1. Big data y Almacenamiento Distribuido

1.1. Big Data

BIg Data datos que no pueden ser procesados por métodos tradicionales.
Se caracterizan por las 5 *V's*:

- Volumen: cantidad de datos.
- Velocidad: velocidad a la que se generan los datos.
- Variedad: variedad de datos.
- Veracidad: calidad de los datos.
- Valencia: valor de los datos.

1.1.1. Volumen

El impedimento de las computadoras a manejar grandes catidades de datos.
Cluster: un conjunto de computadoras que trabajan enconjunto y pueden ser vistas como un sistema único.

1.1.2. Velocidad

La velocidad con la cual se generan los mismos. Para procesar los datos se necesita que el tiempo de procesamiento sea menor al tiempo de generación de los datos, sino estos se aculan y el algoritmo puede descartar una gran volumen de datos.

1.2. Clase 6 - Spark - 1 - Spark

Definición 1.1. (Map-Reduce) Procesamiento distribuido de datos utilizando un cluster.

- Modelo de programación para procesar grandes volúmenes de datos.
- Sugue de la necesidad de procesar grandes volúmenes de datos de forma escalable.
- El usuario especifica una funcion una función **map** que procesa un par clave/valor para generar un conjunto de pares clave/valor intermedios.
- Se debe especificar una función **reduce** que combina todos los valores asociados a una misma clave intermedia.

Definición 1.2. (Map)

- Transfroma nuestros datos.
- Debe ser aplicada a cada dato de nuestro ser.

- Puede ser paralelizada y distribuirse entre las distintas máquinas de un cluster.

Algunas diferencias dependientes de la implementación:

- **Hadoop:** $Map(k, v) \rightarrow list(k2, v2)$
- **spark:** $Map(r) \rightarrow list(r')$

Definición 1.3. (Reduce)

- Combina los resultados del map.
- Es necesario procesar los datos de todas las máquinas del cluster.
- Reduce locales en paralelo y reduce entre máquinas mediente esta de shuffle & sort.

Algunas diferencias dependientes de la implementación:
Hadoop: $RecudeByKey((k, v), f) \rightarrow list(k, v)$

- El sistema agrupa todos los registros para los cuales la clave es la misma.
- Requiere que todos los registros de igual clave estén en la misma máquina que ejecute el reduce: Shuffle & Sort.

Spark:

- La función reduce toma dos valores para dar como resultado la combinación de ambos.
- El resultado de un reduce estre dos registros es un input del siguiente reduce.
- Opereaciones **conmutativas** y **asociativas** de modo de poder ejecutarse distribuidas.

Definición 1.4. (Cluster) Conjunto de computadoras que trabajan juntas y pueden ser vistas como un sistema único.

Definición 1.5. (Almacenamiento distribuido)

- FIleSystem Distribuido (DFS)
- Encargado de gastionar cómo y dónde guardar la información en una computadora, y cómo poder consultarla.
- Almacenar grandes volúmenes de datos en multiples equipos.
- Replicación de datos para tolerancia a fallos.
- Tolerancia a fallos.
- Alta disponibilidad (seguir funcionando aunque un nodo falle).

- Relativo a bajo costo.

Ejemplos de SD:

- GDS (Google File System)
- HDFS (Hadoop Distributed File System)
- CEPH (Ceph File System)
- S3 (Amazon Simple Storage Service)

1.3. Clase 6 - Spark - 2 - Spark

1.4. YouTube

MapReduce vs Spark link:

- MapReduce: procesamiento de datos distribuido en disco.
- Spark: procesamiento de datos distribuido en memoria.
- Spark es 100 más rápido que MapReduce.
- Spark es más fácil de usar que MapReduce.
- Spark es más flexible que MapReduce.
- Spark es más costoso que MapReduce.
- Spark es más complejo que MapReduce.
- Spark es más nuevo que MapReduce.

Spark trabaja con RDDs (Resilient Distributed Datasets) que son colecciones de objetos que se pueden dividir en particiones y distribuir entre los nodos del cluster. Los RDDs son inmutables y tolerantes a fallos.
Los RDDs:

- Spark utiliza RDDs para almacenar datos.
- Son como tuplas o listas.
- Datos Distribuidos.
- Tolerantes a fallos.
- Opereaciones paralelizables.
- Habilidad para datos de multiples fuentes.

Opereaciones:

- Transformaciones: crean un nuevo RDD a partir de uno existente: map, filter, flatMap, sample, union, intersection, distinct, groupByKey, reduceByKey, sortByKey, join, cogroup, cartesian.
  - RDD.filter(): Aplica una función y devuelve los elementos que son verdaderos, parecido a filter de python.
  - RDD.map(): Transforma cada elemento sin cambiar el número de elementos, parecido a pandas.apply().
    Estraer la primer letra de una lista de nombres.
  - RDD.flatMap(): Transforma cada elemento y cambia el número de elementos.
    Convertir el corpus de un text a una lista de palabras.
- Acciones: devuelven un valor al programa driver después de ejecutar un cálculo en un RDD: reduce, collect, count, first, take, takeSample, takeOrdered, saveAsTextFile, saveAsSequenceFile, saveAsObjectFile, countByKey, foreach.
  - Collect: Devuelve todos los elementos del RDD como una matriz.
  - Count: Devuelve el número de elementos en el RDD.
  - First: Devuelve el primer elemento del RDD.
  - Take: Devuelve un número especificado de elementos del RDD.
- Lazy evaluation: las transformaciones no se ejecutan hasta que no se ejecuta una acción.