TP 7: clacul parallèle et distribué avec spark

Réaliser par :

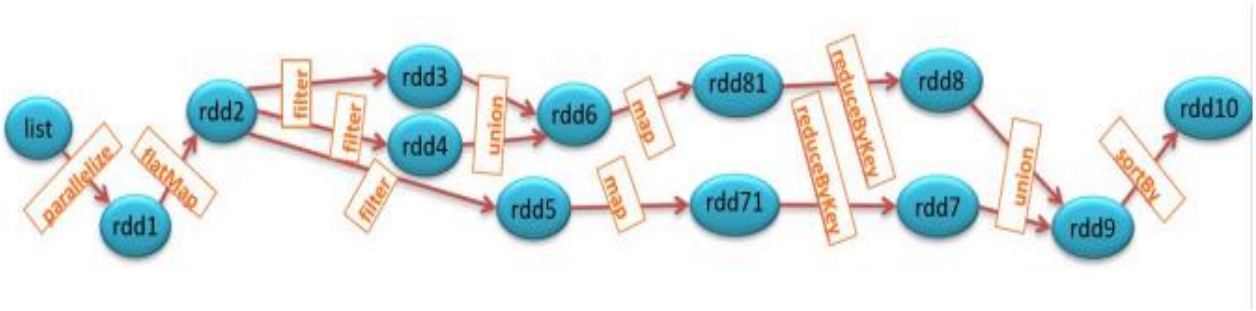
Mohamed AIT LAHCEN

Encadrant :

Mr. Abdelmajid

Exercice1 :

Ecrivez une application Spark qui permet de réaliser ce lignage.




```
public static void main(String[] args) {
    SparkConf conf= new SparkConf().setAppName("Word count").setMaster("local[*]");
    JavaSparkContext sc = new JavaSparkContext(conf);
    JavaRDD<String> rdd1=sc.textFile( path: "/home/med/eclipse-workspace/rdd-tp1/etudiants");
    rdd1.foreach(r->System.out.println(r));
    JavaRDD<String> rdd2=rdd1.flatMap(f->Arrays.asList(f.split( s: " ")).iterator());
    JavaRDD<String> rdd3=rdd2.filter(e->e.length()>5);
    JavaRDD<String> rdd4=rdd2.filter(e->e.length()<=5);
    JavaRDD<String> rdd5=rdd2.filter(e->e.length()>7);
    JavaRDD<String> rdd6=rdd3.union(rdd4);
    JavaRDD<String> rdd71=rdd5.map(f->f+"_rdd71");
    JavaRDD<String> rdd81=rdd6.map(f->f+"_rdd81");
    JavaPairRDD<String, Integer> rdd82=rdd81.mapToPair(f->new Tuple2(f, 8));
    JavaPairRDD<String, Integer> rdd8=rdd82.reduceByKey((v1,v2)->v1+v2);
    JavaPairRDD<String, Integer> rdd72=rdd71.mapToPair(f->new Tuple2(f, 7));
    JavaPairRDD<String, Integer> rdd7=rdd72.reduceByKey((v1,v2)->v1+v2);
    JavaPairRDD<String, Integer> rdd9= rdd8.union(rdd7);
    JavaPairRDD<String, Integer> rdd10 = rdd9.sortByKey();
    rdd10.foreach(el->System.out.println(el));
```

Le fichier etudiants.txt qui contient les noms des étudiants qui représentent les données d'entrée :

```
mohamed khalid rachid sofiane
adil zoubir sara ilyas
abdlhakim rachid khalid safa
mohamed rachid adil alaa
```

Voici la sortie de l'application :

```
(abdlhakim_rdd71,7)
(abdlhakim_rdd81,8)
```

```
(adil_rdd81,16)
(alaa_rdd81,8)
(ilyas_rdd81,8)
(khalid_rdd81,16)
```

```
22/04/21 02:51:43 INF
(safa_rdd81,8)
(sara_rdd81,8)
```

```
(mohamed_rdd81,16)
(rachid_rdd81,24)
```

```
22/04/21 02:51:44 INFO E
(sofiane_rdd81,8)
(zoubir_rdd81,8)
```

Exercice2 :

1- Déterminer le total des ventes par ville.

```
mohamed-ait
public static void main(String[] args) {
    SparkConf conf = new SparkConf().setAppName("total des ventes par ville").setMaster("local[*]");
    JavaSparkContext cxt=new JavaSparkContext(conf);
    JavaRDD<String> rdd1=cxt.textFile( path: "ventes.txt");
    JavaRDD<String> rdd2=rdd1.filter(f->f.isEmpty()==false);
    JavaPairRDD<String, Double> rddVente=rdd2.mapToPair(r->{
        String els[]=r.split( s: " ");
        return new Tuple2<String, Double>(els[1],Double.valueOf(els[3]));
    });
    rdd1.foreach(f->System.out.println(f));
    JavaPairRDD<String, Double> rddTotalVille=rddVente.reduceByKey((v1,v2)->v1+v2);
    rddTotalVille.foreach(nameTuple-> System.out.println(nameTuple._1()+" "+nameTuple._2()));
}
```

```
22/04/21 02:14:37 INFO ShuffleBlockFetcherIterator:
ouarzazate 290000.0
marakech 410000.0
22/04/21 02:14:37 INFO Executor: Finished task 1.0
```

2-calculer le prix total des ventes des produits par ville pour une année donnée.

```
mohamed-ait
public static void main(String[] args) {
    SparkConf conf = new SparkConf().setAppName("total des ventes par ville pour une année donnée").setMaster("local[*]");
    JavaSparkContext cxt=new JavaSparkContext(conf);
    JavaRDD<String> rdd1= cxt.textFile( path: "ventes.txt");
    JavaRDD<String> rdd2=rdd1.filter(f->f.isEmpty()==false);
    JavaPairRDD<String, Double> rddVentes=rdd2.mapToPair(r->{
        String els[]=r.split( s: " ");
        return new Tuple2<String,Double>(els[0]+els[1],Double.valueOf(els[3]));
    });
    JavaPairRDD<String, Double> rddTotalVilleAnne=rddVentes.reduceByKey((v1,v2)->v1+v2);
    rddTotalVilleAnne.foreach(f->System.out.println(f._1()+"----->"+f._2()));
}
```

```
2018ouarzazate----->90000.0
2018marakech----->140000.0
2019marakech----->110000.0
2020marakech----->160000.0
2019ouarzazate----->70000.0
2020ouarzazate----->130000.0
```

Exercice3 :

Calculer :

1- La température minimale moyenne :

```
public static void main(String[] args) {
    SparkConf conf = new SparkConf().setAppName("la temperature maximale moyenne").setMaster("local[*]");
    JavaSparkContext cxt=new JavaSparkContext(conf);
    JavaRDD<String> rdd1=cxt.textFile( path: "temperatures.csv");
    JavaRDD<String> rdd2=rdd1.filter(f->{
        String[] els=f.split( s: ",");
        return els[2].equals("TMIN");
    });
    //construire un rdd des temperatures minimaux
    JavaRDD<Double> rddTmps=rdd1.map(s->{
        String[] els=s.split( s: ",");
        return Double.valueOf(els[3]);
    });
    //calculer la somme des temperatures minimaux:
  Double tmpSum=rddTmps.reduce((v1, v2)->v1+v2);
  //claculer la moyenne des temperatures minimiaux :
    double moyTmp=tmpSum/rddTmps.count();
    System.out.println("la temperature minimale moyenne est \t"+moyTmp);
}
```

```
22/04/21 02:26:01 INFO DAGScheduler: Job 1 finished: count at Applicatio
la temperature minimale moyenne est     59.49122687239708
22/04/21 02:26:01 INFO SparkContext: Invoking stop() from shutdown hook
```

2- La température maximale moyenne :

```
mohamed-ait
public static void main(String[] args) {
    SparkConf conf = new SparkConf().setAppName("la temperature minimale moyenne").setMaster("local[*]");
    JavaSparkContext cxt=new JavaSparkContext(conf);
    JavaRDD<String> rdd1=cxt.textFile( path: "temperatures.csv");
    JavaRDD<String> rdd2=rdd1.filter(f->{
        String[] els=f.split( s: ",");
        return els[2].equals("TMAX");
    });
    //construire un rdd des temperatures maximaux
    JavaRDD<Double> rddTmps=rdd1.map(s->{
        String[] els=s.split( s: ",");
        return Double.valueOf(els[3]);
    });
    //calculer la somme des temperatures maximaux:
    Double tmpSum=rddTmps.reduce((v1, v2)->v1+v2);
    //claculer la moyenne des temperatures minimiaux :
    double moyTmp=tmpSum/rddTmps.count();
    System.out.println("la temperature maximale moyenne est \t"+moyTmp);
```

```
22/04/21 02:36:44 INFO DAGScheduler: Job 1 finished: count at Application2.
la temperature maximale moyenne est     59.49122687239708
22/04/21 02:36:44 INFO BlockManagerInfo: Removed broadcast_1_piece0 on med
```

3-La température maximale moyenne :

```
public static void main(String[] args) {
    SparkConf conf = new SparkConf().setAppName("la temperature maximale la plus elevée").setMaster("local[*]");
    JavaSparkContext cxt=new JavaSparkContext(conf);
    JavaRDD<String> rdd1=cxt.textFile( path: "temperatures.csv");
    JavaRDD<String> rdd2=rdd1.filter(f->{
        String[] els=f.split( s: ",");
        return els[2].equals("TMAX");
    });
    //construir un rdd qui regroupe les temperatures minimaux
    JavaRDD<Double> rddTmps=rdd1.map(s->{
        String[] els=s.split( s: ",");
        return Double.valueOf(els[3]);
    });
    //déterminer la temperature la plus elevée des temperatures maximales :
    Double tmpMax=rddTmps.reduce((v1,v2)->Math.max(v1,v2));
    System.out.println("la temperature la plus elevée des temperatures maximaux est \t"+tmpMax);
}
}
```

```
22/04/21 02:39:32 INFO DAGScheduler: Job 0 finished: reduce at Application3.java:23, t

la temperature la plus elevée des temperatures maximaux est     1676.0

22/04/21 02:39:32 INFO SparkContext: Invoking stop() from shutdown hook
```

4-La température maximale moyenne :

```java
public static void main(String[] args) {
    SparkConf conf = new SparkConf().setAppName("Température minimale la plus basse.").setMaster("local[*]");
    JavaSparkContext cxt=new JavaSparkContext(conf);
    JavaRDD<String> rdd1=cxt.textFile( path: "temperatures.csv");
    JavaRDD<String> rdd2=rdd1.filter(f->{
        String[] els=f.split( s: ",");
        return els[2].equals("TMIN");
    });
    //construir un rdd qui regroupe les temperatures minimaux
    JavaRDD<Double> rddTmps=rdd1.map(s->{
        String[] els=s.split( s: ",");
        return Double.valueOf(els[3]);
    });
    //déterminer la temperature la plus elevée des temperatures maximales :
    Double tmpMin=rddTmps.reduce((v1,v2)->Math.min(v1,v2));
    System.out.println("Température minimale la plus basse: \t"+tmpMin);
}
```

```
22/04/21 02:42:19 INFO DAGScheduler: Job 0 finished: reduce at Application4.java:22, t
Température minimale la plus basse:     -328.0
22/04/21 02:42:19 INFO SparkContext: Invoking stop() from shutdown hook
```

5-Les top 5 des stations météo les plus chaudes :

```
JavaRDD<String> rdd2=rdd1.filter(f->{
    return f.split( s: ",")[2].equals("TMAX");
});
JavaPairRDD<String,Integer> rdd3=rdd2.mapToPair(f->new Tuple2<>(f.split( s: ",")[0],Integer.parseInt(f.split( s: ",")[3])));
JavaPairRDD<String, Iterable<Integer>> rdd4=rdd3.groupByKey();
JavaPairRDD<Integer,String> rdd5=rdd4.mapToPair(f->{
    Iterator<Integer> it=f._2.iterator();
    Integer max=Integer.MIN_VALUE;
    Integer i=0;
    while(it.hasNext()){
        if(max<=(i=it.next())){
            max=i;
        }
    }
    return new Tuple2<Integer,String>(max,f._1());
});
JavaPairRDD<Integer,String> sortedRdd=rdd5.sortByKey( ascending: true);
List<Tuple2<Integer,String>> els=sortedRdd.take( num: 5);
els.forEach(f-> System.out.println(f._2()+"-->"+f._1()));
```

```
GME00127462-->256
USC00413270-->272
UK000047811-->291
UK000056225-->300
SWE00139148-->302
```

5-Les top 5 des stations météo les plus froides :

```
JavaRDD<String> rdd2=rdd1.filter(f->{
    return f.split( s: ",")[2].equals("TMIN");
});
JavaPairRDD<String,Integer> rdd3=rdd2.mapToPair(f->new Tuple2<>(f.split( s: ",")[0],Integer.parseInt(f.split( s: ",")[3])));
JavaPairRDD<String, Iterable<Integer>> rdd4=rdd3.groupByKey();
JavaPairRDD<Integer,String> rdd5=rdd4.mapToPair(f->{
    Iterator<Integer> it=f._2.iterator();
    Integer min=Integer.MAX_VALUE;
    Integer i=0;
    while(it.hasNext()){
        if(min>=(i=it.next())){
            min=i;
        }
    }
    return new Tuple2<Integer,String>(min,f._1());
});
JavaPairRDD<Integer,String> sortedRdd=rdd5.sortByKey( ascending: true);
List<Tuple2<Integer,String>> els=sortedRdd.take( num: 5);
els.forEach(f-> System.out.println(f._2()+"-->"+f._1()));
```

```
CA006158350-->-328
SWE00139148-->-280
GME00127462-->-175
GM000004204-->-171
AU000005901-->-141
```