Amina Dahmouni

Classe : II-BDCC2

[Compte Rendu]

TP 4 : MapReduce et YARN

Big Data: Fondements et Architectures de stockage

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

Exercice 1 :

1. Étant donné une liste d'employés avec leur département et leur salaire, trouvez le salaire maximum et minimum dans chaque département.

Map :

Q1_Map.java | Employee.csv

```java
package enset;
import ...
public class Q1_Map extends MapReduceBase implements Mapper<LongWritable, Text,Text, DoubleWritable> {
    @Override
    public void map(LongWritable key, Text value, OutputCollector<Text, DoubleWritable> output,
                    Reporter reporter) throws IOException {
                String employees[]=value.toString().split( s: ",");
                System.out.println(employees.length);
                output.collect(new Text(employees[2]),new DoubleWritable(Double.parseDouble(employees[4])));
    }
}
```

Reduce :

Employee.csv | Q1_Reduce.java

```java
package enset;
import ...
public class Q1_Reduce extends MapReduceBase implements Reducer<Text, DoubleWritable, Text,Text> {
    @Override
    public void reduce(Text text, Iterator<DoubleWritable> values, OutputCollector<Text,Text> outputCollector,
                       Reporter reporter) throws IOException {
        Double maxSalary = 0.00, minSalary = 0.00, var;
        maxSalary = minSalary = values.next().get();
        while (values.hasNext()) {
            var = values.next().get();
            if (var > maxSalary) {
                maxSalary = var;
            }
            else minSalary = var;
        }
        String max_min="Le salaire  Min="+String.valueOf(minSalary)+" le salaire Max="+String.valueOf(maxSalary);
                System.out.println(max_min);
                outputCollector.collect(text,new Text(max_min));
    }
```

Application.java :

Employee.csv | Application.java

```java
import ...
public class Application {
    public static void main(String[] args) throws IOException {
        JobConf conf=new JobConf();
        conf.setJobName("Min Max des salaires des employées par département");
        conf.setJarByClass(Application.class);

        conf.setMapperClass(Q1_Map.class);
        conf.setReducerClass(Q1_Reduce.class);

        conf.setMapOutputKeyClass(Text.class);
        conf.setMapOutputValueClass(DoubleWritable.class);

        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(Text.class);

        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

/*      FileInputFormat.addInputPath(conf,new Path("Employee.csv"));
        FileOutputFormat.setOutputPath(conf,new Path("./output1"));*/

        FileInputFormat.addInputPath(conf,new Path(args[0]));
        FileOutputFormat.setOutputPath(conf,new Path(args[1]));

        JobClient.runJob(conf);
    }
}
```

Résultat :

Open | part-00000 (5) ~/Downloads | Save

```
1 finance        Le salaire  Min=9000.0  le salaire Max=17000.0
2 informatique   Le salaire  Min=20000.0 le salaire Max=24000.0
```

2. Étant donné une liste d'employés avec leur département, trouvez le nombre d'employés dans chaque département.

Map :

Q2_Map.java

```java
package enset.Exercice1.Q2;
import ...
public class Q2_Map extends MapReduceBase implements Mapper<LongWritable, Text,Text, IntWritable> {
    @Override
    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output,
                    Reporter reporter) throws IOException {
                String employees[]=value.toString().split( s: ",");
                System.out.println(employees.length);
                int valeur=1;
                output.collect(new Text(employees[2]),new IntWritable(valeur));
    }
}
```

Reduce :

Q2_Reduce.java

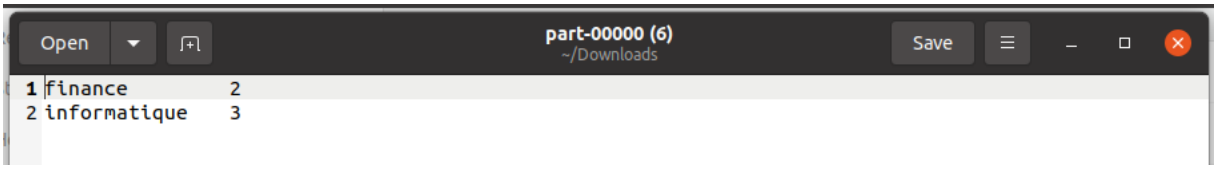
```java
package enset.Exercice1.Q2;
import ...
public class Q2_Reduce extends MapReduceBase implements Reducer<Text, IntWritable,Text,IntWritable> {
    @Override
    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output,
                       Reporter reporter) throws IOException {
        int  nbrEmployee=0;
        while (values.hasNext()) {
            nbrEmployee += values.next().get();
        }
        output.collect(new Text(key), new IntWritable(nbrEmployee));
    }
}
```

Application.java :

Application.java

```java
package enset.Exercice1.Q2;
import ...
public class Application {
    public static void main(String[] args) throws IOException {
        JobConf conf=new JobConf();
        conf.setJobName("Nombre des employées par département");
        conf.setJarByClass(Application.class);

        conf.setMapperClass(Q1_Map.class);
        conf.setReducerClass(Q1_Reduce.class);



        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);

        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

/*      FileInputFormat.addInputPath(conf,new Path("Employee.csv"));
        FileOutputFormat.setOutputPath(conf,new Path("./output2"));*/

        FileInputFormat.addInputPath(conf,new Path(args[0]));
        FileOutputFormat.setOutputPath(conf,new Path(args[1]));

        JobClient.runJob(conf);
    }
}
```

Résultat :



Exercice 2 :

Map :

MapTemp.java

```
package enset.Exercice2;
import ...
public class MapTemp extends MapReduceBase implements Mapper<LongWritable, Text, Text, DoubleWritable> {
    @Override
    public void map(LongWritable longWritable, Text value, OutputCollector<Text, DoubleWritable> outputCollector,
                    Reporter reporter) throws IOException {
        String date= value.toString().split( s: "\",\"")[1];//extraire la date
        String month=date.toString().split( s: "-")[0];//extraire le mois
        String Temp=value.toString().split( s: "\",\"")[13];//extraire la température
        Temp.replace( charSequence: ",", charSequence1: ".");
        Double Tempdouble=Double.parseDouble(Temp);
        System.out.println(Tempdouble);
        outputCollector.collect(new Text(month),new DoubleWritable(Tempdouble));
```

Reduce :

ReduceTemp.java

```
package enset.Exercice2;
import ...
public class ReduceTemp extends MapReduceBase
        implements Reducer<Text, DoubleWritable,Text,DoubleWritable> {
    @Override
    public void reduce(Text key, Iterator<DoubleWritable> values, OutputCollector<Text, DoubleWritable> output,
                       Reporter reporter) throws IOException {
        Double max = Double.MIN_VALUE, min = Double.MAX_VALUE, var;
        while (values.hasNext()) {
            var = values.next().get();
            max=Math.max(var,max);
            min=Math.min(var,min);
        }
        output.collect(new Text( string: "La température maximale  du mois: "+key.toString()), new DoubleWritable(max));
        output.collect(new Text( string: "La température minimale  du mois: "+key.toString()), new DoubleWritable(min));
    }
```

Application.java :

Application.java

```java
package enset.Exercice2;
import ...
public class Application {
    public static void main(String[] args) throws IOException {
        JobConf conf=new JobConf();
        conf.setJobName("La température minimale et maximale pour chaque mois");
        conf.setJarByClass(enset.Exercice1.Q1.Application.class);

        conf.setMapperClass(MapTemp.class);
        conf.setReducerClass(ReduceTemp.class);

        conf.setMapOutputKeyClass(Text.class);
        conf.setMapOutputValueClass(DoubleWritable.class);

        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(Text.class);

        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

/*      FileInputFormat.addInputPath(conf,new Path("Employee.csv"));
        FileOutputFormat.setOutputPath(conf,new Path("./output1"));*/

        FileInputFormat.addInputPath(conf,new Path(args[0]));
        FileOutputFormat.setOutputPath(conf,new Path(args[1]));

        JobClient.runJob(conf);
    }
}
```

Résultat :

part-00000 (7) | part-00000 (8)

```
La température maximale  du mois: 01   45.1
La température minimale  du mois: 01   -127.1
La température maximale  du mois: 02   20.1
La température minimale  du mois: 02   -105.1
La température maximale  du mois: 03   30.1
La température minimale  du mois: 03   -62.1
```