Université de Bordeaux I

Programmation large échelle

Compte rendu du TP 6
Map Reduce Patterns TopK

Réalisé par :

GAMELIN Antoine
LASSOUANI Sofiane

Université Bordeaux 1 (2016–2017) – Master Génie Logiciel (UE PLE)

TABLE DES MATIÈRES

Chapitre 1

Introduction

1.1 Introduction

Le but de ce TP est de mettre en oeuvre le pattern Top K vue en cours. Ce pattern consiste à sortir les K plus (grande/petite) valeur d'un ensemble d'entrée.

Chapitre 2

Exercice sur le Pattern TopK

2.1 Introduction :

Cette premiére partie du TP est consacré au Top K,qui sert a rerouver les k villes les plus peuplés. Nous utiliserons le fichier **worldcitiespop.txt**, de type CSV, qui contient les noms de ville avec leurs populations et des informations annexes.

2.2 Exercice 1 :

2.2.1 CityWritable

Pour bien mener ce TP nous avons décider de créer un nouveau type de donnée qui implemente writable. Cette structure permet d'avoir un accès plus rapide aux différents champs du fichier csv et qu'on ne soit pas obligé de split à chaque fois la chaine de caractère.

2.2.2 Mapper

Concernant le Mapper, nous avons utilisé la fonction setup afin d'initialiser un arbre binaire TreeMap en java. L'avantage du TreeMap par rapport au SortedArray, c'est que lorsqu'on travaille avec de nombreuses données il est plus rapide à l'execution.

Le TreeMap reçoit comme clef le nombre de population et en valeur, un CityWritable avec toutes les informations.

Le mapper, ajoute chaque ville au TreeMap, et quand le nombre d'élément est supérieur à k, nous supprimons à chaque fois l'element avec la plus petite population.

Dans ce TP le mapper n'envoi aucun message, c'est le clean up qui lors de son appel retourne les 10 plus grandes villes.

Afin de tester notre job nous avons initialisé la variable k à 10 sans passer par la configuration.

2.2.3 Reducer

Le reducer récupère le résultat des différents mappers et fonctionne comme le mapper il possède un TreeMap et supprime à chaque fois la ville avec la plus faible population quand la taille du TreeMap dépasse k.

2.3 Exercice 2 :

Dans cette partie il nous est demandé d'écrire un combiner, qui va alléger la charge au reducer. En effet sur un datanode, plusieurs mapper peuvent être executé, est au lieu d'envoyer les directement toutes les données au reducer, nous utilisons un combiner pour envoyer moins de données sur le reseaux.

2.3.1 Combiner

Le but du combiner est presque égale au reducer, la seule différence, c'est qu'en sortie au lieu de renvoyer un <NullWritable,Text> nous renvoyons un <NullWritable, CityWritable> pour que le réduceur puisse interpréter le résultat.

2.4 Exercie 3 :

Dans cette partie nous avons ajouté un paramètre au lancement du programme permettant de définir le nombre k de ville plus grande.

Chapitre 3

Conclusion

Ce tp nous a permis de mettre en pratique le pattern Top K étudié en cours .

Ce pattern est très simple d'implémentation et permet de traiter les fichiers très rapidement grâce à Hadoop.

Nous avons cependant rencontrés un petit problème, le reducer ne s'executais pas car l'entête de la classe réducer n'était pas identique que la méthode réduce.