Bérénice Faltrept
Kevin Baptista

M2 Génie Logiciel
Rapport TD6 : Top-K Pattern

Notre programme se lance de la forme suivante : yarn jar tp6-mapreduce-0.0.1.jar <file_input> <rep_output> <top_size> <hadoop_right_name>

On demande un nom de répertoire temporaire car nous détruisons ce répertoire ainsi que son contenu, or hadoop bloque le programme du fait de la préexistence d'un dossier du même nom.
Le dernier argument est votre identifiant hadoop pour ouvrir les fichiers, ne sachant pas les droits que vous avez donnés dans votre propre environnement.

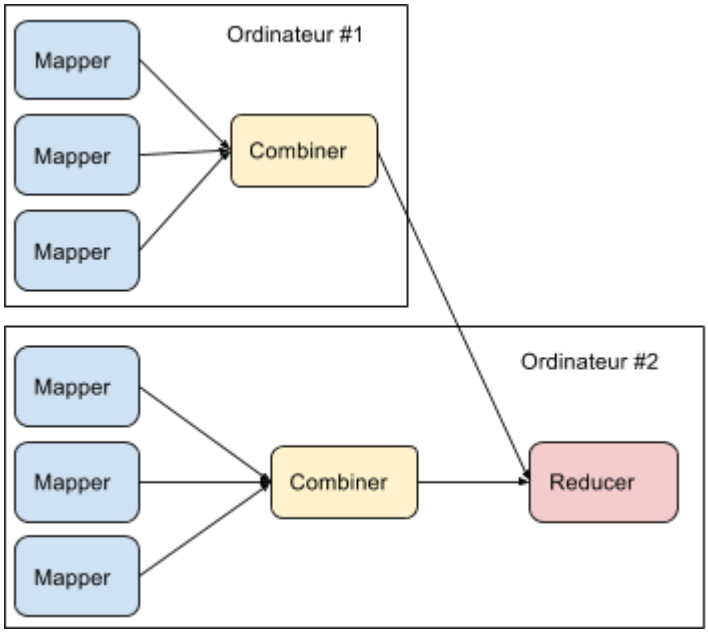
Exercice 1 :

Nous avons repris l'exemple de Top-K présent dans le cours sur les Patterns en l'adaptant simplement au traitement des villes que nous avions en entrée. L'usage du TreeMap nous a inquiété au premier abord, du fait de son occupation en mémoire mais puisque nous nous assurons que sa taille ne dépasse pas celle du Top, cela n'est finalement pas problématique.

Exercice 2 :

Nous avons intégré un Combiner dont l'intérêt est de répartir le traitement entre les machines afin de transmettre moins de valeurs à l'unique Reducer de ce TP.

Le traitement est fondamentalement le même que dans le Mapper puis le Reducer.



Exercice 3 :

Nous avons eu un problème pour passer en paramètre le nombre d'éléments à conserver. Notre problème était qu'on définissait le paramètre après avoir défini le job. Nous avons donc défini l'objet configuration (en lui donnant l'argument définissant le nombre de villes à conserver) puis l'avons donné au job lors de son instanciation.

Nous avons chronométré nos résultats à l'aide de la commande time :

100		1 000		10 000	
real	0m15,405s	real	0m15,376s	real	0m15,335s
user	0m4,652s	user	0m4,024s	user	0m4,708s
sys	0m0,156s	sys	0m0,116s	sys	0m0,156s

Nous avons donc un traitement en temps linéaire grâce à la bonne répartition de travail entre les différentes machines et les différentes parties du programme (mappeur, combiner, reducer).