Sistemas Distribuidos

Spark I

Carlos Gómez Gómez -1460840
Julio Velásquez Cárdenas - 1397896

Pregunta 1: Calcular el cuadrado de cada número de nuestro array data

Para empezar hemos definido nuestro array de la siguiente manera.

```
scala> val data=Array(1,5,10,11,12,21,50,51,99)
data: Array[Int] = Array(1, 5, 10, 11, 12, 21, 50, 51, 99)
```

Esto nos creará un array de enteros de 9 elementos. Para calcular el cuadrado habremos de usar la transformación *map* definiendo como fórmula que cada elemento deberá de ser multiplicado por sí mismo. De esta manera, obtendremos el siguiente resultado.

```
scala> val cuadrado=data.map(a=>a*a)
cuadrado: Array[Int] = Array(1, 25, 100, 121, 144, 441, 2500, 2601, 9801)
```

Pregunta 2: Convertir el Array de enteros data a un Array de String y luego almacenar el inverso de cada String (reverse) en un nuevo RDD

Volveremos a definir nuestro array data como se muestra en la siguiente imagen. Lo estamos definiendo como un array de enteros.

```
scala> val data=Array(1,5,10,11,12,21,50,51,99)
data: Array[Int] = Array(1, 5, 10, 11, 12, 21, 50, 51, 99)
```

Usando el siguiente comando, transformaremos este array de enteros en un array de strings gracias a la transformación *map* en la que diremos que cada elemento del array sea transformado en un string usando el *toString* y lo invertiremos usando la acción *reverse*.

```
scala> val inverso=data.map(a => a.toString.reverse)
inverso: Array[String] = Array(1, 5, 01, 11, 21, 12, 05, 15, 99)
```

Pregunta 3: Crear un nuevo RDD con los primeros 5 elementos del RDD anterior

Para crear un nuevo RDD con los 5 primeros elementos de cualquier array, se usará la acción *take()* sobre nuestro array *inverso* del ejercicio anterior como se puede observar en la captura de pantalla siguiente.

```
scala> val primeros=inverso.take(5)
primeros: Array[String] = Array(1, 5, 01, 11, 21)
```

Pregunta 4: Crear un nuevo RDD a partir de nuestro fichero README.md donde solo se almacenan las palabras de un texto sin repetición o duplicados. ¿Cuántas palabras son? ¿Qué porcentaje del total representan?

El primer paso para realizar esta pregunta es el de leer el fichero README.md, para ello se utiliza el siguiente comando.

```
scala> val text_file = sc.textFile("README.md")
text_file: org.apache.spark.rdd.RDD[String] = README.md MapPartitionsRDD[1] at textFile at <console>:24
```

Después se eliminan las palabras que son repetidas mediante el siguiente comando.

```
scala> val text_reduced = text_file.flatMap(line => line.split(" ").map(w=>(w,1))).
reduceByKey((a, b) => a+b).filter(_._2 == 1)
```

Por último para contar las palabras del texto se utiliza el siguiente comando:

```
scala> text_reduced.count()
res0: Long = 211
```

Para calcular el porcentaje se tienen que dividir el las palabras del texto reducido entre las del texto normal, para primero se tienen que contar las palabras del texto estándar. Si se quiere que el valor del porcentaje sea en decimales se han de convertir los datos de entero a float.

```
scala> val text_count = text_file.flatMap(line => line.split(" "))
text_count: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[6] at flatMap at <console>:25

scala> text_count.count()
res3: Long = 568

scala> val porcentaje = (text_reduced.count().toFloat / text_count.count().toFloat) * 100
porcentaje: Float = 37.14789
```