Supercomputing for Big Data ET4310
Assignment 2
Submission deadline - 21st October

Six degrees of separation from Kevin Bacon

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

First, we need to download the film and actor database of the Internet Movie Database (IMDb). We have downloaded and created a preformatted version of the database that you can download at any of the following links.
https://onedrive.live.com/?id=1878D3BD92115E62%213679&cid=1878D3BD92115E62&group=0&parId=root&authkey=%21ABM6dRoL-j6wdRo&action=locate
https://www.asuswebstorage.com/navigate/s/8F037585BDD2499BAD6A22A83D7ADBB3W
https://www.dropbox.com/s/81zaja8wqwl16e5/actors.list.zip?dl=0

You must use spark version 1.4 or greater (Pre built for Hadoop 2.4 and later) for this assignment.

The program can be written in either Python or Scala. Use the templates given on the blackboard for that purpose. You are allowed to modify the code (But do not change the main class, project and file names in the template) and run script, but not allowed to change the spark-env.sh file. The run script in the template takes two parameters: the first parameter is the number of CPU cores the program can use, while the second parameter is the input file path. Do not add any additional parameters or remove any parameters. Moreover, you must use the $SPARK_HOME environment variable. So do not change that in the run script.

Follow the following steps to complete the assignment. It is not necessary to follow the exact same steps as long as your program gives the correct results.

Initialization:

1. Parse the input file to generate a key value pair RDD with actors as keys and the list of movies they have worked in as values <actor, List<movies>>. You must use the newAPIHadoopFile function for that purpose.
2. From the previous RDD, generate an RDD with actors and the movies they acted in <actor, movie>.
3. Map the RDD into another key value pair with actors and their collaborating actors <actor, actor>
4. Reduce this RDD into a pair of actors and a list of collaborating actors <actor, List<collaboratingActors>>
5. Create a new RDD describing actors and their distances to Kevin Bacon <actor, distance>, where distance=infinity, except for Kevin Bacon

Analysis:

1. Join the previous RDDs into a new one showing the collaborating actors with a given actor at a specific distance <actor, <distance, List<collaboratingActors>>>
2. Use this list to generate the following key value pairs <collaboratingActor, distance+1>
3. Reduce this list by taking the minimum function of the distance at each key <collaboratingActor, minDistance>
4. Iterate 6 times and identify the list of actors at a distance of 6 from Kevin Bacon. For example, after the first iteration, Kevin will have a distance of 0, everybody who worked directly with him a distance of 1, while all the rest will have a distance of infinity. This connectivity will spread further in the following iterations.
5. You must write the following to a file "actors.txt".
    1. Total number of actors
    2. Total number of movies (and TV shows etc). Note that your program should use the full titles, which therefore means that different episodes of the same show are treated as different titles.
    3. Total number of actors at distance 1 to 6 (Each distance information on new line)
    4. The name of all actors at distance 6 sorted alphabetically (ascending order), with each actor's name on new line.
6. Optimize your code for memory and performance. You have to check the performance on any of the computers in the Veemhal lab.
7. Write a short report describing your work and results. State clearly all the optimizations that you performed and the improvement in performance that you got from each one of them (also summarised in table).

Marks distribution:

1.	Parsing of input file to create the <actor, List<movies>> RDD	(15)
2.	Implementation of the rest of the code	(30)
3.	Results written to the actors.txt file	(10)
4.	Optimizations and performance	(30)
5.	Report	(15)