Premier pas avec Hadoop MapReduce

Guitard Alan Willian Ver Valen Paiva

23 septembre 2016

Table des matières

1 Introduction

Ces exercices avaient pour objectif de nous apprendre les bases de la programmation _à grande échelle en nous soumettant des exercices de parsing basique sur une fichier massif, exercices dont les programmes-solutions prendrait énormément de temps et de mémoire CPU à faire sur une seule machine, tandis qu'ils ne prendraient que quelques secondes sur un cluster.

2 Exercice 1 : Filtrage

L'objectif de cet exercice est de programmer un mapper et un reducer pour lire le fichier massif worldcitiespop.txt de taille 144MB, une liste de ville, leur population et d'autres informations, qui est structuré comme suit :

— une première ligne décrivant le contenu des lignes, dont les champs sont séparés par des virgules

— Les champs de chaque villes structurés comme décrit dans la première ligne.

et d'écrire seulement les villes qui ont un champ de population valide dans un fichier output.txt.

2.1 Le Mapper

Le mapper se charge de lire et de filtrer les bonnes villes. Cela tient en quelques lignes :

```
String[] line = value.toString().split(",");
if(!line[4].equals(""))
{
    context.write(value, new IntWritable(1) );
}
```

La variable `value` est un paramètre qui est un objet `Text` contenant une ligne du fichier. Cette fonction est donc appelé une fois par ligne. La variable `value` est séparé en un tableau en fonction des virgules et la case numéro 4 est testée. Si elle est vide, la ville ne contient pas d'informations de population et n'est donc pas écrite.

2.2 Le Reducer

Le reducer se limite à ces deux lignes :

```
for(IntWritable value : values)
        context.write(key, null );
```

Cela a pour effet d'écrire dans le fichier les lignes renvoyés par le mapper.

3 Exercice 2 : Compteurs

Pour cet exercice, nous allons reprendre le code écrit pour l'exercice 1. Le but de celui-ci est d'utiliser la fonction `getCompteur(String,String)` pour compter trois valeurs : `nb_cities`, qui est le nombre de villes valide, `nb_pop`, le nombre de villes avec une population renseignée et `total_pop` qui est le nombre d'habitants de toutes les villes. Ces compteurs doit être réunis dans un groupe que l'on nommera "WCP".

3.1 Le Mapper

La difficulté ici était d'apprendre à se servir des compteurs. La chose à comprendre était que les compteurs sont instanciés dans la fonction `getCompteur(String namegroup,String namecompteur)` si ses paramètres ne sont pas connus du contexte. Voici l'utilisation que nous en avons fait :

```
String[] line = value.toString().split(",");
if(!line[4].equals("Population"))
{
    if(!line[4].equals(""))
        {
            context.write(value, new IntWritable(1) );
            context.getCounter("WCP","total_pop").increment(Integer.
                parseInt(line[4]));
            context.getCounter("WCP","nb_pop").increment(1);
        }
    context.getCounter("WCP","nb_cities").increment(1);
}
```

Nous appelons donc les compteurs au bon endroit en fonction de ce que l'on veut compter. Le premier paramètre est le nom du groupe, "WCP". Comme ce groupe n'est pas connu par le contexte, il est créé. Le deuxième paramètre est le nom du compteur et également, comme il n'est pas connu, il est créé.

Pour afficher leur valeur à la fin du programme, rien n'est besoin de coder, yarn l'affiche automatiquement à la fin du programme.