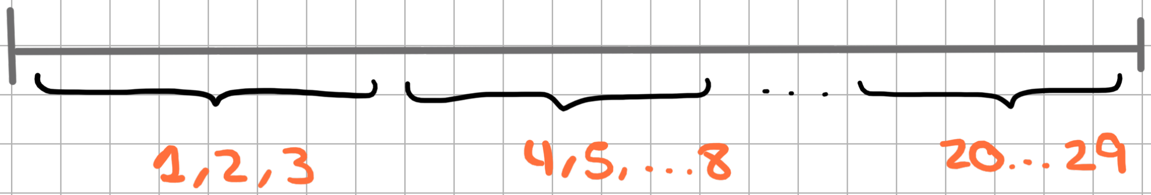
→ Spark Spark

- RDD : can apply many transformation, but it is inmutable so everytime it changes a new one is created.

  Use lazy evaluation: transformations are not executed until actions occur.

- Datasets are represented as list of entries



Dataset broken into partitions → Partitions are stored in each worker's memory

$$\text{RDD} = \overset{\text{spark context}}{\text{sc}}.\text{parallelize}\,(\overset{\text{list dataset}}{\text{list(data)}}\,,\ \underset{\text{n° of partitions}}{8})$$

Tienen muchos atributos para sacar la información

- Funciones: Spark tutorial student

→ RDD.map(f) : each item will be applied the function F

→ RDD.toDebugString(): print the transformation hierarchy

→ RDD.collect( ): gathers the entries from all partitions into the driver
↖ te devuelve todas las entries de la lista inicial.

→ RDD.count() : count the number of elements of the RDD

→ RDD.filter(F) : F returns True or False
↑ if it is true that element is passed to the new RDD
después se suele usar collect() para pritear cuales se han filtrado

→ RDD.filter(lambda x : x/2 == 0)
Another way of returnin T or F and filtering the results.

→ RDD.first() = return the first element of the RDD

→ RDD.take(n) : return the first n elements

→ RDD.takeOrdered(): list sorted in ascending order

→ RDD.top () : list ordered in descending order

→ RDD.reduce(): there are given 2 parameters.
the function should be associative and conmutative

→ RDD.takeSample(): returns an array with random sample
(seed = 1, (withreplacement = TRUE)
↳ se pueden repetir

→ RDD.countByValue(): counts a unique value in a RDD

→ RDD.flatmap(): like map, each input can be map to zero or more outputs.

map lo mete en listar y flatmap lo va metiendo en cada index

→ RDD.groupByKey(). works on pair RDD
.reduceByKey(). pair tuple (key, value)
↖ mejor
cuantas veces se repite ↗

→ RDD.mapValues() simplemente mapea todos los values sin grouparlos ni nada

→ combineByKey() and foldByKey()

→ Advanced transformations: .cache()