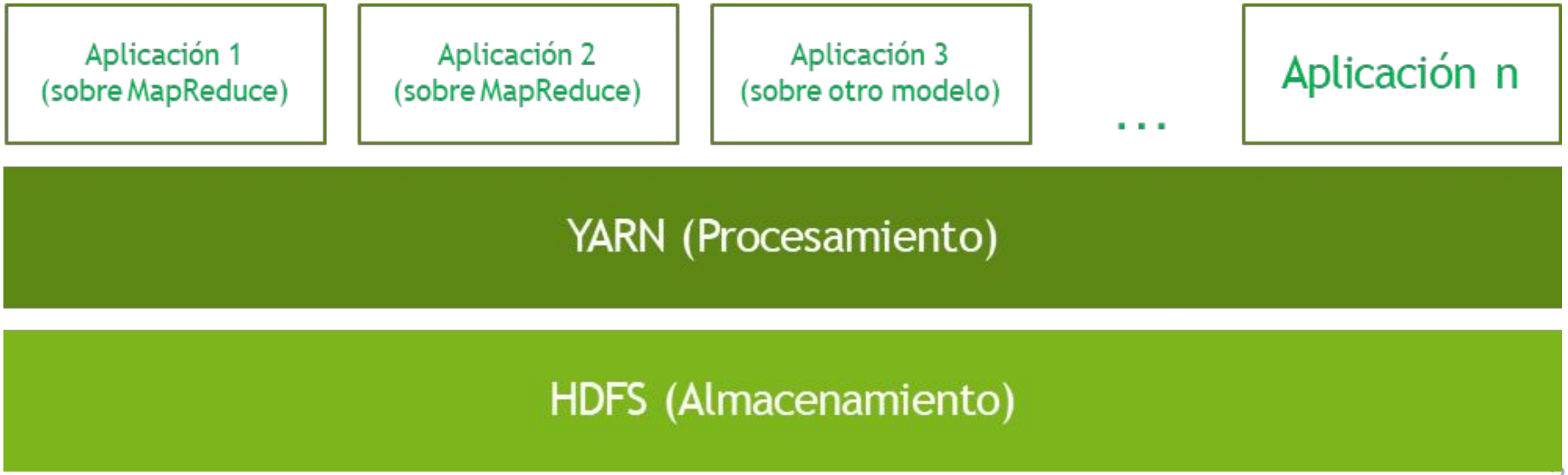
Introducción Hadoop

HDFS

Hadoop Core

El core de Hadoop está formado por HDFS y YARN, junto con un conjunto de librerías generales.



HDFS

Hadoop Distributed File System es un sistema de ficheros distribuido diseñado para ejecutarse sobre **hardware commodity** y optimizado para almacenar ficheros **de gran tamaño.**

Tiene capacidad para **escalar horizontalmente** hasta volúmenes de Petabytes y miles de nodos, y está diseñado para poder dar soporte a **múltiples clientes** con acceso concurrente.

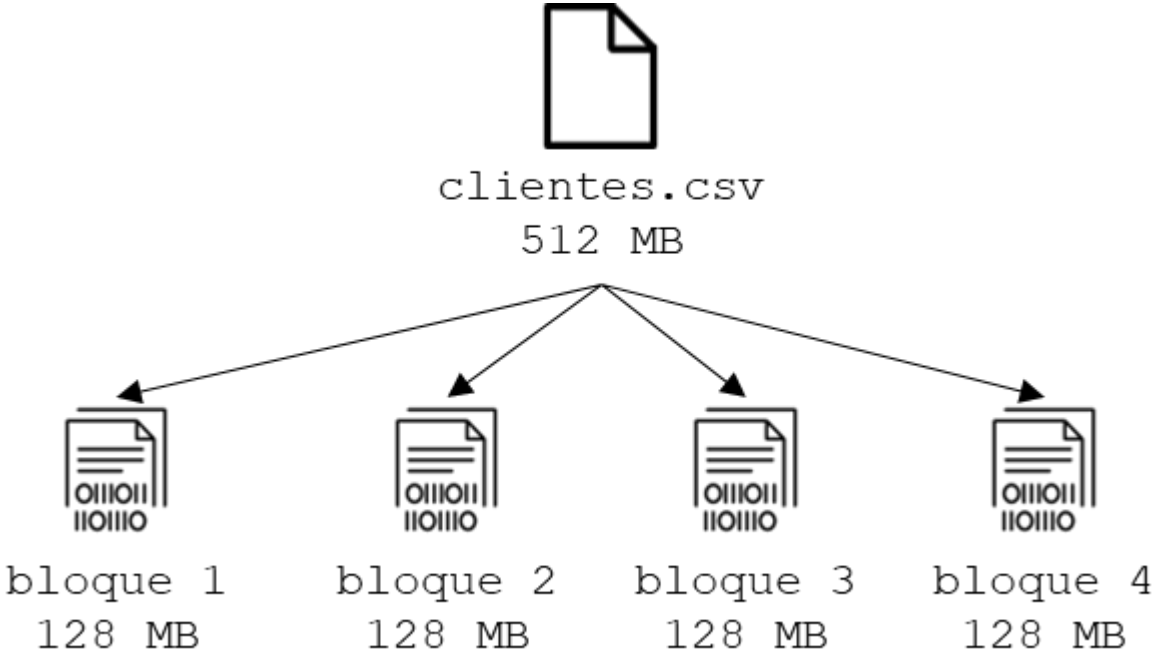
HDFS

No establece **ninguna restricción sobre los datos**, ya que éstos pueden ser estructurados, semiestructurados o no disponer de ninguna estructura, como el caso de imágenes o vídeos.

Para conseguir tener tolerancia a fallos, los ficheros se dividen en bloques, de un tamaño por defecto de **128 Mb**, que son **almacenados y replicados** en distintos nodos.

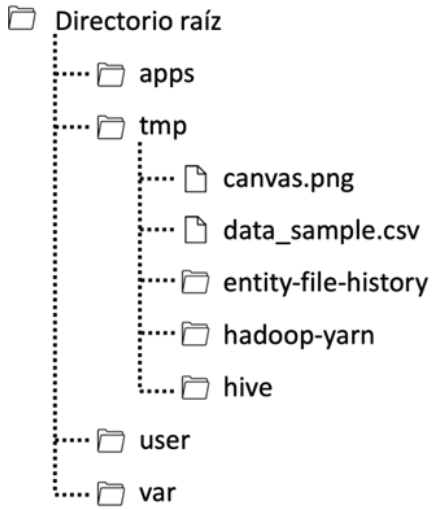
Tener un tamaño de bloque tan grande permite un throughput de lectura/escritura elevado, pero se penaliza las operaciones aleatorias.

HDFS


clientes.csv
512 MB
bloque 1
128 MB
bloque 2
128 MB
bloque 3
128 MB
bloque 4
128 MB

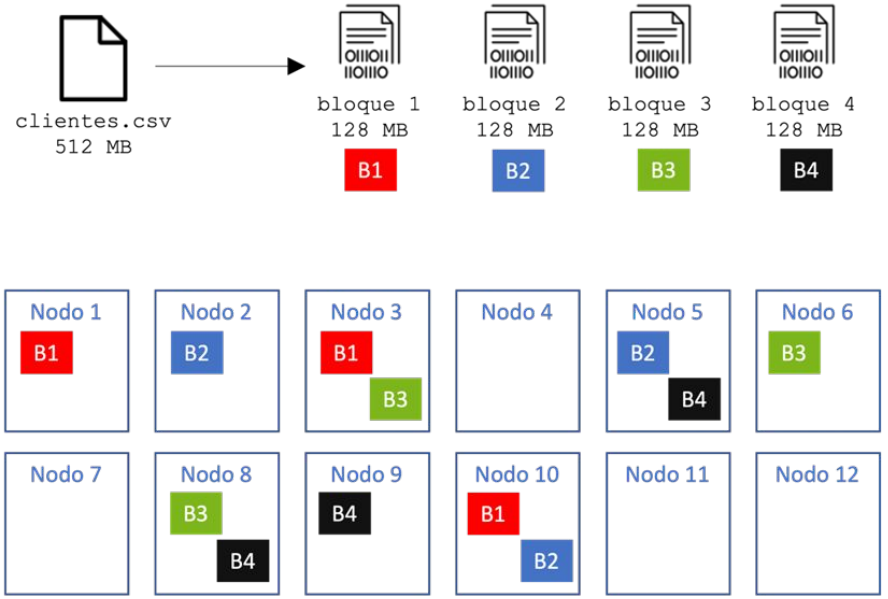
HDFS

Utiliza **replicación** de los datos entre distintos nodos para sobreponerse a errores y proporciona a los clientes un **interfaz similar a los sistemas de ficheros tradicionales**, con organización de los ficheros en **espacios de nombres** (directorios), y funcionalidades para lectura, búsqueda, listado de ficheros, etc. sin necesidad de conocer la ubicación de los datos o la organización del clúster.



HDFS

La replicación se hace de forma **automática**, no siendo necesario que los usuarios o clientes de HDFS conozcan dónde están ubicados los diferentes bloques de un fichero.



HDFS

La arquitectura de HDFS consta de distintos servicios y tipos de nodo, aunque fundamentalmente son:

- **Namenode**: actúa de maestro, manteniendo la metainformación de todo el sistema de ficheros, esto es, el espacio de nombres, la ubicación de los bloques y su replicación. Dispone de un Secondary Namenode para HA.
- **Datanode**: actúan de worker / esclavo, contienen físicamente los bloques pero no gestionan su metainformación, y controlan la validez de los bloques (checksums).

HDFS

La replicación se realiza de forma automática coordinada por el Namenode. Por defecto los bloques se replican en 3 datanodes, aunque puede modificar el número de réplicas para ficheros / directorios de forma individual.

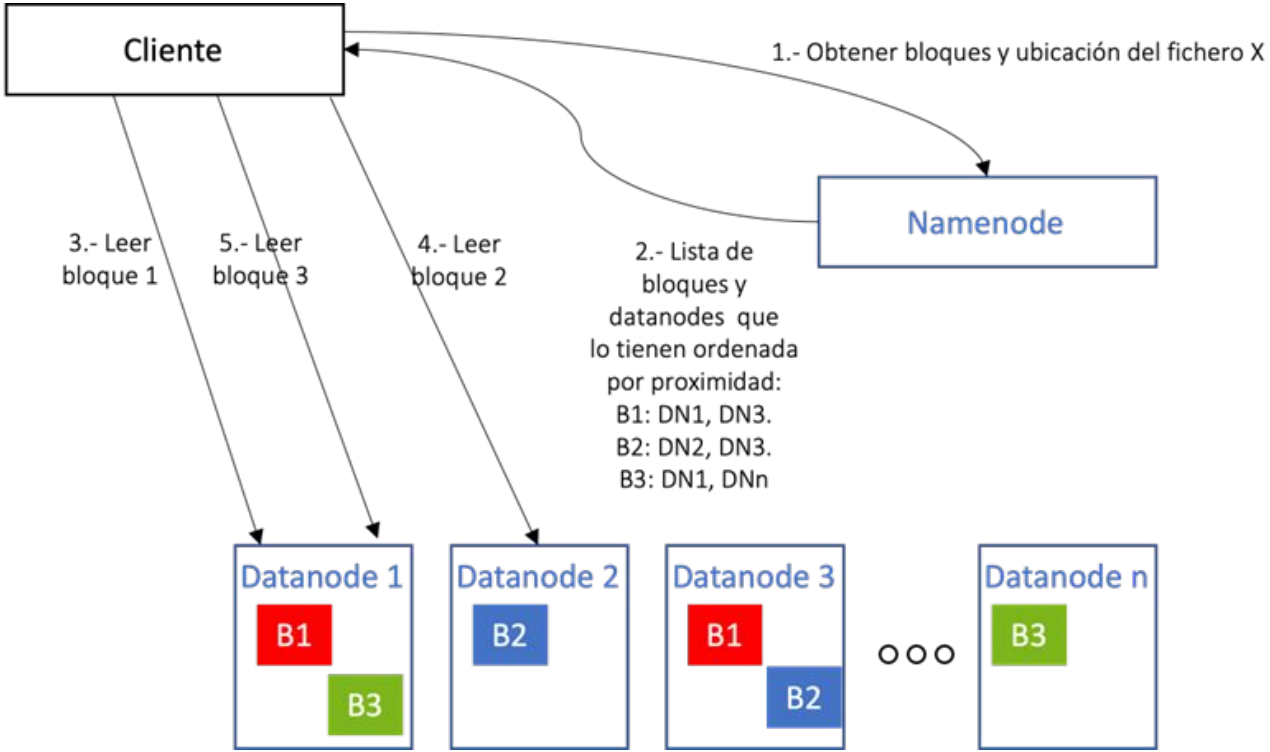
HDFS

HDFS soporta operaciones similares a los sistemas Unix:

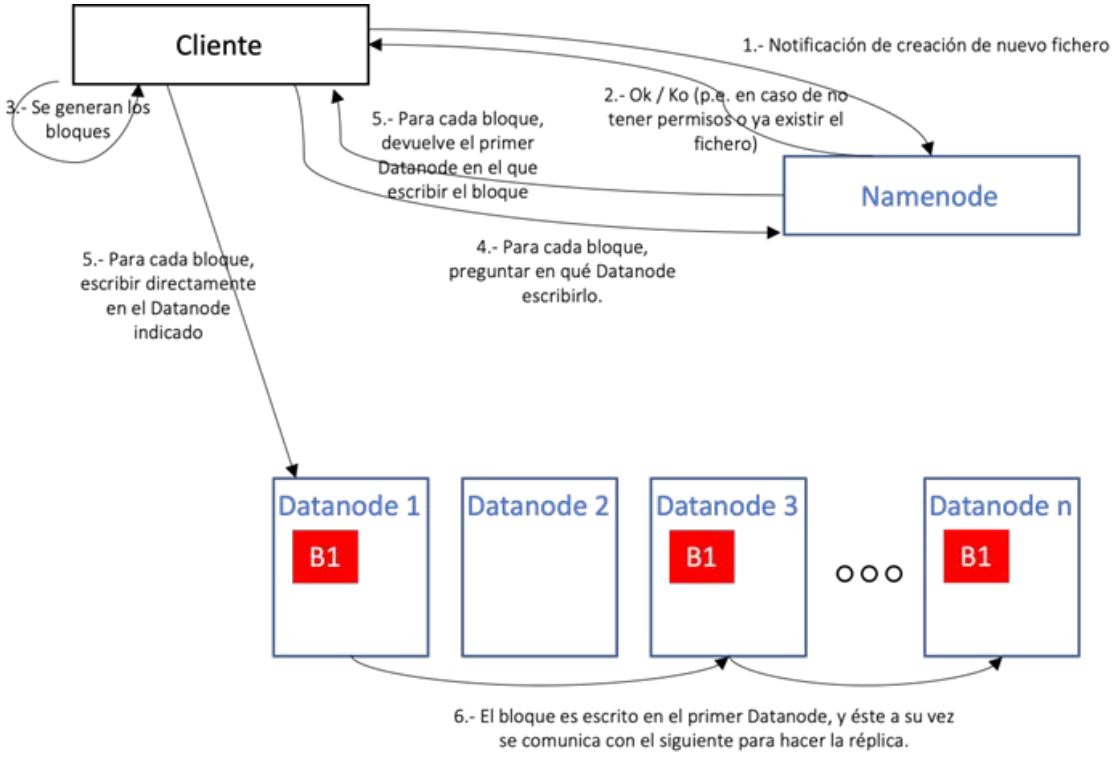
- Lectura, escritura o borrado de ficheros.
- Creación, listado o borrado de directorios.
- Usuarios, grupos y permisos.

HDFS es un sistema de ficheros append-only, no se permite modificar bloques. Esta filosofía append- only condiciona mucho el funcionamiento de otras aplicaciones que utilizan HDFS como almacenamiento (Hive, HBase, etc.).

Lectura en HDFS



Escritura en HDFS



HDFS

Interfaces de acceso:

- Java API.
- RestFul API (WebHDFS).
- NFS interface (HDFS NFS Gateway).
- Librería C.
- Cliente de línea de comandos. Algunos comandos: mkdir, ls, put, get, cat, cp, rm, mv, setrep.

YARN

YARN

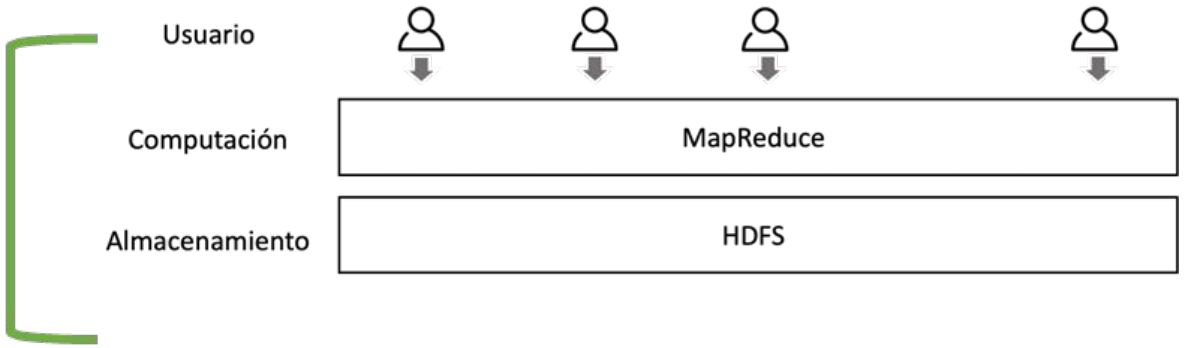
YARN (Yet Another Resource Negotiator) es un **gestor de recursos de computación y memoria** que no estaba presente en las primeras versiones de Hadoop, que usaban únicamente MapReduce como paradigma de computación.

Habilita el procesamiento de los datos almacenados en HDFS de forma concurrente y utilizando tecnologías diferentes de MapReduce (por ejemplo, Storm para event processing).

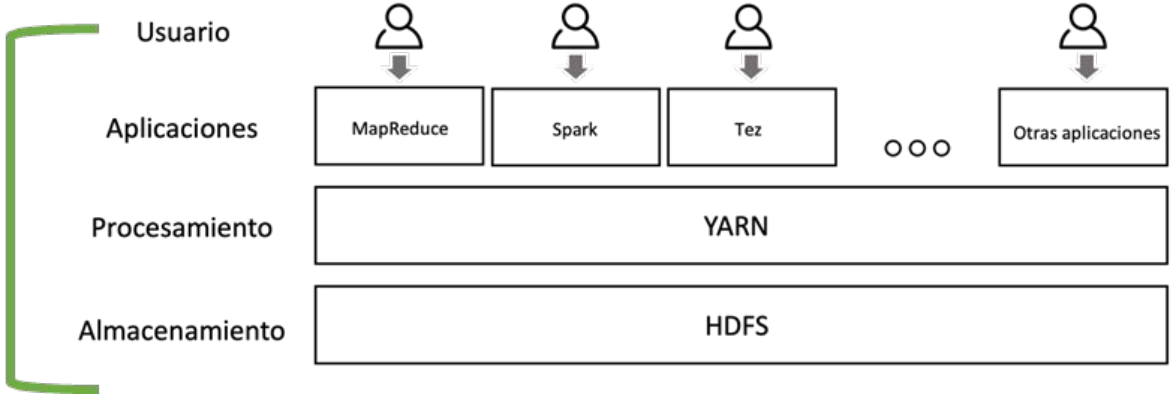
La mayoría de aplicaciones y frameworks de Hadoop se ejecutan sobre YARN en lo que se refiere a ejecución distribuida (Spark, Storm, Tez, etc.).

YARN

Hadoop sin YARN



Hadoop con YARN



YARN

Principales responsabilidades de YARN:

- **Planificar/organizar** las **tareas** y **gestionar** los **recursos** de CPU y memoria.
- Habilitar el uso del cluster para entornos **multi-tenant**, compartidos y seguros.

Las aplicaciones necesitan recursos de CPU y memoria para ejecutarse, que son llamados Containers.

YARN se ocupa de la asignación de contenedores a las aplicaciones, estableciendo prioridades, turnos, etc.

YARN

La arquitectura de YARN, de tipo maestro/esclavo, consta de dos tipos de nodos:

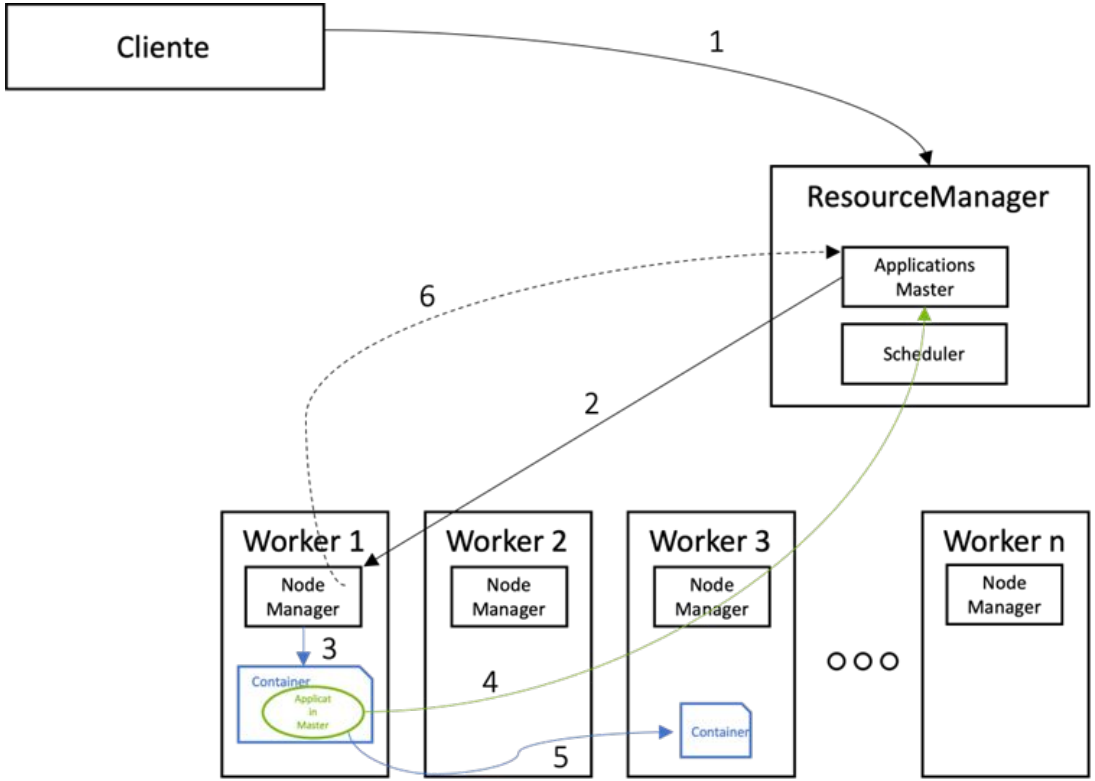
ResourceManager (master): maestro de YARN, organiza las tareas y gestiona los recursos disponibles. Dispone de un Scheduler (pluggable) para la organización de tareas. Los Schedulers permiten establecer prioridades, reservar recursos, etc. para las aplicaciones. Los principales son:

- Capacity Scheduler.
- Fair Scheduler.
- FIFO Scheduler.

YARN

NodeManager (worker): lanza y maneja las aplicaciones en los contenedores de que dispone, monitoriza los recursos (CPU, red y memoria) e informa al ResourceManager periódicamente.

YARN


Cliente
1
ResourceManager
Applications Master
Scheduler
6
2
Worker 1
Node Manager
3
Container
Applicat in Master
Worker 2
Node Manager
4
5
Worker 3
Node Manager
Container
Worker n
Node Manager

YARN

Algunas características de YARN:

- **Multitenancy**: permite acceso simultáneo de distintos motores / aplicaciones sobre los mismos datos.
- **Escalable**: permite incrementar la capacidad de procesamiento añadiendo nuevos nodos worker.
- **Robusto**: ofrece configuraciones de alta disponibilidad para el Resource Manager, que es el único punto único de fallo.
- **Compatibilidad**: es compatible con las aplicaciones que se ejecutan sobre MapReduce.
- **Estándar**: unifica el framework y ciclo de vida de ejecución de aplicaciones sobre Hadoop.

MapReduce

MapReduce

MapReduce es un **paradigma de computación** creado por Google para resolver la creación de índices de búsqueda de páginas.

Fue el motor de procesamiento de Hadoop en sus **primeras versiones.**

Se basa en la capacidad de **dividir** el procesamiento sobre los datos en múltiples nodos y en paralelo, resolviendo la limitación de la computación de grandes volúmenes de datos sobre una única máquina y abstrayendo al desarrollador de las tareas de orquestación de los sistemas de computación distribuida tradicionales.

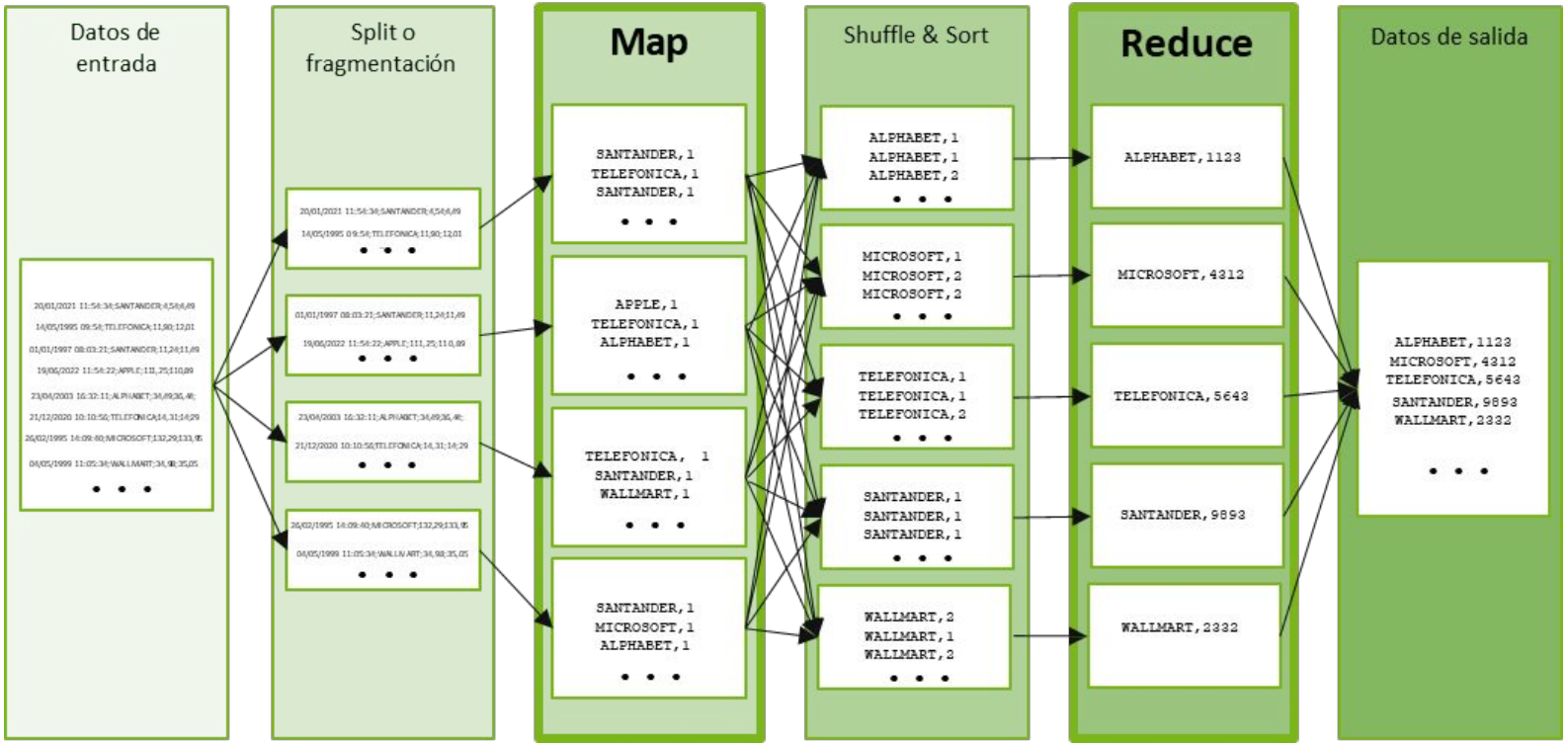
MapReduce

Frente a los sistemas de computación tradicionales, MapReduce ejecuta la lógica **en los nodos donde residen** los datos en lugar de moverlos por la red, lo que mejora el rendimiento en órdenes de magnitud.

MapReduce está orientado a procesamiento offline y es tolerante a fallos, ya que un nodo del clúster puede fallar o caerse sin que el proceso de computación se aborte.

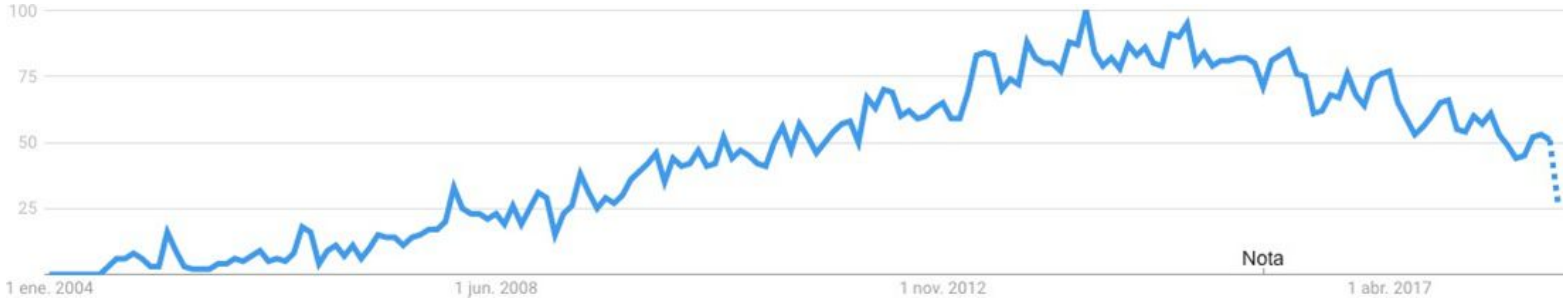
Permite distintos lenguajes de programación: Java, Python, C++, Perl, etc.

MapReduce


Datos de entrada
Split o fragmentación
Map
Shuffle & Sort
Reduce
Datos de salida
SANTANDER,1
TELEFONICA,1
SANTANDER,1
APPLE,1
TELEFONICA,1
ALPHABET,1
TELEFONICA, 1
SANTANDER,1
WALLMART,1
SANTANDER,1
MICROSOFT,1
ALPHABET,1
ALPHABET,1
ALPHABET,2
MICROSOFT,1
MICROSOFT,2
MICROSOFT,2
TELEFONICA,1
TELEFONICA,1
TELEFONICA,2
SANTANDER,1
SANTANDER,1
SANTANDER,1
WALLMART,2
WALLMART,1
WALLMART,2
ALPHABET,1123
MICROSOFT,4312
TELEFONICA,5643
SANTANDER,9893
WALLMART,2332
ALPHABET,1123
MICROSOFT,4312
TELEFONICA,5643
SANTANDER,9893
WALLMART,2332

MapReduce

Cada vez se utiliza menos debido a que nuevos frameworks de procesamiento, como Tez o Spark, mejoran su rendimiento en órdenes de magnitud.



FIN