

BDLE – 5IN852 – REVISIONS (Examen du 7 février 2020)

TABLEAU de séance

Texte EN BLEU écrit pendant la séance du 8/01/2021

Exercice 1 : Données touristiques**8 pts**

On considère le schéma. Chaque table est un dataset.

Visite (photoID, personID, date, lat, lon) *on connaît la (latitude, longitude) d'une photo*

Intérêt (POI, lat, lon, catégorie) *catégorie vaut hôtel, musée, restau, ...*
on connaît la (latitude, longitude) d'un point d'intérêt POI.

Place (photoID, pays, ville) *il y a 50 pays et 20 villes par pays en moyenne*

Il y a 10 000 tuples dans Visite, 11 000 dans Place et 1000 dans Intérêt.

On suppose que tous les attributs sont indépendants.

La répartition initiale d'un dataset est **aléatoire** (ie., ne dépend pas d'un attribut) sur **10 machines**, avec **une partition par machine**, et le même nombre d'objets dans chaque partition.

L'ordonnancement du traitement en plusieurs étapes suit le principe map-reduce de spark : rassembler dans une étape (*stage*) toutes les opérations pouvant être faites avant un transfert (*shuffle*) qui répartit les données pour d'autres étapes. On quantifie, si possible, les transferts de données en nombre d'objets transférés.

Question 1. Dans Place, une photo peut être associée à plusieurs pays. Par exemple, on peut avoir les tuples (photo1, England, London) et (photo1, United-Kingdom, London) pour une photo prise à Londres. On considère la requête :

E1 = Place.groupBy("photoID", "ville").agg(collect_list("pays").as("listeP"))

Décrire l'exécution de E1 :

Réponse :

Requête en SQL: select photoID, ville, collect_list(pays) as listeP

From Places

Group by photoID, ville

Voir le TP sur le explain d'une requête group by. Attention l'agrégation n'est pas un count ni un avg mais un collect_list

Fonction partielle dans chaque partition contenant 1100 tuples : agréger les pays pour avoir un ensemble (photo, ville, listePays). On obtient A(photo, ville, listePays) Rmq : card(A)= 10 000 car il y a 10 000 photoID distincts

Repartition de A selon (photo, ville) correspond au shuffle

Fonction complète dans chaque « nouvelle » partition obtenue après la répartition *shuffle*

Pour tous les tuples ayant les mêmes (photo,ville) faire l'union des listePays. On obtient le résultat demandé E1

Question 2. On considère Place1 (photoID, pays, ville) où photoID est **unique**. Expliquer brièvement comment obtenir Place1 à partir de Place. Vous pouvez répondre par une expression en syntaxe Dataframe.

En SQL :

E2a: On suppose que photoID → Ville

with T as (select photoID, ville, collect_list(pays) as listeP

From Places

Group by photoID, ville)

Select photoID, ville, first(listeP) as pays from T

E2b: sans supposer que photoID → Ville

with T as (select photoID, collect_list(ville, pays) as listeVP

From Places

Group by photoID)

Select photoID, first(listeVP) as pays from T

Remarque sur l'exécution parallèle de E2a et E2b : leur contenu n'est **pas** déterministe. Pour rendre le résultat déterministe il faut imposer un **ordre** avant de faire le first

E2c :

with T as (select photoID, ville, collect_list(pays) as listeP

From Places

Group by photoID, ville)

Select photoID, ville, first(array_sort(listeP)) as pays from T

En syntaxe dataframe :

Place1 = E1.withColumn('pays', first(array_sort(listeP)).alias('pays')).select('photoID', 'ville', 'pays')

REMARQUE : il y a aussi la méthode dropDuplicates("photoID")

⚠ distinct() 也有 Hashagg ⇒ Exchange Hashpartitioning ⇒ Hashagg

Question 3. Soit la requête affichant le nombre de villes dans chaque pays :

E3 = Place1.groupBy("pays").agg(countDistinct("ville").as("nv")).orderBy("pays")

Proposer une exécution avec **un seul** transfert (= **1 seule répartition**) qui répartit les données par **intervalle** (et non par hachage). Décrire les étapes partielles et complètes du traitement.

Rappel répartition par intervalle : opérateur Exchange RangePartitionning dans le explain

On veut E3(pays, nv) trié par pays

Initialement Place1 est partitionné par photoID.

由上之 groupBy . 然後

échange entre machine (Exchange hashpartitioning) : #Machine 15 - 9 machine

Nº anonymat:

count \Rightarrow Tagg \rightarrow count page 3

Stage 1

Fonction partielle : regrouper par pays et agréger pour garder la liste des villes distinctes et trier par pays

On obtient A(pays, listeV) trié par pays

Partial HashAgg \Rightarrow Partial list

Chaque partition peut contenir 50 valeurs de pays. On suppose que la répartition par pays est uniforme entre les 10 machines, donc chaque machine gère 5 pays et reçoit 5 tuples des 9 autres machines

Echange de $5 * 9 = 45$ éléments (pays, listeV). Soit un total de $10 * 45 = 450$ éléments

Autrement dit, alors on peut appliquer le facteur $9/10$ car chaque machine M_i « garde » un dixième des éléments qui n'ont pas besoin d'être transférés, donc $50 * 10 * 9/10 = 450$ éléments transférés (broadcast)

Répartition par 10 intervalles de valeurs sur le domaine de pays (1 intervalle destiné à une machine)

Stage 2

Fonction complète : par fusion des morceaux (shuffle read) reçus car ils sont déjà triés. La fusion doit éliminer les villes en double pour chaque pays. Puis compter le nombre de villes

partial (HashAggregate) \Rightarrow Exchange Hashpartitioning \Rightarrow HashAgg

Rmq : autre solution A n'est pas trié par pays, et faire le tri dans la fonction complète.

Proposer une expression E3b donnant un résultat équivalent à celui de E3. E3b doit utiliser la fonction distinct() mais pas la fonction countDistinct().

\neq schema équivalent (1) (2) (3)

(1)

(2)

(3)

E3b = Place1.select('pays', 'ville').distinct().groupBy('pays').agg(count(1).as("nv")).orderBy("pays")

Rmq : plusieurs réponses possibles : count(1) ou count(*) ou count(ville)

E3b = Place1.select("pays", "ville").distinct().groupBy("pays").agg(count(1).as("nv")).orderBy("pays")

Décrire l'exécution de E3b. Préciser les calculs (partiels/complets) et les transferts

Stage 1

Fonction partielle : projeter sur pays, ville obtenir l'ensemble des (pays, ville) distincts

function = []

Répartition par (pays, ville) par hachage (car le distinct() provoque sur une répartition par hachage quelles que soient les opérations qui sont traitées à la suite) Hashagg \Rightarrow Exchange \Rightarrow Hashagg

Stage 2 distinct 1 \rightarrow 2

Fonction pour compléter le distinct : rassembler les (pays, ville) identiques provenant de différentes partitions

On obtient A(pays, ville) sans doublon.

Stage 3 count 2 \rightarrow 3

Fonction partielle (pour le group by+agg) agréger par pays et count pour obtenir (pays, nv) puis tri par pays

Stage 4 order by 3 \rightarrow 4

Répartition par intervalle du domaine de pays

Fonction pour compléter le group by+agg : fusion de morceaux reçus et sum des nv pour chaque pays

Autre solution :

Faire la 1ere répartition par (pays, ville) mais par intervalle de pays pour éviter la 2^e répartition

Question 4 . Jointure entre Visite et Intérêt. On veut associer les photos avec les points d'intérêt ayant les mêmes coordonnées (lat, lon) :

N° anonymat:

E4 = Visite.join(Intérêt, ["lat", "lon"]).select("photoID", "POI", "catégorie").

Décrire l'exécution de E4 en effectuant une jointure par hachage parallèle

Répartition de Visite : $10\ 000 * 9/10 = 9\ 000$ tuples transférés

Répartition de Intérêt : $1000 * 9/10 = 900$ tuples transférés

P₁₆ Visite

Décrire l'exécution de E4 en effectuant une jointure par broadcast ou diffusion de la plus petite des deux tables

Diffusion de Intérêt qui sera « répliquée » sur les 10 machines.

$1000 \text{ tuples} * 10 \text{ machines} * 9/10 = 9\ 000$ tuples transférés

Sur chaque partition de visite, jointure avec Intérêt et projeter sur (photoID, POI, catégorie)

每台机器 \Rightarrow 有 lat
lon
 $\frac{9}{10}$ 台机器的 machine
新的 partition

自己单独传播 \Rightarrow
Internet

