



ÉCOLE CENTRALE DE LYON

MSO 3.1 - TECHNOLOGIES INFORMATIQUES DU BIG DATA
OPTION INFORMATIQUE
RAPPORT

TP2 – Spark

Élèves :
Thibaud MIQUEL

Enseignant :
Stéphane DERRODE

Table des matières

1	Présentation du TP	3
1.1	Objectif du TP	3
1.2	Présentation de Spark	3
2	Réponse aux questions	4
2.1	Partie 2 : The MovieLens Database	4

Table des figures

1	Top résultat de la requête	4
2	Résultat sous le top malgré le nombre de notation des films	4
3	Top résultat de la requête finale	5
4	Code de la requête finale	5

1 Présentation du TP

1.1 Objectif du TP

L'objectif de ce TP est d'utiliser la bibliothèque pyspark pour coder des requêtes et les exécuter directement sur Hadoop en passant par Docker.

1.2 Présentation de Spark

Spark est une API de programmation parallèle sur des données. L'objet principal de Spark est le RDD : Resilient Distributed Dataset. C'est un dispositif pour traiter une collection de données par des algorithmes parallèles robustes. Un RDD ne contient pas vraiment de données, mais seulement un traitement.

Ce traitement n'est effectué que lorsque cela apparaît nécessaire. On appelle cela l'évaluation paresseuse. D'autre part, Spark fait en sorte que le traitement soit distribué sur le cluster, donc calculé rapidement, et n'échoue pas même si des machines tombent en panne. Spark permet donc d'écrire des traitements complexes composés de plusieurs phases map-reduce.

<https://perso.univ-rennes1.fr/pierre.nerzic/Hadoop/tp4.pdf> [https://perso.univ-rennes1.fr/pierre.nerzic/Hadoop/slide 8](https://perso.univ-rennes1.fr/pierre.nerzic/Hadoop/slide%208.pdf) <https://studylibfr.com/doc/3997831/hadoop—semaine-5> <https://www.bestcours.com/documents/0582-outils-hadoop-pour-le-bigdata.pdf> page 60

2 Réponse aux questions

2.1 Partie 2 : The MovieLens Database

Ma requête consiste à renvoyer une liste triée des films sur leur notation moyenne par ordre décroissant.

J'ai construit ma requête sans utiliser la fonction moyenne pour faire toute la fonctionnalité moi même. J'ai donc tout d'abord sommé dans un premier dataframe pris de ratings.csv toutes les notations de chaque film en gardant les id des films comme clé. Puis dans un 2eme dataframe j'ai compté le nombre de fois où chaque film est noté en gardant également comme clé l'id des films. Je joins ensuite ces 2 dataframes sur les id puis je divise la somme des notes par le nombre de fois où le film est noté pour obtenir la note moyenne, enfin je trie cette liste par moyenne de notation en ordre décroissant.

Pour obtenir également le nom du film je joins ce dernier dataframe à un dataframe récupéré via movies.csv sur les id de film.

Voici le résultat de cette requête :

```
"1151", ((5.0, 10.0, 2), 'L'assommoir (1944)'))
"102217", ((5.0, 5.0, 1), 'Bill Hicks: Revelations (1993)'))
"27223", ((5.0, 5.0, 1), 'My Sassy Girl (Yeopgijeogin geunyeo) (2001)'))
"53", ((5.0, 10.0, 2), 'Lamerica (1994)'))
"53578", ((5.0, 5.0, 1), 'La Vierge (2007)'))
"60727", ((5.0, 5.0, 1), 'Watching the Detectives (2007)'))
"83969", ((5.0, 5.0, 1), 'Down Argentine Way (1940)'))
"96608", ((5.0, 5.0, 1), 'Runaway Brain (1995)'))
"109591", ((5.0, 5.0, 1), 'Hunting Elephants (2011)'))
"109241", ((5.0, 5.0, 1), 'On the Other Side of the Tracks (De l'autre côté du périph) (2012)'))
"118834", ((5.0, 5.0, 1), 'National Lampoon's Bag Boy (2007)'))
"124804", ((5.0, 5.0, 1), 'Snowflake (2012)'))
"126088", ((5.0, 5.0, 1), 'A Flintstones Christmas Carol (1994)'))
"126921", ((5.0, 5.0, 1), 'The Fox and the Hound 2 (2006)'))
"131898", ((5.0, 5.0, 1), 'Spring Santa (2013)'))
"136583", ((5.0, 5.0, 1), 'Tom and Jerry: Shiver Me Whiskers (2006)'))
"140113", ((5.0, 5.0, 1), 'Hollywood Chainsaw Hookers (1988)'))
"141928", ((5.0, 5.0, 1), 'Bloodbuckling Bostards (2015)'))
"156025", ((5.0, 5.0, 1), 'Ice Age: The Great Egg-Scapade (2016)'))
"158882", ((5.0, 5.0, 1), 'All Yours (2016)'))
"4788", ((5.0, 5.0, 1), 'Moscow Does Not Believe in Tears (Moskva slezom ne verit) (1979)'))
"5880", ((5.0, 5.0, 1), 'Crucel Romance'))
"95175", ((5.0, 5.0, 1), 'Front of the Class (2008)'))
"95211", ((5.0, 5.0, 1), 'Presto (2008)'))
"109631", ((5.0, 5.0, 1), 'Garden of Words'))
"136445", ((5.0, 5.0, 1), 'George Carlin: Back in Town (1996)'))
"140605", ((5.0, 5.0, 1), 'George Carlin: Jammin' in New York (1992)'))
"143511", ((5.0, 5.0, 1), 'Human (2015)'))
"147250", ((5.0, 5.0, 1), 'The Adventures of Sherlock Holmes and Doctor Watson'))
"147278", ((5.0, 5.0, 1), 'The Adventures of Sherlock Holmes and Dr. Watson: Bloody Signature (1979)'))
"163072", ((5.0, 5.0, 1), 'Winnie Pooh (1990)'))
"163925", ((5.0, 5.0, 1), 'Mings'))
"172280", ((5.0, 5.0, 1), 'Winter in Prostokvashino (1984)'))
"173019", ((5.0, 5.0, 1), 'Fugitives (1986)'))
"174551", ((5.0, 5.0, 1), 'Obsession (1965)'))
"179133", ((5.0, 5.0, 1), 'Loving Vincent (2017)'))
"7096", ((5.0, 5.0, 1), 'Rivers and Tides (2001)'))
"3096", ((5.0, 5.0, 1), 'My Man Godfrey (1957)'))
```

FIGURE 1 – Top résultat de la requête

```
"912", ((4.26, 424.0, 100), 'Casablanca (1942)'))
"98", ((4.27724598889225, 804.5, 200), 'Usual Suspects'))
"1197", ((4.2329436619731, 681.0, 142), 'Princess Bride'))
"3030", ((4.230769230769231, 55.0, 13), 'Yojimbo (1961)'))
"932", ((4.217913862472626, 97.0, 23), 'To Catch a Thief (1955)'))
"1190", ((4.215539818426565, 889.5, 211), 'Star Wars: Episode V - The Empire Strikes Back (1980)'))
"103688", ((4.214285714285714, 29.5, 7), 'Conjuring'))
"1198", ((4.2075, 841.5, 200), 'Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)'))
"152", ((4.205882328411771, 71.5, 17), 'Crumb (1994)'))
"1089", ((4.20220076335878, 550.5, 131), 'Reservoir Dogs (1992)'))
"73223", ((4.2, 21.0, 5), 'Geri Lino Kicked the Heron's Nest'))
"134", ((4.2, 21.0, 5), 'Yanya on 42nd Street (1984)'))
"5580", ((4.2, 42.0, 10), 'Top Secret! (1984)'))
"3384", ((4.2, 42.0, 10), 'Key Largo (1948)'))
"3838", ((4.2, 21.0, 5), 'Face in the Crowd'))
"55988", ((4.1875, 33.5, 8), 'Man from Earth'))
"3675", ((4.1875, 33.5, 8), 'White Christmas (1954)'))
"7921", ((4.181818181818182, 46.0, 11), 'High Plains Drifter (1973)'))
"5965", ((4.166666666666667, 12.5, 3), 'Duellists'))
"2505", ((4.166666666666667, 50.0, 12), 'King and I'))
"3075", ((4.166666666666667, 12.5, 3), 'Repulsion (1965)'))
"3053", ((4.166666666666667, 12.5, 3), 'Endless Summer'))
"52444", ((4.166666666666667, 12.5, 3), 'Control (2007)'))
"156371", ((4.166666666666667, 12.5, 3), 'Everybody Wants Some (2016)'))
"25841", ((4.166666666666667, 12.5, 3), 'Stage Door (1977)'))
"6027", ((4.166666666666667, 12.5, 3), 'Dogfight (1991)'))
"4444", ((4.166666666666667, 12.5, 3), 'Way of the Dragon'))
"358", ((4.164133738601824, 1370.0, 129), 'Forrest Gump (1994)'))
"1116", ((4.16176479828255, 566.0, 183), 'Monty Python and the Holy Grail (1975)'))
"3883", ((4.157884720842105, 29.0, 10), 'All About My Mother (Todo sobre mi madre) (1999)'))
"5690", ((4.15625, 66.5, 16), 'Grave of the Fireflies (Hotaru no haka) (1988)'))
"2028", ((4.150250597448089, 779.5, 188), 'Saving Private Ryan (1998)'))
"2807", ((4.142857142857143, 87.0, 21), 'Doctor Zhivago (1965)'))
"68157", ((4.136363636363637, 364.0, 88), 'Inglourious Basterds (2009)'))
"1281", ((4.133333333333334, 62.0, 15), 'Great Dictator'))
```

FIGURE 2 – Résultat sous le top malgré le nombre de notation des films

Je me suis rendu compte que ce résultat n'était pas très intéressant car je n'ai pas tenu compte du nombre de fois minimal qu'un film devrait être noté pour apparaître dans ce top. C'est pourquoi dans les résultats ci-dessus il n'y a que très peu de film connu dans les meilleurs résultats.

Dans un 2eme temps j'ai donc rajouté comme argument à donner par l'utilisateur un seuil de nombre de notation minimum pour qu'un film apparaisse dans ce top. La requête que j'ai formulé est donc la suivante : `spark-submit --deploy-mode client --master local[2] Q1_movie.py input/ratings.csv input/movies.csv 15` où 15 le nombre de notation minimum qu'un film doit avoir.

```
('3468', ((4.333333333333333, 78.0, 18), '"Hustler"'))
('1204', ((4.3, 193.5, 45), 'Lawrence of Arabia (1962)'))
('246', ((4.293103448275862, 124.5, 29), 'Hoop Dreams (1994)'))
('1235', ((4.288461538461538, 111.5, 26), 'Harold and Maude (1971)'))
('168252', ((4.28, 107.0, 25), 'Logan (2017)'))
('2959', ((4.272935779816514, 931.5, 218), 'Fight Club (1999)'))
('1213', ((4.25, 535.5, 126), 'Goodfellas (1990)'))
('930', ((4.25, 85.0, 20), 'Notorious (1946)'))
('3508', ((4.25, 76.5, 18), '"Outlaw Josey Wales"'))
('912', ((4.24, 424.0, 100), 'Casablanca (1942)'))
('50', ((4.237745098039215, 864.5, 204), '"Usual Suspects"'))
('1197', ((4.232394366197183, 601.0, 142), '"Princess Bride"'))
('933', ((4.217391304347826, 97.0, 23), 'To Catch a Thief (1955)'))
('1196', ((4.2156398104265405, 889.5, 211), 'Star Wars: Episode V - The Empire Strikes Back (1980)'))
('1198', ((4.2075, 841.5, 200), 'Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1957)'))
('162', ((4.205882352941177, 71.5, 17), 'Crumb (1994)'))
('1089', ((4.202290076335878, 550.5, 131), 'Reservoir Dogs (1992)'))
('356', ((4.164133738601824, 1370.0, 329), 'Forrest Gump (1994)'))
('1136', ((4.161764705882353, 566.0, 136), 'Monty Python and the Holy Grail (1975)'))
('3083', ((4.157894736842105, 79.0, 19), 'All About My Mother (Todo sobre mi madre) (1999)'))
('5690', ((4.15625, 66.5, 16), 'Grave of the Fireflies (Hotaru no haka) (1988)'))
('2028', ((4.1462765957446805, 779.5, 188), 'Saving Private Ryan (1998)'))
('2067', ((4.142857142857143, 87.0, 21), 'Doctor Zhivago (1965)'))
('68157', ((4.136363636363637, 364.0, 88), 'Inglourious Basterds (2009)'))
```

FIGURE 3 – Top résultat de la requête finale

On voit ici un résultat beaucoup plus cohérent avec des noms de films particulièrement connus comme : "Logan", "Fight Club" et "Usual Suspect".

```
5 #!/usr/bin/env python
6 #-*- coding: utf-8 -*-
7 import sys
8 from pyspark import SparkContext
9
10 if __name__ == "__main__":
11
12     # Creation d un contexte Spark
13     sc = SparkContext(appName="Spark Count")
14     sc.setLogLevel("ERROR") # Valid log levels include: ALL, DEBUG, ERROR, FATAL, INFO, OFF, TRACE, WARN
15
16     tableau_ratings = sc.textFile(sys.argv[1]).map(lambda line: line.split(',')) # userId movieId rating
17     tableau_movies = sc.textFile(sys.argv[2]).map(lambda line: line.split(','))
18
19     paires_ratings = tableau_ratings.map(lambda champs: (champs[1],champs[2])) # movieId rating
20     paires_movies = tableau_movies.map(lambda champs: (champs[0],champs[1])) # movieId title
21
22     paires_ratings = paires_ratings.filter(lambda duet: duet[1]!='' and duet[1]!='rating') # on retire les valeurs manquantes et le titre
23     paires_movies = paires_movies.filter(lambda duet: duet[0]!='movieId') # on retire le titre
24
25     paires_ratings = paires_ratings.map(lambda champs: (champs[0],float(champs[1]))) # movieId rating
26
27     paires_ratings_sum= paires_ratings.reduceByKey(lambda v1,v2 : v1+v2) # on somme les notes de tous les films
28
29     paires_ratings_count = tableau_ratings.map(lambda champs: (champs[1],1)) # movieId 1
30     paires_ratings_count= paires_ratings_count.reduceByKey(lambda v1,v2 : v1+v2) # on compte le nombre de fois où chaque film est noté
31
32     threshold = float(sys.argv[3])
33     paires_ratings_count = paires_ratings_count.filter(lambda pair: pair[1]>threshold) # on donne un nombre minimale de notation de film
34
35     classement = paires_ratings_sum.join(paires_ratings_count) # movie id, (sum des ratings, count des ratings)
36     classement = classement.map(lambda champs: (champs[1][0]/champs[1][1],(champs[0],champs[1][0],champs[1][1]))) # rapport sum/count,(movie id, sum des ratings
37     classement = classement.sortByKey(ascending=False) # classement décroissant par rapport sum/count
38     classement_join = classement.map(lambda champs: (champs[1][0],(champs[0],champs[1][1],champs[1][2]))) # movie id, (rapport sum/count,sum des ratings, count
39
40     classement_final = classement_join.join(paires_movies) # on join les 2 dataframes pour obtenir le nom des movies
41
42     classement_final.saveAsTextFile("classement")
43
44     # Arrêt du contexte Spark
45     sc.stop()
```

FIGURE 4 – Code de la requête finale