Bérénice Faltrept
Kevin Baptista

M2 Génie Logiciel
Rapport TD7 : Pattern Join

Partie 1 : Reduce-Side Join

usage

yarn jar tp7-mapreduce-0.0.1.jar ReduceSideJoin /user/bfaltrep/cities.txt /user/bfaltrep/region.txt /user/bfaltrep/tp7
va générer les couples Région-Ville dans le fichier /user/bfaltrep/tp7/part-r-00000

Reducer

Contrairement à la possibilité énoncé en TP pour le traitement du **Reducer**, qui consistait à créer une List contenant les Villes associés à une région, nous avons choisi de faire plus de calcul et de préserver notre espace mémoire :
nous faisons un premier parcours de notre Iterable pour trouver le nom de la région puis nous faisons un second parcours afin d'écrire les valeurs finales.

Partie 2

usage

yarn jar tp7-mapreduce-0.0.1.jar ScalableJoin /user/bfaltrep/cities.txt /user/bfaltrep/region.csv /user/bfaltrep/tp7
va générer les couples Région-Ville dans le fichier /user/bfaltrep/tp7/part-r-00000

Organisation des données dans le TaggedKey

Nous avons fait le choix de placer trois variables dans la classe **TaggedKey** afin de faciliter les différents accès aux données nécessaires :

- **String** _country : contient le nom du pays du contenu associé à la clé ;
- **Sring** _region : contient l'ID de la région du contenu associé à la clé ;
- **boolean** _side : si le contenu associé à la clé est une région, vaut ***false*** et ***true*** sinon ;

Système de comparaison

Chaque **WritableComparator** a besoin d'une méthode de comparaison spécifique dans le TaggedKey :

JoinGrouping	JoinSort
A besoin d'une comparaison entre les clés naturelles uniquement.	A besoin d'une comparaison entre les clés naturelles mais aussi de distinguer la région de ses villes pour chaque clé.

	Comme elle utilise tout les champs du **TaggedKey**, nous avons choisit de Override la méthode de comparaison par défaut.
public int compareTo(TaggedKey tk)	public int compareTo(Object o)

Reducer

Nous devons parler d'une incompréhension que nous avons rencontrés lors de ce TD :
Nous voulions vérifier l'existence de la région dans les valeurs associées à une Key :

- On a donc utilisé la valeur **_side** de **key** pour déterminer si on travaille sur une ville ou une région.
- Si notre iterable est actuellement sur une région, on récupère le nom de la région dans une chaine de caractère.
- On itère ensuite sur les valeurs pour écrire dans le context le nom de la région suivie du nom de la ville.

Nous constatons que cela fonctionne, donc les multiples instances de clés associés aux valeurs sont conservées et accessibles via l'Iterator. Cela entre en conflit avec le concept d'un programme MapReduce, pour lequel le but du Reducer est de traiter un ensemble de valeurs pour UNE clé.
Nous accédons aux instances de clés via l'argument *values*, et non par l'argument *key* passé en paramètre.
Quand nous avons écrit ce code, il nous paraissait juste, mais après relecture, nous nous demandons pourquoi le MapReduce nous permet un tel accès indirect qui ne nous semble pas très régulier.
Y aurait-il eu une solution plus propre pour vérifier l'existence de notre région dans le Reducer (et ainsi garantir que nous opérons un innerJoin) ?