Función aggregate() en Spark

La función aggregate() permite devolver un tipo de dato distinto al que contiene el RDD con el que estemos trabajando, cosa que no permite por ejemplo reduce() que devuelve el mismo tipo de dato que tenga el RDD aunque la filosofía de ambas funciones sea la misma: Aplicar una operación/es para "reducir" o "agregar" los elementos del RDD.

Al igual que fold, hay que suministrarle un valor "neutro" del tipo que vamos a devolver inicialmente.

La función aggregate necesita dos funciones como parámetros además del valor "neutro":

- Una función que combina los elementos de nuestro RDD en un "acumulador" del tipo que vamos a devolver.
- Otra función que combine dos "acumuladores" ya que cada nodo acumula sus propios resultados localmente.

Ejemplo calcular la media de un RDD con aggregate

```
scala> val rdd = sc.parallelize (List(1,2,3,3))
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[15] at parallelize at <console>:21
```

El tipo de entrada del rdd es Int, el tipo de salida será una tupla de Ints luego el acumulador "interno" será una tupla de Int's y empezará en (0,0)

- Primera función: Combina los elementos de nuestro rdd con el acumulador. `acc` es la tupla del acumulador, los `value` son los valores de nuestro RDD y el tipo a retornar tiene que ser del tipo que sea el acumulador (tupla en nuestro caso). En esta función en concreto sumamos a la primera parte de la tupla del acumulador, los valores del RDD (value's) y a la segunda parte de la tupla del acumulador un 1 para llevar el conteo de elementos sumados:

```
scala> val f1 = (acc:(Int,Int),value:Int)=>(acc._1 + value, acc._2+1)
f1: ((Int, Int), Int) => (Int, Int) = <function2>
```

- Segunda función: mergea dos acumuladores, es decir la entradas de la función son dos acumuladores (dos tuplas en nuestro caso) y hacemos la operación que sea contra esos acumuladores, siempre y cuando devuelvan el tipo del acumulador (tupla en nuestro caso). En el ejemplo necesitamos sumar juntas las primeras partes de las tuplas como primer elemento y las segundas partes de las tuplas como segundo elementos, es decir sumamos las sumas parciales del rdd y la suma de los conteos.

```
scala> val f2 = (acc1:(Int,Int),acc2:(Int,Int)) => (acc1._1 + acc2._1 , acc1._2 + acc2._2)
```

f2: ((Int, Int), (Int, Int)) => (Int, Int) = <function2>

Por último invocamos a `aggregate` para obtener la tupla de (suma,contador) con el acumulador inicialmente seteado a (0,0) y calculamos la media:

```
scala> val res = rdd.aggregate ((0,0)) (f1,f2)
res: (Int, Int) = (9,4)

scala> val avg = res._1 / res._2.toDouble
avg: Double = 2.25
```