- **Introducción a la Ciencia de Datos (Optativa)**

2025

ICD2025

In pioneer days they used oxen for heavy pulling, and when one ox couldn’t budge a log, they didn’t try to grow a larger ox. We shouldn’t be trying for bigger computers, but for more systems of computers.

—Grace Hopper

Big Data

IT

“Big Data”, es una referencia a los sistemas que manipulan grandes conjuntos de datos. Las dificultades más habituales en estos casos se centran en la captura, el almacenamiento, búsqueda, compartición, análisis y visualización.

SOFTWARE

"Big Data", es un término aplicado a conjuntos de datos que superan la capacidad del software habitual para ser capturados, gestionados y procesados en un tiempo razonable. Los tamaños del "Big Data" se hallan constantemente en aumento.

- ¿Cuántos datos hay en Internet?
- *¿Qué tan grande es el Big Data?*
- ¿Qué factores acelerarán aún más la generación de datos?

- Se estimó que la cantidad de datos en el mundo era de 44 zettabytes a principios de 2020.

- Para 2025, se espera que la cantidad de datos generados cada día alcance los 463 exabytes en todo el mundo.

- Google, Facebook, Microsoft y Amazon almacenan al menos 1200 petabytes de información.

Medida	Equivalencia
Byte	8 bits
Kilobyte	1024 bytes
Megabyte	1024 kilobytes
Gigabyte	1024 megabytes
Terabyte	1024 gigabytes
Petabyte	1024 terabytes
Exabyte	1024 petabytes
Zettabyte	1024 exabytes
Yottabyte	1024 zettabytes

Source: https://estadiferencias.blogspot.com/

NOTA:

1PB equivale a 20 millones de archivos de ~47,6 MB

- El mundo gasta casi USD 1 millón por minuto en productos básicos en Internet.

- Para 2025, habría 75 mil millones de dispositivos de Internet de las cosas (IoT) en el mundo.

- Para 2030, nueve de cada diez personas de seis años o más serían digitalmente activas.

Source: https://seedscientific.com/

- Terabytes.
- Transacciones.
- Tablas, archivos.

[~ >30 TB]

- Datos estructurados (BDR, hojas de cálculo) (~%20).
- Datos no estructurados (imágenes, videos).
- Semi estructurados (HTML, JSON)

- Batch.
- Real/near time.
- Streams.

Fuentes: personas vs. IOT

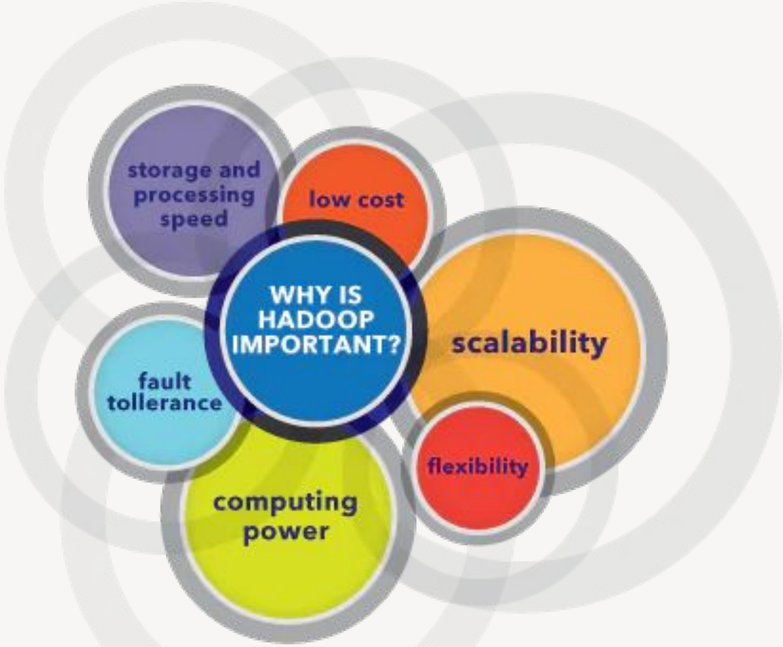
- Correlaciones.
- Estadístico.
- Eventos.

- Integridad.
- Disponibilidad.
- Responsabilidad.

Dato crudo → sistema → información

Una herramienta para Big Data...

Source: https://hadoop.apache.org/



Source:
https://www.sas.com/es_ar/insights/big-data/hadoop.html#hadoopimportance

Data Analytics

- **Batch processing**
  - MapReduce
  - Spark
- **Stream processing**
  - Spark streaming
  - Apache Flink
- **Graph processing**
  - Pregel/Giraph
  - GraphX
- **Query processing**
  - Pig or HIVE

Data Management

- **File-system**
  - GFS/HDFS
- **Distributed database**
  - BigTable or Hbase
  - Spanner

Resource Management

- **Cluster manager**
  - Mesos or YARN
- **Co-ordination service**
  - ZooKeeper or Chubby

Is Hadoop Dead? The Rise of Delta Lake and Iceberg - Java Code Geeks
https://www.javacodegeeks.com/2025/04/is-hadoop-dead-the-rise-of-delta-lake-and-iceberg.html

Is Hadoop Dead? A Deep Dive into Its Rise, Fall, and What Still Survives | by Tauseef | Medium
https://tauseef-09.medium.com/is-hadoop-dead-a-deep-dive-into-its-rise-fall-and-what-still-survives-630edac31446

Almacenamiento y análisis de datos de Hadoop

Año	Capacidad de almacenamiento	Velocidad de transferencia	Tiempo requerido
1990	1370 MB	4,4 MB/s	¿?
Más de 20 años después	1 TB	100 MB/s	¿?

Almacenamiento y análisis de datos de Hadoop

Año	Capacidad de almacenamiento	Velocidad de transferencia	Tiempo requerido
1990	1370 MB	4,4 MB/s	~ 5 minutos
Más de 20 años después	1 TB	100 MB/s	~ 3 hs.

- **¿Cuál sería la forma de reducir el tiempo de lectura?**

Almacenamiento y análisis de datos de Hadoop

- **¿Cuál sería la forma de reducir el tiempo de lectura?**

La forma de reducir el tiempo es leer desde múltiples discos a la vez

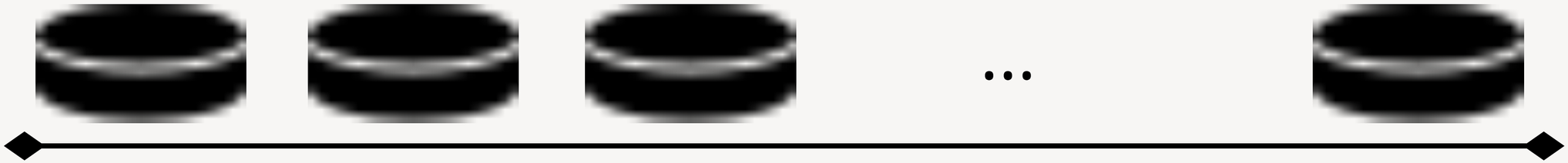
Problema

¿Qué tiempo llevaría leer (en paralelo) 1 TB si se dispone de 100 dispositivos de almacenamiento con un centésimo de un 1TB almacenado en c/u de ellos?

Almacenamiento y análisis de datos de Hadoop

Problema

¿Qué tiempo llevaría leer (en paralelo) 1 TB si se dispone de 100 dispositivos de almacenamiento con un centésimo de un 1TB almacenado en c/u de ellos?



1TB

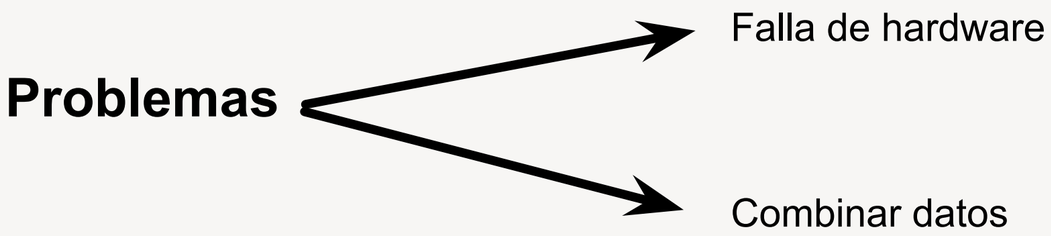
Lectura ~ 2 minutos

Usar solo una centésima de disco puede parecer un desperdicio. Pero podemos almacenar 100 conjuntos de datos, y proporcionar acceso compartido a ellos. Es probable que los usuarios de un sistema de este tipo estén dispuestos a compartir el acceso a cambio de tiempos de análisis más cortos, y estadísticamente, que sus trabajos de análisis probablemente se diseminarían a lo largo del tiempo.

¿Qué problemas existen?

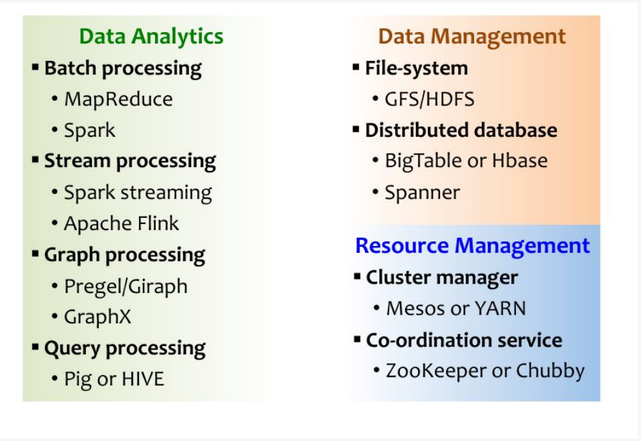
Almacenamiento y análisis de datos de Hadoop

¿Qué problemas existen?



<u>Una opción:</u>

Hadoop es una plataforma confiable y escalable para almacenamiento y análisis de datos. Además, debido a que se ejecuta en hardware básico y es de código abierto, es asequible.



HDFS

Hadoop Distributed File System

- Google File System (GFS)
- Open source: Hadoop Distributed File System (HDFS)

Sistema de archivos (filesystem): métodos y estructuras de datos que un SO utiliza para seguir la pista de los archivos de un disco o partición; es decir, es la manera en la que se organizan los archivos en el disco.

Cuestiones de diseño de HDFS

- Archivos muy grandes (hasta los petabytes)
- Acceso a datos en tiempo real (write-once, read-many-times)
- Hardware básico

¿En qué escenario no es una buena opción HDFS? (al menos por ahora)...

- Acceso a datos de baja latencia
- Montones de archivos pequeños

HDFS: Blocks

(concepto base)

- **Sistema de archivo para un solo disco**

Tamaño de bloque: cantidad mínima de datos que puede leer o escribir.

Los bloques del sistema de archivos suelen tener un tamaño de algunos kilobytes, mientras que los bloques de disco normalmente tienen 512 bytes (df y fsck).

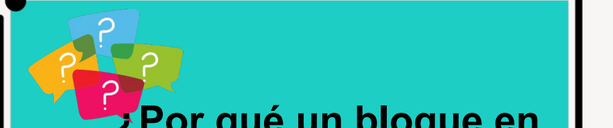
- **HDFS**

Tamaño de bloque: 128 MB por defecto.

Un archivo en HDFS que es más pequeño que un solo bloque no ocupa el valor total de almacenamiento subyacente de un bloque (por ejemplo, un archivo de 1 MB almacenado con un tamaño de bloque de 128 MB utiliza 1 MB de espacio en disco, no 128 MB.)

HDFS: Blocks

(concepto base)



¿Por qué un bloque en HDFS es tan grande?

Razón: minimizar el costo de las búsquedas. Si se poseen menos bloques, se poseen menos búsquedas, y para poseer menos bloques, se debe aumentar el tamaño de estos. Por lo tanto, la transferencia de un archivo grande hecho de múltiples bloques opera a la velocidad de transferencia del

<u>Un cálculo rápido:</u>

Si el tiempo de búsqueda es de alrededor de 10 ms. y la velocidad de transferencia es de 100 MB/s, para hacer que el tiempo de búsqueda sea el 1% del tiempo de transferencia,

¿Cuánto debe ser el tamaño del bloque?

HDFS: Blocks

(concepto base)

<u>Un cálculo rápido:</u>

Si el tiempo de búsqueda es de alrededor de 10 ms. y la velocidad de transferencia es de 100 MB/s, para hacer que el tiempo de búsqueda sea el 1% del tiempo de transferencia, ¿Cuánto debe ser el tamaño del bloque?

El tamaño del bloque debe ser de 100 MB. El valor predeterminado es en realidad 128 MB, aunque muchas instalaciones HDFS usan tamaños de bloque más grandes.

* Esta cifra seguirá en alza a medida que las velocidades de transferencia crezcan con las nuevas generaciones de unidades de disco.

HDFS: Blocks

(concepto base)

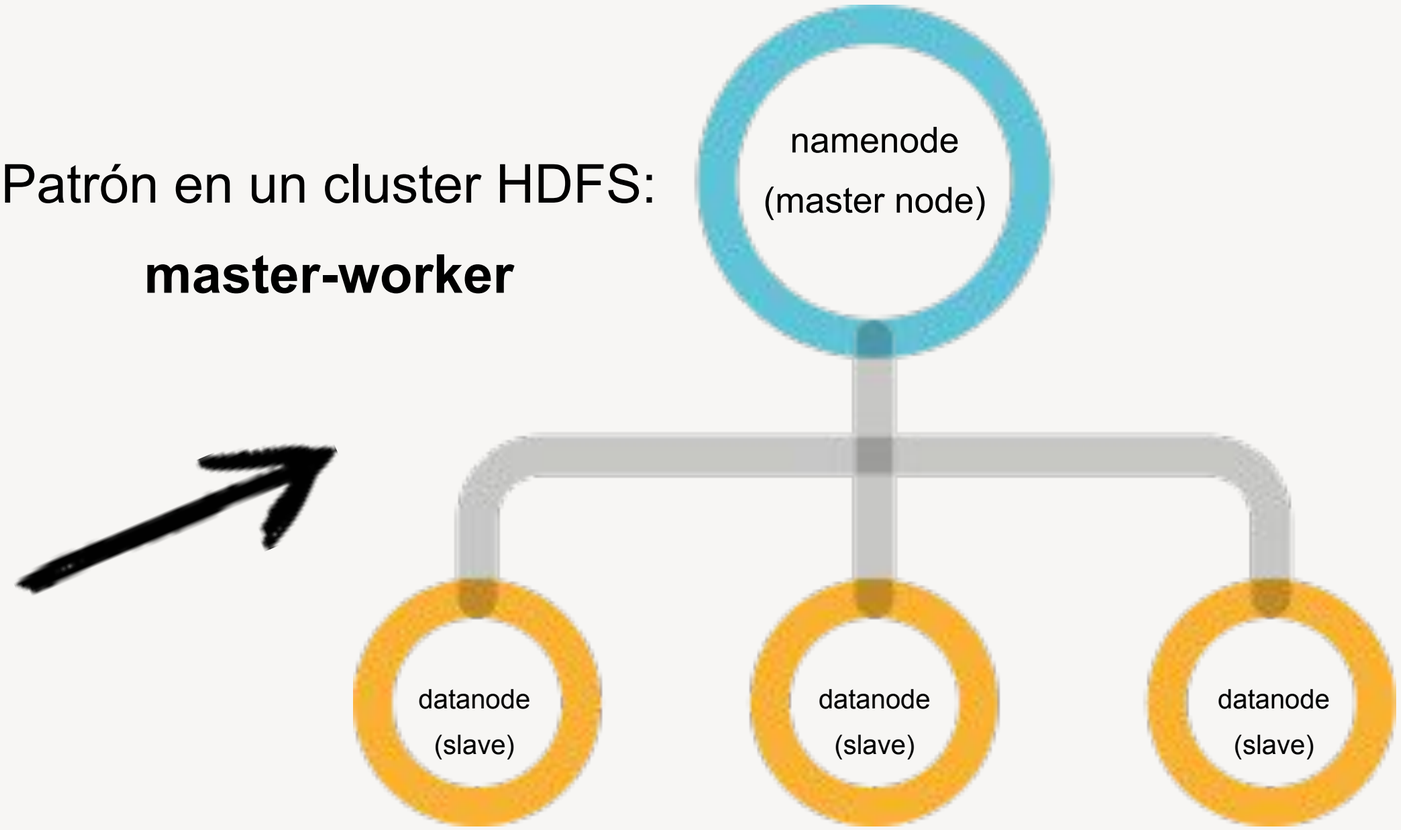
Beneficios de la abstracción de bloques

- **Simplifica la administración del almacenamiento (easy storage): Los bloques son de tamaño fijo, es fácil calcular cuántos se pueden almacenar en un disco determinado.**
- **Elimina los problemas de metadatos, pues los bloques son solo trozos de datos que se almacenan, los metadatos de archivos como ser los permisos, no necesitan almacenarse con los bloques, por lo que otro sistema puede manejar esto por separado.**
- **Integridad de datos (data integrity): Los bloques se adaptan bien a la replicación para proporcionar tolerancia y disponibilidad ante fallas. Para prevenir los problemas ante bloques dañados y fallas del disco, cada bloque se replica en una pequeña cantidad de máquinas separadas físicamente (normalmente tres -redundancia). Por lo tanto, si un bloque no está disponible, se puede leer una copia desde otra ubicación de forma transparente para el cliente. Un bloque que ya no está disponible debido a daños o fallas de la máquina se puede replicar desde sus ubicaciones alternativas a otras máquinas para devolver el factor de replicación al nivel normal.**

HDFS: namenode, datanode

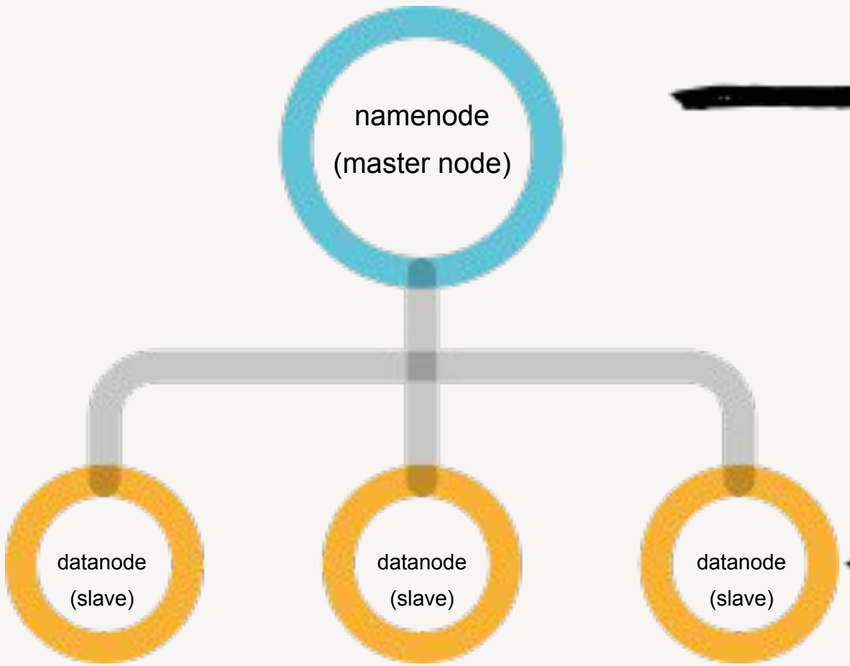
Patrón en un cluster HDFS:

master-worker



Qué bloques componen cada fichero,
en qué orden, y en qué máquinas están almacenados.

HDFS: namenode, datanode

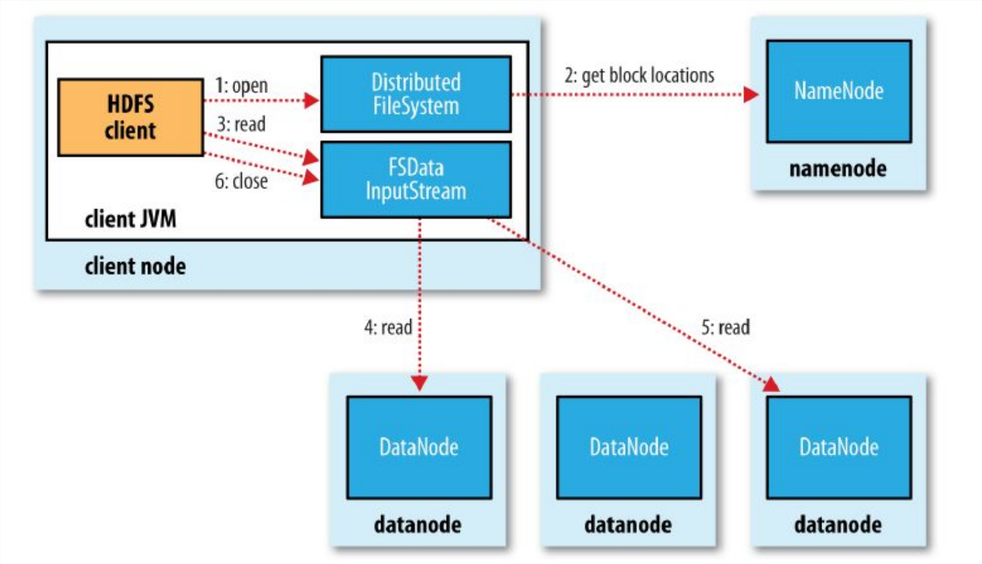


- Administra el espacio de nombres del sistema de archivos.
- Mantiene el árbol del sistema de archivos (construido a partir de los metadatos).
- Conoce los datanodes en los que se encuentran todos los bloques para un archivo dado.
- No almacena las ubicaciones de bloques de forma persistente, esta información se reconstruye a partir de los datanodes cuando se inicia el sistema.
- Sin el namenode, no puede funcionar el sistema de archivos.

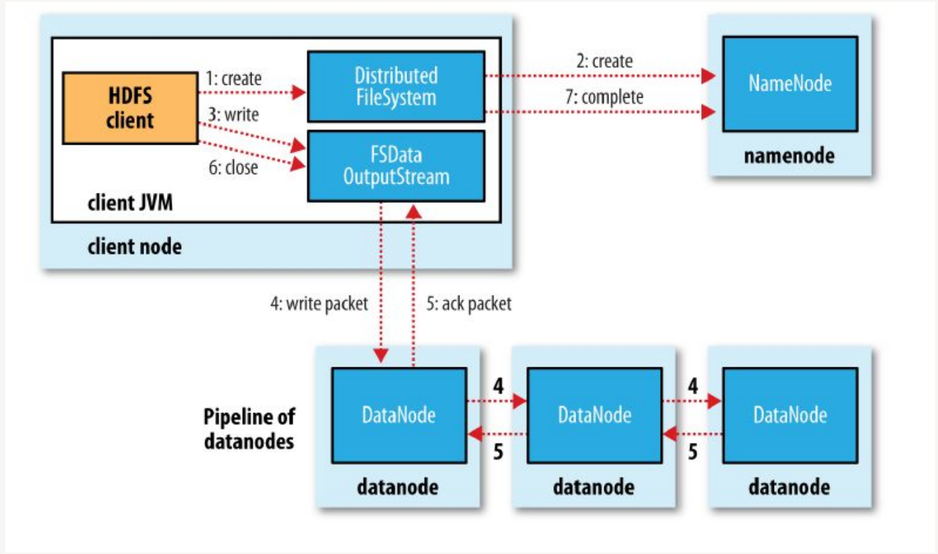
- Almacenan y recuperan bloques cuando los clientes o el namenode lo solicita.
- informan periódicamente al namenode con listas de bloques que están almacenando.
- se encarga de distribuir los bloques entre los diferentes datanodes procurando que estos estén balanceados.

Hay muchas formas de interactuar con HDFS,
incluidas Ambari Views, HDFS Web UI, WebHDFS y la línea de comando.

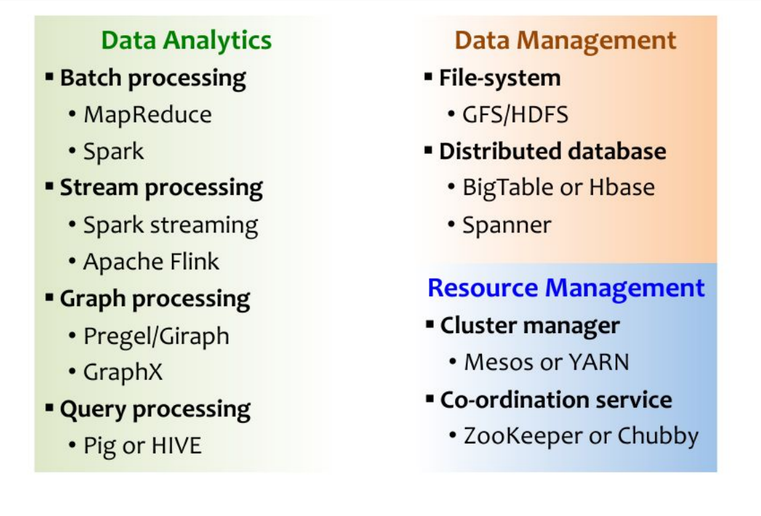
HDFS: Lectura y escritura



- El cliente solicita al namenode que le proporcione un fichero.
- El namenode consulta en la tabla de metadatos los bloques que componen el fichero y su localización en el clúster.
- El namenode devuelve al cliente una lista de bloques, así como los equipos en los que puede encontrar cada uno de ellos.
- El cliente contacta con un equipo para obtener cada bloque.
- El cliente compone el archivo.



- El cliente solicita al namenode realizar una escritura de un archivo.
- El namenode realiza algunas comprobaciones previas, para comprobar que se puede escribir el fichero y que el cliente tiene permisos para hacerlo.
- El cliente particiona el fichero en bloques, que escribirá de forma secuencial. Para cada bloque, el namenode le proporciona una lista de datanodes en los que se escribirá.
- El cliente contacta con el primer datanode para pedirle que escriba el bloque. Este datanode se encargará de propagar el bloque al segundo, etc.
- El último datanode en escribir el bloque devolverá una confirmación (ACK) hacia atrás, hasta que el primer datanode mandará la confirmación al cliente.
- Una vez que el cliente ha escrito todos los bloques, manda una confirmación al namenode de que el proceso se ha completado.



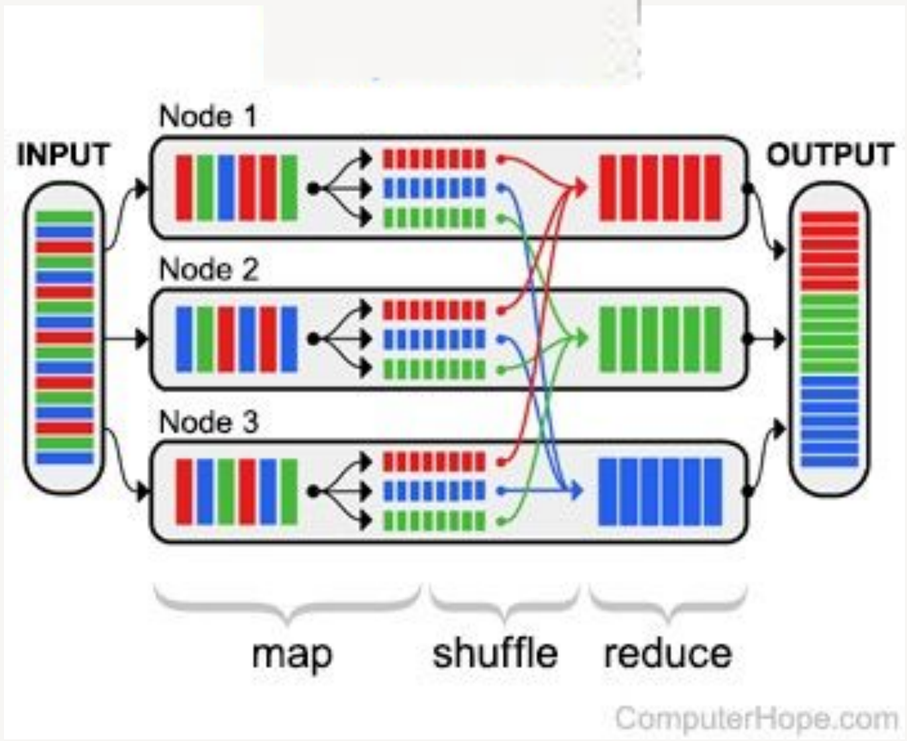
MapReduce: ¿Qué es?

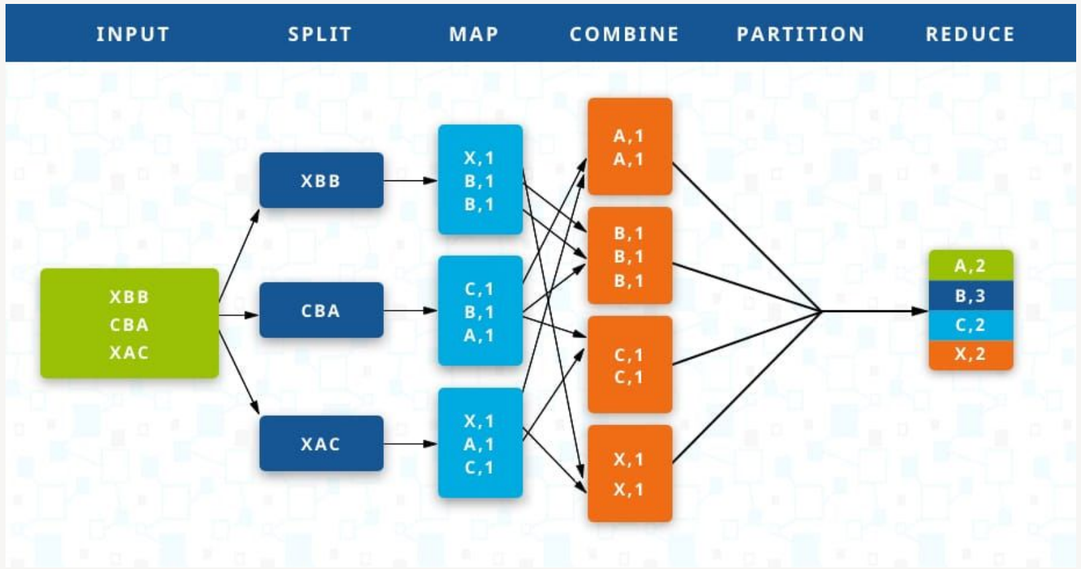
- Un modelo de programación.
- Brinda soporte a la computación paralela sobre grandes colecciones de datos (Big Data) en grupos de computadoras (nodos) y al commodity computing.
- El Algoritmo MapReduce está inspirado principalmente en el modelo de Programación Funcional (métodos: Map y Reduce).
- MapReduce es un algoritmo de procesamiento de datos distribuidos, introducido por Google en su documento técnico de MapReduce (“MapReduce: Simplified Data Processing on Large Clusters” - 2004).
- Utiliza la técnica Divide and Conquer para procesar los datos, es decir, divide la tarea de entrada en sub tareas más pequeñas y manejables (deben ser ejecutables de forma independiente) para ejecutarlas en paralelo.
- Se han escrito implementaciones de bibliotecas de MapReduce en diversos lenguajes de programación como C++, Java y Python.

MapReduce: ¿Cúando se aplica?

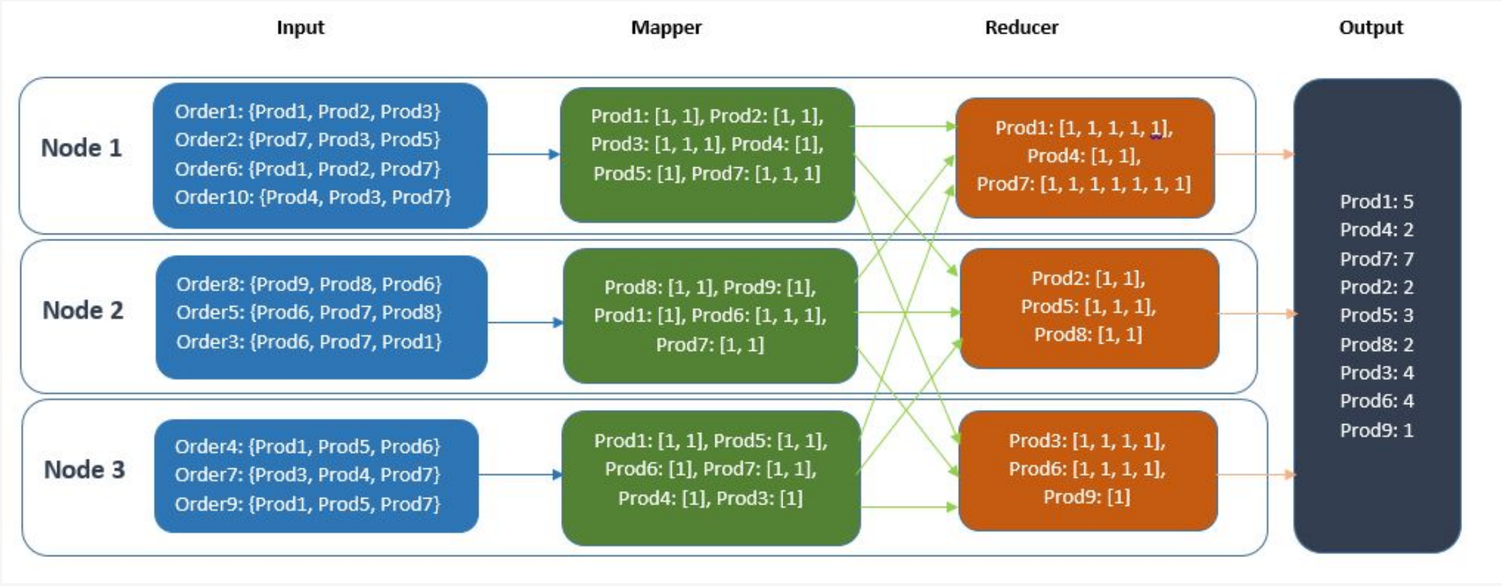
- MapReduce se emplea en la resolución práctica de algunos algoritmos susceptibles de ser paralelizados o abordables con las operaciones de map() y de reduce().
- MapReduce no es la solución para cualquier problema, de la misma forma que cualquier problema no puede ser resuelto eficientemente por MapReduce. Por regla general se abordan problemas con datasets de gran tamaño, alcanzando los petabytes de tamaño. Es por esta razón que suele ejecutarse en sistema de archivos distribuidos (GFS, HDFS).
- Los desarrolladores ETL (extract, transform, load) son los principales beneficiarios del uso de este tipo de modelo de programación.

MapReduce





Source:
https://vipanchikatthula.github.io/post/mapper-reducer-implementation/



Source:
https://www.alachisoft.com/resources/docs/ncache/prog-guide/mapreduce-overview.html

MapReduce: función Map

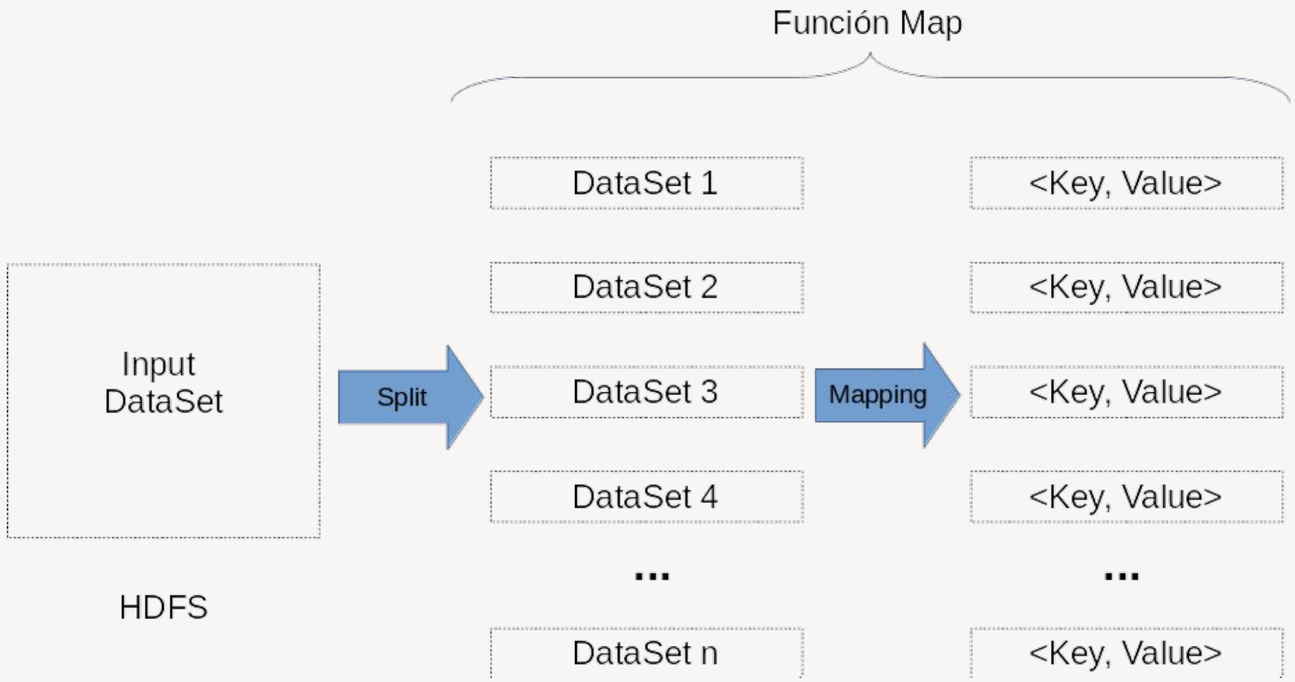
Toma tareas de entrada (por ejemplo un DataSet) y los divide en subtareas más pequeñas (divide el DataSet en partes). Luego realiza el cálculo requerido en cada subtarea en paralelo.

A su vez, la función Map, contempla dos pasos:

a- **Splitting**: paso encargado de la división del DataSet de entrada en Sub-DataSets más pequeños.

b- **Mapping**: toma los Sub-DataSets más pequeños y realiza la acción o cálculo requerido en cada uno de ellos.

La salida de la función Map, es un conjunto de pares de clave y valor de la forma <Key, Value>.



MapReduce: función Shuffle

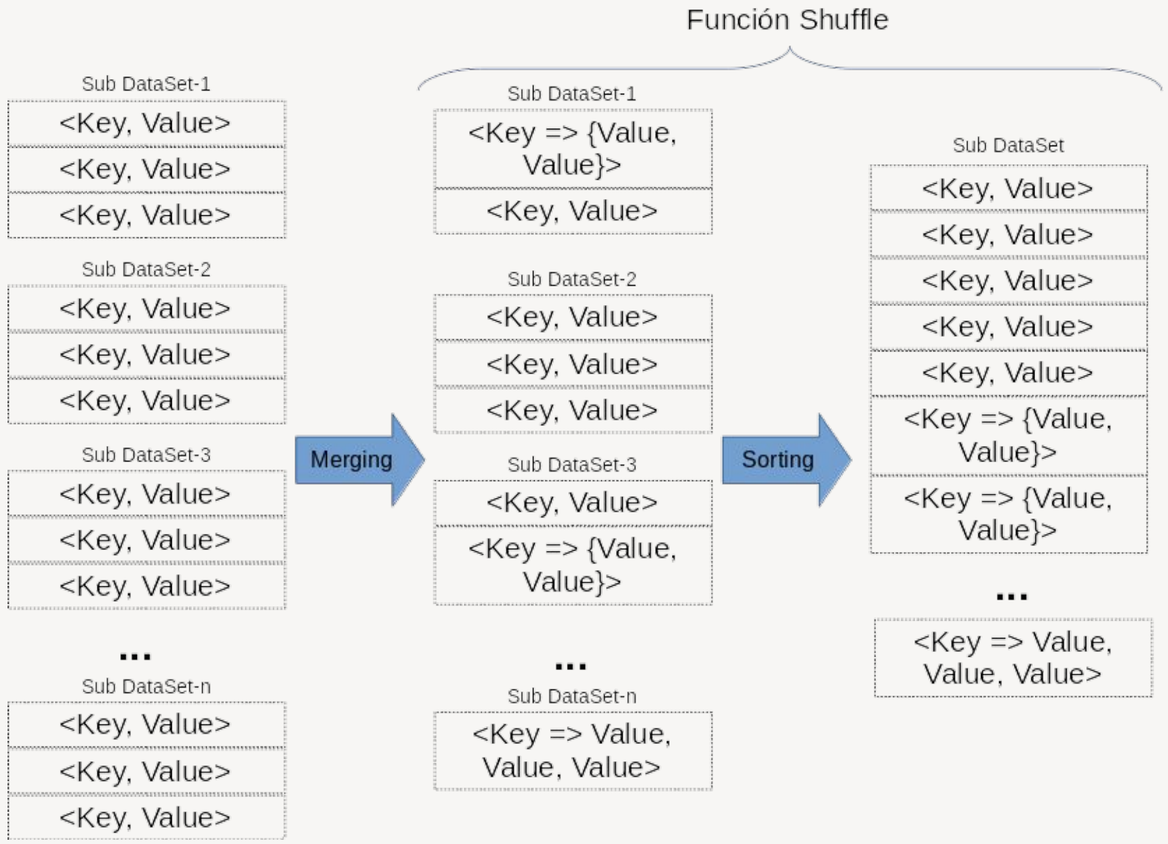
La función Shuffle (barajar) también es conocida como "Función de combinación".

Toma salida la función Map y realiza los siguientes pasos para cada <Key, Value>:

a- **Merging**: combina todos los pares <Key, Value> que tienen las mismas claves (es decir, agrupa los pares <Key, Value> por Key). Este paso devuelve <Key, List <Value>>.

b- **Sorting**: toma la entrada del paso Merging y ordena todos los pares <Key, Value> por Key.

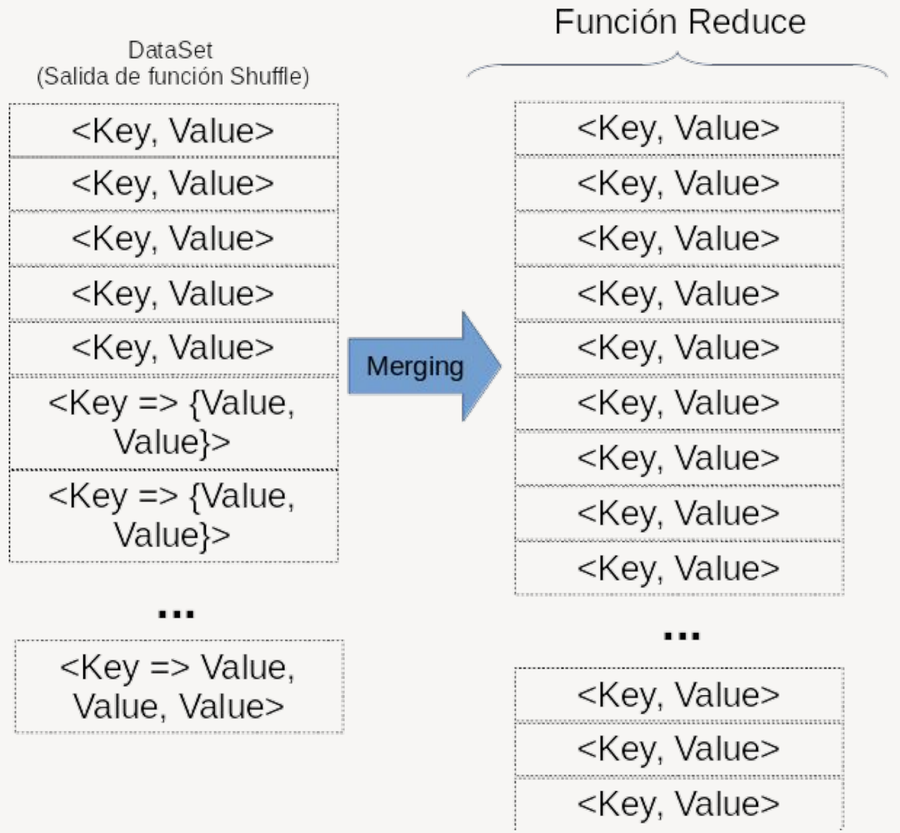
La salida de la función Shuffle, es <Key, List <Value>> pero con pares <Key, Value> ordenados.

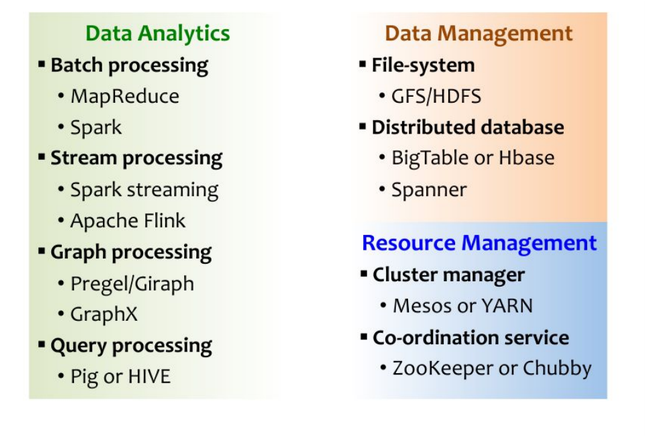


MapReduce: función Reduce

Toma los datos arrojados por la función Shuffle (<Key, List <Value>>) y realiza operaciones de reducción:

El resultado de la función Reduce es similar a la salida del de la función Map. Sin embargo, en la salida de Reduce, los pares <Key, Value> son diferentes de los pares <Key, Value> de la función Map, pues pares finales <Key, Value> son pares computados y ordenados.





Pig

Plataforma relativamente fácil para crear aplicaciones Apache MapReduce.

Motivaciones:

- **Crear aplicaciones MapReduce con mayor abstracción (> conocimiento en los datos).**
- **Permitir realizar análisis ad-hoc de grandes conjuntos de datos.**
- **Limitaciones de MapReduce (rígido, flujo de datos de una sola entrada).**

Google a YAHOO! Microsoft

Características de Pig

- **Lenguaje de flujo de datos (dataflow language)**

Problema

Supongamos que se desea obtener el conjunto de urls de páginas clasificadas como correo no deseado que tienen un alto nivel de pagerank.

Características de Pig

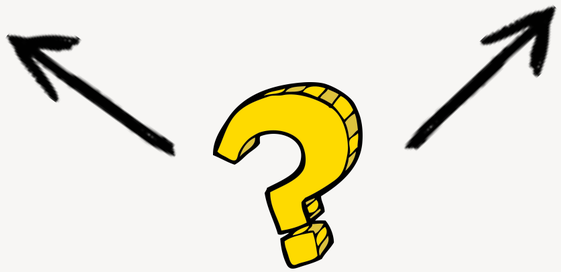
- **Lenguaje de flujo de datos (dataflow language)**

Problema

Supongamos que se desea obtener el conjunto de urls de páginas clasificadas como correo no deseado que tienen un alto nivel de pagerank.

```
spam_urls = FILTER urls BY isSpam(url);
culprit_urls = FILTER spam_urls BY pagerank > 0.8;
```

```
spam_urls = FILTER urls BY pagerank > 0.8;
culprit_urls = FILTER spam_urls BY isSpam(url);
```



Características de Pig

- Capacidad de operar sobre archivos de entrada sin ninguna información de esquema (quick start and interoperability).

¿Qué sucede con la coherencia transaccional?

Pig solo admite cargas de trabajo de análisis de datos de solo lectura, y esas cargas de trabajo tienden a centrarse en el escaneo, por lo que no se requiere la coherencia transaccional ni las búsquedas basadas en índices.

Características de Pig

Pig Latin es un lenguaje de programación no tradicional enfocado en el flujo de datos en lugar de las operaciones de programación tradicionales utilizadas por lenguajes como Java o Python.

Un script de Pig Latin especifica uno o más conjuntos de datos de entrada y luego identifica las operaciones que se aplicarán. Estas operaciones pueden incluir filtrado de registros, unión de dos conjuntos de datos y división de un conjunto de datos según ciertos criterios.
Luego, se define cómo escribir los datos en una o más fuentes de salida.

Pig Lat
(SQL) y

:o nivel