Compte rendu du projet Map-Reduce

Gouttefarde Léo

Vendredi 16 Décembre 2016

Table des matières

1 Prise en main

1.1 Exécution locale

1. "Map input records" correspond au nombre de lignes du fichier d'entrée à traiter.
   "Map output records" correspond au nombre de mots du fichier d'entrée.
2. "Reduce input records" correspond au nombre de mots fournis à l'étape reduce.
   Ainsi il s'agit du nombre total de mots ("Map output records") moins le nombre de mots retirés lors de l'étape combine.
3. "Reduce input groups" correspond au nombre de mots différents présents à l'étape reduce.

1.2 Premier contact avec HDFS

Dans HDFS, le chemin vers mon répertoire personel est /user/lgout.

1.3 Exécution sur le cluster

Le nombre de "splits" lus sur HDFS est 5. Ce compteur correspond au nombre de fichiers d'entrée.

1.4 Combiner et nombre de reducers

1. Il n'y a aucune différence entre les deux répertoires de résultats
   C'est normal car un combiner est un mini-reducer placé à la sortie du map et qui permet de réduire le nombre de couples sans affecter le résultat.
   En effet la correction du programme ne doit pas en dépendre.
2. Les valeurs "Combine input records" et "Combine output records" permettent de vérifier que le combiner a bien fonctionné car elles indiquent le nombre de valeurs traitées par le combiner.
   De plus on observe que la valeur de "Reduce input records" a bien été réduite avec le combiner.
3. La différence entre "Combine input records" et "Combine output records" correspond au nombre de valeurs qu'il a été possible de retirer en les traitant avec le combiner (et donc au gain obtenu).
   En effet on peut facilement vérifier que cette différence est égale à la différence entre les valeurs "Reduce input records" avec et sans combiner.
   Sans combiner, on a "Reduce input records" = 421739.
   Avec combiner, on a "Combine input records" = 421739, "Combine output records" = 85301 et "Reduce input records" = 85301.
   On voit ainsi bien que "Reduce input records" sans combiner est égal à "Combine input records" avec combiner.
   Avec le combiner, "Reduce input records" est égal à "Combine output records" et donc la différence précisée précédemment est bien égale (différence de 336438).

Pour finir, le mot le plus utilisé dans "Les misérables" est 'de' qui est utilisé 16757 fois.

2 Top-tags Flickr par pays en 1 job

2.1 Map et Reduce

Voir l'implémentation Java réalisée (fichier Question2_1.java).

2.2 Combiner

Question préliminaire

Pour pouvoir utiliser un combiner, il faudrait des couples avec la clé fournie par la classe Country et une valeur contenant le tag concerné et son nombre d'occurrences.

Un exemple de couple pourrait être 'FR' → ('france', 563).

En Java on utilisera le type Text pour les clés et le type StringAndInt pour les valeurs.

Tags les plus utilisés en France

Avec K = 5, une exécution MapReduce sur le fichier /data/flickr.txt donne le résultat suivant pour la France : FR [france : 563, bretagne : 67, spain : 115, españa : 71, europe : 75].

On observe que les tags les plus utilisés en France sont 'france', 'spain', 'europe', 'españa' et 'bretagne'.

Question finale

Dans le reducer, nous avons une structure en mémoire dont la taille dépend du nombre de tags distincts. C'est un problème car dans un contexte de traitement d'un grand nombre de données (big data) par exemple la mémoire sera insuffisante.

Ainsi il faudrait pouvoir utiliser autre chose qu'une simple structure HashMap, afin d'éliminer les tags le moins présents dès que possible.

3 Top-tags Flickr par pays en 2 jobs

Question préliminaire

Premier job

Les clés / valeurs en entrée sont respectivement de type LongWritable et Text pour lire le fichier texte d'entrée (comme précédemment).

Les clés / valeurs intermédiaires sont respectivement de type Text et IntWritable. En effet le type Text permet de stocker la concaténation du pays et du tag concerné comme clé, et le type IntWritable permet de stocker le nombre d'occurrences.

Les clés / valeurs en sortie sont respectivement de type Text et IntWritable là encore, en effet le reducer va additionner tous les nombres d'occurrence et en calculer la somme pour chaque clé pays/tag. On utilise ce reducer également comme combiner.

Second job

Les clés / valeurs en entrée sont respectivement de type Text et IntWritable pour lire les valeurs écrites en binaire par le reducer du job 1.

Les clés / valeurs intermédiaires sont respectivement de type Text et StringAndInt. En effet le type Text permet de stocker le pays comme clé, et le type StringAndInt permet de stocker le tag et son nombre d'occurrences total.

Les clés / valeurs en sortie sont respectivement de type Text et MinMaxPriorityQueue, en effet pour chaque clé de pays le reducer va utiliser une file de priorité sur les tags pour n'en garder que les K plus populaires.

Le combiner est içi identique au reduce sauf qu'on écrit chaque tag en sortie, au lieu de la file de priorité.

Question finale

Non, on a pas la garantie d'obtenir toujours le même résultat d'une exécution à l'autre, car l'ordre des tags lors de l'étape reduce dépend des différents mappeurs.

Ainsi on a pas la garantie d'obtenir toujours le même résultat, il faudrait également trier par tag en plus du nombre d'occurrences pour garantir un résultat déterministe. Notre résultat dépend de l'ordre de récupération des éléments depuis les différents mappeurs.

En revanche, dans les tests en local et sur le cluster hadoop, nous obtenons les mêmes résultats à chaque exécution, certainement en raison de conditions d'exécutions très similaires.