containing ratings of that user. So all users with more than 40 ratings are filtered and shortlisted_users dictionary is created for the user key with the average of their ratings. This is then appended to the tuplist, which is then sorted in the ascending order. Lastly, the first element of this list is printed which is the lowest average rating of all the users who rated for more than 40 times.

areducer3.1.py | amapper3.2.py | areducer3.2.py

```
last_user = None
mydict1 = {}
tuplist = []
shortlisted_users = {}

for line in sys.stdin:
    user, rating = line.strip().split('\t')
    rating = float(rating)
    user = str(user)
    if user in mydict1:
        mydict1[user].append(rating)
    else:
        rating_list = []
        rating_list.append(rating)
        mydict1[user] = rating_list
for key in mydict1:
    getlist = mydict1[key]
    if len(getlist) > 40:
        shortlisted_users[key] = round(sum(mydict1[key])/len(mydict1[key]), 2)
        tuplist.append(tuple((key, shortlisted_users[key])))
tuplist.sort(key=lambda y: y[1], reverse=False)

print(tuplist[0][0], '\t', tuplist[0][1])
```

3.3

1. In this case mapper requires the genre column and the rating column, this can simply be taken using the negative indexing in order to prevent the indexing complexity if we count from 0, due to inconsistency of title splitting due to ‘,’, negative indexing solves this very simply.

amapper3.3.py

```
#!/usr/bin/python3


import sys
import re
next(sys.stdin)

for line in sys.stdin:
    data = line.strip().split(',')
    print(data[-4],'\t',data[-2])
```

2. In this case again, the dictionary creation logic remains the same but there is one major difference, the genre contains multiple genres separated by ‘|’, hence, these genres will be split and each will be counted with the cumulative rating that was assigned to the combined genre combination. For this we first attain the rating and the genre string from the sys.stdin. Then we split the genre string with ‘|’. A for loop loops over each genre after the split and creates a dictionary with key as the genre and value as the rating, as explained earlier.

When all lines are read and the dictionary is populated, a for loop is used to iterate over each key of this dictionary called mydict1, and the values of each genre is replaced by the average of the ratings for that genre. This is then appended to the tuplist. After this loop the tuplist contains all genres and their average ratings, this is sorted in descending order and the first element is printed which represents the highest rated genre.

amapper3.3.py × areducer3.3.py

```python
#!/usr/bin/python3
# -*- coding: utf-8 -*-
import sys

last_genre = None
mydict1 = {}
tuplist = []

for line in sys.stdin:
    genre, rating = line.strip().split('\t')
    rating = float(rating)
    parts = genre.split('|')

    for part in parts:
        genre = part
        if genre in mydict1:
            mydict1[genre].append(rating)
        else:
            rating_list = []
            rating_list.append(rating)
            mydict1[genre] = rating_list

for key in mydict1:
    mydict1[key] = round(sum(mydict1[key])/len(mydict1[key]), 2)
    tuplist.append(tuple((key, mydict1[key])))

tuplist.sort(key=lambda y: y[1], reverse=True)

print(tuplist[0][0], '\t', tuplist[0][1])
```

Performance

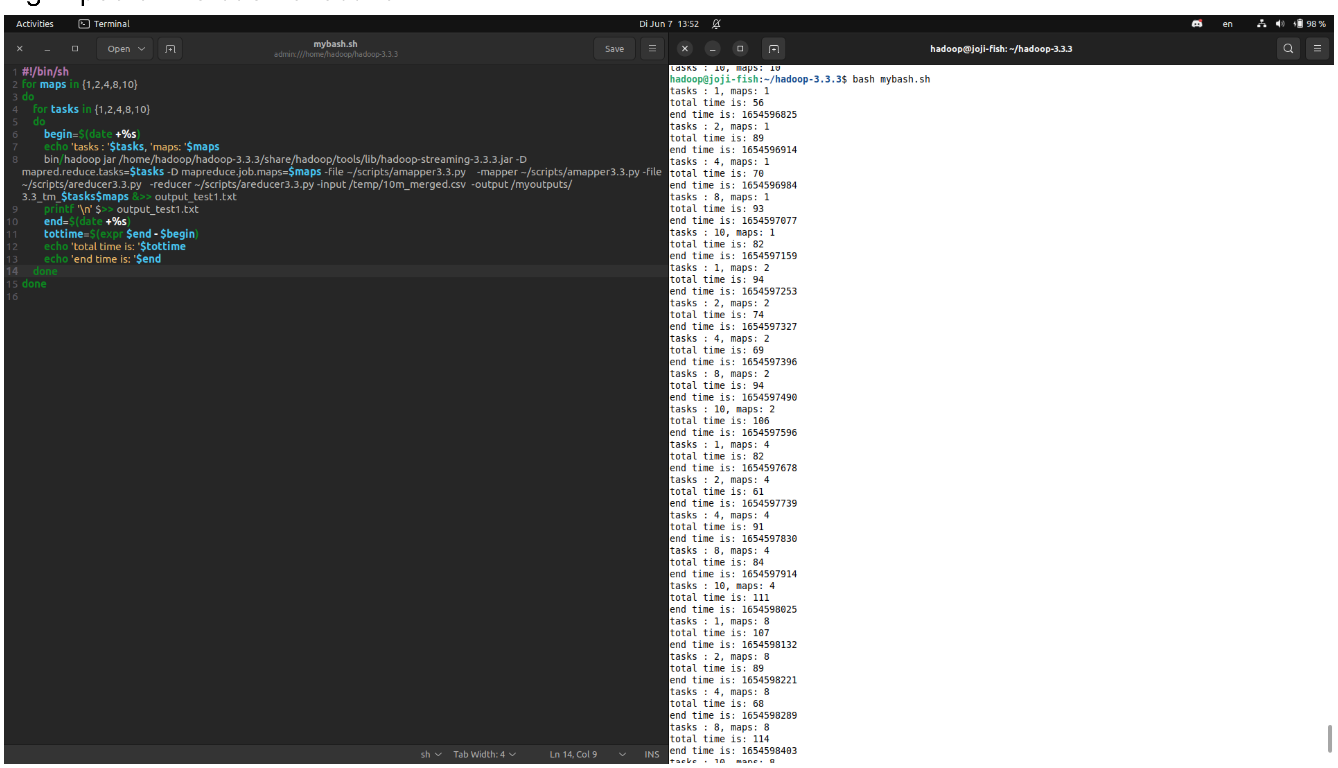
For the performance, I used a range of mappers and reducers to record running times on hadoop. **The factor under observation is how varying the number of mappers and reducers affect the performance.** This was done by automating the hadoop execution using a bash script. I'll explain the script first then share my conclusion on the performance.

The bash script iterates through two nested for loops, one for reducers and one for mappers. The params used for mappers and reducers are mapreduce.job.maps and mapred.reduce.tasks respectively. In the inner loop each loop starts with noting the time and echo-ing the mapper and reducers numbers. Next, the hadoop command is executed with dynamic food loop variables to generate the output files as well as to record the output in a text file through &>> parameter, that not only suppresses printing the output but also appends to the output_test1.txt file. After this, a line is separated through '\n' and the end time is noted. Total time is calculated by taking a difference of begin with end times. This is shown for 3.3, but changing the file name and output file names it was performed for 3.1 and 3.2 exactly the same way.

mybash.sh
admin:///home/hadoop/hadoop-3.3.3

```sh
#!/bin/sh
for maps in {1,2,4,8,10}
do
  for tasks in {1,2,4,8,10}
  do
    begin=$(date +%s)
    echo 'tasks : '$tasks, 'maps: '$maps
    bin/hadoop jar /home/hadoop/hadoop-3.3.3/share/hadoop/tools/lib/hadoop-streaming-3.3.3.jar -D mapred.reduce.tasks=$tasks -D mapreduce.job.maps=$maps -file ~/scripts/amapper3.3.py   -mapper ~/scripts/amapper3.3.py -file ~/scripts/areducer3.3.py  -reducer ~/scripts/areducer3.3.py -input /temp/10m_merged.csv  -output /myoutputs/3.3_tm_$tasks$maps &>> output_test1.txt
    printf '\n' $>> output_test1.txt
    end=$(date +%s)
    tottime=$(expr $end - $begin)
    echo 'total time is: '$tottime
    echo 'end time is: '$end
  done
done
```

A glimpse of the bash execution:

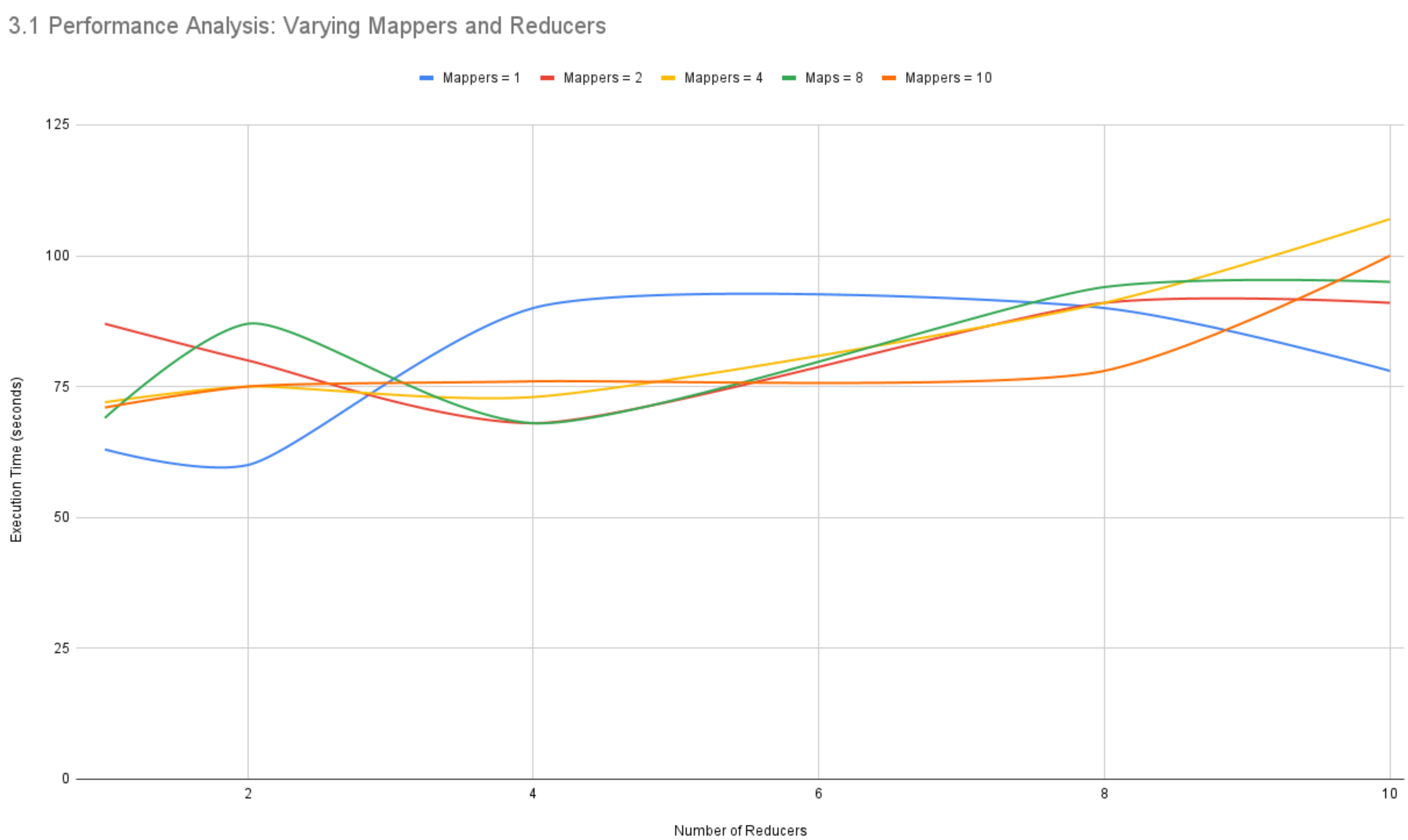


Conclusion

As far as the performance is concerned, I observed an increase in time with increasing reducers. However it is evident from the graph, **as we increase the number of mappers, the execution time decreases and has a positive performance improvement impact on the algorithm's performance**. This is evident by the orange line in the graph that bulges downward to the lowest point at reducer = 2. It does move upward with increasing reducers but shows the best performance time for a specific number of reducers. I think with data and the application this could be better determined so as to how many mappers and reducers can optimally improve the performance, keeping in mind the nodes available and the computational problem size at hand. This could also be due to processing required to share data among an increasing number of mappers and then combining data from multiple reducers. In an actual distributed system, network delays can play a significant role in delays.

3.1 Performance Graphs

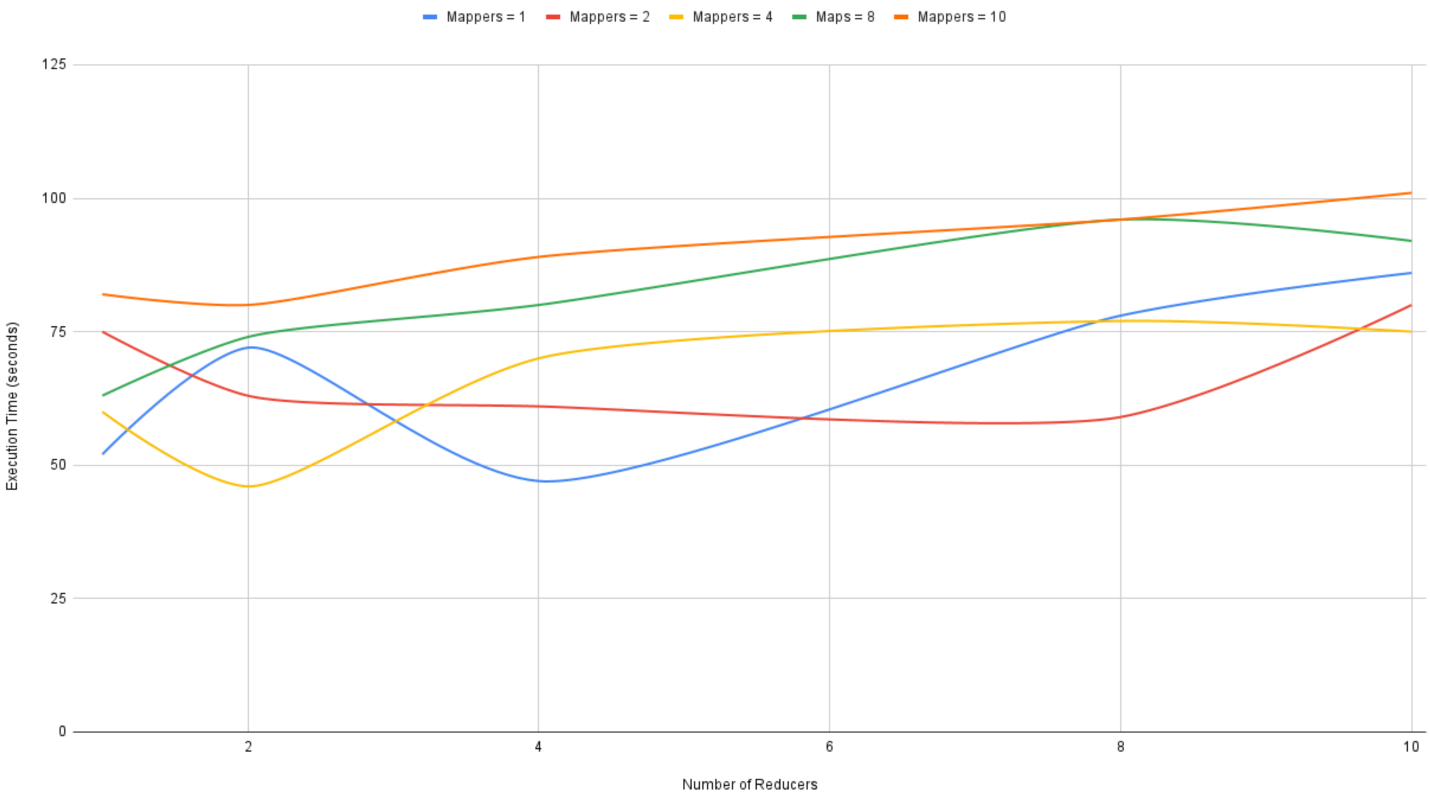
Farjad Ahmed, DDA 05, Performance Analysis									
Maps									
1		2		4		8		10	
Tasks	Time	Tasks	Time	Tasks	Time	Tasks	Time	Tasks	Time
1	63	1	87	1	72	1	69	1	71
2	60	2	80	2	75	2	87	2	75
4	90	4	68	4	73	4	68	4	76
8	90	8	91	8	91	8	94	8	78
10	78	10	91	10	107	10	95	10	100



3.2 Performance Graphs

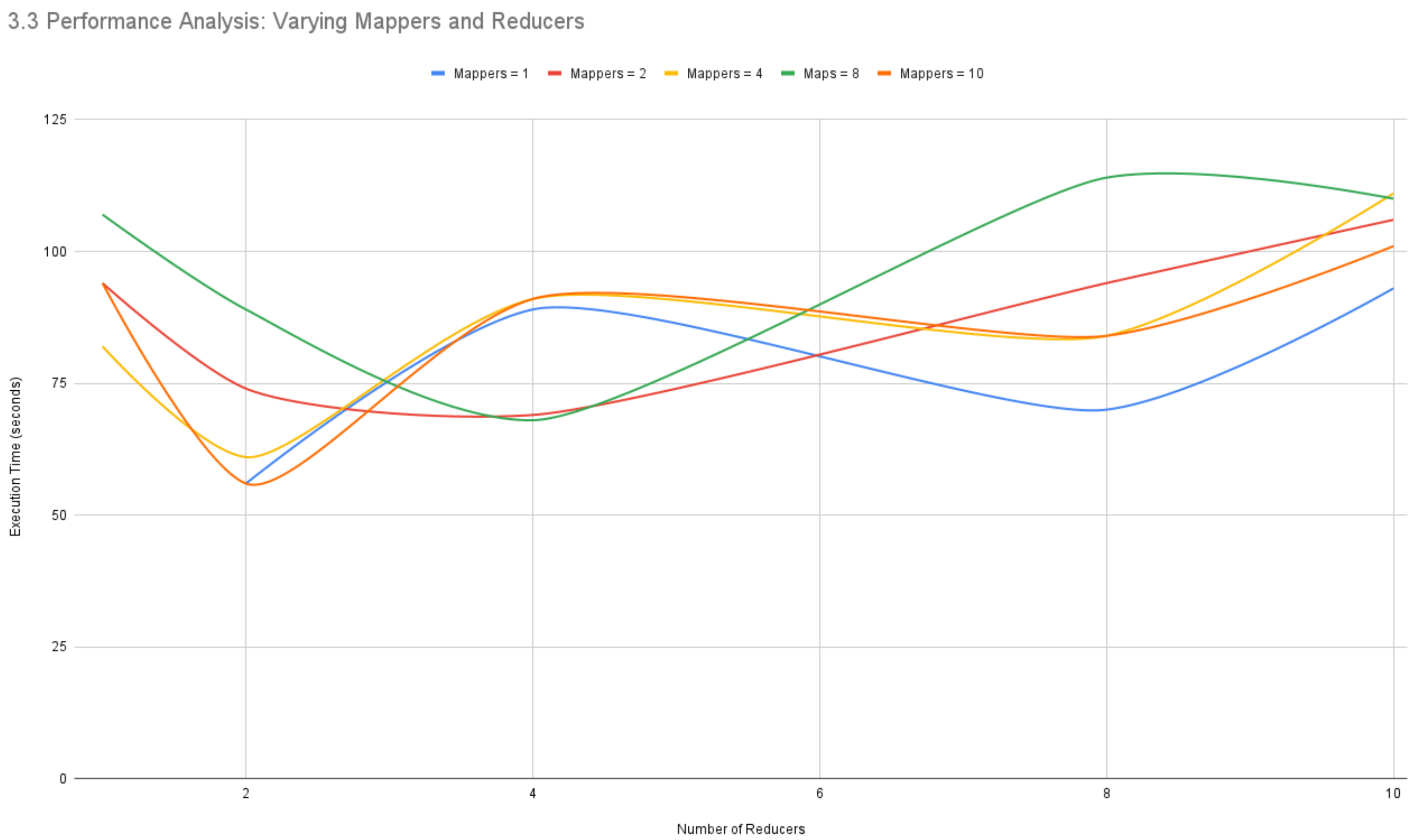
Farjad Ahmed, DDA 05, Performance Analysis									
Maps									
1		2		4		8		10	
Tasks	Time	Tasks	Time	Tasks	Time	Tasks	Time	Tasks	Time
1	52	1	75	1	60	1	63	1	82
2	72	2	63	2	46	2	74	2	80
4	47	4	61	4	70	4	80	4	89
8	78	8	59	8	77	8	96	8	96
10	86	10	80	10	75	10	92	10	101

3.2 Performance Analysis: Varying Mappers and Reducers



3.3 Performance Graphs

Farjad Ahmed, DDA 05, Performance Analysis									
Maps									
1		2		4		8		10	
Tasks	Time	Tasks	Time	Tasks	Time	Tasks	Time	Tasks	Time
1	56	1	94	1	82	1	107	1	94
2	89	2	74	2	61	2	89	2	56
4	70	4	69	4	91	4	68	4	91
8	93	8	94	8	84	8	114	8	84
10	82	10	106	10	111	10	110	10	101



References:
[1] Apache Hadoop Documentation: https://hadoop.apache.org/docs/stable/
[2] https://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/