Rapport Hadoop MapReduce GROUPE KERGALE CARRON BOURGEOIS AUBERT

Ma démarche pour le programme map/reduce a été la suivante : Je me suis inspiré des programmes des Graphs et d'analyse de vente vu respectivement dans le tp2 et tp1. Sachant que le tp1 analyse des ventes est très similaire au projet, j'ai utilisé son programme comme Template surtout pour le reduce ;

Map

Pour la partie map, j'ai commencé par faire en sorte que la première ligne du tableau co2.csv ne soit pas prise en compte.

```
if (value.toString().contains("Marque")){
    return;
}

String node = value.toString();
node = node.replaceAll("\\u00a0"," ");

String[] splitted_node = node.split(",");
```

Chaque node va venir représenter une ligne, et le replaceAll remplace tous les caractères invisibles par des espaces. Et pour venir récupérer chaque colonne de CO2.csv, on split notre ligne (« node »).

Pour chaque colonne, on trie les anomalies de CO2.csv pour que les valeurs soient « propres ».

```
42      // Trie des colonnes
43
44      String[] splitted_space = splitted_node[1].split("\\s+");
45      String colMarque = splitted_space[0];
46      colMarque = colMarque.replace("\"", "");
47
48      String colbonusMalus = splitted_node[2];
49      colbonusMalus = colbonusMalus.replaceAll(" ", "").replace("€1", "").replace("€", "").replace("\"", "");
50      if (colbonusMalus.equals("150kW(204ch)") || colbonusMalus.equals("100kW(136ch)"))
51      {
52          return;
53      }
54      if (colbonusMalus.length() == 1){
55          colbonusMalus="0";
56      }
57
58      String colRejet = splitted_node[3];
59
60      String colCout = splitted_node[4];
61      String[] colCout_splitted = colCout.split(" ");
62      //pour trier les valeurs à 2 ou 3 virgules
63      if(colCout_splitted.length == 2){
64          colCout = colCout_splitted[0];
65      } else if(colCout_splitted.length == 3){
66          colCout= colCout_splitted[0] + colCout_splitted[1];
67      }
```

Chaque colonne est récupérée dans le splitted_node. Et on procède à un nettoyage des valeurs comme à la ligne 49 pour les bonus/malus. Le if à la ligne 50 permet de supprimer les « fausses colonnes » comme à la ligne 14 et 15 de CO2.csv qui sont apparues après le split.

```
context.write(new Text(colMarque), new Text(String.valueOf(Integer.parseInt(colbonusMalus)) + "|" +  String.valueOf(Integer.parseInt(colRejet)) + "|" + String.valueOf(Integer.parseInt(colCout))));
```

Après le fonction map qui a permis de trier et reconstruire le tableau de valeurs, on passe à la fonction reduce

Reduce

La fonction reduce est inspiré de la fonction reduce du tp1 Analyse des ventes. Dans l'utilisation de la boucle while avec l'itérateur. Cela va nous permettre pour chaque marque de calculer la valeur moyenne des bonus/malus, rejetsCO2 et Coût par marque.

```
String bonusMalus;
String rejet;
String cout;
int c=0;
int bonusMalusSum = 0;
int rejetSum = 0;
int coutSum = 0;

Iterator<Text> i = values.iterator();
while(i.hasNext()) {
    String node = i.next().toString();
    System.err.print(key);
    System.err.print("   ");
    System.err.println(node);

    String[] splitted_node = node.split("\\|");
    bonusMalus = splitted_node[0];
    rejet = splitted_node[1];
    cout = splitted_node[2];

    bonusMalusSum += Integer.parseInt(bonusMalus);
    rejetSum += Integer.parseInt(rejet);
    coutSum += Integer.parseInt(cout);
    c++;
}

context.write(key, new Text(bonusMalusSum/c + "\t" + rejetSum/c + "\t" + coutSum/c));
```

La clé est la marque.

On sépare les lignes en plusieurs colonnes, et on calcule la somme pour chaque colonne. A chaque itération, le compteur est incrémenté de 1 et va nous servir à la fin à calculer la moyenne pour chaque colonne (Bonus/Malus, RejetsCO2, Coûts) d'où Somme / compteur .

Driver

Le driver reprend le template du driver des tps suivis en cours.

```java
// Classe Driver (contient le main du programme Hadoop).
public class MapReduceCO2 {

// Le main du programme.
    public static void main(String[] args) throws Exception {
        // Créer un object de configuration Hadoop.
        Configuration conf = new Configuration();
        // Permet a Hadoop de lire ses arguments generiques, recupere les arguments restants dans ourArgs.
        String[] ourArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
        // Obtient un nouvel objet Job: une tache Hadoop. On fourni la configuration Hadoop ainsi qu'une description
        // textuelle de la tache.
        Job job = Job.getInstance(conf, "Graph Job v1");

        // Defini les classes driver, map et reduce.
        job.setJarByClass(MapReduceCO2.class);
        job.setMapperClass(MapCO2.class);
        job.setReducerClass(ReduceCO2.class);

        // Defini types cle/valeurs de notre programme Hadoop.
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        // Defini les fichiers d'entree du programme et le repertoire des resultats.
        // On se sert du premier et du deuxieme argument restants pour permettre a l'utilisateur de les specifier
        // lors de l'execution.
        FileInputFormat.addInputPath(job, new Path(ourArgs[0]));
        FileOutputFormat.setOutputPath(job, new Path(ourArgs[1]));

        // On lance la tache Hadoop. Si elle s'est effectuee correctement, on renvoie 0. Sinon, on renvoie -1.
        if(job.waitForCompletion(true))
            System.exit(0);
        System.exit(-1);
    }
```

Pour les commandes avec input/output, voir le fichier commande_mapreduce ou sinon ci-dessous :

#

#COMMANDE HADOOP MAPREDUCE

#

Compiler le code

cd /home/AUBERT/projetmbds/mapreduce

javac MapCO2.java ReduceCO2.java MapReduceCO2.java

Construire la hierarchie du .jar et y déplacer le code compilé

mkdir -p org/mapreduceco2

mv MapReduceCO2.class MapCO2.class ReduceCO2.class org/mapreduceco2/

Générer le jar

jar -cvf MapreduceCO2.jar -C . org

#OUTPUT :

#added manifest

#adding: org/(in = 0) (out= 0)(stored 0%)

#adding: org/mapreduceco2/(in = 0) (out= 0)(stored 0%)

#adding: org/mapreduceco2/MapCO2.class(in = 2507) (out= 1162)(deflated 53%)

#adding: org/mapreduceco2/MapReduceCO2.class(in = 1593) (out= 823)(deflated 48%)

#adding: org/mapreduceco2/ReduceCO2.class(in = 2263) (out= 1016)(deflated 55%)

Déplacer le fichier CO2.csv sur HDFS

hadoop fs -mkdir /CO2

supprimer le fichier si besoin

hadoop fs -rm /CO2/CO2.csv

hadoop fs -put /home/AUBERT/projetmbds/mapreduce/CO2.csv /CO2

Vérifiez hadoop fs -cat /CO2/results/part-r-00000a présence sur HDFS:

hadoop fs -ls /CO2

#OUTPUT:

Found 1 items

-rw-r--r--   1 AUBERT supergroup   38916 2021-03-30 21:53 /CO2/CO.csv

Exécutez le programme

hadoop fs -rm /CO2/results/*

hadoop fs -rmdir /CO2/results

hadoop jar MapreduceCO2.jar org.mapreduceco2.MapReduceCO2 /CO2/CO2.csv /CO2/results

#OUTPUT:

########

21/03/30 21:57:49 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032

21/03/30 21:57:50 INFO input.FileInputFormat: Total input paths to process : 1

21/03/30 21:57:50 INFO mapreduce.JobSubmitter: number of splits:1

21/03/30 21:57:50 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1612782394313_0

21/03/30 21:57:50 INFO impl.YarnClientImpl: Submitted application application_1612782394313_0

21/03/30 21:57:50 INFO mapreduce.Job: The url to track the job: http://bigdatalite.localdomai

21/03/30 21:57:50 INFO mapreduce.Job: Running job: job_1612782394313_0085

21/03/30 21:57:57 INFO mapreduce.Job: Job job_1612782394313_0085 running in uber mode : false

21/03/30 21:57:57 INFO mapreduce.Job: map 0% reduce 0%

21/03/30 21:58:01 INFO mapreduce.Job: map 100% reduce 0%

21/03/30 21:58:06 INFO mapreduce.Job: map 100% reduce 100%

21/03/30 21:58:06 INFO mapreduce.Job: Job job_1612782394313_0085 completed successfully

21/03/30 21:58:06 INFO mapreduce.Job: Counters: 49

File System Counters

FILE: Number of bytes read=7874

FILE: Number of bytes written=305777

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=39027

HDFS: Number of bytes written=299

HDFS: Number of read operations=6

HDFS: Number of large read operations=0

HDFS: Number of write operations=2

Job Counters

Launched map tasks=1

Launched reduce tasks=1

Data-local map tasks=1

Total time spent by all maps in occupied slots (ms)=2287

Total time spent by all reduces in occupied slots (ms)=2577

Total time spent by all map tasks (ms)=2287

Total time spent by all reduce tasks (ms)=2577

Total vcore-milliseconds taken by all map tasks=2287

Total vcore-milliseconds taken by all reduce tasks=2577

Total megabyte-milliseconds taken by all map tasks=2341888

Total megabyte-milliseconds taken by all reduce tasks=2638848

Map-Reduce Framework

Map input records=438

Map output records=335

Map output bytes=7198

Map output materialized bytes=7874

Input split bytes=111

Combine input records=0

Combine output records=0

Reduce input groups=16

Reduce shuffle bytes=7874

Reduce input records=335

Reduce output records=16

Spilled Records=670

Shuffled Maps =1

Failed Shuffles=0

Merged Map outputs=1

GC time elapsed (ms)=99

CPU time spent (ms)=1340

Physical memory (bytes) snapshot=454017024

Virtual memory (bytes) snapshot=3792904192

Total committed heap usage (bytes)=326631424

Shuffle Errors

BAD_ID=0

CONNECTION=0

IO_ERROR=0

WRONG_LENGTH=0

WRONG_MAP=0

WRONG_REDUCE=0

File Input Format Counters

Bytes Read=38916

File Output Format Counters

Bytes Written=299

######

Consulter les résultats

hadoop fs -ls /CO2/results

#OUTPUT:

#Found 2 items

#-rw-r--r-- 1 AUBERT supergroup 0 2021-03-30 21:58 /CO2/results/_SUCCESS

#-rw-r--r-- 1 AUBERT supergroup 299 2021-03-30 21:58 /CO2/results/part-r-00000

#Pour visualiser le résultat

hadoop fs -cat /CO2/results/part-r-00000

#OUTPUT:

#########

AUDI -2400 26 191

BENTLEY 0 84 102

BMW -631 39 80

CITROEN -6000 0 347

DS -3000 16 159

HYUNDAI -4000 8 151

JAGUAR -6000 0 271

KIA -3000 15 132

LAND 0 69 78

MERCEDES 7790 187 749

MINI -3000 21 126

MITSUBISHI 0 40 98

NISSAN 5802 160 681

PEUGEOT -3000 15 144

PORSCHE 0 69 89

RENAULT -6000 0 206

SKODA -666 27 98

SMART -6000 0 191

TESLA -6000 0 245

TOYOTA 0 32 43

VOLKSWAGEN -1714 23 96

VOLVO 0 42 72

############

Export du fichier de résultats depuis hdfs

hadoop fs -get /CO2/results/part-r-00000 /home/AUBERT/projetmbds/mapreduce/resultat.txt