**TRABAJANDO CON RDDs**

Arrancar el shell de spark

spark-shell

Para salir

Exit, Ctrl+c

El metodo **Parallelize** convierte una Scala Collection local en un RDD

Val lines = sc.parallelize(List(“pandas”, “i like pandas”))

El método **textFile** lee un fichero de texto desde HDFS o Local y lo transforma en un RDD de tipo String

val relato=sc.textFile("file:/home/BIT/data/relato.txt")

**map()** : que genera un nuevo RDD aplicando alguna función sobre cada línea del RDD(transformacion)

logs.map(x=>x.length).take(5)

**filter()** Toma una función y la aplica a los elementos del RDD que cumplen el filtro.(transformacion)

val jpglogs=logs.filter(x=>x.contains(".jpg"))

Existe una función **contains()** que devuelve las tuplas que contienen la cadena de caracteres que se pase como parámetro.

val jpglogs=logs.filter(x=>x.contains(".jpg"))

Existe una función **count()** que cuenta filas

sc.textFile(logs).filter(line=> line.contains(".jpg")).count()

**reduce()** Opera sobre dos elementos del RDD y devuelve un resultado

val sum = rdd.reduce((x, y) => x + y)

Otro ejemplo, operación **foldLeft** (el tipo de datos de entrada es de un tipo y la salida puede ser de otro)

**def** sum(list: List[Int]): Int **=** list.foldLeft(0)((r,c) **=**> r**+**c)

**def** sum(list: List[Int]): Int **=** list.foldLeft(0)(\_**+**\_)

Existe la función **collect()** para devolver el dataset completo

relato.collect()

**count()**, que cuenta el número de elementos (filas) de un RDD

relato.count()

Para casos en los que queramos usar collect() en datasets muy grandes podemos usar funciones como **saveAsSequenceFile()**

**take(n)**, que devuelve un array de n elementos

ips.take(5)

**saveAsTextfile(file)**, que guarda el RDD a un fichero de texto

ips.saveAsTextFile("file:/home/cloudera/iplist")

No todo es paralelizable y distribuible, la Solución: utilizar la función **aggregate**



Contiene dos funciones

* **Seqop**: operaciones secuenciales con datos de dos tipos diferentes A,B
* **Combop**: operaciones de combinación, siempre del mismo tipo B

Permite cambiar el tipo de los datos que devuelve.

Funcion **distinct(),** que elimina los duplicados.

Rdd.distinct()

Funcion **union()**, que produce una rdd que contenga elementos de ambos contenedores

Rdd.union(other)

Funcion **intersection()**, que guarda solo los elementos que esten en ambos contenedores

Rdd.intersection(other)

Funcion **subtract()**, que elimina el contenido de un rdd

Rdd.subtract(other)

Funcion **cartesian()**, que convierte en cartesianos la otra rdd

Rdd.cartesian(other)

Funcion **flatMap()**, que aplica una función a cada elemento de la rdd y devuelve una rdd del contenido de los iteradores devueltos. A menudo se utiliza para extraer palabras.(transformacion)

Rdd.flatMap(x => x.to(3))

Funcion **countByValue()**, que devuelve el numero de veces que aparece cada elemento.

Rdd.countByValue()

**Bucle** para visualizar las primeras 10 lineas.

scala> for (x <- ips.take(10) ) { print(x) }