IERG4330/ESTR4316/IEMS5730 Spring 2022
Homework #1 (Similar Users Detection)

Release date: Jan 28, 2022

Due date: Feb 16, 2022 (Wed) 11:59:00 p.m. (i.e. midnight)

The solution will be posted soon after the deadline. No late homework will be accepted!

Every Student MUST include the following statement, together with his/her signature in the submitted homework.

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

Signed (Student Fox III Eddie Christopher) Date: February 16, 2022

Name Fox III, Eddie Christopher SID 1155160788

Submission notice:

- Submit your homework via the elearning system.
- Only students who have not taken IERG4300/ESTR4300/IEMS5709 are required to submit this assignment.

General homework policies:

A student may discuss the problems with others. However, the work a student turns in must be created COMPLETELY by oneself ALONE. A student may not share ANY written work or pictures, nor may one copy answers from any source other than one's own brain.

Each student **MUST LIST** on the homework paper the **name of every person he/she has discussed or worked with**. If the answer includes content from any other source, the student **MUST STATE THE SOURCE**. Failure to do so is cheating and will result in sanctions. Copying answers from someone else is cheating even if one lists their name(s) on the homework.

If there is information you need to solve a problem but the information is not stated in the problem, try to find the data somewhere. If you cannot find it, state what data you need, make a reasonable estimate of its value, and justify any assumptions you make. You will be graded not only on whether your answer is correct, but also on whether you have done an intelligent analysis.

IEMS 5730: Spring 2022 Assignment #1

Table of Contents (bolded = submission /graded content)

Signed Academic Honesty Statement……Page 1

Table of Contents……..Page 2

IE-DIC setup…….Page 3

Mr. Job……..Page 6

Initial Attempts with Movie key Based Approach……...Page 6

Attempts with User-ID Key based Approach…….....Page 8

Hybrid Approach with Separate Jobs.....Page 10

Part B MapReduce Job #1.....Page 10

Mapper code.....Page 11

Reducer code..... Page 12

Part A MapReduce Job #1.....Page 21

Mapper code.....Page 22

Initial Reducer code.....Page 24

Final Reducer code.....Page 28

Part A MapReduce Job #2.....Page 36

Mapper code.....Page 38

Reducer code.....Page 41

Part A Final Submission: Top 10 Pairs for Each Dataset.....Page 55

Part B MapReduce Job #2.....Page 56

Initial Mapper code.....Page 57

Mapper Attempt #2 code.....Page 60

Issues / design considerations before final attempts..... Page 62

Final Mapper code.....Page 64

Final Reducer code.....Page 65

Part B Final Submission: Top 3 Similar Users.....Page 75

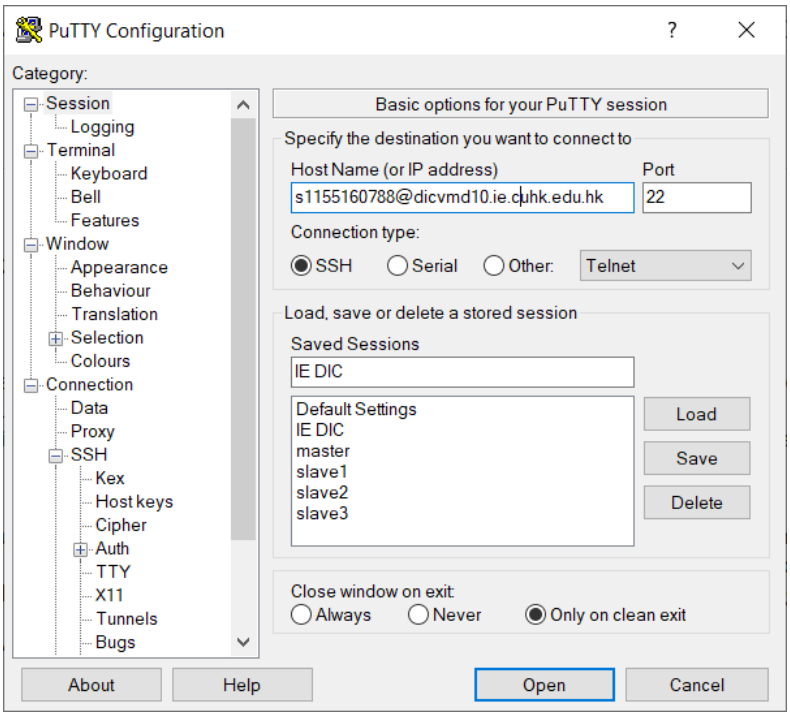
Part C.....Page 77

Part 1: Setup and Analysis

IE-DIC setup:

For this assignment, I opted into using the IE DIC Cluster for this assignment due to its superior calculation capabilities compared to my 4 node cluster. I first follow the IE VPN guide to install GlobalProtect with the appropriate configuration, and connect to VPN.

I set up the hostname to SSH into the IE DIC cluster using the hostname from the Blackboard announcement.



```
s1155160788@dicvmd10:~
Using username "s1155160788".
s1155160788@dicvmd10.ie.cuhk.edu.hk's password:
Last failed login: Mon Feb  7 15:55:47 HKT 2022 from 10.10.54.32 on ssh:notty
There were 2 failed login attempts since the last successful login.
[s1155160788@dicvmd10 ~]$
```

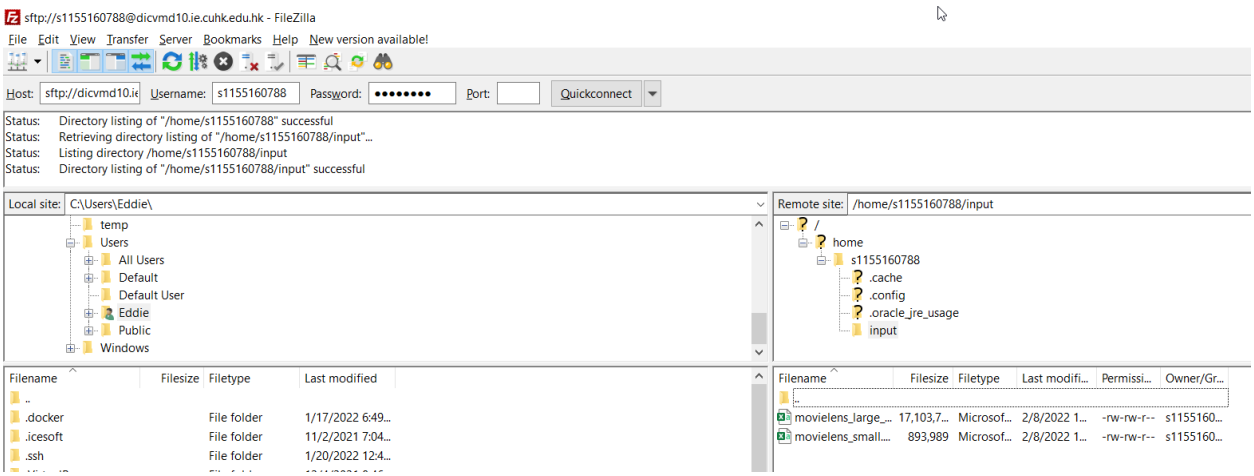
I am able to SSH into the IE DIC cluster using Putty fairly quickly after two failures. The first failure came from trying to use my IE account password, then the second one was from using my IE account username (fie021) with the IE account password, before I remembered to use the IE DIC password provided from the blackboard grade.

```
[s1155160788@dicvmd10 ~]$ ls -al
total 20
drwx------   5 s1155160788 s1155160788  137 Feb  7 16:05 .
drwxr-xr-x. 53 root        root        4096 Jan 28 13:14 ..
-rw-------   1 s1155160788 s1155160788  152 Feb  8 19:55 .bash_history
-rw-r--r--   1 s1155160788 s1155160788   18 Nov 25 00:33 .bash_logout
-rw-r--r--   1 s1155160788 s1155160788  193 Nov 25 00:33 .bash_profile
-rw-r--r--   1 s1155160788 s1155160788  231 Nov 25 00:33 .bashrc
drwxrwxr-x   3 s1155160788 s1155160788   18 Feb  7 15:58 .cache
drwxrwxr-x   3 s1155160788 s1155160788   18 Feb  7 15:58 .config
drwxrwxr-x   2 s1155160788 s1155160788   40 Feb  7 16:02 .oracle_jre_usage
[s1155160788@dicvmd10 ~]$ pwd
/home/s1155160788
[s1155160788@dicvmd10 ~]$ mkdir input
[s1155160788@dicvmd10 ~]$ cd input
[s1155160788@dicvmd10 input]$
```

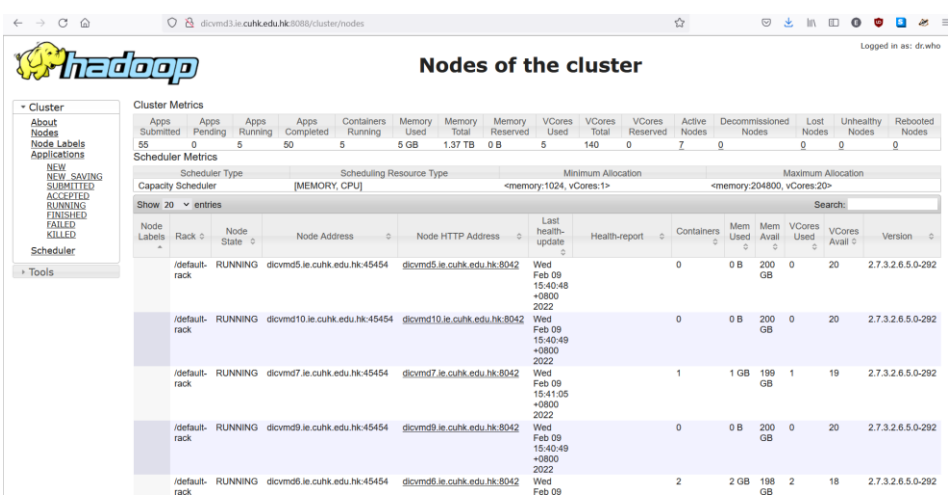
I take a look at my “surroundings” and determine I am in the home directory under my username. I create an input directory for storing the CSV database files.

```
[s1155160788@dicvmd10 input]$ wget https://mobitec.ie.cuhk.edu.hk/ierg4330/static_files/assignments/movielens_small.csv
--2022-02-09 14:03:56--  https://mobitec.ie.cuhk.edu.hk/ierg4330/static_files/assignments/movielens_small.csv
Resolving mobitec.ie.cuhk.edu.hk (mobitec.ie.cuhk.edu.hk)... 137.189.99.12
Connecting to mobitec.ie.cuhk.edu.hk (mobitec.ie.cuhk.edu.hk)|137.189.99.12|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 893989 (873K) [text/csv]
Saving to: 'movielens_small.csv'

100%[===================================================================================================================================================>] 893,989     --.-K/s   in 0.004s

2022-02-09 14:03:56 (229 MB/s) - 'movielens_small.csv' saved [893989/893989]

[s1155160788@dicvmd10 input]$ wget https://mobitec.ie.cuhk.edu.hk/ierg4330/static_files/assignments/movielens_large_updated.csv
--2022-02-09 14:04:28--  https://mobitec.ie.cuhk.edu.hk/ierg4330/static_files/assignments/movielens_large_updated.csv
Resolving mobitec.ie.cuhk.edu.hk (mobitec.ie.cuhk.edu.hk)... 137.189.99.12
Connecting to mobitec.ie.cuhk.edu.hk (mobitec.ie.cuhk.edu.hk)|137.189.99.12|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 17103777 (16M) [text/csv]
Saving to: 'movielens_large_updated.csv'

100%[===================================================================================================================================================>] 17,103,777  --.-K/s   in 0.08s

2022-02-09 14:04:28 (202 MB/s) - 'movielens_large_updated.csv' saved [17103777/17103777]

[s1155160788@dicvmd10 input]$ ls -al
total 17580
drwxrwxr-x 2 s1155160788 s1155160788       68 Feb  9 14:04 .
drwx------ 6 s1155160788 s1155160788      150 Feb  9 13:59 ..
-rw-rw-r-- 1 s1155160788 s1155160788 17103777 Feb  8 11:06 movielens_large_updated.csv
-rw-rw-r-- 1 s1155160788 s1155160788   893989 Feb  8 11:06 movielens_small.csv
[s1155160788@dicvmd10 input]$
```

I use wget command with each of the URL's specified under the [1] and [2] sources of the Assignment 1 PDF (not to be confused with my own source bracket numbering).



I am using Filezilla on my local Windows 10 laptop to send files to and from the IE DIC cluster. Generally, I code on the laptop using Visual Studio Code before sending the files with Filezilla to the IE DIC cluster. As I determined that the IE DIC cluster is using Python Version 2.7.5 using "Python –version", I install this version on my laptop as well.



I confirmed that the Hadoop / YARN Web UI is working well at http://dicvmd3.ie.cuhk.edu.hk:8088/cluster/apps as mentioned in the Blackboard announcement for IE-DIC, as long as I am connected to the IE VPN.

Mr. Job

Regarding the algorithmic setup of this assignment, I figured that I would need at minimum two mapreduce jobs in order to accomplish Part A. I have only programmed in Java once, so I have limited experience. As a result, I wanted to leverage Python to program the MapReduce, using Hadoop Streaming as we did in Assignment 0 and as noted in Assignment 1 PDF. I saw ways to chain multiple MapReduce jobs using Java, but since I wasn't using Java, I needed another way.

I recalled MrJob from Tutorial 5 of IERG 4300 Fall 2020 [1] which could use Python based programming for MapReduce jobs, allowed for local testing, and had the capability to chain MapReduce jobs together. Per blackboard announcement, Mrjob is also already installed on the IE DIC cluster, so I thought it would be perfect for my needs. I started reading the official Mr. Job documentation [2] in preparation for using it, but it seemed fairly complicated, and during my formulation of the MapReduce jobs, I encountered limitations that made Assignment 1 seem impossible by direct chaining of MapReduce jobs, which was the main reason for using Mr. Job, so I ultimately decided against using it, and will program in normal Python.

Initial Attempts with Movie ID Key Based Approach:

Input: The Movielens datasets with X users and Y movies each. Each line containers one user ID and one movie in the format: User ID, Movie ID.

The most obvious approach for the first MapReduce job is to use the User ID as a key and end up with a reducer output that has the movies for each user.

This approach may be the simplest and easiest to implement, but could run into scalability issues, as to solve part A, we would need to compare the list of movies under each user to every other user to see how many they have in common. This is at least an $O(N^2)$ operation and would not scale well into the large dataset. At least not without some tweaking / modifications.

[3] details another possible approach where we reverse the key and the value for the first MapReduce job, such that the movie is the key and the user IDs are the values. This could be better because each user only has a small number of movies assigned to them compared to the number of users and movies, and it is probably not common for users to have many movies in common, so we can instead generate a list of movies and the users that have watched each one, then use further MapReduce jobs to enumerate upon the pairs of users for each movie.

I decided to consider this later approach due to scalability concerns with the large dataset, as the first / obvious method, with 69,878 users in the large dataset would create something around 4,882,934,884 pairs for comparison. Even though many of these pairs would have nothing in common, it would still take up memory.

Map Reduce Job (movie key version) #1:

Mapper #1:

Input: MovieLens Dataset in format User ID, Movie ID

Output key: Movie ID

Output value: User ID

Example with movies A and B in Assignment 1 PDF:

Movie A User A

Movie B User A

Movie C User A

Movie B User B

Movie C User B

Movie D User B

Reducer #1:

Input: Shown above

Output key: Movie ID.

Output value: Array of User ID's that have watched each movie.

Example with full data in Assignment 1 PDF sample table:

Movie A [a]

Movie B [a b c d]

Movie C [a b d]

Movie D [b c]

When drafting this approach, I struggled because even though it had considerable scalability advantages. It was primarily suited for Part A, because we can use the array of users under each movie to enumerate all the permutations of pairs of users for that movie. For example if movie A was watched by users A, B and C, the pairs would be A-B, A-C, and B-C). Through another MapReduce job, we sum up all the permutations to determine how many movies each user pair had watched.

However, when I thought about how the code could be extended to Part B, I didn't see an easy way to calculate similarity. Based on the inclusion-exclusion principle mentioned in Assignment 1 PDF, we could use the movie key approach to calculate the intersection of A and B, which is the numerator of the similarity formula (and what is asked in Part A), but for the denominator

because we would need a per user count of the movies they had watched, because the total denominator is ([A+B] – [Intersection of A and B]). We need to use the second map reduce job to calculate the sum of movies watched by both users in a pair for Part A, and also as the intersection for similarity calculation in Part B, but this MapReduce job no longer has the raw data to work with (at least if we are chaining the MapReduce jobs together), so it would be difficult to get the number of movies individually watched by users A and B in a pair.

Inspired in principle by [4] from the Assignment 1 PDF (but not the code since it is in Java instead of Python / MrJob), I thought to create a compound key of sorts for the first MapReduce job that calculates the total number of movies watched by each user during the first MapReduce job so that when this information is fed into the second MapReduce job, there is enough information for the similarity calculations of Part B.

Unfortunately, I almost immediately realized my error with this approach. It wouldn't be a compound key (which was movie), but a compound value bundled with users. Even if I could get the first mapper to output it as (User A, 1) instead of User A, I wouldn't be able to sum them up, because intermediate output is grouped by the key, so the different compound values for each user would likely go to different reducers.

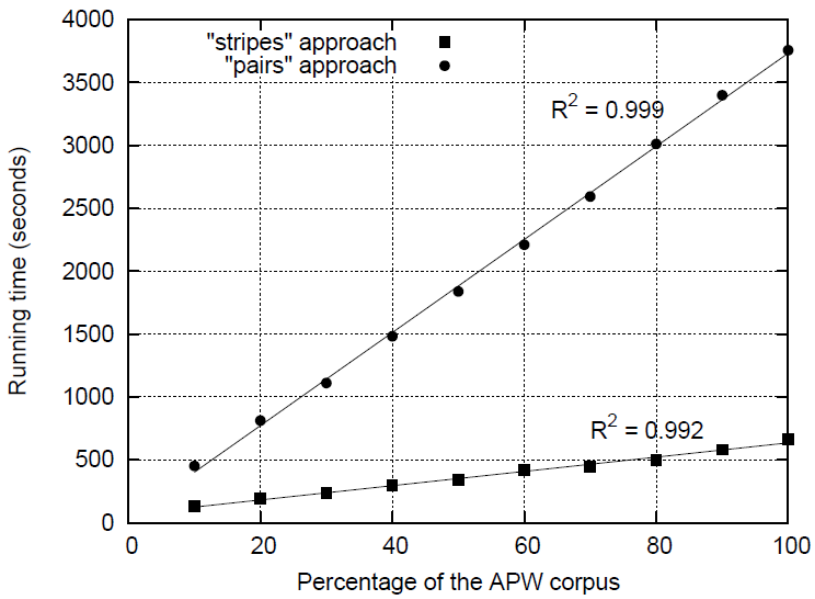
With no clear way to circumvent the issues with extending the code using the movie key approach into Part B, I tried to use the User ID key based approach to continue.

Attempts with User-ID Key based Approach:

I previously calculated that a User ID key based approach, if $O(N^2)$ complexity with the pairs would create 4,882,934,884 pairs, and if we assume that each user in the pair has roughly 8 bytes of information, it would take 39,063,479,072 billion bytes, which is 36.38 GiB. At 36 GiB, it would be acceptable both with my 300 GB HDFS 4 node hadoop cluster and the 10 TB IE-DIC cluster. While the approach would eventually reach a scaling limit, I felt I should focus on designing something good enough for the assignment.

Since I was using an algorithm with relatively high time complexity, I needed to do what I could to keep the memory usage and scaling under control. This similarity user detection problem highly reminds me of one of our scheduled readings for IEMS 5730, JLin's MapReduce book, Chapters 3.1-3.4 [5]. Specifically, in Chapter 3.2, [5] details the Pairs and Stripes algorithms for calculating co-occurrence matrices.

It is noted that the Stripes algorithm can benefit more from local combiners / aggregation, and that the Stripes algorithm scales better as the total size increases [5], however it is more complex to implement. My plan was to implement the Stripes algorithm, as it applies to this problem first unless it is too complex, in which case I would switch to the pairs method. Below is a picture of the runtime comparison between pairs and stripes approaches as the size increases [5].



While thinking about how to formulate my MapReduce jobs, besides [5], I also referred to [6] slides 16-19 and [7] slides 28-36. These slides are mainly a summary of the JLin Map Reduce text, Chapter 3.2.

MapReduce Job #1 (User ID key version):

Mapper Job #1:

Input: Movielens datasets (User ID, Movie ID)

Output key: User ID

Output value: Count of movies watched, Movie ID.

Count was to be 1 during mapper and summed up during reducer stage. This allows the frequencies to be calculated later. Although it is not needed in Part A, summing this information in the reducer for stage 1 allows me to change the key in the output of the mapper in Job #2.

The intermediate output is a bunch of lines as follows (pseudocode):

User A (1 movie a)

User A (1 movie b)

User A (1 movie c)

User B (1 movie b)

User B (1 movie c)

User B (1 movie d)

This is then passed as input to the reducer job.

Reducer Job:

Input: Shown above.

Output key: User ID

Output value: Array that has the number of movies watched by the user first, followed by the movies they have watched.

For User A and B, the output is as follows:

User A (3 a b c)

User B (3 b c d)

In the second MapReduce job, I would need to enumerate all possible pairs of users, then for each pair, I would compare each element of the array of the first user to each element of the array of the second user. Whenever there is a match, I increment the number of common elements. If there is no match, I would output 0.

Unfortunately, there was no clear way to enumerate all the pairs of users, because MapReduce only works one line at a time, so I was stuck both with the approaches based on Movie ID as the key, and those with User ID as the key.

Hybrid Approach with Separate Jobs:

After emailing the TA about this and receiving a response, I learned that I don't *have* to chain the MapReduce jobs together, but can run them separately. I can also use the generated output file of one MapReduce job and load it into the next job. The process of how to do this is described in Slides 40-49 of [8].

Armed with the knowledge that I can make separate MapReduce jobs, I first create a MapReduce job that uses User ID as a key to sum up the total number of movies each individual has watched, and output it to a file. This will be used in Part B. I created this job first because it was the simplest to code, and would allow me to hammer out any unexpected issues / bugs / configuration differences that will inevitably come up with working on the IE DIC cluster. Afterwards, I will work through Part A until completion.

Part B MapReduce Job #1: Calculate the number of movies watched by each individual user

Mapper Input: Movielens Dataset of Users in User ID, Movie ID format.

Output key: User

Output value: 1

Output example: User A 1

Mapper #1 Part B Code:

IndividualCountMapper.py

C: > Users > Eddie > Documents > School > CUHK > 2021 - 2022 Term 2 > IEMS 5730 - Big Data Systems and Information Processing > Week 4 > Assignment 1 > IndividualCountMapper.p

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

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
# split the line into words
    words = line.split(",")
# we only take the first word, the user for this MapReduce job
    user = words[0]
# Print "User, 1"
    print '%s\t%s' % (user, 1)
```

I largely use the code in slides 7-13 of [8], but needed to make some modifications. Firstly, instead of reading every word, I only needed the first number which corresponds to the user. I specify in my split command to use comma as a delimiter so it knows when to split the user from the movie ID, and print User and 1, separated by tab.

```
[s1155160788@dicvmd10 input]$ cat movielens_small.csv | IndividualCountMapper.py > output
-bash: IndividualCountMapper.py: command not found
```

```
[s1155160788@dicvmd10 input]$ type python
python is hashed (/usr/bin/python)
```

When trying to test the code, I was getting an error that the command could not be found, and tried exporting the path of /usr/bin/python into my bash shell in several different ways until I realized that I needed to put “python” before the name of my Python script when invoking it.

```
225	1	1
226	1	1
227	1	1
228	1	1
229	1	1
230	1	1
231	1	1
232	1	1
233	2	1
234	2	1
235	2	1
236	2	1
237	2	1
238	2	1
239	2	1
```

Through local testing, I confirm the behavior of the my Mapper code is working correctly by comparing the small dataset CSV file and confirming that the first number (the user) switches from 1 to 2 at the same line as in the small dataset, line 233, and it does.

Reducer Input: Series of “User A, 1” “User B, 1” text lines.

Reducer Key: User ID

Reducer Value: Sum of numbers for each user.

Reducer output: User ID, Sum of number of movies watched for each user.

Output example:

User A, 3

User B, 3

Reducer #1 Part B Code:

IndividualCountReducer.py 2

C: > Users > Eddie > Documents > School > CUHK > 2021 - 2022 Term 2 > IEMS 5730 - Big Data Systems and Information Processing > Week 4 > Assignment 1 > IndividualCountReducer.py > ...

```python
#!/usr/bin/python
# Code mostly adapted from IERG 4300 Fall Tutorial 4 slides 17-21

from operator import itemgetter
import sys

# Changing word to user to be more accurate.

current_user = None
current_count = 0
user = None

# input comes from STDIN

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

    # Parse the input we got from mapper
    user, count = line.split('\t', 1)

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

    # This If switch works because Hadoop sorts mapout by Key
    # before it is passed to the reducer

    if current_user == user:
        current_count += count
    else:
        if current_user:
            # Write result to STDOUT
            print '%s\t%s' % (current_user, current_count)
```

```python
            print '%s\t%s' % (current_user, current_count)
        current_count = count
        current_user = user
if current_user == user:
    print '%s\t%s' % (current_user, current_count)
```

I adapted the code from [8] slides 17-21 but change word to user and current_word to current_user to more accurately reflect the purpose of this particular reducer.

```
[s1155160788@dicvmd10 input]$ cat output | python IndividualCountReducer.py > output2
```

output2

C: > Users > Eddie >

1	1	232
2	2	29
3	3	39
4	4	216
5	5	44
6	6	314
7	7	152
8	8	47
9	9	46
10	10	140
11	11	64
12	12	32
13	13	31
14	14	48
15	15	135
16	16	98
17	17	105
18	18	502
19	19	703
20	20	242
21	21	443
22	22	119
23	23	121
24	24	110
25	25	26
26	26	21
27	27	135
28	28	570
29	29	81
30	30	34

Through local testing, the reducer code appears to properly take the output from the simulated Mapper job and output reasonable looking sums for the number of movies from each user.

The range 1-610 for users matched up, but I wanted to test an exact number to be sure. I tested User 610, which my reducer said watched 1302 movies. I went into the small dataset to check the line range for user 610 and got lines 99535 – 100836. (100836 – 99535 + 1) = 1302, so the reducer is right (you always add one when counting a total through difference like this).

```
[s1155160788@dicvmd10 input]$ hadoop dfs -copyFromLocal /home/s1155160788/input/movielens_small.csv
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

[s1155160788@dicvmd10 input]$ hadoop dfs -copyFromLocal /home/s1155160788/input/movielens_large_updated.csv
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

[s1155160788@dicvmd10 input]$ hadoop dfs -ls
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Found 2 items
-rw-r--r--   3 s1155160788 student   17103777 2022-02-09 16:28 movielens_large_updated.csv
-rw-r--r--   3 s1155160788 student     893989 2022-02-09 16:28 movielens_small.csv
[s1155160788@dicvmd10 input]$
```

Using [9], I refresh my memory on how to copy the datasets to the HDFS in preparation for executing the first MapReduce job.

I navigate through the directory of IE DIC to /usr/hdp to determine that the Hadoop version is 2.6.5.0-292. This helps me to specify the location of the streaming jar. First I navigate to

/usr/hdp/2.6.5.0-292/hadoop/bin before attempting to execute the command.

Final command for MapReduce Job #1:

“hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar\

-file /home/s1155160788/input/IndividualCountMapper.py’ -mapper IndividualCountMapper.py\

-file /home/s1155160788/input/IndividualCountReducer.py’ -reducer IndividualCountReducer.py\

-input movielens_small.csv\

-output IndividualCountSmall”

```
22/02/09 16:53:03 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/home/s1155160788/input/IndividualCountMapper.py, /home/s1155160788/input/IndividualCountReducer.py] [/usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-29
2.jar] /tmp/streamjob3106698696297126158.jar tmpDir=null
22/02/09 16:53:04 INFO client.AHSProxy: Connecting to Application History server at dicvmd2.ie.cuhk.edu.hk/172.16.5.202:10200
22/02/09 16:53:04 INFO client.AHSProxy: Connecting to Application History server at dicvmd2.ie.cuhk.edu.hk/172.16.5.202:10200
22/02/09 16:53:04 INFO client.RequestHedgingRMFailoverProxyProvider: Looking for the active RM in [rm1, rm2]...
22/02/09 16:53:04 INFO client.RequestHedgingRMFailoverProxyProvider: Found active RM [rm1]
22/02/09 16:53:04 INFO mapred.FileInputFormat: Total input paths to process : 1
22/02/09 16:53:05 INFO mapreduce.JobSubmitter: number of splits:2
22/02/09 16:53:05 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1642685809406_0056
22/02/09 16:53:05 INFO impl.YarnClientImpl: Submitted application application_1642685809406_0056
22/02/09 16:53:05 INFO mapreduce.Job: The url to track the job: http://dicvmd3.ie.cuhk.edu.hk:8088/proxy/application_1642685809406_0056/
22/02/09 16:53:05 INFO mapreduce.Job: Running job: job_1642685809406_0056
22/02/09 16:53:09 INFO mapreduce.Job: Job job_1642685809406_0056 running in uber mode : false
22/02/09 16:53:09 INFO mapreduce.Job:  map 0% reduce 0%
22/02/09 16:53:12 INFO mapreduce.Job: Task Id : attempt_1642685809406_0056_m_000000_0, Status : FAILED
Error: java.lang.RuntimeException: Error in configuring object
        at org.apache.hadoop.util.ReflectionUtils.setJobConf(ReflectionUtils.java:112)
        at org.apache.hadoop.util.ReflectionUtils.setConf(ReflectionUtils.java:78)
        at org.apache.hadoop.util.ReflectionUtils.newInstance(ReflectionUtils.java:136)
```

When running the command, I initially got an “Error in configuring object” error, and much error text after that.

```
[s1155160788@dicvmd10 bin]$ hdfs dfs -ls
Found 4 items
-rw-r--r--   3 s1155160788 student        366 2022-02-09 17:09 IndividualCountMapper.py
-rw-r--r--   3 s1155160788 student       1086 2022-02-09 17:09 IndividualCountReducer.py
-rw-r--r--   3 s1155160788 student   17103777 2022-02-09 16:28 movielens_large_updated.csv
-rw-r--r--   3 s1155160788 student     893989 2022-02-09 16:28 movielens_small.csv
```

I copied the python files to the HDFS and simplified the command in the hopes that would eliminate the errors, but I still received the same errors.

“hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar\

-mapper IndividualCountMapper.py\

-reducer IndividualCountReducer.py\

-input movielens_small.csv\

-output IndividualCountSmall”

It seems that you need to use -file even if the python scripts are copied into the HDFS. I added the -files switches back to the command and tried changing the shebang at the top of both scripts to #!/usr/bin/env python (was originally #!/usr/bin/python) and got errors, but different ones this time.

“hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar\

-file IndividualCountMapper.py -mapper IndividualCountMapper.py\

-file IndividualCountReducer.py -reducer IndividualCountReducer.py\

-input movielens_small.csv\

-output IndividualCountSmall”

```
22/02/09 17:37:42 INFO mapreduce.Job: Task Id : attempt_1642685809406_0060_m_000000_0, Status : FAILED
Error: java.lang.RuntimeException: PipeMapRed.waitOutputThreads(): subprocess failed with code 127
        at org.apache.hadoop.streaming.PipeMapRed.waitOutputThreads(PipeMapRed.java:322)
        at org.apache.hadoop.streaming.PipeMapRed.mapRedFinished(PipeMapRed.java:535)
        at org.apache.hadoop.streaming.PipeMapper.close(PipeMapper.java:130)
        at org.apache.hadoop.mapred.MapRunner.run(MapRunner.java:61)
        at org.apache.hadoop.streaming.PipeMapRunner.run(PipeMapRunner.java:34)
        at org.apache.hadoop.mapred.MapTask.runOldMapper(MapTask.java:453)
        at org.apache.hadoop.mapred.MapTask.run(MapTask.java:343)
        at org.apache.hadoop.mapred.YarnChild$2.run(YarnChild.java:170)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:422)
        at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1869)
        at org.apache.hadoop.mapred.YarnChild.main(YarnChild.java:164)
```

The error I received was “Error: java.lang.RuntimeException: PipeMapRed.waitOutputThreads(): subprocess failed with code 127”. I actually received this error in Assignment 0 before and it was related to the Shebangs at first, so I reverted it back to “#!/usr/bin/python” on both files. This gave me the original error, “Error in configuring object”.

```
[s1155160788@dicvmd10 input]$ cat movielens_small.csv | ./IndividualCountMapper.py
-bash: ./IndividualCountMapper.py: /usr/bin/python^M: bad interpreter: No such file or directory
```

During local testing, I had an issue where I had to put “python” before the name of the script, and had a feeling that this was the reason I was getting this error, because the Hadoop MapReduce program didn’t know what to use to interpret the script. This is supposed to have been taken care of by the “#!/user/bin/python” line at the beginning of the script, but this wasn’t working. I went back to local testing and tried “./” before invoking the name of the script and got the error in the screenshot above: “/usr/bin/python^M: bad interpreter: No such file or directory.” After looking up this error, I discovered in [10] that ^M is a carriage return character that was invisibly introduced to the scripts when transferring my python files which originated on a

windows computer to the linux IE DIC cluster. I used the commands suggested in [10] to fix the files in VIM (:e +ff=d0s, :set ff=unix, :wq), then recopied them to the HDFS. Finally, the MapReduce job ran as expected.

```
[s1155160788@dicvmd10 input]$ hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-strea
ng.jar -file IndividualCountMapper.py -mapper IndividualCountMapper.py -file IndividualCoun
educer.py -reducer IndividualCountReducer.py -input movielens_small.csv -output IndividualC
ntSmall
22/02/09 22:04:51 WARN streaming.StreamJob: -file option is deprecated, please use generic
tion -files instead.
packageJobJar: [IndividualCountMapper.py, IndividualCountReducer.py] [/usr/hdp/2.6.5.0-292/
doop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar] /tmp/streamjob8084368588733409839.ja
tmpDir=null
22/02/09 22:04:52 INFO client.AHSProxy: Connecting to Application History server at dicvmd2
e.cuhk.edu.hk/172.16.5.202:10200
22/02/09 22:04:52 INFO client.AHSProxy: Connecting to Application History server at dicvmd2
e.cuhk.edu.hk/172.16.5.202:10200
22/02/09 22:04:52 INFO client.RequestHedgingRMFailoverProxyProvider: Looking for the active
M in [rm1, rm2]...
22/02/09 22:04:52 INFO client.RequestHedgingRMFailoverProxyProvider: Found active RM [rm1]
22/02/09 22:04:52 INFO mapred.FileInputFormat: Total input paths to process : 1
22/02/09 22:04:52 INFO mapreduce.JobSubmitter: number of splits:2
22/02/09 22:04:52 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1642685809406
063
22/02/09 22:04:52 INFO impl.YarnClientImpl: Submitted application application_1642685809406
063
22/02/09 22:04:52 INFO mapreduce.Job: The url to track the job: http://dicvmd3.ie.cuhk.edu.
:8088/proxy/application_1642685809406_0063/
22/02/09 22:04:52 INFO mapreduce.Job: Running job: job_1642685809406_0063
22/02/09 22:04:56 INFO mapreduce.Job: Job job_1642685809406_0063 running in uber mode : fal
22/02/09 22:04:56 INFO mapreduce.Job:  map 0% reduce 0%
22/02/09 22:05:00 INFO mapreduce.Job:  map 50% reduce 0%
22/02/09 22:05:03 INFO mapreduce.Job:  map 100% reduce 0%
22/02/09 22:05:05 INFO mapreduce.Job:  map 100% reduce 100%
22/02/09 22:05:05 INFO mapreduce.Job: Job job_1642685809406_0063 completed successfully
22/02/09 22:05:05 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=790275
                FILE: Number of bytes written=2046455
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1025307
                HDFS: Number of bytes written=4422
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        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)=182758
                Total time spent by all reduces in occupied slots (ms)=179308
```

```
        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)=182758
                Total time spent by all reduces in occupied slots (ms)=179308
                Total time spent by all map tasks (ms)=3973
                Total time spent by all reduce tasks (ms)=1949
                Total vcore-milliseconds taken by all map tasks=3973
                Total vcore-milliseconds taken by all reduce tasks=1949
                Total megabyte-milliseconds taken by all map tasks=187144192
                Total megabyte-milliseconds taken by all reduce tasks=183611392
        Map-Reduce Framework
                Map input records=100836
                Map output records=100836
                Map output bytes=588597
                Map output materialized bytes=790281
                Input split bytes=246
                Combine input records=0
                Combine output records=0
                Reduce input groups=610
                Reduce shuffle bytes=790281
                Reduce input records=100836
                Reduce output records=610
                Spilled Records=201672
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=158
                CPU time spent (ms)=4490
                Physical memory (bytes) snapshot=5553786880
                Virtual memory (bytes) snapshot=166730346496
                Total committed heap usage (bytes)=6167724032
        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=1025061
        File Output Format Counters
                Bytes Written=4422
22/02/09 22:05:05 INFO streaming.StreamJob: Output directory: IndividualCountSmall
```

```
[s1155160788@dicvmd10 input]$ hdfs dfs -ls
Found 6 items
drwx------   - s1155160788 student          0 2022-02-09 22:05 .staging
-rw-r--r--   3 s1155160788 student        354 2022-02-09 22:03 IndividualCountMapper.py
-rw-r--r--   3 s1155160788 student       1047 2022-02-09 22:03 IndividualCountReducer.py
drwxr-xr-x   - s1155160788 student          0 2022-02-09 22:05 IndividualCountSmall
-rw-r--r--   3 s1155160788 student   17103777 2022-02-09 16:28 movielens_large_updated.csv
-rw-r--r--   3 s1155160788 student     893989 2022-02-09 16:28 movielens_small.csv
```

I confirmed that the MapReduce job finished successfully and it created a directory called IndividualCountSmall in the hdfs. I used the -copyToLocal switch in the hdfs dfs command to move the IndividualCountSmall directory into an output directory.

The result is two files, one called "_SUCCESS" that has nothing, and one called part-00000 that has the data we are interested in. I rename the part file and delete the _SUCCESS file and IndividualCountSmall directory, leaving only the output file, called "IndividualCountSmall"

C: > Users > Eddie >

574	66	345
575	67	36
576	68	1260
577	69	46
578	7	152
579	70	62
580	71	35
581	72	45
582	73	210
583	74	177
584	75	69
585	76	119
586	77	29
587	78	61
588	79	64
589	8	47
590	80	167
591	81	26
592	82	227
593	83	118
594	84	293
595	85	34
596	86	70
597	87	21
598	88	56
599	89	518
600	9	46
601	90	54
602	91	575
603	92	24
604	93	97
605	94	56
606	95	168
607	96	78
608	97	36
609	98	92
610	99	53

Inspecting the output of the MapReduce Job #1 on the small dataset, I see that there are 610 lines (the leftmost number, created by Visual Studio Code), and the totals seem reasonable. I cross check 610 and the total is still 1302. However to my surprise, 610 is on line 569 and the user on line 610 is 99. It seems that MapReduce isn't sorting the keys in a strictly numerical sense, but sorting the keys based off the first character of the key, then for items that share the same first character, by the second character, and so on.

I run the MapReduce job on the large dataset in the same way, only changing the name of the output and input files. Interestingly, when I try to run the command from my output directory, it says there is no mapper and reducer in the output directory, suggesting that putting the mapper and reducer is useless because it is using the file switch to take files from the working directory I execute the command on. Not that I couldn't use files from the HDFS, but I would probably need to explicitly reference the path of the HDFS rather than just using the filename alone with no path.

"hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar\

-file IndividualCountMapper.py -mapper IndividualCountMapper.py\

-file IndividualCountReducer.py -reducer IndividualCountReducer.py\

-input movielens_large_updated.csv\

-output IndividualCountLarge"

```
[s1155160788@dicvmd10 input]$ hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar -file IndividualCountMapper.py -mapper IndividualCountMapper.py -file IndividualCountRedu
cer.py -reducer IndividualCountReducer.py -input movielens_large_updated.csv -output IndividualCountLarge
22/02/10 10:50:25 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [IndividualCountMapper.py, IndividualCountReducer.py] [/usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar] /tmp/streamjob4351341150454523156.jar tmp
Dir=null
22/02/10 10:50:25 INFO client.AHSProxy: Connecting to Application History server at dicvmd2.ie.cuhk.edu.hk/172.16.5.202:10200
22/02/10 10:50:26 INFO client.AHSProxy: Connecting to Application History server at dicvmd2.ie.cuhk.edu.hk/172.16.5.202:10200
22/02/10 10:50:26 INFO client.RequestHedgingRMFailoverProxyProvider: Looking for the active RM in [rm1, rm2]...
22/02/10 10:50:26 INFO client.RequestHedgingRMFailoverProxyProvider: Found active RM [rm1]
22/02/10 10:50:26 INFO mapred.FileInputFormat: Total input paths to process : 1
22/02/10 10:50:26 INFO mapreduce.JobSubmitter: number of splits:2
22/02/10 10:50:26 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1642685809406_0072
22/02/10 10:50:26 INFO impl.YarnClientImpl: Submitted application application_1642685809406_0072
22/02/10 10:50:26 INFO mapreduce.Job: The url to track the job: http://dicvmd3.ie.cuhk.edu.hk:8088/proxy/application_1642685809406_0072/
22/02/10 10:50:26 INFO mapreduce.Job: Running job: job_1642685809406_0072
22/02/10 10:50:31 INFO mapreduce.Job: Job job_1642685809406_0072 running in uber mode : false
22/02/10 10:50:31 INFO mapreduce.Job:  map 0% reduce 0%
22/02/10 10:50:36 INFO mapreduce.Job:  map 50% reduce 0%
22/02/10 10:50:40 INFO mapreduce.Job:  map 100% reduce 0%
22/02/10 10:50:45 INFO mapreduce.Job:  map 100% reduce 100%
22/02/10 10:50:45 INFO mapreduce.Job: Job job_1642685809406_0072 completed successfully
22/02/10 10:50:45 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=15466224
                FILE: Number of bytes written=31398377
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=17235111
                HDFS: Number of bytes written=444374
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        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)=322874
                Total time spent by all reduces in occupied slots (ms)=378580
                Total time spent by all map tasks (ms)=7019
                Total time spent by all reduce tasks (ms)=4115
                Total vcore-milliseconds taken by all map tasks=7019
                Total vcore-milliseconds taken by all reduce tasks=4115
                Total megabyte-milliseconds taken by all map tasks=330622976
                Total megabyte-milliseconds taken by all reduce tasks=387665920
        Map-Reduce Framework
                Map input records=1569491
                Map output records=1569491
                Map output bytes=12327236
                Map output materialized bytes=15466230
```

```
        Map-Reduce Framework
                Map input records=1569491
                Map output records=1569491
                Map output bytes=12327236
                Map output materialized bytes=15466230
                Input split bytes=262
                Combine input records=0
                Combine output records=0
                Reduce input groups=53184
                Reduce shuffle bytes=15466230
                Reduce input records=1569491
                Reduce output records=53184
                Spilled Records=3138982
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=168
                CPU time spent (ms)=8280
                Physical memory (bytes) snapshot=5693517824
                Virtual memory (bytes) snapshot=166746636288
                Total committed heap usage (bytes)=6166151168
        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=17234849
        File Output Format Counters
                Bytes Written=444374
22/02/10 10:50:45 INFO streaming.StreamJob: Output directory: IndividualCountLarge
[s1155160788@dicvmd10 input]$
```

IndividualCountLarge

C: > Users > Eddie > Docum

53152	9956	3
53153	9957	72
53154	9959	12
53155	996	2
53156	9960	57
53157	9961	2
53158	9962	6
53159	9963	10
53160	9964	22
53161	9965	17
53162	9967	2
53163	9970	34
53164	9971	4
53165	9972	85
53166	9973	31
53167	9974	1
53168	9975	64
53169	9978	9
53170	9979	81
53171	998	7
53172	9980	101
53173	9982	3
53174	9985	15
53175	9986	18
53176	9988	230
53177	9989	6
53178	9991	2
53179	9992	1
53180	9993	19
53181	9994	3
53182	9996	128
53183	9997	1
53184	9999	1
53185		

The input is as expected in terms of numbers, but has 53184 lines instead of the 69,878 users mentioned in the Assignment 1 PDF. I chalk this up to several reasons. Firstly, users that haven't watched any movies wouldn't show up in the raw dataset, so they wouldn't be counted. Secondly, the number of users might be reduced in the updated dataset for time / performance concerns compared to the original large dataset that was in mind when the PDF for the assignment was created. By examining the dataset, I see that the highest user with a movie count is actually 71,567, so it didn't just stop processing users after 53,184. Likely what happened is when the PDF / assignment was created, the dataset had 69,878 users, and expanded to 71,567 by the most recent iteration of the dataset was taken for our use.

As a quick test of accuracy, I navigate to user 610 which is noted in my MapReduce output to have 6 users. I then go to the large dataset manually and count that 610 has 6 entries, so it matches up.

With this, I consider the first MapReduce job of Part B complete.

Part A MapReduce Job #1: Output the users that have watched each movie

I detail this MapReduce job near the beginning of this document, but am pasting that information again for convenience.

Mapper #1:

Input: MovieLens Dataset in format User ID, Movie ID

Output key: Movie ID

Output value: User ID

Example with users A and B in Assignment 1 PDF:

Movie A User A

Movie B User A

Movie C User A

Movie B User B

Movie C User B

Movie D User B

Part A Mapper #1 code:

MovieArrayMapper.py 1

C: > Users > Eddie > Documents > School > CUHK > 2021 - 2022 Term 2 > IEMS 5730 - Big Data Syste

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

import sys

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

    # split the line into words
    words = line.split(",")

    # we create two variables here, user and movies.
    # The input is in the format User, Movie

    user = words[0]
    movie = words[1]

    # For this mapper, we print the movie, a tab, and then the user.
    print '%s\t%s' % (movie, user)
```

I can use almost the same code as the first MapReduce job for Part B, since the input is identical. The only difference is I create two variables, movie and user, instead of just user, and I print movie as the key, a tab, then user as opposed to printing user as the key, tab, and then a 1. This mapper is called “MovieArrayMapper.py”.

```
141400  610
141408  610
141799  610
142366  610
142420  610
142488  610
142598  610
143385  610
143859  610
144976  610
145951  610
146309  610
147657  610
147662  610
148166  610
148424  610
148626  610
149011  610
149406  610
150401  610
152077  610
152081  610
152372  610
155064  610
156371  610
156726  610
157296  610
158238  610
158721  610
158872  610
158956  610
159093  610
160080  610
160341  610
160527  610
160571  610
160836  610
161582  610
161634  610
162350  610
163937  610
163981  610
164179  610
166528  610
166534  610
168248  610
168250  610
168252  610
170875  610
```

With local testing, the output is as expected where the ending second number is a bunch of 610's, because that corresponds to the 610th and final user in the small dataset, and they watched a lot of movies. I also manually checked the small dataset and indeed, the final line is 610,170875 with a number of 6 digit movie numbers above it. I compared the output to the small dataset in several points such as the beginning and it matched up perfectly. Considering the success of local testing, I performed the commands in [10] and confirmed I could use the script without needing to put Python in front, before moving onto the Reducer.

Reducer #1:

Input: Shown above

Output key: Movie ID.

Output value: Array of User ID's that have watched each movie.

Example with full data in Assignment 1 PDF sample table:

Movie A [a]

Movie B [a b c d]

Movie C [a b d]

Movie D [b c]

Part A Initial Reducer #1 Code:

MovieArrayReducer.py 4 × movielens_small.csv

- 2022 Term 2 > IEMS 5730 - Big Data Systems and Information Processing > Week 4 > Assignment 1 > MovieA

```python
#!/usr/bin/python
from operator import itemgetter
import sys

# input comes from STDIN (standard input)

# Here, I mirror the functionality from the previous reducer, where
# movie is the new user and user is the new count. The difference is
# that user is not a number, so we don't need to do conversion from string
# to integer.

# For current users, we create an empty array to hold all the users for a
# particular movie for printing later.

# I print with space because of restrictions in Python 2 with printing.

current_movie = None
current_users = []
movie = None

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

    # Parse the input we got from mapper
    # This will put the movie of this line in movie and the user in user.

    movie, user = line.split('\t', 1)

    # This is the equivalent of the count based code, but we
    # are appending user to an array.
    if current_movie == movie:
        current_users.append(user)

    else:
        if current_movie:
```

```
    else:
        if current_movie:
            # Write result to STDOUT when movie moves to the next key
            # First we print the user and a tab, before using a for loop
            # to iterate over the list and print the list one by one.

            # prints movie without newline, separated by space
            print '%s' % (current_movie),

            #print the list of current users, separated by spaces
            for X in current_users:
                print X,
            print

            # clear the current users list for the next key.
            del current_users[:]

        # add the first user of the next key.
        current_users.append(user)

        # set current_mmovie equal to the next key.
        current_movie = movie
# Print the last key if needed.
if current_movie == movie:

    print '%s' % (current_movie),
    for X in current_users:
        print X,
    print
```

My first viable reducer code is above. The comments explain most of what I put here, but I will make some additions.

Instead of working with a number that was being summed up, I needed to create an array that was appended to. Because I needed an empty list (so it could be appended to), instead of none, I used [11] to figure out how to create an empty list and append items. I initially wanted to clear the list after each key + value array was printed using .clear() but realized that wouldn't work for Python 2, so [12] taught me how to empty the list in Python 2.

I originally tried using the "from __future__ import print_function" to switch to Python 3's print functions, and coded my print statements with this in mind. I needed to print the string corresponding to the movie key followed by the list in the same line without a new line and Python 3 had end and sep commands that would allow me to do this while separating the items by tab. Python 2 had ways to print on the same line, but it would separate by space.

When originally testing my Python 3 code locally, the Reducer code output nothing so I thought it was because the from future Python 3 print functions weren't being recognized on the IE DIC cluster, so I rewrote all the print functions to Python 2 print statements using [13], and the code still output nothing.

I later learned that the Reducer output nothing due to a coding mistake that altered the logic. I had indented lines 53 and 56 one too many times, which made them as part of the "if current_movie:" statement instead as part of the "else". I initialize the current_movie and current_users as empty, so if it is part of the "if current_movie:" statement, then the lines to set the current_movie and start the current_users list never execute, meaning these variables remain NULL / None, which means the print statements never execute either.

After fixing this, the code would output text, but for many of the movies, it would output the same user multiple times. This was because in the printing for loops to print out the array of users, I put "user" the variable instead of something else like X, so it printed whatever currently was in user the same amount of times.

After changing this, the output had movies and users space separated, with valid users for each movie, but sometimes a given movie would show up more than once with a different subset of the total users. I was testing locally by using the following method, as suggested in [9]: cat data | mapper | sort | reducer

I decided to try cat data | mapper | sort into an output file and examined it, only to see that it would occasionally have movies listed more than once. I think this is some issue with the sorter in Unix, because I ran the MapReduce job using Hadoop and the output was correct.

Using [14], I learned how to configure Visual Studio Code to not use carriage return and just use line feed so that I could send files to and from my laptop and IE DIC without needing to use 3 commands to in VIM to fix the file every time I made changes.

Reducer Command:

"hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar\

-file MovieArrayMapper.py -mapper MovieArrayMapper.py\

-file MovieArrayReducer.py -reducer MovieArrayReducer.py\

-input movielens_small.csv\

-output MovieUsersSmall"

```
95583 448 249
956 474
95624 68
95633 89
95654 177 105 610
95690 125
95717 599
95720 448 534 365 483 222 318 249 111 292
95738 21
95744 448
95761 125
95771 599
95780 599
95796 382
95839 599
95843 89
95858 105 63 610 380 509
95873 599 610 249 318
95875 68 249 380 448 561 483 534 610
959 474
95939 483
95949 222
95965 599
96 474
96004 599 21
96007 380
96020 318 105 233
96079 261 50 52 119 212 73 292 68 279 139 249 62 63 209 205 246 122 305 256
96084 318
961 606
96110 564 365 419 534 448 509 89 111 68
96114 448
96121 292
96150 525
96281 596 448
96283 483
963 288
```

I pulled up the small dataset side by side and verified for a random movie that every user listed was valid, and the only users for that movie, so the output is accurate and everything is included. There is also no double listing of movies, so it seems MapReduce sorts with a superior method compared to what is built into the bash shell.

While this output seems good, there is no sorting on the users. I need to implement some secondary sort in order to make sure it is sorted in numerical order. The reason this is necessary is because in the second MapReduce job, I discard the movie and the output / intermediate key from the mapper is pairs of users, which need to be ordered from least to greatest to work properly.

Final Reducer #1 Code for Part A:

MovieArrayReducer.py 4 ✕ part-00000

C: > Users > Eddie > Documents > School > CUHK > 2021 - 2022 Term 2 > IEMS 5730 - Big Data Systems and

```python
#!/usr/bin/python
from operator import itemgetter
import sys

# input comes from STDIN (standard input)

# Here, I mirror the functionality from the previous reducer, where
# movie is the new user and user is the new count. The difference is
# that user is not a number, so we don't need to do conversion from string
# to integer.

# For current users, we create an empty array to hold all the users for a
# particular movie for printing later.

# I print with space because of restrictions in Python 2 with printing.

current_movie = None
current_users = []
movie = None

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

    # Parse the input we got from mapper
    # This will put the movie of this line in movie and the user in user.

    movie, user = line.split('\t', 1)

    # Integer conversion code readded to convert user into integer
    # for sorting.

    try:
        user = int(user)
    except ValueError:
        # User was not a number, so silently ignore/discard the line
        continue
```

```python

    # This is the equivalent of the count based code, but we
    # are appending user to an array.
    if current_movie == movie:
        current_users.append(user)

    else:
        if current_movie:
            # Write result to STDOUT when movie moves to the next key
            # First we print the user and a tab, before using a for loop
            # to iterate over the list and print the list one by one.

            # sort the array of users numerically in ascending order.

            current_users.sort()

            # prints movie without newline, separated by space
            print '%s' % (current_movie),

            #print the list of current users, separated by spaces
            for X in current_users:
                print X,
            print

            # clear the current users list for the next key.
            del current_users[:]

        # add the first user of the next key.
        current_users.append(user)

        # set current_mmovie equal to the next key.
        current_movie = movie
# Print the last key if needed.
if current_movie == movie:
    # sort the array of users numerically in ascending order.
    current_users.sort()
```

```
# Print the last key if needed.
if current_movie == movie:
    # sort the array of users numerically in ascending order.
    current_users.sort()

    # Print the line
    print '%s' % (current_movie),
    for X in current_users:
        print X,
    print
```

I re-add the code used in the IndividualCountReducer to convert count from a string to an integer, and apply it to the user variable instead. We do not need to guarantee the movie key is sorted by number perfectly because we are discarding the movie key in the next MapReduce job, user is the variable of interest to guarantee perfect numerical sorting.

I also sort the current_users array before printing using the built in Python method for lists detailed in [15].

Using the same reducer command as before:

“hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar\

-file MovieArrayMapper.py -mapper MovieArrayMapper.py\

-file MovieArrayReducer.py -reducer MovieArrayReducer.py\

-input movielens_small.csv\

-output MovieUsersSmall”

```
[s1155160788@dicvmd10 input]$ hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streami
ng.jar -file MovieArrayMapper.py -mapper MovieArrayMapper.py -file MovieArrayReducer.py -redu
cer MovieArrayReducer.py -input movielens_small.csv -output MovieUsersSmall
22/02/11 15:45:02 WARN streaming.StreamJob: -file option is deprecated, please use generic op
tion -files instead.
packageJobJar: [MovieArrayMapper.py, MovieArrayReducer.py] [/usr/hdp/2.6.5.0-292/hadoop-mapre
duce/hadoop-streaming-2.7.3.2.6.5.0-292.jar] /tmp/streamjob6227632849327054724.jar tmpDir=nul
l
22/02/11 15:45:02 INFO client.AHSProxy: Connecting to Application History server at dicvmd2.i
e.cuhk.edu.hk/172.16.5.202:10200
22/02/11 15:45:02 INFO client.AHSProxy: Connecting to Application History server at dicvmd2.i
e.cuhk.edu.hk/172.16.5.202:10200
22/02/11 15:45:02 INFO client.RequestHedgingRMFailoverProxyProvider: Looking for the active R
M in [rm1, rm2]...
22/02/11 15:45:02 INFO client.RequestHedgingRMFailoverProxyProvider: Found active RM [rm1]
22/02/11 15:45:03 INFO mapred.FileInputFormat: Total input paths to process : 1
22/02/11 15:45:03 INFO mapreduce.JobSubmitter: number of splits:2
22/02/11 15:45:03 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1642685809406_0
097
22/02/11 15:45:03 INFO impl.YarnClientImpl: Submitted application application_1642685809406_0
097
22/02/11 15:45:03 INFO mapreduce.Job: The url to track the job: http://dicvmd3.ie.cuhk.edu.hk
:8088/proxy/application_1642685809406_0097/
22/02/11 15:45:03 INFO mapreduce.Job: Running job: job_1642685809406_0097
22/02/11 15:45:07 INFO mapreduce.Job: Job job_1642685809406_0097 running in uber mode : false
22/02/11 15:45:07 INFO mapreduce.Job:  map 0% reduce 0%
22/02/11 15:45:11 INFO mapreduce.Job:  map 50% reduce 0%
22/02/11 15:45:14 INFO mapreduce.Job:  map 100% reduce 0%
22/02/11 15:45:16 INFO mapreduce.Job:  map 100% reduce 100%
22/02/11 15:45:16 INFO mapreduce.Job: Job job_1642685809406_0097 completed successfully
22/02/11 15:45:16 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=1095667
                FILE: Number of bytes written=2657074
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1025307
                HDFS: Number of bytes written=450401
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        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)=179768
                Total time spent by all reduces in occupied slots (ms)=175536
                Total time spent by all map tasks (ms)=3908
                Total time spent by all reduce tasks (ms)=1908
```

```
        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)=179768
                Total time spent by all reduces in occupied slots (ms)=175536
                Total time spent by all map tasks (ms)=3908
                Total time spent by all reduce tasks (ms)=1908
                Total vcore-milliseconds taken by all map tasks=3908
                Total vcore-milliseconds taken by all reduce tasks=1908
                Total megabyte-milliseconds taken by all map tasks=184082432
                Total megabyte-milliseconds taken by all reduce tasks=179748864
        Map-Reduce Framework
                Map input records=100836
                Map output records=100836
                Map output bytes=893989
                Map output materialized bytes=1095673
                Input split bytes=246
                Combine input records=0
                Combine output records=0
                Reduce input groups=9724
                Reduce shuffle bytes=1095673
                Reduce input records=100836
                Reduce output records=9724
                Spilled Records=201672
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=148
                CPU time spent (ms)=4920
                Physical memory (bytes) snapshot=5586993152
                Virtual memory (bytes) snapshot=166754320384
                Total committed heap usage (bytes)=6172442624
        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=1025061
        File Output Format Counters
                Bytes Written=450401
22/02/11 15:45:16 INFO streaming.StreamJob: Output directory: MovieUsersSmall
[s1155160788@dicvmd10 input]$ hdfs dfs -copyToLocal MovieUsersSmall
[s1155160788@dicvmd10 input]$
```

```
96588 21 47 73 111 125 177 212 298 305 306 365 414 448 483 509 514 525 534 583 596
96606 50 105 439
96608 89
96610 15 18 24 50 62 63 105 139 154 177 203 210 249 256 298 305 318 331 332 352 380 414 495 509 522 525 534 560 573 582 599 610
96616 111 298 448
96655 18 89 599
96691 80 249
96726 448
96728 103 122 249 318 414 599
96737 122 249 279 298 305 365 380 414 448 511 534 561 596 599
968 64 137 186 187 274 288 290 292 305 312 313 380 384 387 409 414 422 474 477 505 555 562 571 590 591 599 603 610
96811 47 52 249 305 317 610
96815 380 610
96821 30 62 73 105 143 153 159 212 247 249 280 298 308 365 414 418 509 525 563 596
96829 41 62 105 123 125 152 317 326 339 497
96832 610
96861 62 119 249 380 448 586 610
969 27 50 57 59 63 103 113 131 137 140 156 186 199 217 224 265 271 288 290 325 345 387 414 465 474 517 524 527 555 572 577 590 603 606
```

We can see a subset of the output from running the MapReduce job on the small dataset here. Although the movie keys are not *perfectly* sorted due to being sorted per character in a string format, this is acceptable, as we can see that all the numbers 2nd and beyond signify the users that have watched a movie, and they are sorted perfectly in ascending order. I verified the accuracy of these results to be identical to the previously correct output, just sorted better.

I now run the command on the Large Dataset to confirm there are no scalability issues.

Large dataset command:

“hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar\

-file MovieArrayMapper.py -mapper MovieArrayMapper.py\

-file MovieArrayReducer.py -reducer MovieArrayReducer.py\

-input movielens_large_updated.csv\

-output MovieUsersLarge”

```
[s1155160788@dicvmd10 input]$ hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar -file MovieArrayMapper.py -mapper MovieArrayMapper.py -file MovieArrayReducer.py -reducer
 MovieArrayReducer.py -input movielens_large_updated.csv -output MovieUsersLarge
22/02/11 16:00:40 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [MovieArrayMapper.py, MovieArrayReducer.py] [/usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar] /tmp/streamjob5068518243406832070.jar tmpDir=null
22/02/11 16:00:41 INFO client.AHSProxy: Connecting to Application History server at dicvmd2.ie.cuhk.edu.hk/172.16.5.202:10200
22/02/11 16:00:41 INFO client.AHSProxy: Connecting to Application History server at dicvmd2.ie.cuhk.edu.hk/172.16.5.202:10200
22/02/11 16:00:41 INFO client.RequestHedgingRMFailoverProxyProvider: Looking for the active RM in [rm1, rm2]...
22/02/11 16:00:41 INFO client.RequestHedgingRMFailoverProxyProvider: Found active RM [rm1]
22/02/11 16:00:41 INFO mapred.FileInputFormat: Total input paths to process : 1
22/02/11 16:00:41 INFO mapreduce.JobSubmitter: number of splits:2
22/02/11 16:00:41 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1642685809406_0099
22/02/11 16:00:41 INFO impl.YarnClientImpl: Submitted application application_1642685809406_0099
22/02/11 16:00:41 INFO mapreduce.Job: The url to track the job: http://dicvmd3.ie.cuhk.edu.hk:8088/proxy/application_1642685809406_0099/
22/02/11 16:00:41 INFO mapreduce.Job: Running job: job_1642685809406_0099
22/02/11 16:00:46 INFO mapreduce.Job: Job job_1642685809406_0099 running in uber mode : false
22/02/11 16:00:46 INFO mapreduce.Job:  map 0% reduce 0%
22/02/11 16:00:52 INFO mapreduce.Job:  map 50% reduce 0%
22/02/11 16:00:56 INFO mapreduce.Job:  map 100% reduce 0%
22/02/11 16:01:02 INFO mapreduce.Job:  map 100% reduce 100%
22/02/11 16:01:02 INFO mapreduce.Job: Job job_1642685809406_0099 completed successfully
22/02/11 16:01:02 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=20242765
                FILE: Number of bytes written=40951294
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=17202903
                HDFS: Number of bytes written=9241840
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Rack-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=326876
                Total time spent by all reduces in occupied slots (ms)=476836
                Total time spent by all map tasks (ms)=7106
                Total time spent by all reduce tasks (ms)=5183
                Total vcore-milliseconds taken by all map tasks=7106
                Total vcore-milliseconds taken by all reduce tasks=5183
                Total megabyte-milliseconds taken by all map tasks=334721024
                Total megabyte-milliseconds taken by all reduce tasks=488280064
        Map-Reduce Framework
                Map input records=1569491
                Map output records=1569491
                Map output bytes=17103777
                Map output materialized bytes=20242771
                Input split bytes=262
```

```
        Map-Reduce Framework
                Map input records=1569491
                Map output records=1569491
                Map output bytes=17103777
                Map output materialized bytes=20242771
                Input split bytes=262
                Combine input records=0
                Combine output records=0
                Reduce input groups=8646
                Reduce shuffle bytes=20242771
                Reduce input records=1569491
                Reduce output records=8646
                Spilled Records=3138982
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=170
                CPU time spent (ms)=8520
                Physical memory (bytes) snapshot=5710082048
                Virtual memory (bytes) snapshot=166751670272
                Total committed heap usage (bytes)=6178734080
        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=17202641
        File Output Format Counters
                Bytes Written=9241840
22/02/11 16:01:02 INFO streaming.StreamJob: Output directory: MovieUsersLarge
[s1155160788@dicvmd10 input]$ hdfs dfs -copyToLocal MovieUsersLarge
[s1155160788@dicvmd10 input]$
```

The code executes successfully without taking too long.

User:	s1155160788
Name:	streamjob5068518243406832070.jar
Application Type:	MAPREDUCE
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Fri Feb 11 16:00:41 +0800 2022
Elapsed:	18sec
Tracking URL:	History
Log Aggregation Status	SUCCEEDED
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	<DEFAULT_PARTITION>

According to the IE DIC web UI, the job takes 18 seconds to execute, and via the console output, we can tell there is 1 reducer and 2 mappers based on text under the JobCounters section.

I was not expecting this to be the test of scalability. The second mapper for Part A will probably be the ultimate test of scalability in this assignment as it has to generate every possible pair of users under each movie, which will scale rapidly.

```
8993 4867 11760 18737 19635 19742 23858 30274 30687 31327 38234 48618 51033 53315 56347
8994 1751 3434 5913 6477 9683 16625 18454 19635 23140 23858 30687 31327 32545 35846 3668
8995 8892 19635 23858 30687 30977 32046 35442 40154 44041 47345 48618 48793 53315 59269
8996 3301 5678 6018 7265 8173 16929 19635 21385 24937 27468 30846 32680 42791 44631 4893
8997 4538 16929 16958 19635 25493 32046 35442 40154 44041 47345 48618 48793 53315 59269
8998 596 2361 3301 3783 3817 4024 4538 4956 5529 5539 5865 5913 5917 6281 6477 6757 6959
8999 3301 4538 5539 5913 5917 6281 6477 6881 7252 8114 8395 8892 9900 10759 10858 12296
90 121 522 585 719 914 1319 2768 3052 3487 5349 5423 5770 5829 7434 7605 8010 8338 8430
9000 1786 5913 8352 8395 8892 10759 10858 11490 12834 17922 19635 19742 19928 20342 2088
9001 3274 4078 4538 6477 9196 9809 16227 17755 19635 30977 32046 35442 40154 40398 46071
9002 820 4538 6616 6757 6946 7534 8706 11760 13048 20887 22842 23176 28288 28665 31854 4
9003 4538 8517 11760 23176 35828 47345 49955 49979 51057 59269 59598 61880 67385
9004 1302 1993 2801 3219 3301 3646 3817 4078 4538 6014 6082 7217 7678 8892 9196 9355 100
9005 78 190 198 543 2801 3793 3817 4517 4538 5238 5480 6014 6018 7612 7678 7711 8734 889
9006 20887 59269 67385
9007 13900 19635 53315 59269 66220 67385
9008 2801 4538 7508 9900 17922 19635 23176 35828 38928 40751 41517 42791 47345 50156 515
9009 36687 38527 48432
901 51 214 274 288 310 351 375 595 642 643 742 824 949 988 1025 1325 1327 1401 1724 1771
9010 493 1461 1485 1559 1841 2785 3560 4198 4712 4834 5018 5040 5277 5511 5565 5913 6067
9011 7838 8683 11032 13614 14895 16681 19635 24774 27685 34683 36666 41517 43356 46199 4
9012 10680 13900 26116 31890 59269 67385
9013 9139 12296 16003 16227 16928 18059 19379 21106 25038 28934 30977 32826 33476 34420
9014 6757 8973 13900 16458 16929 19635 20932 24476 34683 40398 50156 53315 53649 57696 5
9015 212 5939 16727 19635 30158 38738 41956 42791 49582 51033 56707 59269 59598 69664
9016 3817 13606 16929 38928 42791 47046 62239
9017 2471 5241 5917 16727 16929 19635 31254 33359 39799 46296 48784 50068 53742
```

Sample output above. As expected, there are many more users for each movie on average for the large dataset compared to the smaller dataset. The movies are sorted in their mostly correct way, as in the small dataset, while the users are sorted in perfect numerical ascending order. There are 8646 lines here, which seems to indicate there was 8646 movies total. This is close enough to the original 10,000 mentioned in the Assignment 1 PDF given this is an updated dataset which may have reduced numbers for performance reasons. I validated the output by comparing it manually to the LargeDataset in several spots, and it was always accurate and complete.

With this, I consider the first MapReduce job for Part A complete.

Part A MapReduce Job #2: Output the amount of movies each pair of users have both watched

Input: MovieUsersSmall and MovieUsersLarge, the output of the 1st MapReduce jobs for Part A on the small and large datasets respectively.

Format of input: Movie ID, space separated list of all users who have watched that movie, in ascending order.

Output key: User A,User B (a pair of users)

Output value: 1

Full output: User A,User B 1

Example with table from Assignment 1 PDF:

Table Input:

Movie A [a]

Movie B [a b c d]

Movie C [a b d]

Movie D [b c]

Mapper output:

User A,User B 1

User A,User C 1

User A,User D 1

User B,User C 1

User B,User D 1

User C,User D 1

User A,User B 1

User A,User D 1

User B,User D 1

User B,User C 1

Part A Mapper #2 Code:

WatchPairsMapper.py 2

C: > Users > Eddie > Documents > School > CUHK > 2021 - 2022 Term 2 > IEMS 5730 - Big Data Systems and I

```python
#!/usr/bin/python
from operator import itemgetter
from itertools import combinations # used for iterating all possible pairs
import sys

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

    # First I add every element of the line to a list, including the movie.
    # By default, split separates by space, which we want.
    users = line.split()

    # remove the first element, which is the movie.
    users.pop(0)

    # Use the combinations module to list every pair
    # Print every pair followed by 1 on its own line
    # Format: User A,User B (composite key) 1

    for pair in combinations(users, 2):
        user = ','.join(pair)
        print user,
        print '%s' % (1)
```

The comments explain the code fairly well. I confirmed with [16] that the .split method returns a list, which allows me to use .pop(0) to remove the first element, which is the movie, leaving only the users. I learned in [17] that I can import combinations from itertools to go through a list and generate each unique pair of elements in lexicographic order, which is exactly what I needed. As I ensured in the previous MapReduce job that the pairs were sorted in ascending numerical order, no pairs are double counted. I referred to [18] for syntax, and [19] for efficiently printing the elements returned by Itertools combinations one by one.

Originally, the combinations module outputted each pair like this ('103', '550') for the pair of 103 and 550, but I needed it in the format 103 550, for a final output of 103 550 1, where the 1 is the number of movies the pair of users have both watched (in the Mapper stage).

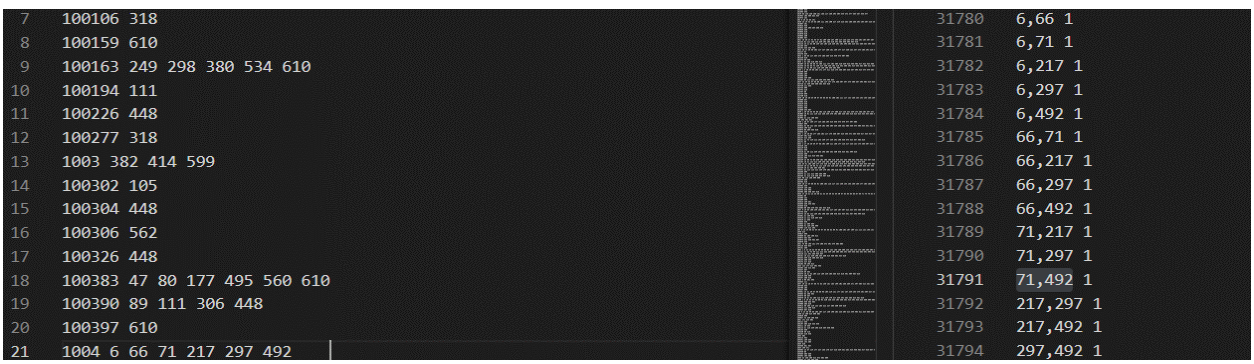
Referring to [20], I learned to use the .join command. At first, I uses ‘ ’.join to separate the two user id’s in the pair by a space. After doing more thinking, I realized I would need to use a character such as a comma to put the user ID’s together so that the two could be treated together as a key. Otherwise, it was possible that some of the user pairs that were identical would go to different reducers if they were only sorted based on the first number (due to the space separation of the original design).

Note: The next few paragraphs before the next screenshot were written when I originally had my output as "User A User B 1" instead of "User A,User B 1" so the search result numbers in the final code are different.

Initially, I validated the output by running the mapper code local on the output of my first MapReduce job on the small dataset, then used ctrl + f in visual studio code, searching with whole words only. When searching for something like "111 125" (without quotes), I was surprised to see a total of 44 results for the output of the 1st MapReduce job, and 105 results for the output of the second mapper, when I expected only 44. Did I have an error in implementation that caused my 2nd mapper to print erroneous entries? I tried tracing manually by enumerating the pairs from the array of users I saw in lexicographic order, but it matched up exactly with the output of my local testing of the 2nd mapper every time.

After awhile, I saw my error. Searching "111 125" would only return results in the output of the first MapReduce job only when users 111 and 125 were right next to each other. If there was any user in-between 111 and 125 that had also watched the movie, there wouldn't be any result.

In other words, the Mapper code I created was correct but because of an incorrect validation method, I also thought the output of the mapper was incorrect.



Here is an example of both the output and validation. On the left is the output of the first MapReduce job, and on the right are lines corresponding to the enumerated pairs of users in the output of the 2nd mapper for line 21 of the 1st MapReduce job. It can be seen through a quick recitation of the possible pairs of users 6, 66, 71, 217, 297, and 492 that every pair is included and there are no erroneous pairs listed.

Reducer #2:

Input: Output of the 2nd mapper.

Format of input: User A,User B 1

Output key: User A,User B (a pair of users)

Output value: The number of movies the pair of users have both watched

Full output example: User A,User B 47

Example with table from Assignment 1 PDF:

Mapper output / reducer input:

User A,User B 1

User A,User C 1

User A,User D 1

User B,User C 1

User B,User D 1

User C,User D 1

User A,User B 1

User A,User D 1

User B,User D 1

User B,User C 1

Reducer output:

User A,User B 2

User A,User C 1

User A,User D 2

User B,User C 2

User B,User D 2

User C,User D 1

Part A Reducer #2 code:

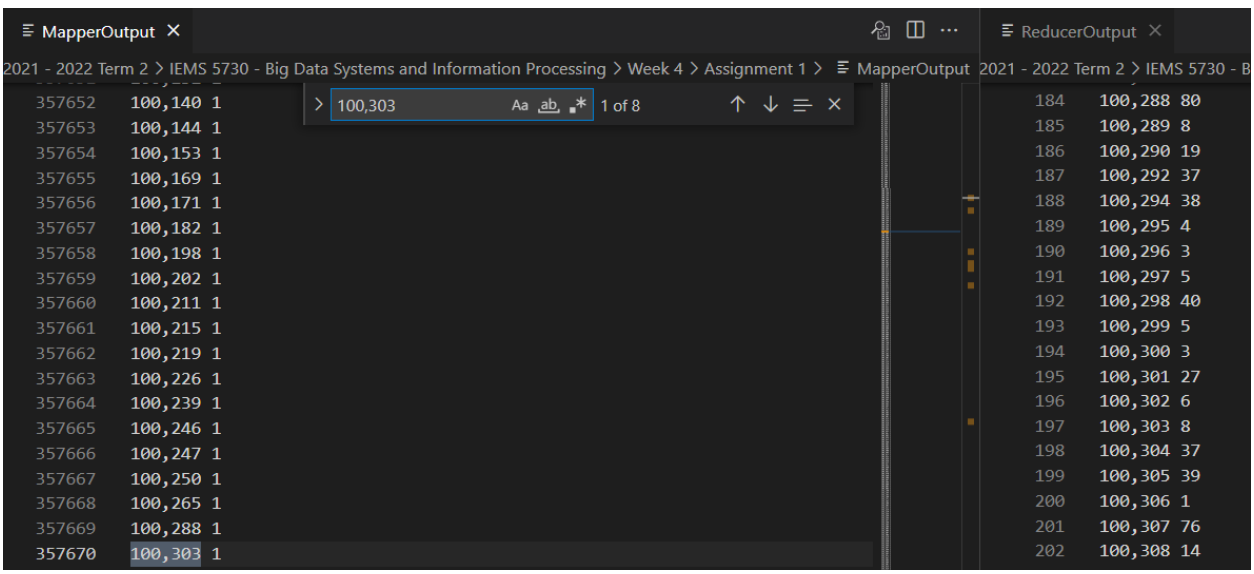
WatchPairsReducer.py 2 ✕ ReducerOutput

2022 Term 2 > IEMS 5730 - Big Data Systems and Information Processing > Week 4 > Assignment 1 > WatchPairsReducer.py

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

from operator import itemgetter
import sys

# This code is based on my IndividualCountReducer code.
# The main difference is the key is a pair of users
# instead of just one user, and the separated is a space rather than a
# tab.

# Changing current_user to current_users instead to reflect
# that the key is a pair of users.

current_users = None
current_count = 0
users = None

# input comes from STDIN

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

    # Parse the input we got from mapper.
    # None reflects default behavior, which is separation by whitespace.
    users, count = line.split(None, 1)

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

    # This If switch works because Hadoop sorts mapout by Key
    # before it is passed to the reducer

```

```
    if current_users == users:
        current_count += count
    else:
        if current_users:
            # Write result to STDOUT
            print '%s\t%s' % (current_users, current_count)
        current_count = count
        current_users = users
if current_users == users:
    print '%s\t%s' % (current_users, current_count)
```

This code is almost exactly the same as the IndividualCountReducer code except I change user to users and current_user to current_users. This works because the reducer is doing the same thing, taking the value from each line, converting it to an integer, and summing up the value for each key. The only other difference is that for the split command, I use the arguments of (None, 1) instead of ('\t', 1) because my input is space separated rather than tab separated.



Here is an example of the output and validation. On the left, is the output from the second MapReduce, and on the right, we can see the output from the Reducer. I would search the mapper output using whole words only for random pairs, such as 100,303, and confirm that the count on the Reducer Output, 8, matches the number of found pairs from the mapper output.

Small Dataset Command:

“hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar\

-file WatchPairsMapper.py -mapper WatchPairsMapper.py\

-file WatchPairsReducer.py -combiner WatchPairsReducer.py -reducer WatchPairsReducer.py\

-input MovieUsersSmall\

-output WatchPairsSmall”

[21] Informs me to put the -combiner command after the mapper but before the reducer. I also need to put it after the -file command for the Reducer so it has the file loaded, since I am reusing the reducer code for local partial aggregation of results to reduce traffic over the network, which should help with performance. Because I am testing with the small dataset, I don’t need to use commands to modify the number of mappers and reducers, or do compression yet.

Because I am using the input from the first MapReduce job, which is MovieUsersSmall, I load it from local into the HDFS.

```
[s1155160788@dicvmd10 input]$ hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streami
ng.jar -file WatchPairsMapper.py -mapper WatchPairsMapper.py -file WatchPairsReducer.py -comb
iner WatchPairsReducer.py -reducer WatchPairsReducer.py -input MovieUsersSmall -output WatchP
airsSmall
22/02/14 10:09:54 WARN streaming.StreamJob: -file option is deprecated, please use generic op
tion -files instead.
packageJobJar: [WatchPairsMapper.py, WatchPairsReducer.py] [/usr/hdp/2.6.5.0-292/hadoop-mapre
duce/hadoop-streaming-2.7.3.2.6.5.0-292.jar] /tmp/streamjob2529774255216821663.jar tmpDir=nul
l
22/02/14 10:09:55 INFO client.AHSProxy: Connecting to Application History server at dicvmd2.i
e.cuhk.edu.hk/172.16.5.202:10200
22/02/14 10:09:55 INFO client.AHSProxy: Connecting to Application History server at dicvmd2.i
e.cuhk.edu.hk/172.16.5.202:10200
22/02/14 10:09:55 INFO client.RequestHedgingRMFailoverProxyProvider: Looking for the active R
M in [rm1, rm2]...
22/02/14 10:09:55 INFO client.RequestHedgingRMFailoverProxyProvider: Found active RM [rm1]
22/02/14 10:09:55 INFO mapred.FileInputFormat: Total input paths to process : 1
22/02/14 10:09:55 INFO mapreduce.JobSubmitter: number of splits:2
22/02/14 10:09:56 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1642685809406_0
156
22/02/14 10:09:56 INFO impl.YarnClientImpl: Submitted application application_1642685809406_0
156
22/02/14 10:09:56 INFO mapreduce.Job: The url to track the job: http://dicvmd3.ie.cuhk.edu.hk
:8088/proxy/application_1642685809406_0156/
22/02/14 10:09:56 INFO mapreduce.Job: Running job: job_1642685809406_0156
22/02/14 10:10:00 INFO mapreduce.Job: Job job_1642685809406_0156 running in uber mode : false
22/02/14 10:10:00 INFO mapreduce.Job:  map 0% reduce 0%
22/02/14 10:10:08 INFO mapreduce.Job:  map 50% reduce 0%
22/02/14 10:10:14 INFO mapreduce.Job:  map 100% reduce 0%
22/02/14 10:10:17 INFO mapreduce.Job:  map 100% reduce 100%
22/02/14 10:10:18 INFO mapreduce.Job: Job job_1642685809406_0156 completed successfully
22/02/14 10:10:18 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=3471064
                FILE: Number of bytes written=7408837
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=581711
                HDFS: Number of bytes written=1659244
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        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)=537096
                Total time spent by all reduces in occupied slots (ms)=210128
                Total time spent by all map tasks (ms)=11676
```

```
        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)=537096
                Total time spent by all reduces in occupied slots (ms)=210128
                Total time spent by all map tasks (ms)=11676
                Total time spent by all reduce tasks (ms)=2284
                Total vcore-milliseconds taken by all map tasks=11676
                Total vcore-milliseconds taken by all reduce tasks=2284
                Total megabyte-milliseconds taken by all map tasks=549986304
                Total megabyte-milliseconds taken by all reduce tasks=215171072
        Map-Reduce Framework
                Map input records=9724
                Map output records=2911932
                Map output bytes=31018668
                Map output materialized bytes=3471070
                Input split bytes=238
                Combine input records=2911932
                Combine output records=291083
                Reduce input groups=164054
                Reduce shuffle bytes=3471070
                Reduce input records=291083
                Reduce output records=164054
                Spilled Records=582166
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=156
                CPU time spent (ms)=8180
                Physical memory (bytes) snapshot=5741842432
                Virtual memory (bytes) snapshot=166733152256
                Total committed heap usage (bytes)=6176112640
        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=581473
        File Output Format Counters
                Bytes Written=1659244
22/02/14 10:10:18 INFO streaming.StreamJob: Output directory: WatchPairsSmall
[s1155160788@dicvmd10 input]$
```

The job completes successfully.

```
248,328 19
248,330 17
248,331 14
248,332 17
248,334 17
248,335 2
248,336 2
248,337 2
248,338 5
248,339 19
248,340 2
248,341 1
248,342 1
248,343 5
248,344 9
248,347 3
248,348 3
248,349 2
248,350 2
248,351 6
248,352 20
```

The output is in the ideal format. I validate the output by comparing it to my locally tested reducer output, and the result is identical in size, number of lines, and the number of movies each pair have watched, the only difference is slightly different sorting due to the MapReduce sorting method being different.

```
[s1155160788@dicvmd10 input]$ cat WatchPairsSmall | sort -nrk 2 | head -n 10 > top10WatchSmall
```

After looking at Sort Unix documentation in [22], I devise a way to take the output from the second MapReduce job from Part A and sort it by the number of movies each pair have both watched in descending order without needing a third MapReduce job.

-nrk 2 parameters for the Sort command sort on the 2nd field (the value), numerically, in reverse (descending instead of ascending) order, then head -n 10 takes the top 10 figures. The results seem reasonable. I save the screenshot of the top10 output for when I detail the final top 10 results for both the small and large dataset at the end of Part A.

For the large dataset, I need to make some optimizations for performance and scalability. A lot has been done in terms of my design, and also using the combiner to improve performance, but I should also increase the number of mappers and reducers, as well as compress the intermediate output to prevent memory issues as suggested in the Assignment 1 PDF.

I refer to [23] Hadoop streaming documentation as well as [24] for optimizing the compression of the output. I am specifically trying to compress the Mapper intermediate output to prevent memory issues.

I refer to [25] linked from the Assignment 1 PDF when thinking about how many mappers and reducers to have. There are 7 active nodes according to the IE DIC cluster website, and it suggests 10-100 mappers per node. According to [26], the MapReduce paper from Google we were asked to read, for 2000 nodes, they used 200,000 mappers and 5,000 reducers. I'll set an upper limit of 350 mappers (50 mappers per node) and set the reducers at 14, for 2 reducers per node. I use [27] to confirm the syntax for setting the mappers and reducers.

I move the MovieUsersLarge into the hdfs using the hdfs dfs -copyFromLocal command.

Large dataset Part A Second MapReduce job command:

"hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar\

-D mapreduce.map.output.compress=true -D mapreduce.output.fileoutputformat.compress=false\

-D mapreduce.job.maps=350 -D mapreduce.job.reduces=14\

-file WatchPairsMapper.py -mapper WatchPairsMapper.py\

-file WatchPairsReducer.py -combiner WatchPairsReducer.py -reducer WatchPairsReducer.py\

-input MovieUsersLarge\

-output WatchPairsLarge"

```
[s1155160788@dicvmd10 input]$ hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar -D mapreduce.map.output.compress=true -D mapreduce.output.fileoutputformat.compress=false
 -D mapreduce.job.maps=350 -D mapreduce.job.reduces=14 -file WatchPairsMapper.py -mapper WatchPairsMapper.py -file WatchPairsReducer.py -combiner WatchPairsReducer.py -reducer WatchPairsRed
ucer.py -input MovieUsersLarge -output WatchPairsLarge
22/02/14 11:52:37 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [WatchPairsMapper.py, WatchPairsReducer.py] [/usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar] /tmp/streamjob4662599010485630975.jar tmpDir=null
22/02/14 11:52:38 INFO client.AHSProxy: Connecting to Application History server at dicvmd2.ie.cuhk.edu.hk/172.16.5.202:10200
22/02/14 11:52:38 INFO client.AHSProxy: Connecting to Application History server at dicvmd2.ie.cuhk.edu.hk/172.16.5.202:10200
22/02/14 11:52:38 INFO client.RequestHedgingRMFailoverProxyProvider: Looking for the active RM in [rm1, rm2]...
22/02/14 11:52:38 INFO client.RequestHedgingRMFailoverProxyProvider: Found active RM [rm1]
22/02/14 11:52:38 INFO mapred.FileInputFormat: Total input paths to process : 1
22/02/14 11:52:38 INFO mapreduce.JobSubmitter: number of splits:350
22/02/14 11:52:38 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1642685809406_0161
22/02/14 11:52:38 INFO impl.YarnClientImpl: Submitted application application_1642685809406_0161
22/02/14 11:52:38 INFO mapreduce.Job: The url to track the job: http://dicvmd3.ie.cuhk.edu.hk:8088/proxy/application_1642685809406_0161/
22/02/14 11:52:38 INFO mapreduce.Job: Running job: job_1642685809406_0161
22/02/14 11:52:42 INFO mapreduce.Job: Job job_1642685809406_0161 running in uber mode : false
22/02/14 11:52:42 INFO mapreduce.Job:  map 0% reduce 0%
22/02/14 11:53:00 INFO mapreduce.Job:  map 1% reduce 0%
22/02/14 11:53:32 INFO mapreduce.Job:  map 2% reduce 0%
22/02/14 11:53:58 INFO mapreduce.Job:  map 3% reduce 0%
22/02/14 11:54:25 INFO mapreduce.Job:  map 4% reduce 0%
22/02/14 11:54:46 INFO mapreduce.Job:  map 5% reduce 0%
22/02/14 11:55:11 INFO mapreduce.Job:  map 6% reduce 0%
22/02/14 11:55:32 INFO mapreduce.Job:  map 7% reduce 0%
22/02/14 11:56:00 INFO mapreduce.Job:  map 8% reduce 0%
22/02/14 11:56:21 INFO mapreduce.Job:  map 9% reduce 0%
22/02/14 11:56:50 INFO mapreduce.Job:  map 10% reduce 0%
22/02/14 11:57:14 INFO mapreduce.Job:  map 11% reduce 0%
22/02/14 11:57:47 INFO mapreduce.Job:  map 12% reduce 0%
22/02/14 11:58:13 INFO mapreduce.Job:  map 13% reduce 0%
22/02/14 11:58:42 INFO mapreduce.Job:  map 14% reduce 0%
22/02/14 11:59:04 INFO mapreduce.Job:  map 15% reduce 0%
22/02/14 11:59:37 INFO mapreduce.Job:  map 16% reduce 0%
22/02/14 12:00:00 INFO mapreduce.Job:  map 17% reduce 0%
22/02/14 12:00:29 INFO mapreduce.Job:  map 18% reduce 0%
22/02/14 12:00:53 INFO mapreduce.Job:  map 19% reduce 0%
22/02/14 12:01:16 INFO mapreduce.Job:  map 20% reduce 0%
22/02/14 12:01:32 INFO mapreduce.Job:  map 21% reduce 0%
22/02/14 12:01:58 INFO mapreduce.Job:  map 22% reduce 0%
22/02/14 12:02:19 INFO mapreduce.Job:  map 23% reduce 0%
22/02/14 12:02:46 INFO mapreduce.Job:  map 24% reduce 0%
22/02/14 12:03:09 INFO mapreduce.Job:  map 25% reduce 0%
22/02/14 12:03:39 INFO mapreduce.Job:  map 26% reduce 0%
22/02/14 12:04:00 INFO mapreduce.Job:  map 27% reduce 0%
22/02/14 12:04:30 INFO mapreduce.Job:  map 28% reduce 0%
22/02/14 12:04:53 INFO mapreduce.Job:  map 29% reduce 0%
22/02/14 12:05:19 INFO mapreduce.Job:  map 30% reduce 0%
22/02/14 12:05:41 INFO mapreduce.Job:  map 31% reduce 0%
22/02/14 12:06:08 INFO mapreduce.Job:  map 32% reduce 0%
22/02/14 12:06:29 INFO mapreduce.Job:  map 33% reduce 0%
```

```
22/02/14 12:06:54 INFO mapreduce.Job:  map 34% reduce 0%
22/02/14 12:07:13 INFO mapreduce.Job:  map 35% reduce 0%
22/02/14 12:07:41 INFO mapreduce.Job:  map 36% reduce 0%
22/02/14 12:08:04 INFO mapreduce.Job:  map 37% reduce 0%
22/02/14 12:08:30 INFO mapreduce.Job:  map 38% reduce 0%
22/02/14 12:08:51 INFO mapreduce.Job:  map 39% reduce 0%
22/02/14 12:09:18 INFO mapreduce.Job:  map 40% reduce 0%
22/02/14 12:09:39 INFO mapreduce.Job:  map 41% reduce 0%
22/02/14 12:10:08 INFO mapreduce.Job:  map 42% reduce 0%
22/02/14 12:10:28 INFO mapreduce.Job:  map 43% reduce 0%
22/02/14 12:10:54 INFO mapreduce.Job:  map 44% reduce 0%
22/02/14 12:11:15 INFO mapreduce.Job:  map 45% reduce 0%
22/02/14 12:11:39 INFO mapreduce.Job:  map 46% reduce 0%
22/02/14 12:11:59 INFO mapreduce.Job:  map 47% reduce 0%
22/02/14 12:12:26 INFO mapreduce.Job:  map 48% reduce 0%
22/02/14 12:12:45 INFO mapreduce.Job:  map 49% reduce 0%
22/02/14 12:13:12 INFO mapreduce.Job:  map 50% reduce 0%
22/02/14 12:13:32 INFO mapreduce.Job:  map 51% reduce 0%
22/02/14 12:13:59 INFO mapreduce.Job:  map 52% reduce 0%
22/02/14 12:14:17 INFO mapreduce.Job:  map 53% reduce 0%
22/02/14 12:14:45 INFO mapreduce.Job:  map 54% reduce 0%
22/02/14 12:15:04 INFO mapreduce.Job:  map 55% reduce 0%
22/02/14 12:15:27 INFO mapreduce.Job:  map 56% reduce 0%
22/02/14 12:15:47 INFO mapreduce.Job:  map 57% reduce 0%
22/02/14 12:16:15 INFO mapreduce.Job:  map 58% reduce 0%
22/02/14 12:16:34 INFO mapreduce.Job:  map 59% reduce 0%
22/02/14 12:17:01 INFO mapreduce.Job:  map 60% reduce 0%
22/02/14 12:17:18 INFO mapreduce.Job:  map 61% reduce 0%
22/02/14 12:17:44 INFO mapreduce.Job:  map 62% reduce 0%
22/02/14 12:18:03 INFO mapreduce.Job:  map 63% reduce 0%
22/02/14 12:18:30 INFO mapreduce.Job:  map 64% reduce 0%
22/02/14 12:18:50 INFO mapreduce.Job:  map 65% reduce 0%
22/02/14 12:19:14 INFO mapreduce.Job:  map 66% reduce 0%
22/02/14 12:19:33 INFO mapreduce.Job:  map 67% reduce 0%
22/02/14 12:19:58 INFO mapreduce.Job:  map 68% reduce 0%
22/02/14 12:20:17 INFO mapreduce.Job:  map 69% reduce 0%
22/02/14 12:20:38 INFO mapreduce.Job:  map 70% reduce 0%
22/02/14 12:20:59 INFO mapreduce.Job:  map 71% reduce 0%
22/02/14 12:21:25 INFO mapreduce.Job:  map 72% reduce 0%
22/02/14 12:21:41 INFO mapreduce.Job:  map 73% reduce 0%
22/02/14 12:22:05 INFO mapreduce.Job:  map 74% reduce 0%
22/02/14 12:22:21 INFO mapreduce.Job:  map 75% reduce 0%
22/02/14 12:22:42 INFO mapreduce.Job:  map 76% reduce 0%
22/02/14 12:22:58 INFO mapreduce.Job:  map 77% reduce 0%
22/02/14 12:23:20 INFO mapreduce.Job:  map 78% reduce 0%
22/02/14 12:23:35 INFO mapreduce.Job:  map 79% reduce 0%
22/02/14 12:23:59 INFO mapreduce.Job:  map 80% reduce 0%
22/02/14 12:24:17 INFO mapreduce.Job:  map 81% reduce 0%
22/02/14 12:24:38 INFO mapreduce.Job:  map 82% reduce 0%
22/02/14 12:24:55 INFO mapreduce.Job:  map 83% reduce 0%
```

```
22/02/14 12:25:17 INFO mapreduce.Job:  map 84% reduce 0%
22/02/14 12:25:36 INFO mapreduce.Job:  map 85% reduce 0%
22/02/14 12:25:59 INFO mapreduce.Job:  map 86% reduce 0%
22/02/14 12:26:19 INFO mapreduce.Job:  map 87% reduce 0%
22/02/14 12:26:44 INFO mapreduce.Job:  map 88% reduce 0%
22/02/14 12:27:04 INFO mapreduce.Job:  map 89% reduce 0%
22/02/14 12:27:29 INFO mapreduce.Job:  map 90% reduce 0%
22/02/14 12:27:48 INFO mapreduce.Job:  map 91% reduce 0%
22/02/14 12:28:11 INFO mapreduce.Job:  map 92% reduce 0%
22/02/14 12:28:26 INFO mapreduce.Job:  map 93% reduce 0%
22/02/14 12:28:46 INFO mapreduce.Job:  map 94% reduce 0%
22/02/14 12:29:03 INFO mapreduce.Job:  map 95% reduce 0%
22/02/14 12:29:29 INFO mapreduce.Job:  map 96% reduce 0%
22/02/14 12:29:45 INFO mapreduce.Job:  map 97% reduce 0%
22/02/14 12:30:06 INFO mapreduce.Job:  map 98% reduce 0%
22/02/14 12:30:27 INFO mapreduce.Job:  map 99% reduce 0%
22/02/14 12:30:57 INFO mapreduce.Job:  map 100% reduce 0%
22/02/14 12:31:13 INFO mapreduce.Job:  map 100% reduce 3%
22/02/14 12:31:19 INFO mapreduce.Job:  map 100% reduce 4%
22/02/14 12:31:25 INFO mapreduce.Job:  map 100% reduce 5%
22/02/14 12:31:40 INFO mapreduce.Job:  map 100% reduce 6%
22/02/14 12:31:55 INFO mapreduce.Job:  map 100% reduce 7%
22/02/14 12:32:12 INFO mapreduce.Job:  map 100% reduce 10%
22/02/14 12:32:15 INFO mapreduce.Job:  map 100% reduce 11%
22/02/14 12:32:24 INFO mapreduce.Job:  map 100% reduce 12%
22/02/14 12:32:36 INFO mapreduce.Job:  map 100% reduce 13%
22/02/14 12:32:51 INFO mapreduce.Job:  map 100% reduce 14%
22/02/14 12:33:12 INFO mapreduce.Job:  map 100% reduce 17%
22/02/14 12:33:15 INFO mapreduce.Job:  map 100% reduce 18%
22/02/14 12:33:24 INFO mapreduce.Job:  map 100% reduce 19%
22/02/14 12:33:33 INFO mapreduce.Job:  map 100% reduce 20%
22/02/14 12:33:51 INFO mapreduce.Job:  map 100% reduce 21%
22/02/14 12:34:13 INFO mapreduce.Job:  map 100% reduce 24%
22/02/14 12:34:16 INFO mapreduce.Job:  map 100% reduce 25%
22/02/14 12:34:22 INFO mapreduce.Job:  map 100% reduce 26%
22/02/14 12:34:31 INFO mapreduce.Job:  map 100% reduce 27%
22/02/14 12:34:49 INFO mapreduce.Job:  map 100% reduce 28%
22/02/14 12:35:04 INFO mapreduce.Job:  map 100% reduce 29%
22/02/14 12:35:13 INFO mapreduce.Job:  map 100% reduce 32%
22/02/14 12:35:22 INFO mapreduce.Job:  map 100% reduce 33%
22/02/14 12:35:28 INFO mapreduce.Job:  map 100% reduce 34%
22/02/14 12:35:46 INFO mapreduce.Job:  map 100% reduce 35%
22/02/14 12:36:05 INFO mapreduce.Job:  map 100% reduce 36%
22/02/14 12:36:13 INFO mapreduce.Job:  map 100% reduce 39%
22/02/14 12:36:20 INFO mapreduce.Job:  map 100% reduce 40%
22/02/14 12:36:29 INFO mapreduce.Job:  map 100% reduce 41%
22/02/14 12:36:44 INFO mapreduce.Job:  map 100% reduce 42%
22/02/14 12:37:02 INFO mapreduce.Job:  map 100% reduce 43%
22/02/14 12:37:13 INFO mapreduce.Job:  map 100% reduce 46%
22/02/14 12:37:20 INFO mapreduce.Job:  map 100% reduce 47%
```

```
22/02/14 12:37:13 INFO mapreduce.Job:  map 100% reduce 46%
22/02/14 12:37:20 INFO mapreduce.Job:  map 100% reduce 47%
22/02/14 12:37:26 INFO mapreduce.Job:  map 100% reduce 48%
22/02/14 12:37:41 INFO mapreduce.Job:  map 100% reduce 49%
22/02/14 12:37:59 INFO mapreduce.Job:  map 100% reduce 50%
22/02/14 12:38:13 INFO mapreduce.Job:  map 100% reduce 53%
22/02/14 12:38:19 INFO mapreduce.Job:  map 100% reduce 54%
22/02/14 12:38:26 INFO mapreduce.Job:  map 100% reduce 55%
22/02/14 12:38:41 INFO mapreduce.Job:  map 100% reduce 56%
22/02/14 12:38:56 INFO mapreduce.Job:  map 100% reduce 57%
22/02/14 12:39:14 INFO mapreduce.Job:  map 100% reduce 60%
22/02/14 12:39:17 INFO mapreduce.Job:  map 100% reduce 61%
22/02/14 12:39:27 INFO mapreduce.Job:  map 100% reduce 62%
22/02/14 12:39:39 INFO mapreduce.Job:  map 100% reduce 63%
22/02/14 12:39:54 INFO mapreduce.Job:  map 100% reduce 64%
22/02/14 12:40:14 INFO mapreduce.Job:  map 100% reduce 67%
22/02/14 12:40:17 INFO mapreduce.Job:  map 100% reduce 68%
22/02/14 12:40:24 INFO mapreduce.Job:  map 100% reduce 69%
22/02/14 12:40:36 INFO mapreduce.Job:  map 100% reduce 70%
22/02/14 12:40:54 INFO mapreduce.Job:  map 100% reduce 71%
22/02/14 12:41:13 INFO mapreduce.Job:  map 100% reduce 74%
22/02/14 12:41:16 INFO mapreduce.Job:  map 100% reduce 75%
22/02/14 12:41:23 INFO mapreduce.Job:  map 100% reduce 76%
22/02/14 12:41:32 INFO mapreduce.Job:  map 100% reduce 77%
22/02/14 12:41:50 INFO mapreduce.Job:  map 100% reduce 78%
22/02/14 12:42:05 INFO mapreduce.Job:  map 100% reduce 79%
22/02/14 12:42:12 INFO mapreduce.Job:  map 100% reduce 82%
22/02/14 12:42:22 INFO mapreduce.Job:  map 100% reduce 83%
22/02/14 12:42:31 INFO mapreduce.Job:  map 100% reduce 84%
22/02/14 12:42:46 INFO mapreduce.Job:  map 100% reduce 85%
22/02/14 12:43:04 INFO mapreduce.Job:  map 100% reduce 86%
22/02/14 12:43:13 INFO mapreduce.Job:  map 100% reduce 89%
22/02/14 12:43:19 INFO mapreduce.Job:  map 100% reduce 90%
22/02/14 12:43:29 INFO mapreduce.Job:  map 100% reduce 91%
22/02/14 12:43:44 INFO mapreduce.Job:  map 100% reduce 92%
22/02/14 12:44:02 INFO mapreduce.Job:  map 100% reduce 93%
22/02/14 12:44:13 INFO mapreduce.Job:  map 100% reduce 96%
22/02/14 12:44:19 INFO mapreduce.Job:  map 100% reduce 97%
22/02/14 12:44:26 INFO mapreduce.Job:  map 100% reduce 98%
22/02/14 12:44:44 INFO mapreduce.Job:  map 100% reduce 99%
22/02/14 12:44:59 INFO mapreduce.Job:  map 100% reduce 100%
22/02/14 12:45:06 INFO mapreduce.Job: Job job_1642685809406_0161 completed successfully
22/02/14 12:45:06 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=536371071
                FILE: Number of bytes written=1873945036
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=54767740
```

```
22/02/14 12:45:06 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=536371071
                FILE: Number of bytes written=1873945036
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=54767740
                HDFS: Number of bytes written=1889373752
                HDFS: Number of read operations=1092
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=28
        Job Counters
                Launched map tasks=350
                Launched reduce tasks=14
                Data-local map tasks=350
                Total time spent by all maps in occupied slots (ms)=97081574
                Total time spent by all reduces in occupied slots (ms)=76820920
                Total time spent by all map tasks (ms)=2110469
                Total time spent by all reduce tasks (ms)=835010
                Total vcore-milliseconds taken by all map tasks=2110469
                Total vcore-milliseconds taken by all reduce tasks=835010
                Total megabyte-milliseconds taken by all map tasks=99411531776
                Total megabyte-milliseconds taken by all reduce tasks=78664622080
        Map-Reduce Framework
                Map input records=8646
                Map output records=367528712
                Map output bytes=5404123645
                Map output materialized bytes=1280829817
                Input split bytes=41650
                Combine input records=367528712
                Combine output records=349442996
                Reduce input groups=137542274
                Reduce shuffle bytes=1280829817
                Reduce input records=349442996
                Reduce output records=137542274
                Spilled Records=698885992
                Shuffled Maps =4900
                Failed Shuffles=0
                Merged Map outputs=4900
                GC time elapsed (ms)=46150
                CPU time spent (ms)=3055120
                Physical memory (bytes) snapshot=680286453760
                Virtual memory (bytes) snapshot=15830950498304
                Total committed heap usage (bytes)=821718155264
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
```

```
        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=54726090
        File Output Format Counters
                Bytes Written=1889373752
22/02/14 12:45:06 INFO streaming.StreamJob: Output directory: WatchPairsLarge
[s1155160788@dicvmd10 input]$
```

User:	s1155160788
Name:	streamjob4662599010485630975.jar
Application Type:	MAPREDUCE
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Mon Feb 14 11:52:38 +0800 2022
Elapsed:	52mins, 26sec
Tracking URL:	History
Log Aggregation Status	SUCCEEDED
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	<DEFAULT_PARTITION>

According to the IE DIC cluster, the job completed successfully in 52 minutes, 26 seconds, and from the console output, we can see it used 350 mappers and 14 reducers.

output
WatchPairsLarge

Filename	Filesize	Filetype	Last modifi...	Permissi...	Owner/Gr...
part-00000	134,926,150	File	2/14/2022 ...	-rw-r--r--	s1155160...
part-00001	134,921,842	File	2/14/2022 ...	-rw-r--r--	s1155160...
part-00002	135,054,379	File	2/14/2022 ...	-rw-r--r--	s1155160...
part-00003	134,949,532	File	2/14/2022 ...	-rw-r--r--	s1155160...
part-00004	134,956,963	File	2/14/2022 ...	-rw-r--r--	s1155160...
part-00005	134,922,991	File	2/14/2022 ...	-rw-r--r--	s1155160...
part-00006	134,934,881	File	2/14/2022 ...	-rw-r--r--	s1155160...
part-00007	135,008,987	File	2/14/2022 ...	-rw-r--r--	s1155160...
part-00008	134,956,313	File	2/14/2022 ...	-rw-r--r--	s1155160...
part-00009	134,979,639	File	2/14/2022 ...	-rw-r--r--	s1155160...
part-00010	134,993,659	File	2/14/2022 ...	-rw-r--r--	s1155160...
part-00011	134,904,453	File	2/14/2022 ...	-rw-r--r--	s1155160...
part-00012	134,916,968	File	2/14/2022 ...	-rw-r--r--	s1155160...
part-00013	134,946,995	File	2/14/2022 ...	-rw-r--r--	s1155160...

There are 14 output files, as I would expect from my specified 14 reducers.

```
[s1155160788@dicvmd10 WatchPairsLarge]$ cat part* > WatchPairsLarge
```

I use this command to quickly combine all the parts, then transferred it through network to my local laptop to validate the results. Because the file was so large it took about 9 minutes to move over to my computer.

WatchPairsLarge

C: > Users > Eddie > Documents > Sc

```
10,10109    1
10,1011 1
10,10228    2
10,10277    1
10,1088 3
10,10963    1
10,11005    1
10,11117    1
10,11306    1
10,11418    1
10,11544    2
10,11593    1
10,11670    1
10,11719    1
10,11908    1
10,12125    3
10,12671    2
10,12965    1
10,13007    1
10,1347 1
10,13476    1
10,14134    5
10,1424 1
10,14358    1
10,15247    1
10,15520    1
10,15807    1
10,15968    3
10,16066    1
10,16073    1
10,16374    1
10,16451    1
10,16514    1
10,16556    1
10,16640    2
10,1697 1
10,17074    1
10,17081    1
```

I quickly realized it wouldn't be possible to validate the results for the large dataset. At 1.84 GB, the file was large and my laptop had difficulty even opening the file in Visual Studio Code. I had to restart Visual Studio Code with more memory, and even, then, it would only show the first 37 lines, and not let me scroll further down. The output appeared to be formatted correctly, and there was counts beyond 1, so the reducer was summing results.

Another issue is that to validate results, I would either need to find all the occurrences of a pair of users within the output of my first MapReduce job manually (a difficult task), or run the second MapReduce job locally using cat data | mapper | sort | reducer. This would take longer than the second MapReduce job took on the cluster most likely, as it is only one node instead of 7, and since the first run took 52 minutes, it would probably take at least 7 hours to generate a local large dataset output from MapReduce job #2, so I decided to just trust the output based on the soundness of the code in the small dataset validations.

```
[s1155160788@dicvmd10 output]$ cat WatchPairsLarge | sort -nrk 2 | head -n 10 > top10WatchLar
ge
```

I ran the same command as before to sort the top 10 results and after waiting awhile (~10 minutes, I went away) the command completed.

Part A Final Submission: Top 10 Pairs for Each Dataset

In this section, there are two screenshots. The first screenshot is the top 10 pairs that have the most amount of movies watched in common with each other for the small dataset. The second screenshot is the same, but for the large dataset. I manually edited the text documents to add colons to match the requested formatting.

The first number on the left, numbered 1-10 is the line number. The second number is the pair of users, and the final number is the number of movies the pair of users have both watched.

Line# UserA,UserB: Number_Of_Movies_In_Common

```
1   414,599:    1338
2   414,474:    1077
3   68,414:     950
4   414,448:    914
5   274,414:    856
6   474,599:    837
7   68,599:     790
8   448,599:    790
9   274,599:    783
10  380,414:    723
```

```
59269,67385:    3631
14463,59269:    2631
59269,68259:    2417
67385,68259:    2180
19635,59269:    2148
27468,59269:    2132
14463,67385:    2115
27468,67385:    2002
19635,67385:    1894
59269,63134:    1872
```

One observation is that there are some pairs that are very common in these top 10 lists, such as 414 and 599 in the small dataset and users 59269 and 67385 in the large dataset. This is to be expected because the users that have the most amount of movies watched are also the most likely to have user pairs with people that have also watched the same movie.

Part B covers relative similarity more in depth than Part A, which is concerned with absolute similarity.

Part B MapReduce Job #2: Find the top similar users to each user

Since we are asked to compute the top 3 similar users to each user, I don't need to compute every possible pair of users, which would make the calculation take much longer. I only need to enumerate pairs of users that have watched at least one movie in common, because if they haven't, there is a similarity score of 0, and it would not be one of the top 3 pairs. As a result, I can use the output of the 2nd MapReduce job of Part A as the input for the 2nd MapReduce job of Part B.

We actually need two inputs for this MapReduce job. To calculate the similarity between a pair of users, we need 4 pieces of information: The pair of users, the movies User A has watched, the movies User B has watched, and the movies both User A and User B have watched. The 2nd Map Reduce job of Part A provides two of these pieces, and the movies User A and User B have watched individually are recorded in the output of the 1st MapReduce job of Part B. To get this later input, we load it during the mapper script as a file, and propagate the needed information to the reducer through printing.

Note: I needed multiple attempts to get Part B working correctly and in a timely manner. I will detail my thought process below, but if you just want to look at the final iteration, you can skip to Page 63.

Initial Mapper #2:

Input:

1. List of pairs that have watched the same movies, and the number of movies they have both watched in the format (UserA,UserB (tab) Number_of_movies)
2. Via loading file: List of mostly sorted users in database with number of movies each user has watched in the format: User (tab) Number_of_movies

Output key: Pair of users, in the format UserA,UserB

Output value: (Number of movies both have watched) (tab) (Number of movies User A has watched) (tab) (Number of movies User B has watched).

Initial Part B Mapper #2 code:

SimilarityMapper.py 1

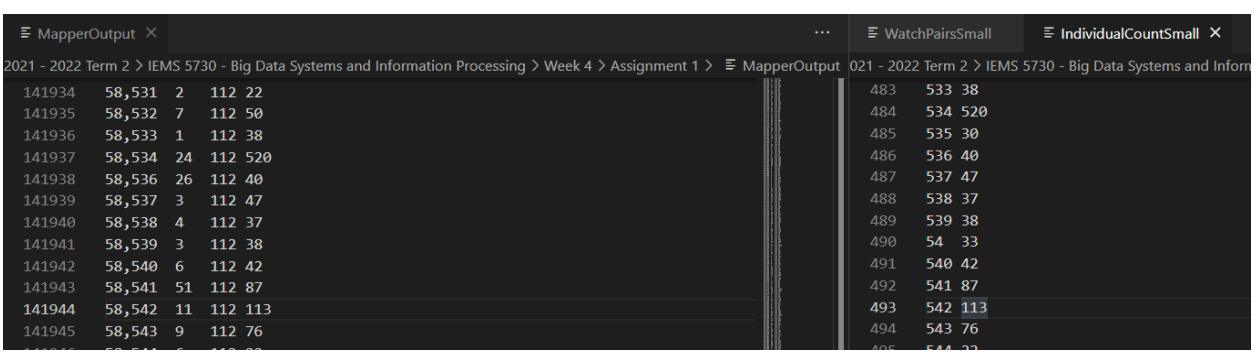
C: > Users > Eddie > Documents > School > CUHK > 2021 - 2022 Term 2 > IEMS 5730 - Big Data Sys

```python
#!/usr/bin/python
from operator import itemgetter
import sys

single_users = []
single_counts = []
list_count = 0 # Used for iterating through the lists later.

with open('IndividualCountSmall') as f:
    for line in f.readlines():
        line = line.strip()

        # Append user and count from each file line to each list.
        single_user, single_count = line.split('\t', 1)
        single_users.append(single_user)
        single_counts.append(single_count)

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

    # Parse the input we got from mapper.
    # Input comes in form of user_pair (tab) count
    users, pair_count = line.split('\t', 1)

    #Here we split up the comma divided pairs into individual users
    user1, user2 = users.split(',', 1)

    # Search through arrays from IndividualCount file to find the
    # individual counts for each user.

    for X in single_users:
        if user1 == X:
            count1 = single_counts[list_count]
            list_count = 0
            break
```

```python
    #Here we split up the comma divided pairs into individual users
    user1, user2 = users.split(',', 1)

    # Search through arrays from IndividualCount file to find the
    # individual counts for each user.

    for X in single_users:
        if user1 == X:
            count1 = single_counts[list_count]
            list_count = 0
            break
        else:
            list_count+= 1

    for X in single_users:
        if user2 == X:
            count2 = single_counts[list_count]
            list_count = 0
            break
        else:
            list_count+= 1

    # Print user pair and all three counts (pair and inidividual)
    print "%s,%s\t%s\t%s\t%s" % (user1, user2, pair_count, count1, count2)
```

I did not need to reference any new sources when making this code. As previously mentioned, I used slides 40-49 of [8] for the loading and reading of files into the Mapper script.

I validated the output by choosing pairs from the Mapper output such as 58,542 and confirming the counts were correct by ensuring the numbers matched the original input data. The first screenshot is matching the MapperOutput to the individual count of one of the users, and the second screenshot is matching the pair count to the count shown for pairs from the previous MapReduce job.



Considerations when planning 2nd Reducer (Attempt #1):

When planning the 2nd reducer, I encountered a problem. It would be possible to calculate the similarity of all the pairs even in the mapper, because I already created all the variables needed for the calculation and printed them out in this initial mapper.

If I calculated the similarity in the mapper, the reducer would take as input, the pair of users as a key and the similarity as the value. But here was a problem; the assignment says we need to calculate the top K similar users for each user, but I wasn't sure how to do that. If we use the pair of users as a key, there is no guarantee that all pairs involving the user will go to the same reducer without a custom partition function. We can't define a partition override on the leftmost user in the pair either, because all the user pairs so far have been in lexicographic order where User A ID < User B ID. Using a partition on the leftmost user would prevent any User A from receiving similarity calculations for pairs it was involved in with users that have an ID less than A, because in the lines where the similarity is calculated for these pairs, User A is on the right side instead of the left.

I thought it was impossible for awhile, or that I would need a 3rd MapReducer before I thought of a way to still get this done in the 2nd MapReduce job.

First, I would need to extend the mapper to calculate the similarity. Afterwards, the output of the mapper is User A (tab) User B (tab) Similarity_Score. This may seem to be an issue because the User B will not receive User A + the similarity score if the key is sorting by User A. I will circumvent this by printing TWICE, the second time with the users switched, so that each user will receive a copy of their partner + the similarity score when sorting by the 1st user key in the reducer stage. The new reducer will then calculate the top K similarity scores for each user.

Attempt #2 for Mapper #2 for Part B:

Input:

1. List of pairs that have watched the same movies, and the number of movies they have both watched in the format (UserA,UserB (tab) Number_of_movies)
2. Via loading file: List of mostly sorted users in database with number of movies each user has watched in the format: User (tab) Number_of_movies

Output key: User A and User B in two separate lines. The first line has User A as the key, the second line has User B as the key.

Output value: User B and User A in two separate lines, with the similarity score in each line.

Line #1: UserA (tab) UserB (tab) Similarity

Line #2: UserB (tab) UserA (tab) Similarity

Part B Mapper #2 Attempt #2 Code:

SimilarityMapper.py 2

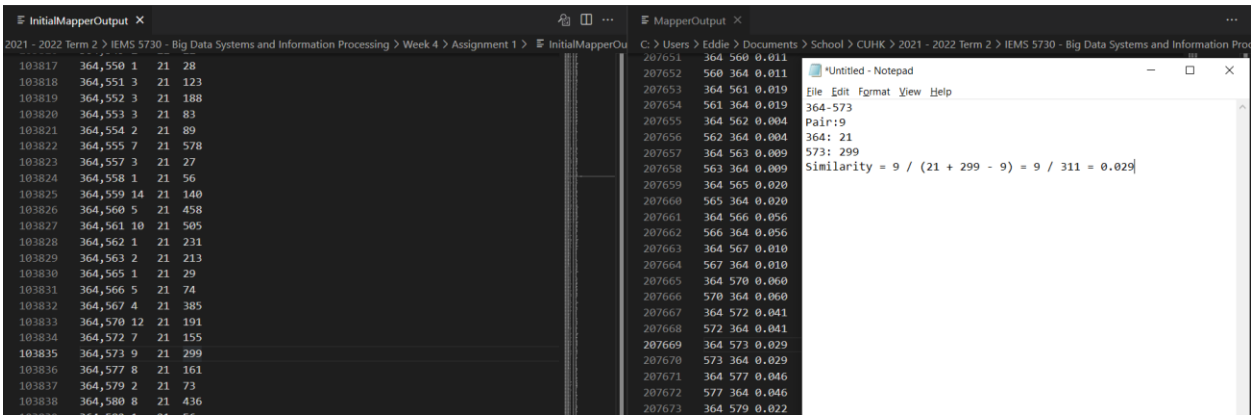
C: > Users > Eddie > Documents > School > CUHK > 2021 - 2022 Term 2 > IEMS 5730 - Big Data System

```python
#!/usr/bin/python
from operator import itemgetter
import sys

single_users = []
single_counts = []
list_count = 0 # Used for iterating through the lists later.

with open('IndividualCountSmall') as f:
    for line in f.readlines():
        line = line.strip()

        # Append user and count from each file line to each list.
        single_user, single_count = line.split('\t', 1)
        single_users.append(single_user)
        single_counts.append(single_count)

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

    # Parse the input we got from mapper.
    # Input comes in form of user_pair (tab) count
    users, pair_count = line.split('\t', 1)

    #Here we split up the comma divided pairs into individual users
    user1, user2 = users.split(',', 1)

    # Search through arrays from IndividualCount file to find the
    # individual counts for each user.

    for X in single_users:
        if user1 == X:
            count1 = single_counts[list_count]
            list_count = 0
            break
```

```python
    # Search through arrays from IndividualCount file to find the
    # individual counts for each user.

    for X in single_users:
        if user1 == X:
            count1 = single_counts[list_count]
            list_count = 0
            break
        else:
            list_count+= 1

    for X in single_users:
        if user2 == X:
            count2 = single_counts[list_count]
            list_count = 0
            break
        else:
            list_count+= 1

    # Now we need to convert each number to a float before calculating the similarity.
    # We need to convert to float instead of integer for division because Python2 will otherwise
    # apply a floor function to the result instead of yielding true decimal division result.

    try:
        pair_count = float(pair_count)
    except ValueError:
        # Was not a number, so silently ignore/discard the line
        continue

    try:
        count1 = float(count1)
    except ValueError:
        # Was not a number, so silently ignore/discard the line
        continue

```

```python
# Now we need to convert each number to a float before calculating the similarity.
# We need to convert to float instead of integer for division because Python2 will otherwise
# apply a floor function to the result instead of yielding true decimal division result.

try:
    pair_count = float(pair_count)
except ValueError:
    # Was not a number, so silently ignore/discard the line
    continue

try:
    count1 = float(count1)
except ValueError:
    # Was not a number, so silently ignore/discard the line
    continue

try:
    count2 = float(count2)
except ValueError:
    # Was not a number, so silently ignore/discard the line
    continue

# Calculate the similarity.

# The equation for similarity is (Number of movies both pairs have watched) / (User A movies + User B movies - number of movies both pairs have watched)
# The numerator is pair_count.
# The denominator is count1 + count2 - pair_count
# Print user pair and all three counts (pair and inidividual)

denominator = count1 + count2 - pair_count
similarity = pair_count / denominator
print "%s\t%s\t%.3f" % (user1, user2, similarity)
print "%s\t%s\t%.3f" % (user2, user1, similarity)
```

Almost everything important about this extended mapper code is written in the comments, however I did discover in [28] that Python 2 uses a floor function if integers are divided, while Python 3 does true division with all types. As a result, I converted to float instead of integer first.



Here is an example of the mapper output (on the right) and one of my validation attempts. On the left is the output of the initial mapper with all the corresponding values related to the count. I choose a random pair from this and calculate the similarity manually. I then need to scroll in the mapper output to the corresponding pair and ensure the similarity is the same. This is made a little more difficult because I tab separated instead of space separated, so I needed to manually scroll through (until I later learned of searching through Visual Studio Code with regular expressions. After doing this several times and the result being the same each time with different pairs, I considered the mapper code validated.

Attempt #1 Reducer #2:

Input: Output of Mapper #2. UserA (tab) UserB (tab) Similarity

Input key: UserA

Input value: UserB (tab) Similarity

Output key: UserA:

Output value: User1 (space) User2 (space) User3 (space)

Each of these users are the three users that are most similar to the user.

Note: I no longer have the code for this Reducer, but I created a list of user2's and a list of similarity, read the input in, and use the zip function from [29] and [30] to sort the similarity and users lists in sync so that even when the similarity was sorted in descending order, the user2 list would sort the same way, even if it wasn't sorted numerically to begin with.

Issues / Design Considerations before the final attempts:

The problem was that the top 3 users output for each user didn't actually have the highest similarity among all pairs. I was running out of time to debug / fix any errors in the code, so I

scaled back the ambitions of my mapper and reducer design. Another issue I had is that even for the small dataset, the MapReduce command took 12 minutes and 20 seconds, which is too much. The large dataset usually takes at least 100x as long to run commands, if not higher, so that would take at least 20 hours. I wasn't sure where the bottleneck was.

Instead of calculating the similarity in the mapper and the top 3 users in the reducer, I would output the users and count variables, as originally planned, then use the reducer to calculate the similarity, and just output the similarity of all pairs. The assignment 1 PDF says using a sort command is allowed after calculating the similarity of all pairs, but I didn't think this would be too useful at first, because even if I sorted in order of highest similarity, there would be a random scattering of users, and I would need to search through manually to create the top pairs for each user. I later learned in [31] that it would be possible to use the unix sort command to filter on multiple columns. I detail this later.

Despite changing the mapper and reducer functions, it was still just as slow. I observed the bottleneck was in the mapper, not the reducer, with the large dataset not able to get a single percent on the mapper after even 15 minutes, so I killed the job. The most likely culprit was lines 33-47 in the mapper, both attempts. For every pair the similarity is being calculated, the mapper code is having it run through 2 for loops to search through an array of random users until it finds the user corresponding to it. This is ok for the small database where there may be 300 comparisons on average twice for a 600 user database for 600 comparisons per calculated pair. The large dataset has around 70,000 users, so 70,000 comparisons per pair would be at least 116x worse.

Finally remembering my data structure class from years ago, I modified the code to use a dictionary instead of iterating through the array of users twice each for loop to provide constant time lookup after the initial process for the mapper of loading the file and adding to the dictionary. Even with these modifications, the small dataset took the same amount of time with 350 mappers and 14 reducers (around 12 minutes), maybe because of the overhead of splitting the small input file 350 ways and needing to load the IndividualCountSmall file in 350 mappers + generate the dictionary for each one is not worth it for such a small dataset. The coordination / overhead / memory consumption involved may be choking things.

Final Part B Mapper #2 Code:

On next page.

SimilarityMapper.py 1

C: > Users > Eddie > Documents > School > CUHK > 2021 - 2022 Term 2 > IEMS 5730 - Big Data Systems and Information Proc

```python
#!/usr/bin/python
from operator import itemgetter
import sys

IndividualCount = {}

# Open IndividualCount file and create dictionary of users and their individual counts.
with open('IndividualCountSmall') as f:
    for line in f.readlines():
        line = line.strip()

        # Append user and count from each file line to each list.
        single_user, single_count = line.split('\t', 1)
        IndividualCount[single_user] = single_count

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

    # Parse the input we got from mapper.
    # Input comes in form of user_pair (tab) count
    users, pair_count = line.split('\t', 1)

    #Here we split up the comma divided pairs into individual users
    user1, user2 = users.split(',', 1)

    # Search through IndividualCount dictionary to find the
    # individual counts for each user.

    count1 = IndividualCount.get(user1)
    count2 = IndividualCount.get(user2)

    # Print User1 (tab) User2 (tab) PairCount (tab) User1Count (tab) User2Count (tab)
    print "%s\t%s\t%s\t%s\t%s" % (user1, user2, pair_count, count1, count2)
```

Final Part B Reducer #2 Code:

On next page.

SimilarityReducer.py 2

C: > Users > Eddie > Documents > School > CUHK > 2021 - 2022 Term 2 > IEMS 5730 - Big Data

```python
#!/usr/bin/python
from operator import itemgetter
import sys

# We first initialize the variables as None.

current_user = None
user1 = None
user2 = None
pair_count = None
count1 = None
count2 = None

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

    # Parse the input we got from mapper.
    # Inputs are tab separated
    user1, user2, pair_count, count1, count2  = line.split('\t')

    # Convert variables to float. (currently string) to float

    try:
        pair_count = float(pair_count)
    except ValueError:
        # Was not a number, so silently ignore/discard the line
        continue

    try:
        count1 = float(count1)
    except ValueError:
        # Was not a number, so silently ignore/discard the line
        continue
```

```
        # Was not a number, so silently ignore/discard the line
        continue

    try:
        count2 = float(count2)
    except ValueError:
        # Was not a number, so silently ignore/discard the line
        continue

    # Calculate the similarity.
    # The equation for similarity is (Number of movies both pairs have watched) / (User A movies + User B movies - number of movies both pairs have watched)
    # The numerator is pair_count.
    # The denominator is count1 + count2 - pair_count
    # Print user pair and all three counts (pair and inidividual)

    denominator = count1 + count2 - pair_count
    similarity = pair_count / denominator

    # Print the users and similarity.
    # We print twice so that each user can have a copy of the similarity of the pair under their primary key (when sorting)

    print "%s\t%s\t%.3f" % (user1, user2, similarity)
    print "%s\t%s\t%.3f" % (user2, user1, similarity)
```

I later realized that I didn't need current_user variable since I was no longer operating on pairs in an order, but this is the code I ran the MapReduce with.

Small dataset command:

"hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar\

-D mapreduce.map.output.compress=true -D mapreduce.output.fileoutputformat.compress=false\

-D mapreduce.job.reduces=7\

-file IndividualCountSmall\

-file SimilarityMapper.py -mapper SimilarityMapper.py\

-file SimilarityReducer.py -reducer SimilarityReducer.py\

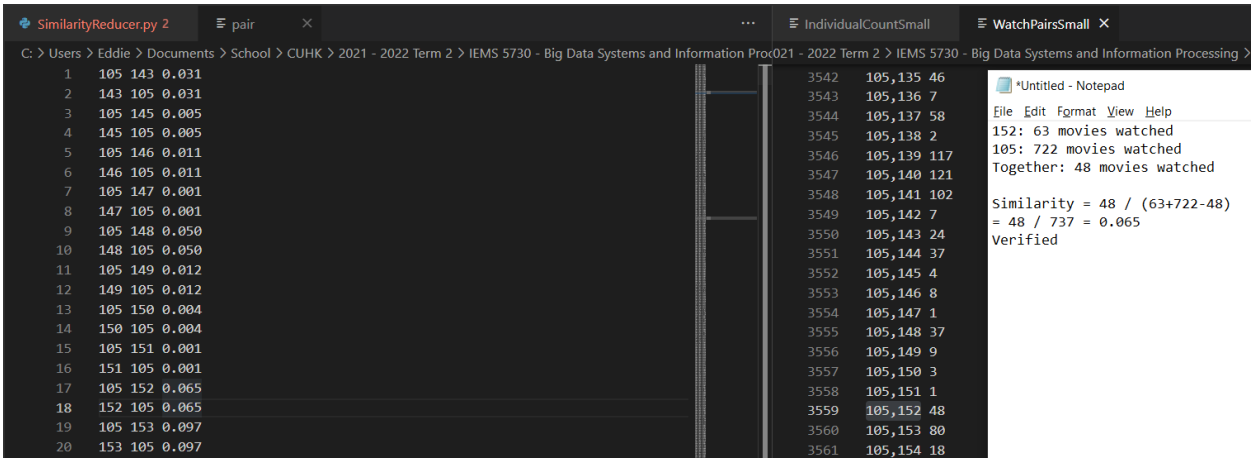
-input WatchPairsSmall\

-output PairSimilaritySmall"

This time, I ran with the reducers set to 7 instead of 14, and didn't specify the mappers so that MapReduce could figure out how many mappers it wanted to run with on its own. It ended up using 2 mappers and finished in only 24 seconds. I attribute this to a combination of changing the for loops iterating through an array to a dictionary structure, and for the vast reduction in the number of mappers, which was too high before.

```
[s1155160788@dicvmd10 input]$ hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streami
ng.jar -D mapreduce.map.output.compress=true -D mapreduce.output.fileoutputformat.compress=fa
lse -D mapreduce.job.reduces=7 -file IndividualCountSmall -file SimilarityMapper.py -mapper S
imilarityMapper.py -file SimilarityReducer.py -reducer SimilarityReducer.py -input WatchPairs
Small -output PairSimilaritySmall
22/02/16 07:42:29 WARN streaming.StreamJob: -file option is deprecated, please use generic op
tion -files instead.
packageJobJar: [IndividualCountSmall, SimilarityMapper.py, SimilarityReducer.py] [/usr/hdp/2.
6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar] /tmp/streamjob590059646868
5525526.jar tmpDir=null
22/02/16 07:42:30 INFO client.AHSProxy: Connecting to Application History server at dicvmd2.i
e.cuhk.edu.hk/172.16.5.202:10200
22/02/16 07:42:30 INFO client.AHSProxy: Connecting to Application History server at dicvmd2.i
e.cuhk.edu.hk/172.16.5.202:10200
22/02/16 07:42:30 INFO client.RequestHedgingRMFailoverProxyProvider: Looking for the active R
M in [rm1, rm2]...
22/02/16 07:42:30 INFO client.RequestHedgingRMFailoverProxyProvider: Found active RM [rm2]
22/02/16 07:42:31 INFO mapred.FileInputFormat: Total input paths to process : 1
22/02/16 07:42:31 INFO mapreduce.JobSubmitter: number of splits:2
22/02/16 07:42:31 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1644821811052_0
248
22/02/16 07:42:31 INFO impl.YarnClientImpl: Submitted application application_1644821811052_0
248
22/02/16 07:42:31 INFO mapreduce.Job: The url to track the job: http://dicvmd1.ie.cuhk.edu.hk
:8088/proxy/application_1644821811052_0248/
22/02/16 07:42:31 INFO mapreduce.Job: Running job: job_1644821811052_0248
22/02/16 07:42:35 INFO mapreduce.Job: Job job_1644821811052_0248 running in uber mode : false
22/02/16 07:42:35 INFO mapreduce.Job:  map 0% reduce 0%
22/02/16 07:42:39 INFO mapreduce.Job:  map 50% reduce 0%
22/02/16 07:42:42 INFO mapreduce.Job:  map 100% reduce 0%
22/02/16 07:42:45 INFO mapreduce.Job:  map 100% reduce 14%
22/02/16 07:42:47 INFO mapreduce.Job:  map 100% reduce 29%
22/02/16 07:42:49 INFO mapreduce.Job:  map 100% reduce 43%
22/02/16 07:42:51 INFO mapreduce.Job:  map 100% reduce 57%
22/02/16 07:42:53 INFO mapreduce.Job:  map 100% reduce 71%
22/02/16 07:42:55 INFO mapreduce.Job:  map 100% reduce 86%
22/02/16 07:42:57 INFO mapreduce.Job:  map 100% reduce 100%
22/02/16 07:42:57 INFO mapreduce.Job: Job job_1644821811052_0248 completed successfully
22/02/16 07:42:57 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=935057
                FILE: Number of bytes written=3270182
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1790554
                HDFS: Number of bytes written=4475984
                HDFS: Number of read operations=27
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=14
```

```
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=7
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=4378
                Total time spent by all reduces in occupied slots (ms)=26112
                Total time spent by all map tasks (ms)=4378
                Total time spent by all reduce tasks (ms)=13056
                Total vcore-milliseconds taken by all map tasks=4378
                Total vcore-milliseconds taken by all reduce tasks=13056
                Total megabyte-milliseconds taken by all map tasks=286916608
                Total megabyte-milliseconds taken by all reduce tasks=1711276032
        Map-Reduce Framework
                Map input records=164054
                Map output records=164054
                Map output bytes=2797541
                Map output materialized bytes=934840
                Input split bytes=238
                Combine input records=0
                Combine output records=0
                Reduce input groups=609
                Reduce shuffle bytes=934840
                Reduce input records=164054
                Reduce output records=328108
                Spilled Records=328108
                Shuffled Maps =14
                Failed Shuffles=0
                Merged Map outputs=14
                GC time elapsed (ms)=455
                CPU time spent (ms)=14830
                Physical memory (bytes) snapshot=8218660864
                Virtual memory (bytes) snapshot=920785666048
                Total committed heap usage (bytes)=9942073344
        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=1790316
        File Output Format Counters
                Bytes Written=4475984
22/02/16 07:42:57 INFO streaming.StreamJob: Output directory: PairSimilaritySmall
```

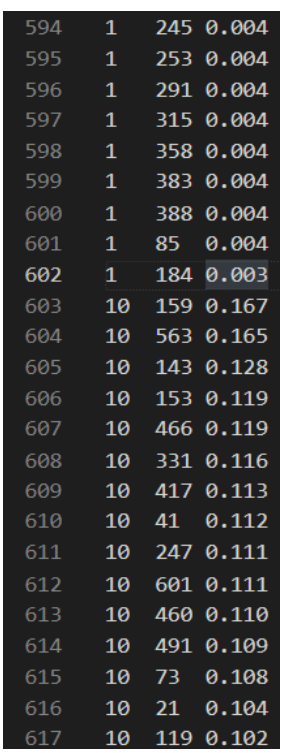


First, I verified that with the dictionary structure, the similarity is still being calculated correctly. The output from the 2nd MapReduce job of Part B is on the left, with the output of the 2nd Map Reduce job of Part A on the right. I use this and the individual count file (the first MapReduce job of part 1) to calculate the similarity manually and confirm it matches.

As mentioned earlier, I use [31] to sort this output on two different fields. The primary sorting field is the first user, then within the same user, I sort by similarity (the third field) in reverse (descending) order so it will show from highest to lowest, essentially a ranking of the similarities for each user compared to the base user.

Sort command: Sort -k1,1 -k3,3r PairsSimilarSmall > pairSimilaritySmallSorted

Sorted output example (small dataset):



594	1	245	0.004
595	1	253	0.004
596	1	291	0.004
597	1	315	0.004
598	1	358	0.004
599	1	383	0.004
600	1	388	0.004
601	1	85	0.004
602	1	184	0.003
603	10	159	0.167
604	10	563	0.165
605	10	143	0.128
606	10	153	0.119
607	10	466	0.119
608	10	331	0.116
609	10	417	0.113
610	10	41	0.112
611	10	247	0.111
612	10	601	0.111
613	10	460	0.110
614	10	491	0.109
615	10	73	0.108
616	10	21	0.104
617	10	119	0.102

The leftmost number is the line number, followed by the first user, the second user, then the calculated similarity. We can see that the output looks reasonable. Every line from 1-602 is user 1 before it moves on to user 10. We can also see that the similarity is highest for the first entries of a user and lowest at the end of their entries. Although this database has 610 users, it is reasonable that 7 pairs out of 609 did not have a movie in common with user 1.

I save writing the top 10 users for the final submission section for Part B.

Large dataset command:

“hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar\

-D mapreduce.map.output.compress=true -D mapreduce.output.fileoutputformat.compress=false\

-D mapreduce.job.reduces=7\

-file IndividualCountLarge\

-file SimilarityMapper.py -mapper SimilarityMapper.py\

-file SimilarityReducer.py -reducer SimilarityReducer.py\

-input WatchPairsLarge\

-output PairSimilarityLarge”

```
[s1155160788@dicvmd10 input]$ hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar -D mapreduce.map.output.compress=true -D mapreduce.output.fileoutputformat.compress=false
 -D mapreduce.job.reduces=7 -file IndividualCountLarge -file SimilarityMapper.py -mapper SimilarityMapper.py -file SimilarityReducer.py -reducer SimilarityReducer.py -input WatchPairsLarge
-output PairSimilarityLarge
22/02/15 20:34:43 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [IndividualCountLarge, SimilarityMapper.py, SimilarityReducer.py] [/usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar] /tmp/streamjob796069545957702
7787.jar tmpDir=null
22/02/15 20:34:43 INFO client.AHSProxy: Connecting to Application History server at dicvmd2.ie.cuhk.edu.hk/172.16.5.202:10200
22/02/15 20:34:43 INFO client.AHSProxy: Connecting to Application History server at dicvmd2.ie.cuhk.edu.hk/172.16.5.202:10200
22/02/15 20:34:44 INFO client.RequestHedgingRMFailoverProxyProvider: Looking for the active RM in [rm1, rm2]...
22/02/15 20:34:44 INFO client.RequestHedgingRMFailoverProxyProvider: Found active RM [rm2]
22/02/15 20:34:44 INFO mapred.FileInputFormat: Total input paths to process : 1
22/02/15 20:34:44 INFO net.NetworkTopology: Adding a new node: /default-rack/172.16.5.210:50010
22/02/15 20:34:44 INFO net.NetworkTopology: Adding a new node: /default-rack/172.16.5.204:50010
22/02/15 20:34:44 INFO net.NetworkTopology: Adding a new node: /default-rack/172.16.5.206:50010
22/02/15 20:34:44 INFO net.NetworkTopology: Adding a new node: /default-rack/172.16.5.207:50010
22/02/15 20:34:44 INFO mapreduce.JobSubmitter: number of splits:14
22/02/15 20:34:44 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1644821811052_0152
22/02/15 20:34:44 INFO impl.YarnClientImpl: Submitted application application_1644821811052_0152
22/02/15 20:34:44 INFO mapreduce.Job: The url to track the job: http://dicvmd1.ie.cuhk.edu.hk:8088/proxy/application_1644821811052_0152/
22/02/15 20:34:44 INFO mapreduce.Job: Running job: job_1644821811052_0152
22/02/15 20:34:48 INFO mapreduce.Job: Job job_1644821811052_0152 running in uber mode : false
22/02/15 20:34:48 INFO mapreduce.Job:  map 0% reduce 0%
22/02/15 20:34:58 INFO mapreduce.Job:  map 1% reduce 0%
22/02/15 20:35:01 INFO mapreduce.Job:  map 2% reduce 0%
22/02/15 20:35:04 INFO mapreduce.Job:  map 3% reduce 0%
22/02/15 20:35:10 INFO mapreduce.Job:  map 4% reduce 0%
22/02/15 20:35:13 INFO mapreduce.Job:  map 5% reduce 0%
22/02/15 20:35:31 INFO mapreduce.Job:  map 7% reduce 0%
22/02/15 20:35:40 INFO mapreduce.Job:  map 8% reduce 0%
22/02/15 20:35:43 INFO mapreduce.Job:  map 9% reduce 0%
22/02/15 20:35:46 INFO mapreduce.Job:  map 10% reduce 0%
22/02/15 20:35:52 INFO mapreduce.Job:  map 11% reduce 0%
22/02/15 20:35:55 INFO mapreduce.Job:  map 12% reduce 0%
22/02/15 20:36:13 INFO mapreduce.Job:  map 14% reduce 0%
22/02/15 20:36:21 INFO mapreduce.Job:  map 16% reduce 0%
22/02/15 20:36:27 INFO mapreduce.Job:  map 17% reduce 0%
22/02/15 20:36:30 INFO mapreduce.Job:  map 18% reduce 0%
22/02/15 20:36:37 INFO mapreduce.Job:  map 19% reduce 0%
22/02/15 20:36:55 INFO mapreduce.Job:  map 21% reduce 0%
22/02/15 20:37:02 INFO mapreduce.Job:  map 23% reduce 0%
22/02/15 20:37:08 INFO mapreduce.Job:  map 24% reduce 0%
22/02/15 20:37:11 INFO mapreduce.Job:  map 25% reduce 0%
22/02/15 20:37:17 INFO mapreduce.Job:  map 26% reduce 0%
22/02/15 20:37:36 INFO mapreduce.Job:  map 29% reduce 0%
22/02/15 20:37:43 INFO mapreduce.Job:  map 30% reduce 0%
22/02/15 20:37:49 INFO mapreduce.Job:  map 31% reduce 0%
22/02/15 20:37:52 INFO mapreduce.Job:  map 32% reduce 0%
22/02/15 20:37:55 INFO mapreduce.Job:  map 33% reduce 0%
22/02/15 20:38:18 INFO mapreduce.Job:  map 36% reduce 0%
22/02/15 20:38:26 INFO mapreduce.Job:  map 37% reduce 0%
```

```
22/02/15 20:38:29 INFO mapreduce.Job:  map 38% reduce 0%
22/02/15 20:38:35 INFO mapreduce.Job:  map 39% reduce 0%
22/02/15 20:38:38 INFO mapreduce.Job:  map 40% reduce 0%
22/02/15 20:39:02 INFO mapreduce.Job:  map 43% reduce 0%
22/02/15 20:39:09 INFO mapreduce.Job:  map 44% reduce 0%
22/02/15 20:39:12 INFO mapreduce.Job:  map 45% reduce 0%
22/02/15 20:39:18 INFO mapreduce.Job:  map 46% reduce 0%
22/02/15 20:39:21 INFO mapreduce.Job:  map 47% reduce 0%
22/02/15 20:39:27 INFO mapreduce.Job:  map 48% reduce 0%
22/02/15 20:39:44 INFO mapreduce.Job:  map 50% reduce 0%
22/02/15 20:39:51 INFO mapreduce.Job:  map 51% reduce 0%
22/02/15 20:39:54 INFO mapreduce.Job:  map 52% reduce 0%
22/02/15 20:40:00 INFO mapreduce.Job:  map 53% reduce 0%
22/02/15 20:40:03 INFO mapreduce.Job:  map 54% reduce 0%
22/02/15 20:40:09 INFO mapreduce.Job:  map 55% reduce 0%
22/02/15 20:40:28 INFO mapreduce.Job:  map 57% reduce 0%
22/02/15 20:40:36 INFO mapreduce.Job:  map 58% reduce 0%
22/02/15 20:40:39 INFO mapreduce.Job:  map 59% reduce 0%
22/02/15 20:40:42 INFO mapreduce.Job:  map 60% reduce 0%
22/02/15 20:40:48 INFO mapreduce.Job:  map 61% reduce 0%
22/02/15 20:40:51 INFO mapreduce.Job:  map 62% reduce 0%
22/02/15 20:41:10 INFO mapreduce.Job:  map 64% reduce 0%
22/02/15 20:41:18 INFO mapreduce.Job:  map 66% reduce 0%
22/02/15 20:41:24 INFO mapreduce.Job:  map 67% reduce 0%
22/02/15 20:41:27 INFO mapreduce.Job:  map 68% reduce 0%
22/02/15 20:41:33 INFO mapreduce.Job:  map 69% reduce 0%
22/02/15 20:41:51 INFO mapreduce.Job:  map 71% reduce 0%
22/02/15 20:41:59 INFO mapreduce.Job:  map 73% reduce 0%
22/02/15 20:42:05 INFO mapreduce.Job:  map 74% reduce 0%
22/02/15 20:42:08 INFO mapreduce.Job:  map 75% reduce 0%
22/02/15 20:42:14 INFO mapreduce.Job:  map 76% reduce 0%
22/02/15 20:42:33 INFO mapreduce.Job:  map 79% reduce 0%
22/02/15 20:42:41 INFO mapreduce.Job:  map 80% reduce 0%
22/02/15 20:42:47 INFO mapreduce.Job:  map 81% reduce 0%
22/02/15 20:42:50 INFO mapreduce.Job:  map 82% reduce 0%
22/02/15 20:42:56 INFO mapreduce.Job:  map 83% reduce 0%
22/02/15 20:43:16 INFO mapreduce.Job:  map 86% reduce 0%
22/02/15 20:43:23 INFO mapreduce.Job:  map 87% reduce 0%
22/02/15 20:43:26 INFO mapreduce.Job:  map 88% reduce 0%
22/02/15 20:43:32 INFO mapreduce.Job:  map 89% reduce 0%
22/02/15 20:43:35 INFO mapreduce.Job:  map 90% reduce 0%
22/02/15 20:43:57 INFO mapreduce.Job:  map 93% reduce 0%
22/02/15 20:44:04 INFO mapreduce.Job:  map 94% reduce 0%
22/02/15 20:44:07 INFO mapreduce.Job:  map 95% reduce 0%
22/02/15 20:44:13 INFO mapreduce.Job:  map 96% reduce 0%
22/02/15 20:44:16 INFO mapreduce.Job:  map 97% reduce 0%
22/02/15 20:44:19 INFO mapreduce.Job:  map 98% reduce 0%
22/02/15 20:44:37 INFO mapreduce.Job:  map 100% reduce 0%
22/02/15 20:44:45 INFO mapreduce.Job:  map 100% reduce 6%
```

```
22/02/15 20:44:37 INFO mapreduce.Job:  map 100% reduce 0%
22/02/15 20:44:45 INFO mapreduce.Job:  map 100% reduce 6%
22/02/15 20:44:51 INFO mapreduce.Job:  map 100% reduce 7%
22/02/15 20:44:58 INFO mapreduce.Job:  map 100% reduce 8%
22/02/15 20:45:01 INFO mapreduce.Job:  map 100% reduce 9%
22/02/15 20:45:07 INFO mapreduce.Job:  map 100% reduce 10%
22/02/15 20:45:25 INFO mapreduce.Job:  map 100% reduce 11%
22/02/15 20:45:40 INFO mapreduce.Job:  map 100% reduce 12%
22/02/15 20:45:55 INFO mapreduce.Job:  map 100% reduce 13%
22/02/15 20:46:13 INFO mapreduce.Job:  map 100% reduce 14%
22/02/15 20:46:32 INFO mapreduce.Job:  map 100% reduce 20%
22/02/15 20:46:35 INFO mapreduce.Job:  map 100% reduce 21%
22/02/15 20:46:41 INFO mapreduce.Job:  map 100% reduce 22%
22/02/15 20:46:47 INFO mapreduce.Job:  map 100% reduce 23%
22/02/15 20:46:53 INFO mapreduce.Job:  map 100% reduce 24%
22/02/15 20:47:05 INFO mapreduce.Job:  map 100% reduce 25%
22/02/15 20:47:23 INFO mapreduce.Job:  map 100% reduce 26%
22/02/15 20:47:38 INFO mapreduce.Job:  map 100% reduce 27%
22/02/15 20:47:56 INFO mapreduce.Job:  map 100% reduce 28%
22/02/15 20:48:11 INFO mapreduce.Job:  map 100% reduce 29%
22/02/15 20:48:20 INFO mapreduce.Job:  map 100% reduce 34%
22/02/15 20:48:23 INFO mapreduce.Job:  map 100% reduce 35%
22/02/15 20:48:29 INFO mapreduce.Job:  map 100% reduce 36%
22/02/15 20:48:32 INFO mapreduce.Job:  map 100% reduce 37%
22/02/15 20:48:38 INFO mapreduce.Job:  map 100% reduce 38%
22/02/15 20:48:47 INFO mapreduce.Job:  map 100% reduce 39%
22/02/15 20:49:06 INFO mapreduce.Job:  map 100% reduce 40%
22/02/15 20:49:21 INFO mapreduce.Job:  map 100% reduce 41%
22/02/15 20:49:39 INFO mapreduce.Job:  map 100% reduce 42%
22/02/15 20:49:54 INFO mapreduce.Job:  map 100% reduce 43%
22/02/15 20:50:09 INFO mapreduce.Job:  map 100% reduce 49%
22/02/15 20:50:15 INFO mapreduce.Job:  map 100% reduce 50%
22/02/15 20:50:21 INFO mapreduce.Job:  map 100% reduce 51%
22/02/15 20:50:27 INFO mapreduce.Job:  map 100% reduce 52%
22/02/15 20:50:33 INFO mapreduce.Job:  map 100% reduce 53%
22/02/15 20:50:51 INFO mapreduce.Job:  map 100% reduce 54%
22/02/15 20:51:06 INFO mapreduce.Job:  map 100% reduce 55%
22/02/15 20:51:24 INFO mapreduce.Job:  map 100% reduce 56%
22/02/15 20:51:42 INFO mapreduce.Job:  map 100% reduce 57%
22/02/15 20:51:59 INFO mapreduce.Job:  map 100% reduce 63%
22/02/15 20:52:05 INFO mapreduce.Job:  map 100% reduce 64%
22/02/15 20:52:08 INFO mapreduce.Job:  map 100% reduce 65%
22/02/15 20:52:14 INFO mapreduce.Job:  map 100% reduce 66%
22/02/15 20:52:20 INFO mapreduce.Job:  map 100% reduce 67%
22/02/15 20:52:35 INFO mapreduce.Job:  map 100% reduce 68%
22/02/15 20:52:53 INFO mapreduce.Job:  map 100% reduce 69%
22/02/15 20:53:08 INFO mapreduce.Job:  map 100% reduce 70%
22/02/15 20:53:26 INFO mapreduce.Job:  map 100% reduce 71%
22/02/15 20:53:48 INFO mapreduce.Job:  map 100% reduce 77%
22/02/15 20:53:51 INFO mapreduce.Job:  map 100% reduce 78%
```

```
22/02/15 20:53:48 INFO mapreduce.Job:  map 100% reduce 77%
22/02/15 20:53:51 INFO mapreduce.Job:  map 100% reduce 78%
22/02/15 20:53:57 INFO mapreduce.Job:  map 100% reduce 79%
22/02/15 20:54:03 INFO mapreduce.Job:  map 100% reduce 80%
22/02/15 20:54:06 INFO mapreduce.Job:  map 100% reduce 81%
22/02/15 20:54:18 INFO mapreduce.Job:  map 100% reduce 82%
22/02/15 20:54:36 INFO mapreduce.Job:  map 100% reduce 83%
22/02/15 20:54:51 INFO mapreduce.Job:  map 100% reduce 84%
22/02/15 20:55:09 INFO mapreduce.Job:  map 100% reduce 85%
22/02/15 20:55:27 INFO mapreduce.Job:  map 100% reduce 86%
22/02/15 20:55:37 INFO mapreduce.Job:  map 100% reduce 91%
22/02/15 20:55:40 INFO mapreduce.Job:  map 100% reduce 92%
22/02/15 20:55:43 INFO mapreduce.Job:  map 100% reduce 93%
22/02/15 20:55:49 INFO mapreduce.Job:  map 100% reduce 94%
22/02/15 20:55:55 INFO mapreduce.Job:  map 100% reduce 95%
22/02/15 20:56:01 INFO mapreduce.Job:  map 100% reduce 96%
22/02/15 20:56:19 INFO mapreduce.Job:  map 100% reduce 97%
22/02/15 20:56:34 INFO mapreduce.Job:  map 100% reduce 98%
22/02/15 20:56:49 INFO mapreduce.Job:  map 100% reduce 99%
22/02/15 20:57:07 INFO mapreduce.Job:  map 100% reduce 100%
22/02/15 20:57:13 INFO mapreduce.Job: Job job_1644821811052_0152 completed successfully
22/02/15 20:57:13 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=784877012
                FILE: Number of bytes written=1645244656
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1891079354
                HDFS: Number of bytes written=4867576072
                HDFS: Number of read operations=63
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=14
        Job Counters
                Launched map tasks=14
                Launched reduce tasks=7
                Data-local map tasks=14
                Total time spent by all maps in occupied slots (ms)=581597
                Total time spent by all reduces in occupied slots (ms)=1505298
                Total time spent by all map tasks (ms)=581597
                Total time spent by all reduce tasks (ms)=752649
                Total vcore-milliseconds taken by all map tasks=581597
                Total vcore-milliseconds taken by all reduce tasks=752649
                Total megabyte-milliseconds taken by all map tasks=38115540992
                Total megabyte-milliseconds taken by all reduce tasks=98651209728
        Map-Reduce Framework
                Map input records=137542274
                Map output records=137542274
                Map output bytes=2733834938
                Map output materialized bytes=857098388
```

```
        Map-Reduce Framework
                Map input records=137542274
                Map output records=137542274
                Map output bytes=2733834938
                Map output materialized bytes=857098388
                Input split bytes=1666
                Combine input records=0
                Combine output records=0
                Reduce input groups=53141
                Reduce shuffle bytes=857098388
                Reduce input records=137542274
                Reduce output records=275084548
                Spilled Records=275084548
                Shuffled Maps =98
                Failed Shuffles=0
                Merged Map outputs=98
                GC time elapsed (ms)=5628
                CPU time spent (ms)=1628530
                Physical memory (bytes) snapshot=56770469888
                Virtual memory (bytes) snapshot=1616260222976
                Total committed heap usage (bytes)=67670376448
        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=1891077688
        File Output Format Counters
                Bytes Written=4867576072
22/02/15 20:57:13 INFO streaming.StreamJob: Output directory: PairSimilarityLarge
```

Application application_1644821811052_0152

User:	s1155160788
Name:	streamjob7960695459577027787.jar
Application Type:	MAPREDUCE
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Tue Feb 15 20:34:44 +0800 2022
Elapsed:	22mins, 28sec
Tracking URL:	History
Log Aggregation Status	SUCCEEDED
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	<DEFAULT_PARTITION>

The job completes successfully in 22 minutes and 28 seconds.

```
[s1155160788@dicvmd10 output]$ cat PairsSimilarLarge | head -n 10
10000   53522   0.006
53522   10000   0.006
10000   53935   0.006
53935   10000   0.006
10000   56707   0.001
56707   10000   0.001
10000   57120   0.014
57120   10000   0.014
10000   57470   0.008
57470   10000   0.008
[s1155160788@dicvmd10 output]$ sort -k1,1 -k3,3r PairsSimilarLarge >  pairSimilarityLargeSorted
[s1155160788@dicvmd10 output]$ ls
IndividualCountLarge  MovieUsersLarge  pairSimilarityLargeSorted  PairsSimilarLarge  top10WatchSmall  WatchPairsSmall
IndividualCountSmall  MovieUsersSmall  pairSimilaritySmallSorted  top10WatchLarge    WatchPairsLarge
[s1155160788@dicvmd10 output]$ cat pairSimilarityLargeSorted | head -n 20
10      1551    0.167
10      48516   0.158
10      11493   0.154
10      65790   0.154
10      35210   0.143
10      37074   0.143
10      40752   0.143
10      50309   0.133
10      753     0.133
10      19592   0.125
10      31611   0.125
10      3707    0.125
10      56682   0.125
10      59397   0.125
10      18964   0.121
10      65491   0.120
10      10028   0.118
10      13530   0.118
10      19327   0.118
10      2191    0.118
[s1155160788@dicvmd10 output]$
```

I use the same sorting command as before “sort -k1,1 -k3,3r” to sort the large pairs, using cat + head commands before and after to confirm the base output and some of the sorted version. The sorted output is 4.753 GB large, so I may need a specialized editor to extract the information.

Part B Final Submission: Top 3 Similar Users

My student ID is 1155160788.

Small Dataset: Section I

The last two digits of my student ID is 88. Users in the small dataset I need to find the top 3 similar users for are 88, 188, 288, 388, 488, 588.

```
88  581 0.157
88  378 0.156
88  247 0.151
88  77  0.149
88  460 0.140
88  417 0.139
88  22  0.136
88  189 0.134
88  61  0.131
```

As an example of my sorted output, the top users for 88 would be 581, 378, and 247. I am manually seeking through my similarity pair list for the small dataset and putting it into a text document that follows the format requested.

Line number (tab) User: User1, User2, User3, in order of similarity in descending order.

top10SimilaritySmall

C: > Users > Eddie > Documents >

```
088: 581, 378, 247
188: 554, 412, 587
288: 590, 480, 68
388: 539, 49, 598
488: 263, 167, 137
588: 126, 56, 379
```

Large Dataset: Section II

The last four digits of my student ID is 0788. Users in the large dataset I need to find the top 3 similar users for are 788 (00788), 10788, 20788, 30788, 40788, 50788, 60788, and 70788.

I was unable to open up the file in Visual Studio Code on my laptop, so instead I opened it with a free editor for Windows called PilotEdit Lite.

```
265352695 788 53822   0.175
265352696 788 30449   0.170
265352697 788 25907   0.165
265352698 788 249 0.164
265352699 788 11646   0.158
265352700 788 5110    0.157
265352701 788 22899   0.156
265352702 788 25887   0.156
265352703 788 56620   0.156
265352704 788 1908    0.153
265352705 788 3860    0.150
265352706 788 44129   0.150
265352707 788 27852   0.149
265352708 788 27662   0.148
265352709 788 66874   0.148
```

Like with the small dataset in Section I, I am seeking through the list and manually putting the top 3 similar users in another file with the desired format, then screenshotting that.

Note: User 10788 does not exist in the database, it goes from 10786 to 10790. This can be seen both in my MapReduce output and the original dataset.

```
224990    10786,4860
224991    10786,4861
224992    10786,4862
224993    10786,4864
224994    10786,5717
224995    10790,35
224996    10790,97
224997    10790,214
224998    10790,617
224999    10790,1053
```

I offer you user 10790 instead.

50788 also does not exist so I am doing user 50789 instead. 70788 does not exist so I am doing user 70789 instead.

```
top10SimilarityLarge ×
2022 Term 2 > IEMS 5730 - Big Data Systems and Information Processing > We
1    788: 53822, 30449, 25907
2    10790 (10788 does not exist): 5152, 37500, 44521
3    20788: 178, 2144, 23991
4    30788: 13358, 17031, 17430
5    40788: 19514, 24218, 25984
6    50789 (50788 does not exist): 1740, 17793, 41871
7    60788: 10077, 27075, 27167
8    70789 (70788 does not exist): 15418, 54351, 924
```

Part C: Running large dataset on Part A multiple times with different number of mappers and reducers

For this part, we need to run Part A 3 times. I will reuse the data from my first run, but I need to run the first MapReduce job with 350 mappers and 14 reducers for consistency with the second part.

The table of my results are below:

Note: MR Job = MapReduce Job, not the MrJob python library.

Note 2: I am determining the total runtime by the cluster webpage, but the mapper and reduce times by the “Total time spent by all map tasks” and “Total time spent by all reducer tasks” in the console output, which won’t exactly add up to the total runtime.

Base data: (time are all in seconds).

Run	# of mappers	# of reducers	MR Job #1 (runtime)	Mapper 1 Runtime	Reducer 1 Runtime	MR Job #2 (runtime)	Mapper 2 Runtime	Reducer 2 Runtime	Total runtime
#1	350	14	754	652	28.5	3146	2110	835	3900
#2	14	14	74	29.4	27.3	2147	1457	670	2221
#3	70	2	153	133.7	7	2292	1491.6	756	2445

Requested data:

Maximum mapper time (Job #1)	Minimum mapper time (Job #1)	Average mapper time (Job#1)	Maximum reducer time (Job#1)	Minimum reducer time (Job#1)	Average reducer time (Job#1)
652	29.4	271.7	28.5	7	20.93

Maximum mapper time (Job #2)	Minimum mapper time (Job #2)	Average mapper time (Job#2)	Maximum reducer time (Job#2)	Minimum reducer time (Job#2)	Average reducer time (Job#2)
2110	1457	1686.2	835	670	753.67

Total job runtimes are in the rightmost column of the base data.

```
22/02/16 09:55:55 INFO mapreduce.Job:  map 20% reduce 0%
22/02/16 09:56:34 INFO mapreduce.Job: Task Id : attempt_1644821811052_0252_m_000000_0, Status : FAILED
Error: java.lang.RuntimeException: PipeMapRed.waitOutputThreads(): subprocess failed with code 1
```

I tried running the second MapReduce job of Part A with just 2 mappers, as that was the default amount of mappers that run if you don't specify anything, but I got errors around 20% of the mapping. The likely reason for this is that 1-2 nodes (can't be more than 2 with 2 mappers) do not have enough allocated memory for my job to handle storing all the intermediate output alone. After trying at least 7 mappers to make sure each of the 7 active nodes on the IE DIC cluster were working on it, it worked.

Explanation:

We can see that there is considerably more variance in the mapper time than the reducer time as we adjust the amount of mappers. As long as the job is able to run without erroring (as in when I tried with just 2 mappers), it seems there is a performance increase to running fewer mappers, as in both MapReduce jobs, the less mappers there were, the faster the runtime.

There also appears to be diminishing returns beyond a certain point of reduction. There is not much difference in performance between 14 and 70 mappers, but a considerable difference between 70 and 350 mappers. This is likely due to performance remaining acceptable until a certain number of mappers and the resulting overhead, contention for shared resources, and

coordination chokes the workflow. 350 mappers is clearly past this point, at least for this cluster setup, as we also saw on Part B of the assignment, where changing the number of mappers from 350 to the default of 2 reduced the runtime of the small dataset job from over 12 minutes to less than 30 seconds.

References:

[1] https://mobitec.ie.cuhk.edu.hk/ierg4300Fall2020/tutorial/tutorial5/

[2] https://mrjob.readthedocs.io/en/latest/

[3] https://stackoverflow.com/questions/13923282/using-mapreduce-to-find-common-items-between-users/13924727

[4] https://techmytalk.com/2014/11/14/mapreduce-composite-key-operation-part2/

[5] Jimmy Lin; Chris Dyer, *Data-Intensive Text Processing with MapReduce* , Morgan & Claypool, 2010.

[6] http://cs.boisestate.edu/~amit/teaching/430/handouts/mapreduce-patterns-handout.pdf

[7] https://www.inf.ed.ac.uk/teaching/courses/exc/slides/hadoop_algo.pdf

[8]https://mobitec.ie.cuhk.edu.hk/ierg4300Fall2020/tutorial/tutorial4/Tutorial4_MapReduce_examples.pdf

[9] https://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/

[10] https://unix.stackexchange.com/questions/32001/what-is-m-and-how-do-i-get-rid-of-it

[11] https://thispointer.com/python-how-to-create-an-empty-list-and-append-items-to-it/

[12] https://www.tutorialspoint.com/different-ways-to-clear-a-list-in-python

[13] https://stackoverflow.com/questions/22556449/print-a-list-of-space-separated-elements

[14] https://stackoverflow.com/questions/48692741/how-can-i-make-all-line-endings-eols-in-all-files-in-visual-studio-code-unix

[15] https://www.programiz.com/python-programming/methods/list/sort

[16] https://docs.python.org/2/library/stdtypes.html#str.split

[17] https://stackoverflow.com/questions/942543/operation-on-every-pair-of-element-in-a-list/942551#942551

[18] https://docs.python.org/2/library/itertools.html#itertools.combinations

[19] https://stackoverflow.com/questions/41680388/how-do-i-iterate-through-combinations-of-a-list

[20] https://stackoverflow.com/questions/12309976/how-do-i-convert-a-list-into-a-string-with-spaces-in-python/12309982

[21] https://www.softcover.io/read/f0b3ea72/introbigdata_book/hadoop_optimized

[22] https://en.wikipedia.org/wiki/Sort_(Unix)

[23] https://hadoop.apache.org/docs/r2.9.2/hadoop-streaming/HadoopStreaming.html

[24] https://stackoverflow.com/questions/5571156/hadoop-how-to-compress-mapper-output-but-not-the-reducer-output/5579954

[25] https://cwiki.apache.org/confluence/display/HADOOP2/HowManyMapsAndReduces

[26] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," OSDI 2004

[27] https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/mapred-default.xml

[28] https://stackoverflow.com/questions/21316968/division-in-python-2-7-and-3-3

[29] https://stackoverflow.com/questions/9764298/how-to-sort-two-lists-which-reference-each-other-in-the-exact-same-way

[30] https://stackoverflow.com/questions/36622977/sorting-one-list-based-on-the-results-of-python-in-descending-order

[31] https://stackoverflow.com/questions/41231186/sort-by-multiple-columns-in-bash