

Hadoop

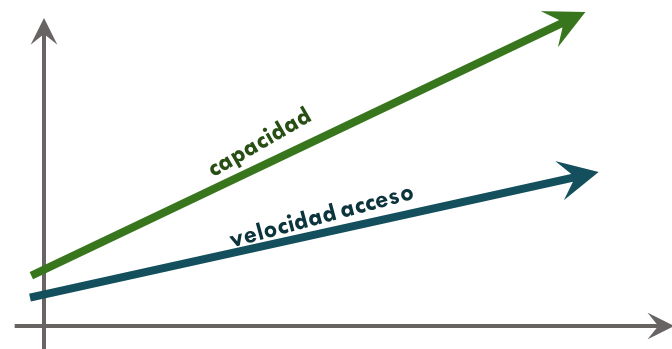


Por dónde empezaremos ...

- Introducción
- Componentes
- Replicación
- Guía de dimensionamiento
- Ventajas y desventajas de HDFS
- Historia
- Distribuciones Hadoop
- Ecosistema Hadoop
- Configuración del entorno
- Creación y ejecución de trabajos
- Opciones de ejecución
- Interfaz para múltiples lenguajes
- ¿Quién usa Hadoop?
- Reflexiones sobre Hadoop

➤ Introducción (1 / 5)

- El aumento de la capacidad de **almacenamiento** es mayor al incremento de la velocidad de acceso.
- Problema si se necesita acceder a grandes conjuntos de datos.





➤ Introducción (2/5)

➤ Posible solución:

➤ Paralelizar acceso a datos.

- Repartir datos en n discos.

- Divide por n el tiempo de acceso a los datos.

➤ Problemas

- Mayor cantidad de hardware → Incremento de la probabilidad de fallos.

- Mayor complejidad para trabajar con los datos.



➤ Introducción (3/5)

➤ Hadoop intenta ser una solución y sus componentes principales abordan los dos problemas planteados.

➤ Hadoop Distributed File System (HDFS)

➤ Sistema distribuido de ficheros con redundancia automática de los datos.

➤ MapReduce

➤ Paradigma de programación sencillo y versátil para programación en paralelo.



➤ Introducción (4/5)

➤ Características de Hadoop

➤ Procesamiento de consultas por lotes

- Especializado en tareas que requieren procesar todo el conjunto de datos o una parte importante de este.

- No adecuado para consultas selectivas o actualizaciones frecuentes, típicas de SQL.

- Permite realizar consultas ad-hoc sobre conjuntos de datos muy grandes en tiempos razonables.



➤ Introducción (5/5)

➤ Ventajas sobre bases de datos relacionales:

- Almacenamiento consecutivo elimina tiempos de búsqueda.
- No requiere estructura fija para los datos.
- Escala linealmente.

➤ Ventajas sobre computación grid:

- Almacenamiento local de los datos.
- No requiere programación a bajo nivel.
- Tratamiento automático de fallos.

Hadoop is a framework that allows for the *distributed* processing of *large data sets* across clusters of computers using a *simple programming model*.



➤ Componentes (1 / 20)

➤ Hadoop Common

- Conjunto de utilidades que dan soporte a otros módulos de Hadoop.

➤ Hadoop Distributed File System (HDFS)

- Sistema de archivos distribuidos que proporciona alto rendimiento en el acceso a datos.

➤ Hadoop Yarn

- Framework para la planificación de tareas y gestión de recursos del cluster.

➤ Hadoop MapReduce

- Un sistema basado en YARN para el procesamiento en paralelo de grandes conjuntos de datos.



➤ Componentes (2/20)

➤ HDFS - Sistema de ficheros distribuidos (1/13)

- Creado a partir del Google File System (GFS)
- Diseñado para trabajar con ficheros muy grandes (TB, PB, ...) y optimizado para la lectura y escritura de grandes volúmenes de datos.
- Su diseño reduce la E/S en la red, es escalable y altamente disponible gracias a las técnicas de replicación y tolerancia ante fallos que implementa.



➤ Componentes (3/20)

➤ HDFS - Sistema de ficheros distribuidos (2/13)

- Puede funcionar en hardware de consumo, no requiere hardware con alta fiabilidad (muy costoso).
 - Añade mecanismos para control y solución de fallos.
- Acceso a datos por lotes
 - Patrón escritura única/múltiples lecturas.
- Tamaño muy grande de bloque
 - 64 MB por defecto (4-32 KB típico para File System).
 - Ficheros pequeños no ocupan bloque entero.



➤ Componentes (4/20)

➤ HDFS - Sistema de ficheros distribuidos (3/13)

- Optimizado para la transmisión y lectura de las peticiones de archivos de gran tamaño, pero no en la búsqueda de muchas peticiones pequeñas.
- Funciona en modo maestro/esclavo con un Namenode y múltiples Datanodes.
- Cada nodo de almacenamiento ejecuta un proceso llamado Datanode que gestiona los bloques en ese nodo, y estos procesos están coordinados por un proceso Namenode maestro que se ejecuta en un nodo independiente.



➤ Componentes (5/20)

➤ HDFS - Sistema de ficheros distribuidos (4/13)

- En lugar de controlar fallos de disco (por ejemplo, usando técnicas de redundancias físicas en las matrices de discos), utiliza replicación. Cada uno de los bloques que comprenden un archivo es almacenado en varios nodos del cluster, y el Namenode monitoriza constantemente cada Datanode para asegurarse de que se sigue manteniendo el factor de replicación. Si esto no sucede, programa la copia de un bloque dentro del cluster.



➤ Componentes (6/20)

➤ HDFS - Sistema de ficheros distribuidos (5/13)

- El Namenode y Datanode es software que se ejecuta en máquinas de bajo coste con sistema operativo Linux.
- HDFS se construye utilizando el lenguaje Java de modo que cualquier máquina que soporte Java puede ejecutar el software Namenode o del Datanode.



➤ Componentes (7/20)

➤ HDFS - Sistema de ficheros distribuidos (6/13)

➤ No recomendado para todos los escenarios:

➤ Accesos con baja latencia.

➤ Optimizado para alta tasa de transferencia, no acceso aleatorio.

➤ Muchos ficheros pequeños

➤ Metadatos en memoria determina un máximo de ficheros que se pueden manejar (~150 bytes por fichero)

➤ Múltiples escritores

➤ No hay soporte para sincronización de escrituras.

➤ Sólo se puede escribir al final del fichero.

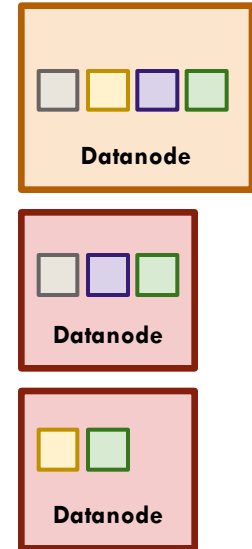


➤ Componentes (8/20)

➤ HDFS - Sistema de ficheros distribuidos (7/13)

➤ Datanodes (1/2)

- Nodos esclavos responsables de leer y escribir las peticiones de los clientes.
- Organizados en racks.
- Cuando un cliente solicita una lectura o escritura de datos, el fichero se divide en bloques.
- Almacenan los bloques de datos.

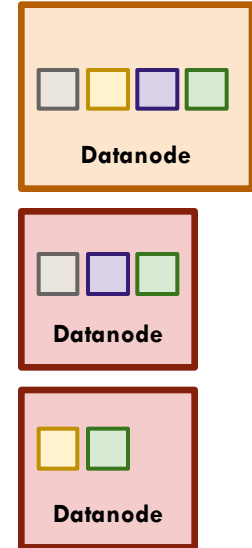


➤ Componentes (9/20)

➤ HDFS - Sistema de ficheros distribuidos (8/13)

➤ Datanodes (2/2)

- Los bloques pueden estar replicados en varios datanodes.
- Alta tolerancia a fallos.
- Cada fichero ocupa uno o varios bloques.
 - Puede no ocupar el bloque entero.

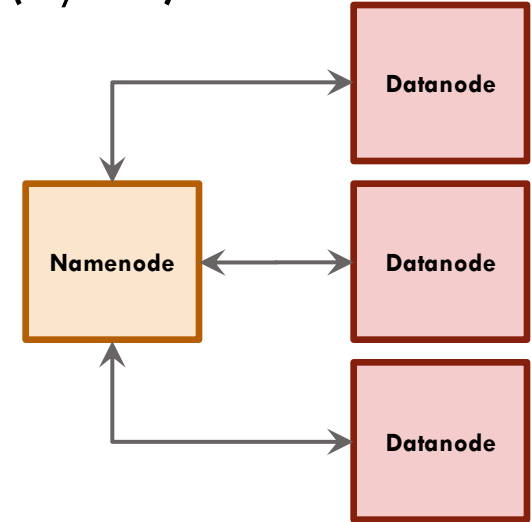


➤ Componentes (10/20)

➤ HDFS - Sistema de ficheros distribuidos (9/13)

➤ Namenode (1/3)

- Es la pieza central de HDFS.
- Nodo maestro al que acceden los clientes.
 - La lectura de los datos de un bloque se hace directamente al datanode que lo contiene.
- Almacena árbol de directorios y metadatos de los ficheros.

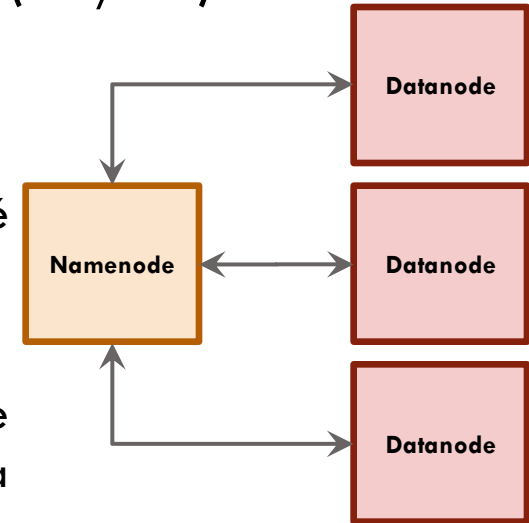


➤ Componentes (11/20)

➤ HDFS - Sistema de ficheros distribuidos (10/13)

➤ Namenode (2/3)

- Coordina replicación de bloque
 - Garantiza que cada nodo esté suficientemente replicado.
- Sabe dónde está cada bloque
 - Al conectar con los datanodes recibe información de los bloques que maneja cada uno.
- No almacena datos en sí mismo.



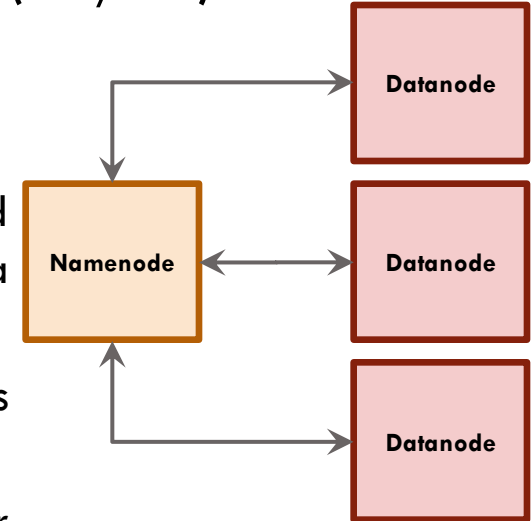
➤ Componentes (12/20)

➤ HDFS - Sistema de ficheros distribuidos (11/13)

➤ Namenode (3/3)

➤ Único punto de fallo

- Pérdida de datos implica incapacidad de saber a qué fichero pertenece cada bloque.
- La recuperación ante su caída no es trivial.
- Réplica inactiva para garantizar disponibilidad.
- Disponible en versión 2.x





➤ Componentes (13/20)

➤ HDFS - Sistema de ficheros distribuidos (12/13)

➤ Secondary Namenode (1/2)

- Su objetivo es mezclar la imagen del Namenode con el log de transacciones ejecutadas → evita que el log crezca demasiado.
- Establece puntos de control del Namenode, descargando periódicamente una imagen así como los logs.
- Suele correr en una máquina separada, ya que este proceso requiere mucha memoria y CPU.
- Mantiene una copia de la imagen del namespace para que pueda ser usada en el caso de que se caiga el Namenode.



➤ Componentes (14/20)

➤ HDFS - Sistema de ficheros distribuidos (13/13)

➤ Secondary Namenode (2/2)

➤ Si el Namenode falla:

- Si se puede reiniciar en el mismo nodo físico, no es necesario parar los Datanodes, sino solo reiniciar el Namenode.
- Si no se puede reiniciar es necesario copiar la imagen o bien del Namenode o bien del Secondary Namenode en otro nodo, al que habrá que unir los nuevos logs. En este caso es necesario reiniciar todo el cluster.

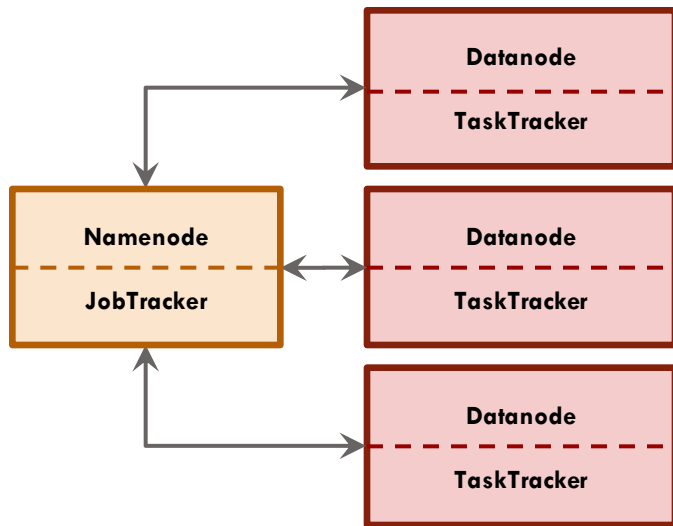


➤ Componentes (15/20)

➤ MapReduce

- Cada ejecución consiste en uno o varios trabajos.
- Cada trabajo se divide en tareas (map o reduce).
- El nodo maestro gestiona los trabajos y reparte las tareas a los nodos esclavos.
- Comparte ordenadores con HDFS:
 - Misma arquitectura de nodos maestros y esclavos.
 - Aprovecha al máximo posible la localidad de los datos.
 - Pero puede funcionar con otros orígenes de datos.

- Componentes (16/20)
 - MapReduce v1 (Classic) (1/2)
 - Servicios
 - JobTracker
 - Gestiona recursos.
 - Registra los trabajos pendientes.
 - Asigna las tareas a los nodos.
 - Mantiene los trabajos cercad de los nodos.
 - Monitorización de TaskTracker.
 - Si falla, los trabajos de ejecución se pierden.

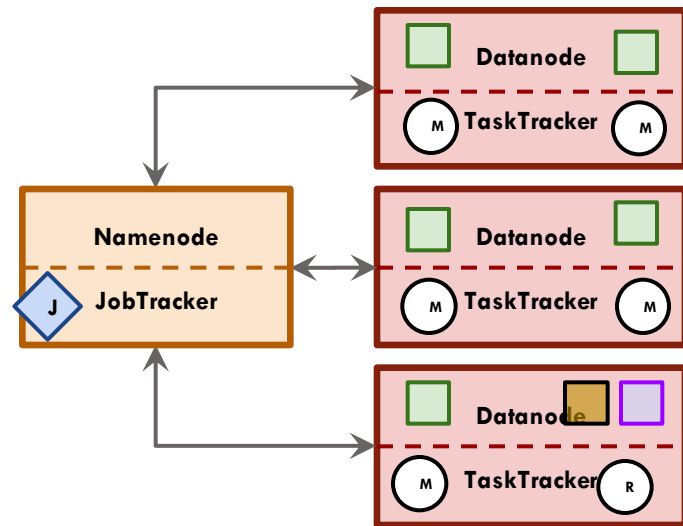


- Componentes (17/20)
 - MapReduce v1 (Classic) (2/2)

- Servicios

- TaskTracker

- Son los nodos.
 - Atienden operaciones MapReduce.
 - Tienen slots asignados para Map y para Reduce.
 - Controla las tareas de ejecución.
 - Notifica al JobTracker acerca del estado del nodo y las tareas.
 - Si falla o se produce un timeout, esa parte del trabajo se replanifica.

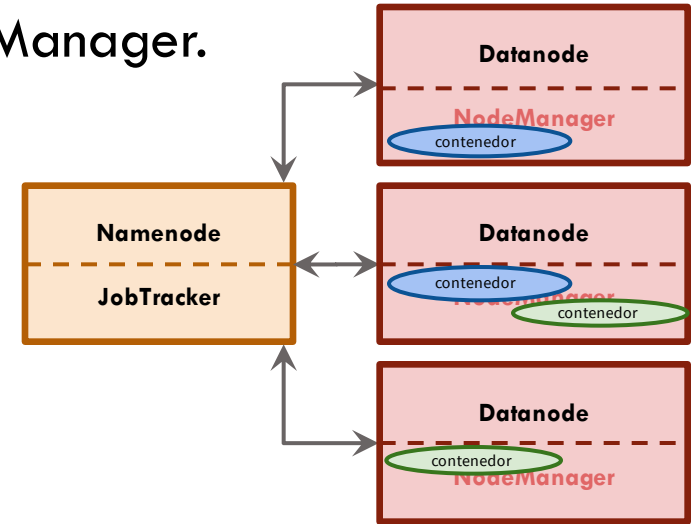


➤ Componentes (18/20)

➤ MapReduce v2 (conocido como YARN⁽¹⁾) (1/3)

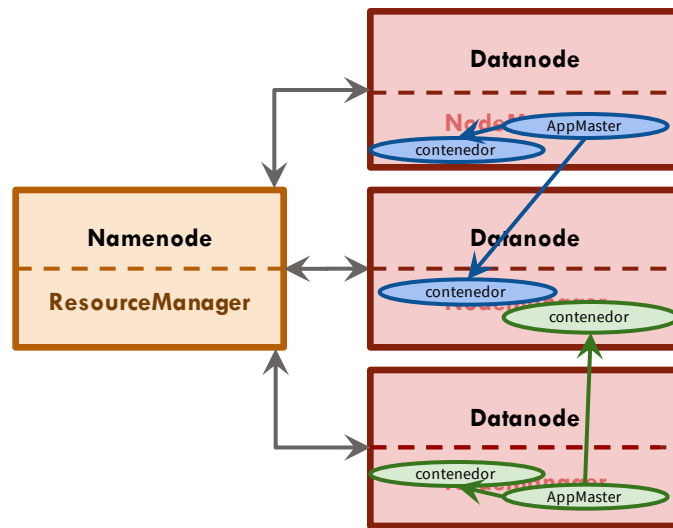
➤ TaskTracker se convierte en NodeManager.

➤ Contenedores ejecutan tareas.



(1) Yet Another Resource Negotiator

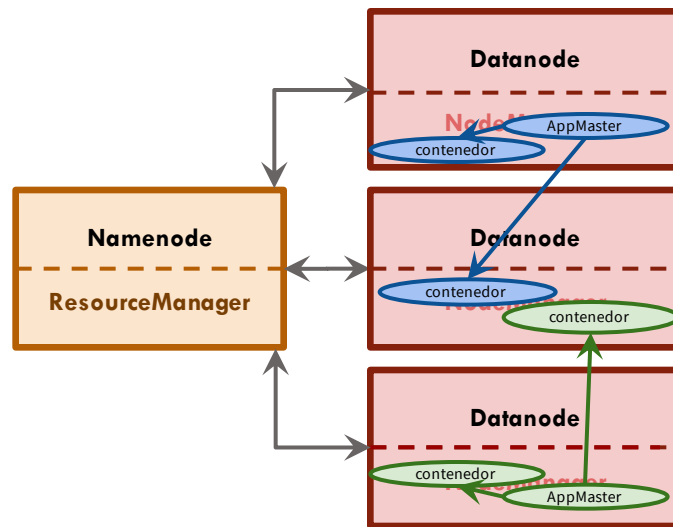
- Componentes (19/20)
 - MapReduce v2 (YARN) (2/3)
 - JobTracker se divide en dos:
 - ResourceManager
 - Gestión de recursos y apps.
 - ApplicationMaster en NodeManager
 - Gestión y monitorización de contenedores.



➤ Componentes (20/20)

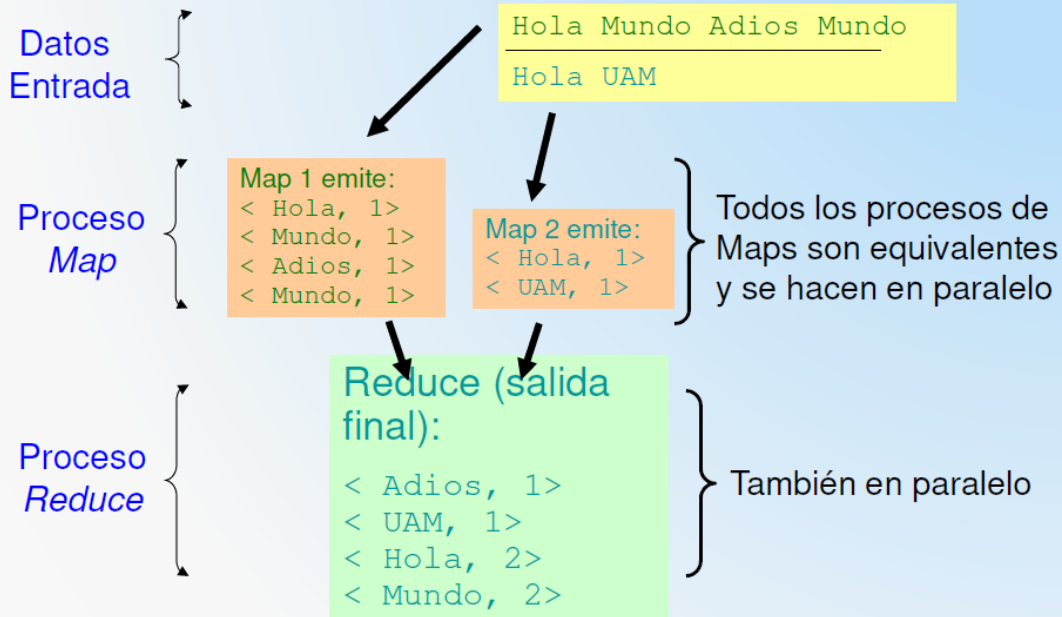
➤ MapReduce v2 (YARN) (3/3)

- Resuelve problemas de escalabilidad para clústeres muy grandes (más de 4000 nodos)
- Permite utilizar modelos de programación diferentes a MapReduce.



➤ Ejemplo de MapReduce

➤ Contar el número de apariciones de cada palabra





➤ Replicación (1 / 3)

- HDFS almacena cada archivo como una secuencia de bloques del mismo tamaño excepto el último.
- Los bloques de un archivo se replican para obtener tolerancia frente a fallos.
- El Namenode toma todas las decisiones con respecto a la replicación de bloques a partir de un mensaje de confirmación de actividad (ack) de cada uno de los Datanodes del cluster.



➤ Replicación (2/3)

- La recepción del mensaje de confirmación implica que el Datanode está funcionando correctamente.
- Por defecto, el factor de replicación es 3:
 - Réplica en nodo del rack local.
 - Réplica en otro nodo del mismo rack.
 - Réplica en otro nodo de diferente rack.
- La probabilidad de fallo de rack es mucho menor que la de fallo de un nodo.



➤ Replicación (3/3)

➤ Con esta política:

- 1/3 de las réplicas están en un nodo.
- 2/3 de las réplicas están en un mismo rack.
- 1/3 de las réplicas restantes se distribuyen uniformemente a través de los racks restantes.



- Guía de dimensionamiento en almacenamiento
 - Default replication factor en HDFS = 3.
 - Deja aproximadamente un 25% disponible.
 - Por tanto, necesitamos aproximadamente cuatro veces el tamaño del dato en crudo para guardarlo en HDFS.
 - Si utilizamos compresión, pensamos aproximadamente en un 60%.
 - Puede haber excepciones dependiendo del tipo de compresión.
 - Datos distintos se comprimen de forma diferente.



- Sobre CPUs
 - De 8 a 12 cores por nodo es lo recomendado.
 - Si es posible el número de cores debe ser igual o superior al número de unidades de disco.
 - Considerar otras aplicaciones que requieran también más CPU como Machine Learning, procesamiento de vídeo, compresión, etc.



➤ Ventajas de HDFS

- Reduce interacciones con el Datanode tanto en la escritura como en la lectura (grandes ficheros).
- Reduce sobrecarga de red al mantener conexión TCP persistente.
- Reduce el tamaño de datos almacenado en el Namenode.

➤ Desventajas HDFS

- Los ficheros pequeños implican el empleo de grandes bloques de datos.



➤ Historia de Hadoop (1/6)

➤ 2002

- Creación del proyecto Nutch, un motor de búsqueda de código abierto.
- No escala para poder procesar miles de millones de páginas.

➤ 2003

- Se publica el paper de Google que describe Google File System.

➤ 2004

- Se implementa NDFS (Nutch Distributed FS) basado en GFS.
- Se publica el paper de Google que describe MapReduce.



➤ Historia de Hadoop (2/6)

➤ 2005

- Se implementa MapReduce para Nutch y se empiezan a portar sus algoritmos a este paradigma.

➤ 2006

- Doug Cutting, uno de los creadores de Nutch, llama al proyecto Hadoop (nombre del juguete de su hijo – elefante amarillo).
- Se ordenan datos de 1.8 TB en 188 nodos → 47,9 horas.
- Clúster de Yahoo! formado por 600 máquinas.



➤ Historia de Hadoop (3/6)

➤ 2007

- Yahoo! usa dos clústers de 1 000 máquinas.
- Hadoop se libera con HBASE y PIG es creado por Yahoo!.

➤ 2008

- Yahoo! usa un clúster de Hadoop de 10000 cores para generar su índice de búsqueda.
- El clúster de Yahoo! trabaja diariamente con 10 TB de datos.
- Un clúster de Hadoop con 910 nodos, ordena 1 TB en 209 seg.; Google ordena 1 TB en 68 seg. con su MapReduce.



➤ Historia de Hadoop (4/6)

➤ 2009

- Yahoo! con 7 clústers y 24.000 máquinas.
- Yahoo! ordena 1 TB en 62 segundos.

➤ 2010

- Yahoo! usa 4.000 nodos para procesar 70 TB.
- Apache Hive y Apache Pig es liberado para su uso
- Facebook usa 2.300 clústers para procesar 40 PB.



➤ Historia de Hadoop (5/6)

➤ 2011

- Yahoo! 42.000 nodos del clúster de Hadoop para procesar PB.
- Lanzada versión 1.0.
 - Primera versión considerada estable, soporte a HBASE.

➤ 2013

- Lanzada versión 2.2
 - Introduce el gestor de recursos distribuido (YARN)

➤ 2014

- Lanzada versión 2.6
- Spark se convierte en proyecto de primer nivel de Apache.



➤ Historia de Hadoop (6/6)

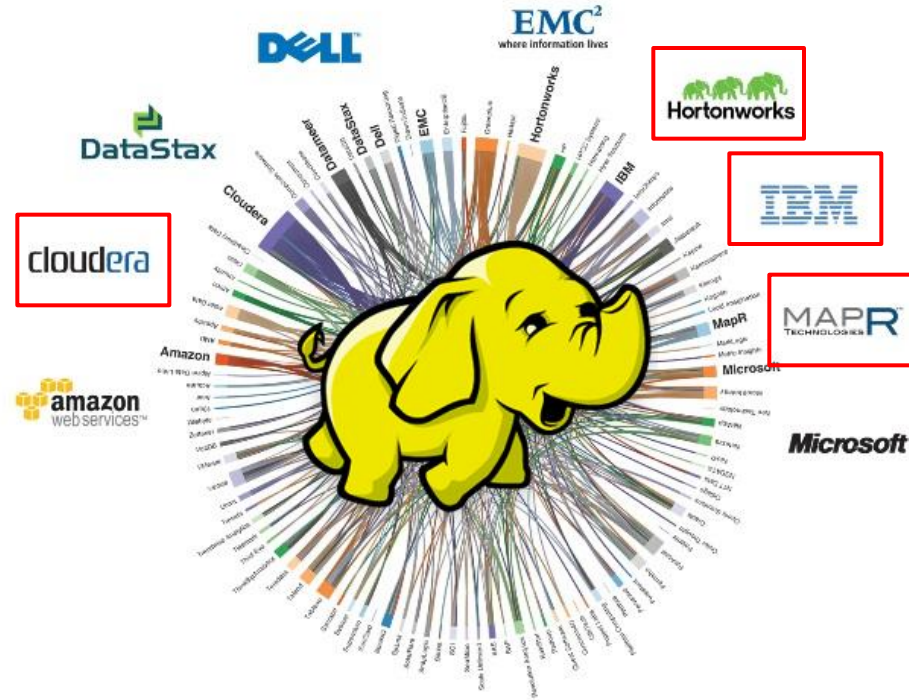
- 2015: lanzada versión 2.7
- 2017: lanzada versión 2.8
- 2018: lanzada versión 3.1.1
- 2019: lanzada versión 3.2.1
- 2020: lanzada versión 3.3.0
- Junio 2021: lanzada versión 3.3.1

<https://hadoop.apache.org/releases.html>

Hadoop



➤ Distribuciones Hadoop

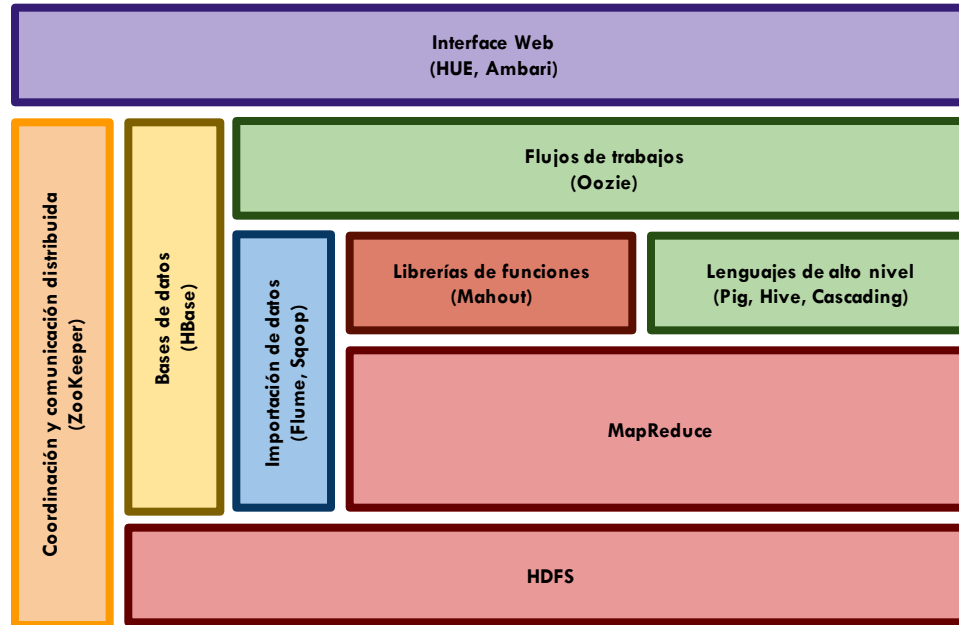




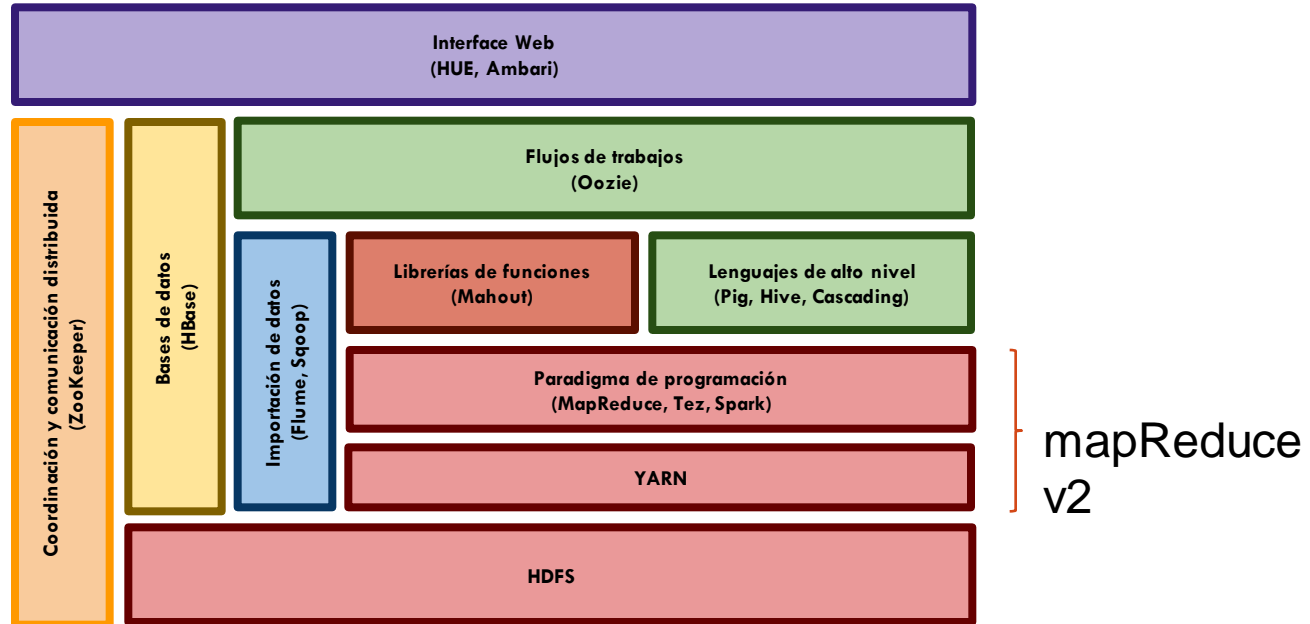
➤ Ecosistema Hadoop (1/3)

- Una motivación importante para usar Hadoop es la enorme comunidad de desarrolladores, empresas y proyectos que lo rodean:
 - Lenguajes de alto nivel (parecidos a SQL).
 - Herramientas de importación de datos.
 - Sistemas de monitorización, administración y coordinación.
 - Distribuciones completas.
 - Interfaces de administración visuales.
 - ...

➤ Ecosistema Hadoop (2/3)



➤ Ecosistema Hadoop (3/3)





- Configuración del entorno (1/10)
 - Descargar última versión estable (3.3.1 junio 2021)
 - <http://hadoop.apache.org/releases.html>
 - Descomprimir
 - Carpetas de Hadoop (1/2)
 - bin y sbin
 - Herramientas de control: permiten levantar/tirar servicios.
 - etc
 - Configuraciones: standalone, pseudodistribuido y clúster.



➤ Configuración del entorno (2/10)

➤ Carpetas de Hadoop (2/2)

➤ share

- Librerías Java: donde se encuentran los ficheros jar.

➤ lib

- Librerías nativas.



- Configuración del entorno (3/10)
 - Ficheros de configuración (1/4)
 - Scripts de inicialización
 - Extensiones .sh o .cmd.
 - Establecen las variables de entorno antes de ejecutar el código Java.
 - Utilizadas por herramientas de control de Hadoop.

- Configuración del entorno (4/10)
 - Ficheros de configuración (2/4)
 - Ficheros de recursos (1/2)
 - Tienen una estructura jerárquica.
 - Documentos XML que permiten configurar gran cantidad de aspectos de los componentes de Hadoop.

```
<?xml version="1.0"?>
<configuration>
  <property>
    <name>mapreduce.job.reduces</name>
    <value>1</value>
    <description>The default number of reduce tasks per job. Typically set to 99%
of the cluster's reduce capacity, so that if a node fails the reduces can
still be executed in a single wave.
Ignored when mapreduce.jobtracker.address is "local".
    </description>
  </property>
</configuration>
```




- Configuración del entorno (5/10)
 - Ficheros de configuración (3/4)
 - Ficheros de recursos (2/2)
 - Se pueden combinar cargando varios documentos.
 - Los valores del primer documento cargado se sobrescriben con los valores del segundo.
 - Los principales son dos ficheros para cada componente (core, mapred, yarn, hdfs): valores por defecto (default) y específicos para una o varias aplicaciones (site).
 - Ejemplos: core-default.xml, mapred-site.xml.



- Configuración del entorno (6/10)
 - Ficheros de configuración (4/4)
 - Ficheros de propiedades
 - Configuración del proyecto Log4j de Apache
 - Librería para registro (log) de información.
 - Permite configurar registro en tiempo de ejecución y no de compilación.

➤ Configuración del entorno (7/10)

- Instalar JDK.
- Establecer variables de entorno.

```
$ export JAVA_HOME=/usr/lib/jvm/default-java  
$ export HADOOP_PREFIX=<carpeta_descomprimida>  
$ export PATH=$PATH:$HADOOP_PREFIX/bin  
$ export CLASSPATH=$CLASSPATH:`hadoop classpath`
```

Comando que lista los directorios que tienen ficheros .jar que nos puedan interesar.

- En entornos distribuidos debe establecerse en todas las máquinas.
 - Modificar ficheros hadoop-env.sh.
 - O establecer variables a nivel de sistema.



➤ Configuración del entorno (8/10)

➤ Modos de ejecución

➤ Standalone

- Todos los servicios se ejecutan en el mismo proceso.
- Trabaja sobre sistema de ficheros local.

➤ Pseudodistribuido

- Todos los servicios se ejecutan en la misma máquina, pero en diferentes procesos.

➤ Cluster

- Entorno de producción.



➤ Configuración del entorno (9/10)

➤ Sistema de ficheros distribuido

➤ Formatear sistema de ficheros

```
$ hdfs namenode -format
```

➤ Levantar servicios namenode y datanode.

- Es posible navegar el sistema de ficheros virtual vía web en el puerto 50070.

➤ Utilizar el comando `hadoop fs` para gestionar los ficheros

- `put`, `get`, `cp`, `ls`, `mkdir`, `rm`, `cat`, ...

```
$ hadoop fs -help
```

- Configuración del entorno (10/10)
 - URLs de recursos
 - Hadoop puede manejar diferentes sistemas de ficheros.
 - Por defecto: propiedad `fs.defaultFS` en `core-site.xml`

| Tipo de sistema | Prefijo en URLs | Ejemplo |
|---------------------------|----------------------|--|
| Sistema de ficheros local | file | file:///var/log |
| HDFS | hdfs | hdfs://localhost:8020/user/vagrant |
| HDFS vía HTTP | hftp, hsftp, webhdfs | webhdfs://example.com/user/vagrant |
| FTP | ftp | ftp://ji:pass@example.com/user/vagrant |
| S3 | s3, s3n | s3n://elasticmapreduce/samples/wordcount/input |

- Creación y ejecución de trabajos (1/10)
 - Ejemplo sencillo: contar palabras (1/3)
 - Mapper

```
public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {  
    private final static IntWritable one = new IntWritable(1);  
    private Text word = new Text();  
  
    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {  
        String line = value.toString();  
        StringTokenizer tokenizer = new StringTokenizer(line);  
        while (tokenizer.hasMoreTokens()) {  
            word.set(tokenizer.nextToken());  
            context.write(word, one);  
        }  
    }  
}
```

- Creación y ejecución de trabajos (2/10)
 - Ejemplo sencillo: contar palabras (2/3)
 - Reducer

```
public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {  
    public void reduce(Text key, Iterable<IntWritable> values, Context context)  
        throws IOException, InterruptedException {  
  
        int sum = 0;  
        for (IntWritable val : values) {  
            sum += val.get();  
        }  
        context.write(key, new IntWritable(sum));  
    }  
}
```


- Creación y ejecución de trabajos (3/10)
 - Ejemplo sencillo: contar palabras (3/3)
 - Driver

```
public class WordCountDriver {  
    public static void main(String[] args) throws Exception {  
        Job job = Job.getInstance();  
        job.setJobName("Word count");  
        job.setJarByClass(WordCountDriver.class);  
        job.setMapperClass(WordCountMapper.class);  
        job.setReducerClass(WordCountReducer.class);  
        job.setOutputKeyClass(Text.class);  
        job.setOutputValueClass(IntWritable.class);  
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        FileOutputFormat.setOutputPath(job, new Path(args[1]));  
        System.exit(job.waitForCompletion(true) ? 0 : 1);  
    }  
}
```

➤ Creación y ejecución de trabajos (4/10)

➤ Compilar código

```
$ javac *.java -d .
```

➤ Crear un JAR

```
$ jar cf wordcount.jar *.class
```

➤ Enviar trabajo

➤ La carpeta de salida no debe existir

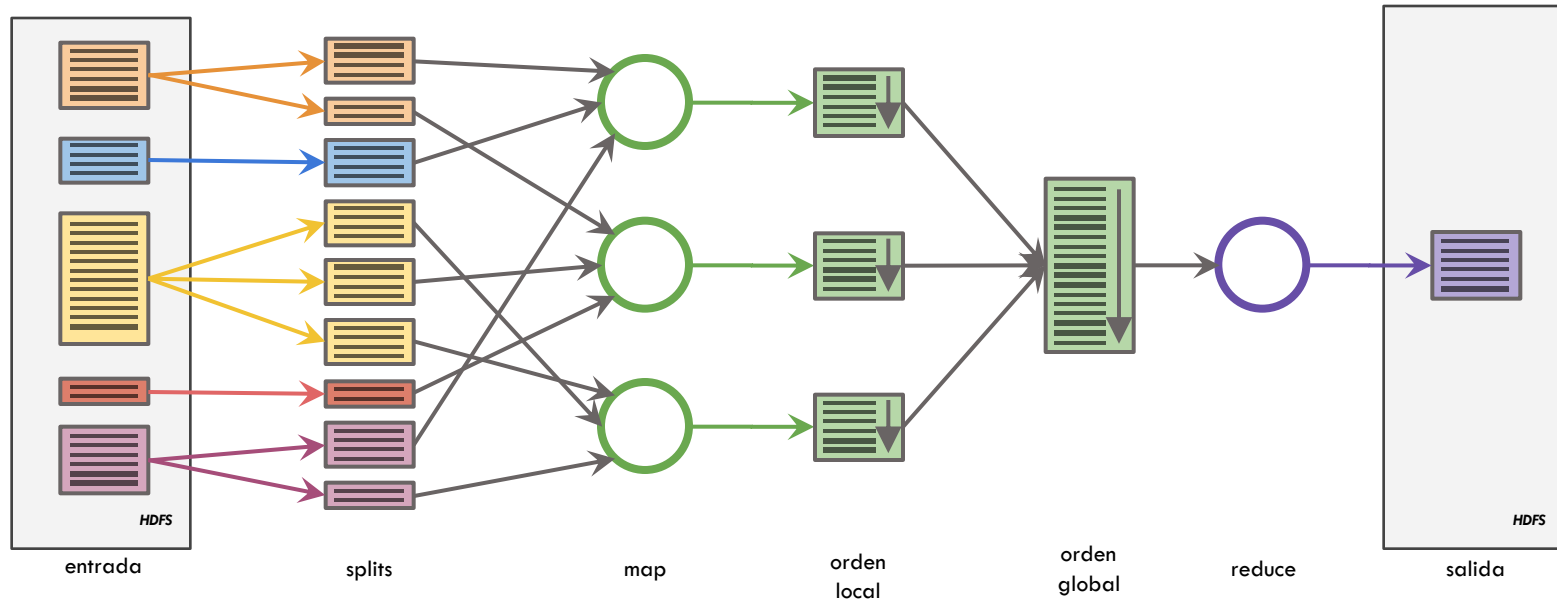
```
$ hadoop jar wordcount.jar mapreduce.WordCountDriver input output
```



Demostración

Ejecución de un trabajo MapReduce en Hadoop

- Creación y ejecución de trabajos (5/10)
 - ¿Cómo se ejecutan las tareas?



- Creación y ejecución de trabajos (6/10)
 - Fichero `_SUCCESS`
 - Indica que la tarea se ha completado correctamente.
 - Ficheros `part-x-nnnnn`
 - Resultados del trabajo
 - $x = m \rightarrow$ resultados del map (cuando no hay reduce)
 - $x = r \rightarrow$ resultados del reduce (lo más habitual)
 - `nnnnn` \rightarrow número de tarea
 - Cada tarea genera un fichero diferente.

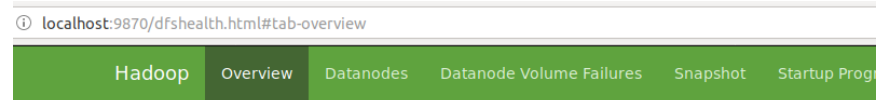


- Creación y ejecución de trabajos (7/10)
 - Hadoop facilita varias interfaces web para realizar consultas:
 - NameNode y DataNode: <http://localhost:50070/>
 - ResourceManager: <http://localhost:8088/>
 - Map Reduce JobHistory: <http://localhost:8188/>

➤ Creación y ejecución de trabajos (8/10)

➤ NameNode Web Interface

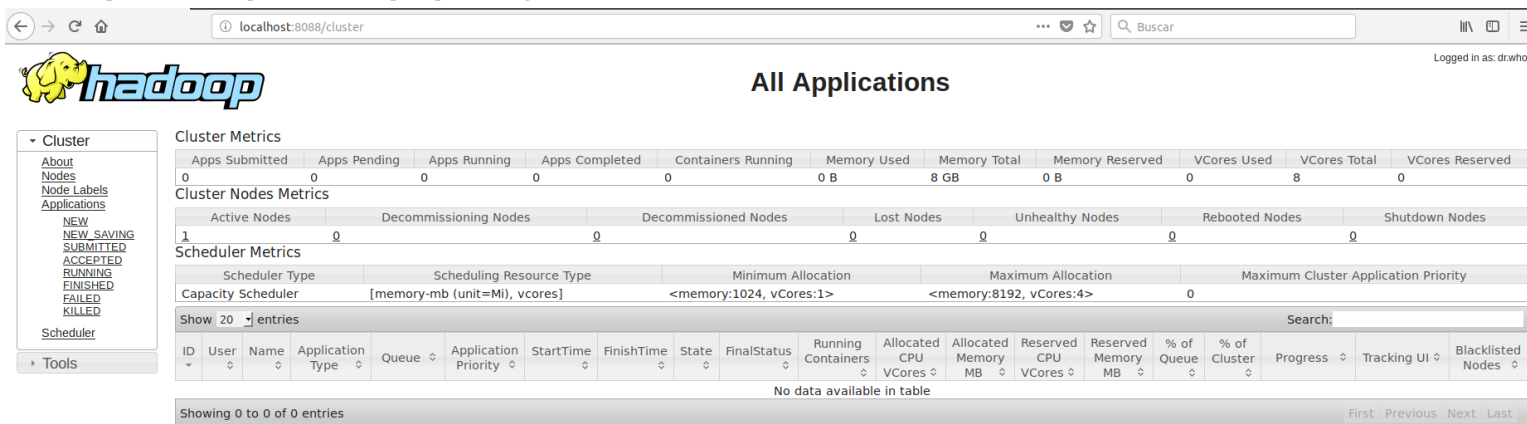
- Muestra un resumen del cluster incluyendo información de capacidad total y restante, así como nodos activos o muertos. Además, permite navegar por el namespace de HDFS y ver los contenidos de los ficheros desde el navegador. También permite el acceso a los logs de Hadoop.



Overview 'localhost:9000' (active)

| | |
|----------------|--|
| Started: | Sat Feb 23 17:43:05 +0100 2019 |
| Version: | 3.1.2, r1019dde65bcf12e05ef48ac71e84550d589e5d9a |
| Compiled: | Tue Jan 29 02:39:00 +0100 2019 by sunilg from branch-3.1.2 |
| Cluster ID: | CID-c634868b-9242-47b0-a8cd-0e94d99ed9a0 |
| Block Pool ID: | BP-204073103-127.0.1.1-1550283266614 |

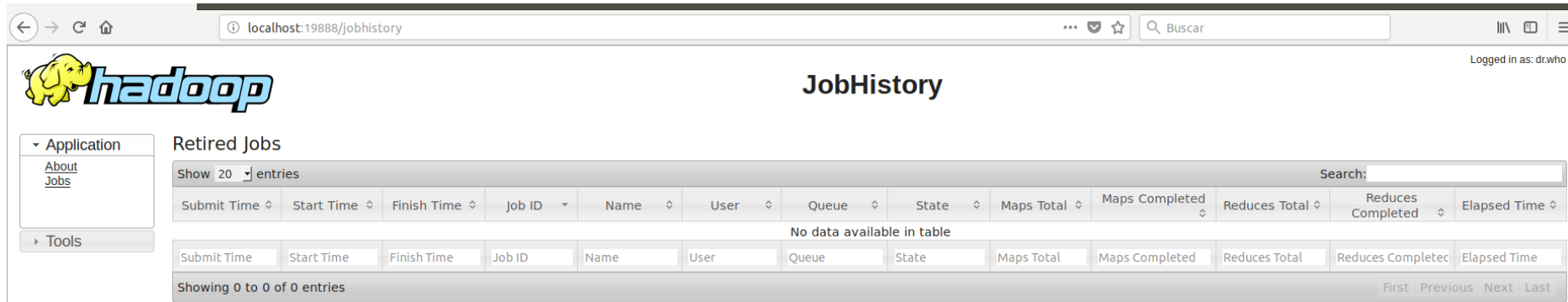
- Creación y ejecución de trabajos (9/10)
- ResourceManager Web Interface
 - Permite consultar las estadísticas y métricas de los recursos usados en el cluster, los trabajos (jobs) que se están ejecutando y los que hay programados.



The screenshot shows the Hadoop ResourceManager Web Interface. The browser address bar indicates the URL is `localhost:8088/cluster`. The page title is "All Applications". On the left, there is a sidebar menu with options like "Cluster", "About Nodes", "Node Labels", "Applications", and "Scheduler". The main content area displays several tables:

- Cluster Metrics:** A table with 11 columns: Apps Submitted, Apps Pending, Apps Running, Apps Completed, Containers Running, Memory Used, Memory Total, Memory Reserved, VCores Used, VCores Total, and VCores Reserved. All values are 0 except for Memory Total (8 GB).
- Cluster Nodes Metrics:** A table with 8 columns: Active Nodes, Decommissioning Nodes, Decommissioned Nodes, Lost Nodes, Unhealthy Nodes, Rebooted Nodes, and Shutdown Nodes. All values are 0.
- Scheduler Metrics:** A table with 5 columns: Scheduler Type, Scheduling Resource Type, Minimum Allocation, Maximum Allocation, and Maximum Cluster Application Priority. Values include "Capacity Scheduler", "[memory-mb (unit=Mb), vcores]", "<memory:1024, vCores:1>", "<memory:8192, vCores:4>", and "0".
- Applications Table:** A table with 20 columns: ID, User, Name, Application Type, Queue, Application Priority, StartTime, FinishTime, State, FinalStatus, Running Containers, Allocated CPU VCores, Allocated Memory MB, Reserved CPU VCores, Reserved Memory MB, % of Queue, % of Cluster, Progress, Tracking UI, and Blacklisted Nodes. The table is currently empty, showing "No data available in table".

- Creación y ejecución de trabajos (10/10)
 - Map Reduce y Job History Web Interface
 - Muestra información y detalles de trabajos Map Reduce ejecutados.



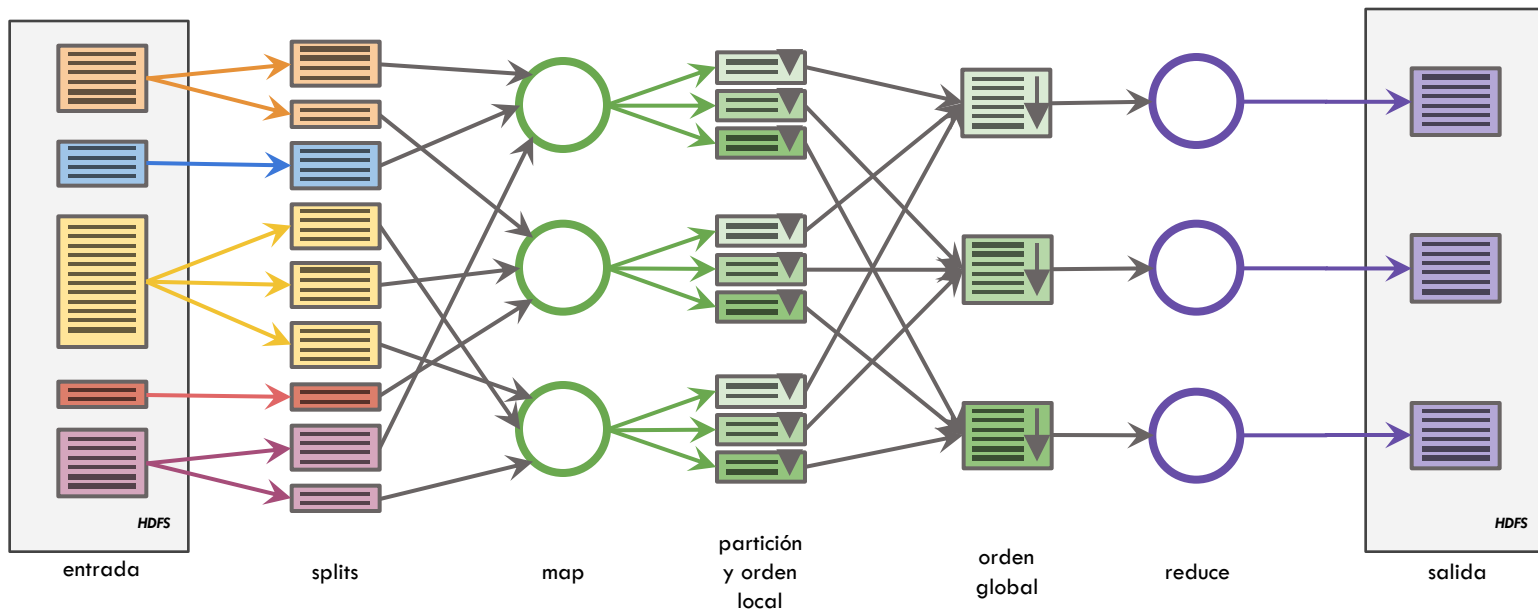
The screenshot shows the Hadoop JobHistory web interface in a browser. The address bar shows 'localhost:19888/jobhistory'. The page title is 'JobHistory'. On the left, there is a sidebar with 'Application' (containing 'About' and 'Jobs') and 'Tools'. The main content area is titled 'Retired Jobs' and shows a table with columns: Submit Time, Start Time, Finish Time, Job ID, Name, User, Queue, State, Maps Total, Maps Completed, Reduces Total, Reduces Completed, and Elapsed Time. The table is currently empty, displaying 'No data available in table'. A search bar is located at the top right of the table area. The bottom of the table shows 'Showing 0 to 0 of 0 entries' and navigation links: First, Previous, Next, Last.



- Opciones de ejecución (1 / 13)
 - Es posible ejecutar varias tareas de reducción en paralelo.
 - Recomendada para tareas de reducción con procesamiento intensivo.
 - Se obtienen varios ficheros de salida.
 - Se utiliza una función de **partición** para repartir las claves entre las diferentes tareas de reducción.
 - Por defecto, se aplica una función hash a las claves.

➤ Opciones de ejecución (2/13)

➤ Número de reducees (1/3)



- Opciones de ejecución (3/13)
 - Número de reducees (2/3)
 - Valor por defecto: 1
 - Función por defecto
 - Valor recomendado: $x * c$
 - $0,95 < x < 1,75$
 - c = máximo total de contenedores
 - Valor posible: 0
 - No se ejecuta función de reducción.
 - Se almacenan los resultados del map.

- Opciones de ejecución (4/13)
 - Número de reducees (3/3)
 - Cambiar por código

```
job.setNumReduceTasks(4)
```

- Fichero de configuración mapred-site.xml

```
<configuration>  
<property>  
  <name>mapreduce.job.reducees</name>  
  <value>4</value>  
</property>  
</configuration>
```



➤ Opciones de ejecución (5/13)

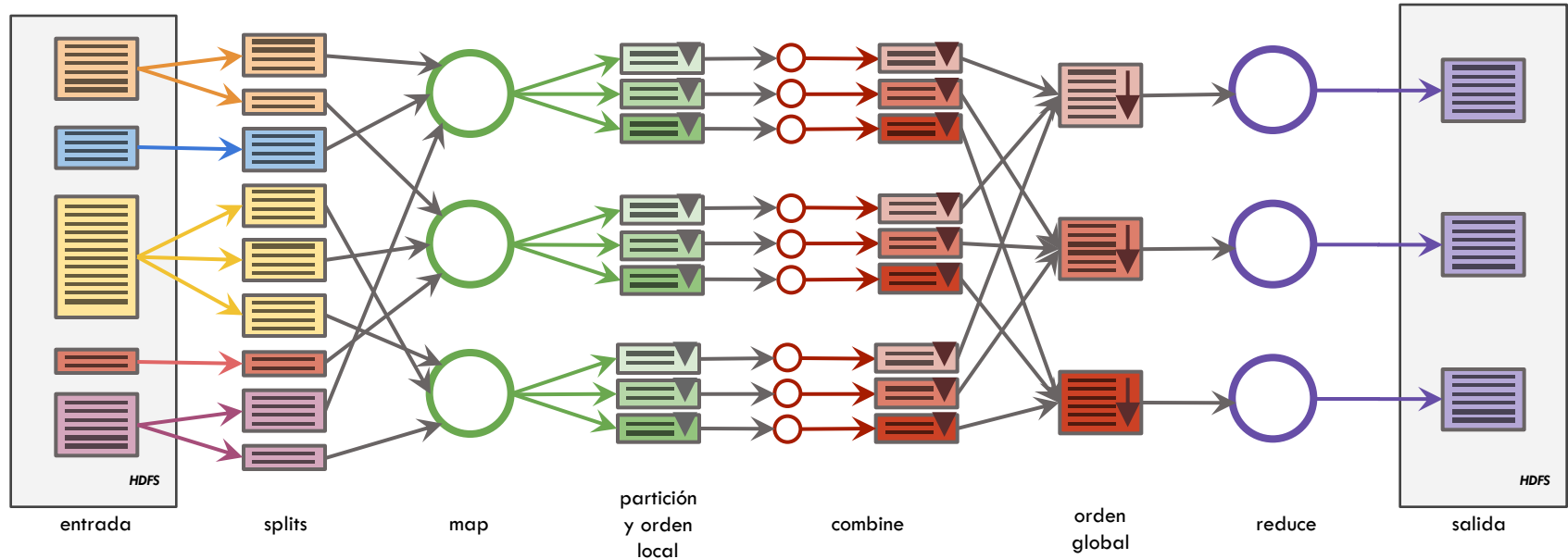
➤ Combiner (1/2)

- Es posible realizar una operación de reducción parcial después del map antes de enviar los resultados a la etapa de reducción.
- Permite reducir información transmitida entre los nodos.
- Los tipos de entrada y salida deben coincidir con los del reducer.

```
job.setCombinerClass(WordCountReducer.class);
```

➤ Opciones de ejecución (6/13)

➤ Combiner (2/2)



➤ Opciones de ejecución (7/13)

➤ Manejo de datos (1/3)

➤ Puede manejar diferentes formatos de entrada y salida

➤ Ficheros de texto

- 1 o más líneas como valor, clave-valor separados por tabulador u otro carácter, documentos XML, ...

➤ Ficheros binarios

➤ Múltiples ficheros, con diferente formato

- Permite trabajar con varios ficheros de E/S.

➤ Bases de datos relacionales

- **¡Las tareas podrían sobrecargar al servidor de origen!!**



- Opciones de ejecución (8/13)
 - Manejo de datos (2/3)
 - Puede utilizar diferentes herramientas para serialización de los datos en formato binario.
 - Writables
 - Puede manejar estructuras complejas mediante código Java.
 - Apache Avro
 - Permite interoperar con otros lenguajes.
 - Permite definir estructuras complejas en formato JSON.
 - Permite definir esquema al escribir y al leer los datos.



- Opciones de ejecución (9/13)
 - Manejo de datos (3/3)
 - Puede utilizar compresión
 - Reduce tamaño de datos para almacenar o transmitir.
 - Aplicable a datos de E/S y a la salida de la función mapeo.
 - Puede trabajar con ficheros agrupados.
 - Permite almacenar varios ficheros pequeños en un mismo bloque.



➤ Opciones de ejecución (10/13)

➤ Contadores

➤ Almacenan información para control y depuración de los procesos.

➤ Contadores de sistema

➤ Por tareas, por trabajo.

➤ Ejemplo: número de tareas de map ejecutadas.

➤ Contadores de usuario

➤ Definidos mediante código Java.

➤ Ejemplo: número de registros con formato inválido.



- Opciones de ejecución (11/13)
 - Hadoop incluye una librería de clases con funcionalidades comunes (1/3)
 - Se encuentran dentro del nombre de espacio `org.apache.hadoop.mapreduce.lib`.
 - Permiten minimizar o eliminar la necesidad de programar tareas frecuentes.
 - JOIN de datos antes de la fase de mapeo.



- Opciones de ejecución (12/13)
 - Hadoop incluye una librería de clases con funcionalidades comunes (2/3)
 - Análisis de una muestra de los datos:
 - Permite particionar los datos para la fase de reducción de forma que sea posible obtener datos ordenados concatenando los diferentes ficheros de salida.
 - Concatenador de mappers:
 - Ejecuta varios mappers en una sola tarea de mapeo o reducción.
 - Permite definir pasos reutilizables por ser más pequeños, pero evita sobrecarga del trasiego de datos.



- Opciones de ejecución (13/13)
 - Hadoop incluye una librería de clases con funcionalidades comunes (3/3)
 - Mappers o reducers comunes
 - Inversión clave \leftrightarrow valor.
 - Tokenizador de palabras.
 - Buscador de expresiones regulares.
 - Agregación de datos.



Ejercicio 1

- Modificar el programa WordCount:
 - Para que el número de Reduces sea 0.
 - Incrementando el número de Reduces a 3.



- Interfaz para múltiples lenguajes (1 / 5)
 - Hadoop provee APIs para escribir las funciones map y reduce en otros lenguajes
 - Hadoop Streaming
 - Permite programar rápidamente mediante scripts (python, perl, etc).
 - Hadoop Pipes
 - Permite programar eficientemente utilizando C++.



- Interfaz para múltiples lenguajes (2/5)
 - Hadoop Streaming: Mapper
 - Recibe datos de texto línea a línea de la entrada estándar.
 - Imprime tuplas de salida, separando clave y valor con TAB.

```
#!/usr/bin/env python3
import re, sys

for line in sys.stdin:
    words = line.split(' ')
    for word in words:
        word = word.strip()
        if word:
            print ("%s\t1" % word)
```

- Interfaz para múltiples lenguajes (3/5)
 - Hadoop Streaming: Reducer
 - Recibe una línea por cada clave-valor emitido.
 - Una clave puede estar en varias líneas, pero todas seguidas.

```
#!/usr/bin/env python3
import re, sys
last_word = None; total = 0
for line in sys.stdin:
    word, count = line.split('\t', 2)
    if last_word != word:
        if total:
            print ("%s\t%d" % (last_word, total))
        last_word = word
        total = 0
    total += int(count)
if total:
    print ("%s\t%d" % (last_word, total))
```



➤ Interfaz para múltiples lenguajes (4/5)

➤ Hadoop Streaming: Enviar trabajo

➤ Utilizar el JAR de Hadoop que incluye la API de Streaming

```
$ hadoop jar hadoop-streaming-2.6.0.jar  
    -files map.py, reduce.py  
    -mapper map.py -reducer reduce.py  
    -input /hadoop/ejemplo3/el_quijote.txt -output output/streaming
```

➤ Equivalente a ejecutar el siguiente script desde la línea de comandos:

```
$ cat /hadoop/ejemplo3/el_quijote.txt | streaming/map.py | sort -t$'\t' -k1,1 | streaming/reduce.py
```

¡¡ La ventaja es que Hadoop lo hace de forma distribuida!!



- Interfaz para múltiples lenguajes (5/5)
 - Hadoop Pipes
 - Se implementan clases de C++ que heredan de las clases Mapper y Reducer que provee Hadoop Pipes.
 - Los métodos map y reduce reciben un objeto contexto.
 - Permite leer entrada y escribir salida.
 - Más parecido a Hadoop MapReduce.
 - Utiliza sockets internamente para la comunicación entre Java y C++
 - Gestión de procesos más eficiente que Hadoop Streaming.



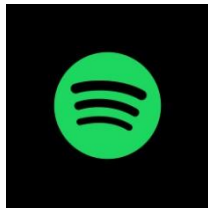
Demostración

MapReduce con Hadoop Streaming

Hadoop



➤ ¿Quién usa Hadoop?





- Reflexiones sobre Hadoop (1 / 9)
 - Viabilidad (1 / 3)
 - ¿Es mi proyecto un caso de Big Data/Hadoop? (1 / 2)
 - De forma genérica se han identificado casos de uso que se sitúan en los siguientes apartados:
 - Big Data Exploration. Adopción de la tecnología para encontrar, visualizar y comprender todos los datos para mejorar la toma de decisiones.
 - Visión 360° del cliente. Ampliar el conocimiento del cliente incorporando fuentes de información internas y externas.
 - Mejoras en seguridad. Disminuir el riesgo, detectar fraude y monitorizar seguridad en tiempo real.



- Reflexiones sobre Hadoop (2/9)
 - Viabilidad (2/3)
 - ¿Es mi proyecto un caso de Big Data/Hadoop? (2/2)
 - De forma genérica se han identificado casos de uso que se sitúan en los siguientes apartados:
 - Análisis operacional. Analizar una amplia variedad de datos operacionales para mejorar los resultados de negocio.
 - Ampliación del Data Warehouse. Integrar Big Data y Data Warehouse para aumentar la eficiencia operacional.



➤ Reflexiones sobre Hadoop (3/9)

➤ Viabilidad (3/3)

- ¿Cuándo HDFS/Mapreduce no parece ser una solución?
 - Si la tarea consiste en procesar trabajos de forma transaccional (con datos aleatorios).
 - Cuando no pueda paralelizarse el trabajo.
 - Cuando se requiera baja latencia de acceso a datos (un acceso muy rápido, sin retrasos).
 - Cuando haya que procesar muchos ficheros muy pequeños.
 - Cuando haya que hacer muchos cálculos con muy pocos datos.



- Reflexiones sobre Hadoop (4/9)
 - Aspectos a tener en cuenta (1 / 4)
 - Fuentes de información
 - Clásicas: BBDD, ficheros, correo, servidores web (intranet).
 - Nuevas: sensores, internet, RRSS, medios audiovisuales.
 - Formato de la información
 - Fijo/Conocido: BBDD (tablas), XML (formato conocido), JSON.
 - Variable: streaming, sensores.



- Reflexiones sobre Hadoop (5/9)
 - Aspectos a tener en cuenta (2/4)
 - Volumen
 - Información en origen (raw, csv).
 - Información en transición.
 - Información en explotación.
 - Volatilidad/Persistencia, Históricos/Crecimiento
 - Variación de los datos en origen.
 - Ciclos de captura de información.



- Reflexiones sobre Hadoop (6/9)
 - Aspectos a tener en cuenta (3/4)
 - Preparación de la información
 - Análisis del contenido de los ficheros (creación de metadatos).
 - Cambio del formato en la información.
 - Consolidación, si es necesaria, de la información recibida.
 - Análisis de los datos
 - Procesos de ejecución de análisis.
 - Cualificación del proceso de análisis (estadístico, detección de patrones, predicción, planificación, etc).



- Reflexiones sobre Hadoop (7/9)
 - Aspectos a tener en cuenta (4/4)
 - Explotación de los resultados
 - Destino de los resultados, almacenamiento, formato, etc.
 - Explotación de los mismos (visualización, decisión en tiempo real, etc.).



- Reflexiones sobre Hadoop (8/9)
 - Diseño de la solución (1/2)
 - Infraestructura
 - Sistemas, red y almacenamiento disponibles en el cliente.
 - Considerar nuevas adquisiciones.
 - Distribución de Hadoop
 - Seleccionar distribución: Apache (gratuita) o de un proveedor.



- Reflexiones sobre Hadoop (9/9)
 - Diseño de la solución (2/2)
 - Componentes adicionales fuera del framework
 - BBDD adicionales.
 - Mensajería.
 - Componentes de integración de mensajes.
 - Integración con componentes adicionales (servidores de aplicaciones, visualización, consola de mandos, etc.).



Ejercicio 2

- Modificar el programa WordCount para que cuente el número de palabras que tienen los nombres de películas del fichero de datos llamado movies.csv.