Assignment

ITNPBD7 Cluster Computing | Comp Sci | Stirling Uni | Spring 2025

This is the assignment *handout*. It describes the *tasks* which comprise the assignment. Each student must submit - via Canvas - files (*specified* below) which capture the important aspects of their work. Each student must work independently to complete the assignment. The submission deadline is 4pm Tuesday 25th March 2025.

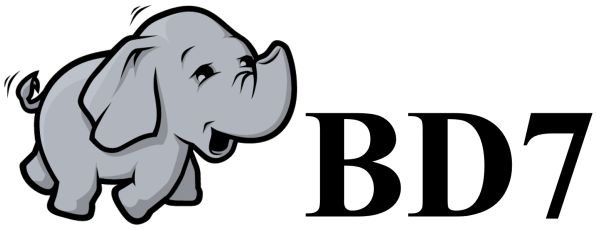

Table of Contents

1. The tasks

1.1. The problem: Best movie(s) by genre

The problem is to analyse a list of movie review ratings to find the film title(s) with the highest average rating for each genre, for a specified set of years and genres. You have been provided with a list of approximately 100,000 ratings from a group of reviewers in a file called `ratings.txt`. Each line in the file contains an individual user's rating for a particular movie as follows:

```TEXT
387 Kickboxer   Action  1989    2.5
599 Lionheart   Action  1990    0.5
599 Tornado!    Action  1996    2
599 Steel   Action  1997    1
217 Steel   Action  1997    3
140 Yojimbo Action|Adventure    1961    1
23  Yojimbo Action|Adventure    1961    3
...
```

The columns are organised as follows: `Reviewer ID`, `Movie Title`, `Genres`, `Year`, `Rating`. The genres entry will contain 1 genre or multiple genres separated by a vertical bar. Where a line contains multiple genres, you should associate that line's (total) rating value with each of the genres. For example:

```TEXT
23  Yojimbo Action|Adventure    1961    3
```

...should in effect become:

```TEXT
23  Yojimbo Action  1961    3
23  Yojimbo Adventure   1961    3
```

Two smaller versions of the ratings data are also supplied in the files `r5.txt` and `r100.txt`. These contain 5 rows of data and 100 rows of data, respectively. You have been also supplied with the two files `years.txt` and `genres.txt` which provide examples of the years and genres that are to be analysed.

- If `years.txt` is empty then all years should be analysed; otherwise only the listed years should be analysed.
- If `genres.txt` is empty then all genres should be analysed; otherwise only the listed genres should be analysed. (Apply this rule after the multiple genres processing that was discussed above.)

1.2. 🦣 Task 1: Implement a Map/Reduce based solution

You should write a Python based Hadoop Map/Reduce solution (program) that will, in a single run, find the film title(s) with the highest average rating for each genre. Your code must use data that is stored on HDFS and must execute correctly on the `hadoop.cs.stir.ac.uk` computer using the supplied `runhadoop.sh` script (after modification of the `xyz123` username to your own Comp Sci username).

The idea of the `years.txt` and `genres.txt` files is that their contents may be changed between (test) runs of your program, and that the output of your program should change accordingly. Your output should be in the following TSV (tab separated value) format. For each genre show the movie that has the highest average ranking for the genre together with the average ranking for that movie based on the range of years to be analysed. Average rankings should be calculated using Python's `decimal` arithmetic and the final results displayed to 4

decimal places. When there is a tie - two or more movies having that same highest average ranking for the genre - be sure to output a line for each. (Please note that the following example output is not the correct answer – it is just used to demonstrate the required structure of the results and to give you an idea of what is required.)

```
Romance Gone with the Wind  4.1579
Action  Man with the Golden Gun, The    3.9751
Drama   Kramer vs. Kramer   4.4290
Comedy  Back to the Future  4.4500
Romance Room with a View, A 3.7610
Horror  Interview with the Vampire: The Vampire Chronicles  3.5957
Drama   Beach, The  4.4290
Mystery Girl with the Dragon Tattoo, The    3.5000
```

TEXT

Code is supplied in the `mapper.py` file to load the content of the `years.txt` and `genres.txt` files into lists. It is up to you to decide how to use these lists. To ensure that single movie ratings with a maximum score do not dominate the results, you are required to only rank movies with at least 15 ratings. A variable (`min_votes`) to specify this is set in the `reducer.py` code although it has been set to a value of 1 for testing purposes (when working with a small set of data, most movies will have less that 15 ratings). You will probably want to adjust this while developing your code but remember to set it to 15 when you submit your code.

Hints

- Implement your solution using the Python code provided on the assignment page as a starting point.
- Remember that your implementation must execute correctly on the `hadoop.cs.stir.ac.uk` computer using the supplied `runhadoop.sh` script (after modification of the `xyz123` username to your own Comp Sci username).
- For this part, submit exactly the 3 files: `mapper.py` , `combiner.py` , `reducer.py` .
- In your submitted `reducer.py` make sure that you have set `min_votes = 15` .
- The default Combiner has been provided for you, will just emit whatever it is sent so that you can focus on the Mapper and Reducer first. Once you have this working, you can consider how to improve the implementation by writing a better Combiner.
- You can use the supplied `simhadoop.sh` script to test your code within a terminal on either your own computer or the `hadoop.cs.stir.ac.uk` computer. This version of `simhadoop.sh` will also produce intermediary files for the Mapper output (`mapout.txt`), Combiner output (`comout.txt`) and the Reducer output (`results.txt`). You will need to edit `simhadoop.sh` to switch between the alternative (test) input files: `r5.txt` , `r100.txt` and (the full) `ratings.txt` .
- Please be aware that when you use the `simhadoop.sh` , all the data from the Mapper or Combiner will be sent to a single Reducer. On Hadoop this may not be the case. If you have built a solution that relies on only one Reducer working, it may work fine with the `simhadoop.sh` script but not work on Hadoop in general and will be incorrect. It is also likely to be highly inefficient so a redesign would be worth considering.
- Apart from testing using `simhadoop.sh` , you should also test your implementation using the `runhadoop.sh` script on the actual `hadoop.cs.stir.ac.uk` computer.
- Testing is extremely important, and you are unlikely to develop a good implementation without substantial testing! Remember to test edge cases, e.g. when `ratings.txt` contains none of the years listed in `years.txt` ; when `genres.txt` is empty; when there is a tie; etc., etc. You should compare your generated output with manually calculated output, for a small set of test cases that you have manually constructed.

For this task you should submit exactly the 3 files:

- `mapper.py`
- `combiner.py`
- `reducer.py`

1.3. ✨ Task 2: Implement a Spark based solution

You should write a Python based Spark RDD solution (program) that solves the problem described in section 1.1. Your solution must build upon the code in the supplied `spark.py` file and must execute correctly on the `hadoop.cs.stir.ac.uk` computer.

This Spark RDD based solution should, for the same inputs, produce the same outputs as your Map/Reduce based solution.

For this task you should submit exactly the 1 file:

- `spark.py`

1.4. 🧪 Task 3: Outline how you have tested

Summarise the important aspects that you have tested.

This might take the form of a list of specific test cases. Where each test case has a short name and a (very) short description of its purpose.

This must be a maximum of one page long; and is likely to short phrases as bullet points or within a table.

For this task you should submit exactly the 1 file:

- `tesing.pdf`

1.5. 🦅 Task 4: Consider a Condor based solution

Assuming that the data set of this problem, was a petabyte in size...

Think about what would be involved in solving the problem on a distributed computation cluster such as Condor instead of using Hadoop. Then summarise the pros & cons of using Condor *-vs-* Hadoop to solve the problem (assuming a petabyte of data). Which *"wins"*?

This must be a maximum of half a page long; and is likely to short phrases as bullet points or within a table.

For this task you should submit exactly the 1 file:

- `condor.pdf`

1.6. Demonstrate your understanding

After you have submitted, you may be required to demonstrate/provide further evidence your understanding. This will take the form of a ≤15 minute recorded session (probably via Teams) in which you will be asked questions about your submission, design decisions, code, *what-if*s and also, what you have learnt form this module.

2. Submission and assessment

2.1. Submission deadline

Submission deadline 4pm Tuesday 25th March 2025.

2.2. Submission format

Files

Filename	Format	Advice
`mapper.py`	Python code file	Must be runnable by `runhadoop.sh` on the `hadoop.cs.stir.ac.uk` computer, and be well formatted.
`combiner.py`	Python code file	
`reducer.py`	Python code file	
`spark.py`	Python code file	Must be runnable on the `hadoop.cs.stir.ac.uk` computer, and be well formatted.
`testing.pdf`	PDF file	≦1 page. Short phrases. Probably as bullet points or within a table.
`condor.pdf`	PDF file	≦½ page. Short phrases. Probably as bullet points or within a table.

- Be succinct - *quality is better than quantity*.
- Code should be correct, readable/maintainable, performant and elegant.

Submission method Canvas. The university have produced a How-to submit (https://canvas.stir.ac.uk/courses/3228/pages/submitting-an-assignment-in-canvas) guidance.

2.3. Assessment type

Individual/group assessment Individual.

Contribution to module mark 100%.

Learning outcomes (LOs) LO1, LO2, LO3, LO4.

These are combined and summarised as:

- Demonstrate a critical understanding of the concepts and principles of using HDFS and Hadoop.
- Design and develop Map/Reduce and Spark based distributed data processing solutions suitable for a cluster of computers.
- Understand and identify when to use a distributed data approach in contrast to a distributed computation approach.
- Demonstrate an understanding of the principles of designing a distributed computation task using Condor.

2.4. Assessment scheme and mark allocation

Scheme

Assessment will be carried out and marks awarded in accordance with the university's *Common Marking Scheme* (https://www.stir.ac.uk/about/professional-services/student-academic-and-corporate-services/academic-registry/regulations/postgraduate-taught-regulations/#d.en.182642)
.

Allocation

Task	Mark allocation
Task 1: Implement a Map/Reduce based solution	60%
Task 2: Implement a Spark based solution	20%
Task 3: Outline how you have tested	10%
Task 4: Consider a Condor based solution	10%

2.5. Extension requests

To learn more about support available to you including extensions for coursework and understand what happens when you submit coursework late, please read Assessment Support and Submission (https://canvas.stir.ac.uk/courses/15765/pages/assessment-support-2).

2.6. Cheating and academic integrity

The University has an Academic Integrity policy (https://canvas.stir.ac.uk/courses/15765/files/3965715?wrap=1) which promotes good academic practice and identifies what we mean by academic misconduct, plagiarism and poor academic practice. It is important that you understand what these terms mean to ensure you avoid these activities and the penalties which may be applied if you are found to have done so. You can learn more about academic integrity by checking out the Academic Integrity and Academic Writing Support materials (https://canvas.stir.ac.uk/courses/15765/pages/academic-integrity-and-academic-writing-support).

2.7. Use of AI

In this module you are allowed to use AI at *level 5* on the university's *AI in Assessments Scale* (AIAS). Basically, this means that you can use AI to help you with all aspects of your work. However, you are expected to deeply understand your work/solutions.

You will be assessed on (the evidence for) what **you** have learnt.

Also, the unversity provides A student guide to using AI tools in assessments (https://canvas.stir.ac.uk/courses/3228/pages/a-student-guide-to-using-artificial-intelligence-tools-in-assessment).