Premier pas avec Hadoop MapReduce: Deuxième partie

Guitard Alan Willian Ver Valen Paiva

1er octobre 2016

Table des matières

1 Introduction

Ces exercices est une suite de l'ancien TP sur le mapping-reducing pour nous apprendre d'autres outils pour manier ce framework de MapReduce. Nous partirons des anciens exercices qui consistait à filtrer les lignes du fichier qui décrit les villes, afin de ne pouvoir travailler que sur les villes valides.

2 Exercice 1 : Histogramme

L'objectif de cet exercice est de classer les villes que l'on as sélectionné dans des classes en fonction de leur population. L'idée est de les classer en fonction du logarithme en base 10 de leur population.

2.1 Le Mapper

Pour que cela puisse fonctionner, il faut arrondir la valeur obtenue à l'unité la plus proche, et utiliser cette valeur en tant que puissance de 10, comme suit :

```
int log_pop = (int)Math.round(Math.log10(Double.parseDouble(line[4])));
int log_pop10 = (int)Math.pow(10,log_pop);
```

Une première classe sera donc celle des villes dont la population va de 0 à 10, la deuxième de 10 à 100, et ainsi de suite.

Pour terminer, cette valeur traduite en objet `Text` et envoyer dans le contexte par la fonction `context.write(result, new IntWritable(1));`.

2.2 Le Reducer

Le Reducer pour cet exercice est simple, il est chargé d'incrémenter le nombre de ville pour chaque groupe. Nous itérons sur le paramètres values en les ajoutant à une variable `sum`. Cette variable qui sera écrit dans le fichier de sortie.

3 Exercice 2 : Résumé

L'exercice est de résumer chaque classe par la population moyenne, maximum et minimum des villes qui la constitue. Pour ceci, un objet implémentant l'interface `Writable` doit être écrit, afin de pouvoir passer les valeurs à écrire dans le fichier en un seul objet.

3.1 Le Mapper

L'objet `AvgMaxMinWritable` va remplacer l'objet `IntWritable` qu'on passait à la fonction `context.write` qui ne servait pas jusque là. Cet objet contient donc les données membres contenant la population, le compteur (qui remplace la variable `sum` de l'exercice 1, et les variables avg, max, et min qui sera utilisé dans le reducer.

Ici, dans le mapper, seul la population est donnée à l'objet `AvgMaxMinWritable` qu'on instance dans le même temps :

```
context.write(result, new AvgMaxMinWritable(1,Integer.parseInt(line
    [4])));
```

3.2 Le Reducer

Ici, on itère sur un objet `values` qui contient maintenant des objets de type `AvgMaxMinWritable`. Dans cette itération, on traite les données. Ce traitement est donc incrémenter le compteur avec la valeur dans `value.getCounter()`, incrémenter avg avec `value.getPop()`, et définir max ou min si la valeur dans `value.getPop()` est plus grande ou plus petite, respectivement.

A la fin de l'itération, ces valeurs sont écrit dans `AvgMaxMinWritable result;`, (avg est divisé par le nombre de ville de la classe pour avoir la moyenne) et l'objet est alors écrit dans le contexte.

4 Exercice 3 : Paramétrage

Cet exercice nous propose de permettre d'ajouter un fichier de paramètre partagé entre les nœuds du cluster. En l'occurrence, nous allons permettre à l'utilisateur de changer la base du logarithme afin qu'il est un contrôle sur les classes construites.

4.1 Le Mapper

La fonction `setup()` doit être écrite afin de récupérer le fichier cache que l'on as ajouté dans le job par la ligne :

```
job.addCacheFile(new Path(args[2]).toUri());
```

Le chemin local du fichier xml que décrit les paramètres à intégrer est donc à envoyer en troisième paramètre dans la ligne de commande.

```
protected void setup(Context context)
{
    URI[] files = context.getCacheFiles(context.
        getConfiguration());
    DataInputStream strm = new DataInputStream(new
        FileInputStream(files[0].getPath()));

    ConfCachFiles cache = new ConfCachFiles();
    cache.readFields(strm);

    strm.close();
}
```

La fonction ci-dessus est celle qui charge les données du fichier cache dans l'objet `ConfCachFiles` créé pour l'occasion. On peut alors récupérer cette valeur dans le mapper par le contexte et la configuration :

```
int steplog = context.getConfiguration().getString("steplog","10");
```

Cette ligne récupère la valeur steplog du fichier xml, ou renvoie 10 si cette valeur n'existe pas.

Pour changer la base du logarithme, la formule est celle-ci :

`Math.log(num) / Math.log(base)`

, où la base est donc la valeur que l'on as récupéré dans le fichier xml.

4.1.1 Le Reducer

Aucun changement n'as été fait dans le reducer.