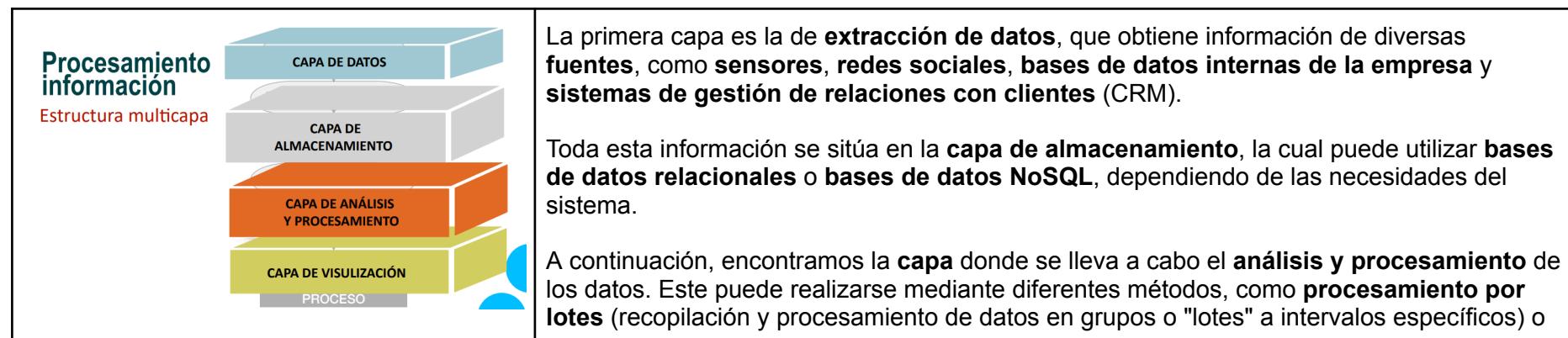
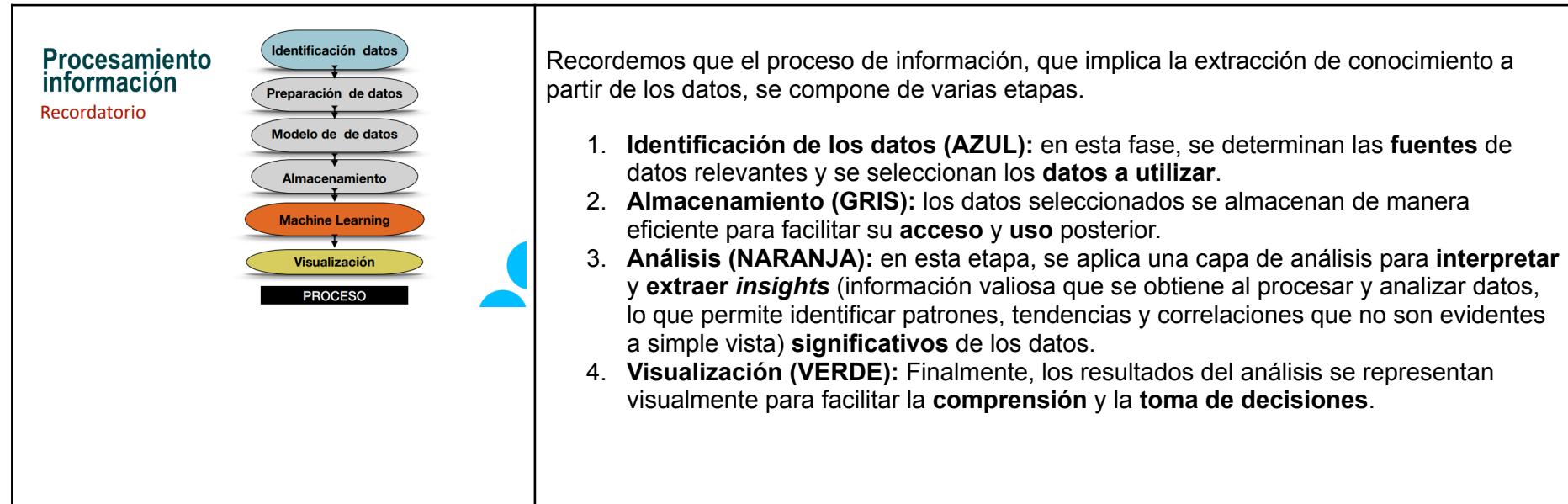
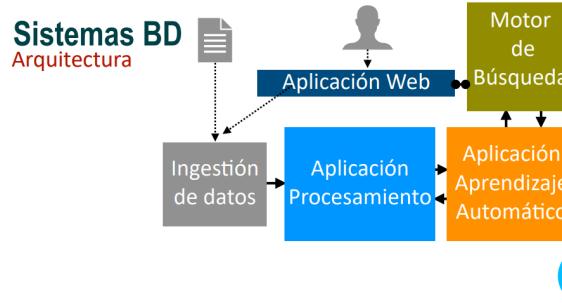


1 - ARQUITECTURA DE LOS SBD

El objetivo es ver una arquitectura para Big Data prototípica y algunos problemas que se deben tener en cuenta cuando se implementa en la nube.



	<p>en tiempo real (streaming: análisis continuo de datos a medida que se generan).</p> <p>Finalmente, en la capa de visualización, se presentan las predicciones, alertas, recomendaciones y explicaciones sobre los fenómenos observados, facilitando así la interpretación y la toma de decisiones.</p>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

 <p>Sistemas BD Arquitectura</p>	<p>Una estructura prototípica de un sistema Big Data se caracteriza por su eficiencia y modularidad, permitiendo gestionar grandes volúmenes de datos. Estas arquitecturas suelen implementarse en servicios en la nube o clústeres, facilitando la escalabilidad y la disponibilidad.</p> <p>Para acceder a estos servicios, se emplea una aplicación web que permite a los usuarios interactuar con el sistema, visualizar resultados y realizar consultas.</p> <p>Los pasos que siguen estos sistemas son:</p> <ol style="list-style-type: none"> 1. Ingestión de datos: es el punto de entrada de la información, donde se recopilan y almacenan los datos. 2. Aplicación de procesamiento: toma los datos ingresados y realiza transformaciones o análisis básicos necesarios antes de pasar la información a otros componentes. 3. Aplicación de aprendizaje automático: usa los datos procesados para entrenar modelos y extraer patrones o información valiosa. 4. Motor de búsqueda: permite realizar consultas sobre los datos procesados, proporcionando respuestas en tiempo real para facilitar el análisis. 5. Aplicación web: es la interfaz de usuario que permite la interacción con el sistema, proporcionando acceso y visualización de los datos y resultados de consultas y análisis.
------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Sistemas BD

Clasificación

- Procesamiento **batch**
- Procesamiento **streaming**
- Procesamiento en **tiempo real**

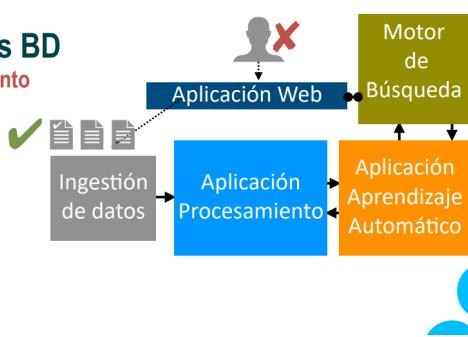
Existen 3 tipos de procesamiento en función de la forma en que se gestionan los datos basadas en la urgencia con la que los datos deben procesarse:

1. **Procesamiento por lotes (Batch Processing):**
2. **Procesamiento en streaming.**
3. **Procesamiento en tiempo real**

A continuación, veremos cómo funcionan estos tipos de procesamiento dentro de la arquitectura del sistema.

Sistemas BD

Procesamiento batch



El **procesamiento por lotes** o **batch** permite manejar grandes volúmenes de datos de manera eficiente. Se utiliza cuando es posible **acumular grandes volúmenes de datos y analizarlos de una sola vez**, sin necesidad de obtener resultados inmediatos. Los trabajos por lotes suelen ser **programados** para ejecutarse en **momentos específicos**, como al final del día.

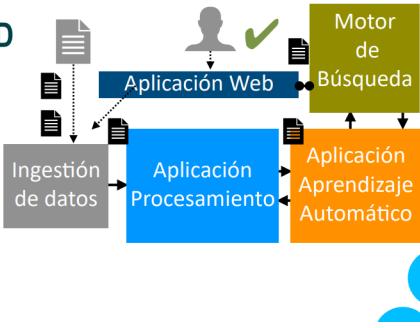
En este método, todos los datos deben ser recopilados y almacenados en la etapa de **ingestión de datos** antes de iniciar el análisis. El procesamiento se realiza **sin la intervención activa del usuario**: una vez almacenada toda la información en la capa de ingestión, **el usuario simplemente lanza la carga de trabajo, y el sistema se encarga de ejecutar el procesamiento de forma autónoma**.

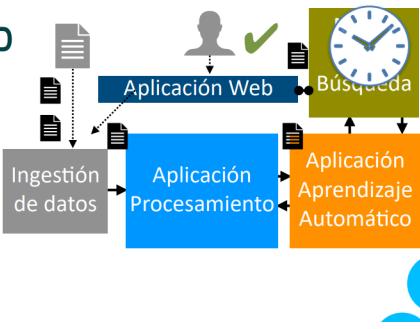
La **capa de procesamiento** toma todos los datos acumulados y los analiza en una **sola ejecución**, lo cual es ideal para tareas que no necesitan resultados inmediatos.

Al finalizar el análisis por lotes, los resultados se pueden enviar a la **aplicación web** para su visualización.

El **motor de búsqueda** interviene después de que el procesamiento por lotes ha finalizado y los datos han sido analizados. Su función principal en este contexto es **permitir que el usuario consulte y recupere información específica de los resultados ya procesados**,

	facilitando la exploración y el análisis posterior.
--	-------------------------------------------------------------------

Sistemas BD Procesamiento streaming	 <p>El procesamiento en streaming es un mix de los otros dos, pero más parecido al tiempo real. Se realiza en tiempo cercano al real, pero en este caso los datos se procesan en pequeños lotes de manera continua y secuencial a medida que llegan. Los datos se recopilan de forma continua y se envían a la capa de procesamiento en micro-lotes, lo que permite una gestión continua del flujo de información. La capa de procesamiento recibe estos datos y realiza operaciones en cada lote. Los datos procesados se envían a la aplicación de aprendizaje automático para generar análisis y predicciones.</p>
-----------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Sistemas BD Procesamiento en tiempo real	 <p>Este método es ideal para aplicaciones que requieren respuestas instantáneas o casi instantáneas a medida que los datos llegan al sistema. La información se recibe en tiempo real, permitiendo su análisis inmediato. A medida que los datos se ingieren, se envían a la aplicación de procesamiento, que los analiza al instante. No es necesario esperar a que se acumulen grandes volúmenes de datos, como en el procesamiento por lotes. Los resultados procesados pueden enviarse rápidamente a una capa de aprendizaje automático, que puede tomar decisiones o hacer predicciones en tiempo real, basándose en los datos más recientes. A través de la aplicación web los usuarios pueden acceder a los datos y visualizar los resultados de procesamiento en tiempo real, utilizando el motor de búsqueda si es necesario, que facilita la recuperación de información específica casi al instante después de que se ha generado.</p>
----------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Servicios en la nube

Problemáticas

- Hackeo
- Protección de datos
- Cuellos de botella en el procesamiento
- Escalabilidad



Todos estos sistemas suelen almacenarse en **clústeres**. Con frecuencia, se prestan como servicios en la **nube**, lo cual introduce una serie de desafíos que deben ser considerados durante el diseño del sistema. Entre estos problemas están:

1. **Problemas de seguridad y hackeo:** este problema se refiere a la posibilidad de que **usuarios no autorizados** accedan, copien o visualicen **información confidencial**. Esto representa un **gran riesgo** a nivel **empresarial y gubernamental**, y, por ello, **se invierten grandes sumas de dinero** en **medidas de seguridad** para **proteger** los datos y **evitar brechas de seguridad**.
2. **Protección de datos:** ocurre porque la **distribución** de datos en múltiples servidores o colecciones de ordenadores plantea riesgos de protección si los **protocolos de seguridad son deficientes** o si el sistema se utiliza **incorrectamente**. Esto puede resultar en la exposición de **información sensible**, lo que no solo representa un riesgo de **seguridad**, sino que también puede dañar la **reputación** de la empresa.
3. **Cuellos de botella en el procesamiento:** ocurren cuando un componente del sistema **no puede manejar el volumen** de solicitudes o tareas que recibe, lo cual **ralentiza el rendimiento** general del sistema. Esto puede suceder, por ejemplo, si muchas tareas se están procesando simultáneamente en un mismo servidor. Para mitigar este problema, los administradores de la infraestructura pueden **dimensionar adecuadamente los recursos**, replicar hardware y **balancear la carga** entre diferentes servidores.
4. **Escalabilidad:** surge cuando **aumenta la demanda de recursos** y el sistema debe **adaptarse** para manejarla **sin pérdida de rendimiento**. La capacidad de expansión dependerá de la **complejidad de los algoritmos** y la **infraestructura**. Esto puede implicar la necesidad de **añadir más recursos** en la nube, como almacenamiento y potencia de procesamiento, para asegurar que el sistema **crezca de manera eficiente**.
5. **Concurrencia:** se da cuando **múltiples** tareas se ejecutan **simultáneamente** y **comparten los mismos recursos**. La concurrencia debe manejarse adecuadamente para **evitar tiempos muertos** o **conflictos entre tareas**. Esto requiere **estrategias de sincronización y coordinación** eficientes, como **bloqueos** o **algoritmos de control de acceso**, para asegurar que los recursos compartidos se usen de manera óptima.

Servicios en la nube

Problemáticas

- Conurrencia
- Tolerancia a fallos
- Disponibilidad

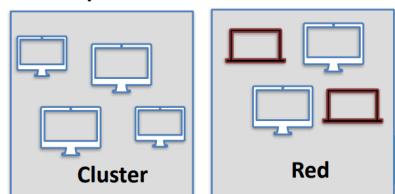


6. **Tolerancia a fallos:** en un sistema distribuido, múltiples nodos trabajan juntos para procesar las tareas. **Si uno de estos nodos falla**, el sistema debe ser capaz de **continuar operando sin pérdida de datos**. Esto implica implementar mecanismos de **redundancia, recuperación automática y liberación adecuada de recursos** para asegurar que **el fallo de un nodo no afecte el procesamiento general**.
7. **Disponibilidad:** los usuarios esperan que los **servicios** estén **accesibles** en todo momento. La disponibilidad implica **garantizar** que los **servidores no se caigan** o que ciertos servicios **no queden inoperativos**. Para lograr una **alta disponibilidad**, es común **replicar servicios críticos** en **varios nodos del clúster**, de modo que si uno falla, otros puedan asumir su carga sin interrumpir el servicio.

2 - SISTEMAS BD - PROCESAMIENTO BATCH

Procesamiento batch

- Recolección y almacenamiento datos **antes** de procesamiento
- Computación distribuida:
MapReduce



El objetivo de este tema es conocer mejor el modelo de computación **MapReduce**, una herramienta fundamental para realizar **procesamiento batch** en **sistemas distribuidos**.

El procesamiento batch se caracteriza por la **recolección y almacenamiento previo** de todos los datos **antes de iniciar el cálculo**. MapReduce permite llevar a cabo **cálculos distribuidos** de manera eficiente, ya sea en **redes de ordenadores heterogéneas** (con **distintos tipos de hardware y software**) o, más comúnmente, en **clústeres** de ordenadores con características de **hardware y software uniformes**.

MapReduce Google 2004

- Máquinas procesamiento MapReduce
- Google File System (GFS)
- BigTable

El proyecto MapReduce se originó en **Google** en **2004** como respuesta a la necesidad de **indexar la web al ritmo en que ésta crecía**. Google desarrolló una solución que combinaba **tres componentes fundamentales**:

- el **modelo de computación MapReduce**
- un **sistema de archivos distribuido** conocido como **Google File System (GFS)**
- una **base de datos no relacional** llamada **BigTable**

MapReduce Apache Software Foundation

- Hadoop MapReduce
- Hadoop YARN
- Hadoop Distributed File System (HDFS) y Hbase

Posteriormente, la **Fundación Apache (Apache Software Foundation)** adoptó este enfoque para crear una implementación de **código abierto** de MapReduce. Esto dio lugar al desarrollo de **Hadoop MapReduce**, que más tarde fue optimizado con **Hadoop YARN**. Es decir, Hadoop YARN es el gestor de recursos y planificación de tareas en Hadoop. Además, se introdujo el **sistema de archivos distribuido**, llamado **Hadoop Distributed File System (HDFS)** como alternativa al **Google File System**, y **HBase** como sustituto de **BigTable** para **gestionar datos** en grandes volúmenes. Por lo tanto, podemos resumir que **Apache Hadoop** está compuesto por los componentes principales:

- **HDFS (Hadoop Distributed File System)**: sistema de archivos distribuido que almacena datos en bloques distribuidos a través de múltiples nodos del clúster.
- **MapReduce**: modelo de programación para procesar grandes cantidades de datos en paralelo.
- **YARN**: gestor de recursos y planificación de tareas en Hadoop.

MapReduce Apache Software Foundation

- Entorno programación
- Escalabilidad
- Tolerante a fallos



La Fundación Apache lanzó así **dos proyectos clave** para el **procesamiento batch o en lotes**: **Apache Hadoop** y **Apache Spark**. **Apache Spark** es una herramienta avanzada para el procesamiento rápido, y se usa tanto para procesamiento batch como para procesamiento **en tiempo real**. Consigue un procesamiento más rápido, ya que almacena los datos en memoria en lugar de en disco, como hace Hadoop.

Ambos entornos de programación comparten tres características esenciales:

1. **Entorno de programación estructurado**: en el que se proveen **herramientas robustas** para programar tareas distribuidas.
2. **Escalabilidad**: permiten **ampliar fácilmente** el número de **nodos** de forma **transparente** para el usuario.
3. **Tolerancia a fallos**: en caso de fallo de un nodo, el sistema **redirige** las tareas a **otros nodos del clúster**, garantizando la **continuidad** del procesamiento sin pérdida de datos.

Estos dos proyectos han revolucionado el procesamiento de grandes volúmenes de datos, facilitando el desarrollo de aplicaciones de computación distribuida.

MapReduce Datos

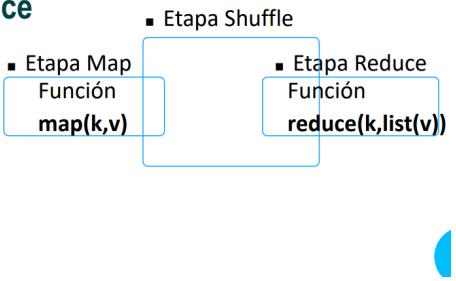
- Registros < clave, valor >
- $< k, v >$



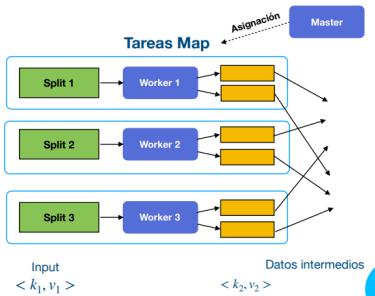
El paradigma de **MapReduce** se basa en una estructura de datos organizada en pares clave-valor.

La **clave** es esencial para la **organización y gestión de los datos**, mientras que el valor puede ser de cualquier **tipo**: desde un vector hasta una matriz u otros tipos de datos.

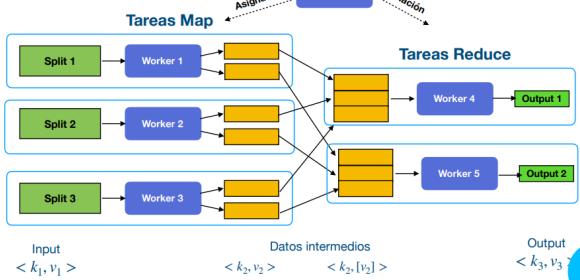
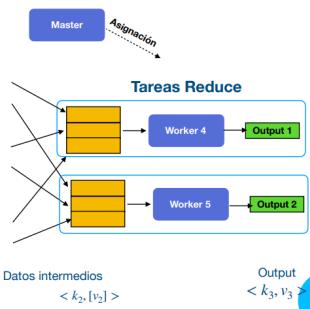
Esta estructura es fundamental, ya que define cómo se representan y manipulan los registros en el **procesamiento distribuido**.

 <p>MapReduce Etapas</p> <ul style="list-style-type: none"> ■ Etapa Map Función <code>map(k,v)</code> ■ Etapa Shuffle ■ Etapa Reduce Función <code>reduce(k,list(v))</code> 	<p>MapReduce es un modelo de procesamiento que consta de 3 etapas: Map, Shuffle y Reduce.</p> <ul style="list-style-type: none"> ● Etapa Map: cada nodo del clúster accede a los datos locales de su propia máquina y realiza un procesamiento inicial. Este procesamiento produce datos intermedios en formato clave-valor, que deben ser reorganizados y reordenados para la siguiente etapa. ● Etapa Shuffle: en esta fase, se recopilan todos los datos intermedios, se almacenan temporalmente en archivos y se trasladan a nuevos nodos donde se ejecutará la fase final, Reduce. ● Etapa Reduce: esta etapa se ejecuta en otros nodos del clúster y procesa los datos intermedios. Aquí, la entrada consiste en pares clave y listas de valores asociados, y se realiza el procesamiento final para producir el resultado. <p>El usuario debe definir 2 funciones:</p> <ol style="list-style-type: none"> 1. Función Map: toma pares clave-valor como entrada y genera los datos intermedios. 2. Función Reduce: utiliza como parámetros de entrada una clave y una lista de valores asociados. Esta función procesa y consolida los datos en la etapa final. <p>No tiene que preocuparse de las funciones que ocurren en la etapa Shuffle, ya que son totalmente transparentes para él.</p>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

MapReduce Etapa Map



MapReduce Etapa Reduce



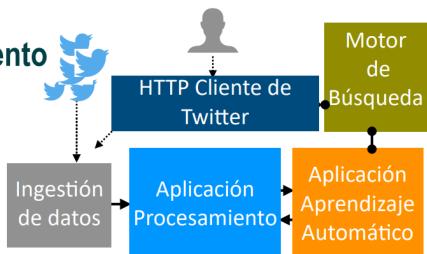
En esta primera imagen del proceso, la "Etapa Map", se muestra cómo el nodo maestro (**Master**) asigna tareas a los nodos trabajadores (**Workers**) para procesar los datos en **paralelo**. Aquí, cada conjunto de **datos iniciales** se divide en partes más **pequeñas** denominadas **splits** (Split 1, Split 2, y Split 3), y cada uno es asignado a un **Worker** (nodo). Cada nodo trabajador ejecuta una función **Map** con su split correspondiente. La salida de esta etapa es un conjunto de **pares clave-valor intermedios** que luego se utilizan en la siguiente etapa.

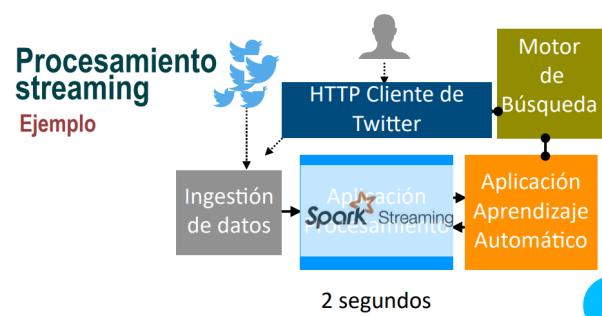
Durante la etapa intermedia **Shuffle**, los pares clave-valor intermedios se **ordenan y agrupan** por **claves comunes**, creando pares de **clave-lista de valores**. Esta agrupación permite que los datos necesarios para cada clave estén listos para ser procesados juntos en la etapa **Reduce**. Todo este proceso es gestionado de manera transparente para el usuario..

En la "Etapa Reduce", observamos cómo los datos agrupados de la etapa **Shuffle** son asignados a los nodos trabajadores para su procesamiento final. Aquí, los **Workers** 4 y 5 reciben los datos **intermedios** (pares clave-listas de valores) y aplican la función **Reduce** definida por el usuario, generando un **output final** para cada clave. Este resultado final es el objetivo del procesamiento de MapReduce y representa la salida total del trabajo.

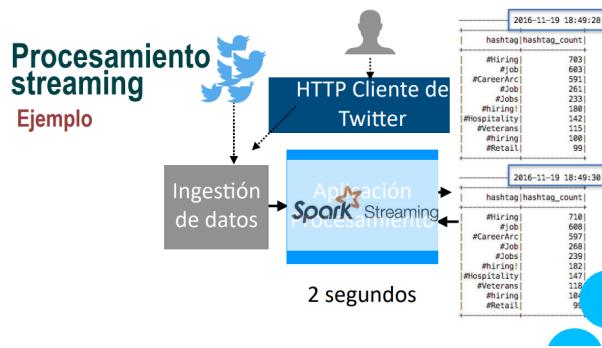
3 - PROCESAMIENTO STREAMING Y TIEMPO REAL

<p>Procesamiento streaming</p> <p>Características</p> <ul style="list-style-type: none">▪ Flujos de datos▪ No hay limitación en tiempo de procesamiento▪ No hay limitación en tiempo generación output▪ Suficiente memoria▪ Tasa de procesamiento > Tasa de entrada	<p>El objetivo de este tema es presentar las características fundamentales del procesamiento en streaming que se utiliza en sistemas de Big Data.</p> <p>La primera característica clave es que el procesamiento en streaming implica un flujo constante de datos, los cuales se analizan conforme se generan. En este tipo de procesamiento, no existen limitaciones de tiempo estrictas para el análisis ni para la obtención de resultados (<i>output</i>). Sin embargo, es fundamental contar con suficiente memoria para almacenar todos los datos generados, evitando así posibles pérdidas de información.</p> <p>Además, aunque en ciertos momentos los datos puedan generarse a una velocidad mayor que la de procesamiento, <u>en promedio</u>, la tasa de procesamiento debe ser superior a la tasa de entrada. Esto es esencial para evitar la acumulación infinita de datos y garantizar un flujo continuo en el análisis (que no se queden datos sin procesar).</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p>Procesamiento streaming</p> <p>Ejemplo</p> 	<p>Veamos un ejemplo de procesamiento en streaming en Big Data. Imagina que queremos desarrollar una aplicación cuyo objetivo es identificar en tiempo real los hashtags más relevantes en Twitter.</p> <p>En esta arquitectura de Big Data, Twitter actúa como la fuente de datos. Para acceder a los tweets, es necesario registrarse en la API de Twitter, lo que nos permite autenticar un usuario y obtener acceso en tiempo real a los tweets que se generan. Esta parte se denomina "ingestión de datos". Informáticamente, tendremos que implementar un cliente HTTP, por ejemplo, en Python, para recibir los tweets en formato JSON, que contendrán el conjunto de datos necesarios.</p> <p>Dado que estamos trabajando con procesamiento en streaming, utilizaremos Apache Spark como motor de procesamiento de la aplicación. Apache Spark es una de las</p>
------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



plataformas más utilizadas para este tipo de tareas y permitirá **leer el flujo de tweets, descomponerlos en palabras, filtrar hashtags y contar sus ocurrencias**. Este procesamiento **se configura para que ocurra en intervalos de 2 segundos**, lo que significa que **cada 2 segundos se analizará el conjunto de tweets más reciente**.



El resultado será una **matriz de hashtags** con sus respectivos **conteos, actualizada cada 2 segundos**. Estos **resultados se mostrarán** a través de una **API**, tal como se observa en la imagen. **La configuración de la aplicación para realizar lecturas periódicas cada 2 segundos es una característica fundamental del procesamiento en streaming, que permite tener una visión en tiempo real de las tendencias en Twitter**.

Procesamiento en tiempo real

Características

- Flujos de datos
- Suficiente memoria
- Tasa de procesamiento > Tasa de entrada

El procesamiento en **tiempo real** comparte ciertas características con el procesamiento en **streaming**: **ambos se basan en un flujo continuo de datos, requieren suficiente memoria y una tasa de procesamiento superior a la tasa de entrada para evitar acumulaciones**. Sin embargo, el procesamiento en tiempo real tiene características distintivas importantes.

Procesamiento en tiempo real

Características diferentes streaming

- Hay limitación en **tiempo** de procesamiento o de obtención **output**
- No se pueden perder datos

La primera diferencia clave es que el **procesamiento en tiempo real** tiene una **limitación de tiempo para generar los resultados**. Por ejemplo, en un sistema de control de tráfico ferroviario, es crucial que las respuestas se generen antes de que puedan producirse conflictos en el sistema ferroviario.

La segunda diferencia es que **no se permite la pérdida de datos**, ya que, en aplicaciones críticas como sistemas de vigilancia, perder fragmentos de video podría significar no detectar un evento importante, como un allanamiento.

Procesamiento Batch/ streaming



Motivación

- ¿Cuanto **tiempo** necesita un cluster MapReduce con 20 nodos para leer 1 TB?
 - Lectura disco duro 100-200 MB/s
 - 1 TB = 10^6 MB
 - MB de disco por nodo = $10^6 / 20$

$$\text{Tiempo} = \frac{10^6}{20 \cdot 200} = 250 \text{ s}$$

En este punto, vamos a enfatizar la **diferencia clave entre el procesamiento por lotes (batch) y el procesamiento en streaming**.

- Una de las diferencias esenciales es cómo cada uno **maneja los datos**.
 - El procesamiento por **lotes** utiliza intensivamente el **disco duro**, como se ve en el modelo **MapReduce**, donde los **resultados intermedios se almacenan en disco**. Este proceso, aunque efectivo para grandes volúmenes de datos, **tiende a ralentizarse** debido al **acceso frecuente al disco**.
 - Por otro lado, el **procesamiento en streaming** realiza la mayor parte de sus cálculos en **memoria RAM**, lo que permite un **análisis casi instantáneo de los datos**, especialmente en escenarios donde la velocidad es crítica. Esta diferencia en el manejo de los datos **afecta** decisivamente el **rendimiento y la eficiencia** de cada método.

Para ilustrar esto, consideremos el siguiente ejemplo. Imaginemos que queremos usar MapReduce con 20 nodos para leer 1 terabyte (TB) de información. Actualmente, los discos duros ofrecen una velocidad de lectura de entre **100 y 200 MB/s**. Un terabyte equivale a 10^6 MB. Por lo tanto, si dividimos este volumen de datos entre los 20 nodos, cada nodo debería leer **$10^6/20 = 50 \text{ MB}$** .

	<p>Por lo tanto, si utilizamos una velocidad de lectura de 200 MB/s, cada nodo tardaría:</p> <p>Tiempo total = 50 MB / 200 MB/s = (es decir) = $10^6 / (20 \times 200) = 250$ segundos = 4,16 minutos</p> <p>Este cálculo muestra cómo el tiempo necesario para procesar grandes volúmenes de datos depende de factores como la cantidad de nodos y la velocidad de lectura en disco.</p>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p>Procesamiento en tiempo real</p> <p> Motivación</p>	<ul style="list-style-type: none"> ■ ¿Cuanto tiempo necesita un esquema MapReduce con 20 nodos para leer 1 TB en memoria RAM? ■ Lectura memoria 10 GB/s= 10^4 MB/s ■ 1 TB = 10^6 MB ■ MB de disco por nodo = $10^6 / 20$ $\text{Tiempo} = \frac{10^6}{20 \cdot 10^4} = 5 \text{ s}$	<p>Si realizamos el mismo proceso utilizando memoria RAM en lugar de disco duro, la velocidad de lectura aumenta significativamente, alcanzando aproximadamente 10 GB/s, es decir, 10.000 MB/segundo.</p> <p>Comparado con los 200 MB/s del disco duro, esta mejora es notable.</p> <p>Con esta velocidad, ahora el clúster de 20 nodos puede leer 1 TB (equivalente a 1 millón de MB) en tan solo 5 segundos, frente a los más de 4 minutos de antes.</p> <p>Este incremento en la velocidad permite que los sistemas operen en tiempo real, mejorando el rendimiento en aplicaciones que requieren respuestas rápidas.</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p>Procesamiento en tiempo real</p> <p>Computación en memoria</p>	<ul style="list-style-type: none"> ■ MapReduce ■ Consultas en tiempo real <p>Precio memoria RAM 1TB= 20000 \$</p>	<p>El procesamiento en streaming o en tiempo real utiliza la computación en memoria RAM con 2 finalidades principales:</p> <ul style="list-style-type: none"> • lograr un alto rendimiento en tareas intensivas como las de MapReduce • permitir consultas en tiempo real <p>Mientras que el procesamiento por lotes con MapReduce puede requerir minutos, horas o incluso días para completar una tarea, la computación en memoria RAM permite obtener resultados en segundos o minutos.</p>
---------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

No obstante, un **aspecto negativo** es el **alto costo** de la **memoria RAM**. Aunque su precio ha ido bajando, actualmente **1 TB de RAM** cuesta alrededor de **4.000 - 6.000 €**. A pesar de esto, las ventajas en cuanto a velocidad y capacidad para realizar **consultas en tiempo real** a veces lo compensan.

4 - PLATAFORMAS PARA BD

En este tema vamos a conocer las características más importantes de Apache Hadoop y Apache Spark.

Primero veamos este video: <https://www.youtube.com/watch?v=sgn6yIERIkw> y haremos las actividades de [este documento](#).



<p>Plataformas BD</p>  <p>Características</p>	<ul style="list-style-type: none"> ■ Apache Software Foundation (Código abierto) ■ Procesamiento batch ■ MapReduce. Capacidad de procesar gran cantidad datos 	<p>Apache Hadoop es un proyecto de código abierto desarrollado por la fundación Apache Software Foundation que se ofrece de manera gratuita y está diseñado para funcionar en plataformas comerciales. Su objetivo principal es el procesamiento de datos por lotes (batch processing), para lo cual implementa el modelo de computación distribuida MapReduce, por lo que es posible almacenar y procesar grandes volúmenes de datos distribuyéndolos en múltiples nodos del sistema.</p>
----------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p>Plataformas BD</p>  <p>Características</p>	<ul style="list-style-type: none"> ■ Escalabilidad ■ Tolerancia a fallos ■ Flexibilidad 	<p>CARACTERÍSTICAS DE HADOOP</p> <ul style="list-style-type: none"> - Escalabilidad: Hadoop permite aumentar el número de nodos (desde unos pocos hasta cientos o miles) de manera transparente al usuario final. Esta escalabilidad es fundamental para el manejo de datos masivos en crecimiento constante. - Tolerancia a fallos: Hadoop emplea un sistema de ficheros distribuidos (HDFS - Hadoop Distributed File System) que duplica la información en varios nodos, garantizando así que, si un nodo falla, los datos se puedan recuperar desde otra ubicación sin interrupciones en el servicio. - Flexibilidad en el almacenamiento de datos: a diferencia de las bases de datos relacionales SQL tradicionales, Hadoop permite almacenar los datos sin una estructura rígida y procesarlos posteriormente según las necesidades.
----------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p>Plataformas BD</p>  <p>Retos</p>	<ul style="list-style-type: none"> ■ No recursividad ni interactividad <ul style="list-style-type: none"> ■ Coste de inicio ■ Múltiples fases Map/Reduce ■ Múltiples archivos 	<p>RETOS</p> <p>Un reto que debe afrontar Hadoop es que no permite la recursividad ni la interactividad</p> <ul style="list-style-type: none"> • No permite la recursividad porque está diseñado para tener un flujo de procesamiento lineal que pasa por distintas fases secuenciales. No es fácil ni eficiente reiniciar y modificar el flujo continuamente como se requiere hacer en un sistema recursivo. • No es interactivo porque no está diseñado para ofrecer respuestas rápidas. El procesamiento en lotes generalmente involucra una gran cantidad de lecturas, escrituras y movimientos de datos a través de varios nodos, por lo que se realiza un
--------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> uso intensivo del disco, lo que suele ralentizar el proceso. Hadoop tiene un coste inicial de arranque significativo debido a su arquitectura distribuida y la gestión que realiza YARN, lo que <u>no es ideal</u> para tareas que requieran respuesta rápida o continua. La dependencia de archivos intermedios y generan grandes cantidades de datos en múltiples archivos, lo cual puede impactar en el rendimiento y la administración del sistema.
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Plataformas BD  Retos	<ul style="list-style-type: none"> Falta programadores <ul style="list-style-type: none"> Conocimientos Java SQL+ Hadoop Administración arte y ciencia <p></p>	<p>Otro reto es la falta de programadores. Trabajar eficientemente con Hadoop requiere conocimientos específicos en el modelo MapReduce y en bases de datos no relacionales. Los programadores Java que no tienen experiencia en estos campos pueden encontrar difícil alcanzar un rendimiento óptimo al inicio. A menudo, en situaciones que requieren consultas rápidas, las empresas optan por utilizar bases de datos relacionales (SQL) en lugar de Hadoop para optimizar el rendimiento.</p> <p>Además, la administración de Hadoop es complicada porque necesita conocimientos avanzados en sistemas operativos y en cómo configurar correctamente sus componentes para que funcione de manera eficiente. A esto se le suele llamar una combinación de "arte y ciencia" porque no solo es necesario saber cómo funciona técnicamente (la "ciencia"), sino que también requiere experiencia y habilidad para optimizarlo según las necesidades específicas de cada caso (el "arte"). En otras palabras, administrar Hadoop bien es algo que lleva práctica y conocimientos profundos.</p>
----------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p>Plataformas BD</p>  <p>Retos</p>	<ul style="list-style-type: none"> ▪ Seguridad de datos ▪ Gestión y gobierno de los datos <p>Por otro lado, la arquitectura distribuida de Hadoop implica desafíos de seguridad. Aunque ha mejorado con el uso de Kerberos como protocolo de autenticación (Kerberos es un sistema de autenticación que ayuda a mejorar la seguridad en entornos distribuidos), se sigue enfrentando problemas en el manejo de la seguridad en entornos complejos.</p> <p>Hadoop no ofrece herramientas integradas para la gestión de la calidad, estandarización y gobernanza de los datos, lo que hace más difícil garantizar su precisión y cumplimiento normativo. A diferencia de otras plataformas, no tiene funciones para validar, organizar ni controlar los datos según estándares predefinidos. Las empresas que usan Hadoop deben invertir más tiempo y recursos en desarrollar soluciones personalizadas para asegurar la integridad de los datos y cumplir con normativas legales, lo que aumenta la complejidad del proceso de gestión.</p>
<p>Plataformas BD</p>  <p>Módulos</p>	<ul style="list-style-type: none"> ▪ HDFS (<i>Hadoop distributed file system</i>) ▪ YARN (<i>Yet Another Resource Negotiator</i>) <p>Hadoop tiene dos módulos principales:</p> <ol style="list-style-type: none"> 1. HDFS (Hadoop Distributed File System): <ul style="list-style-type: none"> ○ Es un sistema de archivos distribuido diseñado para manejar grandes volúmenes de datos de manera eficiente. ○ La unidad básica de almacenamiento en HDFS es el bloque. Cada archivo se divide en bloques, que se almacenan en distintos nodos del clúster. ○ Estos bloques se replican en múltiples nodos para asegurar la tolerancia a fallos y evitar la pérdida de datos en caso de que algún nodo falle. 2. YARN (Yet Another Resource Negotiator): <ul style="list-style-type: none"> ○ Es el motor de gestión de recursos y planificación de tareas de Hadoop. ○ YARN introduce una arquitectura que desacopla la gestión de los recursos del clúster de la ejecución de las tareas (por ejemplo, tareas MapReduce). ○ Los dos componentes principales de YARN son: <ul style="list-style-type: none"> ■ ResourceManager es el componente principal de YARN que se encarga de gestionar los recursos del clúster y asignarlos a las aplicaciones que lo solicitan. Sin embargo, el ResourceManager no sabe cómo ejecutar las tareas específicas de una aplicación. ■ El ApplicationMaster gestiona la ejecución de las tareas
<p>Plataformas BD</p>  <p>Módulos</p>	<ul style="list-style-type: none"> ▪ HDFS (<i>Hadoop distributed file system</i>) <ul style="list-style-type: none"> ▪ Sistema archivos distribuido ▪ Replicación bloques ▪ YARN (<i>Yet Another Resource Negotiator</i>) <ul style="list-style-type: none"> ▪ ResourceManager ▪ ApplicationMaster (<i>tareas MapReduce</i>)

	<p>individuales dentro de una aplicación, como las que se generan en un proceso de MapReduce. Además, se comunica con el ResourceManager para solicitar los recursos necesarios para la ejecución de cada una (recursos: memoria, núcleos de CPU, etc.) y luego coordina la asignación de esos recursos en los distintos nodos del clúster.</p>
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>Hadoop ha evolucionado hasta convertirse en un ecosistema completo de herramientas y aplicaciones diseñadas para implementar soluciones de Big Data de manera eficiente. Además del motor principal definido por módulos como HDFS y YARN, existen numerosas aplicaciones que pueden integrarse para construir una arquitectura Big Data adaptable a diversas necesidades.</p> <h3>Principales componentes del ecosistema Hadoop:</h3> <ol style="list-style-type: none"> HDFS y YARN: como ya se mencionó, estos son los módulos principales de Hadoop. HDFS gestiona el almacenamiento distribuido, mientras que YARN administra los recursos y coordina la ejecución de tareas. MapReduce: es el modelo de programación y procesamiento de datos que permite analizar grandes volúmenes de información de manera distribuida. Cada tarea se divide en "Map" (mapeo de datos) y "Reduce" (reducción y consolidación de los resultados). Apache Sqoop: es una herramienta que facilita la transferencia de datos entre bases de datos relacionales y el entorno Hadoop. Esto permite integrar datos estructurados (por ejemplo, de bases de datos SQL) con el ecosistema de Hadoop y exportar resultados procesados nuevamente a bases de datos relacionales. Herramientas para Bases de Datos No Relacionales: <ul style="list-style-type: none"> Apache HBase: una base de datos NoSQL que permite almacenar grandes volúmenes de datos de manera distribuida y escalable. Es ideal para aplicaciones que requieren consultas rápidas y acceso en tiempo real a grandes conjuntos de datos. Cassandra: otra base de datos NoSQL compatible con Hadoop, enfocada en
--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>la alta disponibilidad y escalabilidad horizontal, adecuada para grandes cantidades de datos distribuidos.</p> <p>5. Apache Flume: se utiliza para la ingestión de grandes cantidades de datos no estructurados, especialmente aquellos generados de manera continua. Los recopila y mueve hacia HDFS o HBase.</p> <p>6. Herramientas para Aprendizaje automático (Machine Learning):</p> <ul style="list-style-type: none"> ○ Apache Mahout: una librería de machine learning que proporciona algoritmos y herramientas para realizar aprendizaje automático sobre grandes volúmenes de datos. Se utiliza en aplicaciones de recomendación, clasificación y clustering. <p>7. Motores de Búsqueda y Análisis:</p> <ul style="list-style-type: none"> ○ En el ecosistema Hadoop también se pueden integrar motores de búsqueda y análisis de datos, permitiendo realizar consultas y recuperar información rápidamente (ejemplos: Apache Solr, Elasticsearch).
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

----- HASTA AQUÍ HADOOP. AHORA SPARK -----

Plataformas BD  Características	<ul style="list-style-type: none"> ▪ Procesamiento batch, streaming y en línea <ul style="list-style-type: none"> ▪ Código abierto ▪ Procesamiento iterativo de algoritmos aprendizaje automático 	<p>La otra plataforma fundamental es Apache Spark, y surge principalmente por las limitaciones de Apache Hadoop, que estaba diseñada para el procesamiento por lotes. Spark, además del procesamiento por lotes, permite el procesamiento en streaming y el tiempo real.</p> <p>También es de código abierto y, a diferencia de Hadoop, permite procedimientos iterativos, en especial para implementar métodos de aprendizaje automático que requieren ejecutar iteraciones sobre los datos, como puede ser realizar cálculos repetitivos sobre los mismos datos para mejorar los resultados. En contraste, Hadoop no es tan eficiente en este tipo de tareas, ya que su arquitectura basada en disco lo hace más lento para procesamientos repetitivos. Spark trabaja los datos sobre memoria RAM.</p>
--------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p>Plataformas BD</p>  <p>Características</p>	<p>Spark ofrece un alto rendimiento debido a 2 características clave:</p> <ul style="list-style-type: none"> • Computación en memoria: almacena datos en la memoria RAM, lo que evita el uso intensivo del disco. Esto es crucial para operaciones repetitivas y mejora el rendimiento al evitar transferencias de disco a memoria constantes. • DAG (Grafos Acíclicos Dirigidos, Directed Acyclic Graph): Spark organiza sus procesos usando DAG (Hadoop lo hacía con MapReduce). Lo hace a través de etapas optimizadas, lo cual permite definir un flujo de ejecución en secuencias de pasos. Cada etapa en este flujo puede optimizarse y paralelizarse, mejorando el rendimiento en comparación con los procesos lineales de MapReduce de Hadoop, que es más rígido, con pasos fijos de "map" y "reduce" que no siempre permiten optimizar cada etapa, lo cual limita la eficiencia de procesos complejos.
<p>Plataformas BD</p>  <p>Características</p>	<p>Spark es compatible con múltiples lenguajes de programación gracias a su API versátil, lo que facilita su uso para programadores de distintas disciplinas:</p> <ul style="list-style-type: none"> • Lenguajes compatibles: Python, R, Scala, Java y SQL. <p>En comparación con Hadoop, su lenguaje principal es Java. La mayoría de las aplicaciones de MapReduce en Hadoop se escriben utilizando la API de Java.</p>



- **RDD Resilient Distributed Datasets**
 - **Estructura de datos distribuida**
 - **Partición** (unidad atómica situada en un nodo)
 - RDD es colección de particiones
 - Coloca automáticamente operaciones en RDDs
 - **Tolerante a fallos**

El núcleo de la arquitectura de Spark es el **RDD**, que es una **estructura de datos distribuida** que **garantiza la tolerancia a fallos**. RDD significa **Conjunto de Datos Distribuido Resiliente**. La característica de "**resiliencia**" hace referencia a la capacidad de los RDDs para recuperarse de fallos de nodos, lo que asegura que el procesamiento de datos no se interrumpa si ocurre un error en algún nodo del clúster.

Un RDD representa una colección de datos distribuidos en **memoria RAM** a través de múltiples nodos en un clúster. La clave de los RDDs es que están diseñados para **procesamiento en memoria** y para ser **tolerantes a fallos**.

- **Estructura de datos distribuida:** los RDDs dividen la base de datos en **fragmentos** llamados "**particiones**". Cada partición se **distribuye** a través de **distintos nodos** de la red. Cada **partición de un RDD es una unidad atómica de datos que se almacena completa en un único nodo**. Esto significa que, aunque un RDD en su totalidad se distribuye a través de varios nodos, **cada partición individual de ese RDD se guarda en un solo nodo**. Si tienes un RDD que representa un conjunto de datos muy grande, este se dividirá en muchas particiones que estarán distribuidas en los diferentes nodos del clúster, pero cada partición estará completa en un solo nodo.
- **Tolerancia a fallos:** cada RDD almacena información sobre sus "padres". Esto significa que si una partición falla, Spark puede reconstruirla desde sus predecesoras, garantizando la estabilidad del sistema sin tener que reiniciar todo el proceso.
- **Automatización de operaciones:** los RDDs permiten **distribuir automáticamente las operaciones a través de sus particiones**, mejorando el **rendimiento** y la **gestión de grandes volúmenes de datos**, sin que el usuario tenga que preocuparse de cómo dividir o distribuir el trabajo.

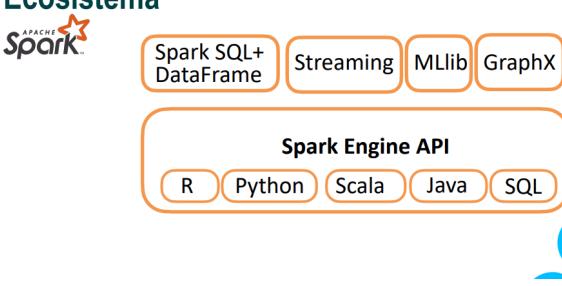
RESUMEN: Aunque tanto Hadoop como Spark dividen los datos en partes para distribuir el trabajo entre nodos, los **RDDs tienen características únicas**:

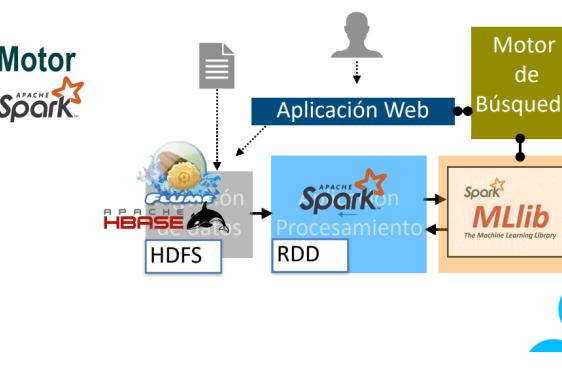
1. **Persistencia en memoria:**

- En Spark, los RDDs pueden mantenerse en **memoria RAM**, permitiendo que los datos procesados se reutilicen sin tener que volver a leerlos desde el disco cada vez que se necesitan, lo cual es especialmente útil en tareas

	<ul style="list-style-type: none"> iterativas (como el aprendizaje automático). ○ En Hadoop, cada tarea de MapReduce se lee y escribe en el disco en cada paso, lo cual es más lento. <p>2. Tolerancia a fallos con “parentesco” (o linaje):</p> <ul style="list-style-type: none"> ○ Los RDDs no solo contienen datos, sino también un registro de "parentesco". Esto significa que cada RDD "sabe" cómo fue creado a partir de otros RDDs (sus "padres"). Si alguna partición falla, Spark no necesita rehacer todo el proceso; puede reconstruir solo las particiones necesarias siguiendo los pasos registrados en el linaje. ○ En contraste, Hadoop utiliza replicación de datos en HDFS para la tolerancia a fallos, duplicando los datos en varios nodos, lo cual consume más espacio y no tiene la flexibilidad del linaje de los RDDs.
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

 <p>Flujo de Datos → Spark Streaming → Lotes datos Entrada → Spark Engine → Lotes datos Procesados</p> <p>Dstream (RDD)</p>	<p>Como hemos mencionado, Spark también soporta procesamiento en streaming, ideal para manejar flujos de datos en tiempo casi real. En este esquema podemos ver cómo el Spark Streaming convierte el flujo de datos en pequeños lotes, llamados "mini-batches" (minilotes) en intervalos de tiempo definidos. Esto significa que en lugar de procesar los datos de manera continua, Spark los procesa en bloques pequeños y rápidos. Cada uno de los mini-batches genera un RDD, que representa los datos de ese intervalo de tiempo.</p> <p>Cada RDD se procesa en paralelo, aprovechando la infraestructura distribuida de Spark, lo cual permite obtener resultados casi en tiempo real.</p>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p>Ecosistema</p>  <pre> graph TD Spark[Apache Spark] --> SparkEngine[Spark Engine API] SparkEngine --> SparkSQL[Spark SQL+ DataFrame] SparkEngine --> Streaming[Streaming] SparkEngine --> MLlib[MLlib] SparkEngine --> GraphX[GraphX] SparkEngine --> Languages[R, Python, Scala, Java, SQL] </pre>	<p>Spark es un ecosistema de herramientas que permite realizar múltiples tareas en un entorno integrado:</p> <ul style="list-style-type: none"> • Componentes principales: <ul style="list-style-type: none"> ◦ Spark SQL: permite ejecutar consultas SQL sobre grandes volúmenes de datos y trabajar con DataFrames, que son estructuras tabulares similares a las tablas de bases de datos. ◦ Spark Streaming: es una extensión de Spark para procesar datos medida que llegan, mediante el uso de micro-batches. ◦ MLlib: es una biblioteca de aprendizaje automático que proporciona algoritmos y herramientas para la creación de modelos predictivos, como clasificación, regresión y clustering. ◦ GraphX: es una herramienta para el procesamiento y análisis de grafos (redes de relaciones), que permite realizar operaciones sobre nodos y aristas (como algoritmos de caminos más cortos). • Lenguajes compatibles: funciona con Python, Scala, Java, y SQL, lo que permite flexibilidad en el uso de distintos lenguajes dentro del mismo ecosistema.
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

 <pre> graph LR Motor[Motor Apache Spark] --> HDFS[HDFS] Motor --> HBase[HBase] Motor --> Flume[Flume] Motor --> App[Aplicación Web] Motor --> Search[Motor de Búsqueda] Motor --> MLlib[MLlib] HDFS --> RDD[RDD] HBase --> RDD Flume --> RDD RDD --> Procesamiento[Procesamiento] Procesamiento --> MLlib App --> Search Search --> MLlib </pre>	<p>El motor de Spark puede integrarse en varios sistemas y configuraciones:</p> <ul style="list-style-type: none"> • Integración con HDFS y HBase: Spark puede trabajar sobre el sistema de archivos distribuido Hadoop (HDFS) y bases de datos NoSQL como HBase. • Compatibilidad con Hadoop: aunque es independiente, Spark puede ejecutarse sobre un clúster Hadoop, utilizando su infraestructura de almacenamiento y recursos de cómputo. • Aplicación web y motores de búsqueda: Spark se puede utilizar en aplicaciones web y combinar con motores de búsqueda, lo que lo hace adecuado para tareas complejas de análisis de datos y recuperación de información.
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Similitudes/ diferencias



■ Características comunes

- Localidad datos
- Ejecución en etapas
- HDFS para la persistencia en disco

■ Diferencias

- En memoria / en disco
- DAG optimizado / manual

SIMILITUDES:

- **Localidad de datos:** ambos sistemas priorizan la **localidad de los datos**, es decir, buscan **minimizar el movimiento de datos en la red**. En lugar de mover grandes volúmenes de datos a los nodos de cómputo, ambos frameworks intentan llevar la computación lo más cerca posible a los datos para **reducir la latencia y mejorar el rendimiento**.
- **Procesamiento en etapas:** tanto Hadoop como Spark dividen los procesos en **etapas de ejecución**. Sin embargo, las etapas en Hadoop suelen ser más limitadas y están centradas en los pasos de *Map* y *Reduce*, mientras que en Spark las etapas se organizan de forma más flexible y optimizada a través de un DAG.

DIFERENCIAS:

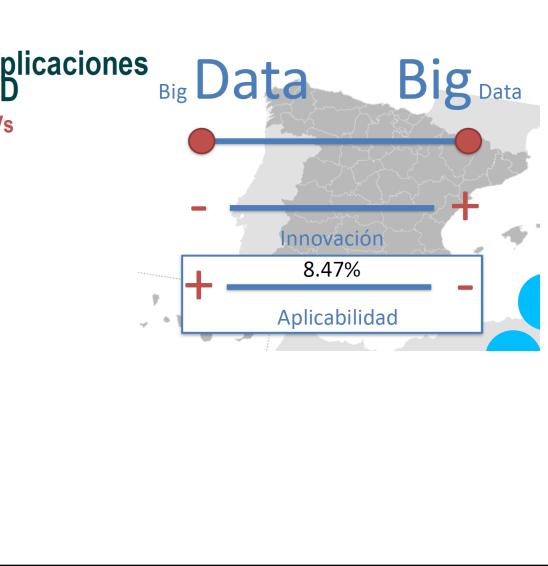
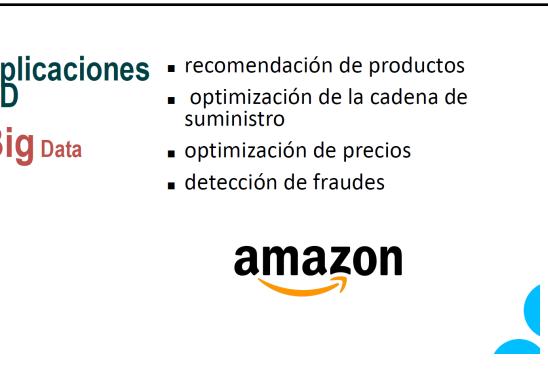
1. Manejo de memoria y rendimiento:

- En **Hadoop**, el procesamiento se realiza en su mayoría a través de lectura y escritura en **disco**, lo que puede ser más lento, especialmente en tareas iterativas.
- En **Spark**, el procesamiento se realiza en **memoria RAM**, lo cual acelera las tareas, especialmente cuando se necesita ejecutar operaciones iterativas o repetitivas, como en el aprendizaje automático.

2. Optimización de etapas y planificación:

- En **Hadoop**, las etapas de procesamiento son definidas principalmente por el programador y siguen un flujo rígido de **MapReduce**, lo que puede limitar la optimización.
- En **Spark**, las tareas se organizan en un **DAG (Directed Acyclic Graph)** de etapas optimizadas. Spark define las etapas y sus dependencias de manera automática, optimizando el flujo de ejecución y permitiendo una mayor paralelización y eficiencia. El DAG permite que Spark divida las tareas en subtareas más pequeñas y elimine redundancias, aumentando el rendimiento.

5 - APLICACIONES DE BD

	<p>Basándonos en las 3Vs: velocidad, variedad y volumen:</p> <p>Distinguimos 2 tipos de “conceptos”:</p> <ul style="list-style-type: none">• BIG data: donde realmente se trata de aplicaciones Big Data. Son difíciles de llevar a cabo, por lo que requieren de grandes infraestructuras.• big DATA: realmente son aplicaciones basadas en datos. Son más sencillas de llevar a cabo. <p>En España, el 99% del tejido empresarial está formado por PYMES. Por lo tanto, es complicado tener sistemas de BIG data, ya que no cuentan con los recursos técnicos, económicos ni humanos necesarios para adoptar tecnologías avanzadas. Es una pena porque aunque producen gran innovación, tienen una difícil aplicabilidad en pequeñas empresas debido a los altos costes y la falta de recursos. En España únicamente se aplica BIG data en alrededor de un 8% de las empresas.</p> <p>A continuación, vamos a diferenciar entre empresas grandes multinacionales (capaces de aportar soluciones con grandes cantidades de datos, BIG data) y empresas pequeñas a nivel español (aportan soluciones basadas en datos, big DATA).</p>
	<p>Empezamos con empresas grandes empresas a nivel mundial que usan el Big Data:</p> <p>Amazon:</p> <ul style="list-style-type: none">• Tiene un sistema de recomendaciones de producto.• Ha optimizado la cadena de suministros• Optimiza los precios.• Detecta fraudes.

<p>Aplicaciones BD</p> <p>Big Data</p> <ul style="list-style-type: none"> ▪ Sistema de recomendación ▪ Diseño de productos (producción de series y películas)  	<p>Netflix:</p> <p>Sabe qué queremos ver, por ello ofrece sistemas de recomendación. No sólo utiliza nuestra forma de navegar por la plataforma, cuántos vídeos vemos seguidos, cuánto tiempo tardamos en elegir. Sino que también lo utiliza para diseñar productos (series y películas) nuevas.</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p>Aplicaciones BD</p> <p>Big Data</p> <ul style="list-style-type: none"> ▪ Fidelización de clientes  	<p>Apple:</p> <p>Utiliza big data para fidelizar a los clientes, sus aplicaciones y dispositivos recogen datos.</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------

<p>Aplicaciones BD</p> <p>Big Data</p> <ul style="list-style-type: none"> ▪ Diseño de productos ▪ Análisis de sentimientos  	<p>Coca cola:</p> <p>Lo utiliza para diseñar nuevos productos, como máquinas que mezclan sabores para crear nuevos productos porque saben que tendrán éxito, como Cherry. También realizan análisis de sentimientos en redes sociales. Saben qué opinamos de sus productos en distintos países o dónde y cuándo lo consumimos.</p>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Aplicaciones
BD**

Big Data



Ubicación de nuevas tiendas

Starbucks:

Tienen aplicaciones que le permiten ubicar las tiendas analizando características como flujo, tipología de los usuarios... así saben de antemano la tasa de éxito que tendrá cada uno de sus locales.

El resto del tema puede seguirse en el siguiente documento: [Plan de digitalización de las PYMES en España 2021-2025](#)