Cours de Big Data

5A – Polytech Paris-Sud – 2023-2024

TD-1 : Map-Reduce en Spark

Stéphane Vialle & Gianluca Quercini

Exercice 1 : Calcul de moyennes sur des séries de données

Question 1.1 : travail sur de petites séries de données

On considère un fichier CSV contenant des mesures *« Année, Mois, Température »,* à raison d'une seule température mesurée par mois (donc 12 mesures par an).

On souhaite arriver à générer des couples *(Année, TempératureMoyenne)* par une approche Map-Reduce en Spark.

- Décrivez une approche Map-Reduce résolvant le problème, sans enrichir les données durant le processus (ne manipulez que des années et des températures).
- Proposez une implantation en Spark en complétant le code suivant :

```
#map-reduce computation ------------------------------------------
def avg_temperature(theTextFile):
    avg = theTextFile \
            .map(lambda line: line.split(",")) \
            . ……..            # nb of line is not limited – TO DO
    return avg

#main code -------------------------------------------------------
Input_hdfs_file_name = ……    #details in the real python source file
Output_hdfs_file_name = ……   #details in the real python source file

sc = SparkContext()

input_text_file = sc.textFile(input_hdfs_file_name)
results = avg_temperature(input_text_file)
results.saveAsTextFile(output_hdfs_file_name)
```

- Votre code Spark maintient-il des RDD jusqu'à sa dernière étape ?
- Votre code Spark contient-il des opérations *Narrow* ou *Wide* ?
- Quels sont les défauts de cet algorithme ?
- On suppose maintenant qu'il n'y a qu'une seule clé (toutes les températures ont été collectées la même année). Soit N_s le nombre de Spark executors, N_t le nombre total de températures lues.
    - Quelle va-t-il arriver si le nombre de données augmente ?
    - Estimez le volume de données injectée sur le réseau durant le *shuffle*.

Question 1.2 : travail sur de grandes séries de données

On considère maintenant un fichier CSV de grande taille (stocké en profitant pleinement d'un système de fichiers distribué HDFS), contenant des mesures *« Année, Mois, Jour, heure, minute, seconde, Température »*. On peut avoir jusqu'à une mesure par seconde certaines années.

Comme précédemment, on souhaite arriver à générer des couples *(Année, TempératureMoyenne)* par une approche Map-Reduce en Spark.

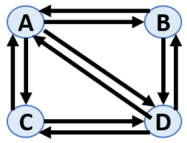
- Combien de mesures peut-il y avoir au maximum pour une année ?
  Peut-on continuer avec l'approche algorithmique de la question 1 ?
- Proposez une approche plus adaptée à ces grandes séries de données, et une implantation Spark.
- Votre code Spark maintient-il bien des RDD ? favorise-t-il des opérations Narrow ? Quelles opérations re-partitionnent les données ?

Exercice 2 : Recherche d'amis communs dans un réseau social

On considère un graphe de réseau social encodé dans un fichier texte.

- Une ligne du fichier est une liste d'identificateurs séparés par des virgules : A, B, C, D
  qui signifie que A a pour amis B, C et D.
- L'ensemble des lignes du fichier décrivent les relations « a pour ami » du graphe. Ex :

  A, B, C, D
  B, A, D
  C, A, D
  D, A, B, C



- On supposera les relations symétriques : si on A, B alors on a B, A.

On souhaite obtenir la liste des amis communs à des couples d'individus :

(A, B), [D]
(A, C), [D]
(A, D), [B, C]
(B, C), [A, D]
(B, D), [A]
(C, D), [A]

Rmq : On ne veut représenter chaque couple qu'une fois (comme ci-dessus). Si on présente les amis du couple (A, B), alors on ne représentera PAS le couple (B, A).

On considère le code Spark suivant à completer :

```
#Search for new friends ----------------------------------------
def common_friends(theTextFile):
  cf = theTextFile                                \
       .map(lambda line: line.split(","))         \
       . ..............      # nb of lines is not limited - TO DO TO DO
  return cf
```

```
#main code ----------------------------------------------------
input_file_name = ……      #details in the real python source file
output_file_name = ……     #details in the real python source file

sc = SparkContext()
text_file = sc.textFile(input_file_name)

cf = common_friends(text_file)
cf.saveAsTextFile(output_file_name)
```

Question 2.1 : production de paires clé-valeur intermédiaires

Dans un premier temps on va chercher à produire des paires clé-valeur représentant UN ami commun à un couple d'individus.

Ex : ((A, D), B) signifiera que B est un ami commun à A et D. La clé est ici un couple.

- Proposez une solution en Spark pour obtenir l'ensemble des paires clé-valeur (Couple, Ami) du graphe à partir du fichier texte représentant le graphe.
- Est-ce que votre solution génère des paires symétriques ?

Question 2.2 : production de la solution complète

- Proposez maintenant une solution complète en Spark, pour obtenir la liste des amis communs des couples d'individus (s'ils ont bien des amis communs) :

  (A, B), [X_1, X_2, … X_n]
  ….

- Restez-vous bien en RDD jusqu'à la fin des traitements ?
- Quels sont vos étapes *Narrow* et vos étapes *Wide* ?

Question 2.3 : modélisation de la complexité du traitement

Soit :

- N_{id} : le nbr d'identificateurs (de nœuds) dans le graphe (i.e. le nbr d'individus dans le réseau social)
- N_{link} : le nbr total de liens bidirectionnels dans le graphe (i.e. le nbr total de relations symétriques entre deux individus)
- n_{link}^{id} : le nbr de liens sortant du nœud d'identificateur *id.*

Le nbr de liens n_{link}^{id} varie bien sûr d'un nœud à l'autre. Mais dans ce TD on fait l'hypothèse simplificatrice que pour chaque nœud : $n_{link}^{id} = \overline{n_{link}}$

- Calculer le nombre de paires intermédiaires ((A,B), X) générées en sortie de la question 1.
- Peut-on considérer que la complexité du problème est directement liée au nombre de paires générées en sortie de la question 1 (et donc en entrée de la question 2) ?

Si N_{id} et N_{link} sont des valeurs totalement indépendantes, cette recherche des amis communs est-elle plus sensible au nombre total de nœuds ou au nombre total de liens ?

Exercice 3 : Calcul de moyennes et d'écarts types sur de grandes séries de données

On repart de la question 2 de l'exercice 1, et de ses mesures *« Année, Mois, Jour, minute, seconde, Température »* dans un fichier CSV de grande taille.

Mais on veut maintenant arriver à des triplets de valeurs : *(Année, TempératureMoyenne, EcartType)*.

On peut exprimer l'écart type de N valeurs x_i (avec : $0 \leq i < N$) selon deux définitions équivalentes :

- Définition 1 (définition classique) : $\sigma = \sqrt{\frac{\sum_{i=0}^{i=N-1}(x_i - \bar{x})^2}{N}}$
- Définition 2 : $\sigma = \sqrt{\overline{x^2} - \bar{x}^2} = \sqrt{\frac{\sum_{i=0}^{i=N-1}(x_i{}^2)}{N} - \left(\frac{\sum_{i=0}^{i=N-1} x_i}{N}\right)^2}$

Question 3.1 : choix d'une expression mathématique

Quelle expression de l'écart type vous semble la plus exploitable dans une approche Map-Reduce et une implantation Spark (et pourquoi) ?

Question 3.2 : algorithme et implantation Spark

Proposez une approche Map-Reduce et un code Spark répondant au problème pour de grandes séries de données.