



Sistemas de Big Data

Curso de especialización en Inteligencia Artificial y Big Data.



Sistemas de ficheros y almacenamiento. Repaso/Introducción

- **Sistemas de ficheros:** Organizan y gestionan el almacenamiento en archivos. Permiten guardar y recuperar datos de forma independiente.

- **RAID:** Tecnología para distribuir datos en múltiples discos y obtener redundancia o mayor capacidad.
- **Sistemas distribuidos:** Distribuyen los ficheros en bloques en un clúster. Permiten ficheros más grandes que cualquier unidad de almacenamiento. Proporcionan redundancia y alta disponibilidad.
 - **HDFS:** Sistema de ficheros distribuido de Hadoop, para almacenar grandes cantidades de datos en clústeres de bajo coste. Permite escalabilidad y tolerancia a fallos. Acceso por línea de comandos o vía API.

Almacenamiento en memoria: Permite velocidades mucho mayores al evitar discos. Se usa para flujos rápidos de datos y análisis en tiempo real. No es persistente por sí mismo.



Bases de datos NoSQL. Repaso/Introducción

Conceptos generales:

- **Bases de datos no relacionales:** No siguen el modelo tradicional de bases de datos relacionales.
- **Permiten flexibilidad al ser schema-on-read:** No requieren un diseño previo estricto, lo que facilita la adaptación a cambios en la estructura de datos.
- **Pueden usar sharding y replicación:** Para lograr escalabilidad y tolerancia a fallos, distribuyendo datos entre múltiples servidores y manteniendo copias redundantes.



Tipos de Bases de datos NoSQL. Repaso/Introducción

Tipos de bases de datos NoSQL:

- **Documentales:** Almacenan documentos XML y/o JSON arbitrarios por clave en "colecciones". Permiten la heterogeneidad de datos y ofrecen indexación parcial.
- **Clave-valor:** Utilizan pares clave-valor arbitrarios.
- **Columnares:** Almacenan datos por columnas en lugar de por filas, lo que ahorra espacio y mejora las lecturas. No están optimizadas para escrituras frecuentes.
- **Grafo:** Representan datos en nodos y relaciones, permitiendo recorridos eficientes en grafos complejos. Sin embargo, pueden ser limitadas en tamaño y requerir estrategias específicas de sharding.
- ...



Conceptos generales que pueden aplicar al caso de NoSQL. Repaso/Introducción

- **Sharding:** Dividir los conjuntos de datos para distribuirlos por un clúster.
- **Replicación:** Hacer copias de los datos en distintos nodos del clúster.
- **Sharding con replicación:** Combinar las dos características anteriores para obtener las ventajas de ambas.

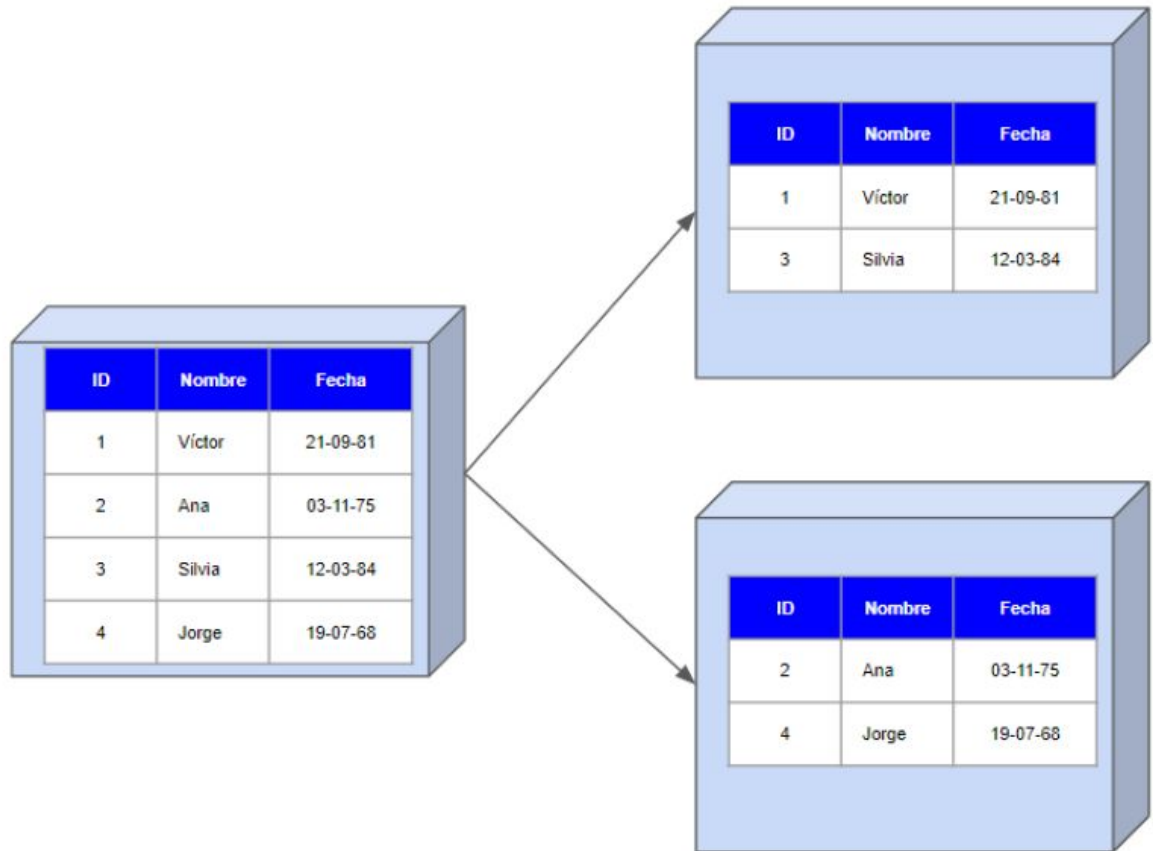
Es importante destacar que tanto el sharding como la replicación no son en absoluto exclusivos de las bases de datos NoSQL, sino que se emplean en muchos otros sistemas.

Tampoco todas las bases de datos NoSQL cuentan con estas características, aunque la mayor parte sí.

Conceptos generales que pueden aplicar al caso de NoSQL.

Repaso/Introducción

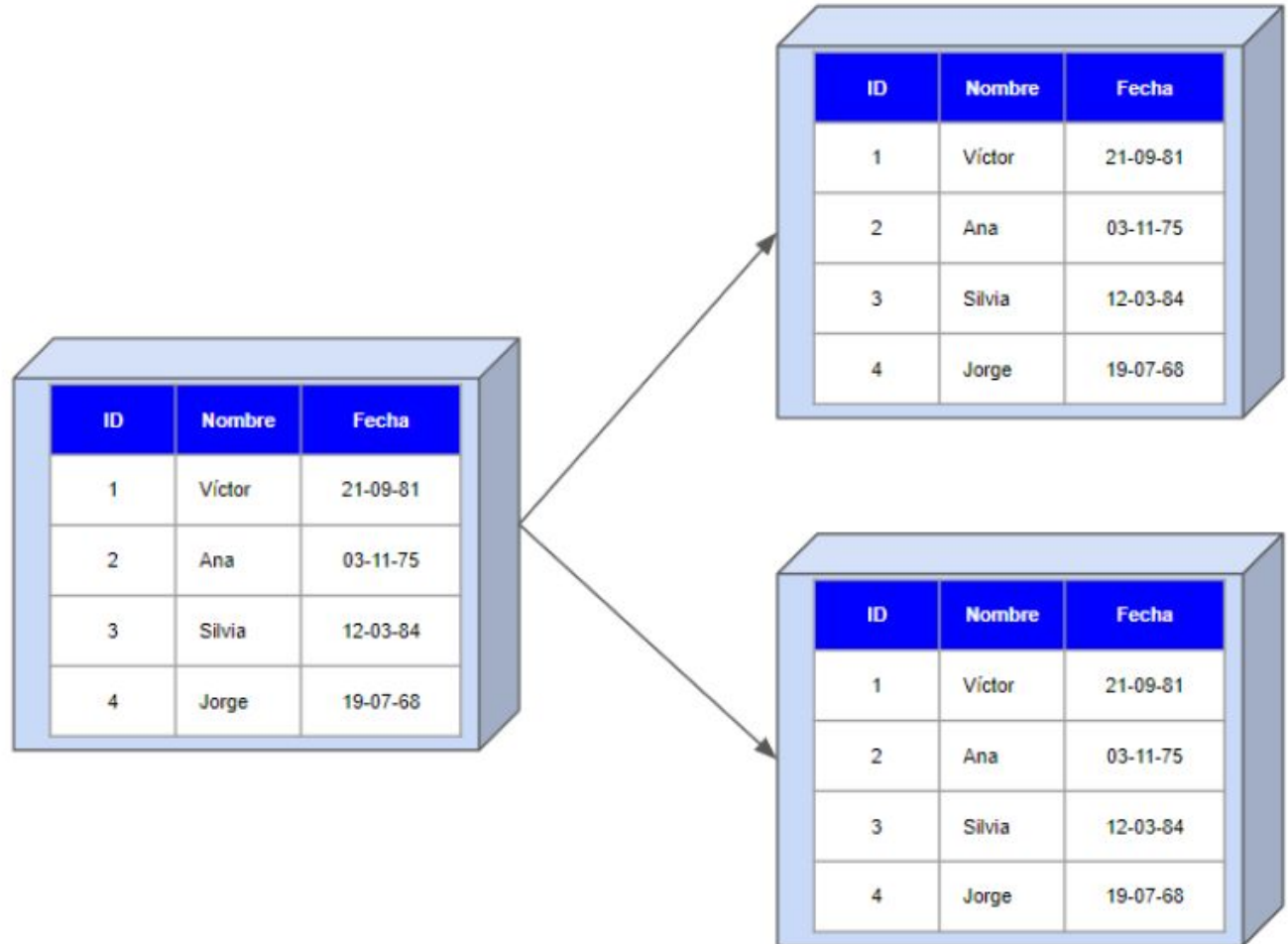
Sharding



Conceptos generales que pueden aplicar al caso de NoSQL.

Repaso/Introducción

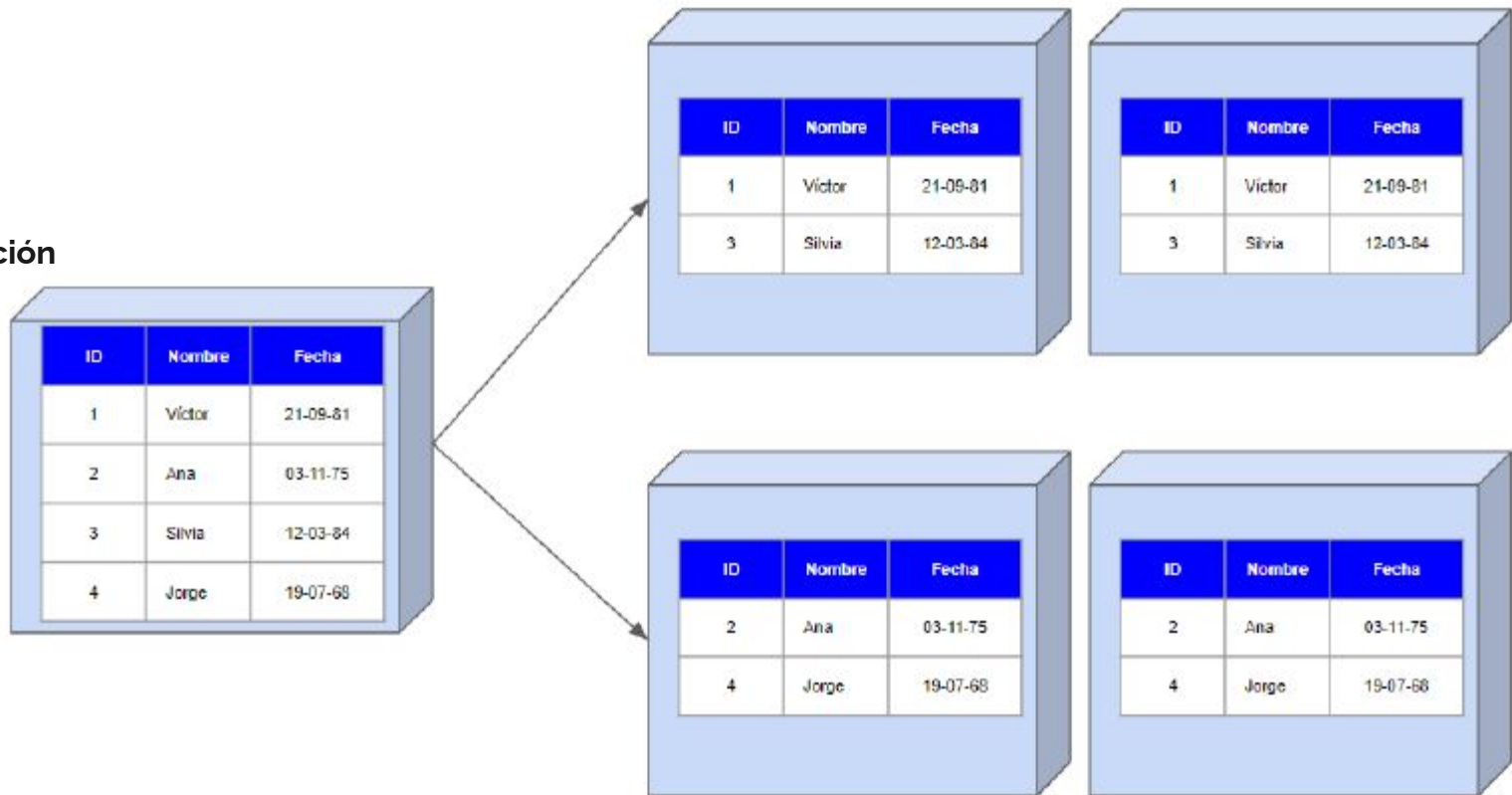
Replicación



Tipos de Bases de datos NoSQL.

Repaso/Introducción

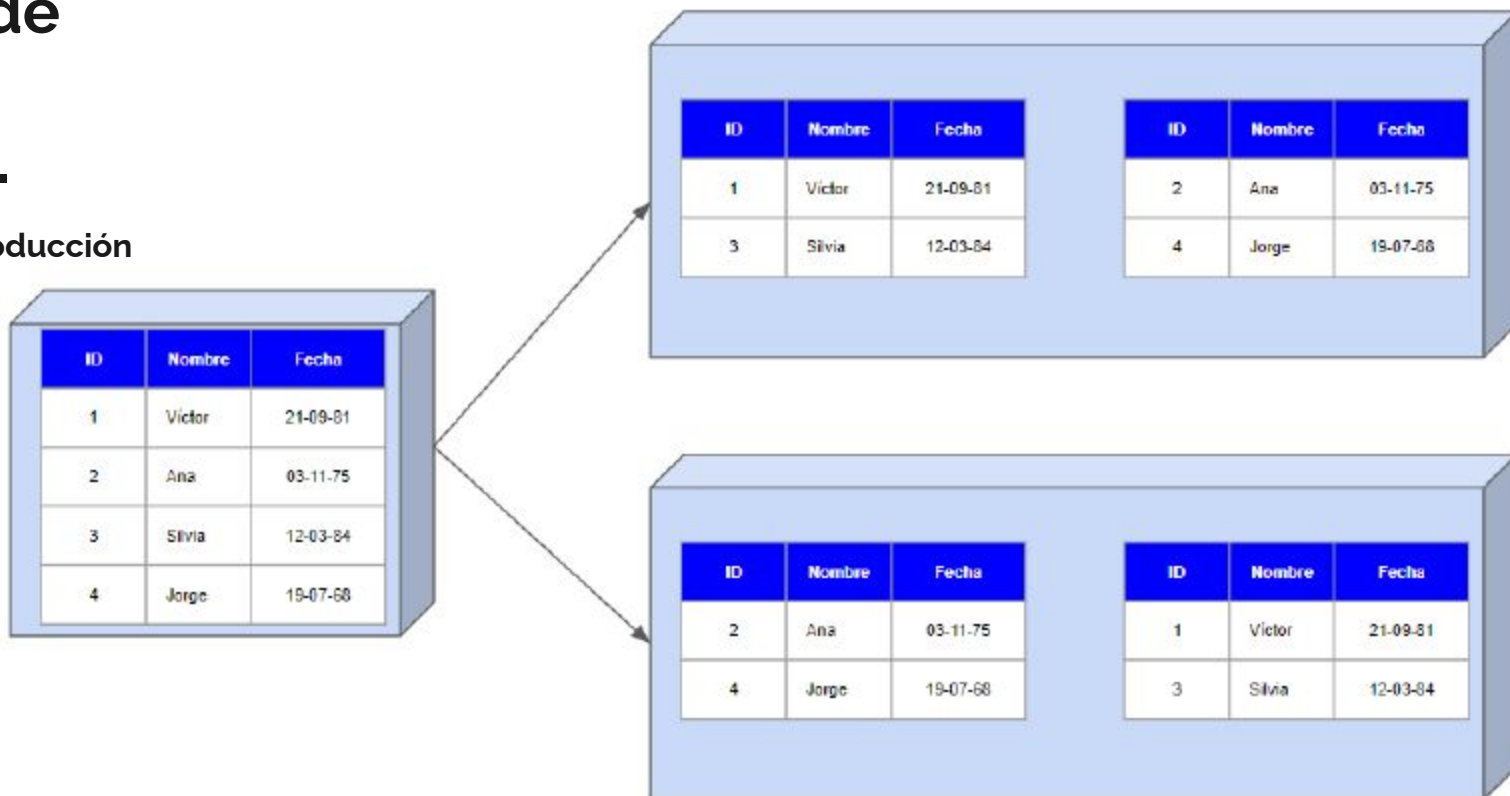
Sharding con replicación (ejemplo 1)



Tipos de Bases de datos NoSQL.

Repaso/Introducción

Sharding con replicación (ejemplo 2)



Capas de los sistemas de Big Data

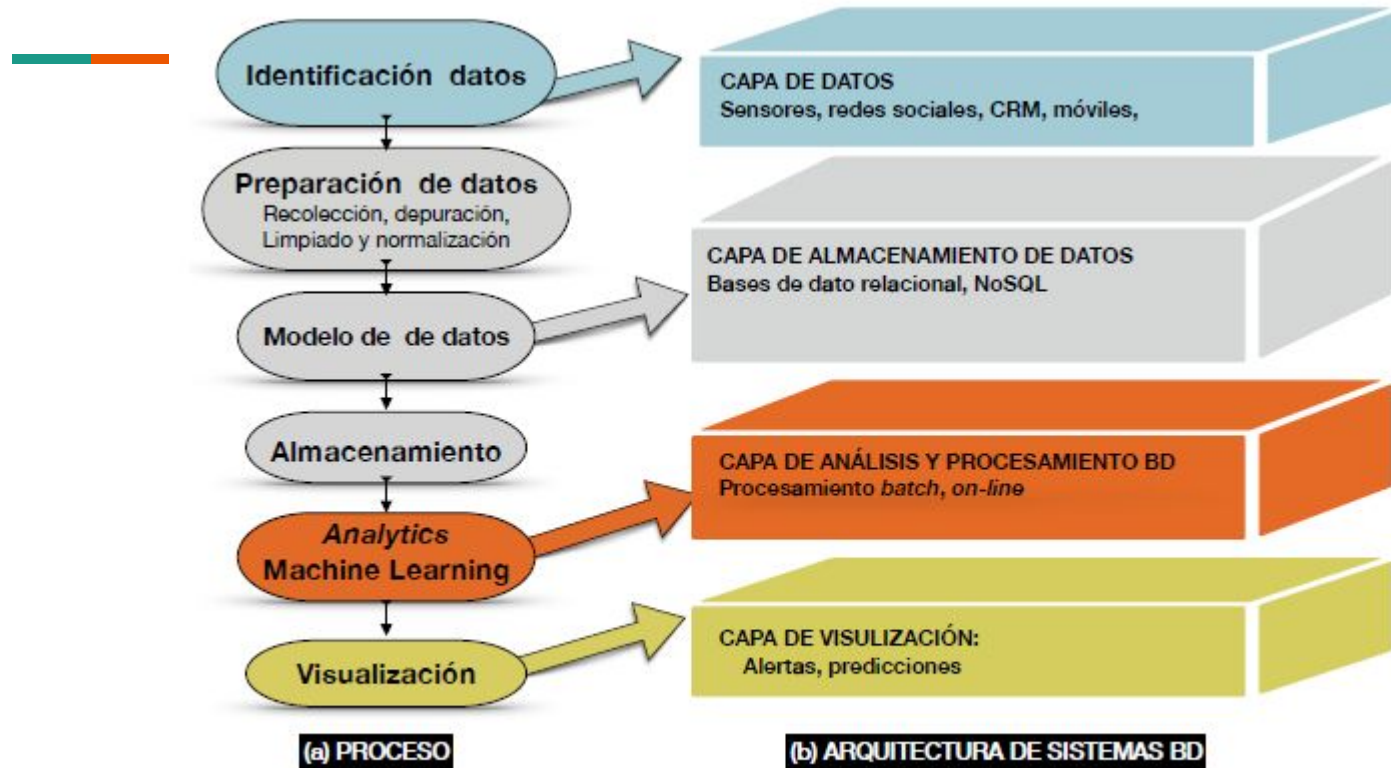


Figura 3.1: Procesos y estructura multicapa en los sistemas Big Data.

Arquitectura de sistemas Big Data

1. Una **aplicación web** donde los usuarios pueden elegir dentro de un catálogo de servicios.
2. Una **aplicación de ingestión de datos** que está diseñada para extraer los registros y procesarlos.
3. Una **aplicación de procesamiento** que funciona como el motor de la arquitectura.
4. Una **aplicación de aprendizaje automático** para obtener información útil para el usuario.
5. Un **motor de búsqueda** para extraer el análisis de nuestros datos.

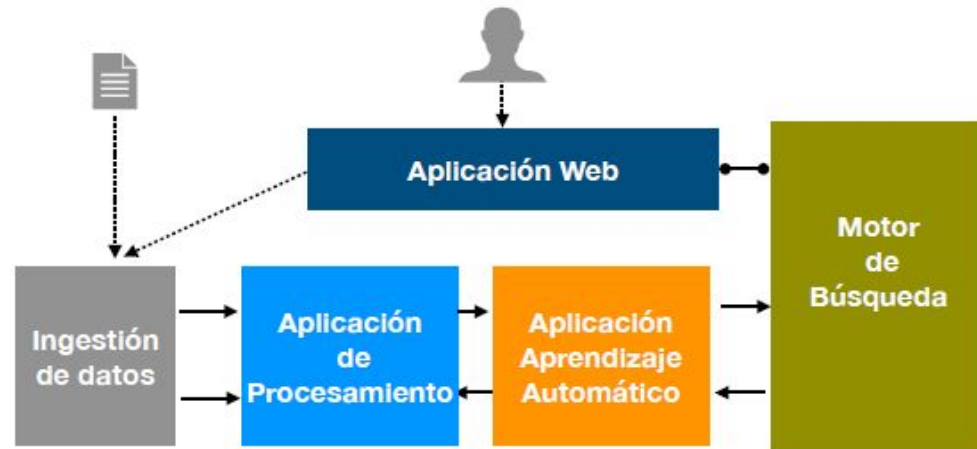


Figura 3.2: Arquitectura para los sistemas Big Data.



Tipología de sistemas Big Data

- **Procesamiento batch o por lotes.** En estos sistemas BD **los datos están todos disponibles y almacenados antes de iniciar el procesamiento.** Cuando se produce la ejecución del programa, debido al elevado coste computacional, ya no se produce ninguna iteración con el usuario hasta la obtención del resultado final.
- **Procesamiento streaming.** Este tipo de **procesamiento y análisis de datos se basan en la implementación de un modelo de flujo de datos en el que los datos asociados a series de tiempo (hechos) fluyen continuamente a través del sistema BD.** Estos sistemas BD no tienen limitaciones de tiempo en el procesamiento del flujo de datos, ni tampoco existe una exigencia en cuanto al tiempo de generación del output por cada input recibido en el sistema..
- **Procesamiento en tiempo real.** En este tipo de procesamiento es necesario una respuesta a tiempo real. El procesamiento en tiempo real se utiliza en aplicaciones donde la velocidad es esencial, como sistemas de control de procesos, alertas de seguridad, juegos en línea y aplicaciones de recomendación en tiempo real.



Implementación en sistemas cloud.

La nube ofrece servicios de computación y almacenamiento, de tal forma que el usuario pueda disponer de estos recursos hardware software sin la necesidad de adquirirlos. La posibilidad de alquilar estos recursos de hardware en vez de comprarlos e instalarlos en el centro de procesamiento de datos (CPD) de la organización presenta ventajas en cuanto ahorro de costes, flexibilidad, ahorro de tiempo de instalación, escalabilidad, información en tiempo real y actualizaciones periódica pero presenta una dependencia de los proveedores y de la conexión a Internet.



Implementación en la nube

Las soluciones cloud presentan los siguientes retos fundamentales que deben acometerse por el propietario y usuario de la infraestructura:

- 1. Hackeo.** Este término hace referencia a las formas de **obtener, copiar, leer o ver información confidencial llevadas a cabo por personas no autorizadas**. Estas acciones se realizan con herramientas sofisticadas que actúan durante el proceso de intercambio de información dentro de redes inseguras. El hackeo es un desafío global en el que numerosas organizaciones gastan millones de euros/dolares para proteger sus infraestructuras. **Muchos de los servicios BD están empleando sistemas criptográficos para evitar ataques y poder preservar un nivel de funcionalidad para los usuarios.**
- 2. Protección de datos.** Este aspecto hace referencia a la **divulgación de información no autorizada debido a protocolos deficitarios o mal uso de los sistemas de información**. Incurrir en estos fallos tiene como consecuencia una pérdida de reputación para las organizaciones.



Implementación en la nube

Las soluciones cloud presentan los siguientes retos fundamentales que deben acometerse por el propietario y usuario de la infraestructura:

3. Cuellos de botella en el procesamiento. Uno de los desafíos clave para los proveedores de servicios en la nube es poder atender tantos servicios como los demandados por los usuarios. Una de las demandas de los clientes es tener la mejor accesibilidad a los servicios en cualquier momento. Si el sistema no funciona como lo esperado se denomina cuello de botella en el procesamiento. El cuello de botella se produce por un incremento del número de procesos realizados en un servidor. Los proveedores de estos servicios deben gestionar estas situaciones desplegando más servidores, replicando la información y el middleware entre los servidores.

4. Escalabilidad. Este aspecto hace referencia a la capacidad del sistema de poder estar disponible cuando se produce un incremento esperado de la demanda. Alcanzar esta característica en los sistemas de BD pasa por el uso de aplicaciones software que usen algoritmos eficientes (con baja complejidad computacional) y ejecuciones concurrentes de las tareas que involucran, además de tener la capacidad de expandir la infraestructura cloud.



Implementación en la nube

Las soluciones cloud presentan los siguientes retos fundamentales que deben acometerse por el propietario y usuario de la infraestructura:

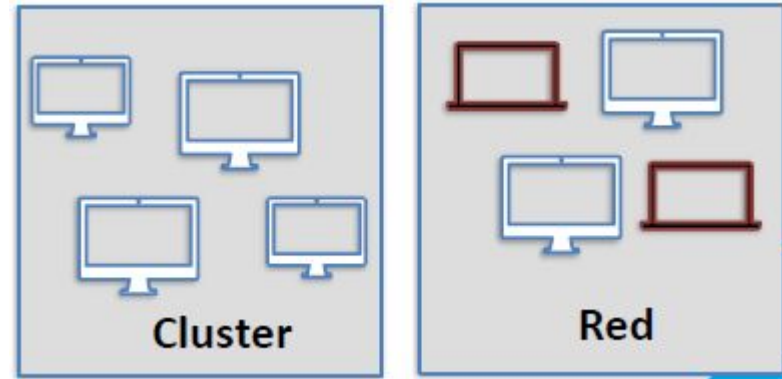
5. Concurrencia. La concurrencia es la ejecución simultánea de varios procesos que comparten recursos. El reto es no tener tiempos muertos entre procesos, lo requiere estrategias óptimas de sincronización.

6. Tolerancia a fallos. Esta característica de los sistemas BD requiere la detección de fallos de una manera controlada de tal forma que se pueda interrumpir durante el menor tiempo posible las componentes con un funcionamiento erróneo. El objetivo es minimizar las pérdidas de datos. Este reto requiere una gestión eficiente de los recursos y de los mecanismos de control, para incorporar y liberar recursos de una forma óptima.

7. Disponibilidad. Los usuarios demandan tener acceso a los servicios en cualquier momento y desde cualquier lugar. Para los proveedores de servicios es esencial que exista disponible al menos una máquina que pueda ejecutar un determinado tipo de servicios. Si un servidor se cae, es necesario que el sistema funcione con los recursos restantes. Los recursos críticos deben ser replicados para asegurar un determinado nivel de servicio.

Procesamiento batch de BD: El marco MapReduce

Un sistema BD con un procesamiento batch posee una arquitectura de computación distribuida donde los datos se recolectan y almacenan antes de que alimenten la capa de análisis. La infraestructura para la computación puede ser un cluster que está formado por multitud de ordenadores con un hardware similar y conectados en una red local o una red de ordenadores distribuidos geográficamente y con un hardware heterogéneo. Estos sistemas llegan a procesar grandes cantidades de datos que pueden llegar a tamaños de petabytes.

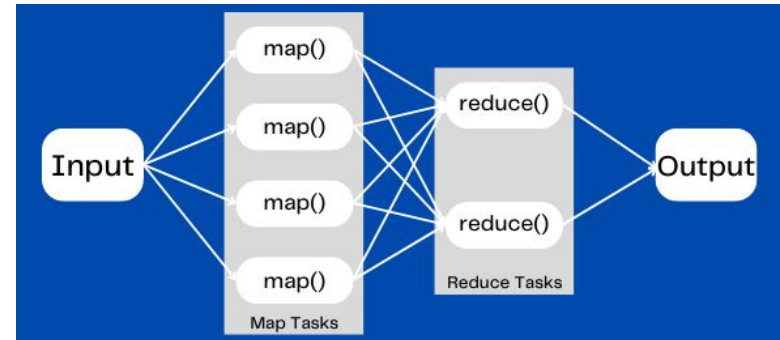


Procesamiento batch de BD: El marco MapReduce

Hasta la fecha, el enfoque principal para el procesamiento por lotes de bases de datos es el conocido como MapReduce, que Google introdujo en 2004 para abordar el desafío de indexar la web a medida que crecía. Esta solución se apoyaba en un sistema de archivos distribuido que permitía el procesamiento simultáneo de volúmenes significativos de información.

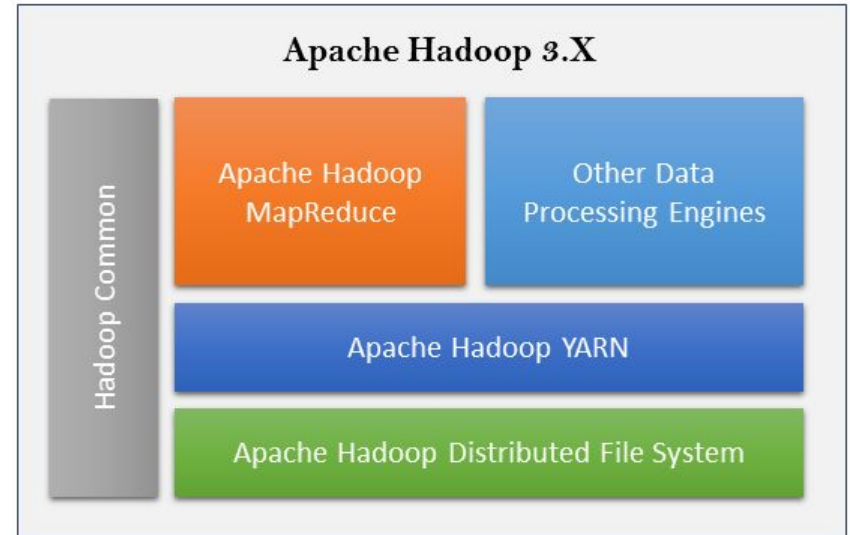
Google propuso que su MapReduce marco contuviese tres módulos:

- una máquina para ejecutar MapReduce,
- un sistema de ficheros distribuidos llamado Google File System (GFS) y
- una base de datos NoSQL distribuida llamada BigTable.



Procesamiento batch de BD: El marco MapReduce

Tras este anuncio la fundación Apache Software Foundation comenzó la implantación en código abierto de MapReduce. Los módulos fundamentales de este proyecto son Hadoop MapReduce y Hadoop YARN como motores de ejecución y Hadoop Distributed File System (HDFS) y Hbase como sustitutos de BigTable





Procesamiento batch de BD: El marco MapReduce

Los requisitos fundamentales de un procesamiento batch son

- **un modelo de programación:** para definir cómo se deben llevar a cabo las tareas de procesamiento en un entorno batch.
- **una escalabilidad automática y lineal:** La escalabilidad automática y lineal implica que un sistema de procesamiento batch pueda aumentar su capacidad para manejar volúmenes crecientes de datos sin requerir intervención manual.
- **ser tolerante a fallos:** Puede lograrse mediante la replicación de datos y tareas, la supervisión constante y la recuperación automática.
- **no tener pérdidas de datos:** utilizando técnicas como el almacenamiento redundante y la confirmación de la integridad de los datos.
- **poder procesar todo (o casi) el conjunto histórico de datos:** fundamental para garantizar que no se omita información relevante durante el proceso de análisis y toma de decisiones.



Procesamiento batch de BD: El marco MapReduce

MapReduce es un modelo de programación distribuido. Los registros se abstraen como una estructura clave-valor, $\langle k; v \rangle$ que permiten almacenarlos y administrar sus operaciones. El valor v puede contener cualquier tipo de dato.

La implementación distribuida del esquema MapReduce se realiza con una arquitectura master-esclavo como se muestra en la Figura 3.3. Los datos de entrada se almacenan en un sistema de ficheros distribuidos y se particionan en subconjuntos que alimentan a los diferentes nodos esclavos.

Procesamiento batch de BD: El marco MapReduce

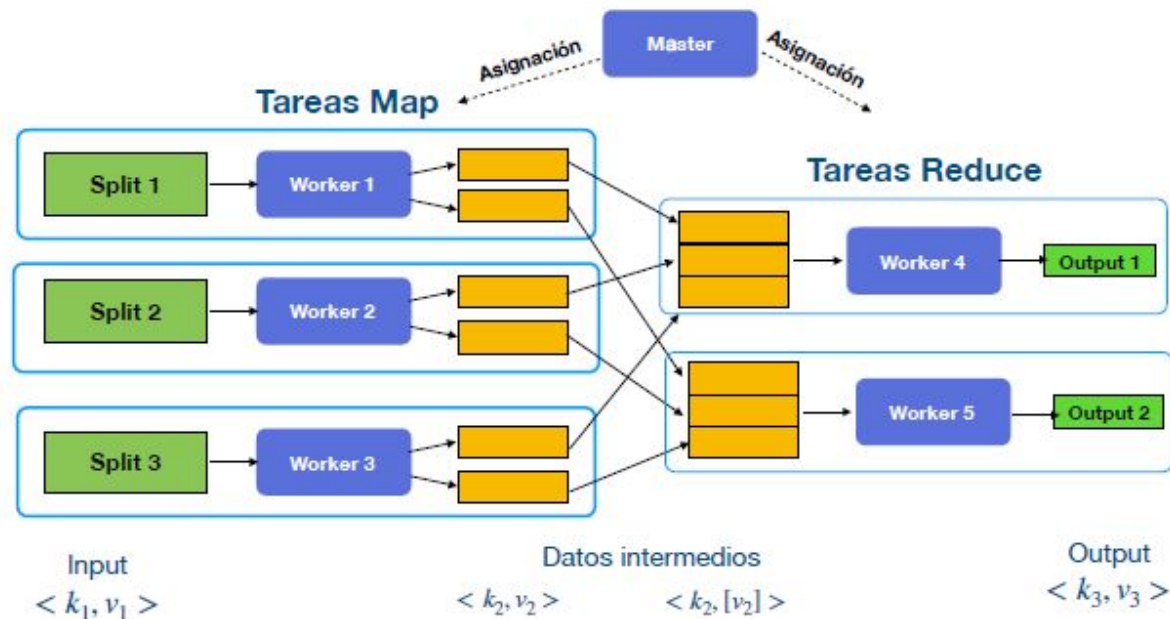


Figura 3.3: Esquema MapReduce.



Procesamiento batch de BD: El marco MapReduce

MapReduce es un enfoque de programación que se utiliza en sistemas distribuidos para procesar grandes conjuntos de datos. En este modelo, los datos se manejan como pares clave-valor, donde la clave (k) es una etiqueta que identifica un registro de datos y el valor (v) puede ser cualquier tipo de información.

Ejemplo: imagina que tienes una colección de datos que representa registros de ventas de una tienda en línea. Cada registro de venta consta de una clave (k) que podría ser un número de identificación único para la venta, y un valor (v) que contiene toda la información sobre esa venta, como el nombre del cliente, los productos comprados, el precio, la fecha, etc.: Por ejemplo:

- Clave (k): Número de factura
- Valor (v): Detalles de la compra, como cliente, productos, fecha y precio.

Este enfoque permite una forma eficiente de procesar grandes cantidades de información, ya que se divide en pequeñas tareas (map) y se combina en resultados finales (reduce), lo que facilita el procesamiento distribuido y paralelo de datos masivos.



Procesamiento batch de BD: El marco MapReduce

MapReduce consta de tres fases fundamentales:

1. **Map (Mapeo):** En esta etapa, cada nodo esclavo procesa inicialmente los datos locales y produce resultados intermedios que se almacenan en un área de almacenamiento temporal. El nodo maestro se encarga de asegurar que cada dato se procese una sola vez.
2. **Shuffle (Ordenamiento y Agrupamiento):** Posteriormente, los nodos esclavo llevan a cabo la fase de "shuffle". En esta etapa, los datos se redistribuyen en función de las claves de salida (k2), de manera que todos los datos relacionados con una misma clave (k2) se agrupen en un mismo nodo esclavo.
3. **Reduce (Reducción):** En la última fase, los nodos esclavo procesan cada grupo de datos de salida, agrupados por clave, y realizan las operaciones necesarias para obtener los resultados finales.



Procesamiento batch de BD: El marco MapReduce

Ejemplo de MapReduce: Análisis de Registros de Acceso a un Sitio Web

Supongamos que tenemos registros masivos de acceso a un sitio web y deseamos analizarlos para obtener información valiosa, como las páginas más visitadas y la distribución horaria de las visitas. MapReduce puede ser una herramienta poderosa para este propósito.

Registro 1: URL="/pagina1", Hora="09:30"

Registro 2: URL="/pagina2", Hora="10:15"

Registro 3: URL="/pagina1", Hora="11:45"

Registro 4: URL="/pagina3", Hora="12:30"

Registro 5: URL="/pagina1", Hora="14:20"

Registro 6: URL="/pagina2", Hora="15:45"



Procesamiento batch de BD: El marco MapReduce

Ejemplo de MapReduce: Análisis de Registros de Acceso a un Sitio Web

Paso 1: Map (Mapeo)

Inicialmente, dividimos los registros de acceso en trozos más pequeños y distribuimos estos fragmentos a los nodos de procesamiento.

Cada nodo realiza la fase de mapeo, examinando los registros de acceso asignados y extrayendo información relevante, como la URL de la página visitada y la hora del acceso.

Cada nodo emite pares clave-valor, donde la clave es la URL de la página y el valor es un 1 para indicar una visita.

- Fragmento 1 (Nodo 1):
 - Clave: `"/pagina1"`
 - Valor: 1
- Fragmento 2 (Nodo 2):
 - Clave: `"/pagina2"`
 - Valor: 1
- Fragmento 3 (Nodo 1):
 - Clave: `"/pagina1"`
 - Valor: 1
- Fragmento 4 (Nodo 2):
 - Clave: `"/pagina3"`
 - Valor: 1
- Fragmento 5 (Nodo 1):
 - Clave: `"/pagina1"`
 - Valor: 1
- Fragmento 6 (Nodo 2):
 - Clave: `"/pagina2"`
 - Valor: 1



Procesamiento batch de BD: El marco MapReduce

Ejemplo de MapReduce: Análisis de Registros de Acceso a un Sitio Web

Paso 2: Shuffle (Ordenamiento y Agrupamiento)

Luego, los resultados intermedios de los nodos se mezclan y se agrupan en función de la URL de la página. De esta manera, se agrupan todas las visitas a una misma página en un nodo para el siguiente paso.

- Clave: `"/pagina1"`
 - Valor: `[1, 1, 1]`
- Clave: `"/pagina2"`
 - Valor: `[1, 1]`
- Clave: `"/pagina3"`
 - Valor: `[1]`



Procesamiento batch de BD: El marco MapReduce

Ejemplo de MapReduce: Análisis de Registros de Acceso a un Sitio Web

Paso 3: Reduce (Reducción)

Ahora, un conjunto de nodos se encarga de procesar cada grupo de visitas a una página específica. En la fase de reducción, se suman los valores (los 1's) para obtener el número total de visitas a esa página. También, se analiza la distribución horaria de las visitas y se genera un informe con la hora más común en la que la página fue visitada.

Este enfoque permite analizar grandes volúmenes de datos de registro de manera eficiente y paralela, lo que sería muy difícil de hacer manualmente o con métodos tradicionales.


- Clave: `"/pagina1"`
 - Valor: `[1, 1, 1]`
 - Resultado: Total de visitas a `"/pagina1"` = 3
- Clave: `"/pagina2"`
 - Valor: `[1, 1]`
 - Resultado: Total de visitas a `"/pagina2"` = 2
- Clave: `"/pagina3"`
 - Valor: `[1]`
 - Resultado: Total de visitas a `"/pagina3"` = 1

- `"/pagina1"`: 3 visitas
- `"/pagina2"`: 2 visitas
- `"/pagina3"`: 1 visita

Procesamiento batch de BD: El marco MapReduce

Ejemplo de MapReduce: Análisis de Registros de Acceso a un Sitio Web

MAP



Registro 1: URL="/pagina1", Hora="09:30"
Registro 2: URL="/pagina2", Hora="10:15"
Registro 3: URL="/pagina1", Hora="11:45"
Registro 4: URL="/pagina3", Hora="12:30"
Registro 5: URL="/pagina1", Hora="14:20"
Registro 6: URL="/pagina2", Hora="15:45"

- Fragmento 1 (Nodo 1):
 - Clave: "/pagina1"
 - Valor: 1
- Fragmento 2 (Nodo 2):
 - Clave: "/pagina2"
 - Valor: 1
- Fragmento 3 (Nodo 1):
 - Clave: "/pagina1"
 - Valor: 1
- Fragmento 4 (Nodo 2):
 - Clave: "/pagina3"
 - Valor: 1
- Fragmento 5 (Nodo 1):
 - Clave: "/pagina1"
 - Valor: 1
- Fragmento 6 (Nodo 2):
 - Clave: "/pagina2"
 - Valor: 1

SHUFFLE

- Clave: "/pagina1"
 - Valor: [1, 1, 1]
- Clave: "/pagina2"
 - Valor: [1, 1]
- Clave: "/pagina3"
 - Valor: [1]

REDUCE

- Clave: "/pagina1"
 - Valor: [1, 1, 1]
 - Resultado: Total de visitas a "/pagina1" = 3
- Clave: "/pagina2"
 - Valor: [1, 1]
 - Resultado: Total de visitas a "/pagina2" = 2
- Clave: "/pagina3"
 - Valor: [1]
 - Resultado: Total de visitas a "/pagina3" = 1

- "/pagina1": 3 visitas
- "/pagina2": 2 visitas
- "/pagina3": 1 visita



Procesamiento batch de BD: El marco MapReduce

Ejemplo2 de MapReduce: Seguimiento de Actividad de Usuarios

Imaginemos que tenemos una lista de comentarios de usuarios y queremos descubrir cuándo un usuario hizo su primer comentario, cuándo hizo el último y cuántos comentarios hizo en total. En el contexto de MapReduce, la idea es llevar la lógica de procesamiento a los datos, en lugar de mover los datos a un solo lugar centralizado. Este proceso se lleva a cabo a través de las funciones map y reduce.

El proceso comienza cuando el servidor principal duplica la función map en un conjunto de servidores. Cada servidor local lee los datos de los archivos XML y crea pares clave-valor $\langle k;v \rangle$. La clave (k) es el identificador del usuario, y el valor (v) es una 3-tupla que almacena [primera vez; última vez; conteo]. La función map completa los valores de "primera vez" y "última vez" con las fechas de creación de los comentarios y establece el valor "conteo" en 1, indicando que el usuario ha realizado un comentario.



Procesamiento batch de BD: El marco MapReduce

Ejemplo de MapReduce: Seguimiento de Actividad de Usuarios

La función reduce procesa los pares clave-valor de un usuario para encontrar las fechas mínimas y máximas, así como para sumar el número total de comentarios. La fecha mínima se calcula tomando el mínimo entre la fecha mínima actual y la fecha mínima del valor actual. Lo mismo ocurre con la fecha máxima, pero utilizando un operador mayor que. El conteo de cada valor se agrega a una suma acumulativa. Una vez que se han determinado las fechas mínimas y máximas para todos los datos de entrada, el conteo final se convierte en nuestro valor de salida. La clave de entrada se guarda en el sistema de archivos junto con el valor de salida, es decir, $\langle k_3; v_3 \rangle$.

Procesamiento batch de BD: El marco MapReduce

MapReduce



Tarea Map

$\langle k_2, v_2 \rangle$

| Clave | Mín | Max | Cuenta |
|---------|-----|-----|--------|
| 1856 | 7 | 7 | 1 |
| 1856 | 10 | 10 | 1 |
| 1856 | 23 | 23 | 1 |
| 2341839 | 12 | 12 | 1 |
| 2341839 | 3 | 3 | 1 |

Procesamiento batch de BD: El marco MapReduce

MapReduce



Tarea Reduce

Input de la función *reduce*

| | | Clave | Mín | Max | Cuenta |
|------------------|---|---------|-----|-----|--------|
| $< k_2, [v_2] >$ | { | 1856 | 7 | 7 | 1 |
| | | 1856 | 10 | 10 | 1 |
| | | 1856 | 23 | 23 | 1 |
| | | 2341839 | 12 | 12 | 1 |
| | | 2341839 | 3 | 3 | 1 |

Procesamiento batch de BD: El marco MapReduce

MapReduce



Tarea Reduce

Itera la **función *reduce***
para cada clave

$$\text{minActual} \leftarrow \min\{\text{minActual}, \text{value}\}$$
$$\text{maxActual} \leftarrow \max\{\text{maxActual}, \text{value}\}$$
$$\text{Cuenta} \leftarrow \text{Cuenta} + 1$$

Procesamiento batch de BD: El marco MapReduce

MapReduce



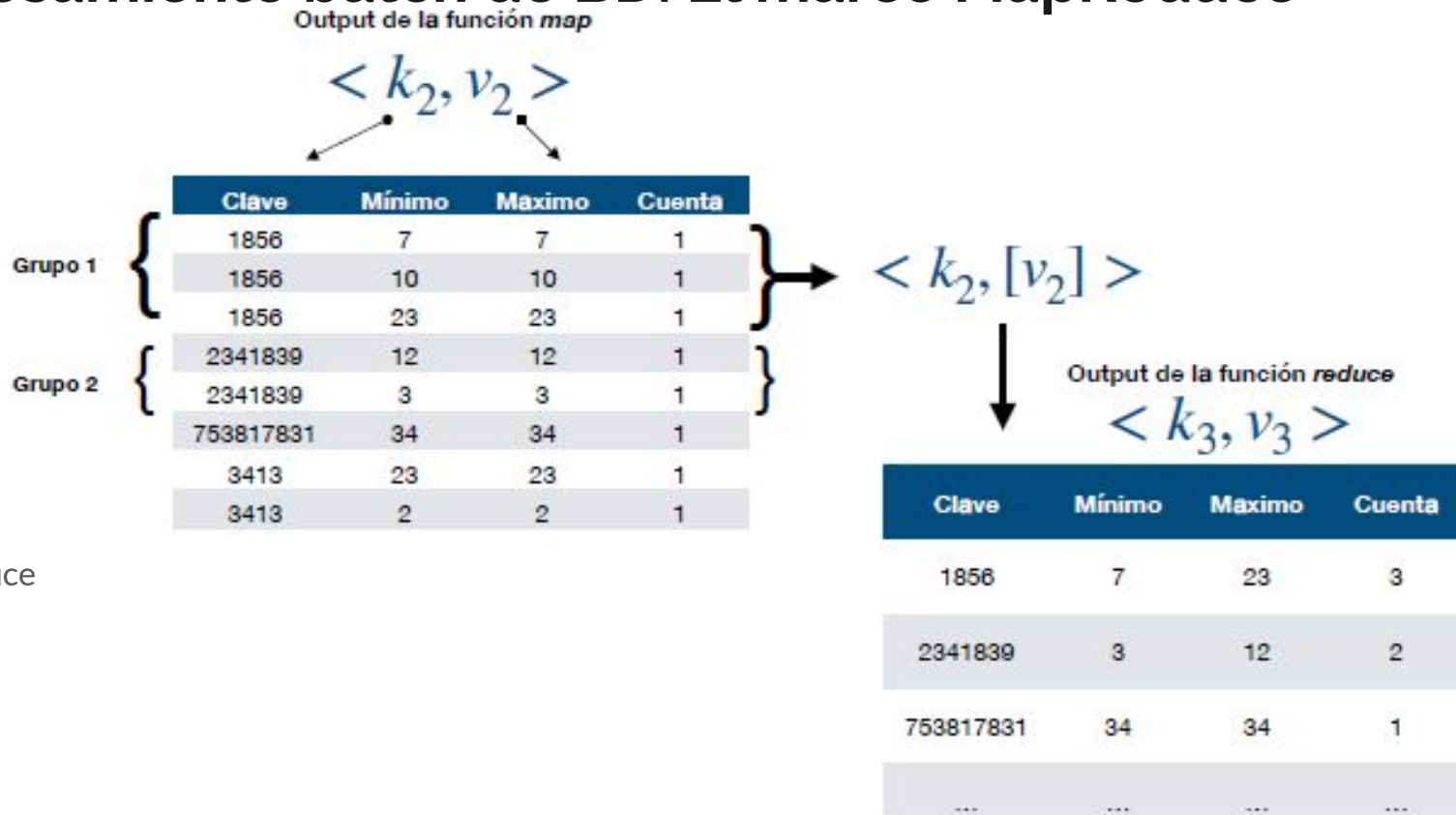
Tarea Reduce

Procesando . . .

| Clave | Min | Max | Cuenta |
|-------|-----|-----|--------|
| 1856 | 7 | 7 | 1 |
| 1856 | 10 | 10 | 1 |
| 1856 | 23 | 23 | 1 |

$\langle k_3, v_3 \rangle = [1856, 7, 23, 3]$

Procesamiento batch de BD: El marco MapReduce



Ejemplo MapReduce

Figura 3.4: Ejemplo de MapReduce.



Procesamiento stream de BD

En este sistema BD los datos llegan continuamente a la capa de análisis y son procesados inmediatamente. Estos sistemas BD no tienen limitaciones de tiempo en el procesamiento del flujo de datos, ni tampoco existe una exigencia en cuanto al tiempo de generación del output por cada input recibido en el sistema.

Los sistemas BD streaming deben disponer de suficiente memoria para almacenar entradas en cola, además la tasa media de procesamiento debe ser mayor que la tasa media de entrada al sistema para que la cola de procesamiento (necesidades de almacenamiento) no creciese hasta el infinito.

Ejemplos de este tipo de plataformas BD son Apache Storm, Flink o Spark Streaming.



Procesamiento stream de BD

Por ejemplo, un sistema diseñado para el recuento de tweets conteniendo ciertas palabras clave es un sistema procesamiento stream donde los datos se generan dinámicamente y en ciertos momento de alta intensidad de tráfico la respuesta se puede ralentizar e incluso el sistema, si fuese necesario, puede descartar un número de tweets.



Procesamiento en tiempo real BD

Los sistemas BD en tiempo real, a diferencia del procesamiento stream, si imponen limitaciones al tiempo de respuesta, que en muchos casos es inferior a unos pocos segundos.

Una cuestión esencial a analizar en un sistema BD en tiempo real es que consecuencias tiene el sobrepasamiento de la capacidad del sistema BD. Si el sistema BD pierde la entrada puede hacer caso omiso de la pérdida y seguir procesando sin detenerse. En según qué aplicaciones, esto no supone un problema crítico pero en otros sistemas esto no es admisible.

Por ejemplo, un sistema de control del tráfico ferroviario no puede en ningún momento perder el control del sistema por motivos de seguridad.



Procesamiento en tiempo real BD

Las técnicas de procesamiento en tiempo real se pueden agrupar en dos categorías:

- i) Computación en memoria cuyo objetivo de estas técnicas es reducir el coste de ejecución de sistemas basados en MapReduce, consiguiendo ejecución de ciertos trabajos en menos de segundos.
- ii) consultas en tiempo real sobre datos estructurados y no estructurados.



Procesamiento en tiempo real BD

La lentitud de ciertos sistemas basados en un esquema MapReduce proviene por dos hechos fundamentales:

- i) los procedimientos para el inicio de la ejecución de los procesos no están optimizados y
- ii) se almacenan grandes bloques de datos sobre discos duros. La tecnología actual permite tasas de lectura sobre estos dispositivos entre 100 y 200 megabytes por segundo.

Se debe poner de manifiesto que la computación en memoria no significa que la base de datos completa sea almacenada en memoria. Estos sistemas almacenan en memoria los datos que son usados frecuentemente en el trabajo iterativo.



Procesamiento en tiempo real BD

Las consultas en tiempo real en sistemas BD hace referencia a un funcionamiento interactivo, permitiendo respuestas en segundos o minutos. En procesamiento batch se admiten respuestas en horas o días.

La primera solución en este campo fue Dremel desarrollada por Google. Esta herramienta para conseguir realizar consultas en tiempo real sobre BD usa dos técnicas fundamentales:

- i) un formato novedoso columnas para estructuras anidadas
- ii) un algoritmo de agregación escalable para computación paralela.

Procesamiento. Cálculos

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 = $10^6 / 20$

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

Procesamiento. Cálculos.

Procesamiento en tiempo real



Motivación

- ¿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 = $10^6 / 20$

$$\text{Tiempo} = \frac{10^6}{20 \cdot 10^4} = 5 \text{ s}$$



Procesamiento. Coste

Procesamiento en tiempo real

Computación
en memoria

- MapReduce
- Consultas en tiempo real

Precio memoria RAM 1TB= 20000 \$





Comparativa stream vs tiempo real

| Aspecto | Procesamiento en Stream de Bases de Datos | Procesamiento en Tiempo Real |
|-------------------------|--|--|
| Naturaleza de los datos | Fluyen en secuencias continuas y sin fin. | Se enfoca en eventos que ocurren instantáneamente. |
| Almacenamiento de datos | Los datos pueden almacenarse durante un tiempo definido o hasta que se procesen. | Los datos generalmente no se almacenan a largo plazo, se procesan inmediatamente. |
| Retraso | Puede introducir cierto retraso en el procesamiento para agrupar datos en bloques más grandes. | Se caracteriza por su baja latencia, las decisiones se toman en milisegundos o menos. |
| Ejemplos de uso | Análisis de registros de servidores en tiempo real, monitoreo de sensores, seguimiento de eventos en redes sociales. | Control de procesos industriales, detección de fraudes, procesamiento de transacciones financieras en tiempo real. |

Plataformas para sistemas BD



Figura 3.5: Soluciones para implementar sistemas BD.



Plataformas para sistemas BD. Apache Hadoop

Apache Hadoop es una de las tecnologías más populares para el procesamiento batch de BD. Es una plataforma de software de código abierto para almacenar datos y ejecutar aplicaciones en un cluster de ordenadores.

Hadoop implementa el módulo Hadoop distributed file system (HDFS) que es esencial para la escalabilidad del sistema. HDFS es un sistema de archivos distribuidos que permite de forma transparente a los programadores almacenar un archivo de datos en diferentes máquinas de un cluster de Hadoop. Cada fichero que se almacena en HDFS se divide en bloques y es cada uno de estos bloques lo que se replica en diferentes nodos del cluster. HDFS permite escalar de una máquina a miles de ellas.



Plataformas para sistemas BD. Apache Hadoop

Se ha creado un ecosistema alrededor de Hadoop con productos como Apache Hive, Apache Pig, Apache Zookeeper, Apache Sqoop, entre otros.

Por ejemplo, Apache Sqoop permite interactuar entre las tradicionales bases de datos relacionales con Hadoop. Sqoop permite leer datos desde una base de datos relacional y escribir dichos datos en HDFS y viceversa.

Además de poder emplear base de datos relacionales en lugar de emplear HDFS, existen plug-in para la entrada y salida de datos que permiten usar bases de datos NoSQL como HBase o Cassandra, HBase permite almacenar grandes cantidades de datos estructurados o usar Mahout como herramienta de aprendizaje automático.



Plataformas para sistemas BD. Apache Hadoop

Tiene otras características propias como definición de tareas más complejas empleando YARN (Yet Another Resource Negotiator). El módulo YARN desacopla la gestión de los recursos de la planificación de las tareas. Por un lado, el denominado ResourceManager se encarga de la gestión de los recursos a nivel global dentro del cluster y el ApplicationMaster planifica las tareas por cada una de las aplicaciones que se ejecutan dentro del cluster.



Plataformas para sistemas BD. Apache Hadoop

Algunas de las limitaciones de Hadoop son:

- Bajo rendimiento para la computación en línea e iterativa. Hadoop no permite la recursividad ni procedimientos interactivos. Su concepción batch hace que todos los datos deban estar preparados antes de iniciar el trabajo. El coste computacional para iniciar un proceso, copiando los códigos y planificando la ejecución, lo limita para un uso iterativo o de consultas en tiempo real. Existen en el mercado extensiones del marco MapReduce que permiten iteración y recursión como Twister o HaLoop.

- Bajo rendimiento de computación. El uso intensivo que hace del disco duro ralentiza su procesamiento y además su diseño es inadecuado para el cálculo en memoria.

- Baja capacidad de intercomunicación.

Plataformas para sistemas BD. Apache Hadoop

Plataformas
BD



Características

- *Apache Software Foundation (Código abierto)*
- Procesamiento **batch**
- **MapReduce.** Capacidad de procesar gran cantidad de datos

Plataformas para sistemas BD. Apache Hadoop

Plataformas BD



Características

- *Apache Software Foundation*
(Código abierto)
- Procesamiento **batch**
- **MapReduce.** Capacidad de procesar gran cantidad de datos
- Escalabilidad
- Tolerancia a fallos
- Flexibilidad

Plataformas para sistemas BD. Apache Hadoop

Plataformas BD



Retos

- No **recursividad** ni **interactividad**
 - Coste de **inicio**
 - **Múltiples** fases Map/Reduce
 - Múltiples **archivos**
 - Falta **programadores**
 - Conocimientos **Java**
 - **SQL+ Hadoop**
 - **Administración** arte y ciencia
 - **Seguridad** de datos
 - **Gestión** y **gobierno** de los datos

Plataformas para sistemas BD. Apache Hadoop

Plataformas
BD



Módulos

- **HDFS** (*Hadoop distributed file system*)
 - Sistema archivos **distribuido**
 - **Replicación bloques**
- **YARN** (*Yet Another Resource Negotiator*)
 - *ResourceManager*
 - *ApplicationMaster (tareas MapReduce)*



Plataformas para sistemas BD. Apache Spark

Framework de programación de código abierto usado para el procesamiento batch, streaming y tiempo real de datos distribuidos. Permite cargar los datos en memoria y consultarlos repetidamente, lo que lo convierte en una herramienta muy adecuada para el procesamiento iterativo y en línea (especialmente para los algoritmos de aprendizaje automático). El desarrollo de Apache Spark fue motivado por las limitaciones del paradigma MapReduce/Hadoop que obliga a seguir un flujo de datos lineal y con un uso intensivo del disco.

Debido a su computación en memoria, tiene una gran velocidad en el procesamiento de datos con respecto al marco Apache Hadoop. Las tareas ejecutadas con Hadoop duran horas o incluso días (dependiendo del volumen de datos a tratar y de la complejidad del algoritmo) mientras que este tiempo es reducido drásticamente si se emplea Spark.

Plataformas para sistemas BD. Apache Spark

- El motor Spark, también conocido como Spark Core (ver figura 3.6)
- El proyecto de código abierto Apache Spark (ver figura 3.7), que es un término paraguas para Spark

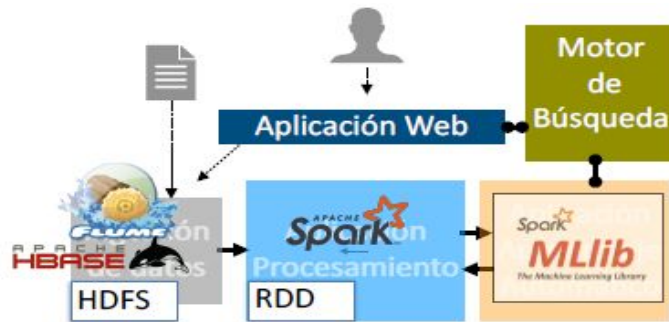


Figura 3.6: Motor Spark en una arquitectura BD.

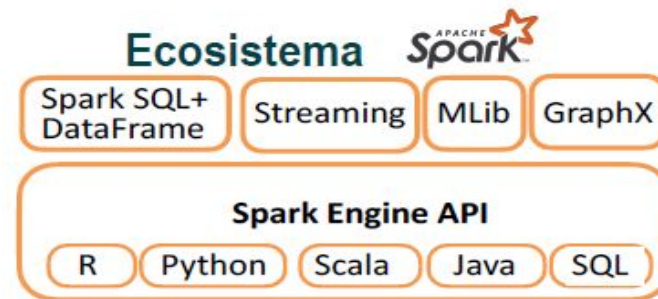


Figura 3.7: Ecosistema Spark.

Plataformas para sistemas BD. Apache Spark

El proyecto de código abierto Apache Spark término paraguas para Spark Core y las aplicaciones Spark que lo acompañan, aplicaciones como:

- Spark SQL (para procesamiento de datos estructurado)
- Spark Streaming (para el procesamiento de datos en streaming)
- Spark MLlib (para la ejecución iterativa de métodos de aprendizaje automático)
- Spark GraphX (para el procesamiento gráfico).

Las diferentes APIs para Python, Scala, Java, SQL o R le permiten a los programadores de Spark eliminar la complejidad que requiere Hadoop. Esto acelera la velocidad de la construcción de aplicaciones.

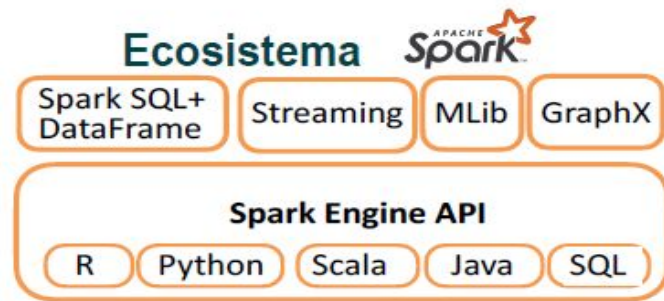


Figura 3.7: Ecosistema Spark.



Plataformas para sistemas BD. Apache Spark

Spark emplea una abstracción para definir la estructura distribuida de los datos denominada **Resilient Distributed Datasets (RDD)** que pueden ser considerada como estructuras de datos paralelos e inmutables. Los RDDs son tolerantes a fallos ya que se preservan información sobre sus padres y se puede recomponer la partición en caso de fallo.



Plataformas para sistemas BD. Apache Spark

Hay dos formas de crear RDDs:

- paralelizando una colección existente en el programa principal: si tenemos una lista de números en Python y desemos realizar operaciones de transformación o análisis en paralelo, puedes convertir esta lista en un RDD. A partir de ese punto, se podrá aprovechar las capacidades de distribución y tolerancia a fallos de Spark para procesar esa colección de datos en un clúster de computadoras.
- referenciando un conjunto de datos en un sistema de almacenamiento externo, como HDFS: si tenemos un conjunto de datos masivo almacenado en HDFS, podemos crear un RDD que lo represente sin tener que traer todos los datos a tu programa local. Spark dividirá automáticamente el conjunto de datos en particiones y las distribuirá en el clúster para su procesamiento paralelo.



Plataformas para sistemas BD. Apache Spark

Los RDDs están diseñados para contener enormes cantidades de datos , que no caben en una sola máquina por lo tanto, los datos tienen que ser particionados a través de múltiples máquinas/nodos.

- Una partición es la unidad atómica (indivisible) de paralelismo en Apache Spark que se almacena en un nodo del cluster.
- Los RDDs en Apache Spark son una colección de particiones.
- Spark particiona automáticamente los RDDs y distribuye las particiones entre diferentes nodos.
- Las operaciones a realizar sobre los RDDs colocan automáticamente las tareas en las particiones, manteniendo la ubicación de los datos persistentes. Los resultados intermedios pueden persistir en la memoria o en el disco para ser reutilizados. También se puede personalizar la particiones para optimizar la colocación de los datos.

Plataformas para sistemas BD. Apache Spark

La figura 3.8 muestra como Spark Streaming procesa los datos en streaming. El primer paso es dividir el flujo de datos en porciones, denominados micro batch. Estos micro batch se transforman en un conjunto de RDDs denominados Dstream (stream discretizado) que es procesado. Spark lee la información de la fuente a intervalos de tiempo fijo (tiene un periodo mínimo de 0.5 segundos).



Figura 3.8: Procesamiento streaming con Spark.



Plataformas para sistemas BD. Apache Spark

Otro acierto de Apache Spark es su posibilidad para poder ejecutarse sobre distintas plataformas, entre ellas la de YARN de Hadoop, posibilitando el procesamiento streaming que Hadoop carece de forma nativa. En este caso Spark tiene la posibilidad de generar RDD desde diversas fuentes de datos como HDFS y HBase.

Hay que elegir cuidadosamente la plataforma donde ejecutar Apache Spark ya que configurar, desplegar, monitorizar, escalar e introducir protocolos de seguridad en un cluster de Apache Spark puede llegar a ser un proceso complejo.



Plataformas para sistemas BD. Diferencias Hadoop y Spark

Actividad: Realice una comparativa detallada entre Apache Hadoop y Apache Spark, destacando las diferencias y similitudes que abarquen los siguientes aspectos.

Diferencias:

- Modelo de Computación.
- Velocidad de Procesamiento.
- Soporte para Múltiples Lenguajes.
- Abstracción de Datos.
- Recuperación de Fallos.

Características Comunes:

- Escalabilidad.
- Procesamiento Distribuido.
- Ecosistema de Componentes.
- Integración con Almacenamiento.



Plataformas para sistemas BD. Hadoop vs Spark

Modelo de Computación:

- Hadoop se basa en el modelo de procesamiento por lotes, lo que significa que es más adecuado para tareas que involucran grandes conjuntos de datos que se procesan en lotes.
- Spark es un modelo de procesamiento en tiempo real o por lotes, lo que lo hace más versátil y adecuado para una variedad de cargas de trabajo, incluyendo procesamiento en tiempo real.

Velocidad de Procesamiento:

- Hadoop utiliza MapReduce, que implica escribir datos intermedios en disco entre pasos de procesamiento, lo que puede ralentizar el procesamiento.
- Spark es considerablemente más rápido que Hadoop debido a su capacidad de mantener datos en memoria y su capacidad de procesamiento en memoria.

Soporte para Múltiples Lenguajes:

- Hadoop principalmente se integra con Java, aunque hay proyectos como Hadoop Streaming que permiten usar otros lenguajes.
- Spark admite múltiples lenguajes de programación, incluyendo Scala, Java, Python y R, lo que facilita el desarrollo de aplicaciones en lenguajes familiares.



Plataformas para sistemas BD. Hadoop vs Spark

Abstracción de Datos:

- Hadoop utiliza principalmente el sistema de archivos distribuidos HDFS (Hadoop Distributed File System) y su modelo de programación MapReduce.
- Spark utiliza RDDs (Resilient Distributed Datasets), que son abstracciones de datos paralelos e inmutables. También ofrece DataFrames y DataSets, que proporcionan estructuras de datos más ricas y optimizadas.

Recuperación de Fallos:

- Hadoop tiene tolerancia a fallos, pero a veces requiere una reescritura de datos intermedios en caso de fallos.
- Spark es conocido por su recuperación de fallos más eficiente, ya que puede reconstruir particiones perdidas de datos en memoria a partir de la información de sus padres.



Plataformas para sistemas BD. Hadoop vs Spark

Características Comunes:

Escalabilidad: Tanto Hadoop como Spark están diseñados para escalar horizontalmente, lo que significa que pueden manejar grandes cantidades de datos agregando más nodos al clúster.

Procesamiento Distribuido: Ambos marcos distribuyen el procesamiento de datos en múltiples nodos de un clúster, lo que permite el procesamiento paralelo y la mejora del rendimiento.

Ecosistema de Componentes: Tanto Hadoop como Spark tienen ecosistemas ricos de componentes y herramientas relacionadas que facilitan el desarrollo y la gestión de aplicaciones de procesamiento de datos a gran escala.

Integración con Almacenamiento: Ambos marcos son compatibles con sistemas de almacenamiento distribuido como HDFS, lo que permite el acceso a datos almacenados en clústeres.



Plataformas para sistemas BD. Otras tecnologías

Storm fue creada por Twitter para el procesamiento stream en BD. Este entorno se ejecuta sobre una máquina virtual de java. Este entorno permite programar rápidamente aplicaciones que procesen grandes flujos de datos.

S4. Es un sistema para el procesamiento stream distribuido, inspirado por el marco MapReduce para procesar stream de datos. Está implementado por Yahoo mediante Java. El stream de datos alimenta a S4 como eventos.

Kafka permite construir una pipeline de fuentes de datos en tiempo real e integrarla en Spark. Esta herramienta recibe los datos desde kafka y los procesa en tiempo real. Esta herramienta no solo es útil para procesamiento en tiempo real, si no también para desarrollar aplicaciones de procesamiento streaming.



Plataformas para sistemas BD. Otras tecnologías

Apache Flink. Es una solución de código abierto que permite analizar tanto datos en modo batch como en tiempo real. La programación de flink y MapReduce comparten muchas similitudes. Esta plataforma permite un procesamiento iterativo y computación en tiempo real sobre un stream de datos recolectado mediante herramientas como flume o Kafka. Apache flink cuenta con FlinkML que es una librería de aprendizaje automático que cuenta con muchos algoritmos muy eficientes y escalables para aplicaciones BD.

Flume es un servicio distribuido empleado para la recolección de datos en tiempo real, con almacenamiento temporal que permite mover eficientemente grandes cantidades de datos.



Aplicaciones de los sistemas BD

En el proceso de digitalización de una PYME cabe destacar las siguientes herramientas:

- ERP, (acrónimo del inglés Enterprise Resource Planning) es un conjunto de aplicaciones de software integradas, que permiten automatizar las prácticas de negocio relacionadas con los aspectos operativos o productivos de la empresa.
- CRM, (acrónimo de Customer Relationship Management) es una aplicación que permite centralizar en una única Base de Datos todas las interacciones entre una empresa y sus clientes. Esta herramienta recopila la información histórica de las interacciones de los clientes con la empresa. El objetivo es gestionar las acciones comerciales (captación de clientes, fidelización, etc.) a partir de un cuadro de mando detallado.
- Plataformas de venta online. Este sistema de compra y venta de productos y servicios emplea Internet como medio principal de intercambio, permitiendo acceder a más clientes, no tener horarios y reducir los costes. Los aspectos negativos son la falta de confianza de los clientes, productos y servicios no tangibles, competencia, dificultades técnicas en su despliegue y requieren un tiempo para obtener resultados.



Aplicaciones de los sistemas BD

1. Incremento del comercio usando información de la propia empresa.

- a) Búsqueda de nuevas oportunidades de venta a través de segmentación de clientes y venta cruzada de productos
- b) Análisis de clientes e historial de transacción, para identificar clientes potenciales para la contratación de servicios o productos complementarios, o que puedan contratar servicios de mayor valor.
- c) Segmentación de cliente para afinar mensajes de marketing y proporcionar ofertas específicas.
- d) Introducción en los CRM de estrategias de personalización de cliente para establecer lazos más estrechos con clientes, lo cual mejora la lealtad.



Aplicaciones de los sistemas BD

2. Incremento del comercio usando información externa a la empresa.

- a) Análisis de datos de navegación web y hábitos de consumo.
- b) Marketing que explota las redes sociales para maximizar la difusión de los productos y servicios.

3. Mejoras de Procesos.

- a) Automatización y control de procesos.
- b) Introducción de filtros inteligentes para la mejora del comercio electrónico.
- c) Cuadro de Mandos en tiempo real, la información siempre está disponible sin esperas de actualización de los datos
- d) Diseño de productos o servicios.
- e) Análisis de proveedores.



Aplicaciones de los sistemas BD

**Aplicaciones
BD**

Big Data

- recomendación de productos
- optimización de la cadena de suministro
- optimización de precios
- detección de fraudes





Aplicaciones de los sistemas BD

Aplicaciones
BD

Big Data

¿Qué quiere ver una persona?

- Sistema de recomendación
- Diseño de productos (producción de series y películas)

NETFLIX

Aplicaciones de los sistemas BD

**Aplicaciones
BD**

Big Data

- Fidelización de clientes



Aplicaciones de los sistemas BD

Aplicaciones
BD

- Diseño de productos
- Análisis de sentimientos

Big Data



Aplicaciones de los sistemas BD

Aplicaciones
BD

Big Data

- Ubicación de nuevas tiendas

