Université Constantine 2
جامعة قسنطينة 2

TDSAI

Chapitre 3 : Hadoop
-partie 2-

Dr. S.ZERABI
Faculté des NTIC
Soumeya.zerabi@univ-constantine2.dz

Etudiants concernés

Faculté/Institut	Département	Niveau	Spécialité
Nouvelles technologies	IFA	M1	SDIA

MapReduce

- MapReduce est un framework **Java** permettant d'écrire des programmes afin d'assurer le traitement **parallèle et distribué** des données **massives** sur **plusieurs nœuds**.
- Chaque nœud traite les données qui sont stockées.
- Comporte principalement deux tâches:

1. **map**
2. **reduce.**

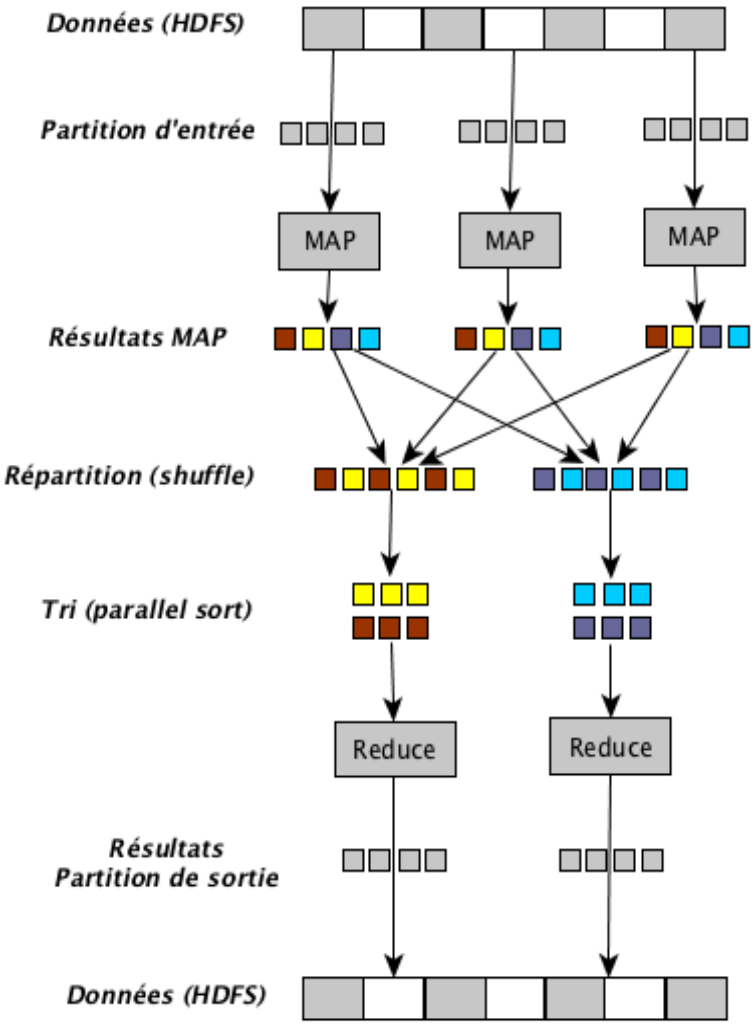
- Entre ces deux tâches:

1. **Shuffle** (mélanger)
2. **Sort** (trier).

© Dr Zerabi S.

Etapes d'exécution d'un job MapReduce

Un job MapReduce comprend plusieurs phases :



Split:
Division des données en blocs traitables séparément,

Map:
Application de la fonction map sur toutes les paires (clé, valeur) formées à partir des données d'entrée, cela produit d'autres paires (clé, valeur) en sortie,

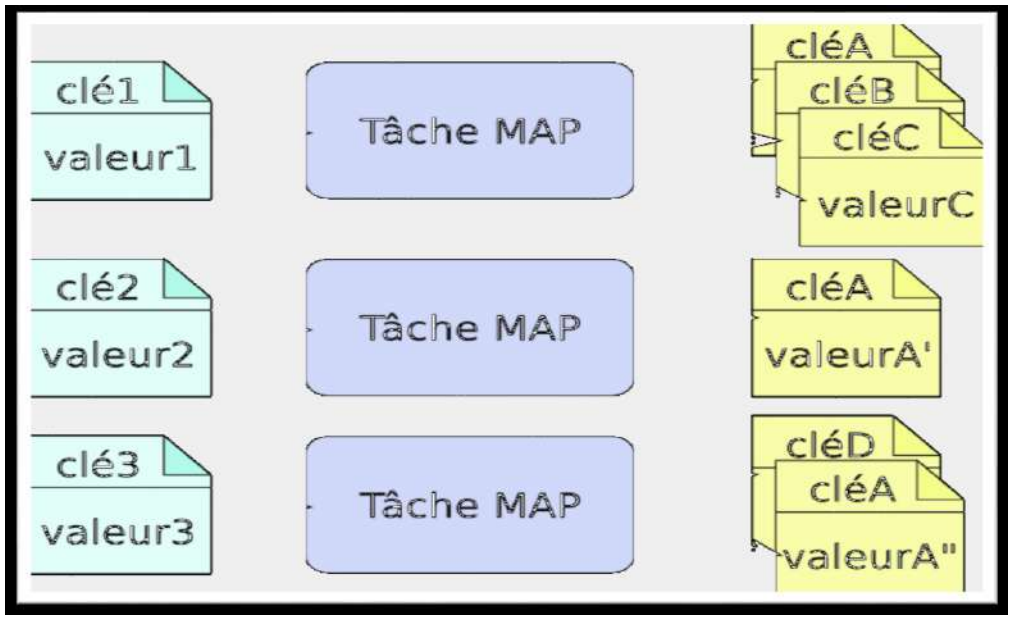
Shuffle&Sort:
Redistribution des données afin que les paires produites par Map ayants les mêmes clés soient sur les mêmes machines,

Reduce:
Agrégation des valeurs des paires ayant la même clé pour obtenir le résultat final.

© Dr Zerabi S.

La fonction Map

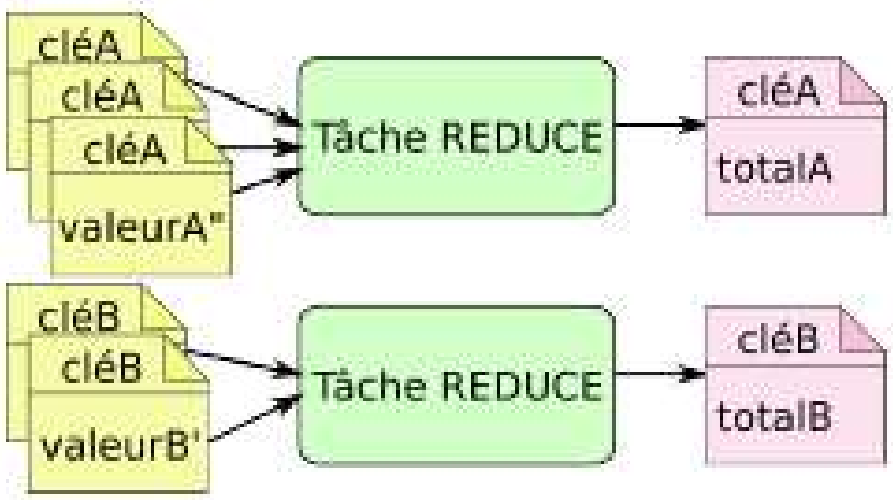
- Applique une transformation sur les données en entrée.
- Emettre en sorite **1..n** paires (clé, valeur) comme résultat intermédiaire.
- Il se peut que les mêmes clés **et/ou** valeurs soient produites.
- La fonction Map est **impérative.**



© Dr Zerabi S.

La fonction Reduce

- La fonction Reduce reçoit **une liste de paires** en entrée.
- Ce sont les valeurs intermédiaires produites par les fonctions de Map (mappers).
- Reduce combine ces valeurs intermédiaires pour chaque clef puis produit un ensemble de paires en sortie.
- Les paires *produites* par la fonction Reduce peuvent avoir la **même clé** que celle de l'entrée.
- La fonction Reduce est **optionnelle.**



© Dr Zerabi S.

Signatures

Les fonctions map et reduce ont les signatures suivantes:

- map : (in_key,in_value) ↦ [(intermediate_key,intermediate_value)]
- reduce : (intermediate_key,[intermediate_value]) ↦ [(out_key, out_value)]

© Dr Zerabi S.

MapReduce- Exemple

Word Count !

Compter le nombre d'occurrences de chaque mot dans un ensemble de textes.

Données : un ensemble de textes

Le jour se lève sur notre grisaille, sur les trottoirs de nos ruelles et sur nos tours
[...]

Le jour se lève sur notre envie de vous faire comprendre à tous que c'est à notre tour
[...]

(**Grand Corps Malade,** Le Jour se lève. *Extrait*)

Si le texte est grand (la collection *Wikipedia* qui contient environ 27 milliards de mots), la solution séquentielle ne suffira pas et il est nécessaire de réaliser ce comptage de **manière distribuée**.

Intervention de MapReduce!!!!

MapReduce-Exemple

Problème: *parmi un ensemble de textes, compter le nombre d'occurrences de chaque mot.*

- **Données input:** un ensemble de fichiers textes
- **Map:** sur chaque texte, décomposer en mots, et à chaque mot *m*, ajouter la valeur 1.
- **Shuffle & Sort:** le système regroupe/trie les paires selon la clé *m, dans une liste [(m,[1,1, ...]),...]*
- **Reduce:** les valeurs 1 sont additionnées pour chaque mot : *[(m,Σ1),...]*

La fonction map()

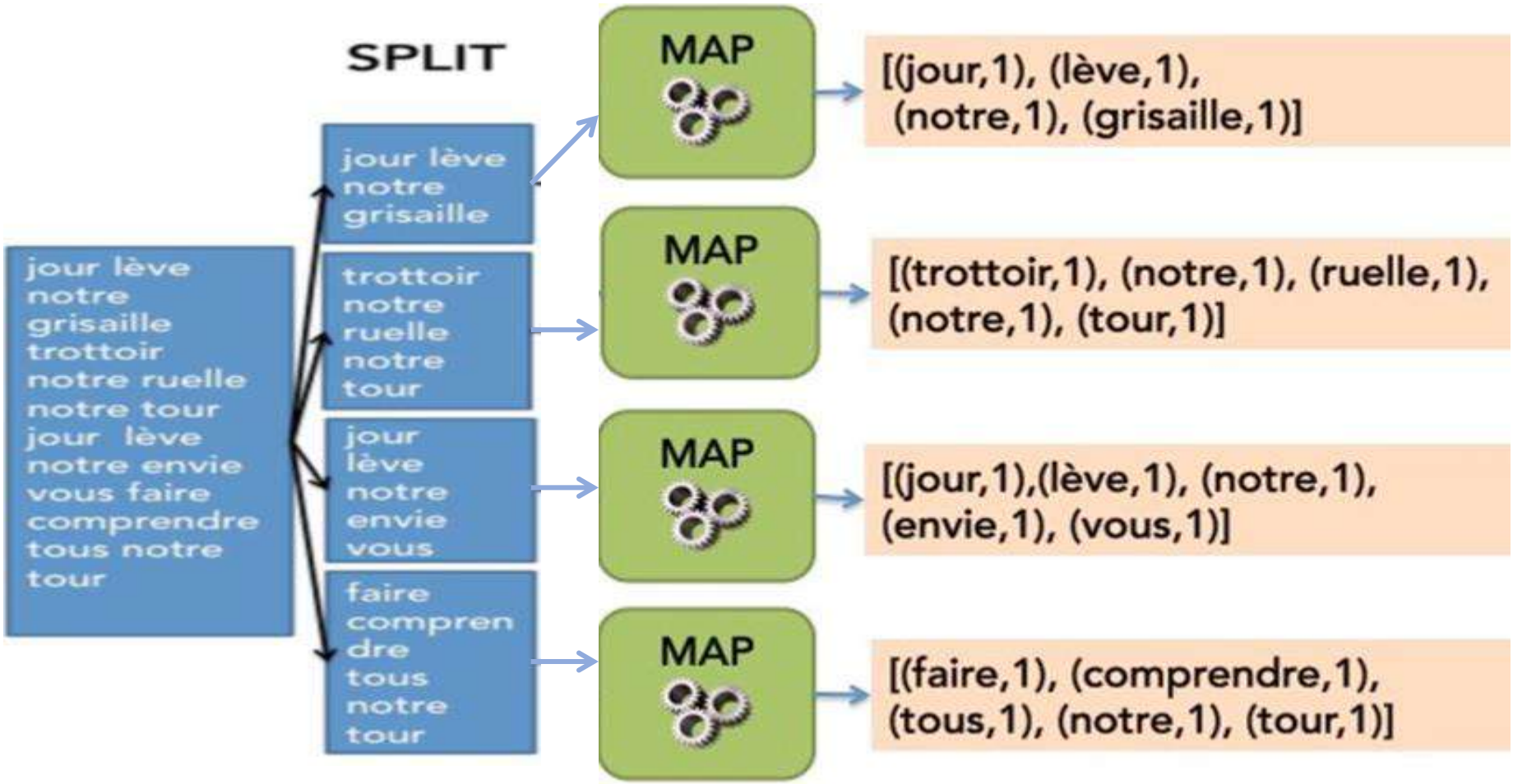
- La fonction map () va décomposer le texte du fragment fourni en entrée.
- elle va générer pour chaque mot une paire (mot, 1).

```
def map(key,value):
    list=[]
    for word in value.split():
        list.append((word, 1))
    return list
```

- La fin de map () est un ensemble de «clé, valeur» équivalent à <mot,1>
- Toutes les fonctions map () s'exécutent en parallèle.

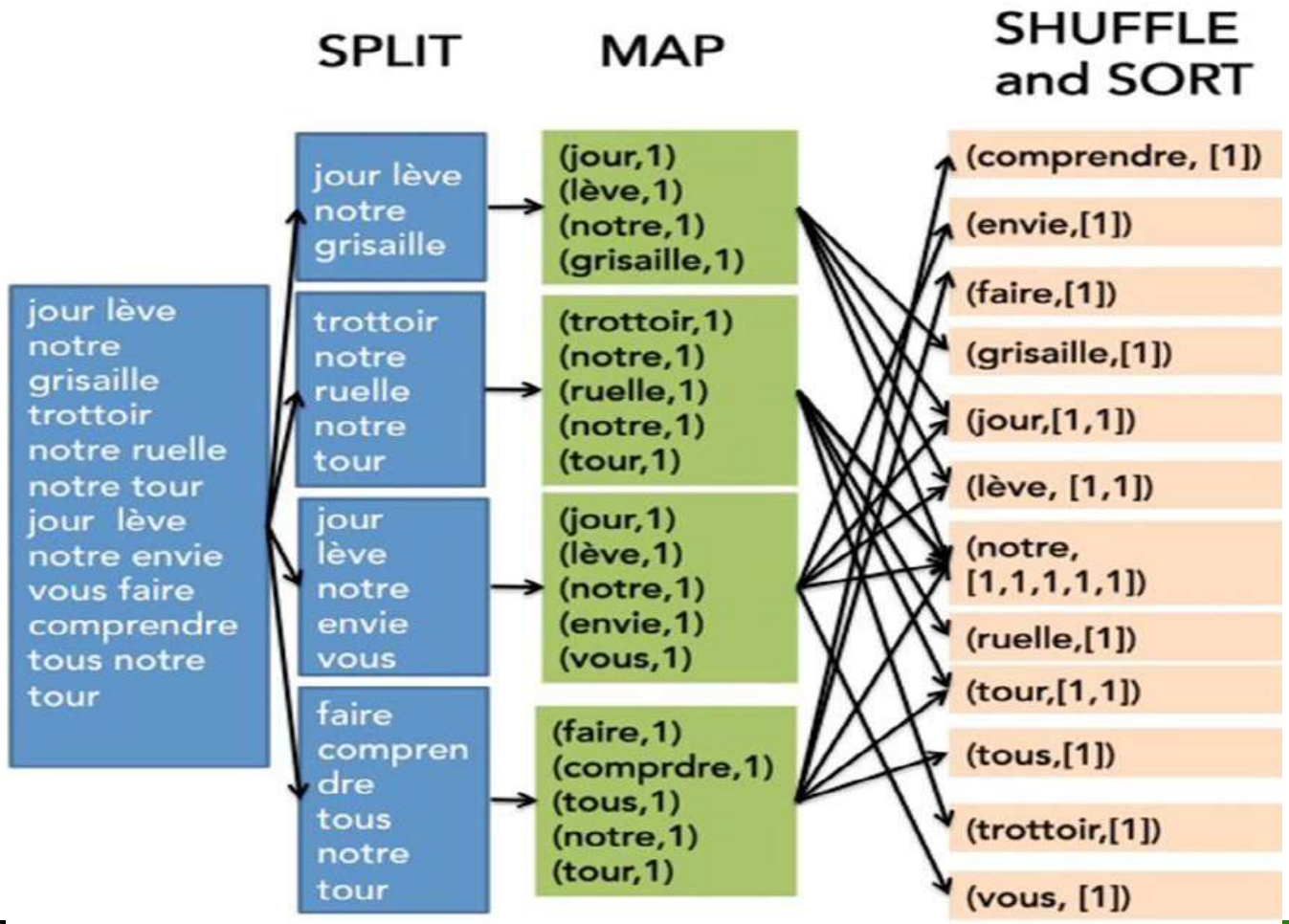
MapReduce -Exemple

La fonction map()



L'étape Shuffle @ sort

- Elle consiste à regrouper puis trier, par clé commune, les résultats intermédiaires fournis par la fonction map().



© Dr Zerabi S.

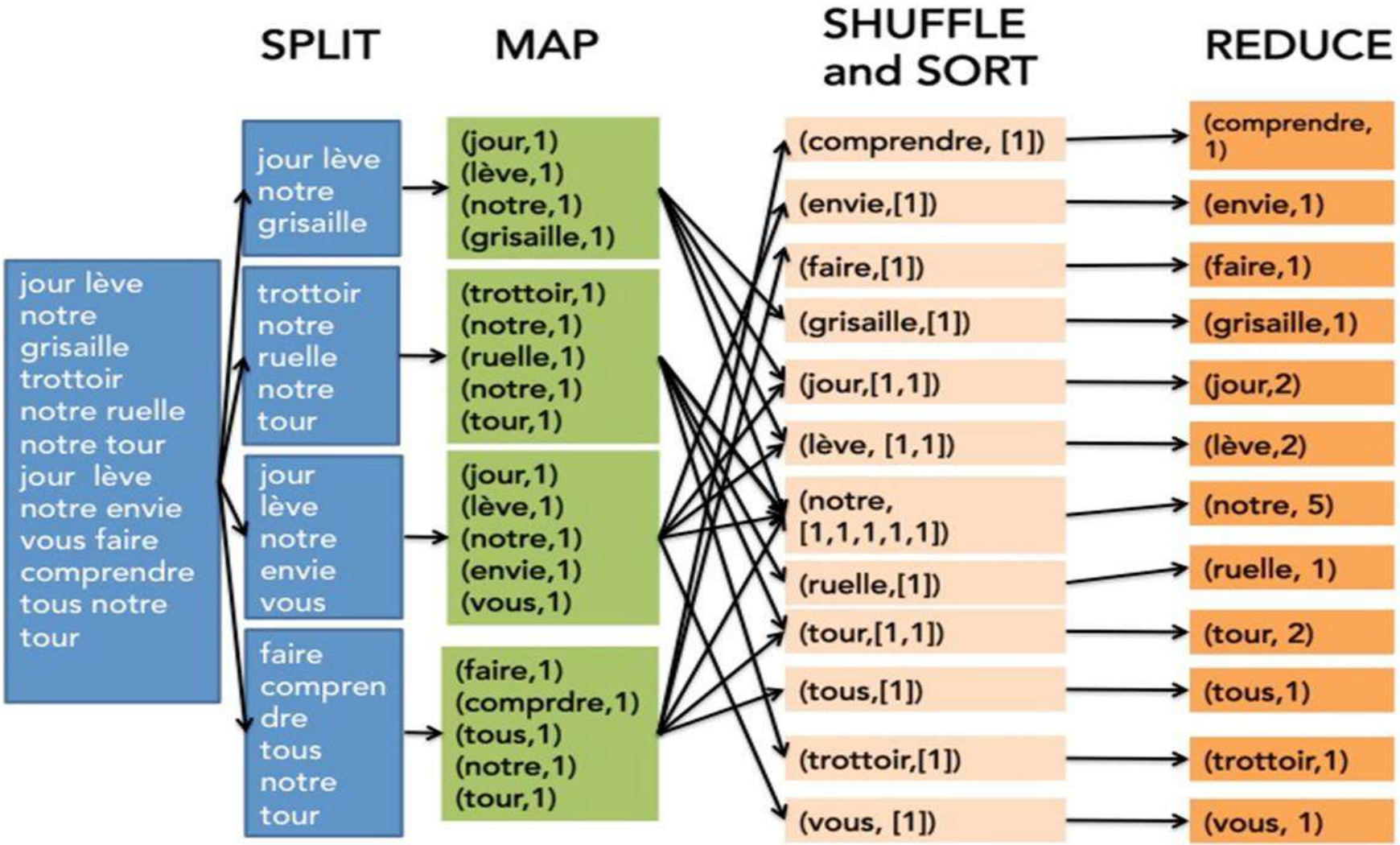
La fonction reduce()

- La fonction « reduce » va sommer toutes les valeurs de la liste associée à une clé unique.
- Cette fonction est appliquée à chaque paire (clé, liste_de_valeurs) en parallèle.

```
def reduce(key, values):
    result = 0
    for c in values:
        result = result +c
    return (key, result)
```

MapReduce -Exemple

La fonction reduce()



Implémentation d'un job MapReduce

Le programmeur d'un job MapReduce doit définir trois classes :

- Une sous classe de **Mapper**: chargé de lire les données stockées sur le disque puis les traiter **(la fonction map()).**
- Une sous classe de **Reducer**: chargé de consolider les résultats issus du mapper puis de les traiter et écrire le résultat sur disque **(la fonction reduce()).**
- Une classe **Driver**: chargé de configurer le job puis le soumettre pour exécution **(*main()*).**

© Dr Zerabi S.

Types de données en MapReduce

MapReduce utilise les types de données suivants:

1. Byte->ByteWritable
2. Short->ShortWritable
3. Integer->IntWritable
4. Long->LongWritable
5. Float->FloatWritable
6. Double->DoubleWritable
7. String->Text
8. Null->NullWritable

Exemples de manipulation de données MapReduce

```
IntWritable val = new IntWritable(5);

int v = val.get();

val.set(3);

Text word = new Text(''bonjour'');
```


© Dr Zerabi S.

Importation des packages

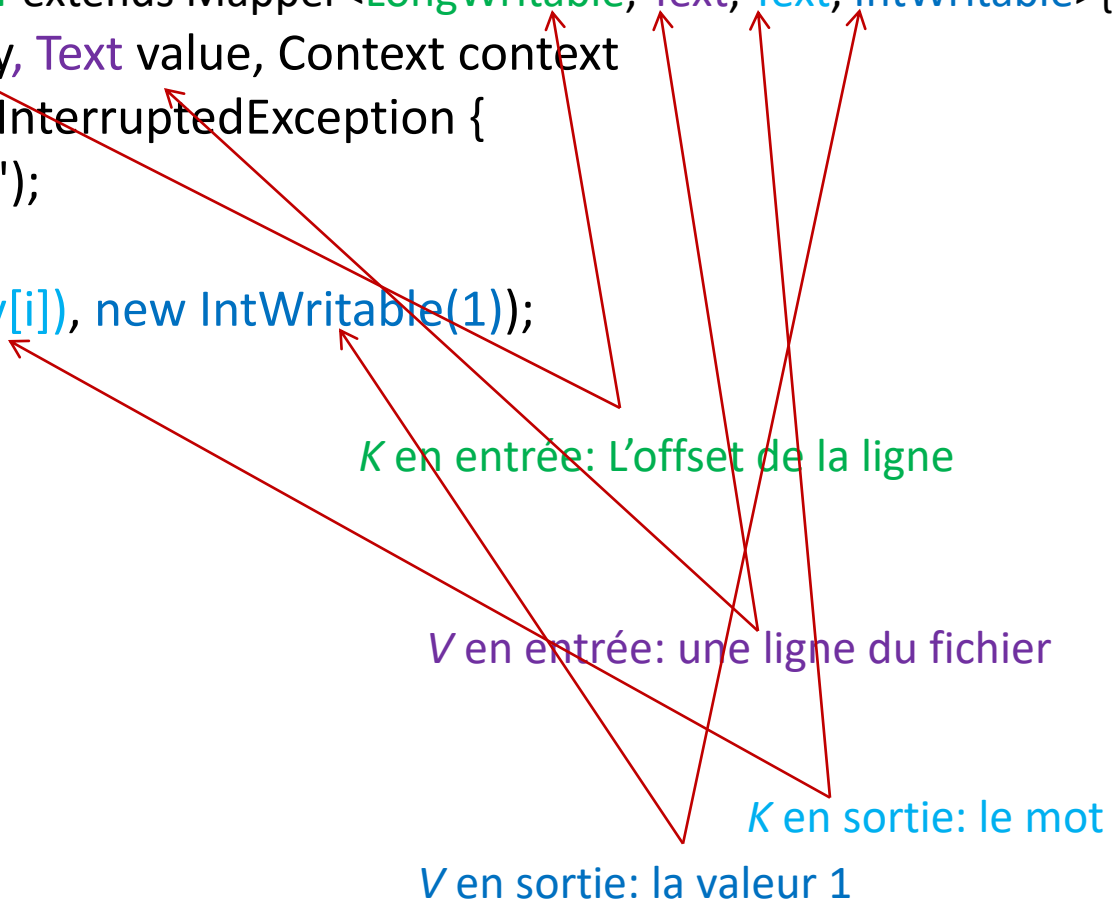
```
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```


© Dr Zerabi S.

Exemple: Word Count-Mapper.java

Créer une instance de la classe abstraite Mapper

```
public static class TokenizerMapper extends Mapper<LongWritable, Text, Text, IntWritable>{
public void map(LongWritable key, Text value, Context context
                ) throws IOException, InterruptedException {
String v[]=value.toString().split(" ");
For (i:0;i<=v.lenght();i++)
            context.write(new Text(v[i]), new IntWritable(1));
    }
   }
```



Exemple: Word Count-Reducer.java

```
public static class IntSumReducer
extends Reducer < Text, IntWritable, Text, IntWritable > {
 private IntWritable result = new IntWritable();
 public void reduce(Text key, Iterable < IntWritable > values,
Context context)
 throws IOException,
 InterruptedException {
  int sum = 0;
  for (IntWritable val: values) {
   sum += val.get();
  }
  result.set(sum);
  context.write(key, result);
 }

}
```


© Dr Zerabi S.

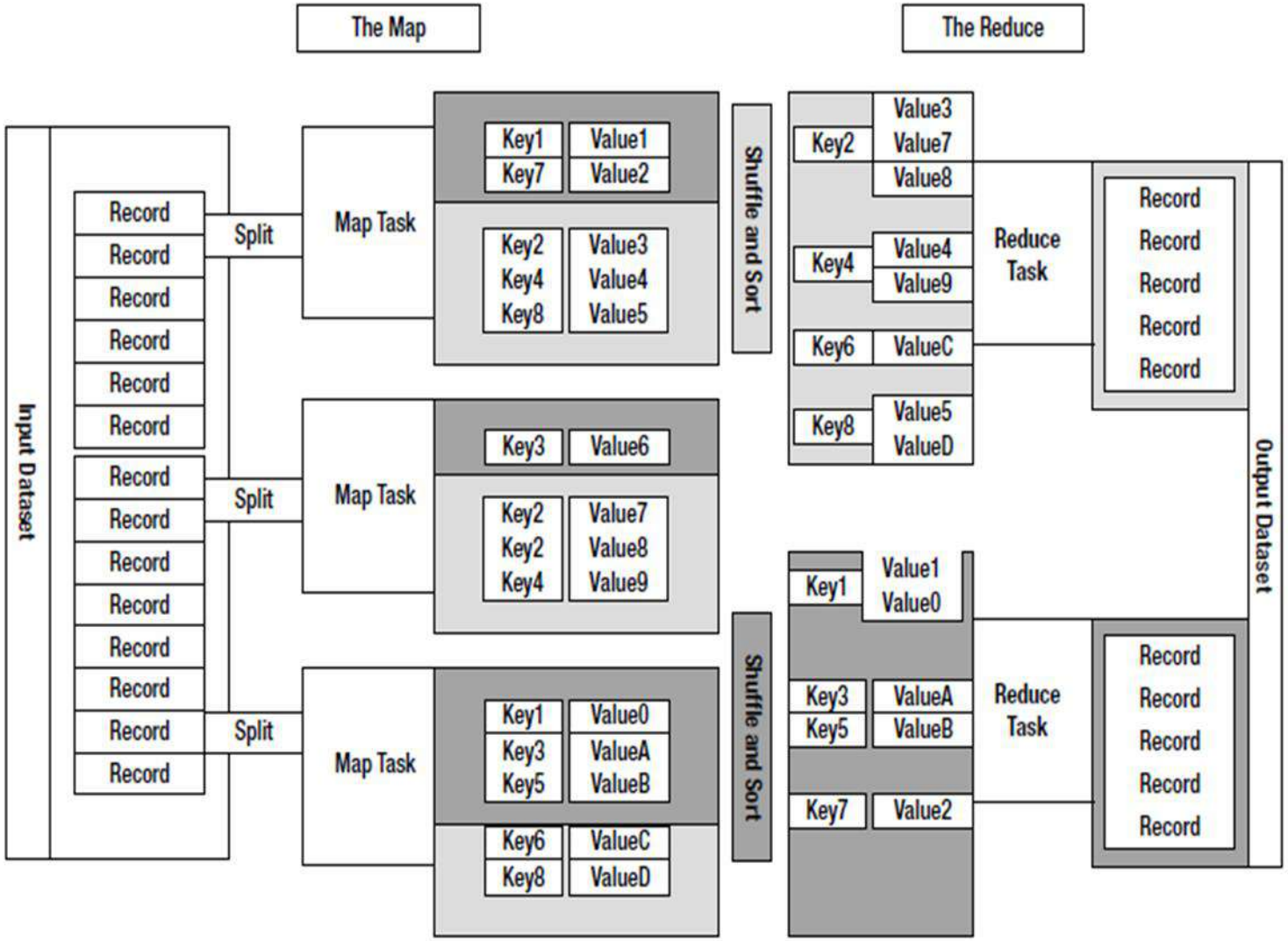
Exemple: Word Count-Driver

La classe Driver crée un Job faisant référence aux deux classes précédentes.

```
public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
  Job job = Job.getInstance(conf, "word count");
  job.setJarByClass(WordCount.class);
  job.setMapperClass(TokenizerMapper.class);

  job.setReducerClass(IntSumReducer.class);
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(IntWritable.class);
  FileInputFormat.addInputPath(job, new Path(args[0]));
  FileOutputFormat.setOutputPath(job, new Path(args[1]));
  System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```


© Dr Zerabi S.

Autre exemple

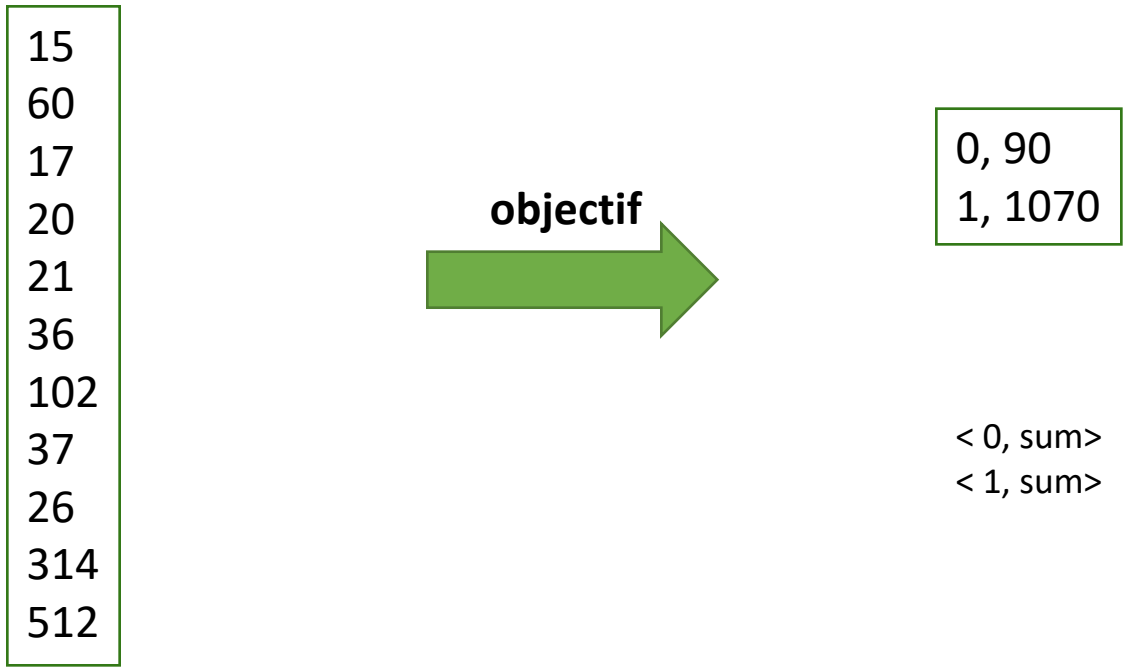


© Dr Zerabi S.

Exemple 1: pairs et impairs

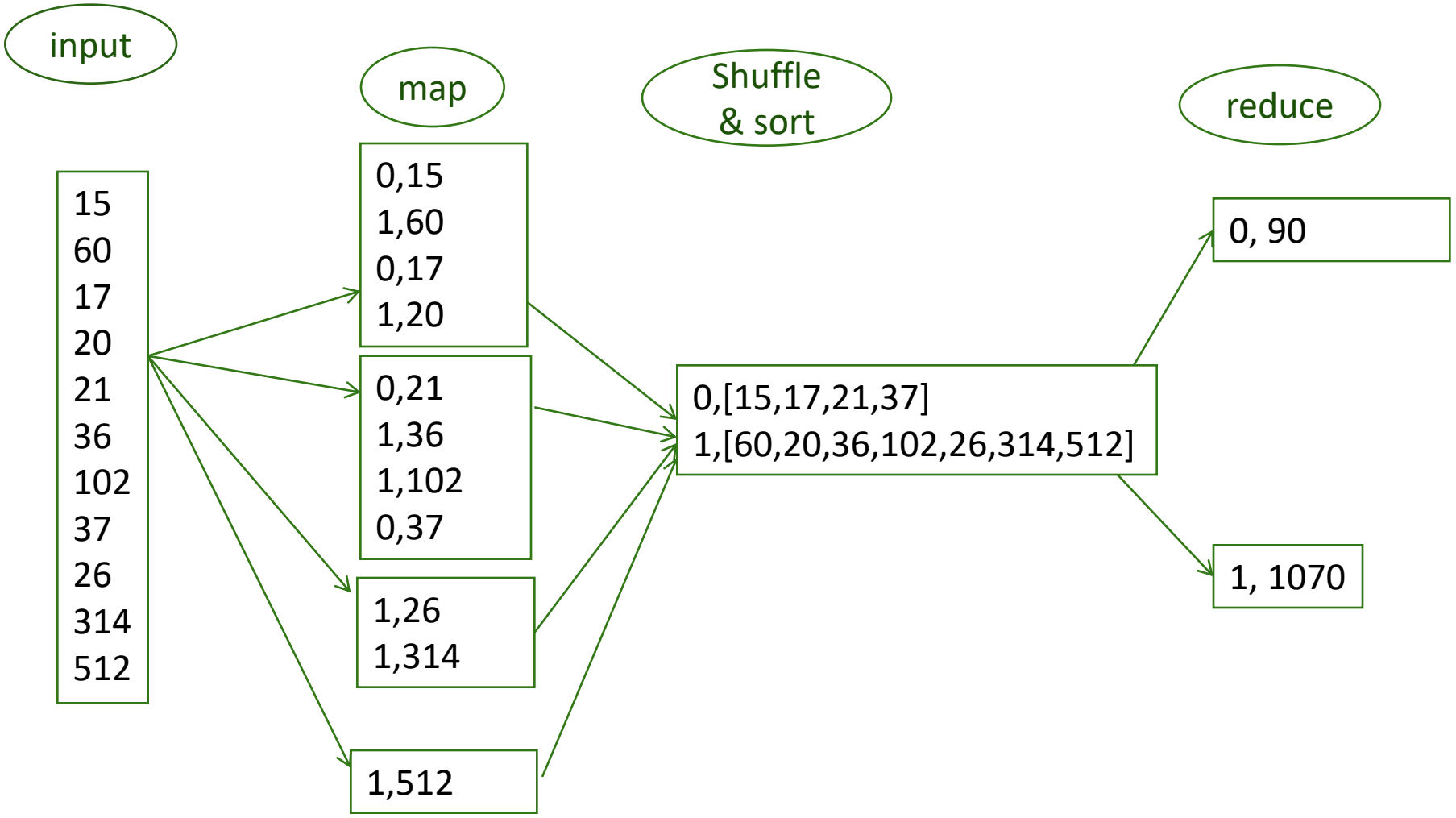
Problème:

Calculer la somme des nombres pairs et impairs dans un fichier .txt



© Dr Zerabi S.

Exemple 1: pairs et impairs



© Dr Zerabi S.

Mapper

CheckMapper.java | SumReducer.java | pairImpair.java

```
1  package pairImpair;
2  import java.io.IOException;
3  import org.apache.hadoop.io.IntWritable;
4  import org.apache.hadoop.io.LongWritable;
5  import org.apache.hadoop.io.Text;
6  import org.apache.hadoop.mapreduce.Mapper;
7
8  public class CheckMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
9
0      public void map(LongWritable key, Text value, Context context) throws
1      IOException, InterruptedException {
2          String v[]=value.toString().split(" ");
3          for (int i=0;i<v.length; i++){
4           int n = Integer.parseInt(v[i]);
5              if (n%2 == 0)
6                context.write(new Text("pair"), new IntWritable(n));
7               else
8                context.write(new Text("impair"), new IntWritable(n));
9
0
1      }
2
3  }}
```

Reducer

CheckMapper.java | SumReducer.java | pairImpair.java

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class SumReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {

        int sum = 0; int v=0;
    if (key.toString().equals("pair")){
        for (IntWritable val : values) {
                v= val.get();
                sum += v; }
     context.write(key, new IntWritable(sum));
    }
    else

      if (key.toString().equals("impair"))
          for (IntWritable val : values) {
          v= val.get();
          sum+= v;}
          context.write(key, new IntWritable(sum));
    }}
```

Driver

CheckMapper.java | SumReducer.java | pairImpair.java

```
public class pairImpair {

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "JobName");
        job.setJarByClass(pairImpair.class);
        // TODO: specify a mapper
        job.setMapperClass(CheckMapper.class);
        // TODO: specify a reducer
        job.setReducerClass(SumReducer.class);

        // TODO: specify output types
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        // TODO: specify input and output DIRECTORIES (not files)
        FileInputFormat.setInputPaths(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        if (!job.waitForCompletion(true))
            return;
    }
}
```

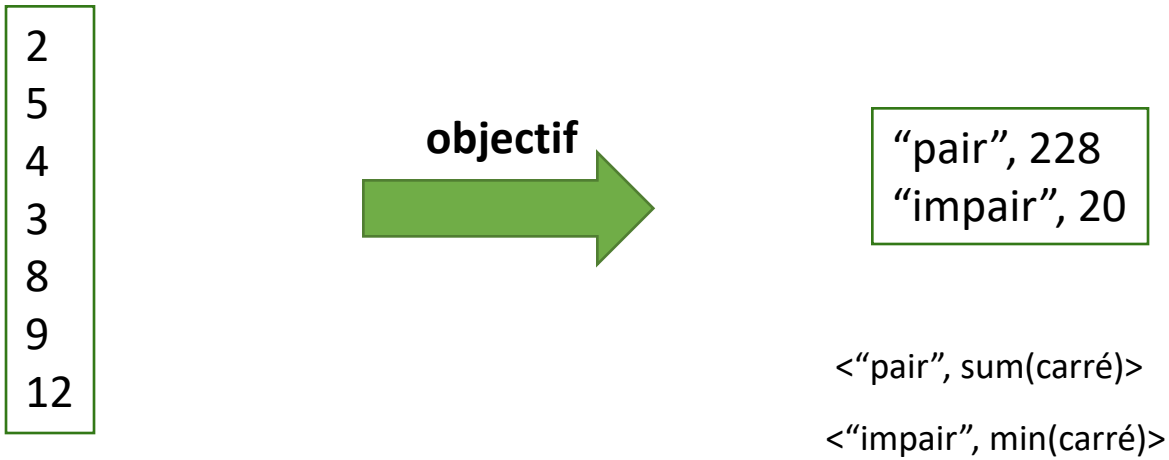
Solution Spark

```
rdd1=sc.sparkContext.parallelize([15,60,17,20,21,36,102,37,26,314,512])
rdd2=rdd.groupBy(lambda x: 'pair' if x%2==0  else 'impair').mapValues(sum).collect()
print(rdd2)
```

Exemple 2: carré

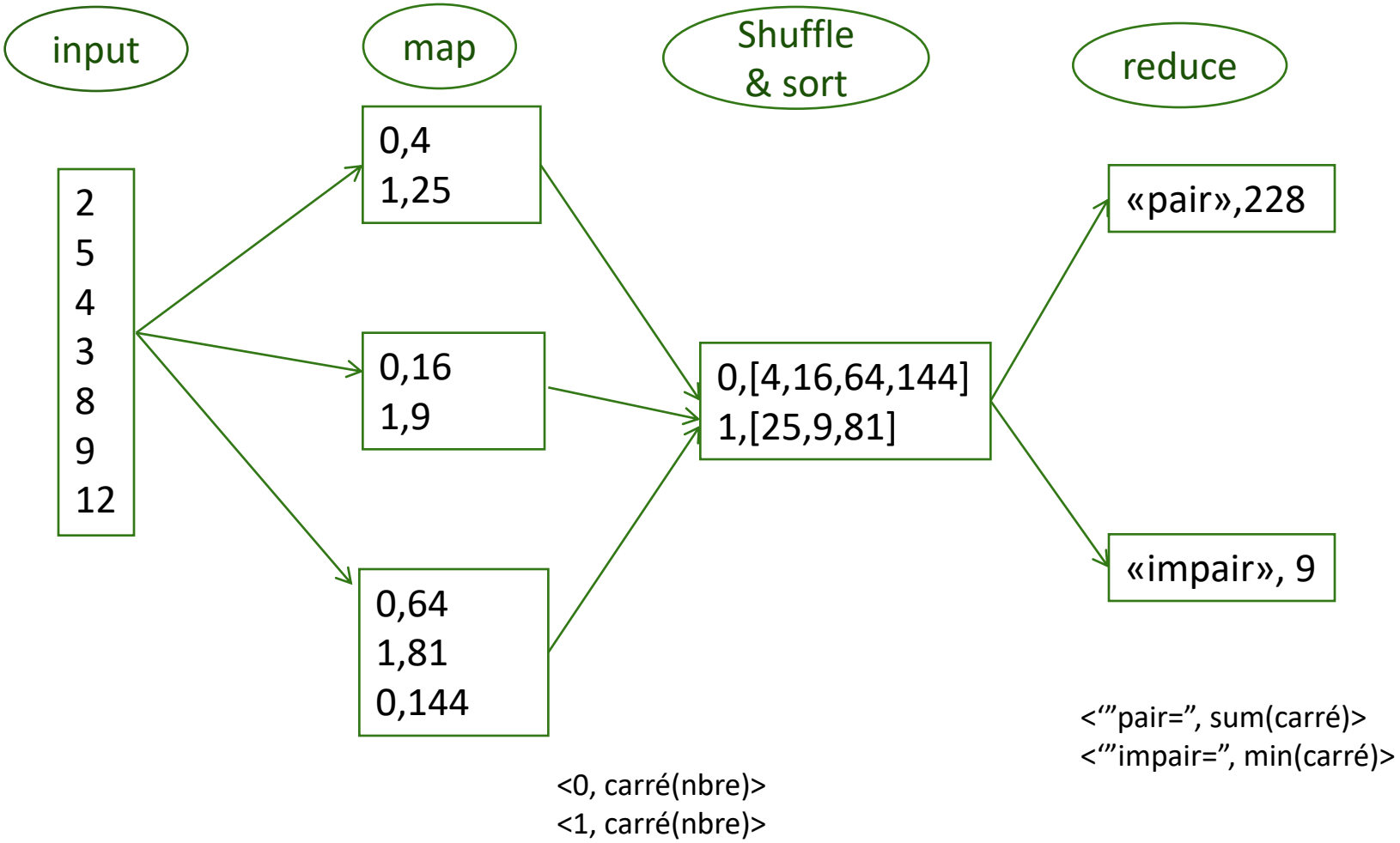
Problème:

Calculer la somme des carrés des nombres pairs et le min des carrés des nombres impairs du fichier suivant.



© Dr Zerabi S.

Exemple 2: carré



© Dr Zerabi S.

Exemple 3: total des ventes

Problème:

Calculer le total des ventes par magasin.

Date	Magasin	Produit	Prix
20/02/2019	Cne	Jouet	2000
31/03/2019	Alger	Parfum	5000
01/05/2020	Sétif	Robe	4500
04/07/2020	Cne	Pantalon	3500

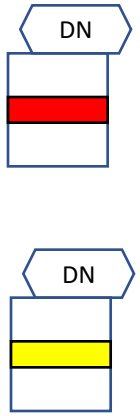
Clé	Valeur
?	?

<magasin, sum(prix)>

© Dr Zerabi S.

Exemple 3: total des ventes

Date	Magasin	Produit	Prix
20/02/2019	Cne	Jouet	2000
31/03/2019	Alger	Parfum	5000
01/05/2020	Sétif	Robe	4500
04/07/2020	Cne	pantalon	3500



Step 1: map

Clé	Valeur
Cne	2000
Alger	5000

Clé	Valeur
Sétif	4500
Cne	3500

Step 2: shuffle

Clé	Valeur
Cne	2000
Alger	5000
Sétif	4500
Cne	3500

<magasin, prix>

Step 3: sort

Clé	Valeur
Alger	5000
Cne	[2000, 3500]
Sétif	4500

Step 4: reduce

Clé	Valeur
Alger	5000
Cne	5500
Sétif	4500

<magasin, sum(prix)>

Exemple 4: durée des appels

Problème: Une administration souhaite connaitre quel est le service le plus demandé par sa clientèle en fonction de la **durée totale** des appels téléphoniques. Le fichier contenant tous les appels reçus, il est sous la forme (service, date, durée d'appel).

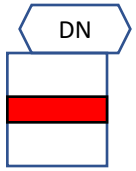
Il faut deux jobs comme suit:

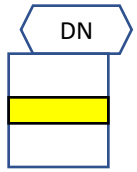
- **Données:** le fichier des appels **(1 appel par ligne),**
- **Map1:** reçoit une paire **(offset, ligne**) et extrait la paire **(service, durée),**
- **Reduce1:** additionne toutes les valeurs des paires qu'elle reçoit et produit une seule paire en sortie **(service, durée totale).**
- **Map2:** reçoit la paire **(service, durée totale)** et la retourne telle qu'elle est.
- **Reduce2:** calcule le max des valeurs des paires qu'il reçoit et produit une seule paire en sortie **(service, max(durée)).**

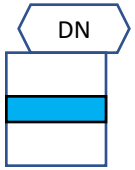
© Dr Zerabi S.

Exemple 4: durée des appels

Service	Date	Durée
Réception	20/03/2022	10
Bureau 2	31/05/2022	20
Réception	01/06/2022	5
Bureau 2	05/07/2022	30
Bureau 1	26/09/2022	4
Réception	30/09/2022	15
Bureau 1	01/10/2022	11
Réception	15/10/2022	5

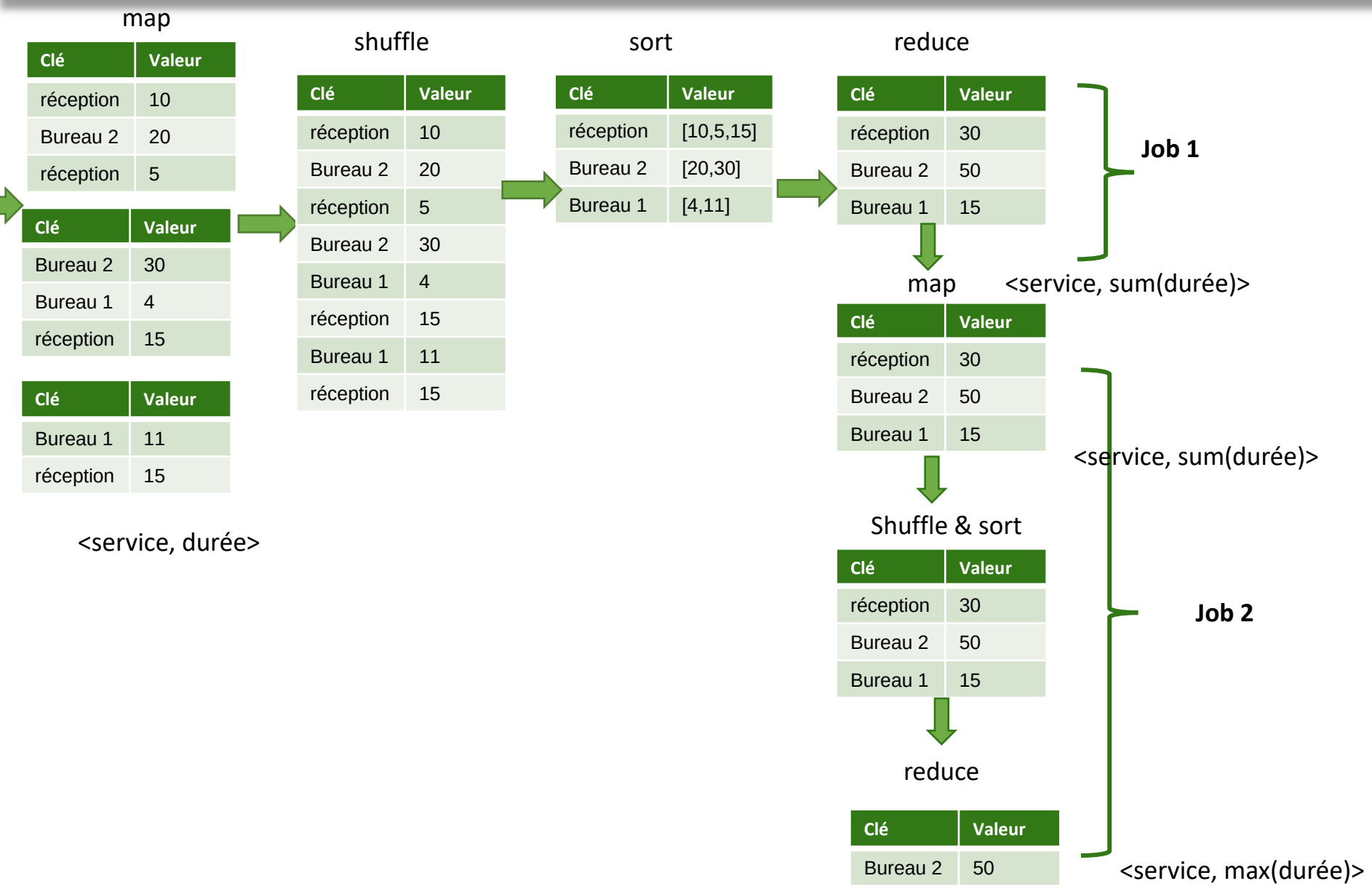






© Dr Zerabi S.

Exemple 4: durée des appels



© Dr Zerabi S.