

# DDA Lab

## Simran Kaur

### 311443

## Exercise 5

### Exercise 1: Preparing your Hadoop infrastructure

Exercise 1 is an installation guide rather than a graded exercise. You are not required to submit a "solution" for Exercise 1 in order to earn points. However demonstrating that your installation was successful with a solution to the verification exercise 1.3 would be appreciated. You must provide solutions for Exercise 2 on-wards.

## Exercise 1.1 Setting up a Hadoop Infrastructure

localhost:9870/dfshealth.html#tab-overview

Hadoop

Overview

Datanodes

Datanode Volume Failures

Snapshot

Startup Progress

Utilities

Overview

'localhost:9000' (active)

Started:	Thu Jun 09 14:13:25 +0200 2022
Version:	3.2.2, r7a3bc90b05f257c8ace2f76d74264906f0f7a932
Compiled:	Sun Jan 03 10:26:00 +0100 2021 by hexiaoqiao from branch-3.2.2
Cluster ID:	CID-d02e4fbb-534d-4dce-8d27-213db6b872b3
Block Pool ID:	BP-1546234320-192.168.56.1-1654468268260

Summary

Security is off.

Safemode is off.

24 files and directories, 13 blocks (13 replicated blocks, 0 erasure coded block groups) = 37 total filesystem object(s).

Heap Memory used 106.59 MB of 241.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 54.25 MB of 55.5 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

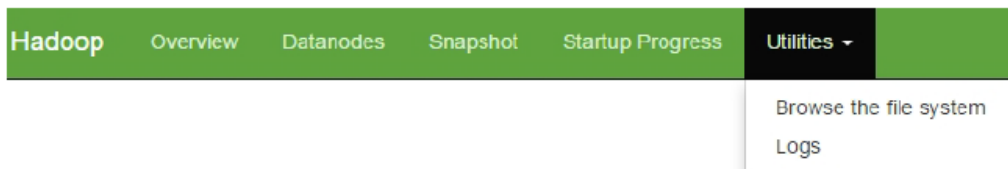
Configured Capacity:	573.9 GB
Configured Remote Capacity:	0 B
DFS Used:	736.68 MB (0.13%)

## Exercise 1.2: Basic Hadoop operations

## Exercise 1.2: Basic Hadoop operations

In this exercise, you are going to get familiar with Hadoop by playing around some basic Hadoop commands. Report the results that you get to your submitted .pdf to answer this exercise. All Hadoop commands are invoked by the `$HADOOP_HOME/bin/hadoop` command. Running the `hadoop` without any arguments prints the description for all commands.

1. Check Hadoop version: `hadoop version`.
2. List files in HDFS: `hadoop fs -ls /`
3. Create a `hadoopdemo` directory: `hadoop fs -mkdir /hadoopdemo`
4. Create several sub-directories nested in `hadoopdemo`, e.g. `text_files`, `raw_data`
5. Transfer and store a data file from local systems to Hadoop:  
`hadoop fs -put file.txt /hadoopdemo/text_files`
6. Check directories and files using Hadoop User Interface  
`http://localhost:50070 > Utilities > Browse the file system`.



7. Remove the sub-directory `hadoop fs -rm -r /hadoopdemo/text_files`
8. Change the content of `file.txt` in the local system and overwrite it in Hadoop  
`hadoop fs -put -f file.txt /hadoopdemo/text_files`
9. Read the content of the file: `hadoop fs -cat /hadoopdemo/text_files/file.txt`

### Check Hadoop version: `hadoop version`.

```
Administrator: Command Prompt

C:\hadoop-3.2.2\sbin>hadoop version
Hadoop 3.2.2
Source code repository Unknown -r 7a3bc90b05f257c8ace2f76d74264906f0f7a932
Compiled by hexiaoqiao on 2021-01-03T09:26Z
Compiled with protoc 2.5.0
From source with checksum 5a8f564f46624254b27f6a33126ff4
This command was run using /C:/hadoop-3.2.2/share/hadoop/common/hadoop-common-3.2.2.jar

C:\hadoop-3.2.2\sbin>
```

### Create a `hadoopdemo` directory: `hadoop fs -mkdir /hadoopdemo`

## Browse Directory

/

Go!

Show

25

entries

Search:

Permission

Owner

Group

Size

Last Modified

Replication

Block Size

Name

drwxr-xr-x

simra

supergroup

0 B

Jun 09 14:27

0

0 B

hadoopdemo

Showing 1 to 1 of 1 entries

Previous

1

Next

Hadoop, 2021.

## List files in HDFS: `hadoop fs -ls /`


```
C:\hadoop-3.2.2\sbin>hadoop fs -ls /
Found 1 items
drwxr-xr-x - simra supergroup          0 2022-06-09 14:27 /hadoopdemo

C:\hadoop-3.2.2\sbin>
```

Create several sub-directories nested in hadoopdemo, e.g. text files, raw data

Transfer and store a data file from local systems to Hadoop:

Read the content of the file: `hadoop fs -cat /hadoopdemo/text files/file.txt`

 Administrator: Command Prompt

```
C:\hadoop-3.2.2\sbin>hadoop fs -ls /
Found 1 items
drwxr-xr-x  - simra supergroup          0 2022-06-09 14:27 /hadoopdemo

C:\hadoop-3.2.2\sbin>hadoop fs -mkdir /hadoopdemo/text_files

C:\hadoop-3.2.2\sbin>hadoop fs -ls /hadoopdemo
Found 1 items
drwxr-xr-x  - simra supergroup          0 2022-06-09 14:32 /hadoopdemo/text_files


C:\hadoop-3.2.2\sbin>hadoop fs -put "C:\Users\simra\Downloads\myfile.txt" /hadoopdemo/text_files

C:\hadoop-3.2.2\sbin>hadoop fs -cat /hadoopdemo/text files/myfile.txt
cat: `/hadoopdemo/text': No such file or directory
cat: `files/myfile.txt': No such file or directory

C:\hadoop-3.2.2\sbin>hadoop fs -cat /hadoopdemo/text_files/myfile.txt
{'a': 2, 'b': 3}
{'c': 7, 'd': 5, 'a': 3}
{'a': 2, 'b': 3}
{'c': 7, 'd': 5, 'a': 3}

C:\hadoop-3.2.2\sbin>
```

**Remove the sub-directory `hadoop fs -rm -r /hadoopdemo/text files`**

 Administrator: Command Prompt

```
C:\hadoop-3.2.2\sbin>hadoop fs -rm -r /hadoopdemo/text_files
Deleted /hadoopdemo/text_files

C:\hadoop-3.2.2\sbin>_
```

## Exercise 1.3: WordCount MapReduce example

### Exercise 1.3: WordCount MapReduce example

Before going further to the exercise, let's learn some useful commands to interact with MapReduce jobs. General usage: `hadoop job [GENERIC_OPTIONS]`.

1. List generic options available in a Hadoop job: `hadoop job`
2. Check the status of job: `hadoop job -status <JOB-ID>`
3. List all running jobs: `hadoop job -list`
4. Check the history of job output-dir: `hadoop job -history <DIR>`
5. Kill a job: `hadoop job -kill <JOB-ID>`
6. To monitor your cluster and your jobs, you can also check the web interfaces of the Hadoop components instead of using the command line. The default address: <http://localhost:8088/cluster/apps>

In order to practically understand how data flows through a `map` and `reduce` computational pipeline, you are going to implement a WordCount example. WordCount is a simple application that counts the number of occurrences of each word in a given input set. More specifically, the input key is a file and the input value is a string, e.g. the file's content, while the output key is a word and the equivalent output value is an integer. Further descriptions about the WordCount application can be found here <http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>.

In this exercise, you are going to run a version of WordCount program that is already shipped with Hadoop installation. The successful execution of the program indicates that your Hadoop infrastructure is configured correctly and ready to use. The text dataset used for this exercise can be found here: <https://www.mirrorservice.org/sites/ftp.ibiblio.org/pub/docs/books/gutenberg/1/0/101/101.txt>.

In order to prove that you have done this exercise, you are asked to describe your solution step-by-step, e.g. command lines that you use, pictures of output, pictures of Hadoop User Interface, from the first command to last one. Please include the screenshot of the final console output for this step in your report.

**Making a directory in Hadoop file system by the name wordcount followed by creating a subdirectory Input and the text file is inserted into the subdirectory.**

Administrator: Command Prompt

```
C:\hadoop-3.2.2\sbin>hadoop fs -mkdir /wordcount  
C:\hadoop-3.2.2\sbin>hadoop fs -mkdir /wordcount/Input  
C:\hadoop-3.2.2\sbin>hadoop fs -put "C:\DDA\101.txt" /wordcount/Input  
C:\hadoop-3.2.2\sbin>_
```

The screenshot shows the Hadoop web interface at localhost:9870. A modal window titled "File information - 101.txt" is open, displaying details for a file in the /wordcount/Input directory. The file is 678064 bytes in size and is available on the node LAPTOP-UIRH6OE0. The file content is displayed as a Project Gutenberg eBook titled "The Project Gutenberg eBook of Hacker Crackdown, by Bruce Sterling".

**File information - 101.txt**

Download    Head the file (first 32K)    Tail the file (last 32K)

Block information -- Block 0

Block ID: 1073741872  
 Block Pool ID: BP-1546234320-192.168.56.1-1654468268260  
 Generation Stamp: 1048  
 Size: 678064  
 Availability:  
 • LAPTOP-UIRH6OE0

**File contents**

The Project Gutenberg eBook of Hacker Crackdown, by Bruce Sterling

This eBook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at [www.gutenberg.org](http://www.gutenberg.org)

\*\* This is a COPYRIGHTED Project Gutenberg eBook, Details Below \*\*

Close

**Running the hadoop jar file by providing the path of the jar file, Input directory and the output directory where output would be stored.**

The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt". The command executed is:

```
C:\hadoop-3.2.2\sbin>hadoop jar "C:\hadoop-3.2.2\share\hadoop\mapreduce\hadoop-mapreduce-examples-3.2.2.jar" wordcount /wordcount/Input /wordcount/Output
```

The output shows the progress of the job, including the number of splits (1), the number of tokens (0), and the output directory (/wordcount/Output). The job is running on a local node (localhost:8080) and is using the Hadoop MapReduce framework.

```
2022-06-09 15:22:46,839 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2022-06-09 15:22:46,959 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2022-06-09 15:22:46,959 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2022-06-09 15:22:47,630 INFO input.FileInputFormat: Total input files to process : 1
2022-06-09 15:22:47,671 INFO mapreduce.JobSubmitter: number of splits:1
2022-06-09 15:22:47,804 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local270339556_0001
2022-06-09 15:22:47,806 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-09 15:22:48,001 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2022-06-09 15:22:48,003 INFO mapreduce.Job: Running job: job_local270339556_0001
2022-06-09 15:22:48,005 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2022-06-09 15:22:48,018 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-06-09 15:22:48,018 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup_temporary folders under output directory:false, ignore cleanup failures: false
2022-06-09 15:22:48,019 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2022-06-09 15:22:48,086 INFO mapred.LocalJobRunner: Waiting for map tasks
2022-06-09 15:22:48,087 INFO mapred.LocalJobRunner: Starting task: attempt_local270339556_0001_m_000000_0
2022-06-09 15:22:48,123 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-06-09 15:22:48,123 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup_temporary folders under output directory:false, ignore cleanup failures: false
2022-06-09 15:22:48,138 INFO util.ProcfsBasedProcessTree: ProcfsBasedProcessTree currently is supported only on Linux.
2022-06-09 15:22:48,201 INFO mapred.Task: Using ResourceCalculatorProcessTree : org.apache.hadoop.yarn.util.WindowsBasedProcessTree@708902b9
2022-06-09 15:22:48,216 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/wordcount/Input/101.txt:0+678064
2022-06-09 15:22:48,289 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2022-06-09 15:22:48,289 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2022-06-09 15:22:48,291 INFO mapred.MapTask: soft limit at 83886080
```

Administrator: Command Prompt

```
C:\hadoop-3.2.2\sbin>hadoop fs -cat /wordcount/Output/part-r-00000
```

Administrator: Command Prompt

```
world. 21
world." 1
world? 1
worlds 5
worlds. 2
worldview 2
worldview. 1
worn 1
worried 1
worry 2
worry, 1
worse 5
worse, 1
worse--and 1
worse-feared 1
worse. 1
worse; 1
worship 1
worst 9
worst, 1
worst-case 1
worst. 1
worth 19
worth" 1
worth, 1
worth. 2
worth? 1
worthwhile. 1
worthy 4
would 214
```

localhost:9870/explorer.html#/wordcount/Output

Hadoop Overview Datanodes

Browse Directory

/wordcount/Output

Show 25 entries

Permission

Owner

-rw-r--r--

simra

-rw-r--r--

simra

Showing 1 to 2 of 2 entries

Hadoop, 2021.

File information - part-r-00000

Download Head the file (first 32K) Tail the file (last 32K)

Block information -- Block 0

Block ID: 1073741873  
Block Pool ID: BP-1546234320-192.168.56.1-1654468268260  
Generation Stamp: 1049  
Size: 233636  
Availability:

- LAPTOP-UIRH6OE0

File contents

mall-scale 5  
small-timers 1  
small-town 1  
small. 1  
smaller 3  
smaller, 1  
smart 5  
smart, 2

Close



## Exercise 2: Analysis of Airport efficiency with Map Reduce

### Exercise 2: Analysis of Airport efficiency with Map Reduce (10 points)

In this exercise you will download data from Bureau of Transportation Statistics' homepage<sup>[1]</sup>. On the Bureau' homepage you will download data by selecting following fields:

- Filter geography: all
- Filter year: 2017

---

<sup>1</sup>[https://www.transtats.bts.gov/DL\\_SelectFields.aspx?gnoyr\\_VQ=FGJ&Q0\\_fu146\\_anzr=b0-gvzr](https://www.transtats.bts.gov/DL_SelectFields.aspx?gnoyr_VQ=FGJ&Q0_fu146_anzr=b0-gvzr)

- Filter period: January
- Time period: FlightDate
- Airline: Reporting Airline, Flight Number Reporting Airline
- Origin: Origin
- Destination: Dest
- Departure Performance: DepTime, DepDelay
- Arrival Performance: ArrTime, ArrDelay

You will get a CSV file containing 450017 lines. An example of a line is as follows  
1/1/2017 12:00:00 AM,AA,307,DEN,PHX,1135,-10,1328,-17

In this exercise, you are going to write a python code using **Hadoop MapReduce** to accomplish several requirements:

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]
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.

In [2]:

```
import pandas as pd
```



In [78]:

```
file = pd.read_csv('C:\DDA\T_ONTIME_REPORTING.csv')
file.head(10)
```

Out[78]:

	FL_DATE	OP_UNIQUE_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	DEP_TIME	DEP_DI
0	1/1/2017 12:00:00 AM	AA	307	DEN	PHX	1135.0	
1	1/1/2017 12:00:00 AM	AA	307	PHX	PDX	1502.0	
2	1/1/2017 12:00:00 AM	AA	309	DCA	MIA	646.0	
3	1/1/2017 12:00:00 AM	AA	310	MIA	LGA	1402.0	
4	1/1/2017 12:00:00 AM	AA	311	PHX	DEN	2310.0	
5	1/1/2017 12:00:00 AM	AA	311	SEA	PHX	1800.0	
6	1/1/2017 12:00:00 AM	AA	312	ORD	MSP	1357.0	
7	1/1/2017 12:00:00 AM	AA	313	MIA	BWI	2035.0	
8	1/1/2017 12:00:00 AM	AA	314	MIA	ORD	2135.0	
9	1/1/2017 12:00:00 AM	AA	315	DFW	SAT	1330.0	

**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].**

**Mapper :**

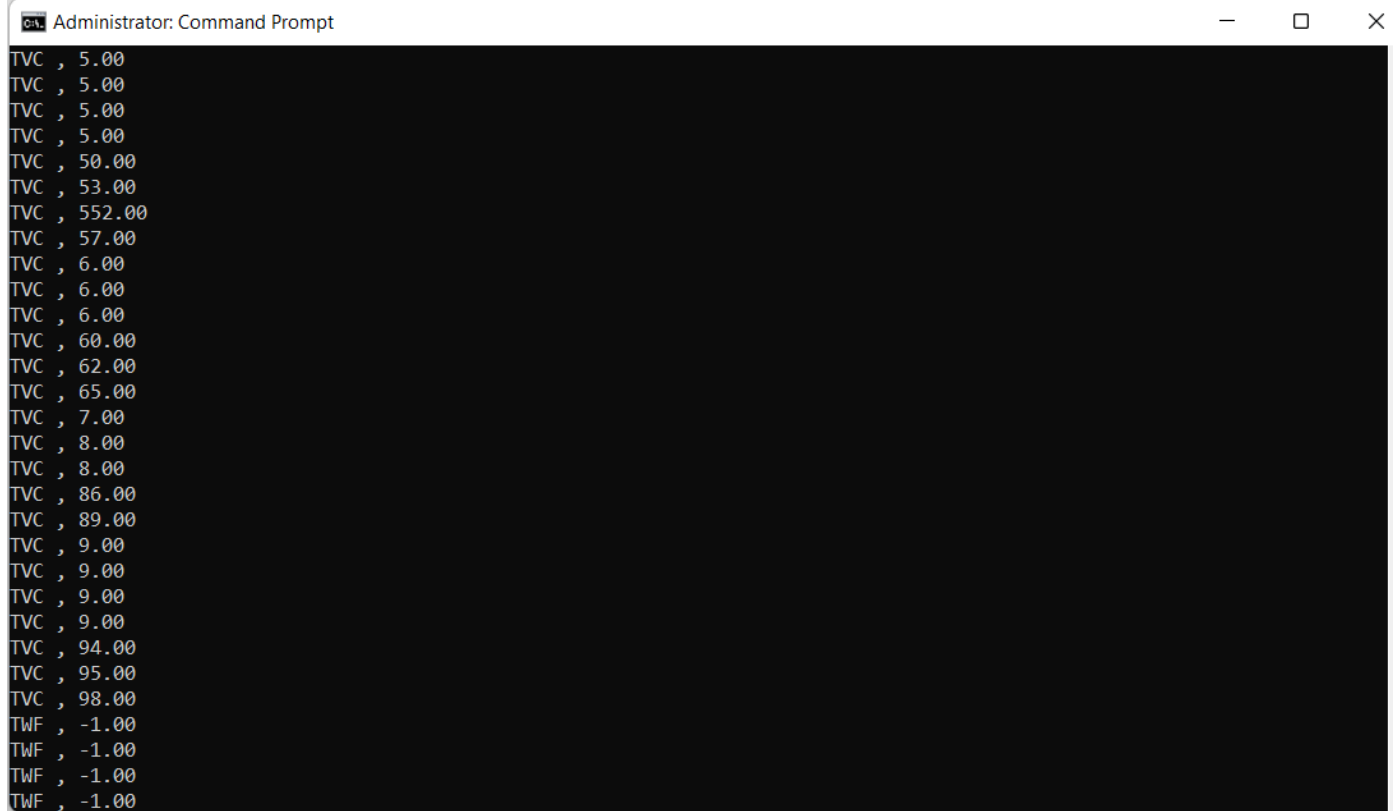
**Mapper takes origin and departure delay from each row of the dataframe and write it to the output file and shuffler works in between such that all the same origins come together to make the work of reducer easier.**

In [ ]:

```
#!/usr/bin/python39
import sys
# Storing the file in input
input = sys.stdin
for line in input:
    airport = line.split(',')[3]
    delay = line.split(',')[6]
    # if delay is not None then print it
    if delay:
        print(airport, ',', delay)
```

Administrator: Command Prompt

```
C:\hadoop-3.2.2\sbin>hadoop fs -mkdir /Airline
C:\hadoop-3.2.2\sbin>hadoop fs -mkdir /Airline/Input
C:\hadoop-3.2.2\sbin>hadoop fs -put "C:\DDA\T_ONTIME_REPORTING.csv" /Airline/Input
C:\hadoop-3.2.2\sbin>hadoop jar C:\hadoop-3.2.2\share\hadoop\tools\lib\hadoop-streaming-3.2.2.jar -mapper "python D:\mas
tersdata\DDALab\Exercise5\mapper.py" -input /Airline/Input/T_ONTIME_REPORTING.csv -output /Airline/Output
2022-06-09 16:53:02,740 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2022-06-09 16:53:02,858 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2022-06-09 16:53:02,858 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2022-06-09 16:53:02,876 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2022-06-09 16:53:03,488 INFO mapred.FileInputFormat: Total input files to process : 1
2022-06-09 16:53:03,557 INFO mapreduce.JobSubmitter: number of splits:1
2022-06-09 16:53:03,707 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local49149863_0001
2022-06-09 16:53:03,721 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-09 16:53:03,900 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2022-06-09 16:53:03,902 INFO mapreduce.Job: Running job: job_local49149863_0001
2022-06-09 16:53:03,903 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2022-06-09 16:53:03,906 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2022-06-09 16:53:03,920 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-06-09 16:53:03,920 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
2022-06-09 16:53:03,963 INFO mapred.LocalJobRunner: Waiting for map tasks
2022-06-09 16:53:03,969 INFO mapred.LocalJobRunner: Starting task: attempt_local49149863_0001_m_000000_0
2022-06-09 16:53:04,003 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-06-09 16:53:04,003 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
2022-06-09 16:53:04,014 INFO util.ProcfsBasedProcessTree: ProcfsBasedProcessTree currently is supported only on Linux.
```



```
Administrator: Command Prompt
TVC , 5.00
TVC , 5.00
TVC , 5.00
TVC , 5.00
TVC , 50.00
TVC , 53.00
TVC , 552.00
TVC , 57.00
TVC , 6.00
TVC , 6.00
TVC , 6.00
TVC , 60.00
TVC , 62.00
TVC , 65.00
TVC , 7.00
TVC , 8.00
TVC , 8.00
TVC , 86.00
TVC , 89.00
TVC , 9.00
TVC , 9.00
TVC , 9.00
TVC , 9.00
TVC , 94.00
TVC , 95.00
TVC , 98.00
TWF , -1.00
TWF , -1.00
TWF , -1.00
TWF , -1.00
```

## Reducer :

**Reducer calculates the minimum, maximum and average departure delay for each airport.**

In [ ]:

```
#!/usr/bin/python39
import sys
input = sys.stdin
current_airport = None      # this variable is to check whether current and next item are s
current_delay = 0           # to store the total delay for each airport
count = 0
for line in input:
    line = line.strip()
    airport = line.split(',')[0]
    delay = line.split(',')[1]
    try:
        delay = float(delay)
    except ValueError:
        continue
    # this is to do calculation for the same airport
    if delay > 0:
        if current_airport == airport:
            current_delay += delay
            count += 1
            if min_delay > delay:
                min_delay = delay
            if max_delay < delay:
                max_delay = delay
        else:
            if current_airport:      # now when the airport is different we need to print
                print(current_airport, ',', max_delay, ',', min_delay, ',', current_delay/c
                count = 0
            else:
                print('Airport, Max Delay, Min Delay, Average Delay')
            current_airport = airport
            current_delay = delay
            count += 1
            min_delay = delay
            max_delay = delay
# for the last line
if current_airport == airport:
    print(current_airport, ',', max_delay, ',', min_delay, ',', current_delay/count)
```

```

Administrator: Command Prompt

C:\hadoop-3.2.2\sbin>hadoop fs -rm -r /Airline/Output
Deleted /Airline/Output

C:\hadoop-3.2.2\sbin>hadoop jar C:\hadoop-3.2.2\share\hadoop\tools\lib\hadoop-streaming-3.2.2.jar -mapper "python D:\mas
tersdata\DDALab\Exercise5\mapper.py" -reducer "python D:\mastersdata\DDALab\Exercise5\reducer.py" -input /Airline/Input/
T_ONTIME_REPORTING.csv -output /Airline/Output
2022-06-09 17:12:49,190 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2022-06-09 17:12:49,307 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2022-06-09 17:12:49,307 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2022-06-09 17:12:49,324 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2022-06-09 17:12:49,926 INFO mapred.FileInputFormat: Total input files to process : 1
2022-06-09 17:12:49,995 INFO mapreduce.JobSubmitter: number of splits:1
2022-06-09 17:12:50,125 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local77471089_0001
2022-06-09 17:12:50,126 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-09 17:12:50,301 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2022-06-09 17:12:50,304 INFO mapreduce.Job: Running job: job_local77471089_0001
2022-06-09 17:12:50,304 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2022-06-09 17:12:50,307 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2022-06-09 17:12:50,315 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-06-09 17:12:50,315 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
2022-06-09 17:12:50,358 INFO mapred.LocalJobRunner: Waiting for map tasks
2022-06-09 17:12:50,363 INFO mapred.LocalJobRunner: Starting task: attempt_local77471089_0001_m_000000_0
2022-06-09 17:12:50,397 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-06-09 17:12:50,398 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
2022-06-09 17:12:50,409 INFO util.ProcfsBasedProcessTree: ProcfsBasedProcessTree currently is supported only on Linux.
2022-06-09 17:12:50,453 INFO mapred.Task: Using ResourceCalculatorProcessTree : org.apache.hadoop.yarn.util.WindowsBase
dProcessTree@47cccb87

```

```

Administrator: Command Prompt

Map output bytes=5700432
Map output materialized bytes=6583392
Input split bytes=110
Combine input records=0
Combine output records=0
Reduce input groups=26942
Reduce shuffle bytes=6583392
Reduce input records=441477
Reduce output records=298
Spilled Records=882954
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=9
Total committed heap usage (bytes)=676331520
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=27064598
File Output Format Counters
Bytes Written=11341
2022-06-09 17:12:56,375 INFO streaming.StreamJob: Output directory: /Airline/Output

```

```

Administrator: Command Prompt
C:\hadoop-3.2.2\sbin>hadoop fs -cat /Airline/Output/part-00000
Airport, Max Delay, Min Delay, Average Delay
ABE , 794.0 , 1.0 , 85.71698113207547
ABI , 263.0 , 1.0 , 68.25
ABQ , 911.0 , 1.0 , 32.265060240963855
ABR , 1259.0 , 1.0 , 117.47619047619048
ABY , 291.0 , 4.0 , 68.0
ACT , 202.0 , 1.0 , 49.05555555555556
ACV , 578.0 , 1.0 , 60.78260869565217
ACY , 670.0 , 1.0 , 43.729166666666664
ADK , 41.0 , 4.0 , 20.8
ADQ , 73.0 , 12.0 , 29.5
AEX , 354.0 , 1.0 , 73.73770491803279
AGS , 1130.0 , 1.0 , 66.01428571428572
ALB , 981.0 , 1.0 , 42.845283018867924
AMA , 298.0 , 1.0 , 38.535714285714285
ANC , 1195.0 , 1.0 , 34.89967637540453
APN , 278.0 , 4.0 , 59.36842105263158
ASE , 1110.0 , 1.0 , 67.51714285714286
ATL , 1470.0 , 1.0 , 46.87010611050128
ATW , 1065.0 , 1.0 , 83.97222222222223
AUS , 1246.0 , 1.0 , 33.92505854800937
AVL , 425.0 , 1.0 , 72.18181818181819
AVP , 783.0 , 2.0 , 121.62068965517241
AZO , 268.0 , 1.0 , 67.45454545454545
BDL , 549.0 , 1.0 , 32.23551401869159
BET , 72.0 , 1.0 , 16.416666666666668
BFL , 290.0 , 1.0 , 51.705882352941174
BGM , 216.0 , 2.0 , 72.41666666666667

```

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

### Mapper :

In [ ]:

```

#!/usr/bin/python3
import sys
# Storing the file in input
input = sys.stdin
for line in input:
    line = line.strip()
    airport = line.split(',')[3]
    delay = line.split(',')[6]
    # if delay is not None then print it
    if delay:
        print(airport, ',', delay)

```

### Reducer :

In [ ]:

```
#!/usr/bin/python39
import sys
input = sys.stdin
current_airport = None    # this variable is to check whether current and next item are same
current_delay = 0         # to store the total delay for each airport
count = 0
airports = {}             # this dictionary will store airport as keys and average delay as value
top_airport = []          # list containing 10 top airports
for line in input:
    line = line.strip()
    airport = line.split(',')[0]
    delay = line.split(',')[1]
    try:
        delay = float(delay)
    except ValueError:
        continue
    if delay > 0:
        if current_airport == airport:
            current_delay += delay
            count += 1
        else:
            if current_airport:
                airports[current_airport] = current_delay/count
                count = 0
            else:
                print('Top 10 Airports')
            current_airport = airport
            current_delay = delay
            count += 1

if current_airport == airport:
    airports[current_airport] = current_delay/count
top_airport = [(k, v) for k, v in sorted(airports.items(), key=lambda item: item[1])][:10]
for tup in top_airport:
    print(tup)
```



```

Administrator: Command Prompt

C:\hadoop-3.2.2\sbin>hadoop jar C:\hadoop-3.2.2\share\hadoop\tools\lib\hadoop-streaming-3.2.2.jar -mapper "python D:\mastersdata\DDALab\Exercise5\mapper2.py" -reducer "python D:\mastersdata\DDALab\Exercise5\reducer2.py" -input /Airline/Input/T_ONTIME_REPORTING.csv -output /Airline/Output2
2022-06-09 17:31:54,773 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2022-06-09 17:31:54,896 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2022-06-09 17:31:54,896 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2022-06-09 17:31:54,913 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2022-06-09 17:31:55,553 INFO mapred.FileInputFormat: Total input files to process : 1
2022-06-09 17:31:55,619 INFO mapreduce.JobSubmitter: number of splits:1
2022-06-09 17:31:55,761 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local2072791626_0001
2022-06-09 17:31:55,762 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-09 17:31:55,942 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2022-06-09 17:31:55,945 INFO mapreduce.Job: Running job: job_local2072791626_0001
2022-06-09 17:31:55,945 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2022-06-09 17:31:55,948 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2022-06-09 17:31:55,956 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-06-09 17:31:55,957 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
2022-06-09 17:31:56,002 INFO mapred.LocalJobRunner: Waiting for map tasks
2022-06-09 17:31:56,007 INFO mapred.LocalJobRunner: Starting task: attempt_local2072791626_0001_m_000000_0
2022-06-09 17:31:56,041 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-06-09 17:31:56,041 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
2022-06-09 17:31:56,053 INFO util.ProcfsBasedProcessTree: ProcfsBasedProcessTree currently is supported only on Linux.
2022-06-09 17:31:56,096 INFO mapred.Task: Using ResourceCalculatorProcessTree : org.apache.hadoop.yarn.util.WindowsBasedProcessTree@11f53760
2022-06-09 17:31:56,108 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/Airline/Input/T_ONTIME_REPORTING.csv:0+27064598
2022-06-09 17:31:56,130 INFO mapred.MapTask: numReduceTasks: 1

```

```

Administrator: Command Prompt

C:\hadoop-3.2.2\sbin>hadoop fs -cat /Airline/Output2/part-00000
Top 10 Airports
('LSE ', 2.0)
('CDV ', 13.0)
('ITO ', 13.188811188811188)
('GUM ', 15.217391304347826)
('BET ', 16.416666666666668)
('EAU ', 18.4)
('BJI ', 19.285714285714285)
('RKS ', 20.3)
('ADK ', 20.8)
('MMH ', 21.0)

C:\hadoop-3.2.2\sbin>_

```

## Exercise 3: Analysis of Movie dataset using Map and Reduce

## Exercise 3: Analysis of Movie dataset using Map and Reduce (10 points)

For this exercise you will use movielens10m dataset available at <http://files.grouplens.org/datasets/movielens/ml-10m-README.html>. The movielens10m dataset consists of 10 million rating entries by users. There are two main files required to solve this exercise 1) rating.dat and 2) movie.dat. The rating.dat file contains userId, movieId and ratings (on scale of 1 to 5). The movie.dat file contains information about movies i.e. movieId, Title and Genres.

In this exercise, you are going to write a python code using Hadoop MapReduce to accomplish several requirements, note as your dataset is large you also need to perform some performance analysis i.e. how many mappers and reducers you used, what is the performance increase by varying number of mappers and reducers. Please also note that although you may have very few physical cores i.e. 2 or 4 or 8, but you can still experiment with a large number of mappers and reducers. Plot the execution time vs the number of mappers  $\times$  reducers for each task below.

1. Find the movie title which has the maximum average rating?
2. Find the user who has assign lowest average rating among all the users who rated more than 40 times?
3. Find the highest average rated movie genre? [Hint: you may need to merge/combine movie.dat and rating.dat in a preprocessing step.]

In [82]:

```
movies = pd.read_csv(r"C:\Users\simra\Downloads\ml-10m\ml-10M100K\movies.dat", sep='::', header=0, names=['movie_id', 'movie_title', 'genres'])
```

```
C:\Users\simra\anaconda3\lib\site-packages\pandas\util\_decorators.py:311: ParserWarning: Falling back to the 'python' engine because the 'c' engine does not support regex separators (separators > 1 char and different from '\s+' are interpreted as regex); you can avoid this warning by specifying engine='python'.
```

```
return func(*args, **kwargs)
```

In [83]:

```
movies
```

Out[83]:

	movie_id	movie_title	genres
0	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
1	2	Jumanji (1995)	Adventure Children Fantasy
2	3	Grumpier Old Men (1995)	Comedy Romance
3	4	Waiting to Exhale (1995)	Comedy Drama Romance
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
10676	65088	Bedtime Stories (2008)	Adventure Children Comedy
10677	65091	Manhattan Melodrama (1934)	Crime Drama Romance
10678	65126	Choke (2008)	Comedy Drama
10679	65130	Revolutionary Road (2008)	Drama Romance
10680	65133	Blackadder Back & Forth (1999)	Comedy

10681 rows × 3 columns

In [84]:

```
ratings = pd.read_csv(r"C:\Users\simra\Downloads\ml-10m\ml-10M100K\ratings.dat", sep='::',
                      names=['user_id', 'movie_id', 'rating', 'timestamp'])
```

C:\Users\simra\anaconda3\lib\site-packages\pandas\util\\_decorators.py:311: ParserWarning: Falling back to the 'python' engine because the 'c' engine does not support regex separators (separators > 1 char and different from '\s+' are interpreted as regex); you can avoid this warning by specifying engine='python'.  
 return func(\*args, \*\*kwargs)

In [85]:

```
ratings
```

Out[85]:

	user_id	movie_id	rating	timestamp
0	1	122	5.0	838985046
1	1	185	5.0	838983525
2	1	231	5.0	838983392
3	1	292	5.0	838983421
4	1	316	5.0	838983392
...	...	...	...	...
10000049	71567	2107	1.0	912580553
10000050	71567	2126	2.0	912649143
10000051	71567	2294	5.0	912577968
10000052	71567	2338	2.0	912578016
10000053	71567	2384	2.0	912578173

10000054 rows × 4 columns

In [86]:

```
movie_rating = pd.merge(movies, ratings, on='movie_id')
```

In [87]:

```
movie_rating
```

Out[87]:

	movie_id	movie_title	genres	user_id	rating	tin
0	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	5	1.0	85
1	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	14	3.0	113
2	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	18	3.0	111
3	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	23	5.0	84
4	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	24	5.0	86
...	...	...	...	...	...	...
10000049	65133	Blackadder Back & Forth (1999)	Comedy	24495	4.0	123
10000050	65133	Blackadder Back & Forth (1999)	Comedy	33384	3.0	123
10000051	65133	Blackadder Back & Forth (1999)	Comedy	40570	2.0	123
10000052	65133	Blackadder Back & Forth (1999)	Comedy	45430	2.5	123
10000053	65133	Blackadder Back & Forth (1999)	Comedy	68151	5.0	123

10000054 rows × 6 columns



In [88]:

```
movie_rating.to_csv('movie_ratings.csv', index=False, sep = ';')
```

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

The mapper first outputs movie title and rating given to that movie and the reducer calculates average rating for each movie and then sorts the dictionary to get maximum average rating and finally outputs all the movies that have the

## highest average rating.

### Mapper :

In [ ]:

```
#!/usr/bin/python39
import sys
input = sys.stdin
next(input)          # ignore the column names
for line in input:
    movie_title = line.split(';')[1]
    rating = line.split(';')[4]
    # if rating is not None then print it
    if rating:
        print(movie_title, ';', rating)
```

### Reducer :

In [ ]:

```
import sys
input = sys.stdin
current_movie = None      # this variable is to check whether current and next movie are
current_rating = 0        # to calculate ratings for each movie
avg_rating = {}          # dictionary to store average ratings for each movie
count = 0
for line in input:
    movie = line.split(';')[0]
    rating = line.split(';')[1]
    try:
        rating = float(rating)          # if the rating is available then only c
    except ValueError:
        continue
    if movie == current_movie:
        current_rating += rating
        count += 1
    else:
        if current_movie:
            avg_rating[current_movie] = current_rating/count
            count = 0
        else:
            print('Movie with highest average rating:')
            current_movie = movie
            current_rating = rating
            count += 1
if current_movie == movie:
    avg_rating[current_movie] = current_rating/count

top_movie = max(avg_rating.items(), key=lambda item: item[1])    # movie with the max aver
for key, value in avg_rating.items():
    if value == top_movie[1]:
        print(key, value)
```

```
Administrator: Command Prompt

C:\hadoop-3.2.2\sbin>hadoop fs -mkdir /Movies

C:\hadoop-3.2.2\sbin>hadoop fs -mkdir /Movies/Input

C:\hadoop-3.2.2\sbin>hadoop fs -put "C:\DDA\movie_ratings.csv" /Movies/Input

C:\hadoop-3.2.2\sbin>hadoop jar C:\hadoop-3.2.2\share\hadoop\tools\lib\hadoop-streaming-3.2.2.jar -mapper "python D:\mas
tersdata\DDALab\Exercise5\mapper_movie.py" -reducer "python D:\mastersdata\DDALab\Exercise5\reducer_movie.py" -input /Mo
vies/Input/movie_ratings.csv -output /Movies/Output_
```

```
Administrator: Command Prompt

Map output records=10000049
Map output bytes=325119528
Map output materialized bytes=345119907
Input split bytes=624
Combine input records=0
Combine output records=0
Reduce input groups=89424
Reduce shuffle bytes=345119907
Reduce input records=10000049
Reduce output records=6
Spilled Records=29484832
Shuffled Maps =6
Failed Shuffles=0
Merged Map outputs=6
GC time elapsed (ms)=279
Total committed heap usage (bytes)=3000500224

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=710272431
File Output Format Counters
Bytes Written=250
2022-06-09 23:11:46,888 INFO streaming.StreamJob: Output directory: /Movies/Output

C:\hadoop-3.2.2\sbin>
```

```
Administrator: Command Prompt

C:\hadoop-3.2.2\sbin>hadoop fs -cat /Movies/Output/part-00000
Movie with highest average rating:
Blue Light, The (Das Blaue Licht) (1932) 5.0
Fighting Elegy (Kenka erejii) (1966) 5.0
Satan's Tango (S|it|intang|) (1994) 5.0
Shadows of Forgotten Ancestors (1964) 5.0
Sun Alley (Sonnenallee) (1999) 5.0

C:\hadoop-3.2.2\sbin>
```



In [3]:

```
map_red = {'Mapper:1': [55762, 55527, 55719, 58061],
           'Mapper:2': [56124, 54919, 58308, 61277],
           'Mapper:3': [52861, 54483, 56602, 60025],
           'Mapper:4': [53824, 57279, 55294, 59229]}

time = pd.DataFrame.from_dict(map_red, orient = 'index')
time.columns = ['Reducer:1', 'Reducer:2', 'Reducer:3', 'Reducer:4']
time = time.style.set_properties(**{
    'background-color': 'grey',
    'font-size': '20pt',
})
time
```

Out[3]:

	Reducer:1	Reducer:2	Reducer:3	Reducer:4
Mapper:1	55762	55527	55719	58061
Mapper:2	56124	54919	58308	61277
Mapper:3	52861	54483	56602	60025
Mapper:4	53824	57279	55294	59229

```
Administrator: Command Prompt

C:\hadoop-3.2.2\sbin>hadoop jar C:\hadoop-3.2.2\share\hadoop\tools\lib\hadoop-streaming-3.2.2.jar -D mapreduce.job.reduc
es = 4 -D mapreduce.job.maps = 4 -mapper "python D:\mastersdata\DDALab\Exercise5\mapper_movie.py" -reducer "python D:\ma
stersdata\DDALab\Exercise5\reducer_movie.py" -input /Movies/Input/movie_ratings.csv -output /Movies/Output
packageJobJar: [/C:/Users/simra/AppData/Local/Temp/hadoop-unjar3354253778923317619/] [] C:\Users\simra\AppData\Local\Tem
p\streamjob270802083535699809.jar tmpDir=null
2022-06-10 15:59:24,088 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2022-06-10 15:59:24,244 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2022-06-10 15:59:24,623 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/
simra/.staging/job_1654864782023_0033
2022-06-10 15:59:24,819 INFO mapred.FileInputFormat: Total input files to process : 1
2022-06-10 15:59:24,872 INFO mapreduce.JobSubmitter: number of splits:6
2022-06-10 15:59:24,958 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1654864782023_0033
2022-06-10 15:59:24,960 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-10 15:59:25,108 INFO conf.Configuration: resource-types.xml not found
2022-06-10 15:59:25,109 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-06-10 15:59:25,164 INFO impl.YarnClientImpl: Submitted application application_1654864782023_0033
2022-06-10 15:59:25,204 INFO mapreduce.Job: The url to track the job: http://LAPTOP-UIRH60E0:8088/proxy/application_1654
864782023_0033/
2022-06-10 15:59:25,207 INFO mapreduce.Job: Running job: job_1654864782023_0033
2022-06-10 15:59:32,359 INFO mapreduce.Job: Job job_1654864782023_0033 running in uber mode : false
2022-06-10 15:59:32,359 INFO mapreduce.Job: map 0% reduce 0%
2022-06-10 15:59:42,555 INFO mapreduce.Job: map 17% reduce 0%
2022-06-10 15:59:50,631 INFO mapreduce.Job: map 100% reduce 0%
2022-06-10 15:59:57,751 INFO mapreduce.Job: map 100% reduce 25%
2022-06-10 15:59:59,781 INFO mapreduce.Job: map 100% reduce 100%
2022-06-10 16:00:00,804 INFO mapreduce.Job: Job job_1654864782023_0033 completed successfully
2022-06-10 16:00:00,892 INFO mapreduce.Job: Counters: 55
File System Counters
FILE: Number of bytes read=671939385
```

```
Administrator: Command Prompt

Reduce shuffle bytes=345120015
Reduce input records=10000049
Reduce output records=158
Spilled Records=29484832
Shuffled Maps =24
Failed Shuffles=0
Merged Map outputs=24
GC time elapsed (ms)=1271
CPU time spent (ms)=59229
Physical memory (bytes) snapshot=3320926208
Virtual memory (bytes) snapshot=4136620032
Total committed heap usage (bytes)=2477785088
Peak Map Physical memory (bytes)=319160320
Peak Map Virtual memory (bytes)=420126720
Peak Reduce Physical memory (bytes)=360157184
Peak Reduce Virtual memory (bytes)=427515904

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=710272431
File Output Format Counters
  Bytes Written=5344

2022-06-10 16:00:00,893 INFO streaming.StreamJob: Output directory: /Movies/Output

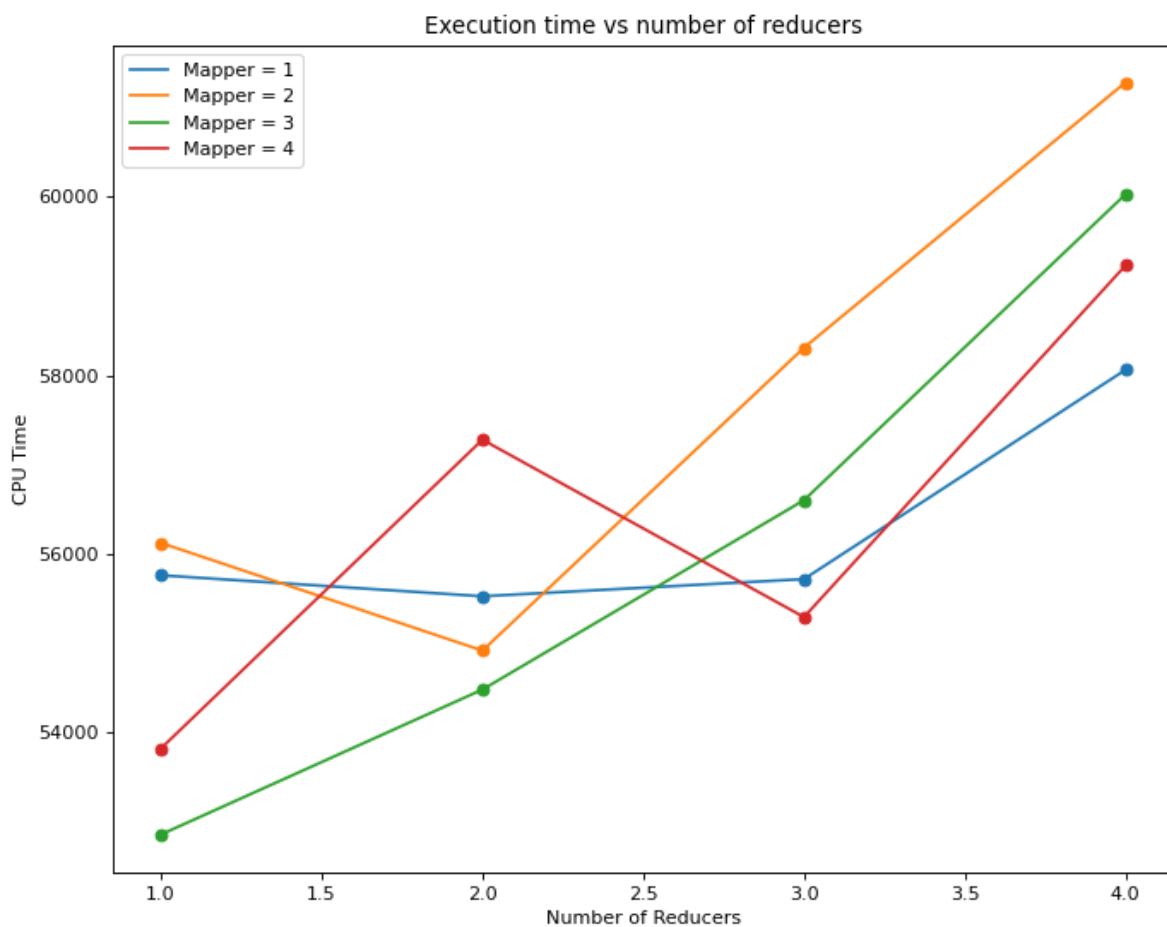
C:\hadoop-3.2.2\sbin>
```

In [5]:

```

import matplotlib.pyplot as plt
plt.figure(figsize=(10, 8), dpi=80)
plt.plot([1,2,3,4], map_red['Mapper:1'], label = 'Mapper = 1')
plt.scatter([1,2,3,4], map_red['Mapper:1'])
plt.plot([1,2,3,4], map_red['Mapper:2'], label = 'Mapper = 2')
plt.scatter([1,2,3,4], map_red['Mapper:2'])
plt.plot([1,2,3,4], map_red['Mapper:3'], label = 'Mapper = 3')
plt.scatter([1,2,3,4], map_red['Mapper:3'])
plt.plot([1,2,3,4], map_red['Mapper:4'], label = 'Mapper = 4')
plt.scatter([1,2,3,4], map_red['Mapper:4'])
plt.legend()
plt.xlabel('Number of Reducers')
plt.ylabel('CPU Time')
plt.title('Execution time vs number of reducers')
plt.show()

```



The plot above gives the execution time when different number of reducers are used for different number of mappers. As the number of mappers and reducers are increased and when the data to be processed is not that large the overhead increases resulting in increased execution time. From the plot we can see that when 3 mappers and 1 reducer is used, the execution time is the minimum.

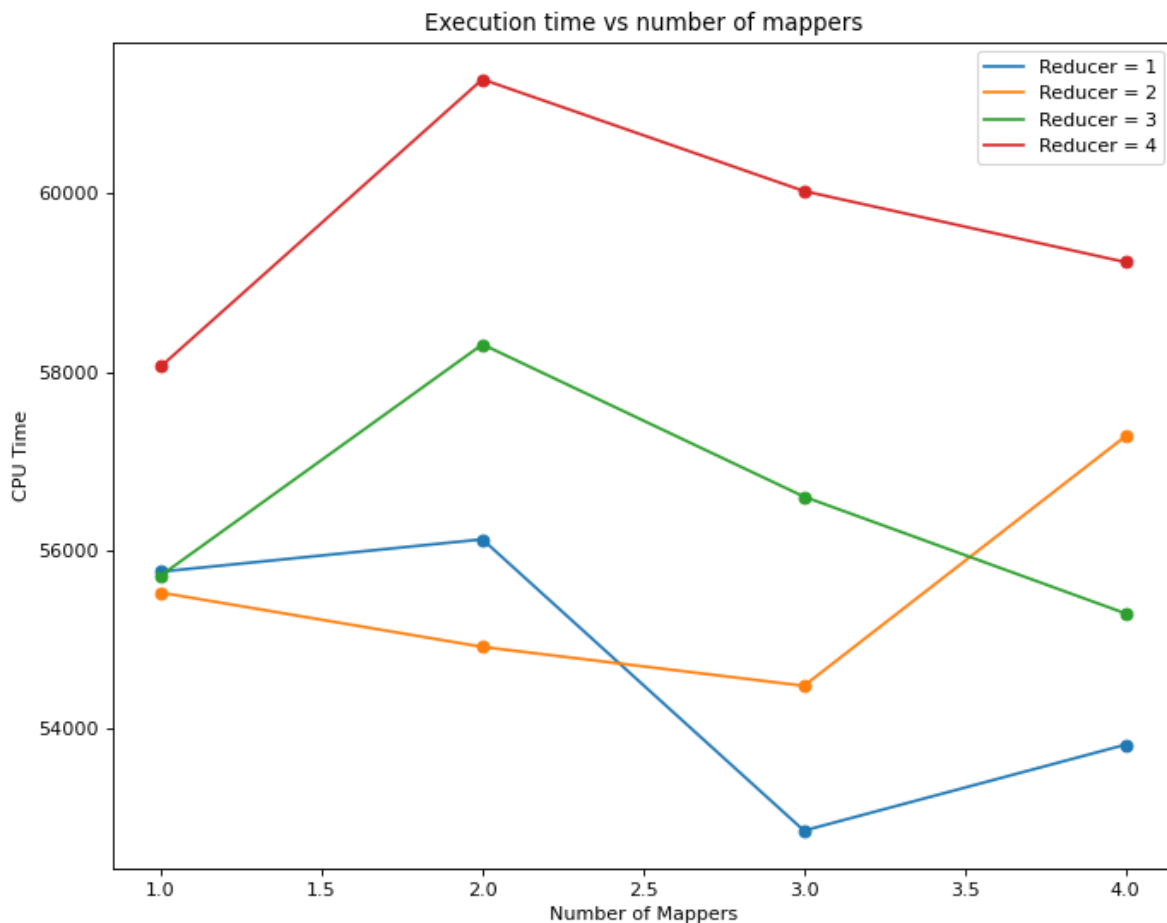
In [6]:

```

red_map = {'Reducer:1': [55762, 56124, 52861, 53824],
           'Reducer:2': [55527, 54919, 54483, 57279],
           'Reducer:3': [55719, 58308, 56602, 55294],
           'Reducer:4': [58061, 61277, 60025, 59229]}

plt.figure(figsize=(10, 8), dpi=80)
plt.plot([1,2,3,4], red_map['Reducer:1'], label = 'Reducer = 1')
plt.scatter([1,2,3,4], red_map['Reducer:1'])
plt.plot([1,2,3,4], red_map['Reducer:2'], label = 'Reducer = 2')
plt.scatter([1,2,3,4], red_map['Reducer:2'])
plt.plot([1,2,3,4], red_map['Reducer:3'], label = 'Reducer = 3')
plt.scatter([1,2,3,4], red_map['Reducer:3'])
plt.plot([1,2,3,4], red_map['Reducer:4'], label = 'Reducer = 4')
plt.scatter([1,2,3,4], red_map['Reducer:4'])
plt.legend()
plt.xlabel('Number of Mappers')
plt.ylabel('CPU Time')
plt.title('Execution time vs number of mappers')
plt.show()

```



**2. Find the user who has assign lowest average rating among all the users who**

## rated more than 40 times?

As done before the mapper outputs the user and rating given by each user and then reducer takes those users that have rated more than 40 movies and then calculates the average rating given by them and finally outputs the user who has assigned lowest average rating.

### Mapper :

In [ ]:

```
#!/usr/bin/python39
import sys
import pandas as pd
input = sys.stdin
next(input)          # ignore the column names
for line in input:
    user_id = line.split(';')[3]          # user
    rating = line.split(';')[4]          # rating
    if rating:          # if rating is not None
        print(user_id, ';', rating)
```

### Reducer :

In [ ]:

```
#!/usr/bin/python39
import sys
input = sys.stdin
current_user = None          # this variable is to check whether current and next user are same
current_rating = 0           # to calculate ratings assigned by each user
avg_rating = {}              # average rating per user
count = 0

for line in input:
    user = line.split(';')[0]
    rating = line.split(';')[1]
    try:
        rating = float(rating)
    except ValueError:
        continue
    if user == current_user:
        current_rating += rating
        count += 1
    else:
        if (current_user is not None) and (count > 40):          # if user is not None and
            avg_rating[current_user] = current_rating/count
            count = 0
        elif current_user is None:
            print('Users who has assigned lowest average rating:')
            current_user = user
            current_rating = rating
            count += 1
if current_user == user and count > 40:
    avg_rating[current_user] = current_rating/count

lowest_rating = min(avg_rating.items(), key=lambda item: item[1])    # users with lowest a
for key, value in avg_rating.items():
    if value == lowest_rating[1]:
        print(key, value)
```

```
Administrator: Command Prompt

C:\hadoop-3.2.2\sbin>hadoop jar C:\hadoop-3.2.2\share\hadoop\tools\lib\hadoop-streaming-3.2.2.jar -mapper "python D:\mas
tersdata\DDALab\Exercise5\mapper2_movie.py" -reducer "python D:\mastersdata\DDALab\Exercise5\reducer2_movie.py" -input /
Movies/Input/movie_ratings.csv -output /Movies/Output2
2022-06-09 23:37:56,549 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2022-06-09 23:37:56,662 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2022-06-09 23:37:56,663 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2022-06-09 23:37:56,679 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2022-06-09 23:37:57,279 INFO mapred.FileInputFormat: Total input files to process : 1
2022-06-09 23:37:57,348 INFO mapreduce.JobSubmitter: number of splits:6
2022-06-09 23:37:57,472 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1614820528_0001
2022-06-09 23:37:57,473 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-09 23:37:57,645 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2022-06-09 23:37:57,648 INFO mapreduce.Job: Running job: job_local1614820528_0001
2022-06-09 23:37:57,648 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2022-06-09 23:37:57,651 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2022-06-09 23:37:57,664 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-06-09 23:37:57,664 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
2022-06-09 23:37:57,728 INFO mapred.LocalJobRunner: Waiting for map tasks
2022-06-09 23:37:57,734 INFO mapred.LocalJobRunner: Starting task: attempt_local1614820528_0001_m_000000_0
2022-06-09 23:37:57,769 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-06-09 23:37:57,770 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
2022-06-09 23:37:57,781 INFO util.ProcfsBasedProcessTree: ProcfsBasedProcessTree currently is supported only on Linux.
2022-06-09 23:37:57,834 INFO mapred.Task: Using ResourceCalculatorProcessTree : org.apache.hadoop.yarn.util.WindowsBase
dProcessTree@255177ab
2022-06-09 23:37:57,845 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/Movies/Input/movie_ratings.csv:0+13
4217728
2022-06-09 23:37:57,865 INFO mapred.MapTask: numReduceTasks: 1
```

```
Administrator: Command Prompt

Map output records=10000049
Map output bytes=128495187
Map output materialized bytes=148495321
Input split bytes=624
Combine input records=0
Combine output records=0
Reduce input groups=414580
Reduce shuffle bytes=148495321
Reduce input records=10000049
Reduce output records=2
Spilled Records=20000098
Shuffled Maps =6
Failed Shuffles=0
Merged Map outputs=6
GC time elapsed (ms)=266
Total committed heap usage (bytes)=2944925696

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=710272431
File Output Format Counters
  Bytes Written=75
2022-06-09 23:39:14,350 INFO streaming.StreamJob: Output directory: /Movies/Output2

C:\hadoop-3.2.2\sbin>
```

```
Administrator: Command Prompt

C:\hadoop-3.2.2\sbin>hadoop fs -cat /Movies/Output2/part-00000
Users who has assigned lowest average rating:
13496 0.18181818181818182

C:\hadoop-3.2.2\sbin>
```



In [11]:

```
map_red2 = {'Mapper:1': [47057, 45606, 43138, 46477],
            'Mapper:2': [46607, 45597, 43345, 47106],
            'Mapper:3': [46384, 49232, 43931, 44375],
            'Mapper:4': [40332, 37407, 41374, 43448]
            }

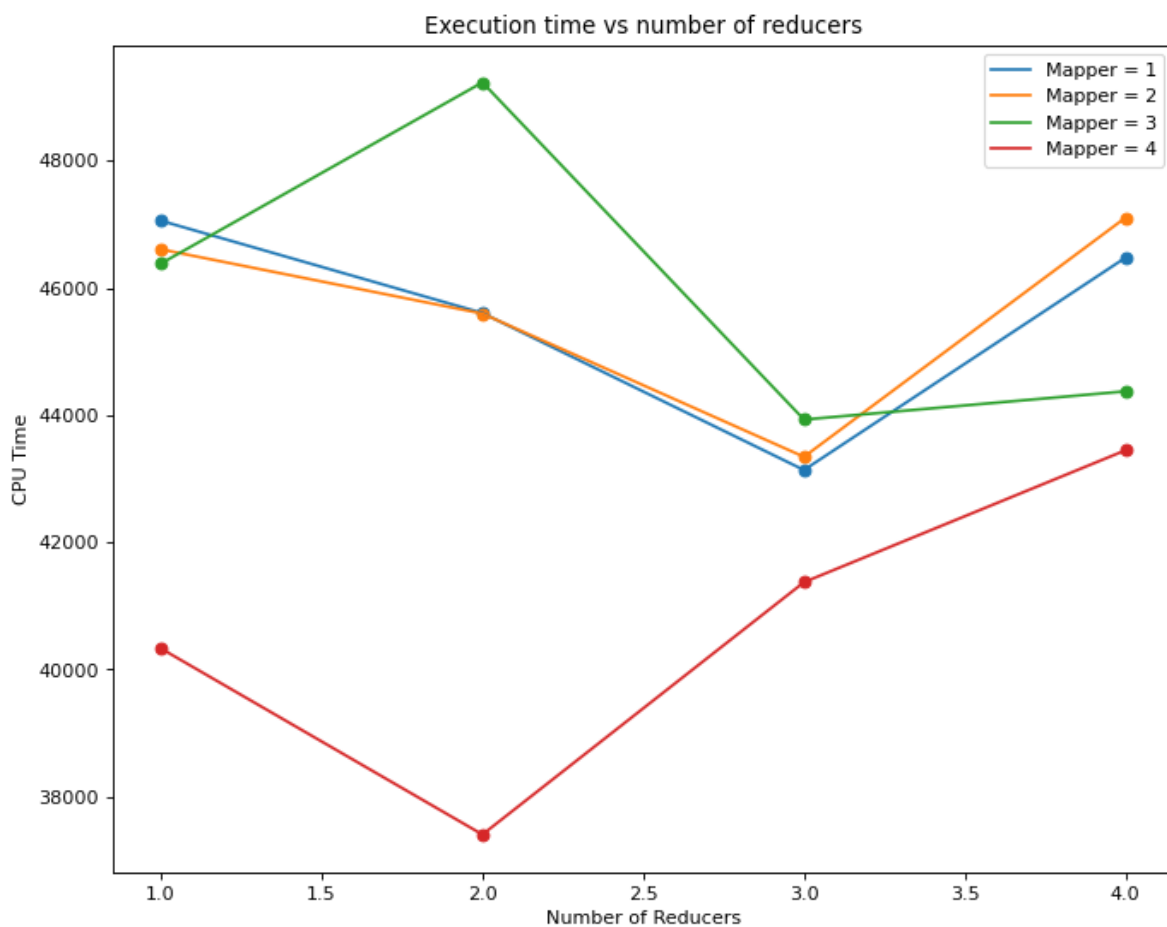
time_ = pd.DataFrame.from_dict(map_red2, orient = 'index')
time_.columns = ['Reducer:1', 'Reducer:2', 'Reducer:3', 'Reducer:4']
time_ = time_.style.set_properties(**{
    'background-color': 'grey',
    'font-size': '20pt',
})
time_
```

Out[11]:

	Reducer:1	Reducer:2	Reducer:3	Reducer:4
Mapper:1	47057	45606	43138	46477
Mapper:2	46607	45597	43345	47106
Mapper:3	46384	49232	43931	44375
Mapper:4	40332	37407	41374	43448

In [12]:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 8), dpi=80)
plt.plot([1,2,3,4], map_red2['Mapper:1'], label = 'Mapper = 1')
plt.scatter([1,2,3,4], map_red2['Mapper:1'])
plt.plot([1,2,3,4], map_red2['Mapper:2'], label = 'Mapper = 2')
plt.scatter([1,2,3,4], map_red2['Mapper:2'])
plt.plot([1,2,3,4], map_red2['Mapper:3'], label = 'Mapper = 3')
plt.scatter([1,2,3,4], map_red2['Mapper:3'])
plt.plot([1,2,3,4], map_red2['Mapper:4'], label = 'Mapper = 4')
plt.scatter([1,2,3,4], map_red2['Mapper:4'])
plt.legend()
plt.xlabel('Number of Reducers')
plt.ylabel('CPU Time')
plt.title('Execution time vs number of reducers')
plt.show()
```



**From the plot above we can see that the execution time is least when 4 mappers and 2 reducers are used.**

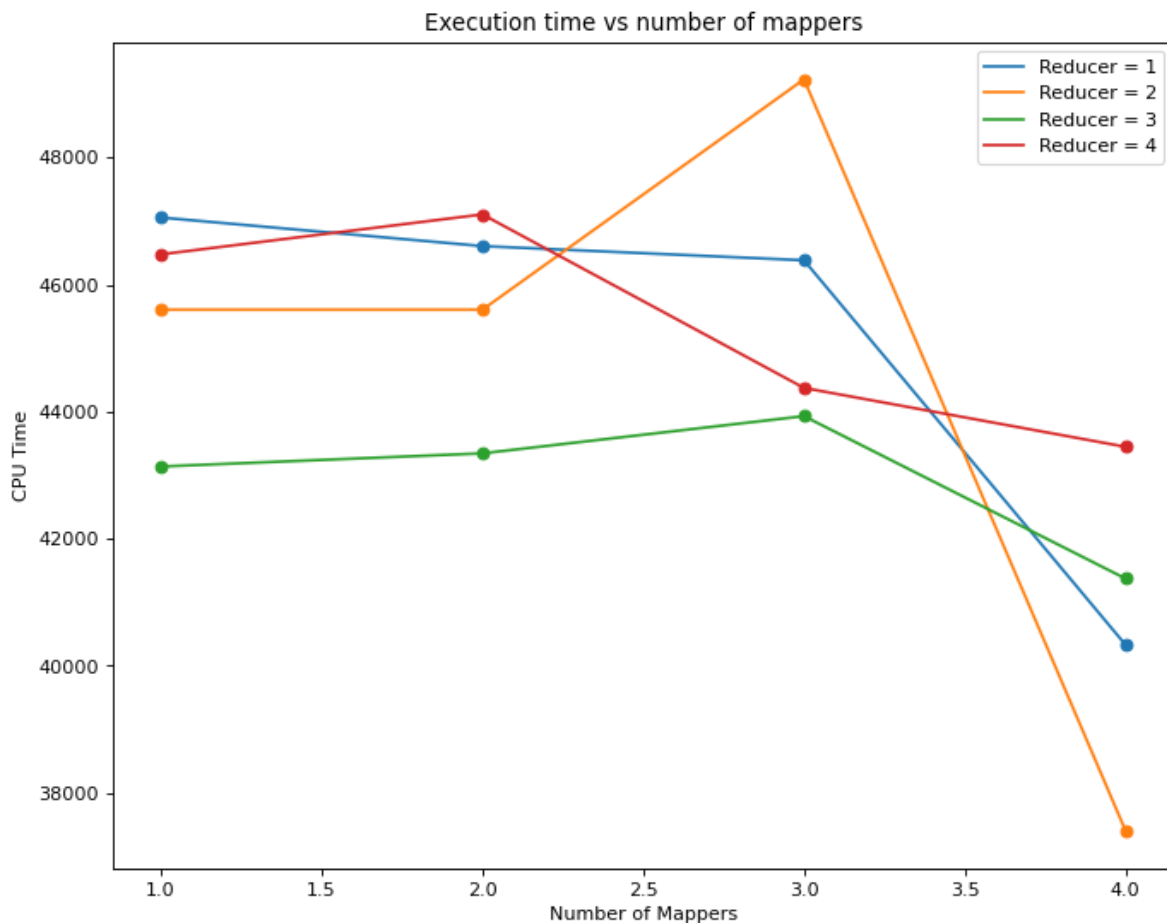
In [14]:

```

red_map2 = {'Reducer:1': [47057, 46607, 46384, 40332],
            'Reducer:2': [45606, 45606, 49232, 37407],
            'Reducer:3': [43138, 43345, 43931, 41374],
            'Reducer:4': [46477, 47106, 44375, 43448]}

plt.figure(figsize=(10, 8), dpi=80)
plt.plot([1,2,3,4], red_map2['Reducer:1'], label = 'Reducer = 1')
plt.scatter([1,2,3,4], red_map2['Reducer:1'])
plt.plot([1,2,3,4], red_map2['Reducer:2'], label = 'Reducer = 2')
plt.scatter([1,2,3,4], red_map2['Reducer:2'])
plt.plot([1,2,3,4], red_map2['Reducer:3'], label = 'Reducer = 3')
plt.scatter([1,2,3,4], red_map2['Reducer:3'])
plt.plot([1,2,3,4], red_map2['Reducer:4'], label = 'Reducer = 4')
plt.scatter([1,2,3,4], red_map2['Reducer:4'])
plt.legend()
plt.xlabel('Number of Mappers')
plt.ylabel('CPU Time')
plt.title('Execution time vs number of mappers')
plt.show()

```



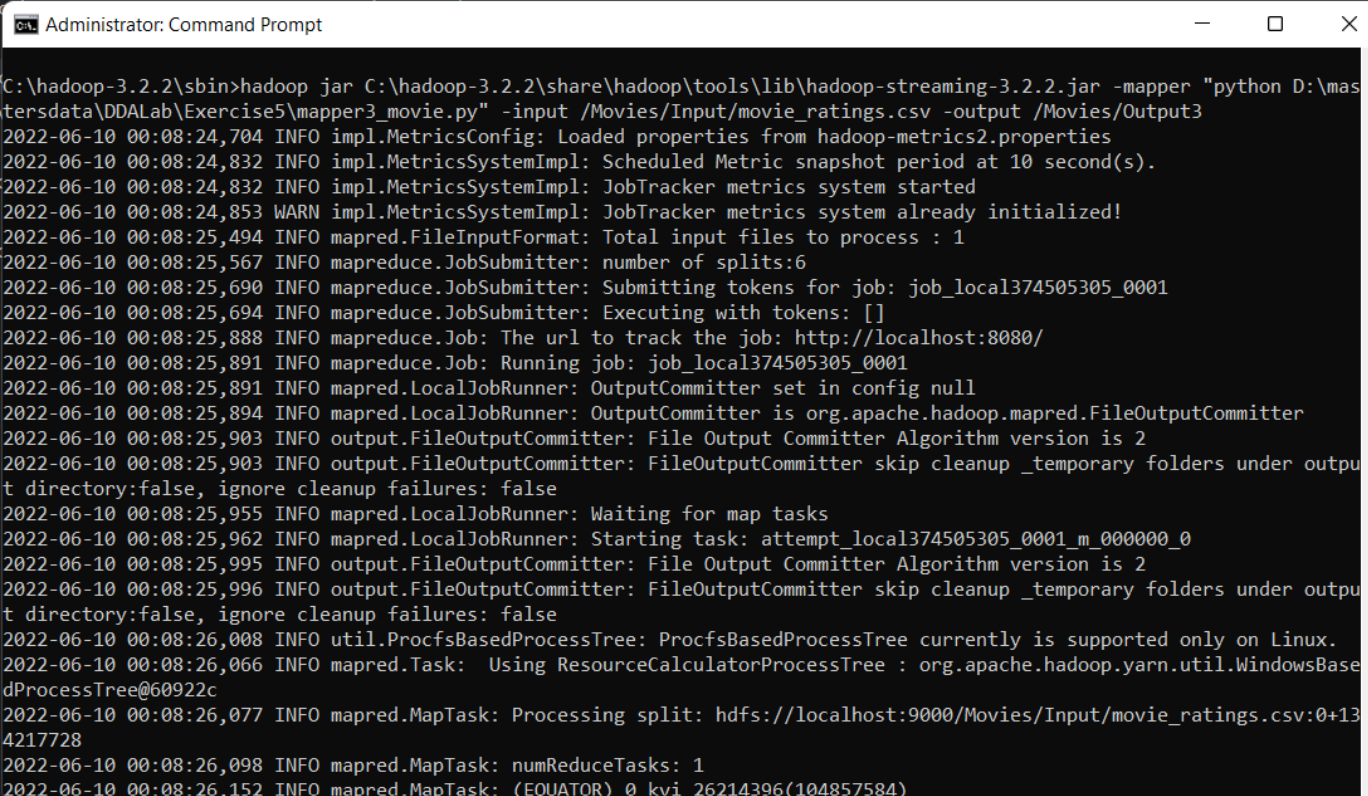
### 3. Find the highest average rated movie genre?

Some of the movies belong to more than one genre and hence the mapper splits these genres as well and assign them the rating of the movie they are a part of. As a output from the mapper we get genre and the corresponding rating. The aim of the reducer is to first find the average rating for each genre and then output the genre that has the highest average rating.

## Mapper :

In [ ]:

```
#!/usr/bin/python39
import sys
import pandas as pd
input = sys.stdin
next(input)
for line in input:
    l_genre = []          # if one movie has more than one genre
    genres = line.split(';')[2]
    rating = line.split(';')[4]
    try:
        genres = genres.strip()
        l_genre = genres.split('|')    # split the genres
        for g in l_genre:
            if rating:
                print(g, ';', rating)    # give each of the genre same rating as they belong to
    except:
        if rating:
            print(genres, ';', rating)
```



```
Administrator: Command Prompt
C:\hadoop-3.2.2\sbin>hadoop jar C:\hadoop-3.2.2\share\hadoop\tools\lib\hadoop-streaming-3.2.2.jar -mapper "python D:\mas
tersdata\DDALab\Exercise5\mapper3_movie.py" -input /Movies/Input/movie_ratings.csv -output /Movies/Output3
2022-06-10 00:08:24,704 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2022-06-10 00:08:24,832 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2022-06-10 00:08:24,832 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2022-06-10 00:08:24,853 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2022-06-10 00:08:25,494 INFO mapred.FileInputFormat: Total input files to process : 1
2022-06-10 00:08:25,567 INFO mapreduce.JobSubmitter: number of splits:6
2022-06-10 00:08:25,690 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local374505305_0001
2022-06-10 00:08:25,694 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-10 00:08:25,888 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2022-06-10 00:08:25,891 INFO mapreduce.Job: Running job: job_local374505305_0001
2022-06-10 00:08:25,891 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2022-06-10 00:08:25,894 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2022-06-10 00:08:25,903 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-06-10 00:08:25,903 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
2022-06-10 00:08:25,955 INFO mapred.LocalJobRunner: Waiting for map tasks
2022-06-10 00:08:25,962 INFO mapred.LocalJobRunner: Starting task: attempt_local374505305_0001_m_000000_0
2022-06-10 00:08:25,995 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-06-10 00:08:25,996 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
2022-06-10 00:08:26,008 INFO util.ProcfsBasedProcessTree: ProcfsBasedProcessTree currently is supported only on Linux.
2022-06-10 00:08:26,066 INFO mapred.Task: Using ResourceCalculatorProcessTree : org.apache.hadoop.yarn.util.WindowsBase
dProcessTree@60922c
2022-06-10 00:08:26,077 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/Movies/Input/movie_ratings.csv:0+13
4217728
2022-06-10 00:08:26,098 INFO mapred.MapTask: numReduceTasks: 1
2022-06-10 00:08:26,152 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
```

[illegible]

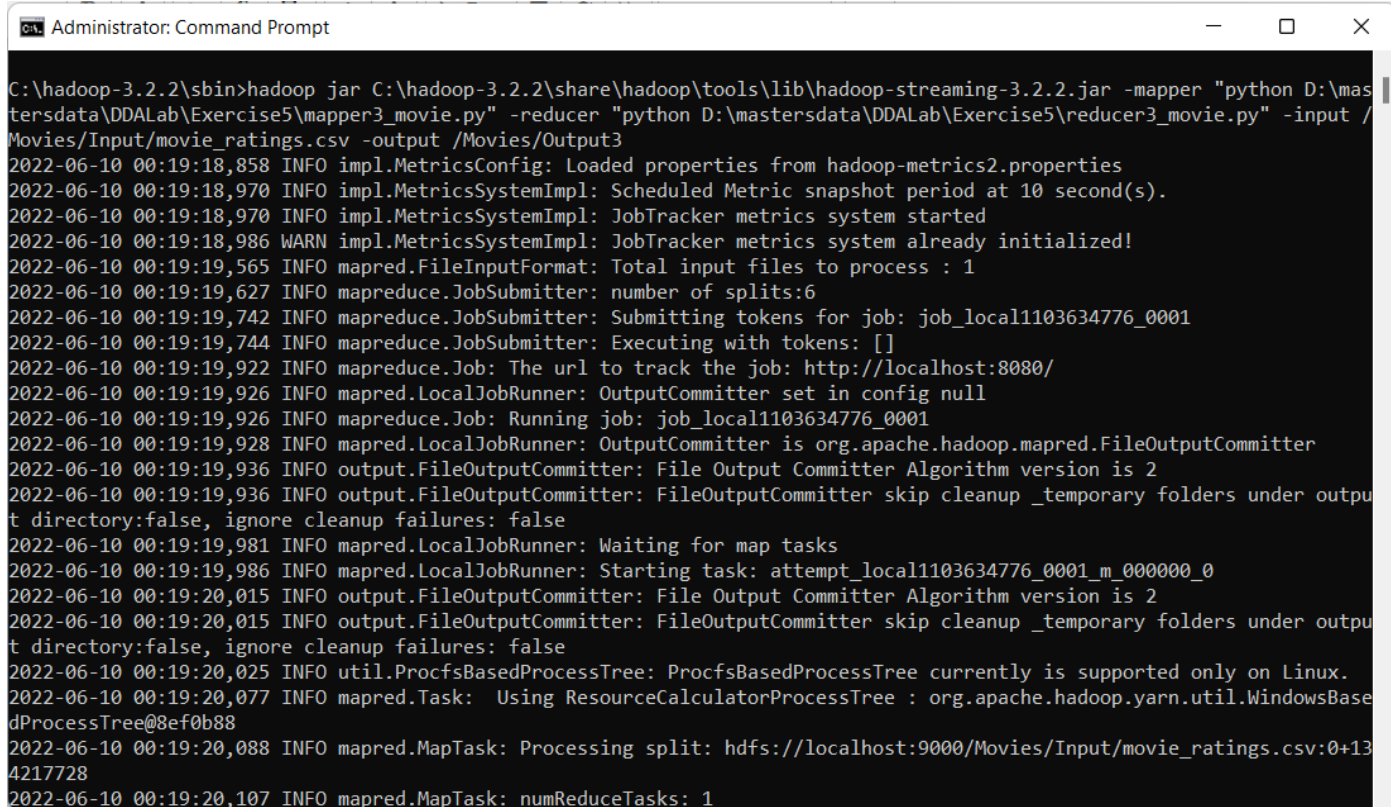
### Reducer :

In [ ]:

```
#!/usr/bin/python39
import sys
input = sys.stdin
current_genre = None    # to check if current and next genre are same
current_rating = 0      # to calculate rating for each genre
avg_rating = {}         # average rating per genre
count = 0

for line in input:
    genre = line.split(';')[0]
    rating = line.split(';')[1]
    try:
        rating = float(rating)
    except ValueError:
        continue
    if genre == current_genre:
        current_rating += rating
        count += 1
    else:
        if current_genre:
            avg_rating[current_genre] = current_rating/count
            count = 0
        else:
            print('Highest average rated movie genre:')
            current_genre = genre
            current_rating = rating
            count += 1
if current_genre == genre:
    avg_rating[current_genre] = current_rating/count

genre_rating = [k for k, v in sorted(avg_rating.items(), key=lambda item: item[1], reverse=True)]
print(genre_rating)
```



```
Administrator: Command Prompt

C:\hadoop-3.2.2\sbin>hadoop jar C:\hadoop-3.2.2\share\hadoop\tools\lib\hadoop-streaming-3.2.2.jar -mapper "python D:\mas
tersdata\DDALab\Exercise5\mapper3_movie.py" -reducer "python D:\mastersdata\DDALab\Exercise5\reducer3_movie.py" -input /
Movies/Input/movie_ratings.csv -output /Movies/Output3
2022-06-10 00:19:18,858 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2022-06-10 00:19:18,970 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2022-06-10 00:19:18,970 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2022-06-10 00:19:18,986 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2022-06-10 00:19:19,565 INFO mapred.FileInputFormat: Total input files to process : 1
2022-06-10 00:19:19,627 INFO mapreduce.JobSubmitter: number of splits:6
2022-06-10 00:19:19,742 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1103634776_0001
2022-06-10 00:19:19,744 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-06-10 00:19:19,922 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2022-06-10 00:19:19,926 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2022-06-10 00:19:19,926 INFO mapreduce.Job: Running job: job_local1103634776_0001
2022-06-10 00:19:19,928 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2022-06-10 00:19:19,936 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-06-10 00:19:19,936 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
2022-06-10 00:19:19,981 INFO mapred.LocalJobRunner: Waiting for map tasks
2022-06-10 00:19:19,986 INFO mapred.LocalJobRunner: Starting task: attempt_local1103634776_0001_m_000000_0
2022-06-10 00:19:20,015 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-06-10 00:19:20,015 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
2022-06-10 00:19:20,025 INFO util.ProcfsBasedProcessTree: ProcfsBasedProcessTree currently is supported only on Linux.
2022-06-10 00:19:20,077 INFO mapred.Task: Using ResourceCalculatorProcessTree : org.apache.hadoop.yarn.util.WindowsBase
dProcessTree@8ef0b88
2022-06-10 00:19:20,088 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/Movies/Input/movie_ratings.csv:0+13
4217728
2022-06-10 00:19:20,107 INFO mapred.MapTask: numReduceTasks: 1
```



Administrator: Command Prompt

```

Combine input records=0
Combine output records=0
Reduce input groups=195
Reduce shuffle bytes=427828260
Reduce input records=25967183
Reduce output records=2
Spilled Records=76429632
Shuffled Maps =6
Failed Shuffles=0
Merged Map outputs=6
GC time elapsed (ms)=424
Total committed heap usage (bytes)=3126853632
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=710272431
File Output Format Counters
  Bytes Written=48
2022-06-10 00:22:04,109 INFO streaming.StreamJob: Output directory: /Movies/Output3
C:\hadoop-3.2.2\sbin>hadoop fs -cat /Movies/Output3/part-00000
Highest average rated movie genre:
Film-Noir

```

In [15]:

```

map_red3 = {'Mapper:1': [175596, 179229, 189570, 179427],
            'Mapper:2': [175484, 188965, 193225, 193554],
            'Mapper:3': [178469, 187742, 194604, 195691],
            'Mapper:4': [175544, 185514, 188104, 181074]}

time_2 = pd.DataFrame.from_dict(map_red3, orient = 'index')
time_2.columns = ['Reducer:1', 'Reducer:2', 'Reducer:3', 'Reducer:4']
time_2 = time_2.style.set_properties(**{
    'background-color': 'grey',
    'font-size': '20pt',
})
time_2

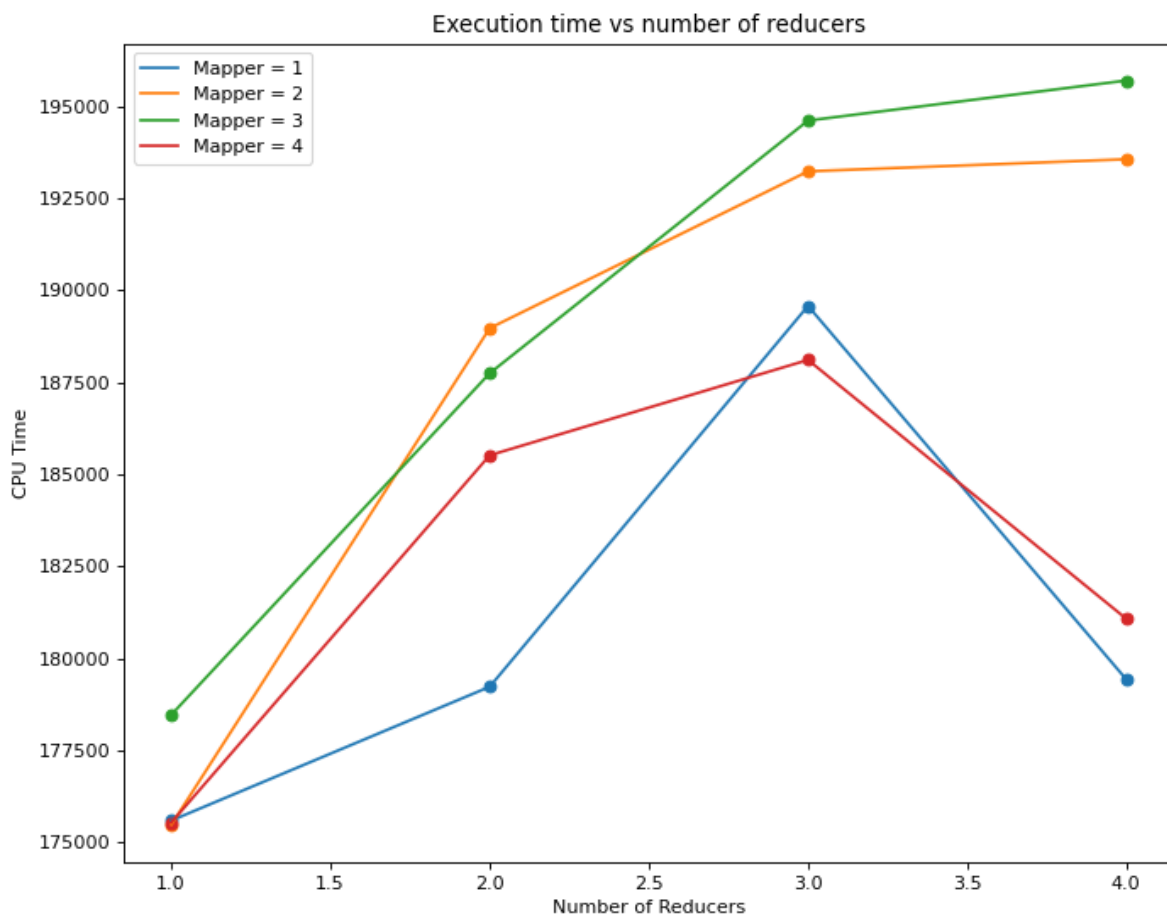
```

Out[15]:

	Reducer:1	Reducer:2	Reducer:3	Reducer:4
Mapper:1	175596	179229	189570	179427
Mapper:2	175484	188965	193225	193554
Mapper:3	178469	187742	194604	195691
Mapper:4	175544	185514	188104	181074

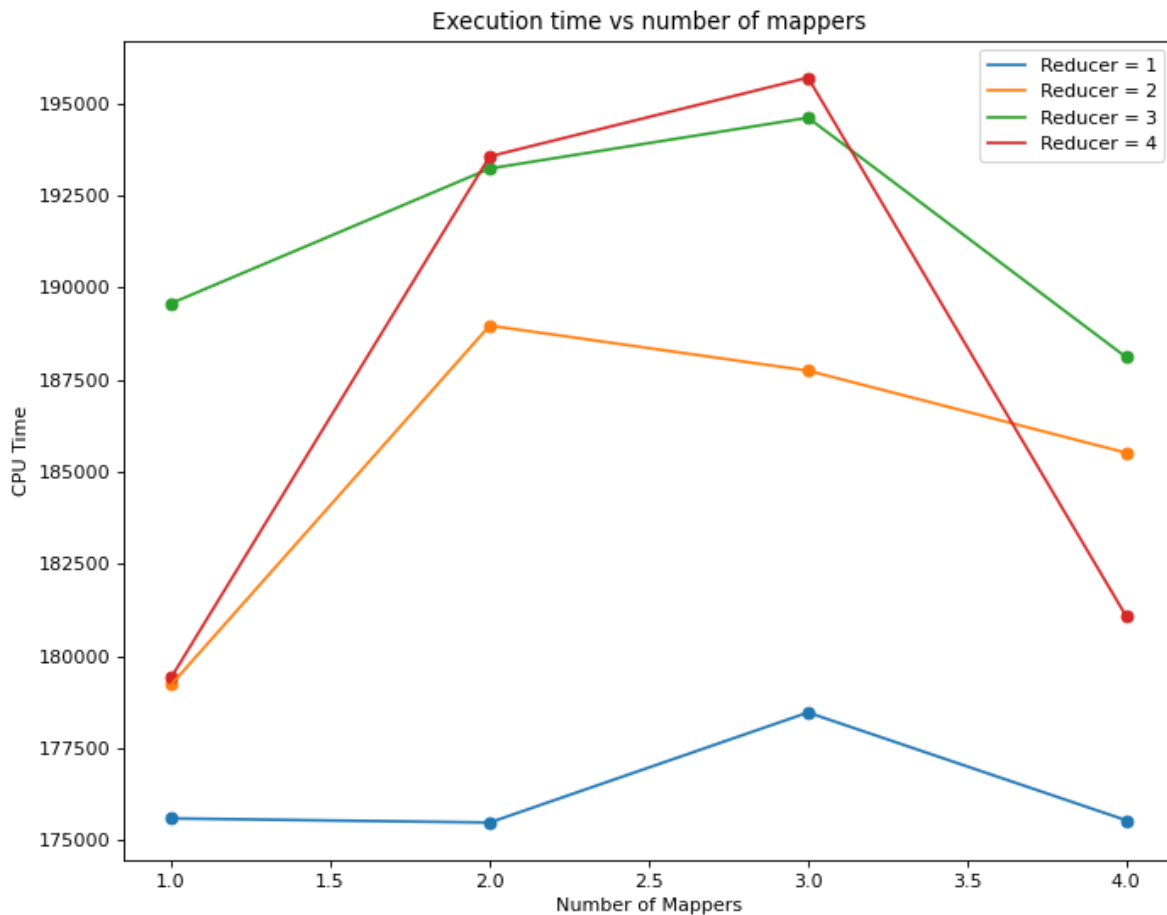
In [16]:

```
plt.figure(figsize=(10, 8), dpi=80)
plt.plot([1,2,3,4], map_red3['Mapper:1'], label = 'Mapper = 1')
plt.scatter([1,2,3,4], map_red3['Mapper:1'])
plt.plot([1,2,3,4], map_red3['Mapper:2'], label = 'Mapper = 2')
plt.scatter([1,2,3,4], map_red3['Mapper:2'])
plt.plot([1,2,3,4], map_red3['Mapper:3'], label = 'Mapper = 3')
plt.scatter([1,2,3,4], map_red3['Mapper:3'])
plt.plot([1,2,3,4], map_red3['Mapper:4'], label = 'Mapper = 4')
plt.scatter([1,2,3,4], map_red3['Mapper:4'])
plt.legend()
plt.xlabel('Number of Reducers')
plt.ylabel('CPU Time')
plt.title('Execution time vs number of reducers')
plt.show()
```



In [17]:

```
red_map3 = {'Reducer:1': [175596, 175484, 178469, 175544],  
            'Reducer:2': [179229, 188965, 187742, 185514],  
            'Reducer:3': [189570, 193225, 194604, 188104],  
            'Reducer:4': [179427, 193554, 195691, 181074]  
}  
plt.figure(figsize=(10, 8), dpi=80)  
plt.plot([1,2,3,4], red_map3['Reducer:1'], label = 'Reducer = 1')  
plt.scatter([1,2,3,4], red_map3['Reducer:1'])  
plt.plot([1,2,3,4], red_map3['Reducer:2'], label = 'Reducer = 2')  
plt.scatter([1,2,3,4], red_map3['Reducer:2'])  
plt.plot([1,2,3,4], red_map3['Reducer:3'], label = 'Reducer = 3')  
plt.scatter([1,2,3,4], red_map3['Reducer:3'])  
plt.plot([1,2,3,4], red_map3['Reducer:4'], label = 'Reducer = 4')  
plt.scatter([1,2,3,4], red_map3['Reducer:4'])  
plt.legend()  
plt.xlabel('Number of Mappers')  
plt.ylabel('CPU Time')  
plt.title('Execution time vs number of mappers')  
plt.show()
```



**Since most of the movies belong to many genres and when all these genres were listed the output from the mapper was very huge and thus we can see the effect of this in our execution time with the increase in the number of mapper and reducer, the execution time first increased and then decreased over a certain value.**