Cours de « Apache Spark »

Create a file named « input.text » containing the folowing :

his is a test
This is a simple test

Introduction :

```
val f = sc.textFile("text.txt")
val w = f.flatMap(l => l.split(" ")).map(word => (word, 1))
```
Nothing happens

When I call an action:
```
w.reduceByKey(_ + _).collect.foreach(println)
```
Spark starts the job

Parallelized collections :

```
scala> val  somedata = Array(1, 2, 3, 4, 5)
scala> val distributedData = sc.parallelize(somedata)
scala> distributedData
```

Dataset, call a distributed file :

```
scala> val distributedFile = sc.textFile("text.txt")
```

Transformations (Map & flatMap) :

```
val file = sc.textFile("test.txt")
val map = file.map(l => l.split(" "))
Val fmap = file.flatMap(l => l.split(" "))
```

Actions (Collect) :

```
val file = sc.textFile("test.txt")
file.map(l => l.split(" ")).collect()
file.flatMap(l => l.split(" ")).collect()
```

Actions (ReduceByKey) :

```
val f = sc.textFile("test.txt") val words = f.flatMap(l => l.split("
")).map(word => (word, 1))
words.collect
// Or words.take(5)
words.reduceByKey(_ + _).collect.foreach(println)
//Or take(5)
```

```
//Or (it gets (this,1),(this,1) => this,(1,1) => (this, 2))
words.reduceByKey((a,b)=>a+b).collect.foreach(println)
```

Actions (Reduce) :

```
val lines = sc.textFile("test.txt")
val lineLengths = lines.map(s => s.length)
val totalLength = lineLengths.reduce((a, b) => a + b)
val totalLength = lineLengths.reduce(_+_)
```

Persistence :

```
val f = sc.textFile("test.txt")
val w = f.flatMap(l => l.split(" ")).map(word => (word, 1)).cache()
w.reduceByKey(_ + _).collect.foreach(println)
```