Assignment-1: MapReduce

Specification | Make Submission | Check Submission | Collect Submission

In this assignment we will be working on the processing of a Movie dataset: https://raw.githubusercontent.com/sidooms/MovieTweetings/master/latest/ratings.dat (https://raw.githubusercontent.com/sidooms/MovieTweetings/master/latest/ratings.dat)

In this dataset, each row contains a movie rating done by a user (e.g., user1 has rated *Titanic* as 10). Here is the format of the dataset: `user_id::movie_id::rating::timestamp`

In this assignment, for each pair of movies A and B, you need to find all the users who rated both movie A and B. For example, given the following dataset (for the sake of illustration we have used U and M to represent users and movies respectively in the example):

```
U1::M1::2::11111111
U2::M2::3::11111111
U2::M3::1::11111111
U3::M1::4::11111111
U4::M2::5::11111111
U5::M2::3::11111111
U5::M1::1::11111111
U5::M3::3::11111111
```

The assumption is that User and Movie names are in String format and Rating is an Integer value. You should ignore the timestamp in the Mapper.

The output of your code should be in the form as below:

```
(M1,M2) [(U5,1,3)]
(M2,M3) [(U5,3,3),(U2,3,1)]
(M1,M3) [(U5,1,3)]
```

where (M,M) shows pairs of movies, [] indicates the list of users and their ratings. For example, (U5,1,3) shows U5 has rated M1 and M2 with 1 and 3 respectively.

(please note that Your output should exactly be formatted as the output example above.)

Tips :

- You may need to implement more than one Mapper/Reducer in this assignment. You need to look at chaining in MapReduce jobs: https://stackoverflow.com/questions/38111700/chaining-of-mapreduce-jobs#answer-38113499 (https://stackoverflow.com/questions/38111700/chaining-of-mapreduce-jobs#answer-38113499)
- You also may need a self-join to find movie pairs, the reduce-side join pattern can help: https://www.edureka.co/blog/mapreduce-example-reduce-side-join/

(https://www.edureka.co/blog/mapreduce-example-reduce-side-join/)

- If the key and the values for a Mapper differ from those of Reduce, you need to set the following configurations:
  job.setMapOutputKeyClass(), job.setOutputKeyClass(), job.setMapOutputValueClass(), job.setapOutputValueClass()
- Do not set **Combiner** in this assignment (~~job.setCombiner()~~)
- If the Value for the Mapper/Reducer is a complex object, you need to implement a Writerable Interface class
- If the Key for the Mapper/Reducer is a complex object, you also need to implement a WritableComparable Interface
- You can use ArrayWritable to store Array values, but you need to implement its toString() function to be able to write the object into a text file.
- Please be aware of iterating over values inside a reducer (Iterable<MyWritable> values). When looping through the Iterable value list, each Object instance is reused internally by the reducer. So if you add them to another list, at the end of the process, all of the elements in the new list will be the same as the last object you added to the list.

Submission Guidelines

Submission Deadline:

Tuesday the 29th of October 2019 17:59

Build your Project

Get and add Hadoop dependency to your project:

1. Create a new Java project in Eclipse

2. Download the dependency to your project: https://github.com/mysilver/COMP9313/raw/master/Hadoop-Core.jar (https://github.com/mysilver/COMP9313/raw/master/Hadoop-Core.jar)

3. Right click on your project and add the Hadoop-Core.jar file to your project:

    Build-Path -> Add External Archives

Submit your Project

Your code must be included (in its entirety) in the file **AssigOne{zid}.java** . Any solution that has compilation errors will receive no more than 5 points for the entire assignment.

You need to test your file before submission, to make sure that it can be compiled/run in the terminal of CSE machines:

```
$ javac -cp ".:Hadoop-Core.jar" AssigOne{zid}.java
$ java -cp ".:Hadoop-Core.jar" AssigOne{zid}  INPUT_PATH OUTPUT_PATH
```

Log in to any CSE server (e.g. williams or wagner) and use the *give command* below to submit your solution:

```
$ give cs9313 assig1 AssigOne{zid}.java
```

where you must replace {zid} above with your own zID.

You can also submit your solution using WebCMS, or Give:

https://cgi.cse.unsw.edu.au/~give/Student/give.php (https://cgi.cse.unsw.edu.au/~give/Student/give.php)

If you submit your assignment more than once, we will only consider the last submission. If you face any problem while submitting your code, please e-mail the Course Admin (Maisie Badami, m.badami@student.unsw.edu.au (mailto:m.badami@student.unsw.edu.au))

Assessment

Your source code will be manually inspected and marked based on readability and ease of understanding. We will run your code to verify that it produces correct results. The code documentation (i.e. comments in your source code) is also important. Below, we provide an indicative assessment scheme (maximum mark: 20 points):

Result correctness	17 points
Code structure and source code documentation (comments)	3 points

Late submission penalty

10% reduction of your marks for the 1st day, 30% reduction/day for the following days.

Plagiarism

This is an *individual assignment* . The work you submit must be your own work. Submission of work partially or completely derived from any other person or jointly written with any other person is not permitted. The penalties for such offence may include negative marks, automatic failure of the course and possibly other academic discipline. Assignment submissions will be examined manually.

Do not provide or show your assignment work to any other person - apart from the teaching staff of this course. If you knowingly provide or show your assignment work to another person for any reason, and work derived from it is submitted, you may be penalized, even if the work was submitted without your knowledge or consent. Pay attention that is also your duty to protect your code artifacts. if you are using any online solution to store your code artifacts (e.g., GitHub) then make sure to keep the reposiroty private and do not share access to anyone.

Reminder: Plagiarism is defined as (https://student.unsw.edu.au/plagiarism) using the words or ideas of others and presenting them as your own. UNSW and CSE treat plagiarism as academic misconduct, which means that it carries penalties as severe as being excluded from further study at UNSW. There are several on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Plagiarism and Academic Integrity (https://student.unsw.edu.au/plagiarism)
- UNSW Plagiarism Procedure (https://www.gs.unsw.edu.au/policy/documents/plagiarismprocedure.pdf)

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism. In particular, you are also responsible for ensuring that your assignment files are not accessible by anyone but you by setting the correct permissions in your CSE directory and code repository, if using one (e.g., Github and similar). Note also that plagiarism includes paying or asking another person to do a piece of work for you and then submitting it as your own work.

UNSW has an ongoing commitment to fostering a culture of learning informed by academic integrity. All UNSW staff and students have a responsibility to adhere to this principle of academic integrity. Plagiarism undermines academic integrity and is not tolerated at UNSW.

The marking scheme for this assignment is as below:

---

1. The output is generated by Hadoop MapReduce correctly **(7 Marks)**

---

2. The output is formatted correctly as described in the spec **(2 marks)**

- Format must be the same as (M2,M3) [(U5,3,3),(U2,3,1)]
- There should be no empty list like: (M2,M3) []

3.Custom Writables are well defined (e.g., appropriate data-types have been used without wasting memory) **(4 marks)**

4. The code could be executed in CSE machines as described in the specification of the assignment **(2 Marks)**

5. The maximum number of Mappers and Reducers is 2 (2 Mappers and 2 Reducers) **(2 Marks)**

6. Documentation and code structure. here you can clearly explain you solution in a short paragraph in the beginning of the program (no more than 300 words) and provide comments describing what each class is doing. **(3 Marks)**