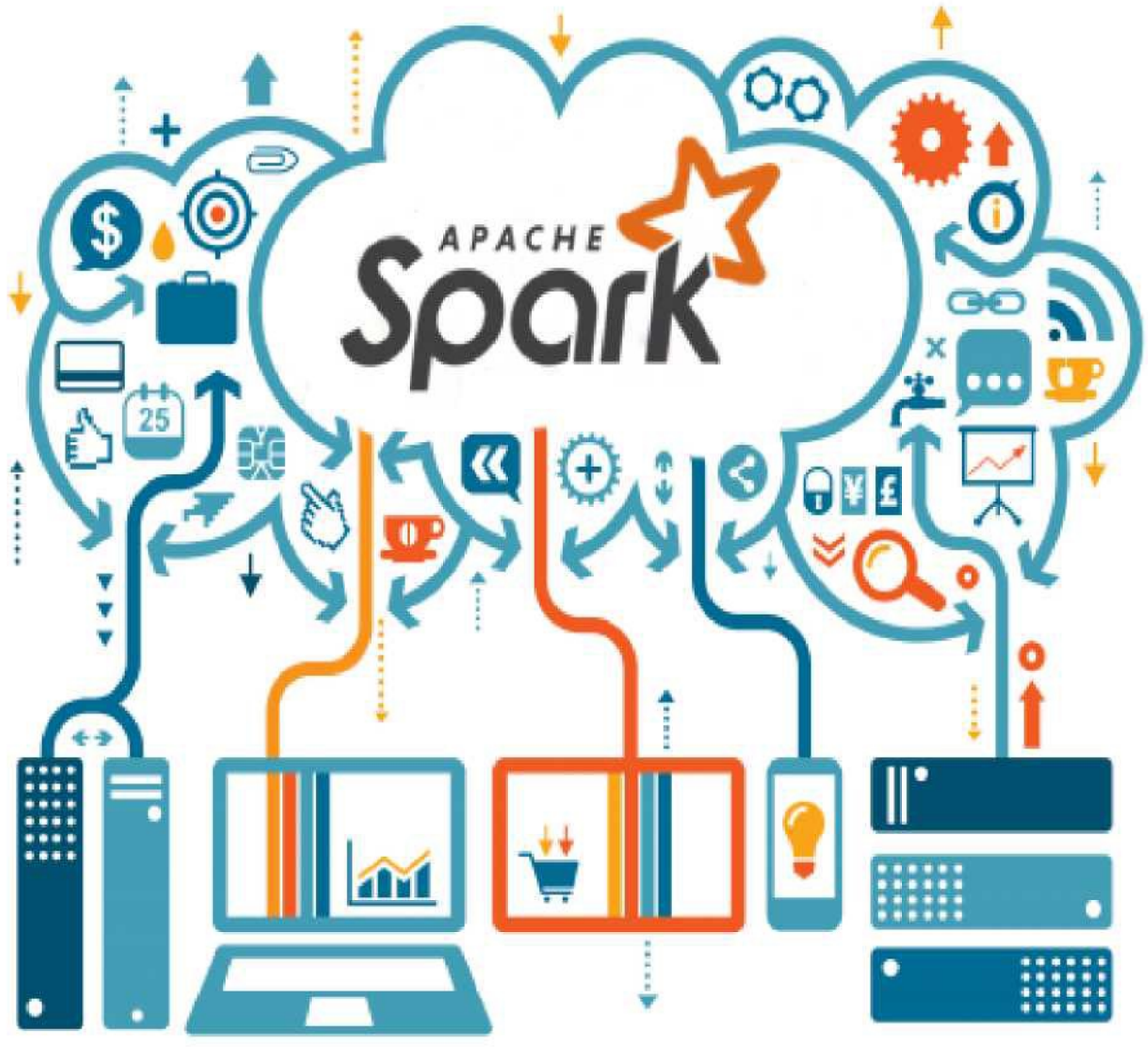
APACHE SPARK
PART II



RDD EVALUACION LENTA

Las transformaciones en RDD se evalúan perezosamente, lo que significa que Spark no comenzará a ejecutarse hasta que se **muestre o se lance una acción.**

• En lugar de pensar en un RDD que contiene datos específicos, podría ser mejor pensar en cada RDD conjunto de instrucciones sobre cómo calcular los datos que construimos a través de transformaciones.

• Spark utiliza la evaluación diferida para reducir el número de pasadas para hacerse cargo de nuestros datos agrupando las operaciones.

RDD AMPLIACIÓN

CREAR UN RDD USANDO MÉTODO PARALLELIZE:

Las particiones son unidades básicas de paralelismo en Apache Spark.
Los RDD en Apache Spark son una colección de particiones.
Ejemplo

```
val spark:SparkSession =
SparkSession.builder().master("local[*]")
        .appName("Ejemplo RDD parallelize")
        .getOrCreate()

val rdd:RDD[Int] = spark.sparkContext.parallelize(List(1,2,3,4,5))
……
```

EJEMPLOS DE TRANSFORMACIONES

1. FLATMAP(): Como map pero cada elemento puede crear cero o mas elementos

val counts = textFile.flatMap(line => line.split(" "))

2. DISTINCT(): Crea un nuevo RDD eliminando duplicados

numeros = sc.parallelize([1,1,2,2,5])

unicos = numeros.distinct()

Resultado: [1,1,2,2,5] → [1,2, 5]

FLATMAP VS MAP

val strings = Seq("1", "2", "foo", "3", "bar")

strings.map(toInt)

res0: Seq[Option[Int]] = List(Some(1), Some(2), None, Some(3), None)

strings.flatMap(toInt)

res1: Seq[Int] = List(1, 2, 3)

- De aqui viene lo de ***flat,*** la lista de flatmap se ‘alisa’

EJEMPLO DE TRANSFORMACIONES

3. COLLECT: Devuelve una lista con todos los elementos del RDD

numeros = sc.parallelize([5,3,2,1,4])

print numeros.collect()

Resultado: =[5, 3, 2, 1,4]

> Cuando se llama a collect todos los datos del RDD se envian al driver program

Hay que estar seguros que caben en memorial!

4**. COUNTBYVALUE:**Devuelve el numero de elementos del RDD

val rdd = sc.parallelize(List(1,1,2,2,2,3,3,3,3), 2)

rdd.countByValue

res: scala.collection.Map[Int,Long] = **Map(2 -> 3, 1 -> 2, 3 -> 4)**

RDD de pares clave-valor (K, V)

Son RDD donde cada elemento de la coleccion es una tupla de dos elementos.

- El primer elemento se interpreta como la clave
- El segundo como el valor
- Se contruyen a partir de otras transformaciones:

EJEMPLO:

```
val words = sc.parallelize(List("avion", "tren", "barco", "coche", "moto", "bici"), 2)
val rdd_with_key = words.keyBy(_.length) // se usa la longitud de la palabra como clave
rdd_with_key.groupByKey.collect(

)
```

res: Array[(Int, Iterable[String])] = Array((4,CompactBuffer(tren, moto, bici)), (5,CompactBuffer(avion, barco, coche)))

EJEMPLOS DE TRANSFORMACIONES

1. groupbyKey: Agrupa todos los elementos del RDD para obtener un unico valor por clave con valor igual a la secuencia de valores

> El resultado sigue siendo una coleccion, esto en un RDD

```
r = sc.parallelizeCEC'A', 1),('C', 2),('A', 3),('B', 4),('B', 5)])
 rr = r.groupByKey()
print rr.collect()
```

> Resultado: [('A', (1,3), ('C', (2,)), ('B', (4,5))]

2. sortbykey: Ordena por clave un RDD de pares (K,V)

> Si le pasas False ordena de forma inversa

```
rdd = sc.parallelize([(’A',1),(‘B‘,2),(‘C',3),('A',4),(‘A‘,5),(‘B',6)])
 res = rdd.sortByKey(False)
print res.collectasMap()
```

EJEMPLOS DE TRANSFORMACIONES

3. REDUCEBYKEY: Agrega todos los elementos del RDD hasta obtener un unico valor por clave

- El resultado sigue siendo una coleccion, esto en un RDD

r = sc.parallelizeCEC'A', 1),('C', 4),('A', 1),('B', 1),('B', 4)])
rr = r.reduceByKey(_+_)

print rr.collectAsMap()

- Resultado: [('A', 2), ('C', 4), ('B', 5)]

TRANSFORMACIONES JOIN

Realiza una operacion join de dos RDD (K,V) y (K,W) por clave para dar un RDD (K,(V,W))

EJEMPO

```
rddl = sc.parallelize([('A',1),('B',2),('C',3)]
 rdd2 = sc.parallelize([('A',4),('B',5),('C',6)])
rddjoin = rdd1.join(rdd2)
print rddjoin.collect()
```

VARIABLES BROADCAST

Las variables de difusión permiten al programador mantener una variable readonly almacenada en caché en cada máquina en lugar de enviar una copia de la misma con tareas.

Se pueden utilizar, por ejemplo, para proporcionar a cada nodo, una copia de un dataset de entrada grande, de una manera eficaz.

Todas las variables de difusión se mantendrán en todos los nodos de trabajo para su uso en una o más operaciones de Spark