Amina Dahmouni

Classe : II-BDCC2

[Compte Rendu]

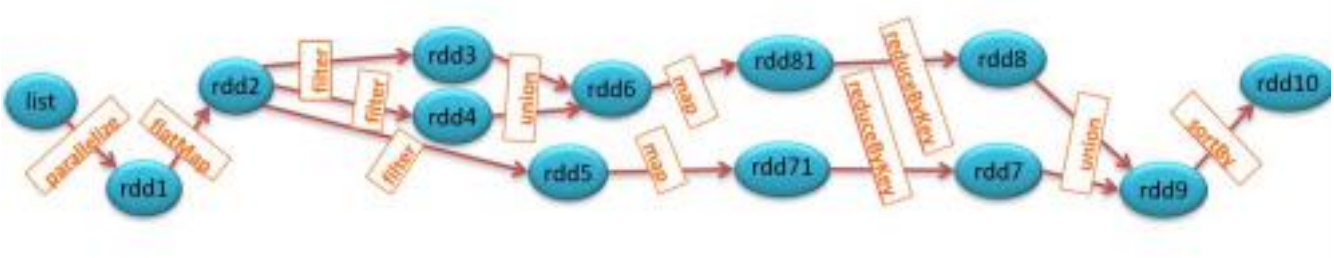
TP 7 : Calcul parallèle et distribué avec spark

Big Data: Fondements et Architectures de stockage

L'Ecole Normale Supérieure de l'Enseignement Technique de Mohammedia (ENSET)

Exercice 1 :

On considère le graphe suivant qui représente le lignée des RDDS :



Le premier RDD (RDD1) est créé en parallélisant une collection contenant des noms des étudiants. Ecrivez une application Spark qui permet de réaliser ce lignage.

Exercice1.java

```
import ...

public class Exercice1 {
    public static void main(String[] args) {
        SparkConf conf=new SparkConf().setAppName("EXercice 1").setMaster("local[*]");
        JavaSparkContext sc=new JavaSparkContext(conf);
        JavaRDD<String> rdd1=sc.parallelize(Arrays.asList("Amina","Safae","Fatima","Bouchra"));
        JavaRDD<String> rdd3=rdd1.filter(a -> a.contains("Amina"));
        rdd3.foreach(a-> System.out.printf( s: "rdd3: "+a));
```

```
rdd3: Amina
```

```
        JavaRDD<String> rdd4=rdd1.filter(a -> a.contains("Fatima"));
        rdd4.foreach(a-> System.out.printf( s: "rdd4: "+a));
```

```
rdd4: Fatima
```

```
        JavaRDD<String> rdd5=rdd1.filter(a -> a.contains("Safae"));
        rdd5.foreach(a-> System.out.printf( s: "rdd5: "+a));
```

```
rdd5: Safae
```

```
        JavaRDD<String> rdd6=rdd3.union(rdd4);
        rdd6.foreach(a-> System.out.printf( s: "rdd6: "+a));
```

```
rdd6: Amina
```

```
        JavaRDD<String> rdd71=rdd5.map( a-> a +" DH");
        rdd71.foreach(a-> System.out.printf( s: "rdd71: "+a));
```

```
rdd71: Safae DH
```

```
        JavaRDD<String> rdd81=rdd6.map( a-> a +" DH");
        rdd81.foreach(a-> System.out.printf( s: "rdd81: "+a));
```

```
rdd81: Amina DH
```

```
rdd81: Fatima DH
```

```
        JavaPairRDD<String,Integer> rdd72=rdd71.mapToPair(s->new Tuple2<>(s,1));
        JavaPairRDD<String,Integer> rdd7=rdd72.reduceByKey((v1, v2)->v1+v2);
        rdd7.foreach(nameTuple-> System.out.println("rdd7: "+"Name: "+nameTuple._1()+" key="+nameTuple._2()));
```

```
rdd7: Name: Safae DH key=1
```

```
        JavaPairRDD<String,Integer> rdd82=rdd81.mapToPair(s->new Tuple2<>(s,1));
        JavaPairRDD<String,Integer> rdd8=rdd82.reduceByKey((v1, v2)->v1+v2);
        rdd8.foreach(nameTuple-> System.out.println("rdd8: "+"Name: "+nameTuple._1()+" key="+nameTuple._2()));
```

```
rdd8: Name: Fatima DH key=1
rdd8: Name: Amina DH key=1
```

```
        JavaPairRDD<String,Integer> rdd9=rdd7.union(rdd8);
        rdd9.foreach(nameTuple-> System.out.println("rdd9: "+"Name: "+nameTuple._1()+" key="+nameTuple._2()));
```

```
rdd9: Name: Safae DH key=1
rdd9: Name: Fatima DH key=1
rdd9: Name: Amina DH key=1
```

```
    JavaPairRDD<String,Integer> rdd10=rdd9.sortByKey();
    rdd10.foreach(nameTuple-> System.out.println("rdd10: "+"Name: "+nameTuple._1()+" key="+nameTuple._2()));
}
```

```
rdd10: Name: Fatima DH key=1
rdd10: Name: Amina DH key=1
rdd10: Name: Safae DH key=1
```

Exercice 2 :

1. On souhaite développer une application Spark permettant, à partir d'un fichier texte (ventes.txt) en entré, contenant les ventes d'une entreprise dans les différentes villes, de déterminer le total des ventes par ville. La structure du fichier ventes.txt est de la forme suivante :

date ville produit prix .

Exercice2.java

```
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import scala.Tuple2;

public class Exercice2 {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName ("Exercice 2" ) .setMaster ("local[*]") ;
        JavaSparkContext sc = new JavaSparkContext(conf);
        JavaRDD<String> rdd1 = sc.textFile( path: "ventes.txt");
        /*
          Question 1
          */
        JavaPairRDD<String, Long> rdd2 = rdd1.mapToPair (s -> new Tuple2<>(s. split( s: " ") [1],
                Long.parseLong (s.split( s: " ")[3])));
        JavaPairRDD<String, Long> rdd3 = rdd2.reduceByKey ((v1, v2) -> v1+v2) ;
        rdd3.foreach(nameTuple->System.out.println("Ville: "+nameTuple._1()+" Total_vente="+nameTuple._2()));
```

```
Ville: Tanger Total_vente=11000
Ville: Rabat Total_vente=7200
Ville: Fes Total_vente=8000
```

2. Vous créez une deuxième application permettant de calculer le prix total des ventes des produits par ville pour une année donnée.

```
/*
  Question 2
  */
JavaPairRDD<String,Integer> rdd4 = rdd1.mapToPair(s-> new Tuple2<>("Années: "+s.split( s: " ")[0]+"  ville:  "+s.split( s: " ")[1],
        Integer.parseInt(s.split( s: " ")[3])));
JavaPairRDD<String,Integer> rdd5=rdd4.reduceByKey((v1, v2) -> v1+v2);
rdd5.foreach(nameTuple->System.out.println(nameTuple._1()+" Total_vente="+nameTuple._2()));
```

```
Années: 2022  ville:  Rabat Total_vente=6000
Années: 2020  ville:  Fes Total_vente=8000
Années: 2019  ville:  Rabat Total_vente=1200
Années: 2022  ville:  Tanger Total_vente=11000
```

Exercice 3 :

Nous souhaitons, dans cet exercice d'analyser les données météorologiques fournies par NCEI (National Centers for Environmental Information) à l'aide de Spark.

Le jeu de données est mis à la disposition du public par le NCEI. L'ensemble de données séparé par année peut être téléchargé à partir du lien suivant : Index of /pub/data/ghcn/daily/by_year (noaa.gov).
L'ensemble de données contient les attributs ID, date, element, value, m-flag, q flag, s-flag et OBS-TIME. Vous trouverez ci-dessous des informations sur ses attributs.

- ID = 11-character station identification code
- YEAR/MONTH/DAY = 8-character date in YYYYMMDD format (e.g. 19860529 = May 29, 1986)
- ELEMENT = 4-character indicator of element type
- DATA VALUE = 5-character data value for ELEMENT
- M-FLAG = 1-character Measurement Flag
- Q-FLAG = 1-character Quality Flag
- S-FLAG = 1-character Source Flag
- OBS-TIME = 4-character time of observation in hour-minute format (i.e. 0700 =7:00 am)

Afficher les statistiques suivantes pour 2020 :

- Température minimale moyenne.

Exercice3.java

```
import ...
public class Exercice3 {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName ("Exercice 3" ) .setMaster ("local[*]") ;
        JavaSparkContext sc = new JavaSparkContext(conf);
        JavaRDD<String> rdd1 = sc.textFile( path: "data.csv");
        JavaPairRDD<String, Long> rdd2 = rdd1.mapToPair (s -> new Tuple2<>(s. split( s: ",") [2],
                Long.parseLong (s.split( s: ",")[3])));
        //rdd2.foreach(nameTuple-> System.out.println(nameTuple._1()+" "+nameTuple._2()));
         /*
          Question 1
          */
        JavaPairRDD<String, Long> rdd3 = rdd2. filter (a -> a._1(). equals ("TMIN"));
        JavaPairRDD<String, Long> rdd4 = rdd3.reduceByKey ((v1, v2) -> v1+v2) ;
        Long nMIN = rdd3.count();
        System.out.printf( s: "Température minimale moyenne ===>");
        rdd4.foreach(nameTuple->System.out.println(nameTuple._1()+" "+nameTuple._2()/nMIN));
```

```
Température minimale moyenne ===>
 TMIN 106
```

• Température maximale moyenne.

```java
/*
  Question 2
  */
JavaPairRDD<String, Long> rdd5 = rdd2. filter (a -> a._1(). equals ("TMAX"));
// rdd5.foreach(nameTuple-> System.out.println(nameTuple._1()+" "+nameTuple._2()));
JavaPairRDD<String, Long> rdd6 = rdd5.reduceByKey ((v1, v2) -> v1+v2) ;
Long nMax = rdd5.count();
System.out.printf( s: "Température maximale moyenne ===>");
rdd6.foreach(nameTuple->System.out.println(nameTuple._1()+" "+nameTuple._2()/nMax));
```

```
Température maximale moyenne ===>
TMAX 153
```

• Température maximale la plus élevée.

```java
/*
  Question 3
  */
JavaRDD<Long> rdd7= rdd1.map ((s -> Long.parseLong(s.split( s: ",")[3])));
List<Long> list = rdd7.collect();
System.out.println("Température maximale la plus élevée "+list.stream().max(Long::compare).get());
```

```
Température maximale la plus élevée 301
```

• Température minimale la plus basse.

```java
/*
  Question 4
  */
System.out.println("Température minimale la plus basse "+list.stream().min(Long::compare).get());
```

```
Température minimale la plus basse -19
```

• Top 5 des stations météo les plus chaudes.

```
/*
  Question 5
  */
JavaRDD<String> rddTMax = rdd1.filter(s -> s.contains("TMAX"));
JavaPairRDD<Integer, String> station_chaudes = rddTMax.mapToPair(s -> new Tuple2<>(Integer.parseInt(s.split( s: ",")[3]),
        s.split( s: ",")[0]));
System.out.println("Les 5  stations meteo les plus chaudes : " + station_chaudes.sortByKey( ascending: false).take( num: 5));
```

```
Les 5  stations meteo les plus chaudes : [(301,ITE00100554), (297,ITE00100554), (296,ITE00100554), (296,ITE00100554), (296,ITE00100554)]
22/04/19 21:13:13 INFO SparkContext: Starting job: sortByKey at Exercice3.java:55
22/04/19 21:13:13 INFO DAGScheduler: Got job 7 (sortByKey at Exercice3.java:55) with 2 output partitions
22/04/19 21:13:13 INFO DAGScheduler: Final stage: ResultStage 10 (sortByKey at Exercice3.java:55)
```

• Top 5 des stations météo les plus froides.

```
/*
  Question 6
  */
JavaRDD<String> rddTMin = rdd1.filter(s -> s.contains("TMIN"));
JavaPairRDD<Integer, String> station_froids = rddTMin.mapToPair(s -> new Tuple2<>(Integer.parseInt(s.split( s: ",")[3]),
        s.split( s: ",")[0]));
System.out.println("Les 5  stations meteo les plus froides. : " + station_froids.sortByKey( ascending: true).take( num: 5));
```

```
Les 5  stations meteo les plus froides. : [(-19,ITE00100554), (-12,ITE00100554), (-11,ITE00100554), (-10,ITE00100554), (-8,ITE00100554)]
22/04/19 21:13:14 INFO SparkContext: Invoking stop() from shutdown hook
22/04/19 21:13:14 INFO SparkUI: Stopped Spark web UI at http://192.168.49.128:4040
```