MapReduce

Paradigma de programación que proporciona un sistema de procesamiento de datos paralelo y distribuido. Está pensado para la solución práctica de algunos problemas que pueden ser paralelizados, pero se ha de tener en cuenta que no todos los problemas pueden resolverse eficientemente con MapReduce, está orientado a resolver problemas con conjuntos de datos de gran tamaño.

Tiene diferentes **características**; tolerancia a fallos, resiliencia, velocidad, procesamiento paralelo, disponibilidad, escalabilidad, y rentabilidad.
El programador sólo tiene que definir el mapper, reducer y driver.

MapReduce se ejecuta en tres fases: Map, Shuffle & Sort y Reduce. Opcional la fase combiner.

- **Map:**
    - Recibe como parámetro un par clave-valor y devuelve una lista de pares clave-valor.
    - Se encarga del mapeo y se aplica en paralelo a cada ítem en la entrada de datos.
    - Se obtiene una lista de pares por cada llamada a la función map.
    - Se agrupan todos los pares con la misma clave de todas las listas y crea un conjunto por cada una de las diferentes claves únicas generadas.
    - Al finalizar el proceso esta información se almacena en el sistema de archivos local

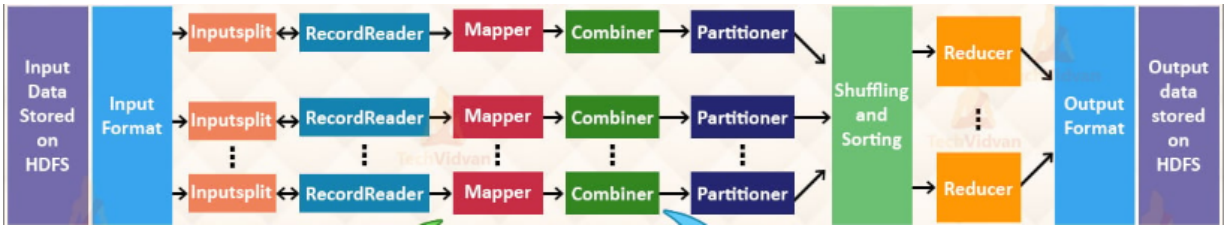
- **Shuffle & Sort**
    - Recibe los pares clave-valor producidos por varios mappers.
    - Ordena los pares clave-valor por clave.
    - Agrupa todos los pares que tienen la misma clave y los organiza en listas de valores asociados a esa clave.
    - Esta fase garantiza que todos los valores asociados a una misma clave se agrupen juntos y estén listos para ser procesados por los reducers.

- **Reduce**
    - Recibe como entrada los pares clave y lista de valores generados por la fase de Shuffle and Sort.
    - Para cada clave única, procesa la lista de valores asociados a esa clave según la lógica definida en la función Reduce.
    - Puede realizar operaciones como sumar los valores, calcular promedios, encontrar el máximo o el mínimo, etc.
    - Produce un **resultado final para cada clave** después de aplicar la función Reduce.

- **Combine (opcional)**

  Se utiliza para mejorar el rendimiento de MapReduce al reducir el volumen de datos que se transfiere entre la fase de Map y la fase de Shuffle and Sort (hace que se ejecute más rápido y permite un rendimiento adicional en los jobs MapReduce).

  Varias salidas de los mapper se reducen localmente a nivel de nodo (Ej: Varias salidas de un mismo nodo duplicadas).



- **Versión 1**

  En la primera versión, Mapreduce consta de dos componentes:

  1. **Job Tracker**: Demonio maestro dentro de hadoop, se encarga de la asignación de recursos y programación de trabajo de los jobs.
  2. **Task Tracker:** Demonio que opera en cada nodo, responsable de la ejecución de las tareas map-reduce.

- **Terminología**

  - **Nodo:** Se puede ver como una especie de ordenadores con hardware básico.
    - **Maestros:** Dividen, coordinan y distribuyen tareas y recursos en tareas más pequeñas.
    - **Trabajadores:** Realizan las tareas reducidas por los maestros sobre los datos a los que han sido asignados.
  - **Clúster:** Conjunto de nodos maestros y esclavos, que trabajan de forma coordinada para almacenar información y/o realizar procesamiento, el tamaño del conjunto dependerá de la tarea a realizar o de la velocidad que se necesite para su ejecución. Se levantan clústers en función de la necesidad obteniendo escalabilidad, procesamiento a medida, etc.
  - **Job:** Es un programa entero, una ejecución completa de Maps y Reduces sobre un dataset.
  - **Task:** Es la ejecución de un solo Map o Reduce sobre una porción de datos, normalmente un bloque

    .

- **WordCount**

  Se basa en la lectura de un archivo de texto y el recuento de la frecuencia de aparición de palabras. Tanto la entrada y la salida son archivos en forma de texto, la *salida consta de una palabra y su correspondiente frecuencia.*

  Cada **mapper** toma una línea como entrada y la divide en palabras, crea un par clave-valor, **shuffle and short** hace que los elementos con la misma clave queden en líneas consecutivas y lo envía al **reducer** para que sume los recuentos de frecuencias por cada palabra y por salida genere un <k,v> con k = palabra y v = suma de frecuencias de k.

A continuación, se puede ver gráficamente cómo funciona el *WordCount*: