Supercomputing for Big Data ET4310 (2016)

Assignment 2

Hamid Mushtaq and Zaid Al-Ars

Introduction

The task of this assignment is to perform specific data analytics on Kevin Bacon to identify the actors (both male and female) linked to him by the six degrees of separation (more info can be found on the Wikipedia page "Six Degrees of Kevin Bacon").

First you need to download the files **actors.list.gz** and **actresses.list.gz** from the following link

ftp://ftp.fu-berlin.de/pub/misc/movies/database/

After downloading, unzip them and remove the header and footers of those two files using the python script that we wrote, given in the following link. The script takes two arguments. The first one is the path of the input file while the second one is the path of the output file (Example: ./rmIMDBHeaderAndFooter.py actors.list mod_actors.list).

https://www.dropbox.com/s/an9bcdqiyzj2tmo/rmIMDBHeaderAndFooter.py?dl=0

Make sure afterwards that the newly created files were indeed correctly produced by using the diff command. Applying a diff with the original and new file should only show the headers and footers as the difference.

You must use the template code given in the link below to complete your assignment. Compile this code by using the command **sbt package**.

https://www.dropbox.com/sh/fi005q4wqtn5gnl/AAC5liMIereNaOEVa0V0Fpz3a?dl=0

The steps required to complete this assignment are described later in the text.

System requirements and Installation

The requirements for software packages are the same as were for Lab 1. However, make sure you have a PC with at least 6 GB of RAM and 4 cores. The run script provided with this assignment runs the program with 5 GB of RAM and we are going to benchmark your program by using this run script on a 4 core computer equipped with 8 GB of RAM.

The first parameter of the run script specifies the number of cores, while the second and third parameters specify the path of the two input files (one for male actors and the other for female actors (actresses)).

Important points

You must keep the following points in mind for this assignment.

- The division of marks is the following

- Code functionality (35%)
- Code (Readability, structure, performance, etc.) (25%)
- Comments (25%)
- Report (15%)

- Each statement in your code must be accompanied by a comment, especially on each RDD transformation or action. This is so that you could demonstrate that you thoroughly understand your code, even if you took a bit of help from somewhere. Also, for each RDD, you must specify the type of its elements in your comments. For example

```
// This is an RDD that describes … The contents of the RDD describes (actor, (movie, list_of_collaborating_actors))
SomeRDD = …
```

- In your report, you don't need to go too much into details, as through your comments, you should already be able to explain your code quite well. Secondly, the report should be no longer than 6 pages. You must use the following template for the report.
  https://www.dropbox.com/s/yzscpb4tvldz2zm/Report.docx?dl=0
  Make sure also that you follow the points given in the template.
- You must submit a folder (in zipped form) containing the code and your report. That zip file should be named as **SBD_L2_YourName_YourStudentNumber**. You must use the template provided for the exercise. You can modify the run script and add more source files if required though.
- After removing the headers and footers from the input files, you are not allowed to make any further changes to the input files.

The algorithm

In this assignment, we will compute the actors and actresses connected to Kevin Bacon at distance from one up to six degrees of separation in Spark. Below, we give an outline of the approach that can be used to solve it. Remember though that you have to include both male and female actors in your computation. That means, keeping some kind of a flag to identify the gender. Moreover, it is not necessary to follow the outlined points below exactly. Try to make your code as memory and performance efficient as possible.

- Create an RDD of type <actor, List<movies>> by using the **newAPIHadoopFile** function.
- From that RDD, generate an RDD with actors and the movies they acted in (<actor, movie>).
- Next create an RDD for actors and their collaborating actors (<actor, actor>).
- Reduce that RDD into an RDD of type <actor, list<actor>>.
- Create a new RDD describing actors and their distances to Kevin Bacon (<actor, distance>), where distance=infinity, except for Kevin Bacon for whom distance is 0.
- Join the previous RDDs into a new one showing the collaborating actors with a given actor at a specific distance (<actor, <distance, list<actor>>>).
- Use this list to generate an RDD of type <actor, distance+1>.
- Reduce this list by taking the minimum function of the distance (<actor, minDistance>).
- Iterate 6 times and identify the list of actors (both male and female) at a distance of 6 from Kevin Bacon. For example, after the first iteration, Bacon will have a distance of 0 and every actor who worked with him would be at distance 1, while the rest of the actors would be at distance of infinite. This connectivity would spread in the next iterations.

The Exercise and its output

In this assignment you have to fulfil the following requirements.

- Implement the given algorithm that generates an output file (the name of that file must be actors.txt) of the exact same format as given in the link below.

  https://www.dropbox.com/s/jv544vkj0e21mk9/sbd_lab2_sample_output.txt?dl=0

  Note that we generated this sample output with a different input. So, your output would not be the same.

- You have to filter out all the TV series (titles starting with “) and only take the titles till the year. For example, for *When the Man Went South (2014) [Two Palms - Ua'i Paame] <8>,* use *When the Man Went South (2014).*
- You only have to see for movies in this decade, which means, movies from year 2011 till now. Ignore the movies where the year is not given. This also means that any actor who hasn’t stared in movies of this decade has to be ignored. Moreover, note that for some movies the year is written as for example (2016/II). In that case just read out the year by looking at the first four characters within the brackets.
- Using **SparkListener** (https://spark.apache.org/docs/2.0.0/api/java/org/apache/spark/scheduler/SparkListener.html), profile the memory consumption of cached RDDs. For uncached RDDs, just report whenever they are completed. The profile output should be logged in a file SparkLog.txt. You can use the **setName** method of RDD, combined with the **name** method of **RDDInfo** (https://spark.apache.org/docs/1.4.0/api/java/org/apache/spark/storage/RDDInfo.html) to display the names of the RDDs properly in the log. Please make use of the given **getTimeStamp** function to print the timestamps each line. Moreover, you must also print the number of partitions of the RDDs. To make things clearer, your output should have a similar format to the following.

  [16:20:59] rdd_movie2ActorActor processed!
  [16:21:43] rdd_actor2listOfActors: memsize = 917MB, diskSize = 0, numPartitions = 16
  [16:21:44] rdd_actor2distance: memsize = 67MB, diskSize = 0, numPartitions = 16

- In the template code there is a flag **compressRDDs**. When set to true, the RDDs must be compressed. You must add the code to do that. Check performance (memory and execution time) with both this flag set to true and false, and report the results in your report.