Faculté sciences & sciences de l'ingénieur
Université Bretagne Sud

Master d'Informatique

UE INF2245

Calcul haute performance pour le «Big Data»

Sujet d'examen terminal

(première session)

Durée : 2 heures

F. Raimbault, N. Courty

mai 2021

Avertissement : Il sera principalement tenu compte dans la notation de la qualité de rédaction, de votre capacité à abstraire une solution et de la rigueur avec laquelle vous la décrivez.

Exercice 1 : MapReduce (7 points)

Les réponses à cet exercice doivent être exprimées de manière algorithmique indépendamment de tout langage de programmation.

On utilise comme jeu de données la base de signaux AISUBS) Le format des fichiers de signaux AIS collectés est décrit sur l'ENT dans UN DOCUMENT DE RÉFÉRENCE accessible depuis la page du cours. Écrivez une, ou un enchaînement de plusieurs, tâches(s) *MapReduce* pour réaliser chacune des opérations suivantes.

Vous prendrez soin de bien préciser les spécifications des fonctions *map* et *reduce* en terme d'entrée et de sortie de couples clé-valeur. Vous détaillerez les opérations effectuées dans les fonctions elles-mêmes et les mécanismes utilisés (exemple : compteurs). Vous discuterez des éventuels problèmes de performance qui pourrait survenir et des optimisations possibles.

1. Nettoyage des données : suppression de toutes les données dupliquées et des données manquantes ou incorrectes, conversion des dates au format EPOCH (Heure Unix).
   La lecture des fichiers AIS est réalisée à l'aide d'un `TextInputFormat` qui renvoie des couples (position dans le fichier, chaîne contenant un signal AIS). Vous disposez

d'une fonction `getFieldsMap(line)` qui permet d'extraire d'une chaîne la valeur des champs listés dans le document de référence [1].

Pour la conversion des dates vous disposez d'une fonction `isoTimeToEpoch(time)` qui convertit une date au format ISO en une date au format EPOCH (un entier long donnant le nombre de secondes écoulées depuis le 1 janvier 1970).

On considère comme données dupliquées les signaux reçus avec les mêmes valeurs de TIME et de MMSI [2].

La vérification des données porte sur la valeur de LATITUDE qui doit être comprise dans l'intervalle $[90, 90]$ et sur celle de LONGITUDE qui doit appartenir à l'intervalle $[-180, +180]$.

Au final, le nettoyage produit un fichier HFDS au même format que le fichier d'origine à l'aide d'un `TextOutputFormat`.

Cette opération devra également afficher le pourcentage de données supprimées car en doublon, le pourcentage de données supprimées car incorrectes et le nombre de données conservées par rapport au nombre de données lues.

2. Construction d'une base des navires : mémorisation des données propres à chaque navire dans un fichier HDFS.

   Cette opération prend en entrée le résultat de l'opération précédente de nettoyage des données. Elle produit en sortie, à l'aide d'un `TextOutputFormat`, un fichier HFDS contenant pour chaque navire (et exclusivement pour les navires [3]) son MMSI, son NAME, son TYPE, et sa dernière position connue.

   Cette opération devra également afficher le nombre de navires présents dans la base.

Exercice 2 : Spark (6 points)

Les réponses à cet exercice doivent être exprimées de manière algorithmique indépendamment de tout langage de programmation.

Cet exercice reprend le même jeu de données que dans l'exercice précédent et les mêmes questions. Les solutions sont à exprimer en terme d'actions et de transformations disponibles en *Spark* sur des *RDD* à définir précisément.

Les fonctions `getFieldsMap(line)` et `isoTimeToEpoch(time)` de l'exercice précédent sont également utilisables ici.

1. Nettoyage des données : suppression de toutes les données dupliquées et des données manquantes ou incorrectes, conversion des dates au format EPOCH (Heure Unix).
2. Construction d'une base des navires : mémorisation des données propres à chaque navire dans un fichier HDFS.

1. A la manière de la méthode `XmlUtils.getAttributesMap()` sur le jeu de données STACKOVERFLOW.

2. Ce cas de figure correspond à un même signal AIS reçu par deux stations différentes qui enregistrent toutes deux le signal.

3. Comme indiqué dans le document de référence, un navire possède un MMSI à 8 chiffres qui commence par un chiffre entre 2 et 7 inclus.

Exercice 3 : CUDA (7 points)

1. Rappelez en quelques lignes les principes de la programmation en CUDA : quels avantages ? Dans quels cas de figure peut on vouloir utiliser ce modèle de programmation ? Quels sont les pré-requis ?
2. Soit le programme suivant :

```
import pycuda.autoinit
import pycuda.driver as drv
import numpy

from pycuda.compiler import SourceModule
mod = SourceModule("""
__global__ void my_function(float *dest, float *a, float *b, int size)
{
  const int i = threadIdx.x;
  if (i<size){
    dest[size - i - 1] = 2*a[i] - b[i];
  }
}
""")

my_function = mod.get_function("my_function")

a = numpy.array([0,1,2,3,4,5,6,7,8,9])
b = numpy.array([9,8,7,6,5,4,3,2,1,0])

dest = numpy.zeros_like(a)
my_function(
        drv.Out(dest), drv.In(a), drv.In(b), 10)
        block=(10,1,1), grid=(1,1))

print(dest)
```

   Quelle est la sortie de l'exécution de ce programme ? Cette sortie est elle déterministe (toujours le même résultat) ? Justifiez votre réponse.
3. Même question si on change l'appel du noyau par :

```
my_function(
        drv.Out(dest), drv.In(a), drv.In(b), 10)
        block=(5,1,1), grid=(2,1))

print(dest)
```

4. Modifiez le noyau `my_function` de manière à ce que le programme s'exécute de la même manière pour l'appel précédent que dans la question 2.