


BIG DATA  
2016-2017

## TP N°6: PATTERN MAP REDUCE

TOP-K

YAHYA BACHIR  
NAJI ZAOU



Dans ce TP on va utiliser le pattern TopK dans un programme Map-Reduce. Ce pattern consiste à trouver les K plus grands éléments d'un ensemble on va l'utiliser pour trouver les 10 villes les plus peuplées du monde, on va travailler sur notre fameux fichier : « worldcitiespop.txt »

### Exercice 1 :

Le but de cette exercice est d'implémenter le pattern Top K, pour cela on va créer un mapper et un reducer afin de sortir les dix villes les plus peuplées du monde.

#### *Class TP6Mapper :*

Notre Mapper va recevoir une partie ou bien un certain nombre de lignes de notre fichier de départ « worldcitiespop.txt », puis il va nous donner le top 'K' des villes les plus peuplées parmi celles qu'il a reçu.

Pour cela on a besoin d'un 'K' fournis par l'utilisateur puis on va utiliser un TreeMap<> qui va prendre comme clé la population et comme valeur les informations de la ville, donc on aura un ensemble de villes triées par leurs population « la clé » du fait que TreeMap<> implémente l'interface SortedMap<>. Donc la ville qui a la plus grande population va être stockée la dernière et celle qui a la plus petite population sera la première. Le mapper place chaque entrée (nouvelle ville) dans notre TreeMap<> puis test si sa taille dépasse le 'K' fournit par l'utilisateur, si c'est le cas, alors on enlève la valeur qui a la clef la plus petite (la 1<sup>ère</sup> dans notre TreeMap<>).

A la fin notre TreeMap va contenir le top 'K' villes parmi celles qu'il a reçu.

*Méthode setup ()* : cette méthode a comme rôle d'initialiser la variables 'K' par la valeur donnée par l'utilisateur.

*Méthode cleanup ()* : dans le cleanup () on va écrire les valeurs de notre TreeMap<>, vu que le traitement de Map sera achevé, alors notre TreeMap contiendra que le TopK des villes (que le mapper a lu).

#### *Class TP6Reducer :*

Notre Reducer va recevoir un certain nombre de liste contenant les tops 'K' villes données par chaque Mappeur et il va nous donner le top 'k' des villes les peuplées parmi eux donc le top 'K' villes peuplées de «worldcitiespop.txt »

Le Reducer va presque faire le même travail que le Mappeur, il va travailler sur le même 'K' donné par l'utilisateur et on va utiliser un TreeMap<>. Comme avec le Mapper.

À la fin notre TreeMap va contenir le top 'K' villes parmi celles qu'il a reçu à partir de tous les Mappeurs.

### *Class TP6Combiner :*

Le Combiner peut être considéré comme un mini-Reducer, la plus grande différence est que chaque ordinateur dans notre cluster a son propre Combiner, contrairement au Reducer, dans notre cas on travaille avec un seul dans tout le cluster. Le rôle du Combiner est de diminuer la charge de travail au Reducer, par exemple sans les combinateurs le Réducteur va recevoir (le nombre d'ordinateurs) \* (le Nombre de Mappeurs par ordinateur) Liste dont il a besoin de trier et extraire Top'K' des villes d'eux, par contre avec les Combinateurs notre Reducer va recevoir (le nombre d'ordinateurs) Listes qui ont besoin d'être triés et puis l'extraction des Top'K' villes d'entre eux sera donc plus rapide

Le combinateur va faire le exact même traitement que le Réducteur.

N.B. : pensez à regarder la JavaDoc.