



UNIVERSIDAD
DE GRANADA



Master Profesionalizante en Ingeniería Informática

Cloud Computing: Servicios y Aplicaciones

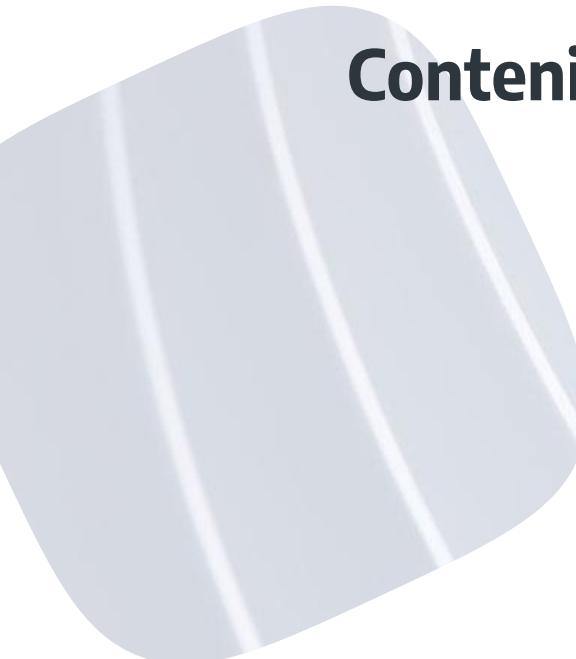


UNIVERSIDAD
DE GRANADA



T5. Aplicaciones de Cloud Computing

5.1. Almacenamiento y procesamiento masivo de datos



Contenido

Concepto de Big Data
MapReduce
Hadoop
HDFS y procesamiento
Spark

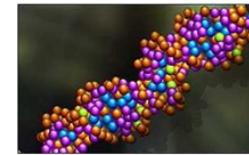


1

Big Data

Un mundo en torno a los datos

Ciencia: Bases de datos de astronomía, genómica, datos medio-ambientales, datos de transporte, ...



Ciencias Sociales y Humanidades: Libros escaneados, documentos históricos, datos sociales, ...



Negocio y Comercio: Ventas de corporaciones, transacciones de mercados, censos, tráfico de aerolíneas, ...

Entretenimiento y Ocio: Imágenes en internet, películas, ficheros MP3, ...



Medicina: Datos de pacientes, pruebas diagnósticas, ...
Industria, Energía, ...: Sensores, ...



Revolución de los datos

Década de los datos

Los datos son el nuevo petróleo



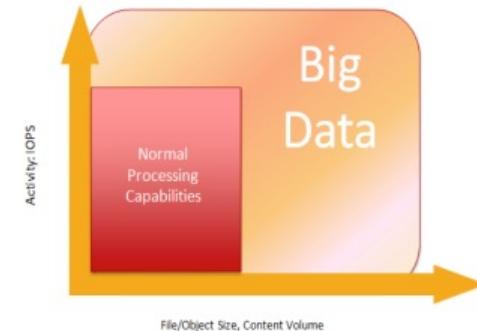
¿Qué es Big Data?



No hay una definición estándar

Big data es una colección de datos grande, complejos, muy difícil de procesar a través de herramientas de gestión y procesamiento de datos tradicionales

“Big Data” son datos cuyo volumen, diversidad y complejidad requieren nueva arquitectura, técnicas, algoritmos y análisis para gestionar y extraer valor y conocimiento oculto en ellos ...





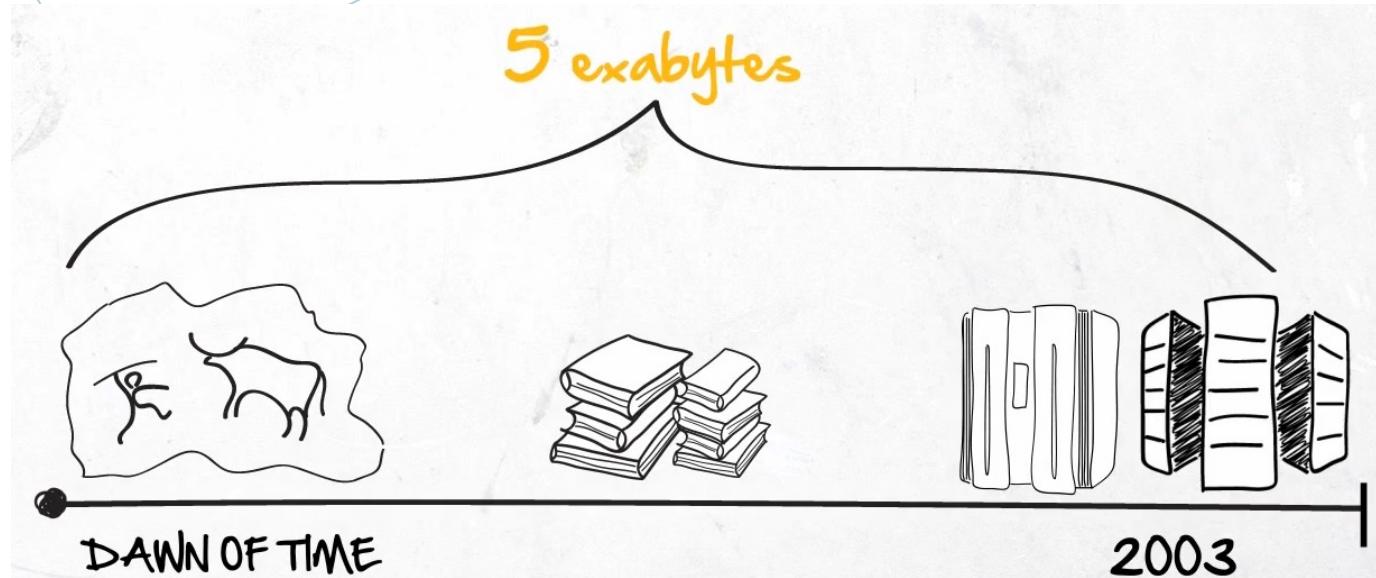
Algunas definiciones ...

“Big Data is the next generation of data warehousing and business analytics and is poised to deliver top line revenues cost efficiently for enterprises”

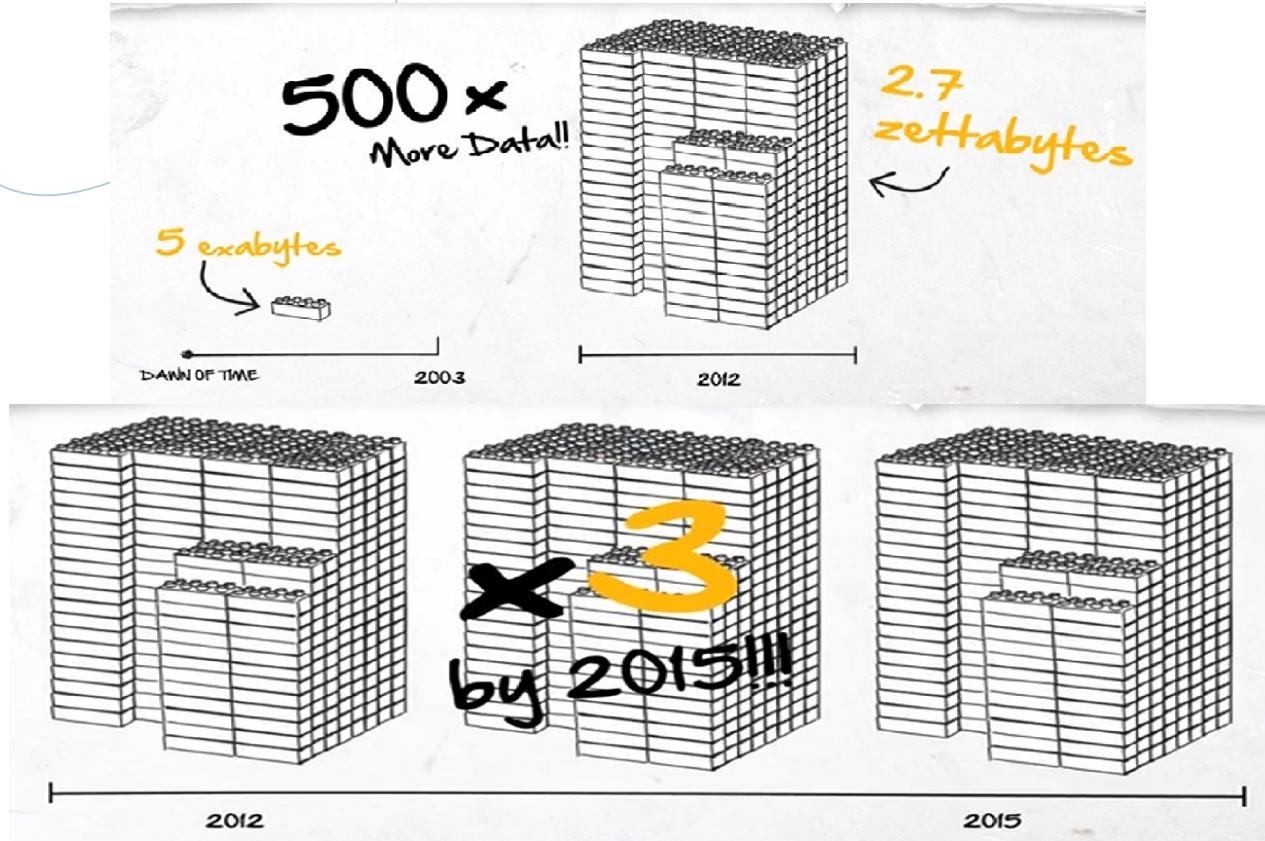
“Big Data exceeds the reach of commonly used hardware environments and software tools to capture, manage, and process it within a tolerable elapse time for its user population”

“Disciplina que se ocupa de todas las actividades relacionadas con los sistemas que manejan grandes conjuntos de datos, principalmente, almacenamiento, búsqueda, compartición, análisis y visualización”

¿Qué es Big Data?



¿Qué es Big Data?



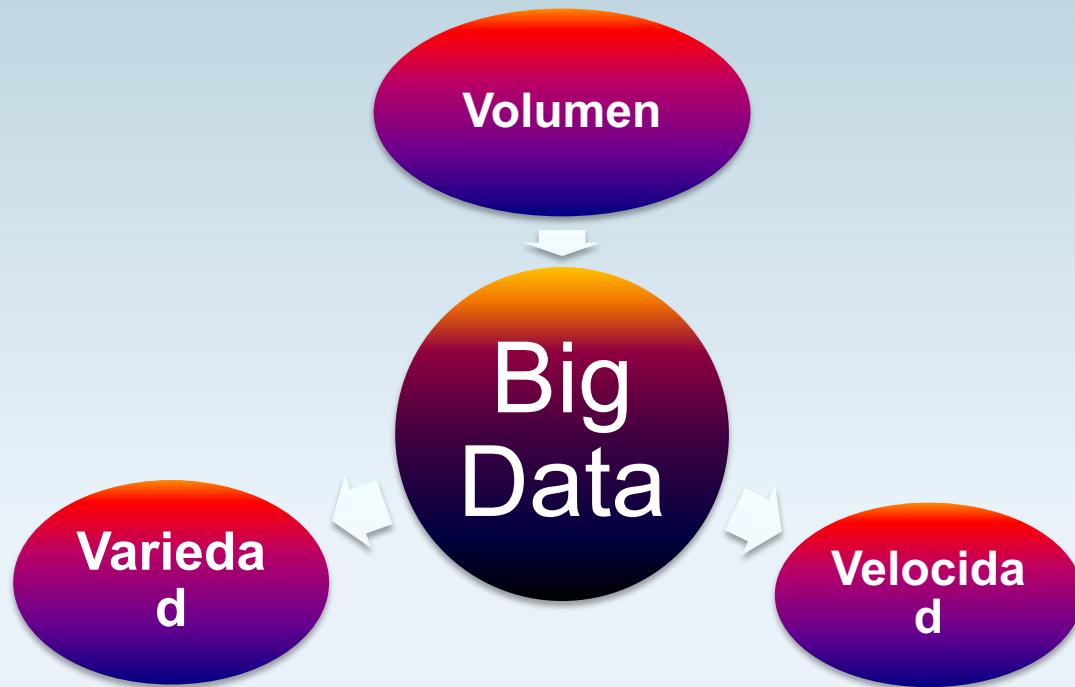
Ley de Moore

“Aproximadamente cada dos años se duplica el número de transistores de un microprocesador” (G.E. Moore, 1965)

Crecimiento en potencia de los equipos

El concepto de **BIG** se desplaza con la tecnología.

Las 3 Vs de Big Data



Volumen

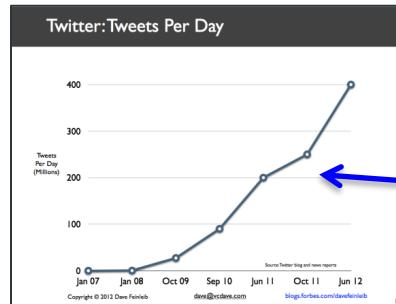
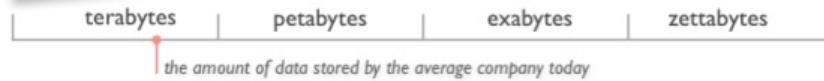


El volumen de datos ...

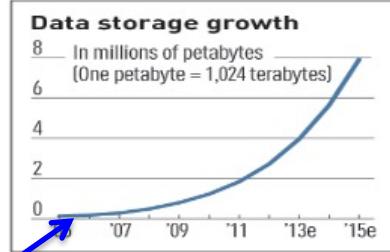
crece exponencialmente

Crecimiento x 44 de
2009 a 2020

De 0.8 zettabytes a 35ZB



The Digital Universe 2009-2020



Crecimiento exponencial en los datos
generados/almacenados

Unidades de información

Kilobyte = 10^3 = 1,000

Megabyte = 10^6 = 1,000,000

Gigabyte = 10^9 = 1,000,000,000

Terabyte = 10^{12} = 1,000,000,000,000

Petabyte = 10^{15} = 1,000,000,000,000,000

Exabyte = 10^{18} = 1,000,000,000,000,000,000

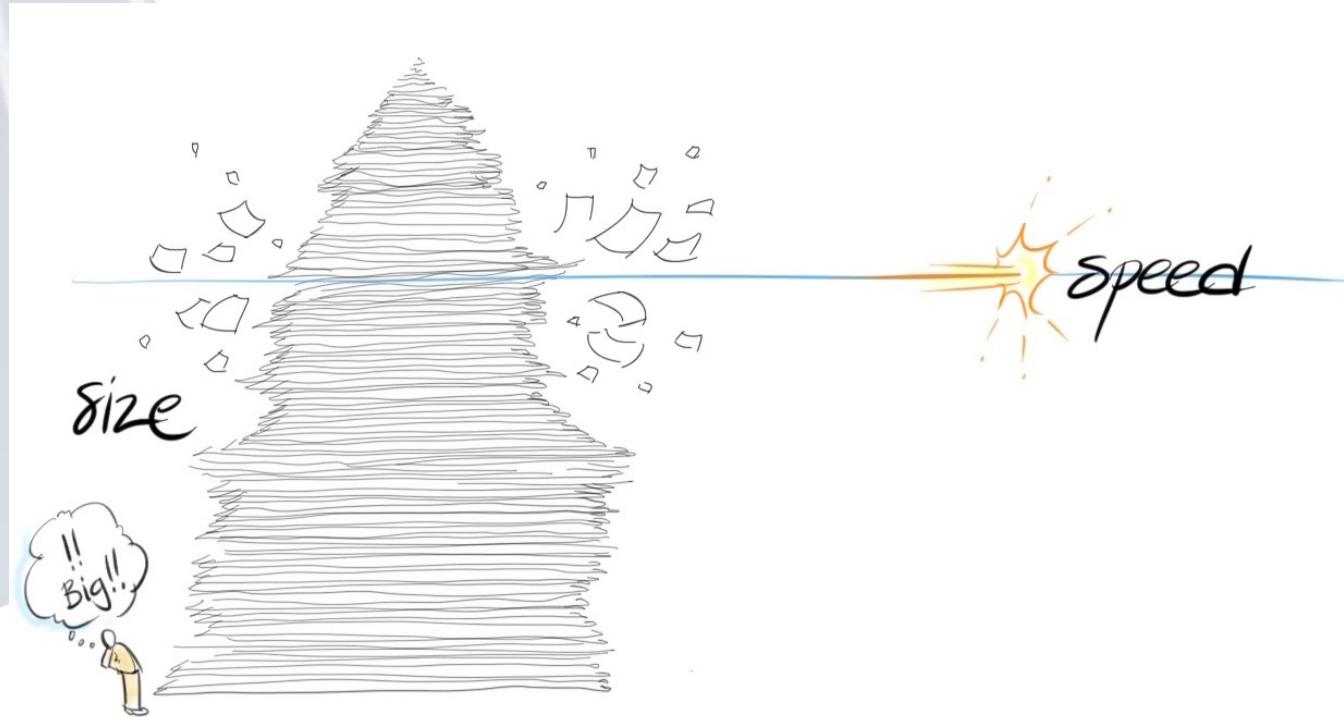
Zettabyte = 10^{21}

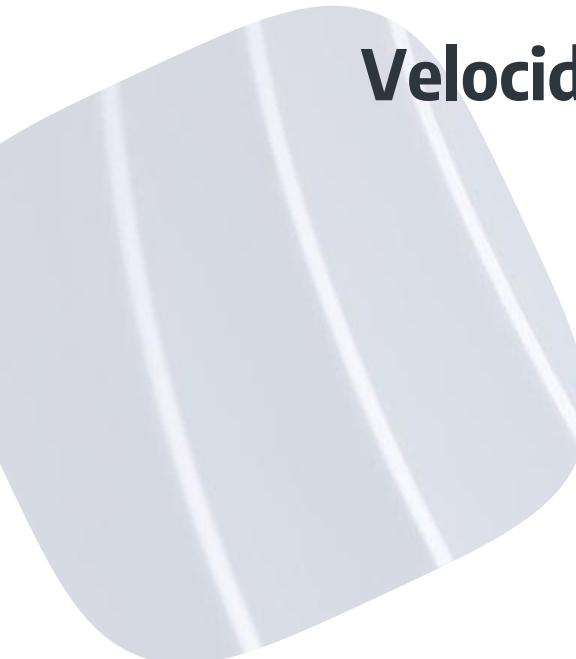
Yottabyte = 10^{24}

Xonabyte = 10^{27}

Hay nombres asignados hasta 10^{63} (luma)

Velocidad





Velocidad

Los DATOS se generan muy rápidamente y necesitan ser procesados rápidamente
Online Data Analytics

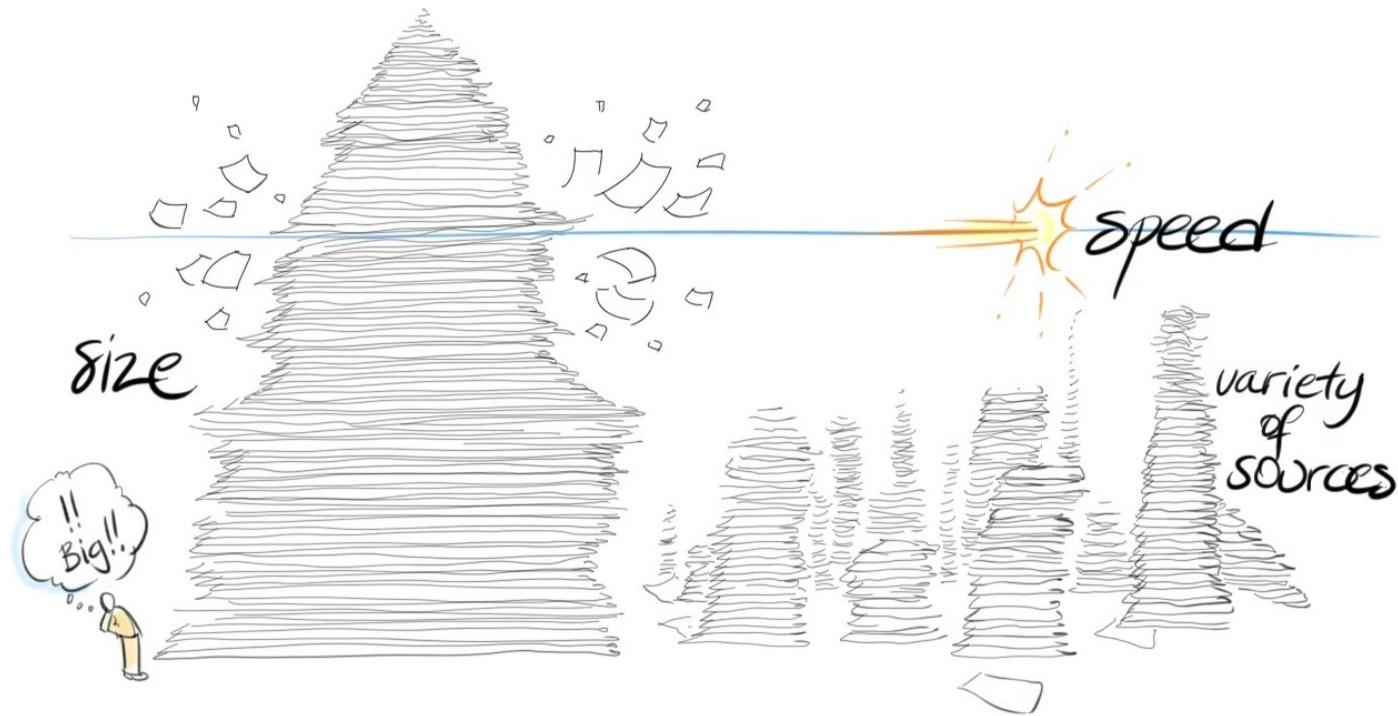
Ejemplos:

Transacciones financieras

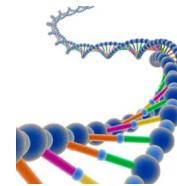
Monitorización industrial

Vigilancia: aduanas, autovías, aeropuertos

Variedad de fuentes

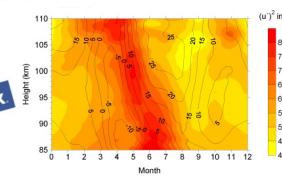
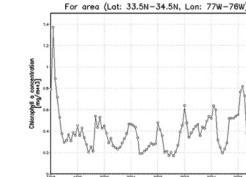
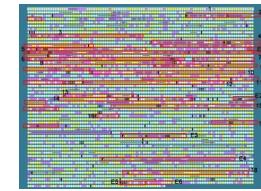


Variedad



Extracción de conocimiento → Análisis
conjunto de esos tipos de datos

Varios formatos y estructuras:
Texto, numéricos, imágenes, audio, video,
secuencias, series temporales, ...
Una sola aplicación puede generar muchos
tipos de datos



Las 8 Vs de Big Data



Es un reto



Big data incluye datos estructurados,
semi-estructurados y no estructurados



¿Quién genera Big Data?



Redes sociales y multimedia
(todos generamos datos)

Social Media



Comercio y transacciones financieras

Instrumentos científicos
(colección de toda clase de datos)



El progreso y la innovación ya no se ven obstaculizados por la capacidad de recopilar datos, sino por la capacidad de gestionar, analizar, sintetizar, visualizar, y descubrir el conocimiento de los datos recopilados de manera oportuna y en una forma **escalable**



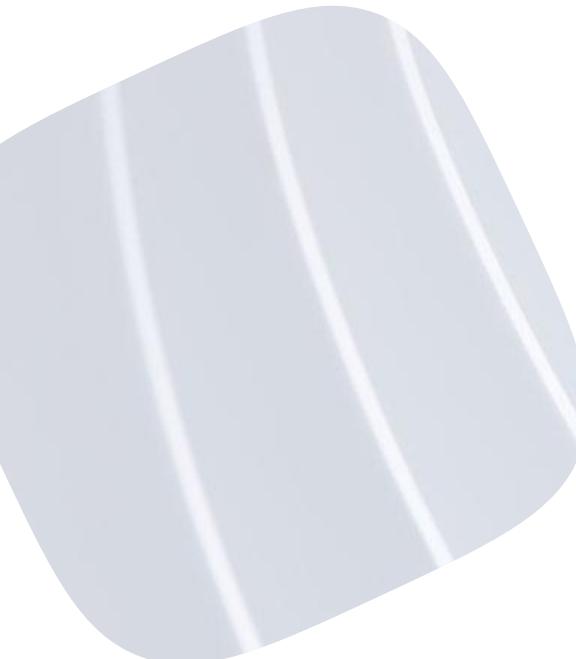
Dispositivos móviles
(seguimiento de objetos)



Redes de sensores
(se miden toda clase de datos)



Fuentes de datos



Big Data Types

Web and Social Media

- Clickstream Data
- Twitter Feeds
- Facebook Postings
- Web Content

Biometrics

- Facial Recognition
- Genetics

Machine-to-Machine

- Utility Smart Meter Readings
- RFID Readings
- Oil Rig Sensor Readings
- GPS Signals

Human Generated

- Call Center Voice Recordings
- Email
- Electronic Medical Records

Big Transaction Data

- Healthcare Claims
- Telecommunications Call Detail Records
- Utility Billing Records

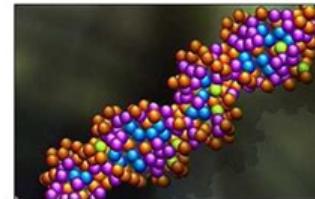
Aplicaciones

Astronomía



- Astronomical sky surveys
- 120 Gigabytes/week
- 6.5 Terabytes/year

Genómica



- 25,000 genes in human genome
- 3 billion bases
- 3 Gigabytes of genetic data

Telefonía



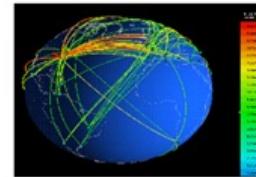
- 250M calls/day
- 60G calls/year
- 40 bytes/call
- 2.5 Terabytes/year

Transacciones de tarjetas de crédito



- 47.5 billion transactions in 2005 worldwide
- 115 Terabytes of data transmitted to VisaNet data processing center in 2004

Tráfico en Internet



- Traffic in a typical router:
- 42 kB/second
 - 3.5 Gigabytes/day
 - 1.3 Terabytes/year

Procesamiento de información WEB

Google™

- 251
- 10k
- 250
- *De tim*



Un par de casos



El gran acelerador de hadrones (del CERN) genera 500 exabytes diarios de datos

El CPD de la NSA en Utah tiene capacidad para 5 Zettabytes



Algunos límites en HDD

HDD con MBR: hasta 2TB. A partir de ahí GPT

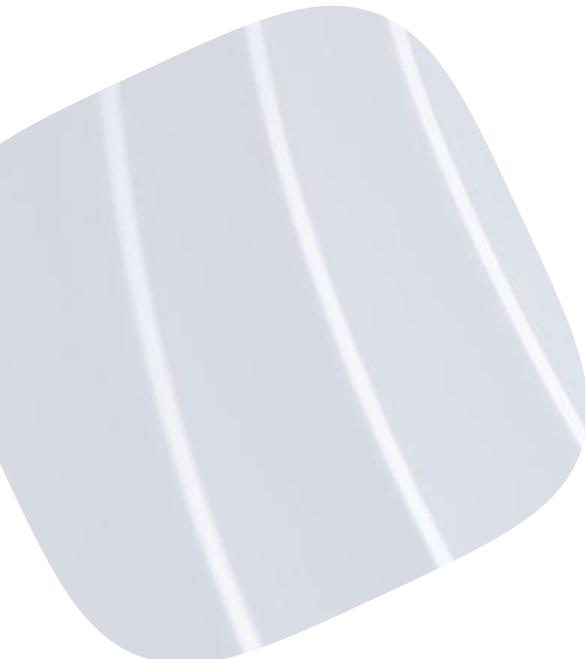
MySQL soporta:

4GB en HDD FAT, o

2TB en NTFS, ext3, HFS+

Ext4 soporta 1024 PB

Big Data: Computación **escalable sin límite**



Da un algoritmo para calcular el mínimo de un conjunto de números (no ordenados)
Búsqueda secuencial
¿Sobre 1PB?



2

Map Reduce

MapReduce



Modelo de programación para dar soporte a la computación paralela de grandes conjuntos de datos sobre clusters de ordenadores

Modelo sencillo, pero **no aplicable a **todo tipo** de problemas**

Sólo problemas descomponibles en Map y Reduce, y con datos representables como pares (clave, valor**)**

J. Dean, S. Gemawhat, “MapReduce: Simplified data processing on large clusters”, Communications of the ACM, 51, 1, 2008, 107—113.

Ejemplo

Problema: Escalabilidad de grandes cantidades de datos

Caso:

Exploración 100 TB en 1 nodo @ 50 MB/sec = 23 días

Exploración en un clúster de 1000 nodos = 33 minutos

Solución → Divide-Y-Vencerás



Una sola máquina no puede gestionar grandes volúmenes
de datos de manera eficiente

Cuando crece el tamaño de los datos ...

¿Qué ocurre cuando el tamaño de los datos aumenta y los requisitos de tiempo se mantienen?

Hay que aumentar los recursos de hardware (número de nodos). Pero *no es suficiente* con añadir más hardware: se tiene que poder **usar eficazmente**.

Se necesitan técnicas y herramientas

Paradigma **MapReduce** (Google)

Modelo de programación: MapReduce

- Modelo de programación de datos paralelo
- Concepto simple, elegante y extensible para múltiples aplicaciones
- **Creado** por Google (2004)
 - Procesaba 20 PB de datos por día (2004)
- **Hadoop**: Versión open-source creada por Doug Cutting at the Yahoo.
- Posteriormente, se convierte en un proyecto de la Fundación Apache (<http://hadoop.apache.org>)

RDBMS vs MapReduce

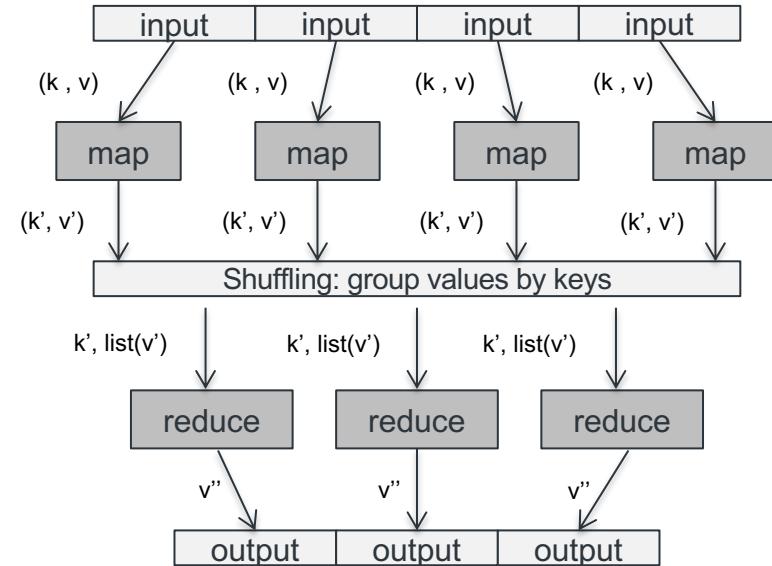
	Traditional RDBMS	MapReduce
Data Size	Gigabytes (<i>Terabytes</i>)	Petabytes (<i>Hexabytes</i>)
Access	Interactive and Batch	Batch
Updates	Read / Write many times	Write once, Read many times
Structure	Static Schema	Dynamic Schema
Integrity	High (ACID)	Low
Scaling	Nonlinear	Linear

Esquema

MapReduce es el entorno más popular para Big Data
 Basado en la estructura {clave, valor}
 Dos operaciones:

1. **Función Map :** Procesa bloques de datos
2. **Función Reduce :** Fusiona los resultados previous de acuerdo a su clave.

Una etapa intermedia de agrupamiento por clave (Shuffling)

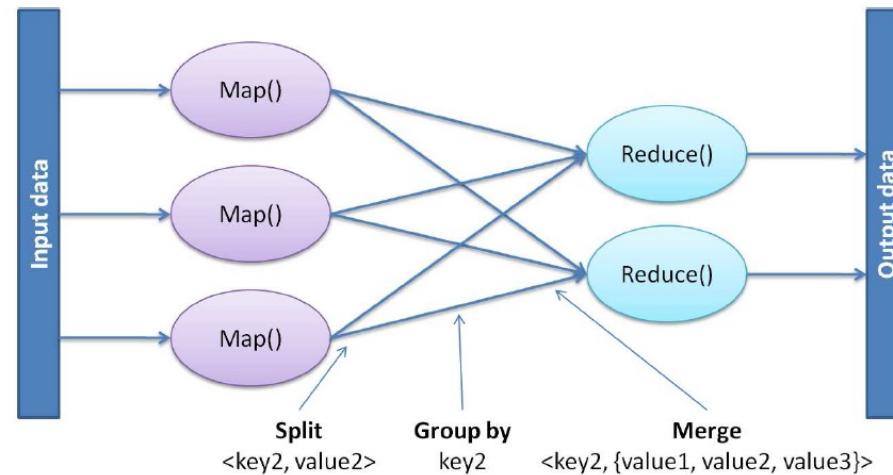


map $(k, v) \rightarrow \text{list}(k', v')$
 reduce $(k', \text{list}(v')) \rightarrow v''$

Flujo de datos MapReduce

map $(k, v) \rightarrow$
 $\text{list}(k', v')$

reduce
 $(k', \text{list}(v')) \rightarrow v''$

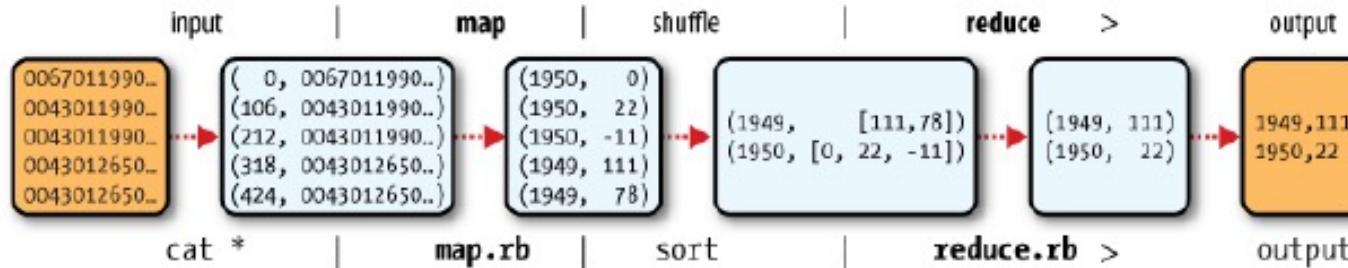


Ejemplo: máximo de temperatura anual

Datos meteorológicos de red de estaciones (EE.UU.)

Datos en líneas de texto

¿Temperatura más alta registrada cada año?



{clave, valor} = {año, temperatura}

Propiedades del modelo (I)

Paralelización automática:

Dependiendo del tamaño de ENTRADA DE DATOS → se crean múltiples tareas MAP

Dependiendo del número de valores intermedios <clave, valor> particiones → se crean tareas REDUCE

Escalabilidad:

Funciona sobre cualquier cluster de nodos/procesadores

Puede trabajar desde 2 a miles de máquinas

Propiedades del modelo (II)

Transparencia programación

Manejo de los fallos del ordenador

Gestión de comunicación entre ordenadores

Tiempo de ejecución:

Almacenamiento (Partición de datos)

Programación de la tarea

Ejecución del proceso

Programación paralela para *dummies*

¿Número de maps?

El número de Maps es independiente del número de nodos disponibles.

Va a estar asociado al tamaño y otras características del problema.

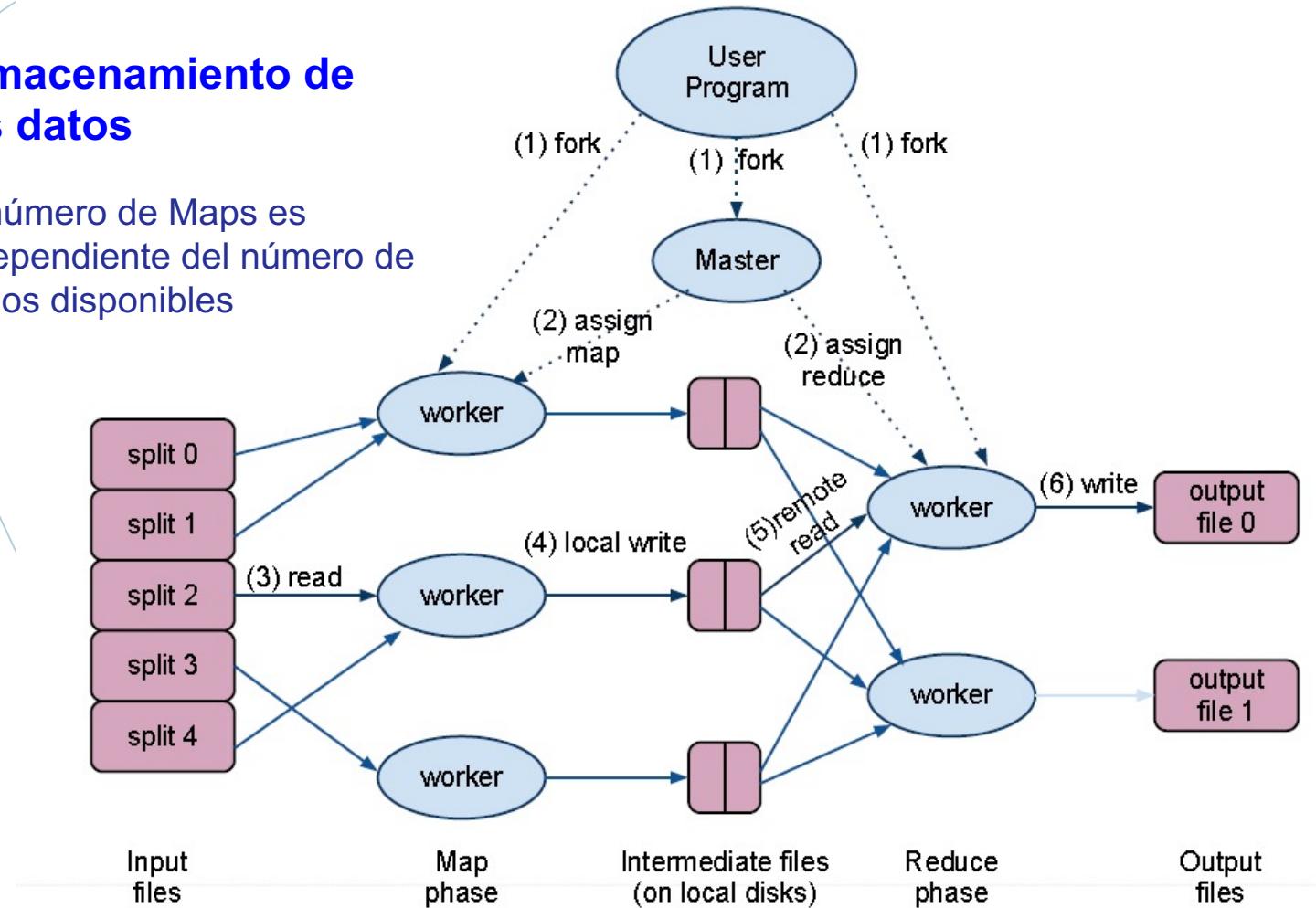


VS



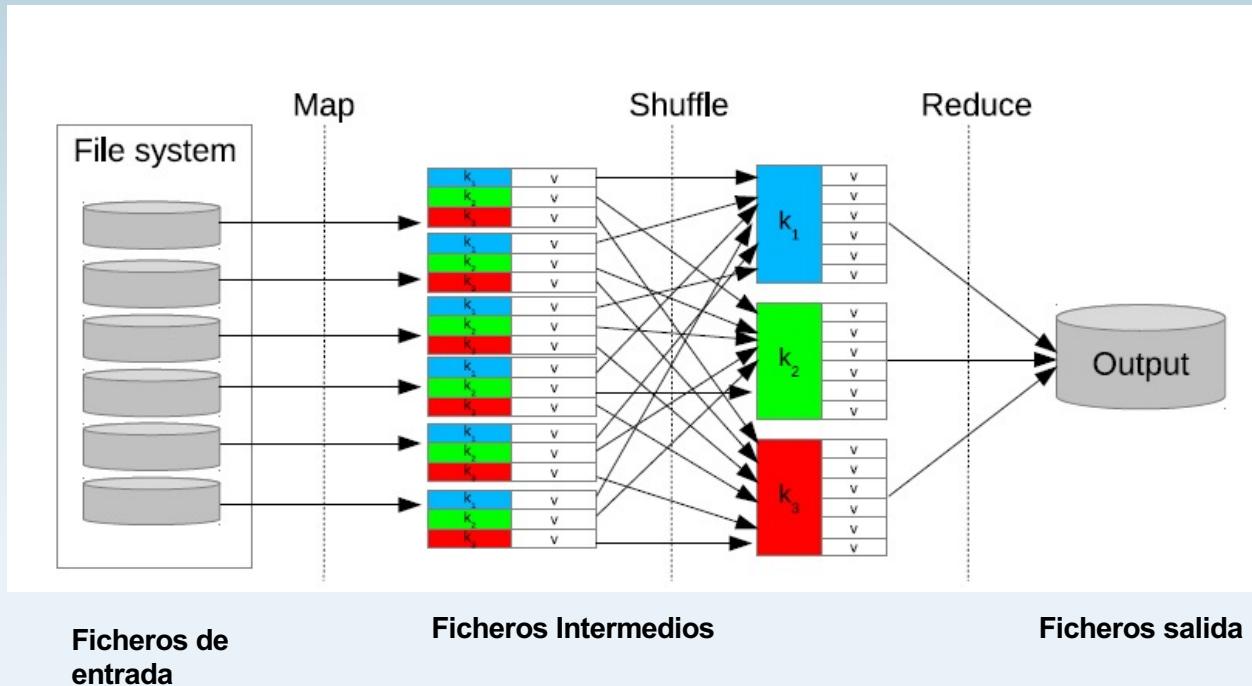
Almacenamiento de los datos

El número de Maps es independiente del número de nodos disponibles



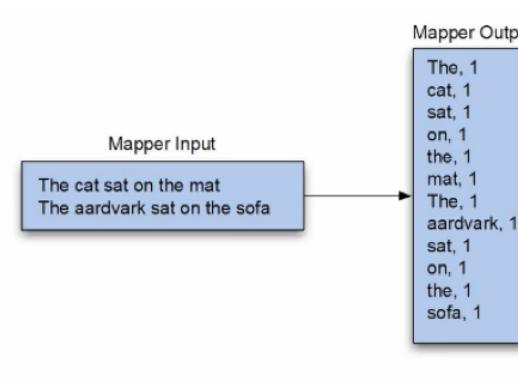
Almacenamiento con copias, normalmente r=3

Flujo de datos, transparente para el programador



Algunas cuestiones: Proceso Map

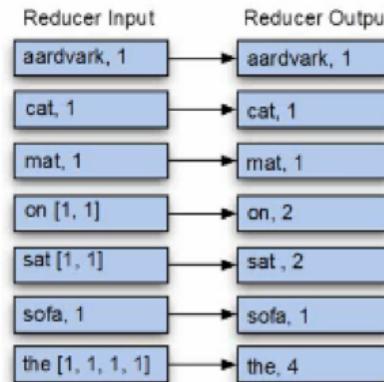
Puede crear conjuntos de datos muy pequeños, lo cual puede ser un inconveniente para algunos problemas.



Ejemplo: Problemas de Clasificación. Nos podríamos encontrar con el problema de falta de densidad de datos y de clases con muy pocos datos (clasificación no balanceada).

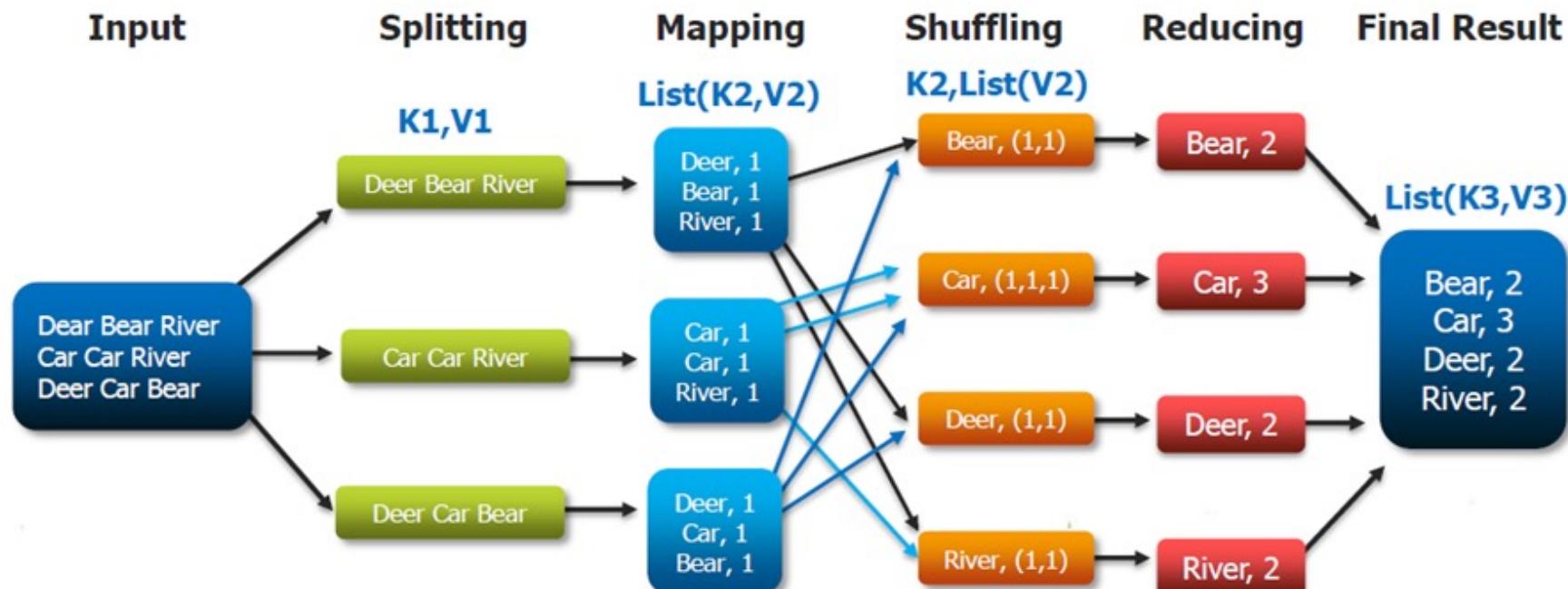
Algunas cuestiones: Proceso Reduce

Debe combinar las soluciones de todos los modelos/procesos intermedios



Esta es la fase más *creativa* porque hay que conseguir crear un modelo global asociado al problema que se desea resolver a partir de los modelos intermedios.

The Overall MapReduce Word Count Process



Ejemplo: Obtener el mínimo de un conjunto de números

Reparto:
k trozos
 $\{x_1, \dots, x_n\} \rightarrow$
 $\{x_1, \dots, x_{n_1}\}$
 $\{x_{n_1+1}, \dots, x_{n_2}\},$
 \dots
 $\{x_{n_{k-1}}, \dots, x_n\}$

Proceso Map:
1. Leer datos del trozo (Split)
2. Calcular mínimo del trozo
3. Salida: $\langle \text{mínimo}, 1 \rangle$

Mezcla (Shuffling):
1. Agrupar por valor:
 $\langle m_1, 1 \rangle$
 $\langle m_2, 1 \rangle$
 $\langle m_k, 1 \rangle$

Proceso Reduce:
Calcular el mínimo de m_1, m_2, \dots, m_k

Resumen

Ventaja frente a los modelos distribuidos clásicos:

El modelo de programación paralela de datos de MapReduce oculta la complejidad de la distribución y tolerancia a fallos

Claves de su filosofía: Es

- **escalable:** se olvidan los problemas de hardware
- **más barato:** se ahorran costes en hardware, programación y administración

MapReduce **no es adecuado para todos los problemas, pero cuando funciona, puede ahorrar mucho tiempo**

3

Hadoop



Hadoop es una
implementación de
código abierto del
paradigma
computacional
MapReduce

<http://hadoop.apache.org/>

Almacenamiento y procesamiento

Consta de dos servicios principales:

Almacenamiento: HDFS.

Procesamiento: MapReduce.



Aporta una serie de ventajas:

Bajo coste: clústeres baratos / cloud computing.

Facilidad de uso.

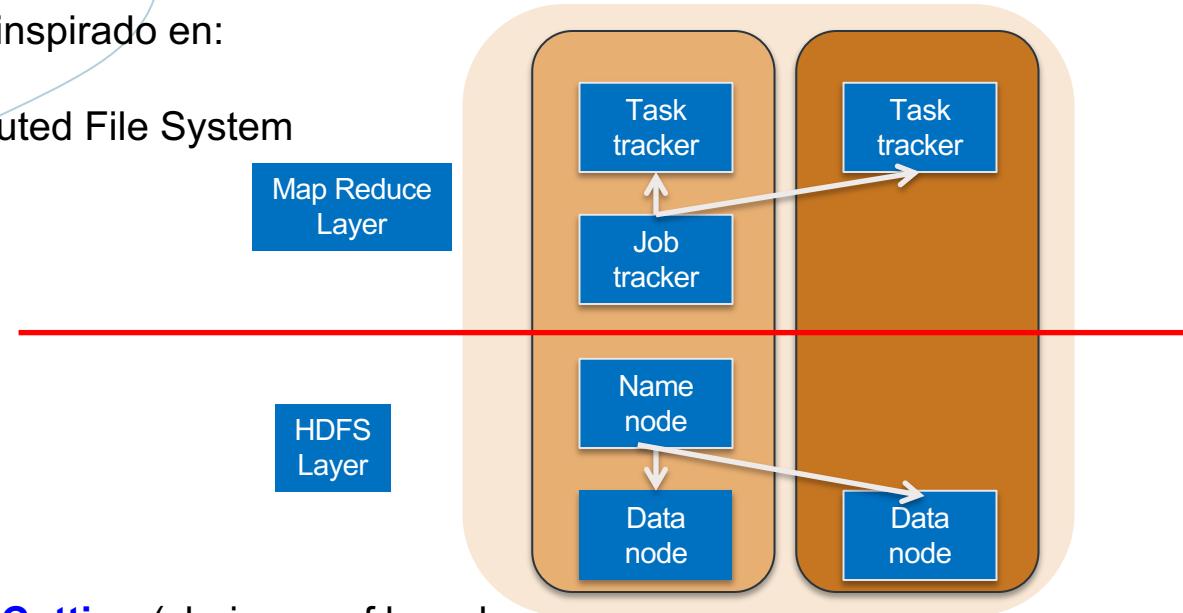
Tolerancia a fallos.

Implementación



Escrito en java e inspirado en:

- MapReduce
- Google Distributed File System



Creado por **Doug Cutting** (chairman of board of directors of the Apache Software Foundation, 2010)

<http://hadoop.apache.org/>



Apache Hadoop



Apache™ Hadoop® es un proyecto que desarrolla software de código abierto fiable, escalable, para computación distribuida

Hadoop se puede ejecutar de tres formas distintas (configuraciones):

- 1. Modo Local / Standalone.** Se ejecuta en una única JVM (Java Virtual Machine). *Útil para depuración*
- 2. Modo Pseudo-distribuido** (simulando así un clúster o sistema distribuido de pequeña escala)
- 3. Distribuido (Clúster)**





Funcionalidad de Hadoop

Gestionar almacenamiento (HDFS) de entradas, pares intermedios y salidas

Particionamiento de datos

Transferencias

Planificación y monitorización de procesos

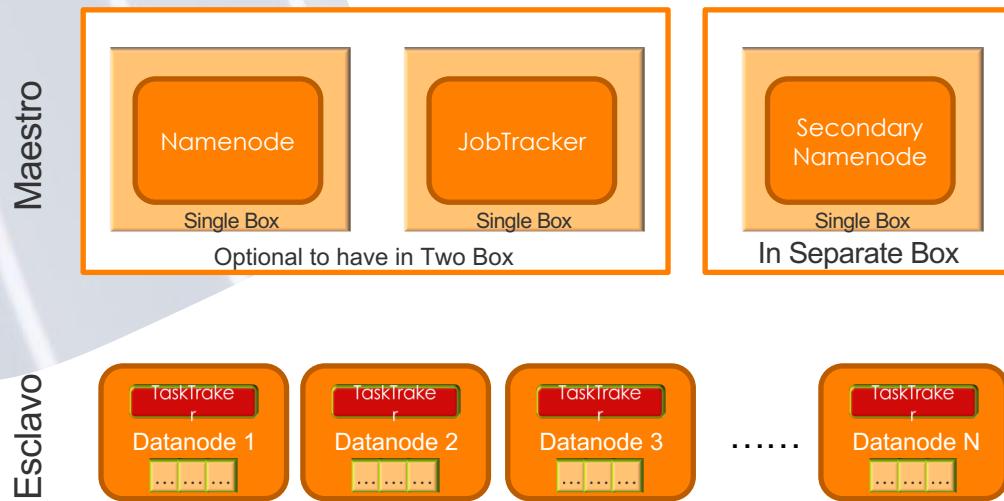
Entorno de ejecución distribuida con bajo conocimiento técnico

Arquitectura

Maestro-Esclavo:

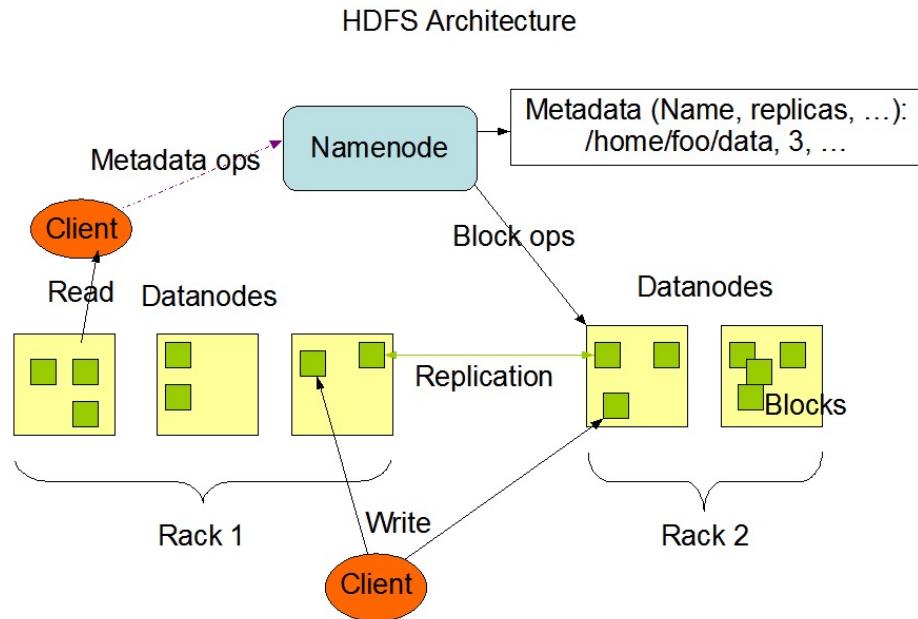
Maestro: NameNode, JobTracker

Esclavo: {DataNode, TaskTracker}, ...,

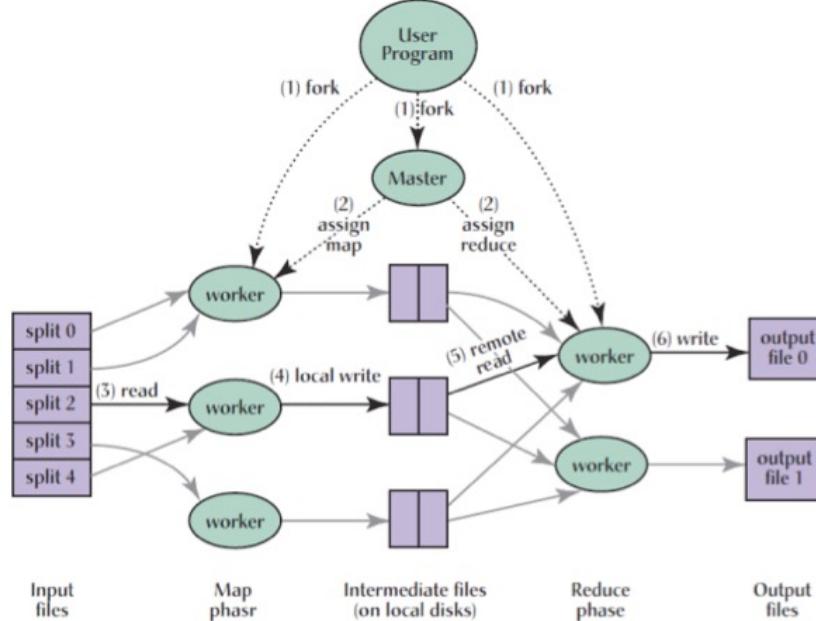


HDFS

- Maestro: NameNode
- Esclavo: {DataNode}, ..., {DataNode}



Procesamiento



Maestro : JobTracker
Esclavo : {tasktracker},...,

Ecosistema Apache Hadoop (1/2)

Incluye más de 150 proyectos



Hbase (la base de datos): sistema de bases de datos NoSQL que corre sobre HDFS (inspirada en Google BigTable).

Hive (el data warehouse): infraestructura de data warehouse construida sobre Hadoop.

Sqoop (la herramienta de ETL): herramienta para transferencia eficiente de datos entre Hadoop y bases de datos relacionales.

Mahout (la plataforma de data mining): algoritmos escalables de aprendizaje automático y minería de datos sobre Hadoop.

Ecosistema Apache Hadoop (2/2)



ZooKeeper (la herramienta de sincronización): servicio centralizado de configuración, nombrado, sincronización distribuida y servicios de grupos para grandes sistemas distribuidos.

Avro (el sistema de serialización): una plataforma para codificar y homogeneizar los datos de forma que se puedan transmitir de forma óptima por la red.

Pig (el helper para analizar grandes volúmenes de datos): lenguaje de alto nivel (de flujo de datos) para facilitar la escritura de programas MapReduce.

Flume (el agregador de logs): capturar, analizar y monitorizar grandes ficheros de log.

Distribuciones

Fuentes: <http://hadoop.apache.org/releases.html>

Sistemas pre-configurados proporcionados por empresas. Las tres distribuciones más extendidas son:

Cloudera (www.cloudera.com): contribuidor activo al proyecto que proporciona una distribución comercial y no-comercial de Hadoop (CDH).

MapR (www.mapr.com).

Hortonworks (www.hortonworks.com).

Cada proveedor ofrece imágenes de VM con Linux y Hadoop ya instalado.



HDFS: ventajas e inconvenientes

Hadoop puede acceder a diferentes tipos de sistemas de ficheros (local, HDFS, KFS, ...). No obstante se recomienda el uso de HDFS.

Entre sus **ventajas** destacan:

- Diseñado para almacenar ficheros muy grandes en commodity hardware.

- Elevado ancho de banda.

- Fiabilidad mediante replicación.

También tiene algunos **inconvenientes**:

- Elevada latencia.

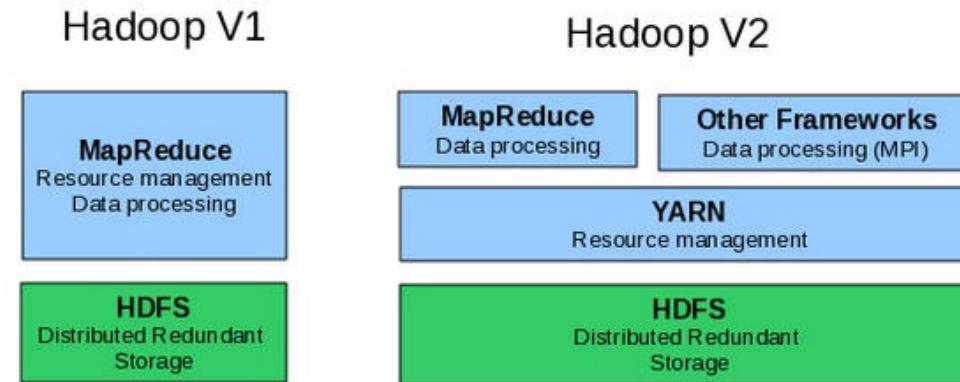
- Poco eficiente con muchos ficheros pequeños

Interfaces

HDFS cuenta con tres interfaces:
API de programación.
Interfaz web:

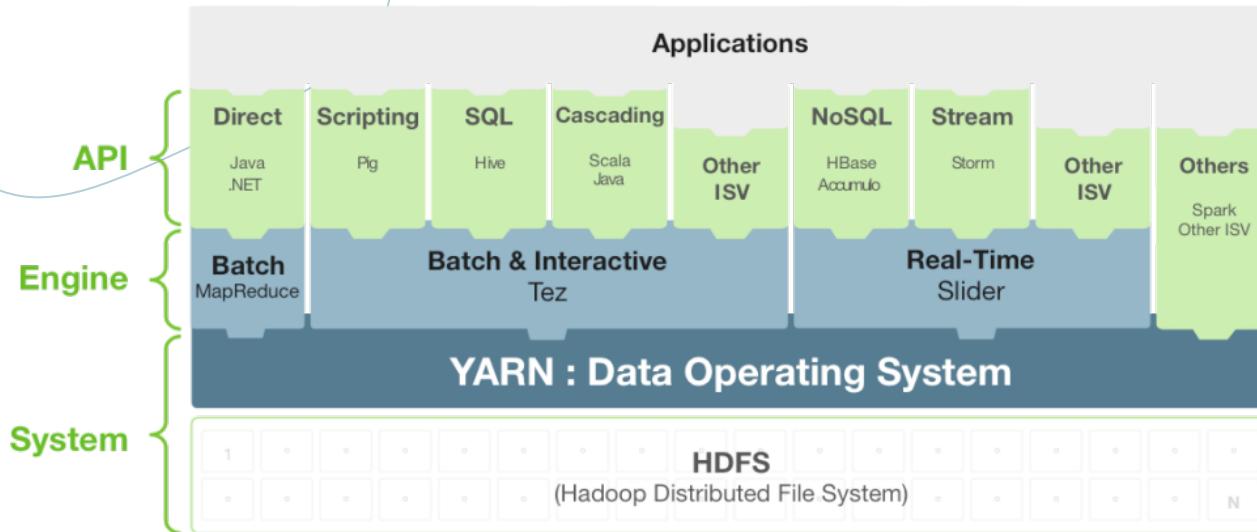
Puerto 50070 del Namenode.
Línea de órdenes:
HDFS tiene su propia shell.
Ayuda: \$ hadoop dfs -help

Evolución de Hadoop



Versión más reciente: **2.10**

Hadoop YARN



Apache Hadoop YARN es el sistema operativo de datos de Hadoop 2, responsable de la gestión del acceso a los recursos críticos de Hadoop. YARN permite al usuario interactuar con todos los datos de múltiples maneras al mismo tiempo

¿Cómo usar Hadoop?



Plataformas Cloud con instalación de Hadoop

Amazon Elastic Compute Cloud (Amazon EC2)
<http://aws.amazon.com/es/ec2/>



Windows Azure
<http://www.windowsazure.com/>

Instalación en un cluster Ejemplo
hadoop.ugr.es, infraestructura del grupo
SCI²S



Cluster hadoop: 16 servidores
Características de cada nodo:

- Microprocesador: Intel core i7-4930K, 12MB cache, 6 núcleos, 12 hebras)
- RAM 64 GB DDR3 1600MHz
- 1 HDD SATA 4TB, 6Gb/s

Limitaciones de Hadoop

Todo lo que no sea secuencial
Procesos iterativos
Procesamiento de grafos
Procesos interactivos

Algoritmos no implementables:

- Descenso en gradiente
- Expectation Maximization
- Boosting

Alternativas a Hadoop



APACHE GIRAPH

(<http://giraph.apache.org/>)

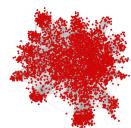
Procesamiento iterativo de grafos



APACHE TEZ

(<http://tez.apache.org/>)

Procesos de datos con estructuras de grafos acíclicos complejas



GPS - A Graph Processing System,

(Stanford) <http://infolab.stanford.edu/gps/>

para Amazon's EC2



Mellon Univ.) <https://github.com/graphlab-code/graphlab>
para Amazon's EC2

Distributed GraphLab
(Carnegie



PrIter (University of

Massachusetts Amherst,
Northeastern University-China)

<http://code.google.com/p/priter/>

Cluster propios y Amazon EC2 cloud



HaLoop

(University of Washington)

<http://clue.cs.washington.edu/node/14>

<http://code.google.com/p/haloop/>

Amazon's EC2



Spark (UC Berkeley)
(Apache Foundation)

<http://spark.incubator.apache.org/research.html>

GPU based platforms

Mars

Grex

GPMR



4

Spark

Spark

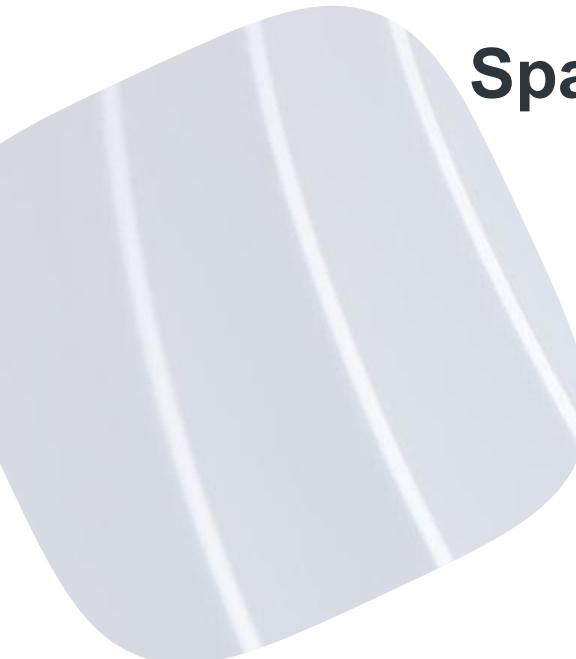


<http://spark.apache.org>

Big Data “In-memory”

It started as a research project at UC Berkeley in the AMPLab, which focuses on big data analytics. It introduces the **resilient distributed datasets (RDD) abstraction, allowing iterative algorithms**. It's about 100 times faster than Hadoop.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M., Shenker, S., and Stoica, I. Resilient Distributed Datasets: A fault-tolerant abstraction for in-memory cluster computing. NSDI 2012.

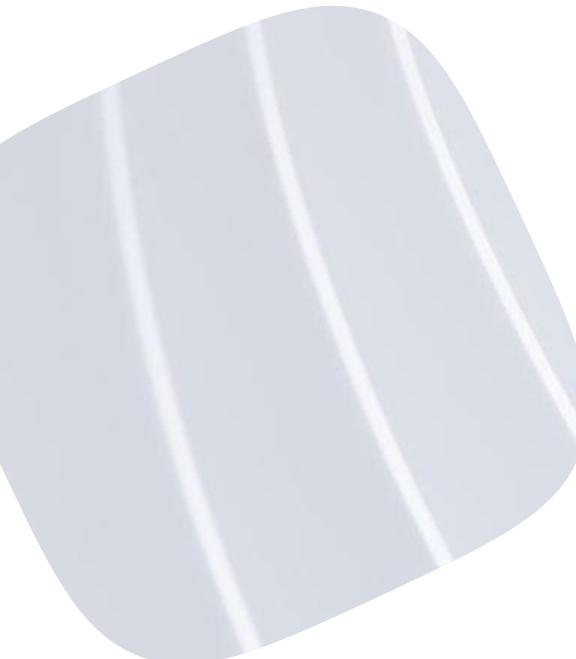


Spark

“A lightning-fast cluster computing technology, designed for fast computation”

It extends the MapReduce model to efficiently use it for more types of computations, including **interactive queries** and **stream processing**.
In-memory cluster computing

Propiedades



Plataforma de computación en cluster
Desarrollada en el AMPLab de Berkeley (UCB) en 2009
Paralelismo de datos implícito y con tolerancia a fallos
Muy rápido: 100x sobre Hadoop
Conjunto rico de herramientas de alto nivel
Desarrollado en Scala
APIs para Scala, java, Python, R,SQL

Versión actual: **2.3.4** (3.0)

Tipos de tareas



- Aplicaciones en serie (batch)**
- Algoritmos iterativos**
- Consultas interactivas**
- Procesamiento de flujos**
- Procesamiento de grafos**

Todo integrado en un mismo marco de trabajo

Plataformas

Metal (cluster)
Hadoop Yarn
Apache Mesos
Kubernetes
Cloud computing
Acceso a datos:

HDFS

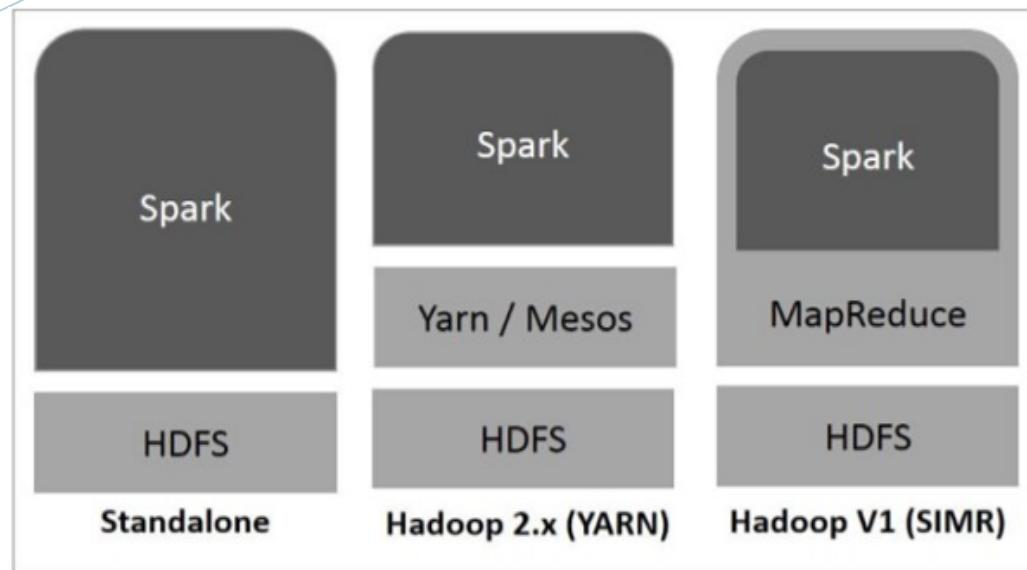
Cassandra

Hbase

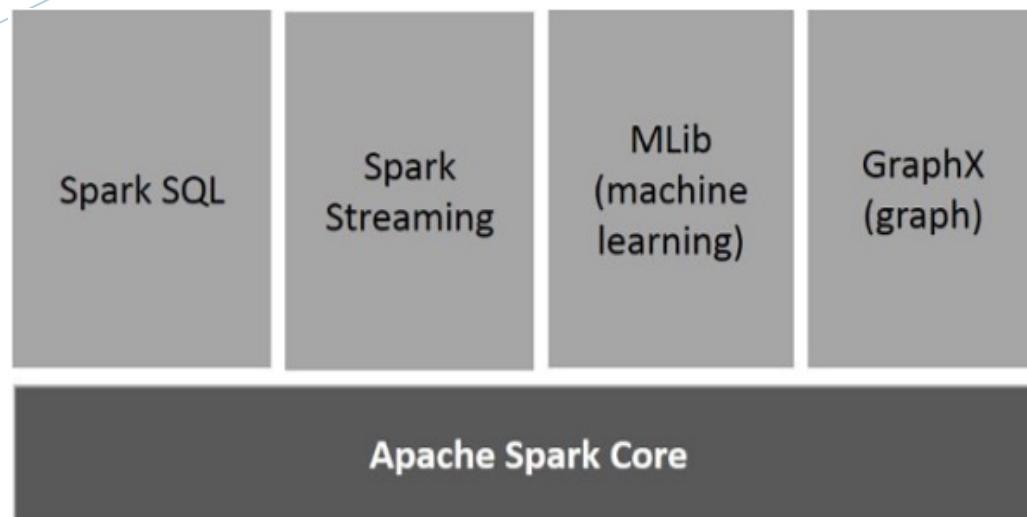
Hive

...

Modos de uso de Spark



Componentes





RDD

Resilient Distributed Datasets

Concepto central en Spark

Conjunto **inmutable** de objetos

Cualquier tipo de objetos

Cada RDD está particionado

lógicamente

Cada partición puede ser procesada en
nodos independientes



RDD (2)

Los RDD se pueden crear mediante operaciones determinísticas sobre datos en memoria, almacenamiento externo u otros RDD
Su reparto permite su procesamiento en paralelo



Inconvenientes de MapReduce

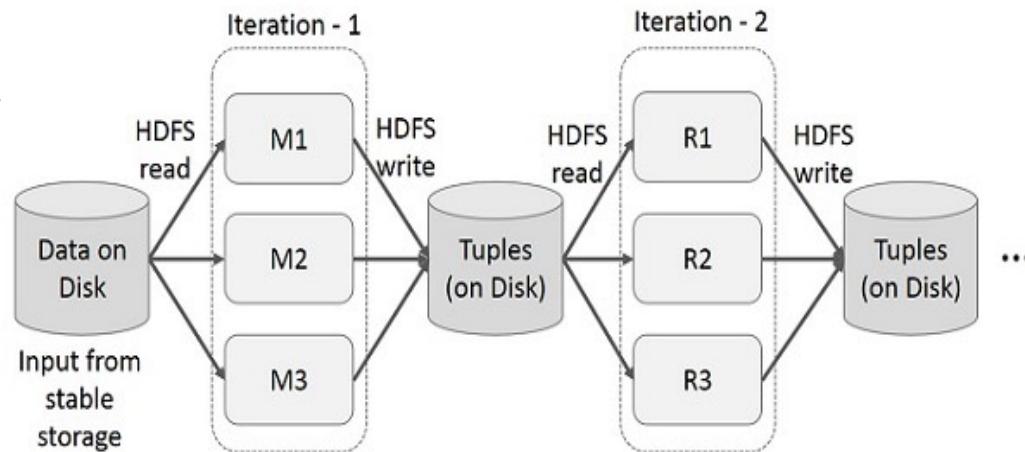
Principal escaso para su eficiencia:
Intercambio de datos entre trabajos
MapReduce

MR requiere almacenar resultados
en disco: replicación, serialización,
E/S disco:

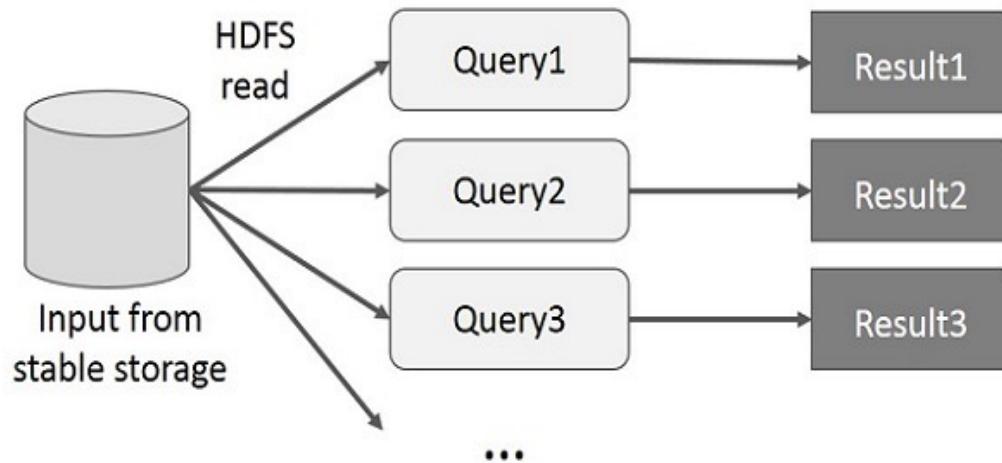
90% del tiempo con HDFS: **lento**

Afecta a procesos iterativos e
interactivos

Operaciones iterativas en MR



Operaciones interactivas en MR



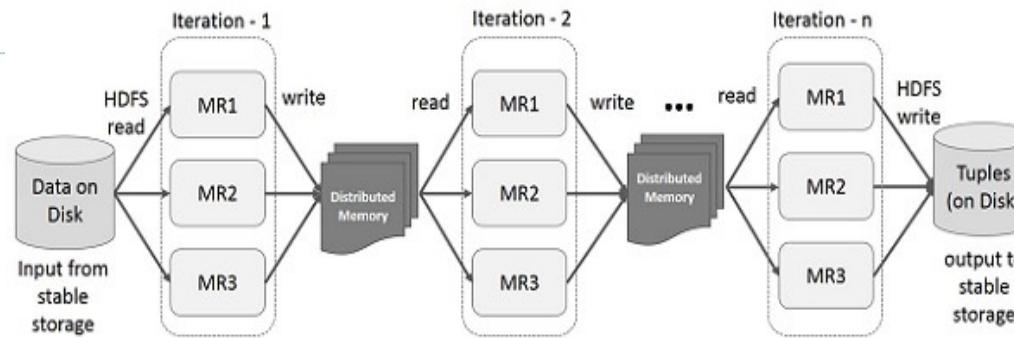
Compartición de datos con RDD

Spark **almacena** el conjunto de datos en **memoria principal**, distribuida por todos los nodos.

La compartición de datos entre procesos en el mismo nodo a velocidad de RAM, no de red + disco

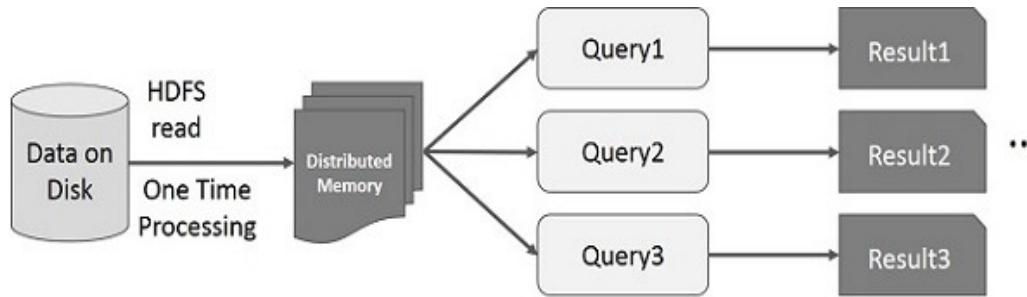
Gestión automática de transferencias desde disco, si no caben al 100% en memoria.

Operaciones iterativas con RDD



Si falta memoria, a disco

Operaciones interactivas con RDD



Los RDDS pueden hacer **persistentes** en memoria y transferir a disco

Cálculo con RDDs

Transformaciones: se aplican sobre un RDD y devuelven otro RDD (generan RDD).

Lazy computing

Acciones: modificaciones directas sobre un RDD

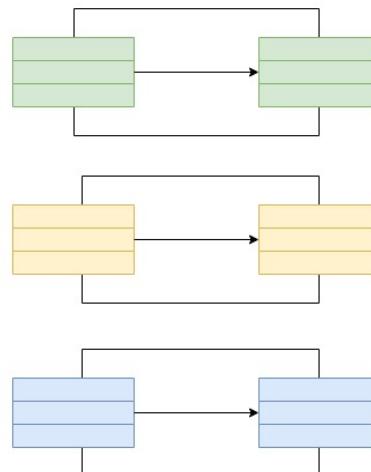
Transformaciones sobre RDD

Nombre	Efecto
map(func)	Se aplica func a cada elemento del RDD
filter(func)	Devuelve los elementos del RDD que cumplen func
flatMap()	Similar a map, pero puede producir varios elementos de salida
mapPartition()	Similar a map, pero aplicado a cada partición independientemente
mapPartitionWithIndex(func)	Similar a la anterior, pero proporcionando un índice sobre la partición
sample()	Muestrea una parte del RDD
union()	Unión de dos RDDs
intersection()	Intersección de dos RDDs
distinct()	Devuelve los elementos distintos

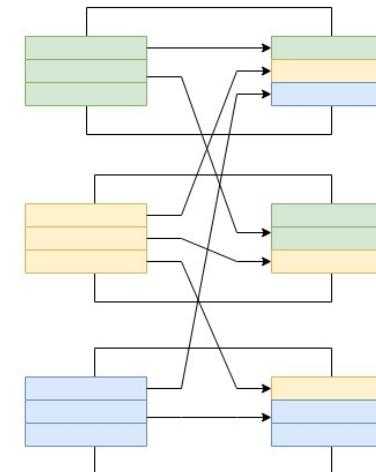
Transformaciones (2)

Nombre	Efecto
aggregateByKey	Agregación por clave
sortByKey	Ordenación por clave
join()	Calcula un “join” interno.
cogroup()	Agregar con agrupación
cartesian	Calcula el producto cartesiano
pipe	Procesa los elementos con un <i>pipe</i> del SO
coalesce	Reduce el número de particiones (útil después de filter)
repartition	Reparticionar, con redistribución de elementos
groupByKey()	Agregación por claves

Transformaciones (3): Narrow vs wide



Transformación narrow



Transformación wide

Nombre	efecto
reduce(func)	Agregar elementos del dataset usando func
collect()	Convertir a <i>array</i>
count()	Contar elementos
first()	Primer elemento del dataset
take(n)	Array con los n primeros elementos
takeSample	
takeOrdered	
saveAsTextFile	
saveAsSequenceFile	
countByKey	
foreach	



Bibliografía

Referencias (1/2)

- A. Holmes, “Hadoop in Practice”, Manning, 2012.
- J.S. Hurwitz et al. “Big Data for Dummies”, Wiley, 2013.
- C. Lam. “Hadoop in Action”, Manning, 2010.
- E. Sammer, “Hadoop Operations”, O'Reilly, 2013.
- T. Wite, “Hadoop: The Definitive Guide”, O'Reilly, 2012.

Referencias (2/2)

Tutorial Spark:

https://www.tutorialspoint.com/apache_spark

H. Karau, et al., “Learning Spark: Lightning-Fast Big Data Analysis”, O'Reilly, 2015

J. Wills, U. Laserson, “Advanced Analytics with Spark: Patterns for Learning from Data at Scale”, O'Reilly, 2017