Un **RDD** es lo mismo que decir que estamos definiendo una **colección, dataset o conjunto de datos.** Un RDD se puede decir que es la abstracción que Spark utiliza para manejar conjunto de datos.

Un RDD son un conjunto de datos que estarán distribuidos por diferentes máquinas , de esta forma se podrá aprovechar el procesamiento paralelo de todas las máquinas.

¿Que estamos ganando con distribuir un fichero?. Por un lado estamos ganando velocidad de procesamiento ,obviamente no es lo mismo procesar un fichero en una máquina que en 8 máquinas en paralelo. Además estamos ganando una escalabilidad horizontal que nos permitirá no solo procesar datos a gran velocidad sin necesidad de aumentar las prestaciones de cada máquina sino que nos permitirá procesar cantidades grandes de datos que en una única máquina sería imposible

¿como se logra esta escalabilidad?

Un RDD será troceado en pequeños trozos para que sean cargados en N máquinas que sí podrán tratar el conjunto de datos entero.

El modelo de datos utilizado por Spark se llama Resilient Distributed Dataset (RDD),que es una colección distribuida de elementos que se pueden procesar en paralelo. Un RDD se puede crear a partir de una colección existente (una lista de Python, por ejemplo) o de un conjunto de datos externo, almacenado como un archivo en la máquina local, HDFS u otras fuentes.

Resilient Distributed Dataset (RDD) es una colección de objetos distribuida de forma inmutable. Cada RDD es dividido en múltiples particiones , las cuales serán procesadas en diferentes nodos dentro del cluster. Se puede crear RDDs de dos maneras, cargando un archivo de datos externo o mediante la distribución de una colección de objetos (ie. list o set) en el programa controlador.

- **Dataset:** Colección de elementos independientes (archivos, objetos, etc) procedentes de algún almacén de datos(fichero,bd)
- **Distributed:** Los elementos de los RDD se agrupan en particiones y pueden ser almacenados en diferentes nodos
- **Resilient:** Los RDDs recuerdan cómo fueron creados, así que si un nodo se cae,Spark tiene la capacidad de recalcular los datos perdidos en otro nodo.