

Ingeniería para el Procesado Masivo de Datos

Dra. Ana Beatriz Medina Ruiz

HDFS y MapReduce

Tema 2: HDFS y MapReduce

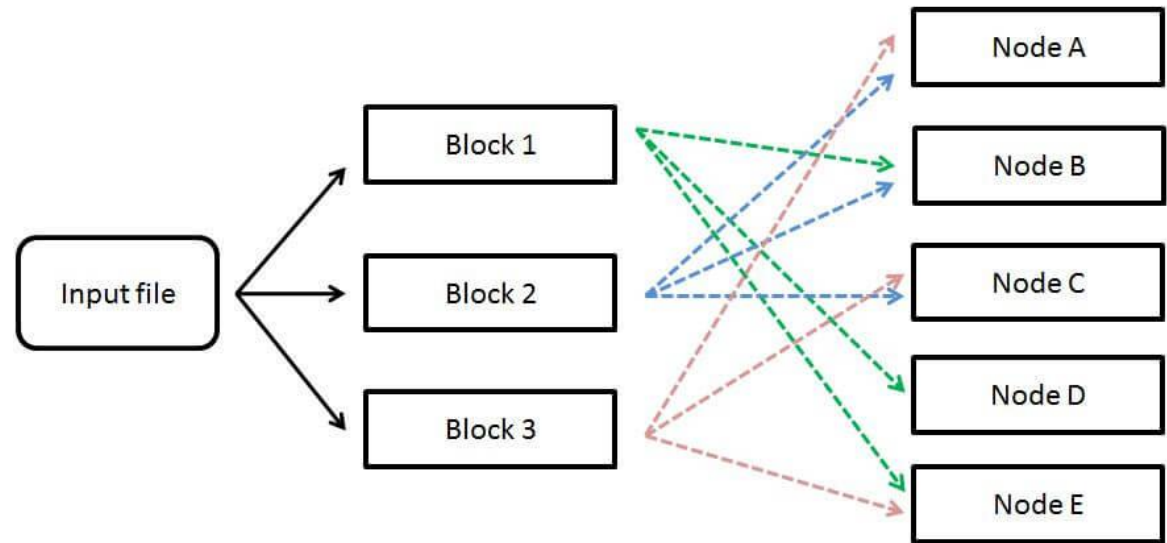
- ▶ 2.1. Introducción a HDFS
- ▶ 2.2. Arquitectura de HDFS
- ▶ 2.3. Comandos de HDFS más frecuentes
- ▶ 2.4. Programación distribuida y MapReduce

Tema 1: Introducción HDFS

HDFS (Hadoop Distributed File System)

Es el sistema de archivos distribuido de Hadoop, diseñado para almacenar grandes volúmenes de datos en un clúster de máquinas.

Se inspira en el Google File System y está optimizado para almacenar archivos grandes que se leen secuencialmente, no para accesos aleatorios frecuentes.



Tema 1: Casos de Uso Práctico de Hadoop

1. Construir una visión comprensiva del cliente

Hoy en día las organizaciones de todos los tamaños interactúan por distintos canales con sus clientes (redes sociales, newsletters , en sus tiendas, visitas personalizadas, etc), pero el comportamiento del cliente es casi totalmente impredecible sin Hadoop.

Hadoop es capaz de almacenar y correlacionar los datos de las transacciones y el comportamiento de navegación en línea, lo que permite identificar las fases del ciclo de vida del cliente para aumentar las ventas, reducir los gastos de inventario y crear una base de clientes leal.

Tema 1: Casos de Uso Práctico de Hadoop

2. Acciones en tiempo real para la toma de decisiones

Cada vez son más las organizaciones se ven con la necesidad de identificar oportunidades en tiempo real, identificar picos que alerten riesgos en sus marcas o incluso los clientes insatisfechos por algún servicio o producto. Con esta premisa además podríamos realizar por ejemplo control de fraudes, control de disponibilidad, entender el comportamiento del cliente, premiarlo con incentivos, identificar problemas de seguridad mediante sensores, etc.

Hadoop posee un gran proyecto llamado Spark (Que también puede ser implementado sin necesidad de Hadoop) donde puedes implementar procesos de análisis en tiempo real vía streaming.

Tema 1: Casos de Uso Práctico de Hadoop

3. Repositorio centralizado de datos

Veámoslo como un DW distribuido un nivel mucho más estratégico donde prima la necesidad de centralizar los datos por ejemplo de todas sus sucursales, tiendas, ventas, etc. minimizando los silos independientes, permitiendo un cross-selling sinérgico, análisis multi-canal, unificación de KPIs y mucho más.

Éste caso se apoya en la capacidad, tanto en almacenamiento como en procesamiento, de utilizar cualquier tipo de dato existente en la organización (o fuera de ella), y con escalabilidad ilimitada.

Hadoop permite, además, abrir los datos a distintos enfoques o tecnologías de procesamiento: predictivos, regresivos, batch, online, MapReduce, SQL, R, SAS, etc. (siempre sobre los mismos datos y sobre la misma plataforma).

Tema 1: Objetivos HDFS (Hadoop Distributed File System)

Escalabilidad horizontal: Permitir el almacenamiento de petabytes de datos mediante la adición de nodos al clúster.

1 Petabyte (PB) = 1,000,000,000,000,000 bytes
(o lo que es lo mismo, 1 cuatrillón de bytes en el sistema decimal)

Tolerancia a fallos: Replicar bloques de datos en múltiples nodos para asegurar la disponibilidad en caso de fallos.

Acceso eficiente: Optimizar el acceso secuencial a grandes archivos, adecuado para procesamiento por lotes.

Big Data, Hadoop, Spark.

Tema 1: **O**bjetivos HDFS (Hadoop Distributed File System)

Costo efectivo: Utilizar hardware común y económico para construir el clúster.

Servidores tipo "commodity" (hardware de bajo costo)

Raspberry Pi (clúster de bajo consumo)

Servidores usados o reacondicionados (data centers o eBay)

Infraestructura de red económica

Ejemplo práctico:

Supón que subes un archivo de 1 GB a HDFS.

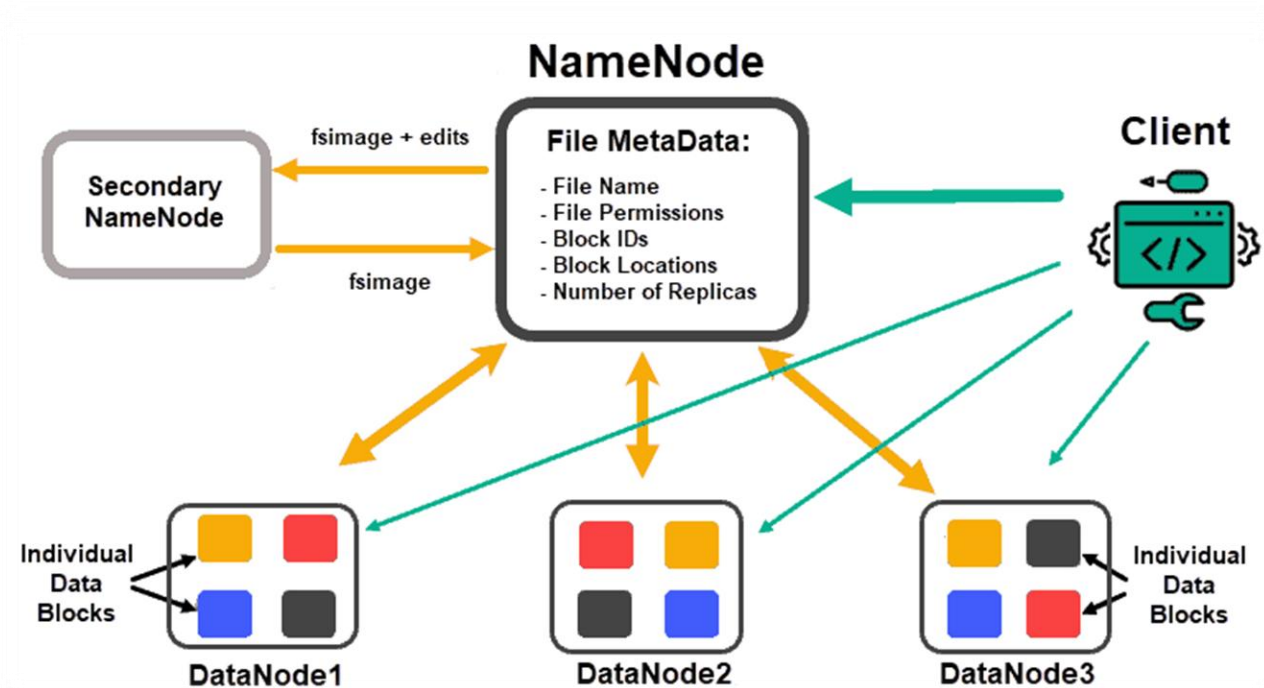
1. HDFS lo divide en 8 bloques de 128 MB.
2. Cada bloque se replica en 3 nodos distintos.
3. Si un nodo que guarda el bloque 2 falla, los otros 2 tienen una copia.
4. Cuando lo lees, el sistema accede a los bloques en paralelo → muy rápido.

Tema 1: Arquitectura a HDFS

HDFS sigue un modelo **maestro-esclavo** y se compone de dos tipos principales de nodos:

NameNode (Nodo Maestro)

•**Función principal:** Gestiona el espacio de nombres del sistema de archivos y coordina el acceso a los archivos por parte de los clientes.



Tema 1: Caso de éxito: Yahoo! y su uso de NameNode con HDFS

Yahoo! fue uno de los primeros grandes adoptantes de Hadoop y HDFS para análisis web masivo.

Implementación:

Usaban un clúster de **más de 40,000 nodos**.

Los archivos de registros de navegación, clics, búsquedas y anuncios se almacenaban en HDFS.

El **NameNode** gestionaba millones de archivos y bloques, distribuidos entre miles de DataNodes.

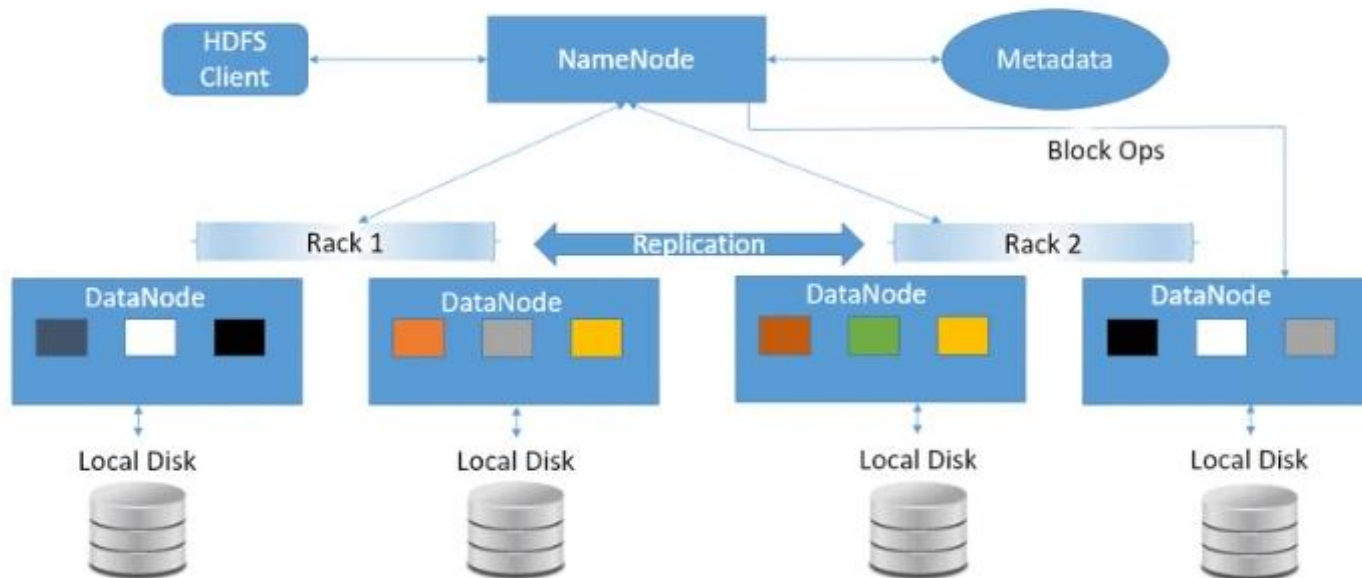
Beneficios logrados:

- El NameNode permitió que cada archivo se dividiera y distribuyera eficientemente sin perder el control.
- Gracias al NameNode, Yahoo! podía **indexar petabytes de datos** y correr miles de trabajos MapReduce cada día.
- HDFS + NameNode ayudaron a optimizar campañas publicitarias en tiempo casi real, gracias al acceso rápido y estructurado a la información.

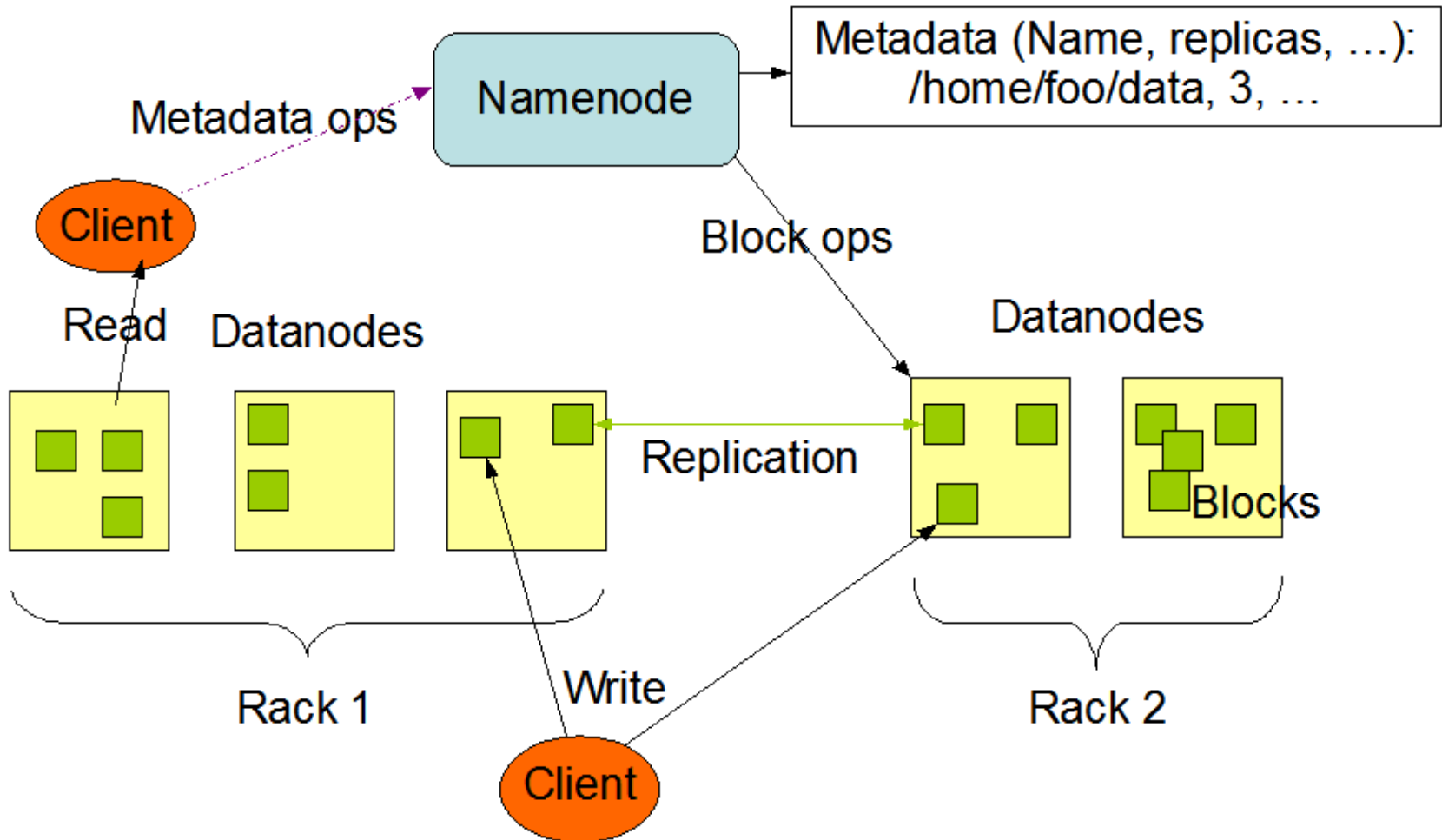
Tema 1: Arquitectura a HDFS

Responsabilidades:

- Almacenan bloques de datos en el disco local.
- Realizan operaciones de lectura y escritura según las instrucciones del NameNode.
- Envían periódicamente información de estado al NameNode para asegurar su correcto funcionamiento.
- Gestionan la replicación y eliminación de bloques según las políticas definidas.



Tema 1: Arquitectura a HDFS



Tema 1: Caso de éxito: Netflix usando HDFS y DataNodes para streaming y análisis

Netflix necesita almacenar y procesar:
Terabytes diarios de logs de usuario.

Información de visualización, comportamiento, calidad de conexión, etc.
Datos analíticos para personalizar recomendaciones.

Uso de DataNodes:

- Netflix usa un clúster basado en Hadoop + HDFS para analizar grandes volúmenes de datos.
- Los **DataNodes almacenan los datos crudos** (eventos de clic, logs, datos de video).
- Miles de nodos trabajan **en paralelo** para procesar la información con Apache Spark y otras herramientas.

Tema 1: Comandos de HDFS más frecuentes

- HDFS puede ser accesado a través del command-line
- Los comandos HDFS tienen el formato:
 - Hdfs dfs – comando
- HDFS a pesar de que no es POSIX, tiene en común algunos de sus comandos como:
 - Cat, chgrp, chmod, chown, cp, du, ls, mkdir, mv, rm, tail.

Hdfs dfs – ls

- Todos los comandos de HDFS usan paths URI como argumentos de sus comandos:
 - Scheme://authority/path
- Scheme para HDFS es hdfs
- Scheme para el file System local es file

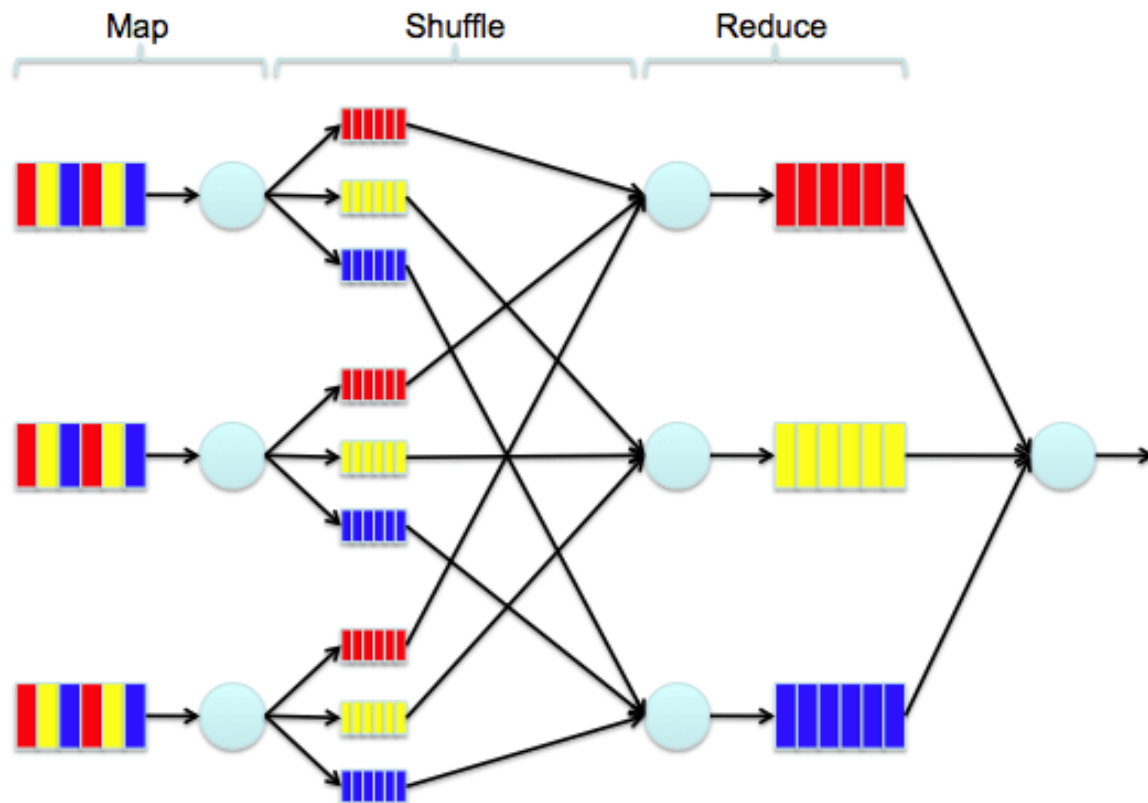
Hdfs dfs – cp file: ///sampleData/Spark/myfile.txt hdfs://rvm.svl.ibm.com:8020/user/sprk/tst/myfile.txt.

Tema 1: Comandos de HDFS más frecuentes

- HDFS tiene además algunos comandos propios:
 - **copyFromLocal:** copia archivos desde el file sistema local a HDFS
 - **get:** Es idéntico a CopyToLocal
 - **copyToLocal:** Copia archivos desde HDFS hacia el file System local.
 - **put:** Es idéntico a CopyFromLoca.
 - **getmerge:** Concatena todos los archivos que se encuentran en u directorio y que cumple con un patrón en un solo archivo.
 - **setrep:** Permite cambiar el número de replicas que se hacen de los bloques de los archivos que se usan como argumento. Recordemos que el factor por defecto es 3.

Tema 1: Programación distribuida y MapReduce

Programación Distribuida es un paradigma que permite desarrollar aplicaciones que se ejecutan en múltiples computadoras (nodos) interconectadas a través de una red. Cada nodo tiene su propia memoria local y puede realizar tareas de forma independiente.



Tema 1: Programación distribuida y MapReduce

MapReduce es un modelo de programación diseñado para procesar grandes cantidades de datos de forma distribuida y paralela. Fue desarrollado por Google y es ampliamente utilizado en el ecosistema de Big Data, especialmente en Apache Hadoop. Este modelo divide el procesamiento de datos en dos fases principales:

- **Map (Mapeo):** Cada nodo procesa una porción de los datos y genera pares clave-valor intermedios.
- **Shuffle and sort (Combinación y Orden):** Aquí se mezclan los resultados de la etapa anterior con todas las parejas clave/valor para combinarlos en una lista y a su vez se ordenan por clave.
- **Reduce (Reducción):** Los pares clave-valor generados se agrupan por clave y se procesan para producir el resultado final.

Tema 1: Caso de éxito: LinkedIn con MapReduce para análisis de datos

LinkedIn necesitaba analizar grandes volúmenes de:

Datos de usuarios.

Interacciones.

Recomendaciones laborales.

Logs de actividad.

Solución:

Implementaron Hadoop + MapReduce para:

- Procesar miles de millones de registros diariamente.
- Construir modelos de recomendación (¿a quién podrías conocer?, empleos sugeridos).
- Detectar tendencias y comportamientos.

Beneficios:

- Reducción drástica del tiempo de procesamiento (de días a horas).
- Aumento en la personalización del contenido mostrado a cada usuario.
- Mejor toma de decisiones con analítica basada en datos masivos.

Tema 1: Caso de éxito empresarial: LinkedIn

LinkedIn es una joya en este ámbito: sus algoritmos de sugerencia de contactos, análisis de actividad profesional y recomendación de empleos descansan en hombros de titanes... es decir, en MapReduce y Hadoop.

¿Qué hizo LinkedIn?

- Procesaba más de 2 petabytes de datos al día con Hadoop/MapReduce.
- Usó programación distribuida para:
- Crear el "People You May Know" (PYMK).
- Realizar análisis de tendencias laborales.
- Detectar anomalías en seguridad y comportamiento de usuarios.





¿Cómo lo logró?

- Dividieron sus enormes datasets en tareas distribuidas con MapReduce.
- Analizaron relaciones y comportamientos de millones de usuarios.
- Optimizaron sus sistemas de recomendación sin saturar servidores individuales.

Resultado:

- Mejor experiencia de usuario, más interacciones, mayor retención.
- Ahorraron costos masivos al no requerir supercomputadoras, solo un clúster de servidores bien orquestado.

Tema 1: RESUMEN

Elemento	Detalle
 Programación distribuida	Divide tareas y las ejecuta en paralelo.
 MapReduce	Modelo de programación para análisis masivo distribuido.
 Ventajas	Escalabilidad, eficiencia, tolerancia a fallos.
 Caso real	LinkedIn usó MapReduce para análisis de comportamiento y personalización de la plataforma.

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

unir

www.unir.net